AVR32SD20/28/32 Preliminary Data Sheet

AVR32SD20/28/32

www.microchip.com Product Pages: AVR32SD20, AVR32SD28, AVR32SD32

Introduction

The AVR® SD Family microcontrollers, designed in compliance with the ISO 26262 functional safety standard, incorporate the AVR® CPU with a hardware multiplier running at clock speeds up to 20 MHz. They offer 32/64 KB of Flash, 4/8 KB of SRAM, and 256 bytes of EEPROM. The microcontrollers are available in 20-, 28-, 32- and 48-pin packages. The family uses our latest technologies, with a flexible and low-power architecture, including an Event System, accurate analog features, and advanced digital peripherals. The AVR® SD provides a dual-core lockstep CPU, Single-Error Correcting and Double-Error Detecting (SECDED) ECC on Flash, EEPROM and SRAM, Error Controller for functional safety, and Program and Debug Interface Disable (PDID) for security.

Features

- AVR CPU in Dual-Core Lockstep (DCLS) Configuration
  - Running at up to 20 MHz
  - Single-cycle I/O access
  - Two-level interrupt controller
  - Two-cycle hardware multiplier
  - Supply voltage range: 2.7-5.5V
- Memories
  - 32 KB in-system-programmable Flash memory with ECC
  - 4 KB SRAM with ECC
  - 256B EEPROM with ECC
  - 512B of user row in Non-Volatile Memory (NVM) that can keep data during chip-erase and be programmed while the device is locked
  - 256B of Boot Row for cryptographic keys, only readable from the Boot Section
- System
  - Power-on Reset (POR)
  - Brown-out Detector (BOD) with programmable levels
  - Voltage Regulator Monitor (VMON)
  - Clock options
    - High-precision internal oscillator with selectable frequency up to 20 MHz
    - PLL up to 48 MHz for high-frequency operation of the TCD
    - Internal 32.768 kHz oscillator
    - External 32.768 kHz crystal oscillator
    - External clock input
    - High-frequency external crystal oscillator

  - Clock Failure Detection (CFD)
  - Clock Frequency Measurement (CFM)
  - Single-pin Unified Program and Debug Interface (UPDI)
  - Three sleep modes
    - Idle with all peripherals running for immediate wake-up
    - Standby with a configurable operation of selected peripherals
    - Power-Down with full data retention
  - Automated Cyclic Redundancy Check (CRCSCAN) program memory scan
    - Verification of the Boot Flash section
    - CRC-16-CCITT or CRC-32 (IEEE 802.3)
  - External interrupt on all general-purpose pins
- Peripherals
  - 6-channel Event System for predictable and CPU-independent inter-peripheral signaling
  - One 16-bit Timer/Counter type A (TCA) with three compare channels for PWM and waveform generation
  - Up to four 16-bit Timer/Counter type B (TCB) with input capture for capture and signal measurements
  - One 12-bit Timer/Counter type D (TCD) optimized for power control
  - One 16-bit Real-Time Counter (RTC) that can run from an external crystal or internal oscillator
  - Up to three USARTs
    - Operation modes: RS-485, LIN client, host SPI, and IrDA
    - Fractional baud rate generator, auto-baud, and start-of-frame detection
  - Two SPI with host/client operation modes
  - Two I^2C with simultaneous host/client operation (dual mode) and dual address match
  - One Configurable Custom Logic (CCL) with up to six programmable Lookup Tables (LUTs)
  - Two 10-bit, 170 ksps, Analog-to-Digital Converters (ADC) with independent voltage reference sources
  - One 10-bit Digital-to-Analog Converter (DAC)
  - Three Analog Comparators (AC)
  - Up to two Zero Cross Detectors (ZCD)
  - Internal 1.024V, 2.048V, 4.096V and 2.500V voltage references, and external reference option
- System Integrity Functions
  - Error Controller (ERRCTRL)
    - Central interface for fault detection
    - Fault handling in hardware according to programmable severity
    - Optional Heartbeat output
    - Optional tri-stating of all I/O pins in case of fault
  - Parity on data buses
  - Dual Watchdogs
    - Synchronous Watchdog Timer (SWDT)
    - Watchdog Timer (WDT) with window mode and separate on-chip oscillator with clock failure detection
  - Voltage Level Monitor (VLM) with interrupt
  - Program and Debug Interface Disable (PDID)
- I/O and Packages

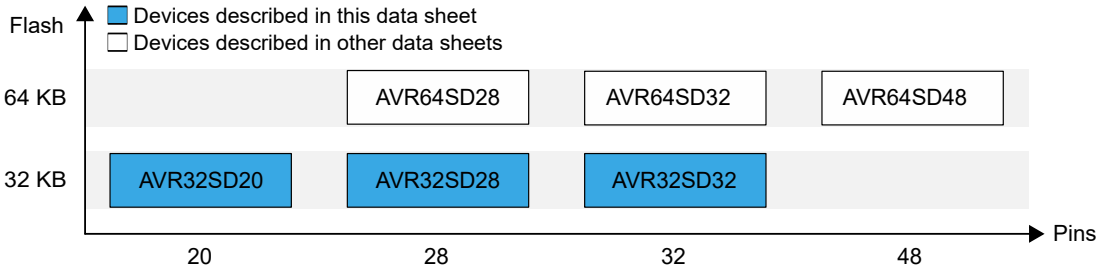
  - Up to 25 programmable I/O pins
  - Multi-voltage I/O with built-in voltage level converters
  - 20-pin SSOP
  - 28-pin VQFN, SSOP, and SPDIP
  - 32-pin VQFN and TQFP
- Temperature Ranges
  - Industrial: -40°C to +85°C
  - Extended: -40°C to +125°C

AVR® SD Family Overview

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

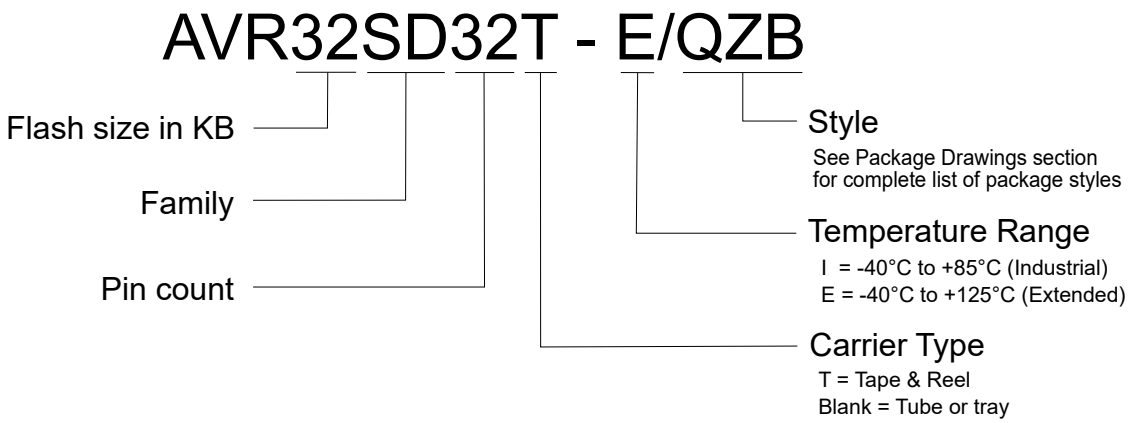
- Vertical migration is possible without code modification, as these devices are fully pin- and feature-compatible
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1. AVR SD Family Overview



A device name in the AVR SD Family is decoded as follows:

Figure 2. AVR® SD Device Designations



Memory Overview

The following table shows the memory overview of the entire AVR SD family.

Table 1. Memory Overview

Devices	AVR32SD20 AVR32SD28 AVR32SD32	AVR64SD28 AVR64SD32 AVR64SD48
Flash memory with ECC	32 KB	64 KB
SRAM with ECC	4 KB	8 KB
EEPROM	256B	256B
User row	512B	512B
Boot row/cryptographic key storage	256B	256B

Peripheral Overview

The following table shows the peripheral overview of the entire AVR SD family.

Table 2. Peripheral Overview

Feature	AVR32SD20	AVR32SD28 AVR64SD28	AVR32SD32 AVR64SD32	AVR64SD48
Pins	20	28	32	48
Max. frequency (MHz)	20	20	20	20
Clock Controller (CLKCTRL) with Clock Failure Detection (CFD) and Clock Frequency Measurement (CFM)	1	1	1	1
Clock Failure Detection (CFD)	2	2	2	2
Clock Frequency Measurement (CFM)	2	2	2	2
16-bit Timer/Counter type A (TCA)	1	1	1	1
16-bit Timer/Counter type B (TCB)	4	4	4	4
12-bit Timer/Counter type D (TCD)	1	1	1	1
Real-Time Counter (RTC)	1	1	1	1
USART	2	3	3	3
SPI	2	2	2	2
TWI/I^2C[1]	1[1]	1[1]	2[1]	2[1]
10-bit ADC (channels)	2 (13)	2 (19)	2 (23)	2 (28)
Analog Comparator (AC)	3	3	3	3
Digital-to-Analog Converter (DAC)	1	1	1	1
Zero Cross Detector (ZCD)	1	2	2	2
Peripheral Touch Controller (PTC)	-	-	-	-
Operational amplifier (OP)	-	-	-	-
Configurable Custom Logic Look-up Table (CCL LUT)	6	6	6	6
Error Controller (ERRCTRL)	1	1	1	1
Synchronous Watchdog Timer (SWDT)	1	1	1	1
Watchdog Timer (WDT)	1	1	1	1
Event System (EVSYS) channels	6	6	6	6
General Purpose I/O	15	21	25	40
PORT	PA[7:0], PC[3:1], PD[7:4]	PA[7:0], PC[3:0], PD[7:1], PF[1,0]	PA[7:0], PC[3:0], PD[7:1], PF[5:0]	PA[7:0], PB[5:0], PC[7:0], PD[7:0], PE[3:0], PF[5:0]
PORT pins with MVIO capability	PC[3:1]	PC[3:0]	PC[3:0]	PC[7:0]
External interrupts	15	21	25	40
CRCSCAN	1	1	1	1
Unified Program and Debug Interface (UPDI)	1	1	1	1

Note:

1. The TWI/I^2C can operate simultaneously as host and client on different pins.

Table of Contents

1. Functional Safety Concept

The devices of the AVR SD Family are general purpose microcontrollers designed according to the ISO 26262 and IEC 61508 safety standards to avoid systematic device faults.

The device family aims to mitigate risks and prevent failures that could lead to hazardous situations, minimize the required development of complex and time-consuming software diagnostics, and reduce the fault detection time for faults in vital functions down to 1 ms.

The AVR SD devices feature autonomous hardware fault detection mechanisms to detect, correct, and report random hardware faults and transient faults in the CPU and the infrastructure it relies upon. This Core-Independent Safety (CIS) makes the devices of the AVR SD Family highly suitable for safety-critical applications with strict safety requirements and demands for fast fault handling. The AVR SD devices suit for safety-critical applications with ISO 26262 (ASIL C) and IEC 61508 (SIL 2) safety requirements.

The primary Core-Independent Safety (CIS) features for the AVR SD devices are:

- Error controller, able to autonomously set the device in a safe state upon detection of a critical fault
- Dual CPU core in lock step configuration with redundant comparators
- ECC protection of Flash, EEPROM and SRAM memories
- Parity protection of data bus, with redundant control signals
- CRC and ECC protection of device configuration and calibration fuses
- Clock frequency monitor and failure detection
- Over- and under-voltage detectors for the Voltage Regulator Monitor (VMON)
- Dual watchdogs to detect hardware and software faults (WDT and SWDT)
- Stack monitor to detect software faults
- Redundant operation of most peripherals through duplication
- Various other integrity check mechanisms

Functional safety documentation and software are available. These are complemented by a rich set of safety-qualified MPLAB® development tools.

1.1 Error Controller

The Error Controller's (ERRCTRL) role is to collect hardware error reports from the various modules in the MCU and present them to the application. The ERRCTRL is always enabled, and it cannot be disabled. The ERRCTRL serves as user interface or front-end for different components of the MCU, providing the capability to detect internal hardware errors. The application can associate various Severity Levels with each HW error. Each error source, such as Instruction ECC error or Stack Pointer Limit, is connected to an Error Channel. Whenever an MCU component detects a hardware error, it signals the ERRCTRL on the associated error channel, causing the error channel's Error Status Flag (ESF) to be set. The error channel's Error Source Control (ESC) register determines the resulting severity.

1.2 Dual CPU Core

The purpose of the Dual-Core LockStep CPU is to achieve high diagnostic coverage and meet strict timing requirements. In this configuration, two identical CPUs execute the same program in parallel, with their internal state being continuously compared by a DCLS comparator. Any mismatch between the CPUs is reported as an error and will issue a machine check reset.

Additionally, the CPU has built-in illegal OPCODE and illegal address detection.

1.3 ECC Protection

All memories on the device are protected by a SEC-DED ECC mechanism, with redundant ECC checkers for latent fault detection. The purpose of having ECC protection is to achieve a high diagnostic coverage without compromising memory sizes and execution timings. Any ECC error will signal the Error Controller, which will take appropriate action according to the ERRCTRL configuration.

Note: Due to the ECC protection, any program and debug interface must write the Flash in 16-bit words.

1.4 Data Bus Parity

The purpose of having parity protection of the data bus is to achieve high diagnostic coverage and increased confidence that the data transferred between bus initiators and bus targets is correct.

1.5 Clock Protection

The clock protection mechanisms are driven by a clock that is independent of the clock being monitored. If a supervised clock is outside the specified range, the Error Controller is notified. The purpose of the clock protection mechanisms is to achieve high coverage and increased confidence that the clocks are operating within specification.

1.6 Voltage Regulator Monitor

A voltage regulator monitor is present on the device to detect over- and under-voltage on the regulated domain. Any detected errors are reported to the Error Controller.

1.7 Watchdogs

Two watchdogs are present on the device, one asynchronous watchdog (WDT) and one synchronous watchdog (SWDT). The asynchronous watchdog is clocked by an independent clock source and will directly reset the device when an error is detected. The synchronous watchdog is clocked by the main clock and counts either clock cycles or CPU instructions. Any SWDT error will signal the Error Controller, which will take appropriate action according to the ERRCTRL configuration.

1.8 Stack Monitor

The purpose of the stack monitor is to detect if the program flow has run away, as in, performed operations that violate the intended memory boundaries. Any stack overflow or underflow errors are reported to the Error Controller.

1.9 Redundancy

Many peripherals and components of the device have more than one instance to provide the possibility of redundant operation, and as such, increase the possible diagnostic coverage of these elements.

1.10 Integrity Checks

Various other mechanisms has been implemented on the device to both detect faults and increase the testability of the device and it's features. These mechanisms include, but are not limited to:

- Fault injection on several modules
- Sleep entry protection
- Watchdog timer clock failure detection
- Separate voltage references for each ADC module
- On Chip Debugging (OCD) and Design For Test (DFT) disabled monitors

2. Security Concept

The AVR SD devices are general purpose microcontrollers that offer fundamental security features to implement secure firmware upgrades and authenticate the application firmware. When using the security features correctly, they protect against remote attacks and some PCB-level attacks where the application code is attempted modified to change the product functionality.

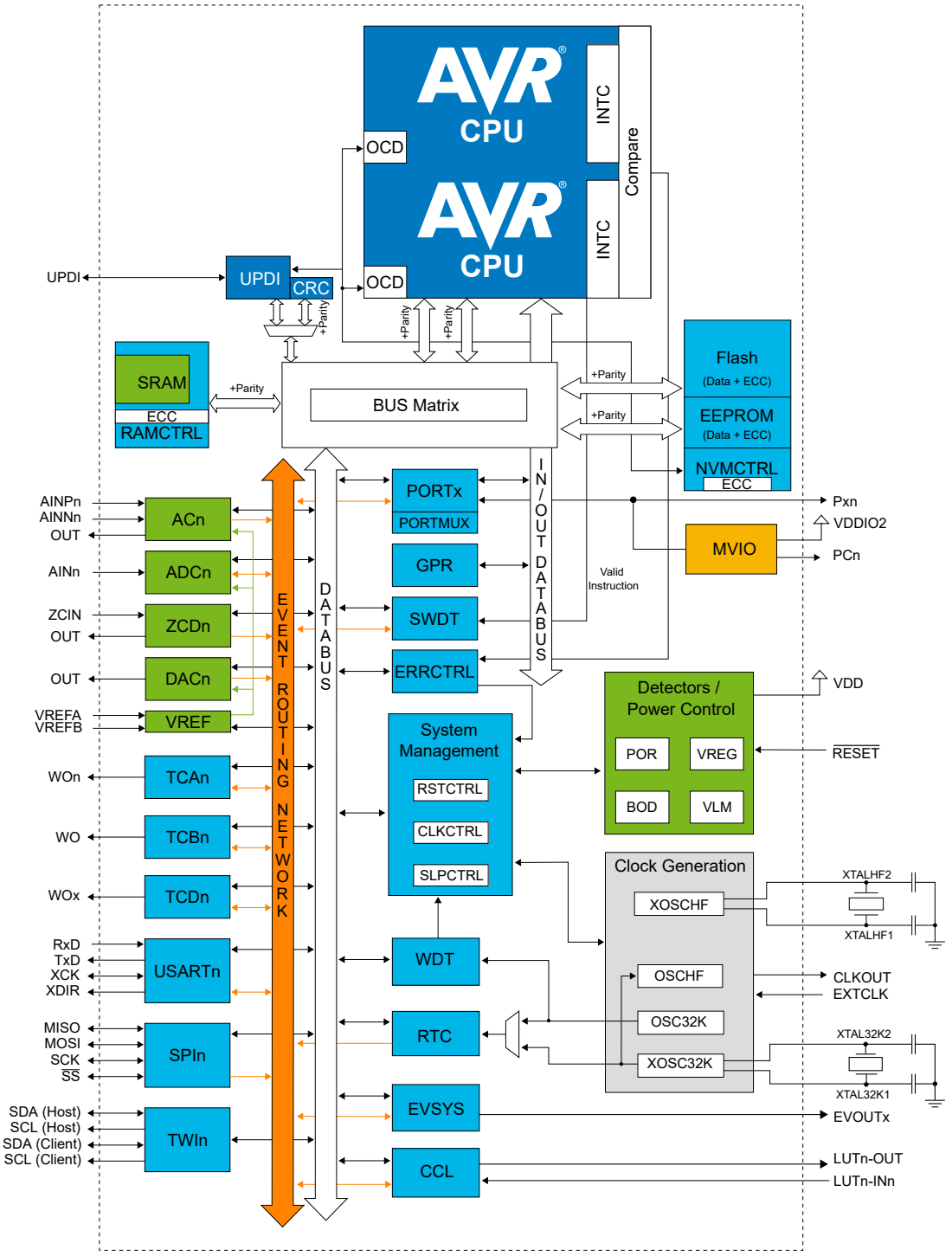
The cornerstone of the security features is the *Program and Debug Interface Disable (PDID)*, a mechanism preventing access to the device's reprogrammable Flash memory over the programming interface (UPDI). After the PDID is activated and the device is locked, the programming interface (UPDI) is prevented from making any changes to the device. The programming interface (UPDI) can still read out the device information and CRC status.

The only way to program the device after activating the PDID is by using software stored in the Boot Code section of the Flash to update the Application Code section software. This application-specific software must be able to receive new data and program the Application Code section. It is impossible to alter code stored in the Boot Code section using this mechanism, as the Boot Code section is only accessible using the programming interface (UPDI).

In addition, there is a separate storage space accessible only by code in the Boot Code section, which can hold any data that is intended to be accessible only from the Boot Code section. One example of this is a cryptographic key to be used to validate data that are sent to a bootloader to update the application software on the device.

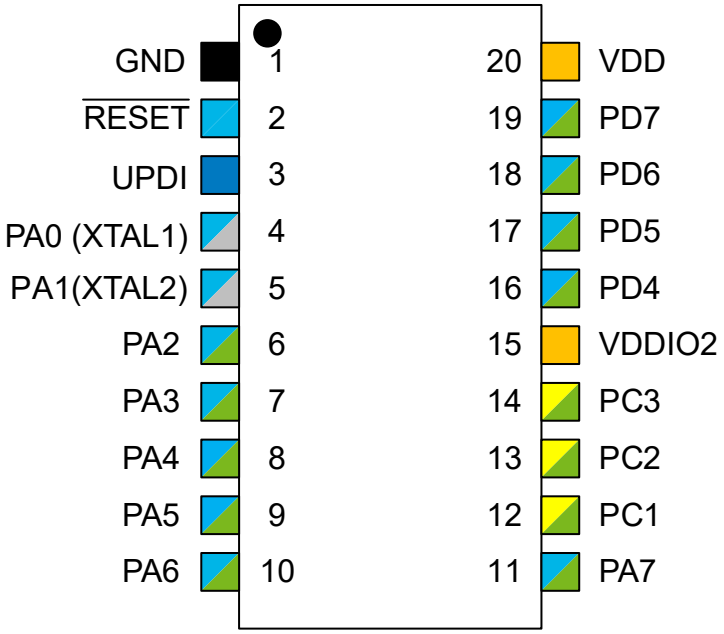
This creates a two-layer security: The device is prevented from being erased or reprogrammed over the programming interface (UPDI), and by that, the code in the Boot Code section is protected. Secondly, the code in the Boot Code section can use a cryptographic key (that is only accessible by code in this section of Flash) to verify that any new application code that is received for the device software update is authentic.

3. Block Diagram



4. Pinout

4.1 20-Pin SSOP



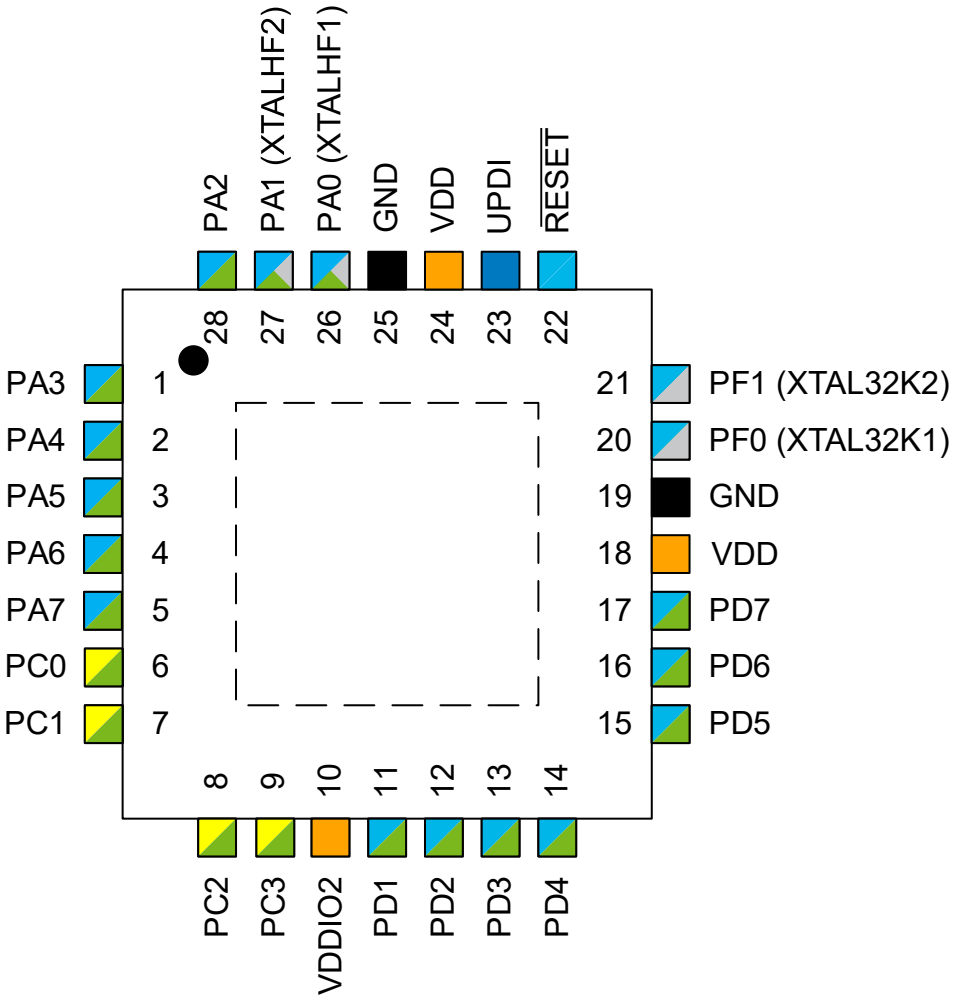
Power

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

Functionality

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

4.2 28-Pin VQFN



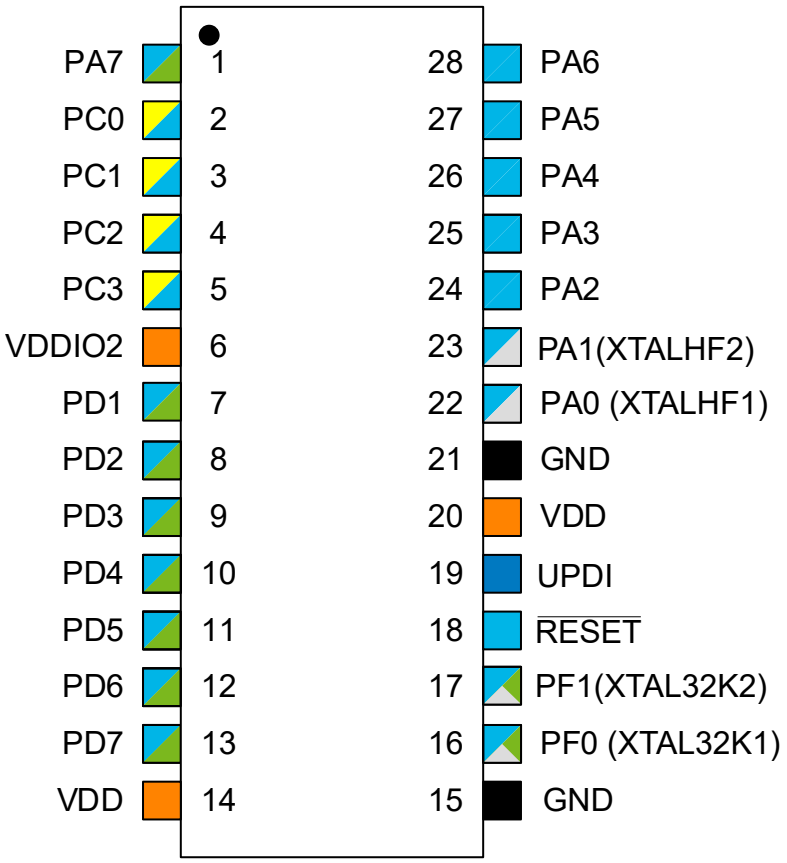
Power

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

Functionality

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

4.3 28-Pin SSOP and SPDIP



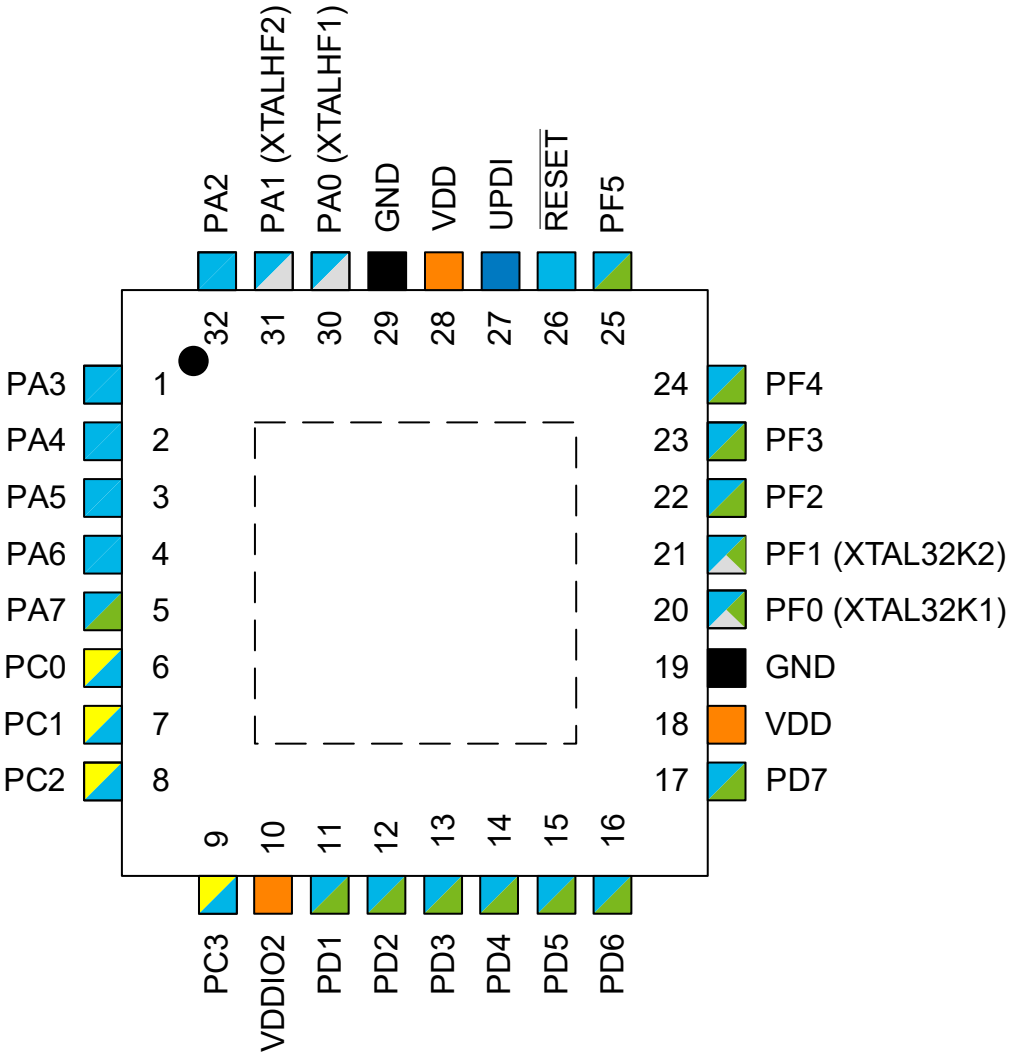
Power

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

Functionality

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

4.4 32-Pin VQFN and TQFP



Power

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

Functionality

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

5. I/O Multiplexing and Considerations

5.1 I/O Multiplexing

VQFN32/ TQFP32	SOIC28/SSOP28/ SPDIP28	VQFN28	SOIC20	Pin name[1,2]	Special	ADCn	ACn	DAC0	ZCDn	USARTn	SPIn	TWIn[4]	TCAn	TCBn	TCD0	EVSYS	CCL-LUTn	ERRCTRL
30	22	26	4	PA0	XTALHF1 EXTCLK					0, TxD	0, MOSI[3]	0, SDA(HC)[3]	0, WO0				0, IN0	
31	23	27	5	PA1	XTALHF2					0, RxD	0, MISO[3]	0, SCL(HC)[3]	0, WO1				0, IN1	
32	24	28	6	PA2		AIN22				0, XCK 0, TxD[3]		0, SDA(HC)	0, WO2	0, WO		EVOUTA	0, IN2	HEART
1	25	1	7	PA3		AIN23				0, XDIR 0, RxD[3]		0, SCL(HC)	0, WO3	1, WO			0, OUT	
2	26	2	8	PA4		AIN24				0, TxD[3]	0, MOSI		0, WO4		WOA			HEART[3]
3	27	3	9	PA5		AIN25				0, RxD[3]	0, MISO		0, WO5		WOB			
4	28	4	10	PA6		AIN26				0, XCK[3]	0, SCK				WOC		0, OUT[3]	HEART[3]
5	1	5	11	PA7	CLKOUT	AIN27	0, OUT 1, OUT 2, OUT		0, OUT[5] 3, OUT	0, XDIR[3]	0, $\overline{SS}$				WOD	EVOUTA[3]		HEART[3]
6	2	6	-	PC0		AIN28				1, TxD	1, MOSI 0, MOSI[3] 0, SCK[3]		0, WO0[3]	2, WO			1, IN0	
7	3	7	12	PC1		AIN29				1, RxD 0, TxD[3]	1, MISO 0, MISO[3] 0, $\overline{SS}$[3] 0, MOSI[3]		0, WO1[3]	3, WO[4]			1, IN1	
8	4	8	13	PC2		AIN30	0, AINN3 1, AINN3		3, ZCIN	1, XCK 0, RxD[3]	1, SCK 0, SCK[3] 0, MISO[3]	0, SDA(C) 0, SDA(HC)[3]	0, WO2[3]			EVOUTC	1, IN2	
9	5	9	14	PC3	TWI Fm+	AIN31	0, AINP4 1, AINP4			1, XDIR 0, XCK[3]	1, $\overline{SS}$ 0, $\overline{SS}$[3] 0, SCK[3]	0, SCL(C) 0, SCL(HC)[3]	0, WO3[3]				1, OUT	
10	6	10	15	VDDIO2														
-	-	-	-	GND														
11	7	11	-	PD1		AIN1			0, ZCIN				0, WO1[3]				2, IN1	
12	8	12	-	PD2		AIN2	0, AINP0 1, AINP0 2, AINP0						0, WO2[3]			EVOUTD	2, IN2	
13	9	13	-	PD3		AIN3	0, AINN0 1, AINP1						0, WO3[3]				2, OUT	
14	10	14	16	PD4	VREFB	AIN4	1, AINP2 2, AINP1			0, TxD[3]	0, MOSI[3]		0, WO4[3]		WOC[3]			HEART[3]
15	11	15	17	PD5		AIN5	1, AINN0			0, RxD[3]	0, MISO[3]		0, WO5[3]		WOD[3]			
16	12	16	18	PD6		AIN6	0, AINP3 1, AINP3 2, AINP3	OUT		0, XCK[3] 1, TxD[3]	0, SCK[3]						2, OUT[3]	
17	13	17	19	PD7	VREFA	AIN7	0, AINN2 1, AINN2 2, AINN0/ AINN2			0, XDIR[3] 1, RxD[3]	0, $\overline{SS}$[3]					EVOUTD[3]		
18	14	18	20	VDD														
19	15	19	1	GND														
20	16	20	-	PF0	XTAL32K1	AIN16				2, TxD			0, WO0[3]		WOA[3]		3, IN0	
21	17	21	-	PF1	XTAL32K2	AIN17				2, RxD			0, WO1[3]		WOB[3]		3, IN1	
22	-	-	-	PF2		AIN18				2, XCK		1, SDA(HC)	0, WO2[3]		WOC[3]	EVOUTF	3, IN2	
23	-	-	-	PF3		AIN19				2, XDIR		1, SCL(HC)	0, WO3[3]		WOD[3]		3, OUT	
24	-	-	-	PF4		AIN20				2, TxD[3]			0, WO4[3]	0, WO[3]				
25	-	-	-	PF5		AIN21				2, RxD[3]			0, WO5[3]	1, WO[3]				HEART[3]
26	18	22	2	$\overline{RESET}$	$\overline{RESET}$													
27	19	23	3	UPDI	UPDI													
28	20	24	-	VDD														
29	21	25	-	GND														

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. To select alternative pin positions, refer to the *Port Multiplexer* section.
4. TWI pins are marked *HC* if they can be used as TWI Host or Client pins and *C* if they can only be used as TWI Client pins.
5. Devices with 20 pins provide ZCD3 only.

6. 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.

6.1 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 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*

6.1.1 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 mechanically bonded to the internal substrate. It serves as a thermal heat sink and provides 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).

6.2 Connection for Power Supply

The basics and details of power supply design lie beyond the scope of these guidelines. See the application notes mentioned at the beginning of this section for more detailed information about this subject.

A decoupling capacitor must be placed close to the microcontroller for each supply pin pair (VDD 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 to keep the distance to the supply pin as short as possible.

If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor 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, ensuring 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.

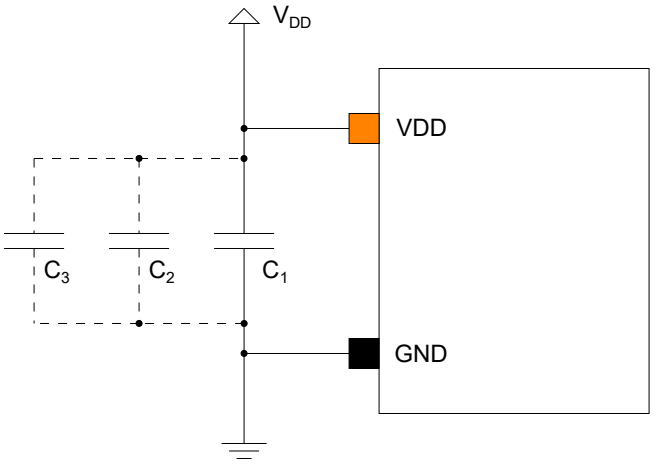
As mentioned at the beginning of this section, all values used in examples are typical values. The actual design may require other values.

6.2.1 Power Supply

For higher 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 figure below shows the recommended connection of the power supply to the device's VDD pin(s).

Figure 6-1. Recommended VDD Connection Circuit Schematic



Typical values (recommended):
C_1: 100 nF (primary decoupling capacitor)
C_2: 1-10 nF (HF decoupling capacitor)
C_3(*): 1 µF (decoupling capacitor - optional)

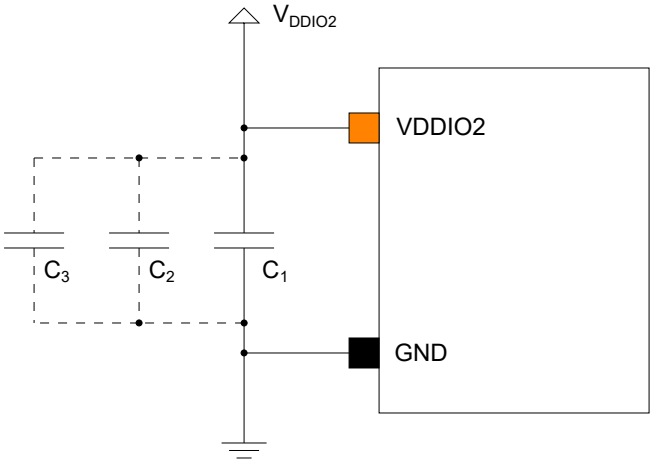
Important: For systems that frequently cycle V_{DD} or experience fast V_{DD} transients, it is recommended to add a decoupling capacitor (C_3) if the power supply slew rate exceeds the slew rate limits. Refer to the *Supply Voltage* section in the *Electrical Characteristics* for details about the power supply's slew rate limits.

6.2.2 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: Connect a separate decoupling capacitor to the pin pair, and keep the trace distance from the pins as short as possible. Each supply pin and its corresponding ground pin must have a decoupling capacitor if more than one MVIO power supply pin.

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

Figure 6-2. Recommended VDDIO2 Connection Circuit Schematic



Typical values (recommended):
C_1: 100 nF (primary decoupling capacitor)
C_2: 1-10 nF (HF decoupling capacitor)
C_3(*): 1 µF (decoupling capacitor - optional)

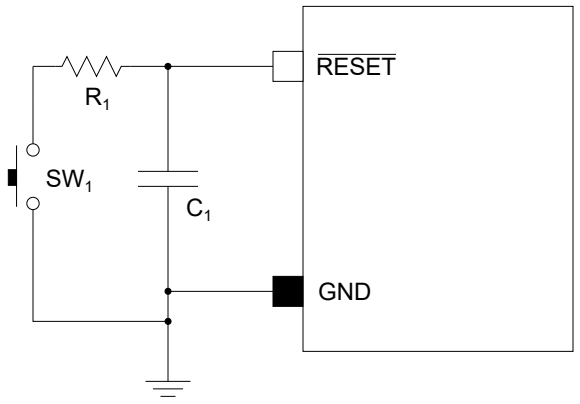
Important: For systems that frequently cycle V_{DDIO2} or experience fast V_{DDIO2} transients, it is recommended to add a decoupling capacitor (C_3) if the power supply slew rate exceeds the slew rate limits. Refer to the *Supply Voltage* section in the *Electrical Characteristics* for details about the power supply's slew rate limits.

6.3 Connection for $\overline{\text{RESET}}$

The $\overline{\text{RESET}}$ pin on the device is active-low with an internal pull-up resistor, and externally pulling the pin low will result in a device Reset. 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 6-3. Recommended External Reset Circuit Schematic



Typical values (recommended):
C_1: 100 nF (filtering capacitor)
R_1: 330Ω (switch series resistance)

Shorting the filtering capacitor may cause a noise spike that can harm the system. To prevent this, a resistor in series with the switch can safely discharge the filtering capacitor preventing a current surge.

6.4 Connection for UPDI Programming

The Unified Program and Debugging Interface (UPDI) connection provides a one-wire interface for external programming and on-chip debugging (OCD). This section is related to the physical connection itself and not the details of the signal protocol and features of the UPDI peripheral. These details are described in the *UPDI* section.

The recommended UPDI connection has changed since its first introduction. For this reason, both connections are described below, with the initial UPDI connection layout named **UPDI Connection v1** while the new UPDI connection layout is named **UPDI Connection v2**. The difference between the two connections is the inclusion of a $\overline{\text{RESET}}$ signal in the connection for v2.

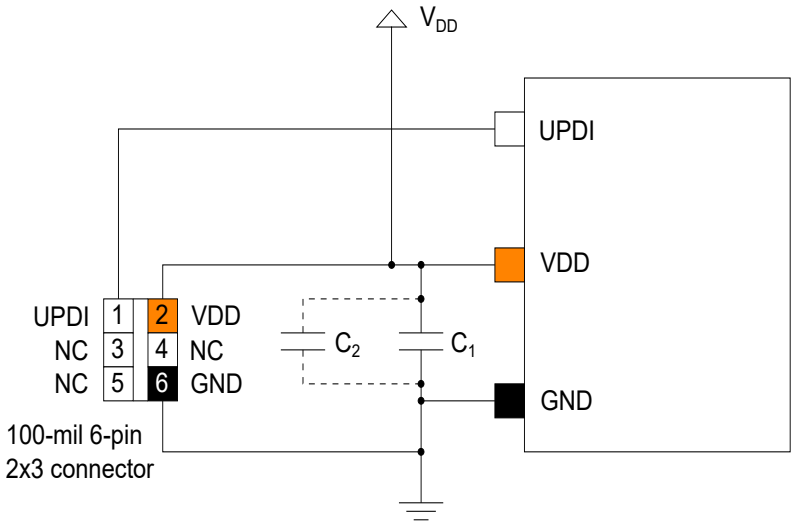
6.4.1 UPDI Connection v1

This was the initial layout for the UPDI connection used by older programming tools (like the Atmel ICE).

The **UPDI Connection v1** is a 100-mil 6-pin 2x3 header. Even though using only three pins for programming, it is recommended to use a 2x3 header since most programming tools using this connection are delivered with 100-mil 6-pin 2x3 connectors.

The following figure shows the recommendation for a UPDI connection to the device using the **UPDI Connection v1**.

Figure 6-4. Recommended UPDI Programming Circuit Schematic



Typical values (recommended):
C_1: 100 nF (primary decoupling capacitor)
C_2: 1-10 nF (HF decoupling capacitor)
NC = Not Connected

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

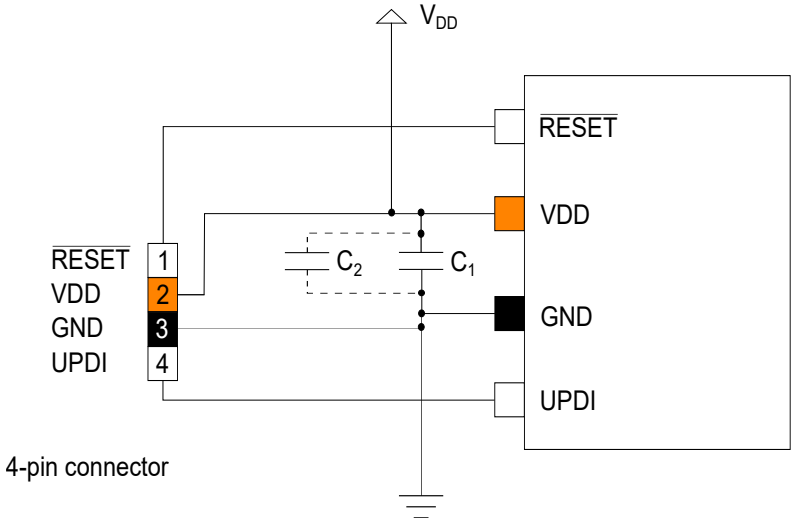
6.4.2 UPDI Connection v2

This connection is compatible with any AVR device but requires an adapter cable for users with older programmers/debuggers like the *Atmel-ICE* and the *Atmel PowerDebugger* with the 100-mil 2x3 header connector. This connection is directly compatible with the programming tool *PICkit™ 4 In-Circuit Debugger*.

The *UPDI Connection v2* is a 100-mil 4-pin 1x4 header. Even though three pins are sufficient for programming many AVR devices, it is recommended to use a single row 100-mil 4-pin header, allowing for the $\overline{\text{RESET}}$ signal to be included. This connector is also compatible with the *PICkit 4* programmer.

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

Figure 6-5. Recommended UPDI Programming Circuit Schematic



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

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

Enabling UPDI using $\overline{\text{RESET}}$

By design or mistake it may be possible to disable UPDI by writing to the appropriate fuse. For details on disabling UPDI, see the *FUSE* sub-section of the *Memories* section. Note that for devices with dedicated UPDI pin, there is no fuse to disable UPDI.

A high-voltage pulse must be applied to the $\overline{\text{RESET}}$ pin to re-enable the UPDI. See the *UPDI* section for details on how to apply the high-voltage pulse to the $\overline{\text{RESET}}$ pin.

Take additional care in the design of the circuit if the $\overline{\text{RESET}}$ pin is connected to other components. If the high-voltage pulse is applied to the $\overline{\text{RESET}}$ pin, other components connected to the line might be damaged. In this case, the design must allow disconnection of these components from the circuit before the high-voltage pulse is applied. One example of this may be a removable jumper.

Note: On devices that feature *Program and Debug Interface Disable (PDID)*, the UPDI cannot be re-enabled using the $\overline{\text{RESET}}$ pin after the PDID feature has been activated.

6.5 Connecting External Crystal Oscillators

The use of external oscillators and the design of oscillator circuits are not trivial because of many variables: V_{DD}, operating temperature range, crystal type and manufacture, loading capacitors, circuit layout, and PCB material. Some typical guidelines to help with the basic oscillator circuit design are presented in this section.

- 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. Mount the load capacitors on the same side of the board and next to the crystal. Do not use sockets.
- 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 be connected to the bottom layer area by 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, see the following application notes:

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

6.5.1 Connection for XTAL32K (External 32.768 kHz Crystal Oscillator)

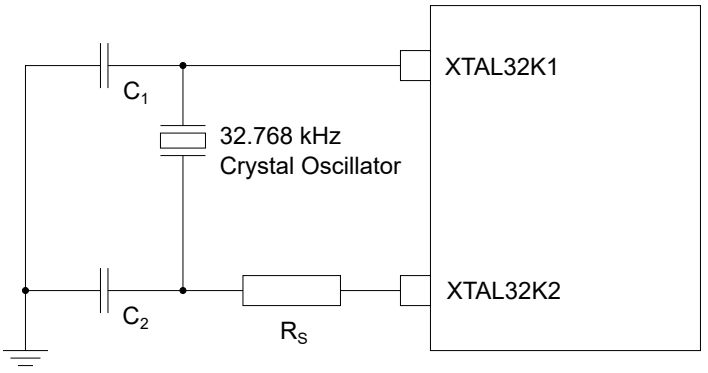
Ultra-low power 32.768 kHz oscillators typically dissipate significantly below 1 µW, and the current flowing in the circuit is, therefore, extremely small. The crystal frequency is highly dependent on the capacitive load.

A series resistor R_S may be required to prevent overdriving the oscillator. The gain from the oscillator driver may sometimes be too high for low-frequency oscillators, and adding impedance with R_S can decrease the gain. The overdrive causes the oscillator to not swing properly, as the

signal will be saturated (clipped or “squashed”). Overdriving the crystal can also lead to the circuit jumping to a higher harmonic.

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

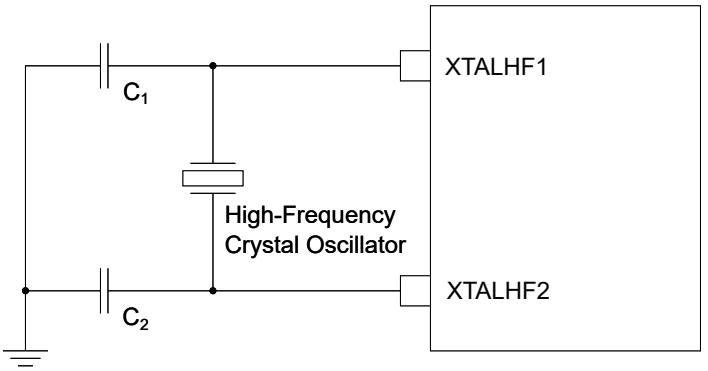
Figure 6-6. Recommended External 32.768 kHz Oscillator Connection Circuit Schematic



6.5.2 Connection for XTALHF (External HF Crystal Oscillator)

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

Figure 6-7. Recommended External High-Frequency Oscillator Connection Circuit Schematic

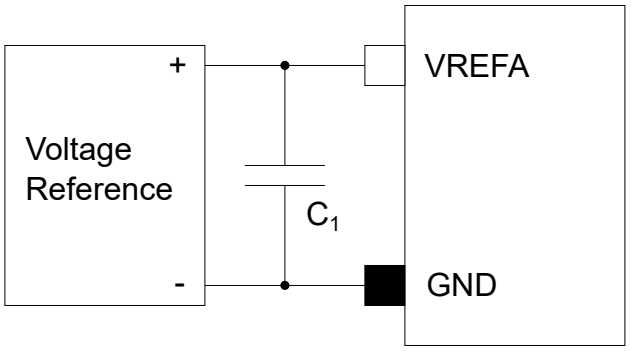


6.6 Connection for External Voltage Reference

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

Additional filtering components may be necessary depending on the type of external voltage reference used.

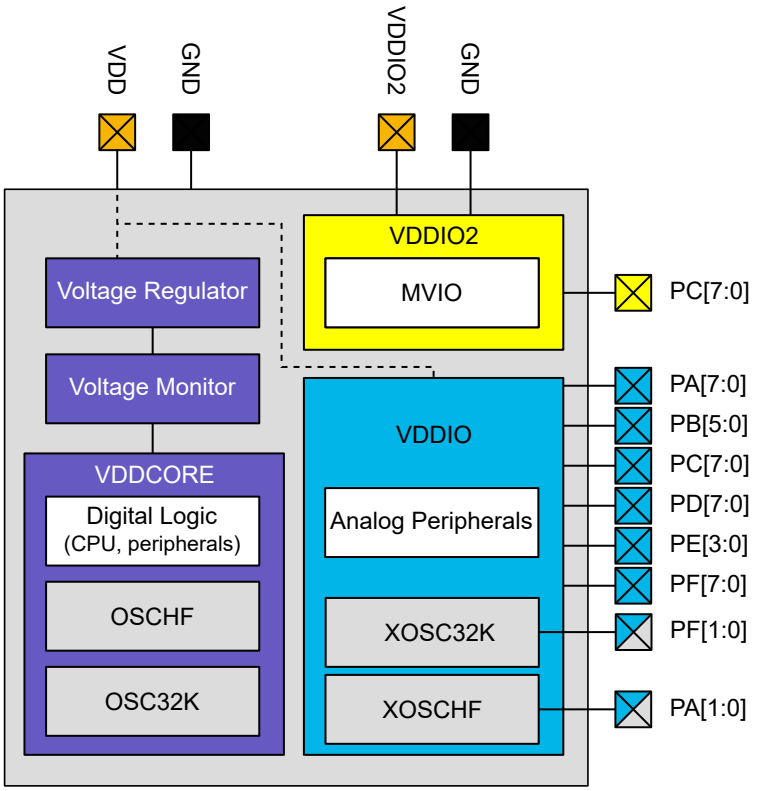
Figure 6-8. Recommended External Voltage Reference Connection



7. Power Supply

7.1 Power Domains

Figure 7-1. Power Domain Overview



The AVR SD Family of devices has several power domains with the following power supply pins:

VDD	Powers I/O lines and the internal voltage regulator
VDDIO2	Powers I/O lines, optionally at a different voltage from V_{DD}

The ground pins, GND, are common to VDD 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 *MVIO - Multi-Voltage I/O* section for further information.

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

7.2 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 *SLPCTRL - Sleep Controller* section for further information.

7.3 Power-Up

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 *MVIO - Multi-Voltage I/O* section for further information.

The Power-On Reset (POR) and the Brown-out Detector (BOD) monitor V_{DD} and will keep the system in Reset if the voltage level is below the respective voltage thresholds. Refer to the *RSTCTRL - Reset Controller* and *BOD - Brown-out Detector* sections for further information. The Voltage Monitor (VMON) will keep the device in Reset if the core voltage VDDCORE is outside the specifications.

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

8. Conventions

8.1 Numerical Notation

Table 8-1. 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

8.2 Memory Size and Type

Table 8-2. 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

8.3 Frequency and Time

Table 8-3. 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

8.4 Registers and Bits

Table 8-4. 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	Bit field names are shown in uppercase. Example: INTMODE.

.....continued

Symbol	Description
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. 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. 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. Writing a '1' to a bit in the CLR register will clear the corresponding bit in both registers. Writing a '1' to a bit in the SET register will set the corresponding bit in both registers. Writing a '1' to a bit in the TGL register will toggle the corresponding bit in both registers.

8.4.1 Addressing Registers from Header Files

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 8-1. 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:

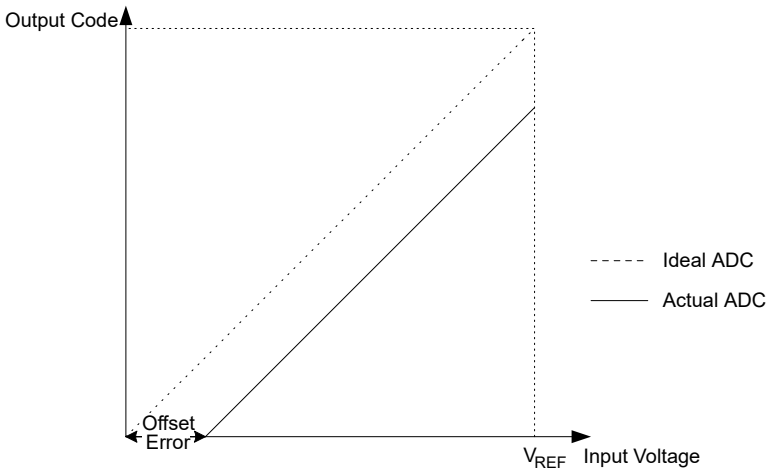
```
// TCA0 in Normal Mode (SINGLE) uses waveform generator in frequency mode
TCA0.SINGLE.CTRL=TCA_SINGLE_WGMODE_FRQ_gc;
```

8.5 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:

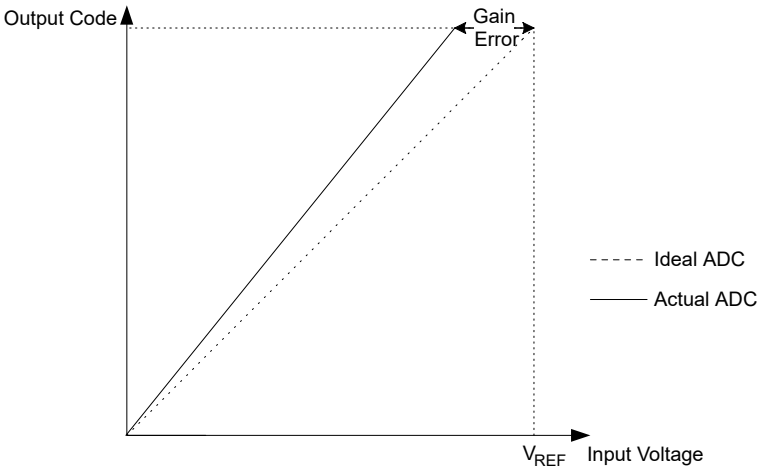
Offset Error The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSb). Ideal value: 0 LSb.

Figure 8-1. Offset Error



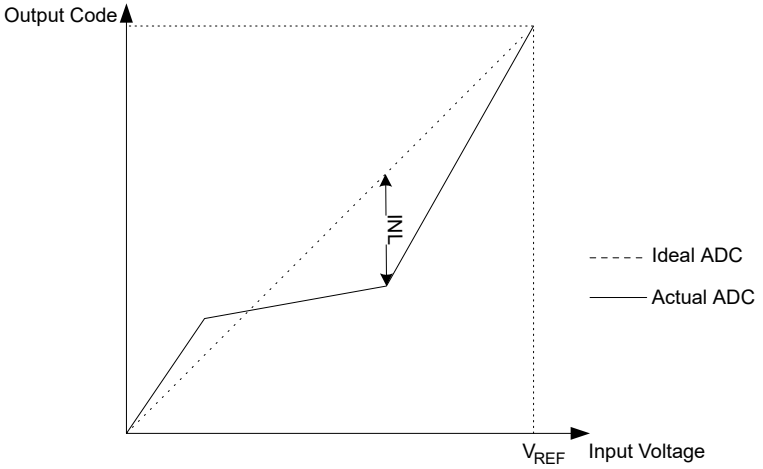
Gain Error After adjusting for offset, the gain error is found as the deviation of the last transition (e.g., 0x3FE to 0x3FF for a 10-bit ADC) compared to the ideal transition (at 1.5 LSb below maximum). Ideal value: 0 LSb.

Figure 8-2. Gain Error



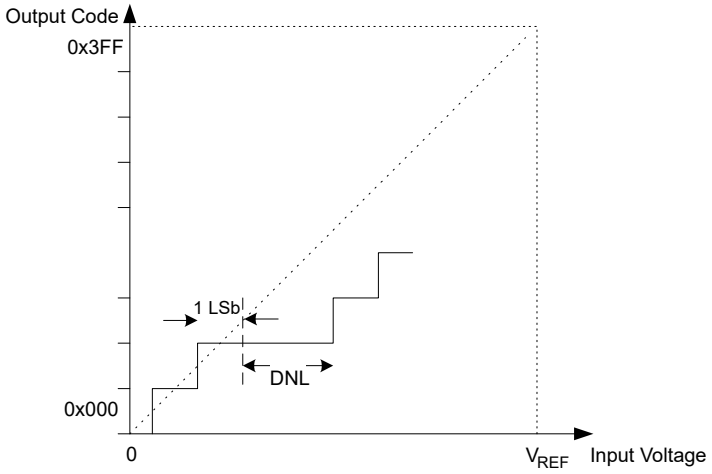
Integral Nonlinearity (INL) After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSb.

Figure 8-3. 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 8-4. 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.

9. BOOTROM - Boot ROM Code

9.1 Overview

The Boot ROM Code controls the start-up sequence of the device.

9.2 Features

- Reset the device and set it to an initial state
  - Control power-up sequence
  - Load calibration fuses
- Execution of actions required by the UPDI
- Trigger CRC scan of Boot ROM and Flash

9.3 Functional Description

The Boot ROM Code executes after every reset before control is handed to the application.

It performs a self-test including a VREG Monitor check that halts booting until the voltage range has been deemed valid, a CRC scan of the Boot ROM contents combined with various other integrity check mechanisms. Any error detected in the self-test or during the execution of the Boot ROM Code will trigger a Machine Check Reset.

If the CRCBOOT bit in the System Configuration 0 (FUSE.SYSCFG0) fuse is set, the Boot ROM will perform a CRC scan of the Boot Section in the Flash. Refer to the *CRCSCAN - Cyclic Redundancy Check Memory Scan* section in the data sheet for more details about CRC.

9.3.1 CRC Scan of Boot ROM

The Boot ROM performs a CRC scan of the Boot ROM as part of the boot process. Any fault detected in the Boot ROM will lead to:

- The Machine Check Reset Flag (MCRF) bit in the Reset Flag (RSTCTRL.RSTFR) register, and the Error Detected During System Start-up (BOOT) bit in the Machine Flags A (RSTCTRL.MCFLAGSA) register are set
- The device is reset

In the case of a random or transient failure in the Boot ROM - The user can observe the nature of such a failure by reading RSTCTRL flag registers through the UPDI interface after the next (successful) boot process.

In the case of a permanent failure in the Boot ROM - The device will be stuck in an endless reset loop, never exiting the boot process. The UPDI is not allowed to read RSTCTRL while in the boot process, so to diagnose this problem, the user will have to poll the BOOTDONE bit in the ASI System Status (UPDI.ASI_SYS_STATUS) register and verify that it never gets set.

In a locked device, the UPDI does not have access to the RSTCTRL flag registers.

9.3.2 CRC Scan of Flash

You can configure the CRCSCAN peripheral to perform a memory scan of the Boot Section in Flash before the device leaves the reset state. If this check fails, the CPU is prohibited from starting normal code execution. CRC scanning of Flash during start-up is enabled and controlled by the CRCBOOT bit in FUSE.SYSCFG0.

A successful CRC scan of Flash will have the following outcome:

- Normal code execution starts
- The DONE flag in the Interrupt Flags (CRCSCAN.INTFLAGS) register will be '1'

- The OK bit in the Status A (CRCSCAN.STATUSA) register will be '1'

A non-successful CRC scan will have the following outcome:

- The OK bit in CRCSCAN.STATUSA will be '0'
- If the device is locked:
  - MCHKRF in RSTCTRL.RSTFR is set and the CRC Scan Error (CRC) bit in RSTCTRL.MCFLAGSA is set
  - The device is reset, restarting execution at the start of the Boot ROM Code
- If the device is not locked:
  - The device is NOT reset
  - The CPU executes an eternal loop in the Boot ROM Code, waiting for a UPDI command or a reset
  - You can observe this condition by using the UPDI interface

A non-correctable 2-bit (ECC2) error detected in Flash during a CRC Scan will cause a CRCSCAN failure. A CRC scan of unprogrammed Flash will always result in ECC2 errors, causing the CRC Scan to fail. Therefore, the entire Boot Section of the Flash must be programmed with ECC-valid content if CRCBOOT in FUSE.SYSCFG0 is set.

10. AVR® CPU

10.1 Features

- 2 x 8-Bit, High-Performance AVR RISC CPUs and Interrupt Controllers Operating in Dual-Core Lockstep Mode
  - 135 Instructions
  - Hardware Multiplier
- Dual-Core Lockstep Support is Implemented by:
  - Internal Observation Points and Stack Limit Check Hardware
  - Duplicated CPU Interrupt Controller (CPUINT)
  - Dual, Redundant Comparators Verifying the Dual CPU Cores (Lockstep)
  - Data Bus Parity Check on Read Memory Data
  - Fault Injection in Comparators
- 32 8-Bit Registers Directly Connected to the ALU
- Stack in RAM
- Stack Pointer Limit Check
- Data Bus Parity Check
- Illegal Opcode Check
- Stack Pointer Accessible in I/O Memory Space
- Direct Addressing of up to 64 KB of 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

10.2 Overview

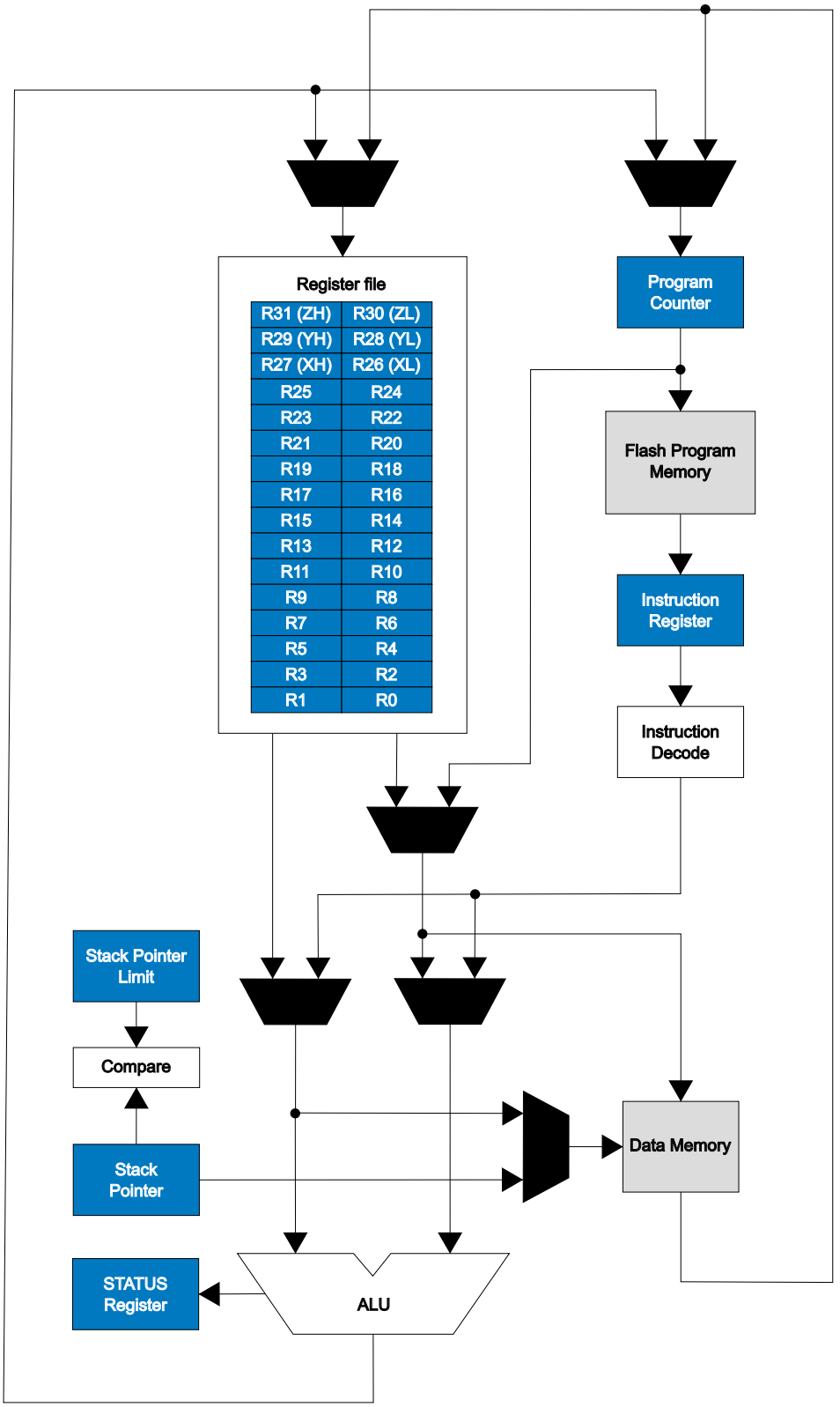
The AVR CPU can access memories, perform calculations, control peripherals, execute instructions from the program memory, and handle interrupts. Interrupt handling is described in a separate section.

10.3 Architecture

The AVR CPU uses a Harvard architecture with separate buses for program and data to maximize performance and parallelism. The instructions in the program memory execute with a single-level pipeline. While one instruction executes, the next instruction is prefetched from the program memory, enabling instructions to be executed on every clock cycle.

Refer to the *Instruction Set Summary* section for an overview of all AVR instructions.

Figure 10-1. AVR CPU Architecture



10.3.1 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 connection with all the 32 general-purpose working registers in the register file. The arithmetic operations between working registers or between a working register and an immediate operand execute in a single clock cycle, and the result is stored in the register file. After an arithmetic or logic operation, the CPU's Status Register (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 an efficient implementation of 32-bit arithmetic, while the hardware multiplier supports signed and unsigned multiplication and fractional formats.

10.3.1.1 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.

10.4 Functional Description

10.4.1 Program Flow

After a reset, the CPU will execute instructions from the lowest address in the flash program memory, `0x0000`. The instruction address on the fetch bus is masked modulo the flash size so that the instruction fetched after the instruction at the last address in the Flash will be the instruction at the first address in the Flash. In other words, the address space wraps around, making it possible to use relative jump instructions past the start and end of the Flash. Note that masking using modulo two assumes that the flash size is a power of two (2^n) to work as intended.

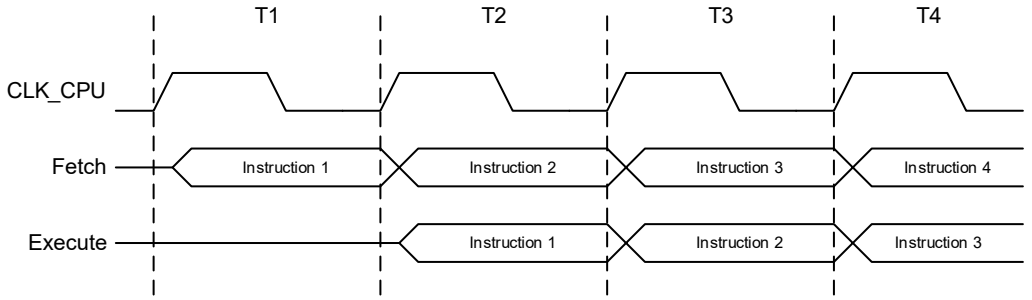
Program flow is supported by conditional and unconditional JUMP and CALL instructions, capable of directly addressing the whole address space. Most AVR instructions use a 16-bit word format, and a limited number use a 32-bit format.

During interrupts and subroutine calls, the return address program counter (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 a reset, the Stack Pointer (SP) 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 SRAM data can easily be accessed through the five different addressing modes supported by the AVR CPU. See the *Instruction Set Summary* section for details.

10.4.2 Instruction Execution Timing

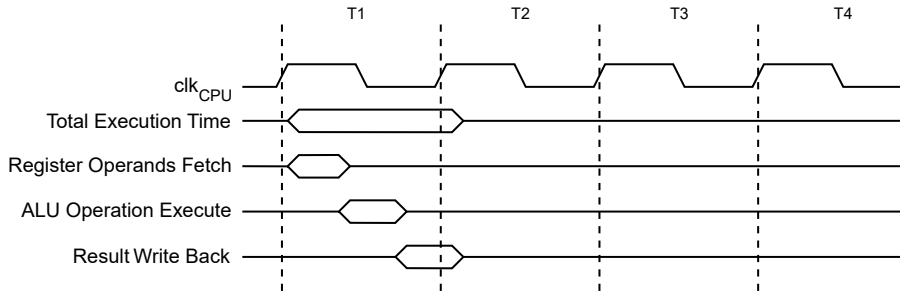
The CPU clock, CLK_CPU, clocks the AVR CPU. No internal clock division is applied. The figure below shows the parallel instruction fetches and executions enabled by the Harvard architecture and the fast-access register file concept, which is the basic pipelining concept enabling up to 1 MIPS/MHz performance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

Figure 10-2. The Parallel Instruction Fetches and Executions



The following figure shows the internal timing concept for the register file. During a single clock cycle, an ALU operation using two register operands executes, and the result is stored in the destination register.

Figure 10-3. Single Cycle ALU Operation



10.4.3 Status Register

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

CPU.SREG is updated after all ALU operations, as specified in the *Instruction Set Summary* section, which 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.

10.4.4 Stack and Stack Pointer

The stack is used to store return addresses after interrupts and subroutine calls and 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. The CPU.SP is implemented as two 8-bit registers accessible in the I/O memory space.

Data are pushed and popped from the stack using the instructions in the table below or by executing interrupts. The stack grows from higher to lower memory locations, implying that pushing data to the stack will decrease the SP, and popping data from the stack will increase the SP.

The SP is automatically set to the highest address of the internal SRAM after a reset. If the stack needs to be allocated to a different SRAM address location (or if multiple stacks are used), the address must fall within the SRAM address space, with sufficient space reserved for the anticipated stack size. See the *SRAM Data Memory* topic in the *Memories* section for the SRAM start address and SRAM size. The new SP must be defined before any subroutine calls execute and interrupts are enabled. See the table below for SP details.

Table 10-1. Stack Pointer Instructions

Instruction	Stack Pointer	Description
PUSH	Decremented by 1	Data are pushed onto the stack
CALL ICALL 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 RETI	Incremented by 2	A return address is popped from the stack with a return from either a subroutine or an interrupt

During interrupts or subroutine calls, the return address is pushed automatically 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). For 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 removed from 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.

10.4.4.1 Stack Pointer Limit Check

The Stack Pointer Limit (SPLIM) register can be programmed with the lower stack limit, meaning the lowest SRAM address location for the stack. The Stack Pointer (SP) is compared continuously to the SPLIM. Whenever SP < SPLIM, a limit check signal is asserted, and the Stack Pointer Limit Error (SPLIM) flag in the Interrupt Flags (INTFLAGS) register is set.

The Stack Pointer Limit Check will also detect a stack underflow, which occurs when a POP instruction makes the SP wrap around from address 0xFFFF to address 0x0000 (the POP instruction increases the SP by one, as described previously).

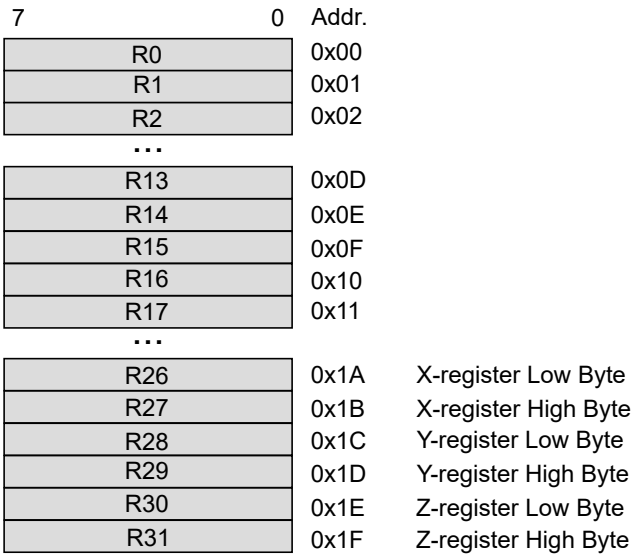
The Stack Pointer Limit Error (SPLIM) flag is also routed to the Error Controller.

10.4.5 Register File

The register file consists of 32 8-bit general purpose working registers used by the CPU. The register file is 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 10-4. AVR® CPU General Purpose Working Registers

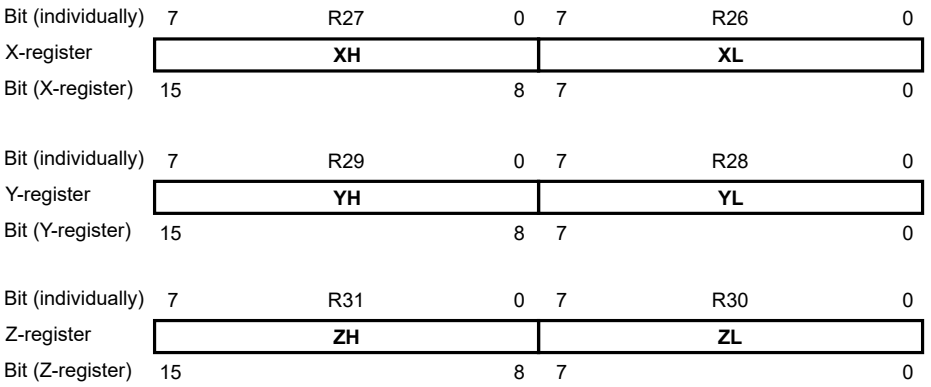


10.4.5.1 The X-, Y-, and Z-Registers

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 an Address Pointer for program memory.

Figure 10-5. The X-, Y-, and Z-Registers



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.

10.4.6 Configuration Change Protection (CCP)

System-critical I/O register settings are protected from accidental modification. Flash self-programming is protected from unexpected execution. This is handled globally by the Configuration Change Protection (CCP) register. An attempted write to a CCP-protected Special Function Register (SFR) outside of a CCP sequence will be discarded and return a bus error response to the bus host attempting the write.

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 register (CPU.CCP).

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 ordinary, and the request is kept pending. After finalizing the CCP period, any pending interrupts are executed according to their level and priority.

The following operations will terminate any active configuration change period immediately:

- Any data write instruction to SFR registers (such as `ST`, `OUT`, `SBI`, ...)
- `SLEEP`
- `LPM`/`SPM`
- Any `LD`/`ST` to flash

It is not possible to restart a configuration change period that has not yet completed by writing to the CPU.CCP register. Any writes to this register before the period has ended will be disregarded. The period must either run out or be aborted by any of the operations listed above.

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

10.4.6.1 Sequence for Write Operation to Configuration Change Protected (CCP) I/O Registers

The following steps are required to write to CCP-protected I/O registers:

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

The configuration change protection re-enables automatically after the CPU executes the write instruction or after four other CPU instructions execute.

10.4.6.2 Sequence for Execution of Self-Programming

Self-programming in this context means the execution of writes (commands) to the Non-Volatile Memory Controller (NVMCTRL). The following steps are required to execute self-programming:

1. The software enables self-programming by writing the Self-Program Memory (SPM) signature to the CCP register (CPU.CCP).
2. Within the next four CPU instructions, the software must execute the appropriate command in the NVM Controller.

The configuration change protection re-enables automatically after the CPU executes the write instruction or after four other CPU instructions execute.

10.4.6.3 Accessing CCP Protected Registers in the CPU

Some of the CPU Special Function Registers (SFRs) are CCP-protected. If accessed outside a CCP sequence, the access is discarded and reported by setting the Bus Error (BUSERR) flag in the Interrupt Flags (INTFLAGS) register.

10.4.7 Data Bus Access

Any access (read or write) to a reserved Special Function Register (SFR) address will abort and return a Data Bus Error.

For more information about the Bus Matrix, refer to the *BUSMATRIX* section.

10.4.8 Data Bus Integrity

The data bus is protected by parity and redundancy:

- The address is protected by a parity bit
- The read data is protected by a parity bit
- The write data is protected by a parity bit
- The read and write strobes are duplicated and consistency checked before being used

A parity error on data received in an `LD` instruction will cause the destination register to be updated, but the value is UNDEFINED. The PARITYD bit in the Interrupt Flag (INTFLAGS) register is set.

A parity error on a fetched instruction will cause the PARITYI bit in the Interrupt Flag (INTFLAGS) register to be set.

Note: This corrupted instruction will be executed and lead to an undefined state in the CPU and/or the memory system. Therefore, it is necessary to enter a safe state by executing the associated NMI and/or Error Controller handling.

The PARITYD and PARITYI flags are routed to the Error Controller.

Parity error injection on the instruction and data bus can be performed using the Inject Parity Error (INJPDD, -DA, -DC, -IA, -IC) bits in the Control A (CTRLA) register.

10.4.9 Data Bus Parity Error Injection

Parity errors can be injected on the address, data and control lines of the instruction bus and data bus, as described in the Control A (CTRLA) register. The read and write strobes are protected by redundancy, so error injection on these is not possible nor needed. The error will be injected on the next bus transfer. Parity errors cannot be injected on accesses to the single-cycle I/O space, only on bus accesses to addresses above 64.

The CTRLA register is on the I/O bus, and a load/store instruction is transferred on the main data bus. The user must respect the timing differences on these buses and use a data memory barrier or similar to ensure that CTRLA has been updated with the command before the load/store arrives. In other words, ensure memory ordering.

See the *BUSMATRIX - Bus Matrix* section for details on the parity injection system, how the CPU and targets can inject parity errors on all bus lines, and where detected errors are reported.

10.4.9.1 Error Injection in Bus Targets

Many bus targets support Error Injection on read data returned from the bus target. This error injection will cause flags in the Interrupt Flag (INTFLAGS) register of the CPU to be set, which ordinarily will request action from the Error Controller and the Interrupt Controller.

Interrupt response can be prevented by writing the Global Interrupt Enable (I) bit in the Status Register (SREG) register to '0', writing the Disable NMI Request (NMIDIS) bit in the Control A (CTRLA) register to '1', and configuring the relevant Error Controller channels to severity NOTIFICATION.

See the *ERRCTRL - Error Controller* section for more information on severity levels.

10.4.9.2 Error Injection in the CPU

The CPU can inject errors in all control signals to the bus targets. This injection should be detected in the bus target and cause the corresponding interrupt flag in the bus target to be set, requesting an interrupt from the interrupt controller and, in most cases, returning a bus error response to the CPU. This bus error response will cause flags in the Interrupt Flags (INTFLAGS) register of the CPU to be set and will request action from the Error Controller and the Interrupt Controller. The Interrupt Flags (INTFLAGS) register in the bus target is connected to the Interrupt Controller as interrupts.

Interrupt response can be prevented by writing the Global Interrupt Enable (I) bit in the Status Register (SREG) register to '0', writing the Disable NMI Request (NMIDIS) bit in the Control A (CTRLA) register to '1', and configuring the relevant Error Controller channels to severity NOTIFICATION.

For details on the error injection connections, see the *BUSMATRIX - Bus Matrix* section.

10.4.10 Bus Error

A bus transaction initiated by the CPU can receive a bus error response.

A bus error on data received in an LD instruction will cause the destination register to be updated, but the value is UNDEFINED. The BUSERR flag in the Interrupt Flag (INTFLAGS) register will be set.

A bus error on a fetched instruction will cause the BUSERR flag in the Interrupt Flag (INTFLAGS) register to be set.

The BUSERR flag is routed to the Error Controller.

Note: This corrupted instruction will execute, and data returned in an LD instruction flagged with bus error will be stored in the register file, which leads to an undefined state in the CPU and/or the memory system. Because of this, entering a safe state by executing the associated NMI and/or Error Controller handling is necessary.

10.4.11 Illegal Opcode

Attempted execution of an illegal opcode results in a NOP operation and the setting of the Illegal Opcode Error (OPC) flag in the Interrupt Flags (INTFLAGS) register. A BREAK instruction is considered an Illegal Opcode when the device is LOCKED. This flag is routed to the Error Controller.

10.4.12 Dual Core Lockstep

The lockstep system executes the same program flow on two identical CPUs operating in lockstep, and a comparator compares the state of the two CPUs. A detected mismatch in their state will trigger a Machine Check Reset directly, without handling by the Error Controller.

The lockstep comparator is located outside the main CPU.

To inject a comparator error, use the Inject Comparator Error (INJCOMP) bit in the Control A (CTRLA) register. Writing this bit will instruct the comparator to inject an error and trigger a Machine Check Reset.

See the *Functional Safety Concept* section for more information about the functional safety mechanisms included in this device family.

10.4.13 On-Chip Debug Capabilities

The AVR CPU includes native On-Chip Debug (OCD) support. It contains 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 *UPDI - Unified Program and Debug Interface* section for details about OCD.

10.4.14 OCD Monitor

The CPU requests a Machine Check Reset if it detects that the OCD system is enabled while the device is in the Locked state.

10.4.15 Interrupts

Table 10-2. Available Interrupts Vectors and Sources

Interrupt Vector Name	Interrupt Source Name	Description	Condition
CPU	PARITYI	Parity Error on Instruction Bus detected	The Parity Error on Instruction Bus (PARITYI) flag in CPU.INTFLAGS is '1'
	PARITYD	Parity Error on Data Bus detected	The Parity Error on Data Bus (PARITYD) flag in CPU.INTFLAGS is '1'
	SPLIM	Stack Pointer Limit Error detected	The Stack Pointer Limit Error (SPLIM) flag in CPU.INTFLAGS is '1'
	BUSERR	Bus Error detected	The Bus Error (BUSERR) flag in CPU.INTFLAGS is '1'
	OPC	Illegal Opcode detected	The Illegal Opcode Error (OPC) flag in CPU.INTFLAGS is '1'

The Interrupt Flags (INTFLAGS) register of the CPU is connected to the CPU Interrupt Controller (CPUINT) as Non-Maskable Interrupts (NMIs). The CPU does not have a corresponding Interrupt Control register since NMIs can, by design, not be masked or disabled. For more information about NMIs, see the *CPUINT - CPU Interrupt Controller* section.

Interrupt response can be prevented by writing the Global Interrupt Enable (I) bit in the Status Register (SREG) register to '0', writing the Disable NMI Request (NMIDIS) bit in the Control A (CTRLA) register to '1' and configuring the relevant Error Controller channels to severity NOTIFICATION.

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

10.4.16 Events

None.

10.4.17 Sleep Mode Operation

The CPU will halt in all sleep modes.

10.4.18 CPU Registers Under Configuration Change Protection

The CPU has registers that are under Configuration Change Protection (CCP). In order to write to these registers, a specific key must first be written to the CPU's CCP register, followed by a write access to the protected register within four CPU instructions.

Attempting to write a protected register in the CPU without following the appropriate CCP unlock sequence leaves the protected register unchanged and returns a bus error response to the bus host attempting the write.

The following registers are under CCP:

Table 10-3. CPU - Registers under Configuration Change Protection

Register	Key
CPU.CTRLA	IOREG
CPU.INTFLAGS	IOREG

10.5 Functional Safety

The Functional Safety aspects of the CPU are detailed in the *Functional Safety Concept* section.

10.6 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	SPLIM	7:0	SPLIML[7:0]							
		15:8	SPLIMH[15:8]							
0x02 ... 0x03	Reserved									
0x04	CCP	7:0	CCP[7:0]							
0x05	CTRLA	7:0	NMIDIS	INJCOMP	INJPDD	INJPDA	INJPDC	INJPIA	INJPIC	SPLOCK
0x06	INTFLAGS	7:0				PARITYI	PARITYD	SPLIM	BUSERR	OPC
0x07 ... 0x0C	Reserved									
0x0D	SP	7:0	SPL[7:0]							
		15:8	SPH[7:0]							
0x0F	SREG	7:0	I	T	H	S	V	N	Z	C

10.7 Register Description

10.7.1 Stack Pointer Limit

Name: SPLIM
Offset: 0x0
Reset: Top of stack
Property: -

The Stack Pointer Limit Register (SPLIM) marks the lower limit of the stack. If the Stack Pointer (SP) has a value lower than SPLIM, a limit check signal is asserted, and the Stack Pointer Limit Error (SPLIM) flag in the Interrupt Flags (INTFLAGS) register is set.

The Stack Pointer Limit Error (SPLIM) flag in the Interrupt Flags (INTFLAGS) register is also routed to the Error Controller.

The CPU.SPLIML and CPU.SPLIMH register pair represents the 16-bit value, CPU.SPLIM. 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
	SPLIMH[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
	SPLIML[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 15:8 – SPLIMH[15:8] Stack Pointer Limit High Byte
These bits hold the MSB of the 16-bit register. The contents of this register can be locked, preventing it from further updates, by the SPLOCK bit in the Control A (CTRLA) register.

Bits 7:0 – SPLIML[7:0] Stack Pointer Limit Low Byte
These bits hold the LSB of the 16-bit register. The contents of this register can be locked, preventing it from further updates, by the SPLOCK bit in the Control A (CTRLA) register.

10.7.2 Configuration Change Protection

Name: CCP
Offset: 0x04
Reset: 0x0
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 writing to protected I/O registers or executing protected instructions within the following four CPU instructions. During this period all interrupts are suspended, but the interrupt flags are set. After completing the period, any pending interrupts execute according to their level and priority.
After writing the protected I/O register signature, CCP[0] will read as '1' as long as the CCP feature is active.
After writing the protected self-programming signature, CCP[1] will read as '1' as long as the CCP feature is active.
CCP[7:2] will always read as '0'.

Value	Name	Description
0x9D	SPM	Allow self-programming
0xD8	IOREG	Un-protect protected I/O registers

10.7.3 Control A

Name: CTRLA
Offset: 0x5
Reset: 0x0
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	NMIDIS	INJCOMP	INJPDD	INJPDA	INJPDC	INJPIA	INJPIC	SPLOCK
Access	R/W	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 – NMIDIS Disable NMI Request
This bit will disable any Non-Maskable Interrupt (NMI) during error injection, so that any error injection that will set a flag in CPU.INTFLAGS does not trigger an NMI.

Value	Name	Description
0x0	ENABLED	NMI is enabled
0x1	DISABLED	NMI is disabled

Bit 6 – INJCOMP Inject Comparator Error
Write this bit to '1' to inject a compare error in the lockstep comparator. This bit is automatically cleared by HW after one instruction has completed.

Bit 5 – INJPDD Inject Parity Error on Data Bus Data
Write this bit to '1' to inject a parity error in the data on the following data bus write access. This bit is automatically cleared after one access is completed. Parity errors will be injected only on main data bus accesses, not on accesses to the bus that address the single-cycle I/O registers (first 64 address locations).

Bit 4 – INJPDA Inject Parity Error on Data Bus Address
Write this bit to '1' to inject a parity error in the address on the following data bus access. This bit is automatically cleared after one access is completed. Parity errors will be injected only on main data bus accesses, not on accesses to the bus that address the single-cycle I/O registers (first 64 address locations).

Bit 3 – INJPDC Inject Parity Error on Data Bus Control
Write this bit to '1' to inject a parity error in the data bus control on the following data bus access. This bit is automatically cleared after one access is completed. Parity errors will be injected only on main data bus accesses, not on accesses to the bus that address the single-cycle I/O registers (first 64 address locations).

Bit 2 – INJPIA Inject Parity Error on Instruction Bus Address
Write this bit to '1' to inject a parity error in the address of the following instruction fetch. This bit is automatically cleared after one fetch is completed.

Bit 1 – INJPIC Inject Parity Error on Instruction Bus Control
Write this bit to '1' to inject a parity error in the control signal of the following instruction fetch. This bit is automatically cleared after one fetch is completed.

Bit 0 – SPLOCK SPLIM Lock
Write this bit to '1' to lock SPLIM. When SPLIM is locked it is not possible to change the SPLIM value.

10.7.4 Interrupt Flags

Name: INTFLAGS
Offset: 0x6
Reset: 0xXX
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
				PARITYI	PARITYD	SPLIM	BUSERR	OPC
Access				R/W	R/W	R/W	R/W	R/W
Reset				x	x	x	x	x

Bit 4 – PARITYI Parity Error on Instruction Bus
This flag is set when a parity error is detected in an instruction read on the instruction bus.
Write this bit to '1' to clear the flag.
The CPU interrupt request will be asserted when this flag is set.

Bit 3 – PARITYD Parity Error on Data Bus
This flag is set when a parity error is detected in the data read on the data bus. PARITYD is also set on parity errors on the bus that addresses the single-cycle I/O registers (first 64 address locations).
Write this bit to '1' to clear the flag.
The CPU interrupt request will be asserted when this flag is set.

Bit 2 – SPLIM Stack Pointer Limit Error
This flag is set when the stack pointer limit is exceeded. In other words, when the stack has overflowed or underflowed.
Write this bit to '1' to clear the flag.
The CPU interrupt request will be asserted when this flag is set.

Bit 1 – BUSERR Bus Error
This flag is set when the bus target returns a bus target error signal on the bus. This error signal is triggered by an illegal address, parity error or other target-dependent error.
Write this bit to '1' to clear the flag.
The CPU interrupt request will be asserted when this flag is set.

Bit 0 – OPC Illegal Opcode Error
This flag is set when an illegal opcode has been fetched and decoded.
Write this bit to '1' to clear the flag.
The CPU interrupt request will be asserted when this flag is set.

Note: This register has no accompanying INTCTRL register since all flags are registered as NMIs, which are always enabled. It is possible to prevent these interrupts by setting the Disable NMI Request (NMIDIS) bit in the Control A (CTRLA) register to DISABLED.

Note: This register is under Configuration Change Protection since the interrupt flags may indicate a hardware error and are considered more severe than ordinary interrupt flags.

10.7.5 Stack Pointer

Name: SP
Offset: 0xD
Reset: 0x7FFF
Property: -

This 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 SRAM is implemented for each device. Unused bits will always read as '0'.

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

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

Bit	15	14	13	12	11	10	9	8
	SPH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	SPL[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 – SPH[7:0] Stack Pointer High Byte
These bits hold the MSB of the 16-bit register SP.

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

10.7.6 Status Register

Name: SREG
Offset: 0xF
Reset: 0x0
Property: -

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

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
Writing a '1' to this bit enables interrupts on the device.
Writing a '0' to this bit disables the 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 Bit Copy Storage
The bit copy instructions, Bit Load (BLD) and Bit Store (BST), use the T bit as the source or destination for the operated bit.
A bit from a register in the register file can be copied into this bit by the BST instruction, and this bit can be copied into a bit in a register in the register file by the BLD instruction.

Bit 5 – H Half Carry Flag
The Half Carry (H) flag is set when there is a half carry in an arithmetic operation that supports this and is cleared otherwise. Half carry is helpful for performing BCD arithmetic.

Bit 4 – S Sign Bit, S = N ⊕ V
The Sign (S) bit is always an exclusive or (XOR) between the Negative flag (N) and the Two's Complement Overflow (V) flag.

Bit 3 – V Two's Complement Overflow Flag
The Two's Complement Overflow (V) flag is set when there is an overflow in the arithmetic operation that support this and is cleared otherwise.

Bit 2 – N Negative Flag
The Negative (N) flag is set when there is a negative result in the arithmetic or logic operation and is cleared otherwise.

Bit 1 – Z Zero Flag
The Zero (Z) flag is set when there is a zero result in the arithmetic or logic operation and is cleared otherwise.

Bit 0 – C Carry Flag
The Carry (C) flag is set when there is a carry in the arithmetic or logic operation and is cleared otherwise.

11. BUSMATRIX - Bus Matrix

11.1 Features

- Performs Bus Arbitration Between Multiple Initiators
- Matrix Interconnect Between Initiators and Targets
- Functional Safety Features
  - Parity of address, data and control signals
  - Redundancy of bus control signals
  - Target can signal Bus Error to the initiator

11.2 Overview

The bus controller handles all access on the data buses, acting as the main interface between the initiators and target buses.

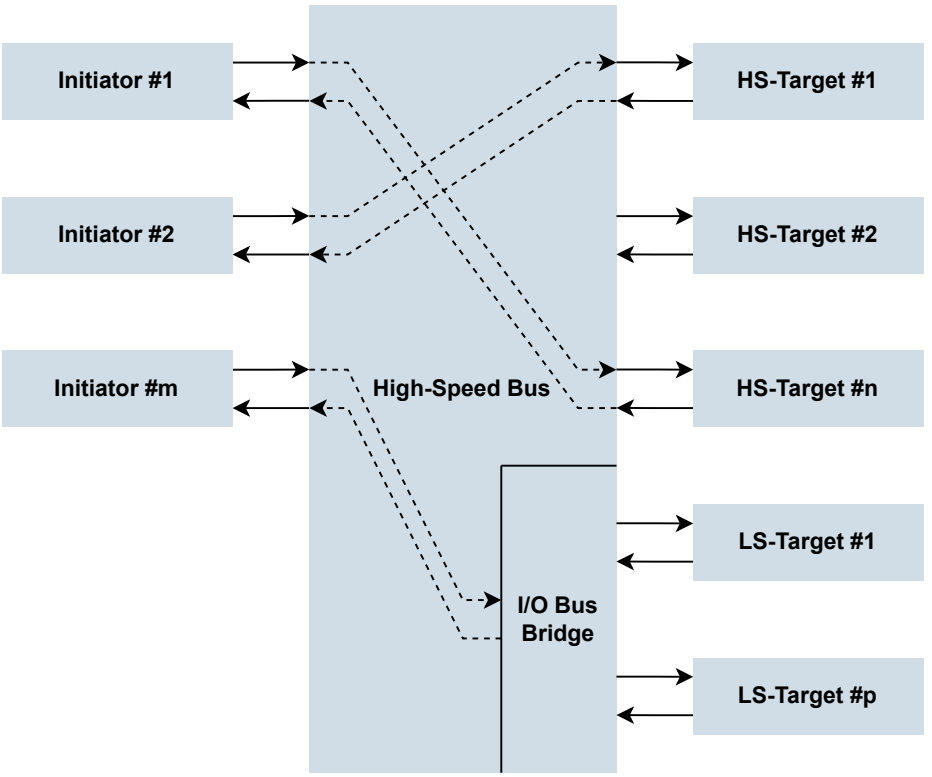
The bus matrix executes any data transfers from the requesting initiators to the selected bus targets. The bus targets can respond to a transfer request by either completing the access or ignoring the access and returning a Bus Error signal. This signal is returned to the bus initiator and handled in a bus-initiator-specific way, usually by discarding the transfer and setting an interrupt flag in the bus initiator.

Parity and other redundancy and consistency checks protect the data bus signal integrity to detect random errors occurring on the data bus. Both data bus initiators and data bus targets can detect parity errors and other data bus consistency errors. Detection of bus-related errors happens in an initiator- or target-specific way, usually by discarding the transfer and setting a bus error interrupt flag in the initiator. Refer to the data sheet of the respective initiator and target for information on how bus errors are handled.

The bus controller handles address decoding and bus arbitration for all the initiators in the system. An unlimited number of initiators and targets is supported, but some restrictions apply. Two initiators can access different target buses at the same time. The arbiter will take action only when a conflict occurs. This bus section is called the high-speed bus. Low-bandwidth peripherals and interfaces are connected to an I/O bus bridge. Only one transfer can be active on the I/O bus at a time. The USART and timers are examples of peripherals connected to the I/O bus. Due to only facilitating one connection at a time, this section is regarded as low-speed.

11.2.1 Block Diagram

Figure 11-1. Bus Matrix - Block Diagram



11.3 Functional Description

11.3.1 Bus Initiator Priority

The bus initiators are assigned a priority. If multiple initiators address the same target simultaneously, the initiator with the highest priority wins the arbitration. The priority is as follows:

Table 11-1. Bus Initiator Priority

Initiator/Target	Priority
CPU Load/Store	1 (highest)
CPU Instruction Fetch	2
UPDI	3
CRC Scan	4 (lowest)

11.3.2 Protection of Data Bus

The signals and data integrity on the data bus are protected by redundancy and parity as follows:

Table 11-2. Protection of Data Bus

Signal Path	Signals	Protection	
		Parity	Redundancy
Initiator → Bus Matrix → Target	Address	x	
	Write Data	x	
	Control Signals	x	x

.....continued

Signal Path	Signals	Protection	
		Parity	Redundancy
Target → Bus Matrix → Initiator	Read Data	x	

11.3.2.1 Generation of Protective Signals

The bus initiator generates the following signals:

- Parity on address
- Parity on write data (data going from initiator to target)
- Parity and redundant signals on control signals

The data bus generates the following signals:

- Redundant control signals

The target generates the following signals:

- Parity on read data (data going from target to initiator)

11.3.2.2 Checking of Protective Signals

The bus initiator checks the following signals:

- Parity of read data
- Bus error response from data bus

The bus matrix checks the parity of addresses mapped to the I/O bus bridge to ensure that the initiator select signals generated from this address are correct. Address parity on high-speed bus addresses is not checked by the bus matrix but is left to the high-speed bus targets.

The target checks the following signals:

- Parity on address
- Parity on write data
- Consistency of bus control signals

11.3.3 Bus Error

The bus system, comprising the bus matrix, the peripheral bus and, the targets on the bus, can return a Bus Error response to the bus initiator that initiated a failing access. The bus error response is signaled on a dedicated bus line between the target and the initiator. A bus error can be generated both by the bus matrix and by the targets connected to the bus. A target that detects a bus error will normally discard the access. In other words, the access attempt will not result in a read or write to the target. Refer to the data sheet section corresponding to the specific target for information on handling bus errors.

11.3.3.1 Bus Error From Bus Matrix

The bus matrix will return a bus error to the bus initiator if the initiator requests a bus transfer to or from a reserved address in the data address space.

11.3.3.2 Bus Error From Bus Target

The bus target will return a bus error to the bus initiator if the initiator requests a bus transfer to or from a reserved address in the bus target address space or if any of the already mentioned checks fail.

A target may return bus error in additional, target-specific ways. Refer to the target's documentation for information on specific bus error triggers. Examples of triggers leading to a bus error response can be:

- Reading from a write-only register

- Writing to a read-only register
- Writing outside of a Configuration Change Protection (CCP) writing sequence to a register under CCP
- Accessing a 16-bit register in the wrong sequence

A target that detects a bus error will normally discard the access as follows:

- A write will not update any Special Function Registers (SFR), memories, or any other state in the target
- A read will cause the target to output an undefined value on the data bus, along with the target error signal

For more information on how various targets handle bus errors, refer to the section on that specific target.

11.3.3.3 Reporting of Bus Error

Each initiator will have a status or interrupt flag register with flags indicating that a transfer originating from that initiator got a bus error reply or if the received data had a parity error. These flags can be used to request an interrupt and/or be connected to the Error Controller.

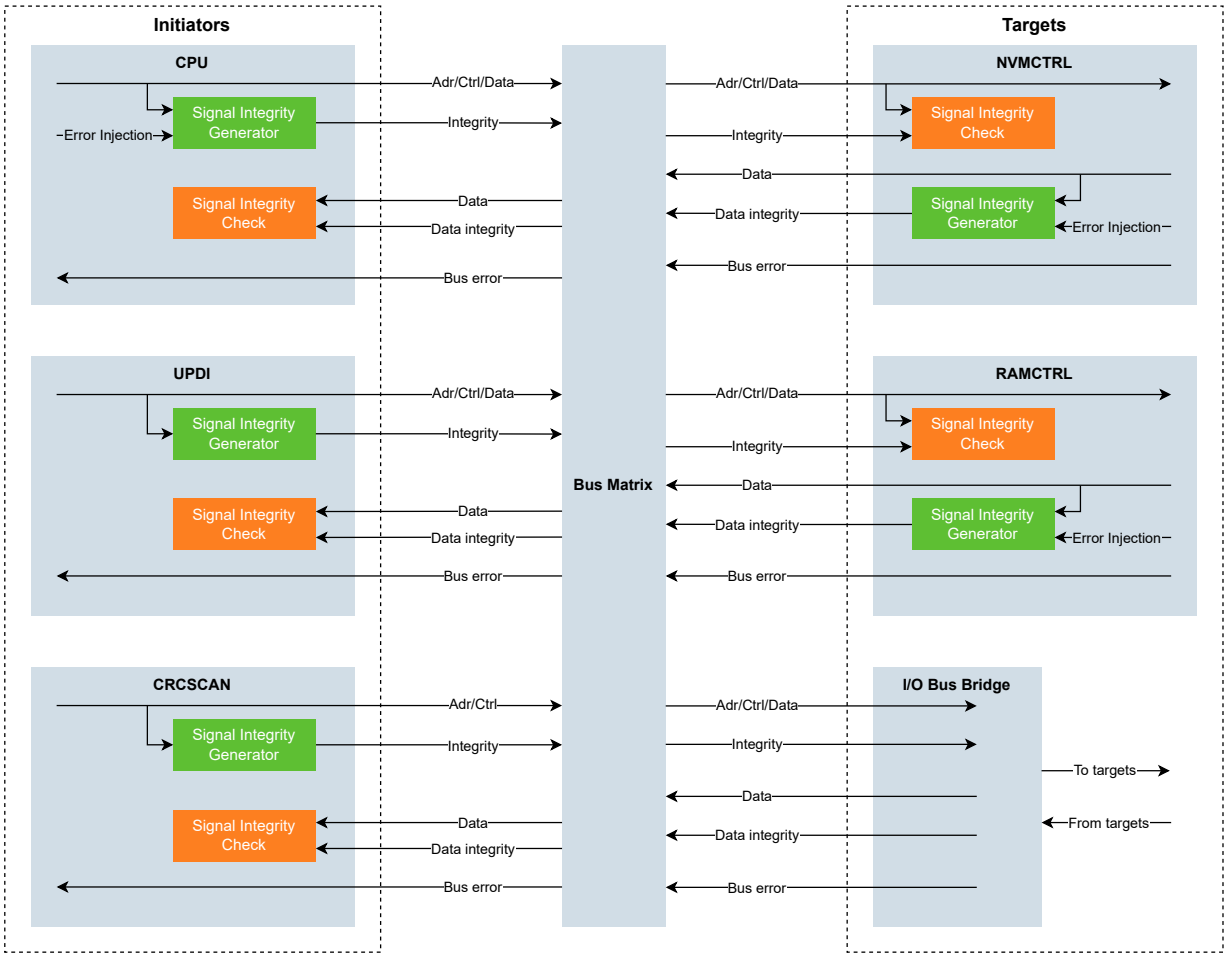
11.4 Functional Safety

The bus matrix is designed for functional safety systems with the following signal integrity protection features:

- Parity of address and data
- Parity and redundancy on control signals
- Target can signal an error to the initiator

The figure below shows how the module's signal integrity protection and error injection interact with typical peripherals.

Figure 11-2. Bus Matrix - Signal Integrity Protection and Error Injection



12. Memories

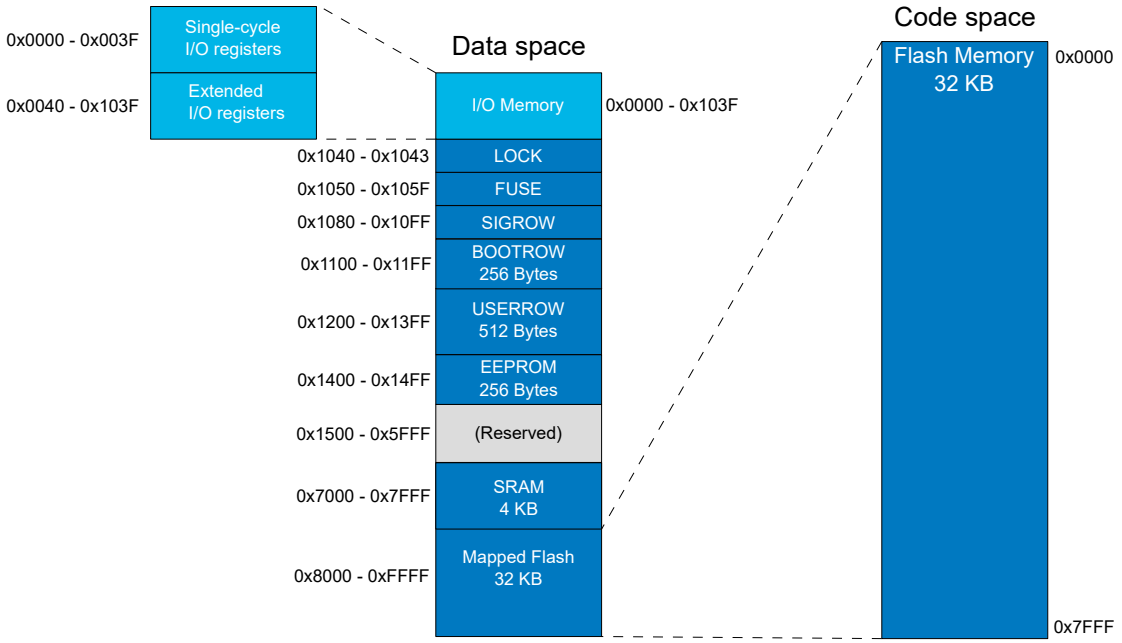
12.1 Overview

The main memories of the AVR32SD20/28/32 devices are SRAM data memory space, EEPROM data memory space, and Flash program memory space. The peripheral registers are located in the I/O memory space.

12.2 Memory Map

The figure below shows the memory map for the 32 KB variant of the AVR SD. Refer to the subsequent sections and the *Peripheral Address Map*> table for further details.

Figure 12-1. Memory Map



12.3 In-System Reprogrammable Flash Program Memory

The AVR32SD20/28/32 contains 32 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 a 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. The code placed in one section may be restricted from writing to addresses in other sections. The Program Counter can address the whole program memory.

Refer to the Code Size (CODESIZE) and Boot Size (BOOTSIZE) descriptions and the *NVMCTRL - 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 documentation of the Nonvolatile Memory Controller (NVMCTRL) peripheral.

The 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 12-1. Physical Properties of Flash Memory

Property	AVR32SD20 AVR32SD28 AVR32SD32
Size	32 KB
Page size	512B
Number of pages	64
Start address in data space	0x8000
Start address in code space	0x0

12.4 Program and Debug Interface Disable (PDID)

After activating the *Program and Debug Interface Disable (PDID)*, the only way to write to the reprogrammable Flash memory (nonvolatile memory - NVM) is from the Boot Code section of the NVM. Consequently, CHIPERASE or other re-programming attempts through the UPDI will fail. Also, any attempt to read out any NVM content will fail.

Use the following procedure to enable the PDID feature (restrict access to NVM):

1. Write `0xB452` to the PDI Configuration (PDICFG) fuse:
   - Provide the NVM Protection Active (NVMACT) key by writing `0xB45` to bits PDICFG[15:4] (KEY)
   - Bits PDICFG[3:2] are unused. Ensure they are zero.
   - Select the Protection Level *NVM Access Disabled* (NVMACCDIS) by writing `0x2` to PDICFG[1:0] (LEVEL)
2. Write the Lock Key Bits (KEY) in the LOCK.KEY fuse to LOCKED.
3. Reset the device.

Once protection level NVMACCDIS is invoked, the following access rules apply:

- NVM access through UPDI is disabled
- Updates to the application software can only be performed by code located in the Boot Code section (bootloader)
- Chip Erase is disabled
- User Row write access is disabled
- CRC status will be available

⚠ WARNING Unlike for locked devices, performing a CHIPERASE through the UPDI interface once the PDID feature is activated is impossible. The only way to alter the NVM content after PDID activation is by executing NVM writes from the Boot Code section (bootloader). The application software must ensure that the bootloader implementation fulfills the security requirements.

12.5 SRAM Data Memory

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

Table 12-2. Physical Properties of SRAM Memory

Property	AVR32SD20 AVR32SD28 AVR32SD32
Size	4 KB
Start address	0x7000

12.6 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) peripheral.

Table 12-3. Physical Properties of EEPROM Memory

Property	AVR® SD Family
Size	256B
Start address	0x1400

12.7 SIGROW - Signature Row

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

All the AVR32SD20/28/32 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). The device ID for the AVR32SD20/28/32 devices consists of three signature bytes, as shown in the following table.

Table 12-4. Device ID

Device Name	Signature Byte Address and Value		
	0x00	0x01	0x02
AVR32SD32	0x1E	0x95	0x52
AVR32SD28	0x1E	0x95	0x53
AVR32SD20	0x1E	0x95	0x54

12.7.1 Signature Row Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	DEVICEID0	7:0	DEVICEID[7:0]							
0x01	DEVICEID1	7:0	DEVICEID[7:0]							
0x02	DEVICEID2	7:0	DEVICEID[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 ... 0x0F	Reserved									
0x10	SERNUM0	7:0	SERNUM[7:0]							
...										
0x1F	SERNUM15	7:0	SERNUM[7:0]							

12.7.2 Signature Row Description

12.7.2.1 Device ID

Name: DEVICEIDn
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.DEVICEID[2:0].

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

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

12.7.2.2 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 *ADC - Analog-to-Digital Converter* section for a description of how to use the value stored in this bit field.

12.7.2.3 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 number 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

12.8 BOOTROW - Boot Row

The AVR32SD20/28/32 devices have a memory section called the Boot Row (BOOTROW). The BOOTROW can be accessed from only the BOOT section on a locked device and may be used to store data such as keys.

For more details, refer to the *System Memory Address Map* and the *NVMCTRL - Nonvolatile Memory Controller* sections.

12.9 USERROW - User Row

The AVR32SD20/28/32 devices have a special 512-byte memory section called the User Row (USERROW). The USERROW can be used for end-production data and is not affected by chip erase. It can be written by the UPDI even if the part is locked, which enables the storage of the 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.

12.9.1 User Row Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	USERROW0	7:0	USERROW[7:0]							
...										
0x01FF	USERROW511	7:0	USERROW[7:0]							

12.9.1.1 User Row

Name: USERROW
Offset: 0x00 + n*0x01 [n=0..511]
Default: 0xFF
Property: R/W

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

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

Bits 7:0 – USERROW[7:0] User Row

12.10 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 fuses are not affected by chip erase. 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 preprogrammed 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'.

12.10.1 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 ... 0x04	Reserved									
0x05	SYSCFG0	7:0	CRCBOOT	CRCSEL					BROWSAVE	EESAVE
0x06	SYSCFG1	7:0	WDTMON[1:0]			MVSYSCFG[1:0]		SUT[2:0]		
0x07	CODESIZE	7:0	CODESIZE[7:0]							
0x08	BOOTSIZE	7:0	BOOTSIZE[7:0]							
0x09	Reserved									
0x0A	PDICFG	7:0	KEY[3:0]						LEVEL[1:0]	
		15:8	KEY[11:4]							

12.10.2 Fuse Description

12.10.2.1 Watchdog Configuration

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

The default value given in this fuse description is the factory-programmed value and must 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 during 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 during Reset.

12.10.2.2 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 start-up.

The default value given in this fuse description is the factory-programmed value and must 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

Notes:

- Refer to *BOD and POR Characteristics* in the *Electrical Characteristics* section for further details
- Values in the description are typical values

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 *BOD - 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 *BOD - 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 *BOD - Brown-out Detector* section for further details.

Value	Name	Description
0x0	DISABLE	BOD disabled
0x1	ENABLE	BOD enabled in Continuous mode

Value	Name	Description
0x2	SAMPLE	BOD enabled in Sampled mode
0x3	-	Reserved

12.10.2.3 System Configuration 0

Name: SYSCFG0
Offset: 0x05
Default: 0x00
Property: -

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

Bit	7	6	5	4	3	2	1	0
	CRCBOOT	CRCSEL					BROWSAVE	EESAVE
Access	R	R					R	R
Default	0	0					0	0

Bit 7 – CRCBOOT CRC Boot
This bit controls whether the Boot section of the Flash will be checked by the CRCSCAN peripheral during the Reset initialization. Refer to the *CRCSCAN - Cyclic Redundancy Check Memory Scan* section for more information about the functionality.

Value	Name	Description
0	DISABLE	No CRC
1	ENABLE	CRC of the Boot section

Bit 6 – CRCSEL CRC Mode Selection
This bit controls the type of CRC performed by the CRCSCAN peripheral. Refer to the *CRCSCAN - Cyclic Redundancy Check Memory Scan* section for more information about the functionality.

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

Bit 1 – BROWSAVE Boot Row Save During Chip Erase
This bit controls whether the Boot Row is erased or preserved during a chip erase.

Value	Name	Description
0	DISABLE	Boot Row is erased during chip erase
1	ENABLE	Boot Row is preserved during a chip erase

Bit 0 – EESAVE EEPROM Save During Chip Erase
This bit controls whether the EEPROM is erased or preserved during a chip erase.

Value	Name	Description
0	DISABLE	EEPROM is erased during chip erase
1	ENABLE	EEPROM is preserved during a chip erase

12.10.2.4 System Configuration 1

Name: SYSCFG1
Offset: 0x06
Default: 0x48
Property: -

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

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

Bits 7:6 – WDTMON[1:0] WDT Monitor Configuration
This bit field controls the WDT Clock Monitor configuration.

Value	Name	Description
0x0	OFF	The monitor is disabled
0x1	ON	The monitor is always on
0x2	SLEEP	The monitor is on in normal and idle mode. The monitor is disabled in Power Down and Standby sleep modes.
0x3	-	Reserved

Bits 4:3 – MVSYSCFG[1:0] MVIO System Configuration
This bit field controls the MVIO system configuration.

Value	Name	Description
0x0	-	Reserved
0x1	DUAL	Dual supply configuration
0x2	SINGLE	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	Name	Description
0x0	0MS	0 ms
0x1	1MS	1 ms
0x2	2MS	2 ms
0x3	4MS	4 ms
0x4	8MS	8 ms
0x5	16MS	16 ms
0x6	32MS	32 ms
0x7	64MS	64 ms

12.10.2.5 Code Size

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

The default value given in this fuse description is the factory-programmed value and must 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
This bit field controls the combined size of the Boot Code section and Application Code section in blocks of 512 bytes. For more details, refer to the *NVMCTRL - Nonvolatile Memory Controller* section.
Note: If FUSE.BOOTSIZE is `0x00`, the entire Flash is the Boot Code section.

12.10.2.6 Boot Size

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

The default value given in this fuse description is the factory-programmed value and must 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 bit field 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 *NVMCTRL - Nonvolatile Memory Controller* section.

12.10.2.7 Programming and Debug Interface Configuration

Name: PDICFG
Offset: 0x0A
Default: 0x0003
Property: -

Note: These fuses are only effective after a Reset (Reset initialization has run) and if the device is in the locked state.

Bit	15	14	13	12	11	10	9	8
	KEY[11:4]							
Access	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	KEY[3:0]						LEVEL[1:0]	
Access	R	R	R	R			R	R
Default	0	0	0	0			1	1

Bits 15:4 – KEY[11:0] NVM Protection Activation Key

Value	Name	Description
0xB45	NVMACT	NVM protection active
Other	-	Not active

Bits 1:0 – LEVEL[1:0] Protection Level

Value	Name	Description
Other	—	Reserved
0x2	NVMACCDIS	Program and Debug Interface Disable (PDID): NVM access through UPDI is permanently disabled.
0x3	BASIC	The UPDI peripheral and the UPDI pin are working as described in the UPDI section

⚠ CAUTION After NVMACCDIS is selected, it is impossible to re-enable UPDI access for device programming. Before entering this mode, fully comprehended the consequences: Chip Erase and User Row writes are blocked, and any programming needs to go through a bootloader in the Boot Code section.

Note: After NVMACCDIS activation, access to NVM is very restricted for external testing. Some testing will be possible, but advanced failure analysis will not be possible. The CRC status is available.

12.11 LOCK - Memory Sections Access Protection

The device can be locked so that the memories cannot be read using the UPDI. The locking protects 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 Lock Key can be read by the CPU or UPDI but can only be programmed or cleared by the UPDI. The device is locked by writing a nonvalid key to the Lock Key (LOCK.KEY) register.

Table 12-5. 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
SIGROW	Yes	No	Yes	No
USERROW	Yes	Yes	Yes	Yes
BOOTROW	Yes	Yes	Yes	Yes
FUSE	Yes	No	Yes	Yes
LOCK	Yes	No	Yes	Yes
Registers	Yes	Yes	Yes	Yes

Table 12-6. 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
SIGROW	Yes	No	No	No
USERROW	Yes	Yes	No	Yes[2]
BOOTROW	Yes[3]	Yes[3]	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 out.
3. Access from boot section only.

Important: The only way to unlock a device is a CHIPERASE. The USERROW will not be erased, but no application data is retained. When the Programming and Debug Interface Configuration (PDICFG) fuse is written accordingly, even the CHIPERASE is disabled, and all programming needs to go through the bootloader.

12.11.1 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]							

12.11.2 Lock Description

12.11.2.1 Lock Key

Name: KEY
Offset: 0x00
Default: 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
Default	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
Default	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
Default	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
Default	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

12.12 I/O Memory

All AVR32SD20/28/32 devices I/O and peripheral registers are located in the I/O memory space. Refer to the *Peripheral Address Map* for further details.

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

12.12.1 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` instruction.

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

- 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.

12.12.2 Extended I/O Registers

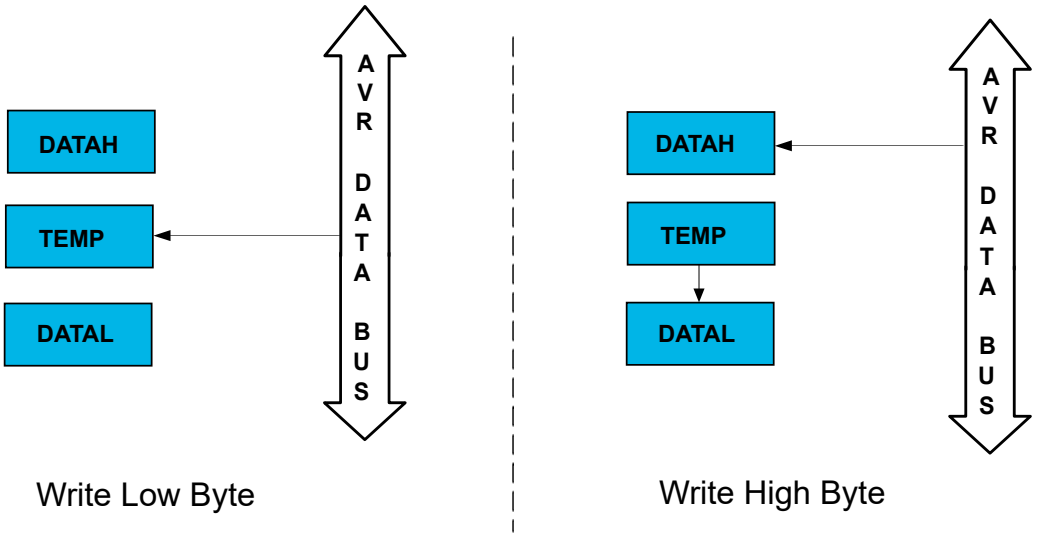
The I/O memory space ranging from 0x0040 to 0x103F can only be accessed by the `LD/LDS/LDD` or `ST/STS/STD` instructions, transferring data between the 32 general purpose working registers (R0-R31) and the I/O memory space.

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

12.12.3 Accessing 16-Bit Registers

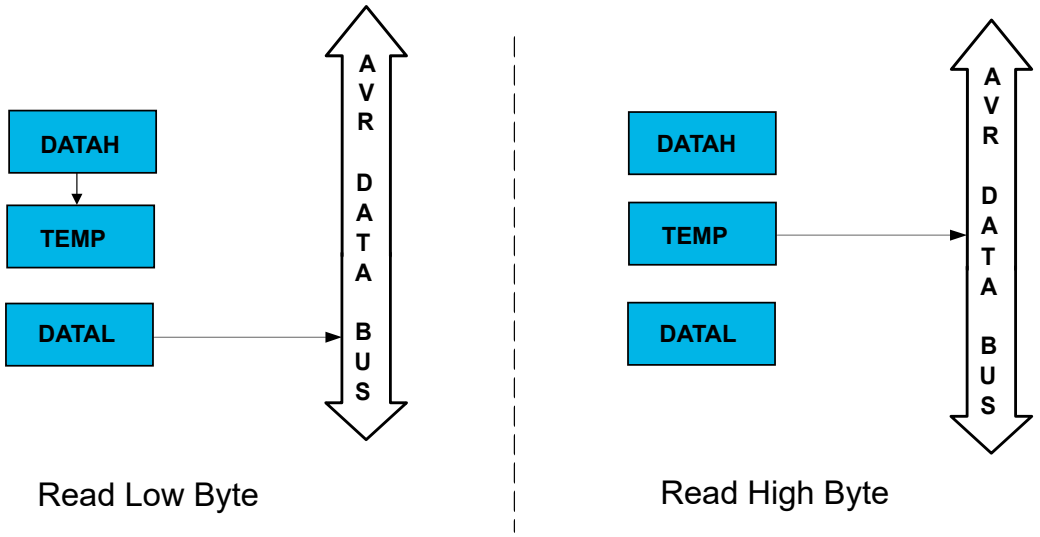
Most of the registers for the AVR32SD20/28/32 devices are 8-bit registers, but the devices also feature a few 16-bit registers. As the AVR data bus has a width of 8 bits, accessing the 16-bit requires two read or write operations. All the 16-bit registers of the AVR32SD20/28/32 devices are connected to the 8-bit bus through a temporary (TEMP) register.

Figure 12-2. 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 register instead of the low byte register, as shown on the left side of 16-Bit Register Write Operation. 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 16-Bit Register Write Operation.

Figure 12-3. 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 register in the same clock cycle, as shown on the left side of 16-Bit Register Read Operation. Reading the high byte register will result in a read from TEMP instead of the high byte register, as shown on the right side of 16-Bit Register Read Operation.

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

The 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 (IRS). To prevent this, the interrupts must 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 IRS.

12.12.4 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 (MSB) must be written last when writing to the register, and the Least Significant Byte (LSB) must be read first when reading the register.

13. Peripherals and Architecture

13.1 Peripheral Address Map

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

Table 13-1. Peripheral Address Map

Base Address	Name	Description	20-Pin	28-Pin	32-Pin
0x0000	VPORTA	Virtual Port A	X	X	X
0x0008	VPORTC	Virtual Port C	X	X	X
0x000C	VPORTD	Virtual Port D	X	X	X
0x0014	VPORTF	Virtual Port F	X	X	X
0x001C	GPR	General Purpose Registers	X	X	X
0x0030	CPU	CPU	X	X	X
0x0040	RSTCTRL	Reset Controller	X	X	X
0x0050	SLPCTRL	Sleep Controller	X	X	X
0x0060	BOD	Brown-Out Reset Detector	X	X	X
0x0070	VREF	Voltage Reference	X	X	X
0x0080	CLKCTRL	Clock Controller	X	X	X
0x0100	MVIO	MVIO Controller	X	X	X
0x0110	WDT	Watchdog Timer	X	X	X
0x0120	CPUINT	Interrupt Controller	X	X	X
0x0130	CRCSCAN	Cyclic Redundancy Check Memory Scan	X	X	X
0x0140	RTC	Real-Time Counter	X	X	X
0x01C0	CCL	Configurable Custom Logic	X	X	X
0x0200	EVSYS	Event System	X	X	X
0x0400	PORTA	Port A Configuration	X	X	X
0x0440	PORTC	Port C Configuration	X	X	X
0x0460	PORTD	Port D Configuration	X	X	X
0x04A0	PORTF	Port F Configuration	X	X	X
0x05E0	PORTMUX	Port Multiplexer	X	X	X
0x0600	ADC0	Analog-to-Digital Converter 0	X	X	X
0x0640	ADC1	Analog-to-Digital Converter 1	X	X	X
0x0680	AC0	Analog Comparator 0	X	X	X
0x0688	AC1	Analog Comparator 1	X	X	X
0x0690	AC2	Analog Comparator 2	X	X	X
0x06A0	DAC0	Digital-to-Analog Converter 0	X	X	X
0x06C0	ZCD0	Zero Cross Detector 0		X	X
0x06D8	ZCD3	Zero Cross Detector 3	X	X	X
0x0800	USART0	Universal Synchronous Asynchronous Receiver Transmitter 0	X	X	X
0x0820	USART1	Universal Synchronous Asynchronous Receiver Transmitter 1	X	X	X
0x0840	USART2	Universal Synchronous Asynchronous Receiver Transmitter 2		X	X
0x0900	TWI0	Two-Wire Interface 0	X	X	X
0x0920	TWI1	Two-Wire Interface 1			X

.....continued					
Base Address	Name	Description	20-Pin	28-Pin	32-Pin
0x0940	SPI0	Serial Peripheral Interface 0	X	X	X
0x0960	SPI1	Serial Peripheral Interface 1	X	X	X
0x0A00	TCA0	Timer/Counter Type A instance 0	X	X	X
0x0B00	TCB0	Timer/Counter Type B instance 0	X	X	X
0x0B10	TCB1	Timer/Counter Type B instance 1	X	X	X
0x0B20	TCB2	Timer/Counter Type B instance 2	X	X	X
0x0B30	TCB3	Timer/Counter Type B instance 3	X	X	X
0x0B80	TCD0	Timer/Counter Type D instance 0	X	X	X
0x0E20	SWDT	Synchronous Watchdog Timer	X	X	X
0x0E30	RAMCTRL	Ram Controller	X	X	X
0x0E40	ERRCTRL	Error Controller	X	X	X
0x0F00	SYSCFG	System Configuration	X	X	X
0x1000	NVMCTRL	Nonvolatile Memory Controller	X	X	X

Table 13-2. System Memory Address Map

Base Address	Name	Description	20-Pin	28-Pin	32-Pin
0x1040	LOCK	Lock bits	X	X	X
0x1050	FUSE	User configuration fuses	X	X	X
0x1080	SIGROW	Signature row	X	X	X
0x1100	BOOTROW	Boot row	X	X	X
0x1200	USERROW	User row	X	X	X

13.2 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 peripheral's Interrupt Flags (peripheral.INTFLAGS) register 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 (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt is enabled, and the interrupt flag is set. Interrupts must be enabled globally for an 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 13-3. Interrupt Vector Mapping

Vector Number	Program Address (word)	Interrupt Vector (name)	Interrupt Source (name)	Description	20-pin	28-pin	32-pin	48-pin
0	0x00	RESET	-	Device Reset interrupt	X	X	X	X

.....continued

Vector Number	Program Address (word)	Interrupt Vector (name)	Interrupt Source (name)	Description	20-pin	28-pin	32-pin	48-pin
1	0x02	NMI	NMINONCRITICAL	Non-Maskable Non-Critical interrupt from ERRCTRL	X	X	X	X
			ERROR	Non-Maskable Error interrupt from CRCSCAN	X	X	X	X
			BUSERR	Non-Maskable Bus Error interrupt from CPU	X	X	X	X
			OPC	Non-Maskable Illegal Opcode interrupt from CPU	X	X	X	X
			PARITYD	Non-Maskable Parity Data Error interrupt from CPU	X	X	X	X
			PARITYI	Non-Maskable Parity Instruction Error from CPU	X	X	X	X
			SPLIM	Non-Maskable Stack Pointer Limit interrupt from CPU	X	X	X	X
2	0x04	BOD_VLM	VLM	Voltage Level Monitoring interrupt from BOD	X	X	X	X
3	0x06	ERRCTRL_INT	NONCRITICAL	Non-Critical interrupt from ERRCTRL	X	X	X	X
4	0x08	CLKCTRL_INT	CFD0	Clock Failure Detect 0 interrupt from CLKCTRL	X	X	X	X
			CFD1	Clock Failure Detect 1 interrupt from CLKCTRL	X	X	X	X
			CFMD0	Clock Frequency Measurement Done 0 interrupt from CLKCTRL	X	X	X	X
			CFMD1	Clock Frequency Measurement Done 1 interrupt from CLKCTRL	X	X	X	X
			CFM0	Clock Frequency Measurement Error 0 interrupt from CLKCTRL	X	X	X	X
			CFM0	Clock Frequency Measurement Error 1 interrupt from CLKCTRL	X	X	X	X
5	0x0A	SLPCTRL_INT	SERR	Sleep Error interrupt from SLPCTRL	X	X	X	X
			VDENTER	VMON Entered Diagnostic Mode interrupt from SLPCTRL	X	X	X	X
			VDEXIT	VMON Exited Diagnostic Mode interrupt from SLPCTRL	X	X	X	X
			VDIS	VMON Disabled interrupt from SLPCTRL	X	X	X	X
			VERR	VMON Error interrupt from SLPCTRL	X	X	X	X
			VOV	VMON Overvoltage interrupt from SLPCTRL	X	X	X	X
			VSLP	VMON Sleep interrupt from SLPCTRL	X	X	X	X
			VUV	VMON Undervoltage interrupt from SLPCTRL	X	X	X	X
6	0x0C	SWDT_INT	BADC	Bad Clear Command interrupt from SWDT	X	X	X	X
			BADPC	Bad Pre-Clear Command interrupt from SWDT	X	X	X	X
			EXP	Expired Counter interrupt from SWDT	X	X	X	X
			UC	Unexpected Command interrupt from SWDT	X	X	X	X

.....continued

Vector Number	Program Address (word)	Interrupt Vector (name)	Interrupt Source (name)	Description	20-pin	28-pin	32-pin	48-pin
7	0x0E	NVMCTRL_ERROR	COMP	Comparator Mismatch interrupt from NVMCTRL	X	X	X	X
			EECC1	EEPROM ECC 1-bit Error interrupt from NVMCTRL	X	X	X	X
			EECC2	Flash ECC Multibit Error interrupt from NVMCTRL	X	X	X	X
			FECC1	Flash ECC 1-bit Error interrupt from NVMCTRL	X	X	X	X
			FECC2	Flash ECC Multibit Error interrupt from NVMCTRL	X	X	X	X
			PARITYA	Parity Address Error interrupt from NVMCTRL	X	X	X	X
			PARITYC	Parity Control Error interrupt from NVMCTRL	X	X	X	X
			PARITYD	Parity Data Error interrupt from NVMCTRL	X	X	X	X
8	0x10	RAMCTRL_INT	COMP	Comparator Mismatch interrupt from RAMCTRL	X	X	X	X
			ECC1	EEPROM ECC 1-bit Error interrupt from RAMCTRL	X	X	X	X
			ECC2	EEPROM ECC Multibit Error interrupt from RAMCTRL	X	X	X	X
			PARITYC	Parity Control Error interrupt from RAMCTRL	X	X	X	X
			PARITYA	Parity Address Error interrupt from RAMCTRL	X	X	X	X
			PARITYD	Parity Data Error interrupt from RAMCTRL	X	X	X	X
9	0x12	CRCSCAN_INT	DONE	Scan Period Done interrupt from CRCSCAN	X	X	X	X
			PERIOD	Period Done interrupt from CRCSCAN	X	X	X	X
10	0x14	MVIO_MVIO	VDDIO2	VDDIO2 interrupt from MVIO	X	X	X	X
11	0x16	RTC_CNT	CMP	Compare interrupt from RTC	X	X	X	X
			OVF	Overflow interrupt from RTC	X	X	X	X
12	0x18	RTC_PIT	PIT	Periodic Interrupt Timer interrupt from RTC	X	X	X	X
13	0x1A	CCL_CCL	LUTn	LUTn interrupt from CCL	X	X	X	X
14	0x1C	PORTA_PORT	PAn	Pin n interrupt from PORTA	X	X	X	X
15	0x1E	TCA0_OVF	OVF	Overflow interrupt from TCA0 in Normal mode	X	X	X	X
			LUNF	Low Byte Underflow interrupt from TCA0 in Split mode	X	X	X	X
16	0x20	TCA0_HUNF	HUNF	High Byte Overflow interrupt from TCA0 in Split mode	X	X	X	X
17	0x22	TCA0_CMP0	CMP0	Compare 0 interrupt from TCA0 in Normal mode	X	X	X	X
			LCMP0	Low Compare 0 interrupt from TCA0 Split mode	X	X	X	X

.....continued

Vector Number	Program Address (word)	Interrupt Vector (name)	Interrupt Source (name)	Description	20-pin	28-pin	32-pin	48-pin
18	0x24	TCA0_CMP1	CMP1	Compare 1 interrupt from TCA0 in Normal mode	X	X	X	X
			LCMP1	Low Compare 1 interrupt from TCA0 Split mode	X	X	X	X
19	0x26	TCA0_CMP2	CMP2	Compare 2 interrupt from TCA0 in Normal mode	X	X	X	X
			LCMP2	Low Compare 2 interrupt from TCA0 Split mode	X	X	X	X
20	0x28	TCB0_INT	CAPT	Capture interrupt from TCB0	X	X	X	X
			OVF	Overflow interrupt from TCB0	X	X	X	X
21	0x2A	TCB1_INT	CAPT	Capture interrupt from TCB1	X	X	X	X
			OVF	Overflow interrupt from TCB1	X	X	X	X
22	0x2C	TCD0_OVF	OVF	Overflow interrupt from TCD0	X	X	X	X
23	0x2E	TCD0_TRIG	TRIGA	Trigger A interrupt from TCD0	X	X	X	X
			TRIGB	Trigger B interrupt from TCD0	X	X	X	X
24	0x30	TWI0_TWIS	DIF	Client Data Transmit or Receive Complete interrupt from TWI0 in Client mode	X	X	X	X
			APIF	Client Address or Stop interrupt from TWI0 in Client mode	X	X	X	X
25	0x32	TWI0_TWIM	RIF	Host Read Complete interrupt from TWI0 in Host mode	X	X	X	X
			WIF	Host Write Complete interrupt from TWI0 in Host mode	X	X	X	X
26	0x34	SPI0_INT	RXCIF	Receive Complete interrupt from SPI0 in Buffered mode	X	X	X	X
			TXCIF	Transmit Complete interrupt from SPI0 in Buffered mode	X	X	X	X
			DREIF	Data Register Empty interrupt from SPI0 in Buffered mode	X	X	X	X
			SSIF	Slave Select interrupt from SPI0 in Buffered mode	X	X	X	X
			IF	Transmit Complete interrupt from SPI0 in Normal mode	X	X	X	X
			WRCOL	Write Collision interrupt from SPI0 in Normal mode	X	X	X	X
27	0x36	SPI1_INT	RXC	Receive Complete interrupt from SPI1 in Buffered mode	X	X	X	X
			TXC	Transfer Complete interrupt from SPI1 in Buffered mode	X	X	X	X
			DRE	Data Register Empty interrupt from SPI1 in Buffered mode	X	X	X	X
			SS	Client Select interrupt from SPI1 in Buffered mode	X	X	X	X
			TXC	Transfer Complete interrupt from SPI1 in Normal mode	X	X	X	X
			WRCOL	Write Collision interrupt from SPI1 in Normal mode	X	X	X	X

.....continued

Vector Number	Program Address (word)	Interrupt Vector (name)	Interrupt Source (name)	Description	20-pin	28-pin	32-pin	48-pin
28	0x38	USART0_RXC	RXCIF	Receive Complete interrupt from USART0	X	X	X	X
			RXSIF	Receive Start-of-Frame interrupt from USART0	X	X	X	X
			ISFIF	Auto-Baud Error interrupt from USART0	X	X	X	X
29	0x3A	USART0_DRE	DREIF	Data Register Empty interrupt from USART0	X	X	X	X
30	0x3C	USART0_TXC	TXCIF	Transmit Complete interrupt from USART0	X	X	X	X
31	0x3E	PORTD_INT	PDn	Pin n interrupt from PORTD	X	X	X	X
32	0x40	AC0_AC	CMP	Compare interrupt from AC0	X	X	X	X
33	0x42	AC1_AC	CMP	Compare interrupt from AC1	X	X	X	X
34	0x44	AC2_AC	CMP	Compare interrupt from AC2	X	X	X	X
35	0x46	ADC0_ERROR	TRIGOVR	Trigger Overrun interrupt from ADC0	X	X	X	X
			SAMPOVR	Sample Overwrite interrupt from ADC0	X	X	X	X
			RESOVR	Result Overwrite interrupt from ADC0	X	X	X	X
36	0x48	ADC0_RESRDY	RESRDY	Result Ready interrupt from ADC0	X	X	X	X
			WCMP	Window Compare interrupt from ADC0	X	X	X	X
37	0x4A	ADC0_SAMPRDY	SAMPRDY	Sample Ready interrupt from ADC0	X	X	X	X
			WCMP	Window Compare interrupt from ADC0	X	X	X	X
38	0x4C	ADC1_ERROR	TRIGOVR	Trigger Overrun interrupt from ADC1	X	X	X	X
			SAMPOVR	Sample Overwrite interrupt from ADC1	X	X	X	X
			RESOVR	Result Overwrite interrupt from ADC1	X	X	X	X
39	0x4E	ADC1_RESRDY	RESRDY	Result Ready interrupt from ADC1	X	X	X	X
			WCMP	Window Compare interrupt from ADC1	X	X	X	X
40	0x50	ADC1_SAMPRDY	SAMPRDY	Sample Ready interrupt from ADC1	X	X	X	X
			WCMP	Window Compare interrupt from ADC1	X	X	X	X
41	0x52	ZCD3_ZCD	CROSSIF	Cross interrupt from ZCD3	X	X	X	X
42	0x54	PORTC_PORT	PCn	Pin n interrupt from PORTC	X	X	X	X
43	0x56	USART1_RXC	RXCIF	Receive Complete interrupt from USART1	X	X	X	X
			RXSIF	Receive Start-of-Frame interrupt from USART1	X	X	X	X
			ISFIF	Auto-Baud Error intterupt from USART1	X	X	X	X
44	0x58	USART1_DRE	DREIF	Data Register Empty interrupt from USART1	X	X	X	X
45	0x5A	USART1_TXC	TXCIF	Transmit Complete interrupt from USART1	X	X	X	X
46	0x5C	PORTF_PORT	PFn	Pin n interrupt from PORTF	X	X	X	X

.....continued

Vector Number	Program Address (word)	Interrupt Vector (name)	Interrupt Source (name)	Description	20-pin	28-pin	32-pin	48-pin
47	0x5E	NVMCTRL_READY	EEREADY	EEPROM Ready interrupt from NVMCTRL	X	X	X	X
48	0x60	TCB2_INT	CAPT	Capture interrupt from TCB2	X	X	X	X
			OVF	Overflow interrupt from TCB2	X	X	X	X
49	0x62	TCB3_INT	CAPT	Capture interrupt from TCB3	X	X	X	X
			OVF	Overflow interrupt from TCB3	X	X	X	X
50	0x64	ZCD0_ZCD	CROSSIF	Cross interrupt from ZCD0	-	X	X	X
51	0x66	USART2_RXC	RXCIF	Receive Complete interrupt from USART2	-	X	X	X
			RXSIF	Receive Start-of-Frame interrupt from USART2	-	X	X	X
			ISFIF	Auto-Baud Error interrupt from USART2	-	X	X	X
52	0x68	USART2_DRE	DREIF	Data Register Empty interrupt from USART2	-	X	X	X
53	0x6A	USART2_TXC	TXCIF	Transmit Complete interrupt from USART2	-	X	X	X
54	0x6C	TWI1_TWIS	DIF	Client Data Transmit or Receive Complete interrupt from TWI1 in Client mode	-	X	X	X
			APIF	Client Address or Stop interrupt from TWI1 in Client mode	-	X	X	X
55	0x6E	TWI1_TWIM	RIF	Host Read Complete interrupt from TWI1 in Host mode	-	X	X	X
			WIF	Host Write Complete interrupt from TWI1 in Host mode	-	X	X	X
56	0x70	PORTB_PORT	PBn	Pin n interrupt from PORTB	-	-	-	X
57	0x72	PORTE_PORT	PEn	Pin n interrupt from PORTE	-	-	-	X

13.3 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.

13.3.1 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]			

13.3.2 Register Description

13.3.2.1 Device Revision ID Register

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

This register is read-only and gives 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. `0x01` = `A`, `0x02` = `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.

14. Getting Started with Software Development

By design, this device has functional safety features that cannot be disabled. The Error Controller (ERRCTRL), in particular, can generate interrupts, float I/O pins, and even reset the device. Refer to the *ERRCTRL - Error Controller* section for additional information. Although these features may be required in a completed system, they can make initial software development difficult because there is not yet any code to properly handle these situations. For getting started with software development, it is thus recommended to immediately configure the ERRCTRL as follows at device start-up:

1. Read the ERRCTRL.ESF (Error Status Flags) register; if it is not zero, write '0xFFFFFFFF' to the register to clear it
2. Place the ERRCTRL in the configuration state by performing a protected write of CONFIG to the STATE bit field of the CTRLA (Control A) register
3. Select an error severity level of NOTIFICATION for each ESCn (Error Source Control n) register by writing NOTIFICATION to the ERRLVL bit field; this ensures that the ERRCTRL will neither generate interrupts nor reset the system
4. Place the ERRCTRL back in the normal state by performing a protected write of NORMAL to the STATE bit field of the CTRLA register

Example code for performing these steps is as follows:

```
if (ERRCTRL.ESF)
{
    ERRCTRL.ESF = 0xFFFFFFFF; // Clear all Error Status Flags
}
_PROTECTED_WRITE(ERRCTRL.CTRLA, ERRCTRL_STATE_CONFIG_gc); // Place ERRCTRL in CONFIG state
ERRCTRL.ESCVREGFAIL = ERRCTRL_ERRLVL_NOTIFICATION_gc; // Select severity level of
ERRCTRL.ESCBUSERR = ERRCTRL_ERRLVL_NOTIFICATION_gc; // NOTIFICATION for all error channels
ERRCTRL.ESCRAM2 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCFLASH2 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCOPC = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCSPLIM = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCRAM1 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCFLASH1 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCVREGWARN = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCCFD0 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCCFD1 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCCFM0 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCCFM1 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCSWDT = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCEEPROM = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCEVSYS0 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
ERRCTRL.ESCEVSYS1 = ERRCTRL_ERRLVL_NOTIFICATION_gc;
_PROTECTED_WRITE(ERRCTRL.CTRLA, ERRCTRL_STATE_NORMAL_gc); // Place ERRCTRL in NORMAL state
```

Attention: As software development progresses, alter the individual severity levels to either NONCRITICAL or CRITICAL where appropriate.

15. GPR - General Purpose Registers

The AVR32SD20/28/32 devices provide four General Purpose registers (GPR[3:0]). These registers can be used for storing any information, and they are particularly useful for storing global variables and interrupt flags. The GPRn are reset to `0x00` only by either a Brown-out Reset (BOR) or a Power-on Reset (POR). These registers, which reside in the address range 0x1C-0x1F, are directly bit-accessible using the `SBI`, `CBI`, `SBIS`, and `SBIC` instructions.

15.1 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]							

15.2 Register Description

15.2.1 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

16. NVMCTRL - Nonvolatile Memory Controller

16.1 Features

- SEC-DED ECC Check of Flash and EEPROM
- Redundant ECC Check Logic
- Error Injection for Diagnostics
- In-System Programmable
- Self-Programming and Bootloader Support
- Configurable Sections for Write Protection:
  - Boot section for bootloader code or application code
  - Application code section for application code
  - Application data section for application code or data storage
- 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 by the application
  - Can be written from UPDI on a locked device
  - Content is kept after a chip erase
- Boot Row for Boot Data:
  - Only accessible from the boot section
  - Optionally erased through a chip erase

16.2 Overview

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

16.2.1 Block Diagram

Figure 16-1. NVMCTRL Block Diagram

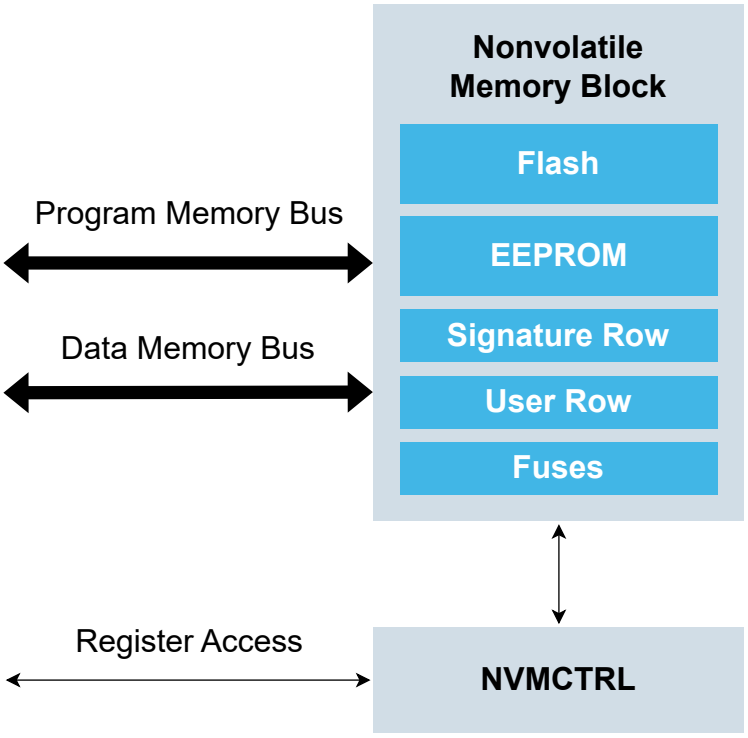
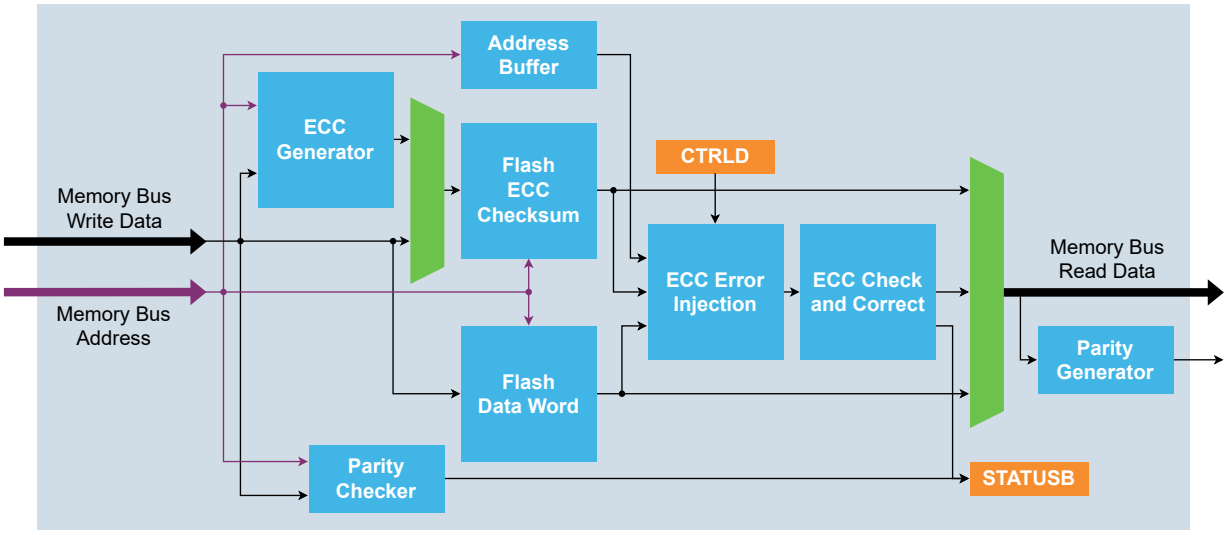


Figure 16-2. ECC System Block Diagram



16.3 Functional Description

16.3.1 Memory Organization

16.3.1.1 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:

Bootloader Code (BOOT) Section

The code placed in the BOOT section has full write access to the entire Flash except for the BOOT section itself. Bootloader software must be placed in this section if used.

Application Code (APPCODE) Section

The code placed in the Application Code section has limited write access and can only write to the Application Data Flash section. This section typically contains the executable application code.

Application Data (APPDATA) Section

The code placed in the Application Data section has no write access. This section typically contains the parameters.

Inter-Section Write Protection

For security reasons, writing to the Flash section where the code is currently executing is impossible. Code writing to the APPCODE section must be executed from the BOOT section, and the code writing to the APPDATA section must be executed from either the BOOT section or the APPCODE section.

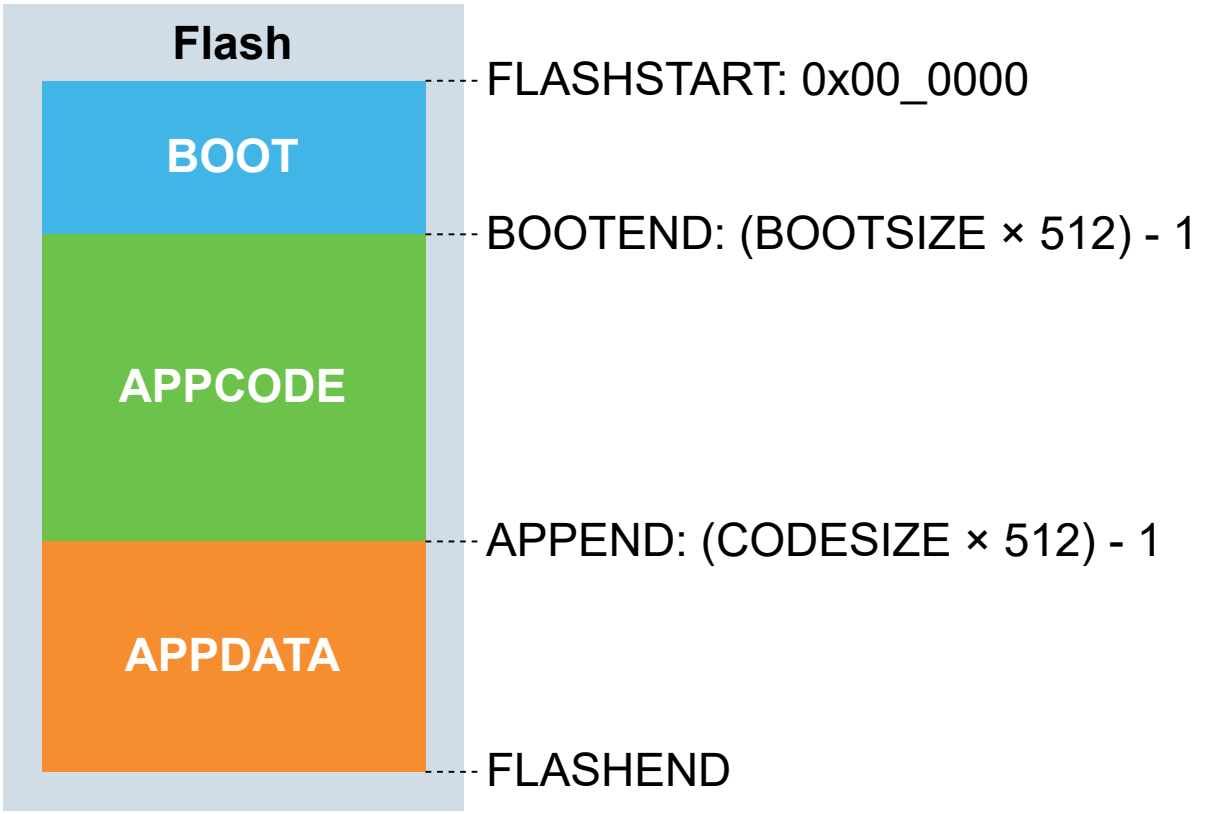
Table 16-1. Write Protection for Self-Programming

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 to APPEND. The remaining area is the APPDATA section.

Figure 16-3. 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 16-2. 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

The application code can be located in the BOOT section if the bootloader software is absent. This increases the flash space available for application code, and still allows for an optional Application Data section.

Notes:

1. After Reset, the default vector table location is at the start of the APPCODE section. The interrupt vector table can be moved to the beginning of the BOOT section if the code running in the BOOT section requires interrupts. Setting the Interrupt Vector Select (IVSEL) bit in the CPUINT.CTRLA register, a bootloader can service interrupts while updating the main code in the Application Code section. Refer to the *CPUINT - CPU Interrupt Controller* section for details.
2. If BOOTEND/APPEND, as result of the BOOTSIZE/CODESIZE fuse setting, exceeds the device FLASHEND, the corresponding fuse setting is ignored, and the default value is used. Refer to *FUSE - Configuration and User Fuses* in the *Memories* section for default values.

Example 16-1. Size of Flash Sections

If FUSE.BOOTSIZE is written to `0x04` and FUSE.CODESIZE 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

In addition 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.

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

Attempting to read a protected memory area with insufficient privileges shall cause the Flash controller to return a no-operation (NOP) status and cause a bus error. Examples of such illegal reads can be:

1. Reading BOOT from APPCODE when BOOTRP is set.
2. Jumping to BOOT from APPCODE and attempting to fetch an instruction from BOOT.

16.3.1.2 EEPROM

The EEPROM is a 256 bytes nonvolatile memory section having 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 can also do a byte erase and write in one operation.

16.3.1.3 Signature Row

The Signature Row contains a Device ID identifying each microcontroller device type and a serial number for each manufactured device. The serial number consists of the production lot, wafer number, and the device's wafer coordinates. The CPU or the UPDI interface can read the Signature Row. However, the Signature Row can't be written or erased.

16.3.1.4 User Row

The User Row is 512 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 an unlocked device and written through UPDI, even on a locked device.

16.3.1.5 Boot Row

The Boot Row is 256 bytes. This section can be used to store various data that are only accessible to the bootloader, such as keys. The Boot Row page is either saved or erased during a chip erase, depending on the FUSE configuration.

The Boot Row is erased/written as ordinary Flash. When erasing the Boot Row, the entire row is erased at once and cannot be accessed from the application code and data sections.

16.3.1.6 Fuses

The fuses contain device configuration values and are copied to their target registers at the end of the start-up sequence. Fuses are not altered by a chip erase.

Only the UPDI can configure the fuses, but both the CPU and the UPDI can read them.

16.3.2 Memory Access

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

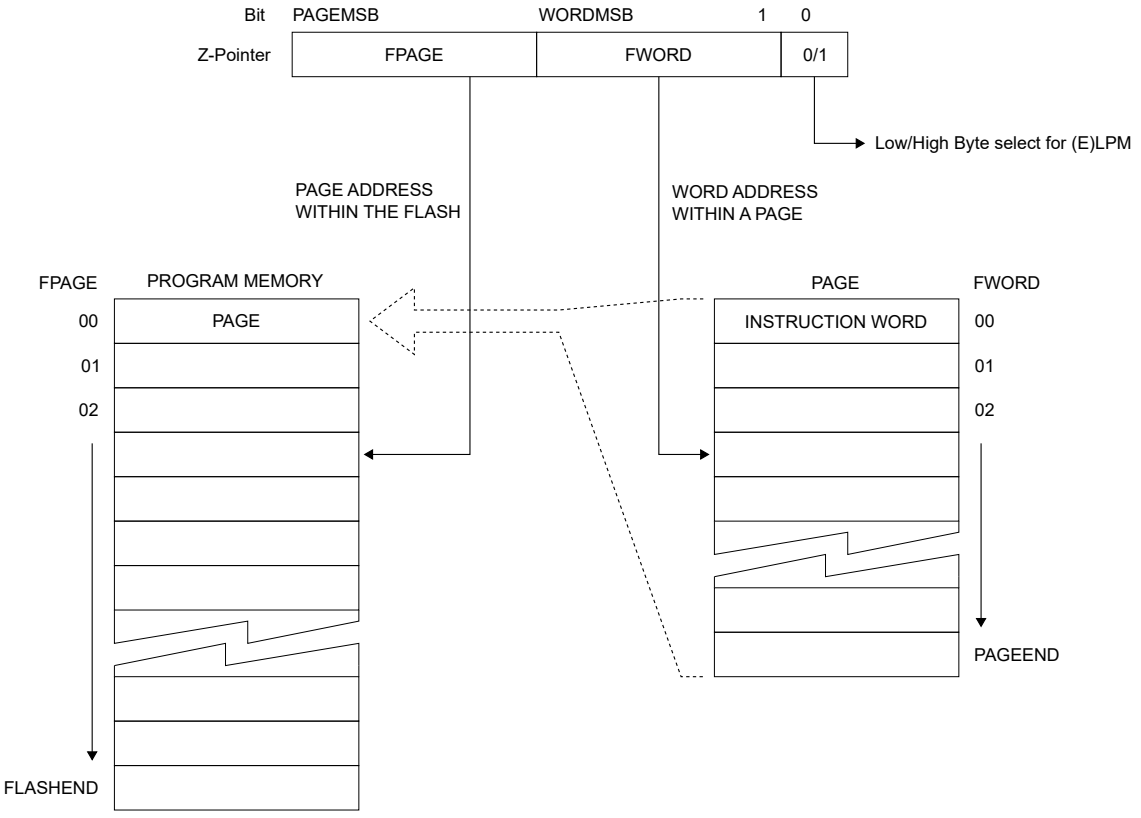
Additionally, when accessed through the CPU data space, the Flash memory is byte accessible, meaning that it shares the same address space and access instructions as SRAM, EEPROM and I/O registers, and it is accessible using `LD`/`ST` instructions.

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, use the Z-pointer for `LPM`/`SPM` access.

Figure 16-4. 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, as shown in the figure above. The word address in the page (FWORD) is held by the Least Significant bits (LSbs) in the Address Pointer, while the Most Significant bits (MSbs) 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. The Least Significant bit (bit 0) in the Address Pointer is used to select the low or high byte in the word address for this. If this bit is '0', the low byte is read, and if this bit is '1', the high byte is read.

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

Addressing Flash in CPU Data Space

The CPU data space is limited to 32 KB. For devices with a 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 Control B (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.

16.3.2.1 Read

Reading the Flash is done using Load Program Memory (LPM) instructions or Load (LD*) instructions with an address according to the memory map. Reading the EEPROM and Signature Row is done using LD* instructions. 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.

16.3.2.2 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 16-3. Programming Granularity

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

Notes:

1. Byte granularity when writing to the CPU data space memory mapped section.
2. One page is 512 bytes.
3. Page granularity when programming from UPDI on a locked device.

16.3.2.3 Command Modes

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

Erasing the entire Flash (CHER) or EEPROM (EECHER) is started by writing commands to the Control A (CTRLA) register. The other write/erase operations are only enabled by writing commands to the CTRLA register and must be followed by writes using ST*/SPM(*) instructions to the memory arrays.
Note: * LPM/SPM cannot be used for EEPROM.

The following sequence must execute to write a command in the CTRLA register:

1. Confirm that any previous operation is completed by reading the Busy (EEBUSY and FBUSY) flags in the STATUS register.
2. Write the appropriate key to the Configuration Change Protection (CPU.CCP) register to unlock the 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 following four instructions.

Note: Skipping the first step will trigger an error in the Status (NVMCTRL.STATUS) register if an operation is still ongoing when issuing a new command.

The following steps are required to perform a write/erase operation in the NVM:

1. Confirm that any previous operation is completed by reading the Busy (EEBUSY and FBUSY) flags in the STATUS register.
2. Optional: If accessing the Flash in the CPU data space, map the corresponding 32 KB Flash section into the data space by writing the FLMAP bit field in the CTRLB register.
3. Write the desired command value to the 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 CTRLA register to clear the current command.

16.3.2.3.1 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 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.

Erasing the address's content is needed before writing to it.

16.3.2.3.2 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.

16.3.2.3.3 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, you can select between erasing 2, 4, 8, 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 16-4. 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.

.....continued		
CMD	Pages Erased	Description
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 the *Flash Addressing for Self-Programming* section for details.

16.3.2.3.4 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 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 the application. The CPU will halt if a new load/store operation starts before the EEPROM erase/write has been completed.

Erasing the EEPROM content is necessary before performing a write to an address.

16.3.2.3.5 EEPROM Erase/Write Mode

The EEPROM Erase/Write (EEERWR) mode enables the EEPROM array for the erase operation, which is then directly followed by a write operation. Several erase/writes can be done while the EEERWR mode is enabled in the 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 the application.

The CPU will halt if a new load or store instruction starts before the erase/write has been completed.

16.3.2.3.6 EEPROM Byte Erase Mode

The EEPROM Byte Erase (EEBER) mode allows 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 running instructions from the Flash. If the CPU starts an erase or write operation while the EEPROM is busy, the CPU will be halted until finishing the current operation.

16.3.2.3.7 EEPROM Multi-Byte Erase Mode

The EEPROM Multi-Byte Erase (EEMBERn) mode allows erasing several bytes in one operation. You can select between erasing 2, 4, 8, 16, or 32 bytes in one operation when enabling the EEMBERn mode.

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 erase the entire range of bytes.

Table 16-5. 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.

The CPU can continue executing instructions from the Flash while erasing the EEPROM. 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 halt until the current operation is finished.

16.3.2.3.8 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 erased by a chip erase, regardless of the EESAVE bit. The read/write protection (BOOTRP, APPCODEWP, APPDATAWP) bits in the CTRLB register 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.

16.3.2.3.9 EEPROM Erase Command

The EEPROM Erase (EECHER) command erases the EEPROM. All EEPROM bytes will read back 0xFF after the operation. The CPU is halted during the EEPROM erase.

16.3.3 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 correctly. These issues are the same on board-level systems using Flash/EEPROM. The internal or an external Brown-out Detector (BOD) is recommended to ensure that the operating voltage is high enough.

Two circumstances may cause Flash/EEPROM corruption when the voltage is too low:

1. A regular write sequence to the Flash, requiring a minimum voltage to operate correctly.
2. The CPU can execute instructions incorrectly when the supply voltage is too low.

The chip erase does not clear fuses. If the BOD is enabled by fuses 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: Taking the following measures may avoid Flash/EEPROM corruption:

1. Keep the device in Reset during periods of insufficient power supply voltage. Do this 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 V_{DD} Reset protection circuit can be used. If a Reset occurs while a write operation is ongoing, the write operation will be aborted.

16.3.4 ECC Correction

The Error Correction Code (ECC) system provides Single Error Correction and Double Error Detection (SEC-DED) capabilities for the Flash and EEPROM. Single-bit errors can be detected and corrected. Double-bit errors cannot be corrected.

The system protects against data being written to or read from the wrong Flash or EEPROM address. The address is included in the ECC parity. Reading instructions or data from NVM is subject to an ECC check, and if the ECC parity does not match that of the read address, an error is flagged in the Interrupt Flags B (INTFLAGSB) register for Flash (FECC1 or FECC2 bits) or EEPROM (EECC1 or EECC2 bits).

An ECC error in the address part of the ECC check word is unrecoverable and cannot be corrected. Such an error indicates that data have been written to or read from the wrong address and will be reported by setting the FECC2 or EECC2 flag in the INTFLAGSB register, and a bus error will be returned to the bus master.

The location of the failing bit is found in the ECC Syndrome (SYNDROME) register, and the failing address is available in the Address (ADDR) register.

The data bus signals are protected through parity and redundancy which protects against errors on the data bus signals, for example, data being read from or written to the wrong address or reads being changed to writes. A detected error on the data bus will lead to the access being discarded and the corresponding PARITY bit in the INTFLAGSB register to be set. For more information, see the *Parity Check* section.

16.3.4.1 ECC Check on EEPROM

The NVM controller has only one set of ECC checkers. These checkers are used for checking the Flash and the EEPROM data. Therefore, the Flash and EEPROM checks must be ran in series, as only one check can be done at a time. The prefetch of the next instruction may need to be stalled in order to serialize the ECC check.

The native width of the Flash is 16 bits, while the width of the EEPROM is 8 bits. For the Flash and EEPROM to use the same ECC generator and checker, the EEPROM data are regarded as 16 bytes wide while being input to the ECC generator and checker. The extension from 8 to 16 bits is done by concatenating `0x00` to the EEPROM byte, with the EEPROM byte being the LSB in the new 16-bit word.

16.3.4.2 ECC Check of All-'1' Data

All bits, both data bits and ECC bits, will read as `1` in an erased Flash. Since the address is baked into the ECC calculation, reading a random word from an erased Flash will usually result in an ECC error which may be problematic when prefetching one address past the last valid Flash instruction or using the Flash to emulate EEPROM or disc. The ECCALL1 bit field in the Control C (CTRLC) register allows disabling of ECC check of Flash words containing all '`1`'s. Enabling the ECCALL1 mechanism will slightly reduce the diagnostic coverage of the ECC mechanism.

The default setting is to check all words, also those containing all '`1`'s. The user must ensure this does not cause an ECC error when prefetching invalid instructions which can be ensured by programming a valid instruction, such as an `NOP`, into all Flash locations that may be prefetched (but later flushed from the pipeline and thus never executed).

16.3.4.3 ECC Check on Accesses from UPDI Bus Initiator

Read accesses from the UPDI bus initiator to Flash or EEPROM will not be subject to ECC check or correction and will not affect flags in INTFLAGSB, ensuring that the debugger/programmer can read a memory range and perform a memory blank check without interference from the ECC system.

16.3.5 Parity Check

Before accessing Flash and EEPROM, the address, write data, and control signals are checked for correct parity. A parity error will cause the memory access to be discarded and the PARITY bit in the INTFLAGSB register to be set.

For reads from Flash and EEPROM, parity on the read data is generated and transmitted on the data bus, allowing the bus initiator to verify the integrity of the read data.

The data bus has additional consistency checks, such as duplicated read/write strobes. Any failed consistency check is flagged as a parity error, i.e., the PARITY flag in the INTFLAGSB register is set, and the transfer is aborted.

16.3.6 Dual ECC Checkers in Lockstep

Critical parts of the ECC checker logic are performed by duplicated lockstep circuitry. A mismatch between the duplicated circuitry in this lockstep logic will set the COMP flag in the INTFLAGSB register.

The ECC encode logic in NVMCTRL is not duplicated. Correctness of written data (and ECC) can be verified, if needed, by a read-back which will pass through the duplicated ECC decode logic. Any error during a write will be automatically detected and flagged when the Flash or EEPROM location eventually is read back.

16.3.7 Error Reporting

NVMCTRL has two bus interfaces, one for instruction fetch and one for data access (LPM/LD/ST/STM). ECC errors occurring in NVM memories are reported as follows:

Table 16-6. ECC Error Information

Memory	Which Information?	How to know?
Flash Fetch or NVM Data	Single- or double-bit	INTFLAGSB
	Word address	ADDR
	ECC Parity bits read from NVM	PARITY
	Failing bit in ECC word	SYNDROME

The NVM controller has several parallel logical interfaces: Flash write, Flash fetch, Flash data, EEPROM write and EEPROM read. A single ECC check circuit performs all ECC checking. Therefore, these checks will be serialized, and only one of the logical interfaces can generate an ECC error at any time. Whenever an ECC error is detected, the ADDR, PARITY and SYNDROME registers are frozen, and the appropriate flags in the INTFLAGSB register are set. The ADDR, PARITY and SYNDROME registers remain frozen as long as any ECC flags in the INTFLAGSB register is set.

An LD or LPM instruction in the address in ADDR is required to retrieve the failing word. Note that this word will already be ECC-corrected, at least in the case of a 1-bit error, so it may appear correct even though the flags indicate otherwise.

16.3.8 Error Bit Addressing

The table below shows how bits are addressed in the ECC error injection and how to pinpoint the localization of the error indicated by the SYNDROME register. Examples:

- A SYNDROME value of 3 indicates bit 3 in the data
- A SYNDROME value of 19 indicates bit 3 in the address
- A SYNDROME value of 43 indicates bit 3 in the parity

Table 16-7. Bit Addressing of ECC Errors

Concatenation	Placement of Bits for ECC Injection and Error localization
INJECT_DATA[46:0]	{ecc_parity[6:0], address[23:0], data[15:0]}

16.3.9 Error Injection

16.3.9.1 ECC Error Injection

Test the ECC logic by injecting single- or double-bit errors into the data read from Flash or EEPROM. An NVM read access (instruction fetch or Flash/EEPROM data) will generate ECC errors, either single-bit corrections or double-bit uncorrectable errors, that are flagged in the INTFLAGSB register.

Error injection will cause flags in the INTFLAGSB register to be set and trigger an interrupt request. An ECC2 error will also return a bus error to the initiator, setting appropriate flags in the initiator (if the CPU is used as the initiator, BUSERR in CPU.INTFLAGS is set), requesting action from the Error Controller and the Interrupt Controller. Interrupt response after error injection can be prevented by:

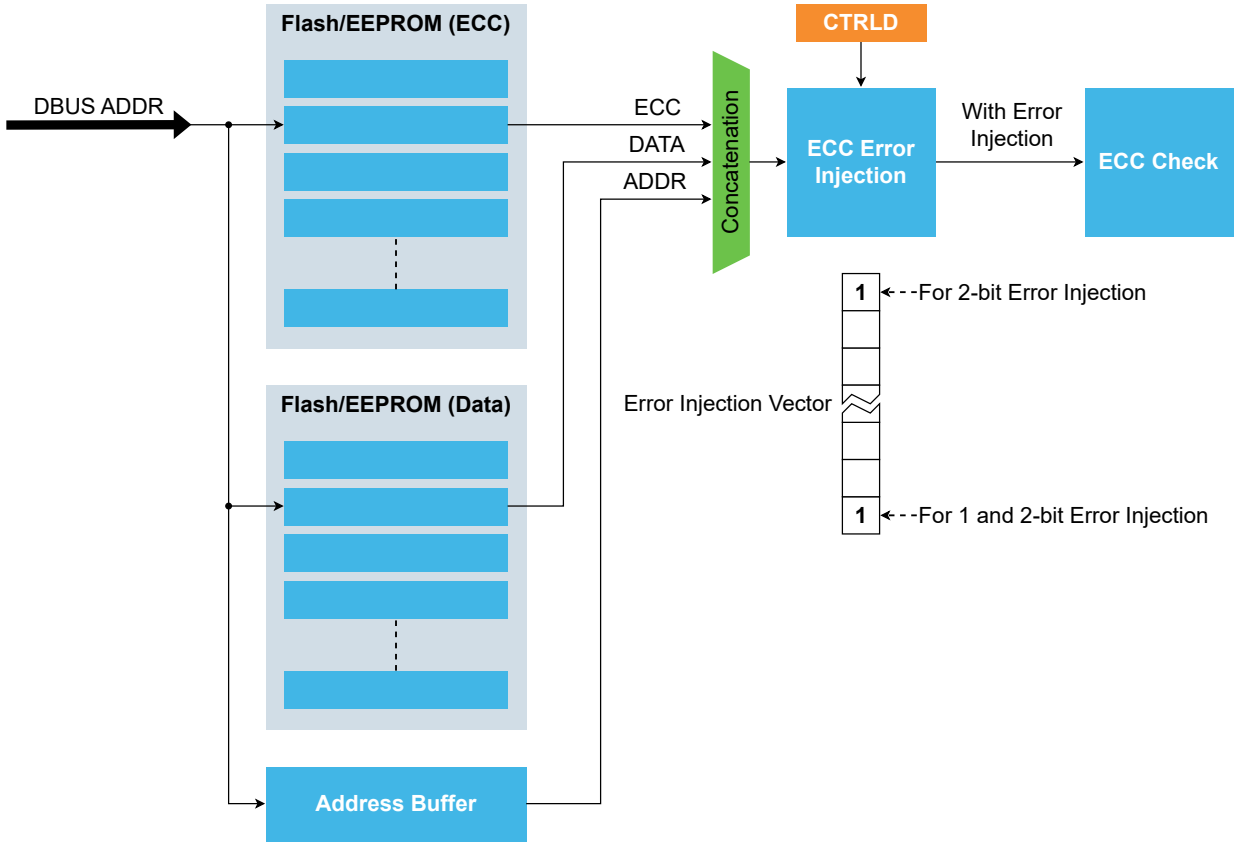
1. Masking interrupt requests from the INTFLAGSB register by clearing the I bit in the CPU.SREG register.
2. Disabling the interrupt request from the initiator. If the CPU is the initiator, this requires setting the NMIDIS bit in the CPU.CTRLA register (since the CPU.INTFLAGS register is connected as NMI).
3. Configuring the relevant Error Controller channels to severity NOTIFICATION.

ECC error injection is enabled by setting the ECC1 or ECC2 bits in the Control D (CTRLD) register to '1'. The next Flash or EEPROM access on the data interface (i.e., LD or LPM) will have a bit error in bit number '0' (the LSb). If ECC2 is set, a bit error will, in addition, be injected in the global parity

bit (the MSb in the *Bit Addressing of ECC Errors* table in the *Error Bit Addressing* section), in other words, a 2-bit error. The bit numbers to insert errors into are hard-coded and not user-selectable to save hardware. The added flexibility of user-addressable bit locations is unnecessary since the ECC checkers are duplicated (hardware redundancy).

The figure below describes error injection into NVM.

Figure 16-5. Error Injection Into NVM



ECC checker comparator mismatch error injection is enabled by setting the COMP bit in the CTRLD register to '1' and CTRLD.DATA to the desired value. The next NVM access will result in a mismatch between the duplicated ECC checkers.

16.3.9.2 Parity Error Injection

Parity errors can be injected on data sent to the data bus (i.e., a load or fetch instruction from the CPU) by setting the PARITYD or PARITYI bits in the CTRLD register.

To test parity error on fetch address, have the CPU inject a parity error on the fetch address by setting the PINJI bit in the CPU.CTRLA register.

To test parity error on data to be stored in NVM, have the CPU inject a parity error on the write data in an `ST`/`SPM` instruction by setting the PINJDD bit in the CPU.CTRLA register. A parity error will also return a bus error to the initiator, setting appropriate flags in the initiator. If the CPU is used as an initiator, the BUSERR bit in the CPU.INTFLAGS register is set.

Note: For more information on how to do error injection and configure the Error Controller, see the *Error Injection in the Error Controller* section.

16.3.10 Interrupts

Table 16-8. Available Interrupt Vectors and Sources

Vector Name	Source Name	Condition	Dependency
NVMCTRL_ERROR	COMP	Mismatch between the duplicated ECC checkers	
	EECC1	Detected a 1-bit error in EEPROM	
	EECC2	Detected a multi-bit error in EEPROM	
	FECC1	Detected a 1-bit error in Flash	
	FECC2	Detected a multi-bit error in Flash	
	PARITYA	Detected a parity error on the bus address signals	
	PARITYC	Detected a parity error on the bus control signals	
	PARITYD	Detected a parity error on the bus data signals	
NVMCTRL_READY	EEREADY	EEPROM Ready	

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags register of the peripheral (NVMCTRL.INTFLAGSA/B).

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

Note: the INTFLAGSB register is routed to ERRCTRL, and are enabled or disabled depending on the ERRCTRL configuration.

An interrupt request is generated when the corresponding interrupt source is enabled and the interrupt flag set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGSA/B register(s) for details on clearing interrupt flags.

16.3.11 Sleep Mode Operation

The NVMCTRL will enter sleep mode if the system is in sleep and there are no more ongoing write/erase operations.

If an NVM write/erase operation is ongoing when the system enters a sleep mode, the Flash memory block, NVMCTRL, and the peripheral clock will remain ON until the operation is finished and will automatically turn off once the operation has completes, which is valid for all sleep modes, including Power-Down.

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

16.3.12 Configuration Change Protection

This peripheral has registers under Configuration Change Protection (CCP). This is a security mechanism to avoid unintentional changes to the NVMCTRL settings. Before writing to these, a specific key must 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 and returns a Bus Error response on the data bus.

The following registers are under CCP:

Table 16-9. NVMCTRL - Registers under Configuration Change Protection

Register	Key
NVMCTRL.CTRLA	SPM
NVMCTRL.CTRLB	IOREG
NVMCTRL.CTRLC	IOREG
NVMCTRL.CTRLD	IOREG
NVMCTRL.INTFLAGSB	IOREG

16.4 Functional Safety

The AVR SD has built in functional safety features in the ECC protection. Random failures can be detected using this feature, as described in the ECC sections of this document.

16.5 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]		EEWP	APPDATAWP	BOOTRP	APPCODEWP
0x02	CTRLC	7:0	ECCALL1[1:0]						BOOTROWWP	UROWWP
0x03	CTRLD	7:0	COMP	PARITYD	PARITYI				ECC2	ECC1
0x04	INTCTRLA	7:0								EEREADY
0x05	INTFLAGSA	7:0								EEREADY
0x06	INTFLAGSB	7:0	COMP	PARITYD	PARITYA	PARITYC	EECC2	EECC1	FECC2	FECC1
0x07	STATUS	7:0			ERROR[2:0]				FBUSY	EEBUSY
0x08	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
0x0B	Reserved									
0x0C	ADDR	7:0	ADDR[7:0]							
		15:8	ADDR[15:8]							
		23:16	ADDR[23:16]							
		31:24								
0x10	PARITY	7:0	PARITY[7:0]							
0x11	SYNDROME	7:0	SYNDROME[7:0]							

16.6 Register Description

16.6.1 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 (CHER) and EEPROM Erase (EECHER) commands start when writing the command. The others commands 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 a No command (NOCMD) or No operation (NOOP) command. If attempting to issue a programming command (except NOCMD or NOOP) while the Flash or EEPROM is busy, a Command Collision error is signalized in the ERROR bit field in the 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 the EESAVE fuse is set. (UPDI access only)
0x30	EECHER	Erase EEPROM
Other	-	Reserved

16.6.2 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]		EEWP	APPDATAWP	BOOTRP	APPCODEWP
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		1	1	0	0	0	0

Bit 7 – FLMAPLOCK Flash Mapping Lock
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit prevents further writing of the FLMAP bit field.

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 accessible through LD/ST instructions.
This bit field controls what part of the Flash, in blocks of 32 KB, will be mapped to the CPU data space.
This bit field is not under Configuration Change Protection.

Value	Name	Mapped Flash Section (KB)			
		8 KB Flash	16 KB Flash	32 KB Flash	64 KB Flash
0	SECTION0	0-8	0-16	0-32	0-32
1	SECTION1				32-64
2	SECTION2				0-32
3	SECTION3				32-64

Bit 3 – EEWP EEPROM Write Protect
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit prevents further updates of the EEPROM.

Bit 2 – APPDATAWP Application Data Write Protection
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit prevents further updates to the Application Data section.

Bit 1 – BOOTRP Boot Section Read Protection
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit protects 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 an NOP instruction. This bit can only be written from the BOOT section. The read protection will only take effect when leaving the BOOT section after the bit is written.

Bit 0 – APPCODEWP Application Code Write Protection
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit prevents further updates to the Application Code section.

16.6.3 Control C

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	ECCALL1[1:0]						BOOTROWWP	UROWWP
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bits 7:6 – ECCALL1[1:0] Erased Flash ECC Check
This bit field configures the ECC-check operation when all '1's are read from Flash.
In an erased Flash, all bits will read as '1' for both the data and ECC bits. Since the address is baked into the ECC calculation, reading a random word from an erased Flash will usually result in an ECC error, which may be problematic when prefetching one address past the last valid Flash instruction or using the Flash to emulate EEPROM or disk.

Value	Name	Description
0x0	CHECKALL	An ECC check on all Flash words, including those read as all '1's
0x1	DISALL	Disregard the ECC check on Flash words read as '1's from the entire Flash
0x2	DISAPPDATA	Disregard the ECC check on Flash words read as '1's from the APPDATA section or USER row
0x3	DISAPPDATAEE	Disregard the ECC check on EEPROM bytes and Flash words read as '1's from the APPDATA, USERROW, or BOOTROW sections

Bit 1 – BOOTROWWP Boot Row Write Protect
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' will enable the Boot Row Write Protect.

Value	Description
0x0	Boot Row Updates disabled
0x1	Boot Row Updates enabled

Bit 0 – UROWWP User Row Write Protect
This bit is cleared by a Reset.
This bit is set by writing a '1' to it.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will enable User Write Row Protect.

16.6.4 Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	COMP	PARITYD	PARITYI				ECC2	ECC1
Access	R/W	R/W	R/W				R/W	R/W
Reset	0	0	0				0	0

Bit 7 – COMP Inject ECC Comparator Mismatch Error
This bit is cleared after injecting an error.
This bit is set by writing a '1' to it.
Writing a '1' to this bit will inject an ECC comparator mismatch error on the next NVM fetch or data read.

Bit 6 – PARITYD Inject Parity Error on Data Read
This bit is cleared after after injecting an error.
This bit is set by writing a '1' to it.
Writing a '1' to this bit will inject a parity error in the data returned from the next Flash or EEPROM data read (using LD or LPM).

Bit 5 – PARITYI Inject Parity Error on Instruction Fetch
This bit is cleared after injecting an error.
This bit is set by writing a '1' to it.
Writing a '1' to this bit will inject a parity error in the next instruction fetched.

Bit 1 – ECC2 Inject 2-bit ECC Error
This bit is cleared after injecting an error.
This bit is set by writing a '1' to it.
Writing a '1' to this bit will inject a 2-bit ECC error on the next NVM data read (i.e., LD or LPM). Error is injected into LSb and Global Parity bit.

Bit 0 – ECC1 Inject 1-bit ECC Error
This bit is cleared after injecting an error.
This bit is set by writing a '1' to it.
Writing a '1' to this bit will inject 1-bit ECC error on the next NVM data read (i.e., LD or LPM). Error is injected into LSb.

16.6.5 Interrupt Control A

Name: INTCTRLA
Offset: 0x04
Reset: 0x00
Property: -

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

Bit 0 – EEREADY EEPROM Ready Interrupt
This bit controls whether the EEPROM Ready interrupt is enabled or not.
Note: 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. Disable the interrupt in the interrupt handler.

Value	Description
0	EEPROM Ready interrupt is disabled
1	EEPROM Ready interrupt is enabled

16.6.6 Interrupt Flags A

Name: INTFLAGSA
Offset: 0x05
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 cleared by writing a '1' to it, or when the EEPROM is busy
This flag is set when the EEPROM is ready to receive commands.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the EEREADY interrupt flag.
Note: This flag is set as soon as the EEPROM is ready after a Reset.

16.6.7 Interrupt Flags B

Name: INTFLAGSB
Offset: 0x06
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	COMP	PARITYD	PARITYA	PARITYC	EECC2	EECC1	FECC2	FECC1
Access	R/W	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 – COMP Comparator Mismatch Error Interrupt Flag
This flag is cleared by writing a '1' to it.
Set when a mismatch between the duplicated ECC checkers is detected. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Comparator Mismatch Error interrupt flag.

Bit 6 – PARITYD Parity Data Error Interrupt Flag
This flag is cleared by writing a '1' to it.
Set when a parity error on data to be written to NVM is detected. A bus error is returned to the bus initiator when this bit becomes set. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Parity Data Error interrupt flag.

Bit 5 – PARITYA Parity Address Error Interrupt Flag
This flag is cleared by writing a '1' to it.
Set when a parity error on an address to be read or written is detected. A bus error is returned to the bus initiator when this bit becomes set. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Parity Address Error interrupt flag.

Bit 4 – PARITYC Parity Control Error Interrupt Flag
This flag is cleared by writing a '1' to it.
Set when a parity error on bus control signals is detected. A bus error is returned to the bus initiator when this bit becomes set. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Parity Control Error interrupt flag.

Bit 3 – EECC2 EEPROM ECC Multi-bit Error Detected Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when a 2-bit (or more) uncorrectable error or an error in the address part is detected. A bus error is returned to the bus initiator when this bit becomes set. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the EEPROM ECC Multi-bit Error Detected interrupt flag.

Bit 2 – EECC1 EEPROM ECC 1-bit Error Detected Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when a 1-bit correctable error in the data part is detected. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the EEPROM ECC 1-bit Error Detected interrupt flag.

Bit 1 – FECC2 Flash ECC Multi-bit Error Detected Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when a 2-bit (or more) uncorrectable error or an error in the address part is detected. A bus error is returned to the bus initiator when this bit becomes set. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Flash ECC Multi-bit Error Detected interrupt flag.

Bit 0 – FECC1 Flash ECC 1-bit Error Detected Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when a 1-bit correctable error in the data part is detected. This bit is routed to ERRCTRL.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Flash ECC 1-bit Error Detected interrupt flag.

16.6.8 Status

Name: STATUS
Offset: 0x07
Reset: 0x00
Property: -

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

Bits 6:4 – ERROR[2:0] Error Code
This 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	An attempt to write a section that is protected
0x3	CMDCOLLISION	A new write/erase command was selected when a write/erase command is already ongoing

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

Bit 0 – EEBUSY EEPROM Busy
This bit will read '1' when an EEPROM programming operation is ongoing.

16.6.9 Data

Name: DATA
Offset: 0x08
Reset: 0x000000
Property: -

Continuously updated with contents read from Flash or EEPROM. In the event of an ECC error, it will then contain the erroneous data read from the memory, comprising one, two, or possibly more bit errors.

The DATA0, DATA1 and DATA2 register triplet represents the 24-bit value DATA register. 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. The MSb of the register is in bit 23.

The data register will contain the last read value from Flash or EEPROM. For EEPROM access, only DATA[14:0] is used. For Flash access, only DATA[22:0] is used. Unused bits will read as '0'.

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

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

Bits 23:0 – DATA[23:0] Data Register
The Data register contains the data of the last memory location that has been accessed. Only the number of bits required to access the memory is used.

16.6.10 Address

Name: ADDR
Offset: 0x0C
Reset: 0x00000000
Property: -

Continuously updated with last accessed NVM address. In the event of an ECC error, its contents are frozen and will remain like this as long as any of the ECC flags in INTFLAGSB are set. Therefore, in the case of multiple sequential ECC errors, the register will contain the address of the first error.

The NVMCTRL.ADDR0, NVMCTRL.ADDR1, NVMCTRL.ADDR2 and NVMCTRL.ADDR3 represent the 32-bit value NVMCTRL.ADDR register.

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`.

The byte [31:24] (suffix 3) can be accessed at offset +`0x03`, but it never contains any data.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

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
This bit field contains the address of the last memory location accessed. Only the number of bits required to access the memory is used.

16.6.11 ECC Parity

Name: PARITY
Offset: 0x10
Reset: 0x00
Property: -

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

Bits 7:0 – PARITY[7:0] Flash Error Parity

This bit field contains the erroneous ECC parity bits read from the memory. In the event of an ECC error, its contents are frozen and will remain like this as long as any of the ECC flags in INTFLAGSB are set. Therefore, in the case of multiple sequential ECC errors, the register will contain the data for the first error.

This register is only updated by ECC errors from NVM (Fetch, Flash data or EEPROM data). This register is reset by Power-on Reset (POR) only.

16.6.12 ECC Syndrome

Name: SYNDROME
Offset: 0x11
Reset: 0x00
Property: -

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

Bits 7:0 – SYNDROME[7:0] Error Syndrome
This bit field contains the ECC syndrome calculated from the erroneous word read from the Flash or EEPROM. SYNDROME identifies which bit is in error. In the event of an ECC error, its contents are frozen and will remain like this as long as any of the ECC flags in INTFLAGSB are set. Therefore, in the case of multiple sequential ECC errors, the register will contain the data for the first error. This bit field is reset by POR only.

17. RAMCTRL - RAM Controller

17.1 Features

- Interface to Internal RAM
- Functional Safety Enhancements
  - SEC-DED ECC check of RAM
  - Redundant ECC check logic ensures no software diagnostics overhead
  - Parity and consistency check of data bus
  - Error injection for diagnostics

17.2 Overview

The RAM Controller (RAMCTRL) interfaces the data bus to internal RAM. SEC-DED ECC and bus parity logic are incorporated into the hardware for improved data integrity with error injection capabilities for latent fault detection.

The ECC logic generates and stores ECC parity data for each write operation to RAM. The ECC parity data are automatically checked using a dual lockstep checker for additional redundancy on each RAM read. Single faults are automatically corrected, while single faults and double faults are reported to the Error Controller (ERRCTRL).

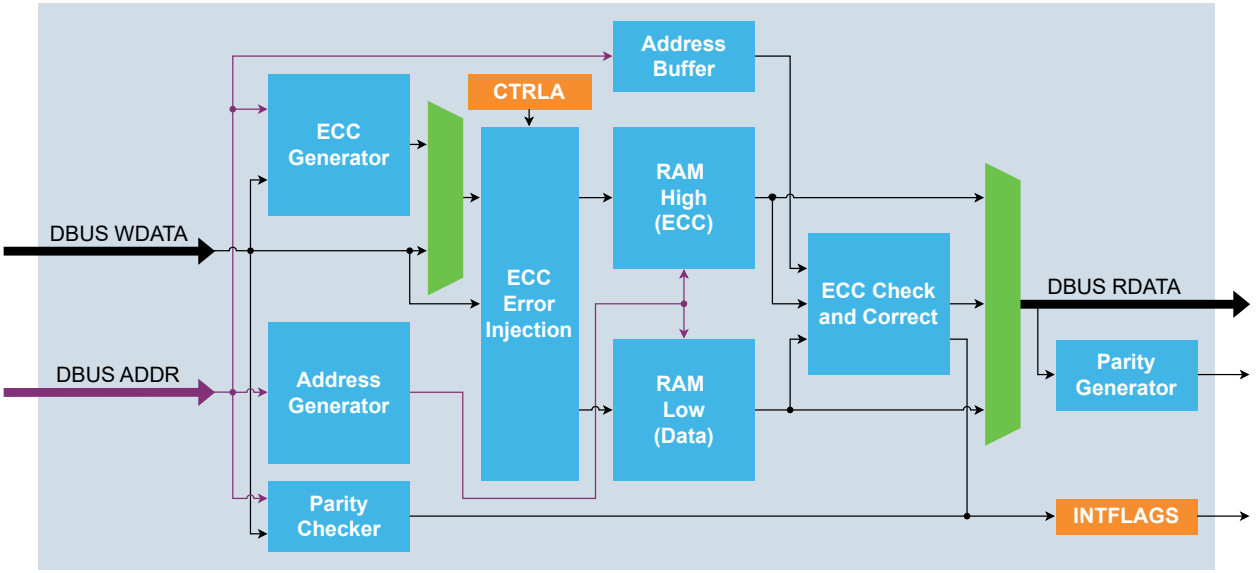
The bus checks the integrity of address, data, and control signals on every RAM data read and write using mechanisms such as parity and redundancy. Integrity errors are reported to ERRCTRL.

Both the ECC and bus parity logic support the intentional provocation of errors through an error injection register interface, allowing verification that errors can be detected as expected.

RAMCTRL is always enabled.

17.3 Block Diagram

Figure 17-1. Block Diagram



17.4 Functional Description

17.4.1 ECC Correction

The RAMCTRL's ECC system provides SEC-DED capabilities to detect and correct single-bit and double-bit errors in the RAM.

The system protects against data written to or read from the wrong RAM address. The write address is coded into the ECC parity. If the ECC parity does not match that of the read address, an error is flagged in either the ECC1 or ECC2 bit of the INTFLAGS register.

An ECC error in the address part of the ECC check word is unrecoverable and cannot be corrected. Such an error indicates that data have been written to or read from the wrong address, as the ECC2 flag in the INTFLAGS register indicates. For more information, see the *Resources Protected by ECC* section.

The data bus signals are protected through parity and redundancy. This protects against errors on the data bus signals, e.g, data being read from or written to the wrong address or reads being changed to writes. A detected error on the data bus will lead to setting one of the PARITY bits in the INTFLAGS register. Note that an erroneous write will not be discarded; the addressed location will be affected by the write, and the contents of the addressed location must be considered corrupted. The application must appropriately recover from this error. Consider the entire RAM corrupt if the parity error is in the address, as there is no way of knowing which address was written to. For more information, see the *Parity Check* section.

The location of the failing bit is found in the SYNDROME register, and the failing address is available in the ADDR register.

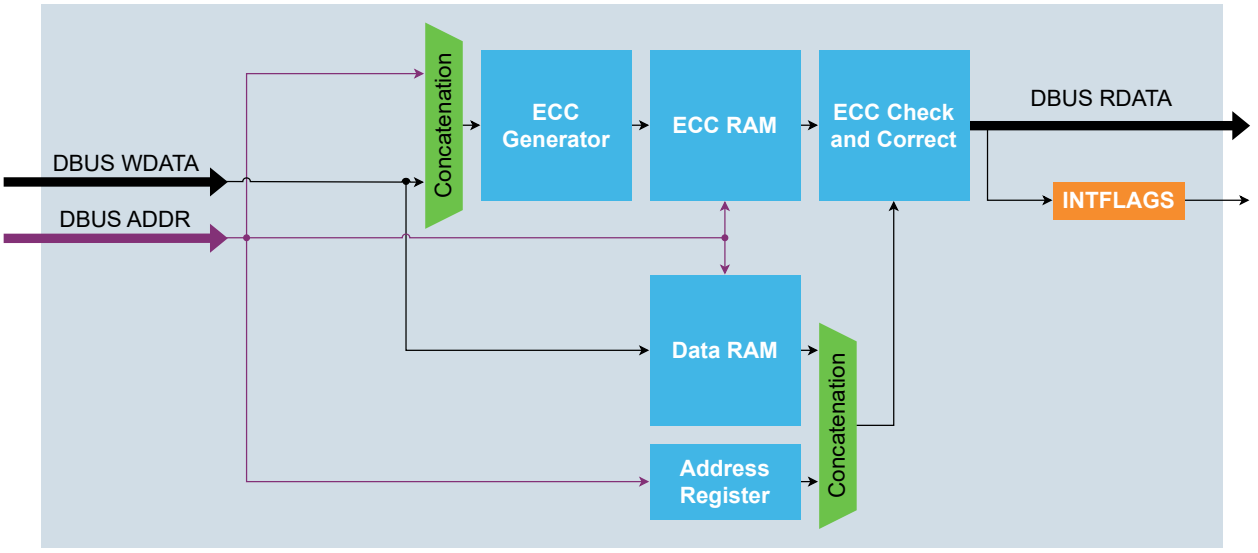
17.4.1.1 Resources Protected by ECC

ECC is generated on the concatenation of address and write data, as shown in the figure below. The *Concat* box performs the concatenation of two multi-bit inputs to a single, wider output, allowing detection of address errors in the read data, i.e., reading data from the wrong address.

Detection of an ECC error sets ECC1 or ECC2 in the INTFLAGS register.

The location of the failing bit is found in the SYNDROME register. The failing address is found in the ADDR register.

Figure 17-2. Calculating ECC on Address and Data



17.4.2 Parity Check

Before writing to RAM, the control signals, address and write data are checked for correct parity. A parity error will cause one of the PARITY bits in the INTFLAGS register to be set.

For reads from RAM, parity on the read data is generated and output on the data bus.

The data bus has additional consistency checks, such as duplicated read/write/control strobes. Any failed consistency check is flagged as a parity error causing PARITYD, PARITYA or PARITYC in the INTFLAGS register to be set.

17.4.3 Dual ECC Checkers

Critical portions of the ECC decode logic are duplicated and continuously compared. A compare error will set the COMP bit in the INTFLAGS register. The ECC encode logic in RAMCTRL is not duplicated. Correctness of written data (and ECC) can be verified, if needed, by a readback, which will pass through a replicated ECC decode logic, and any error during a write will be automatically detected and flagged when the RAM location eventually is read back.

17.4.4 Error Reporting

ECC errors occurring in the RAM are reported as follows:

Table 17-1. ECC Error Information

Memory	Information	Register
RAM Data	Single- or double-bit error	INTFLAGS
	Failing bit in ECC word	SYNDROME
	Failing address	ADDR

17.4.5 Imprecise Errors

All error sources within RAMCTRL are imprecise, meaning the error is flagged one or more cycles after it occurred. For imprecise errors, it may be impossible to know which event or which instruction triggered them. Such errors may be difficult or impossible to recover from and may need to be configured with an ERRCTRL severity of critical.

17.4.6 Error Bit Addressing

The value in the SYNDROME register indicates the source of a single-bit error, as shown in the table below:

Table 17-2. Bit Addressing of ECC Errors

SYNDROME Value	Bit Error
0-7	Bits 0-7, respectively, in RAM Data
8-23	Bits 0-15, respectively, in RAM Address
24-29	Bits 0-6, respectively, in ECC Parity data
30-255	Undefined

17.4.7 Error Injection

17.4.7.1 ECC Error Injection

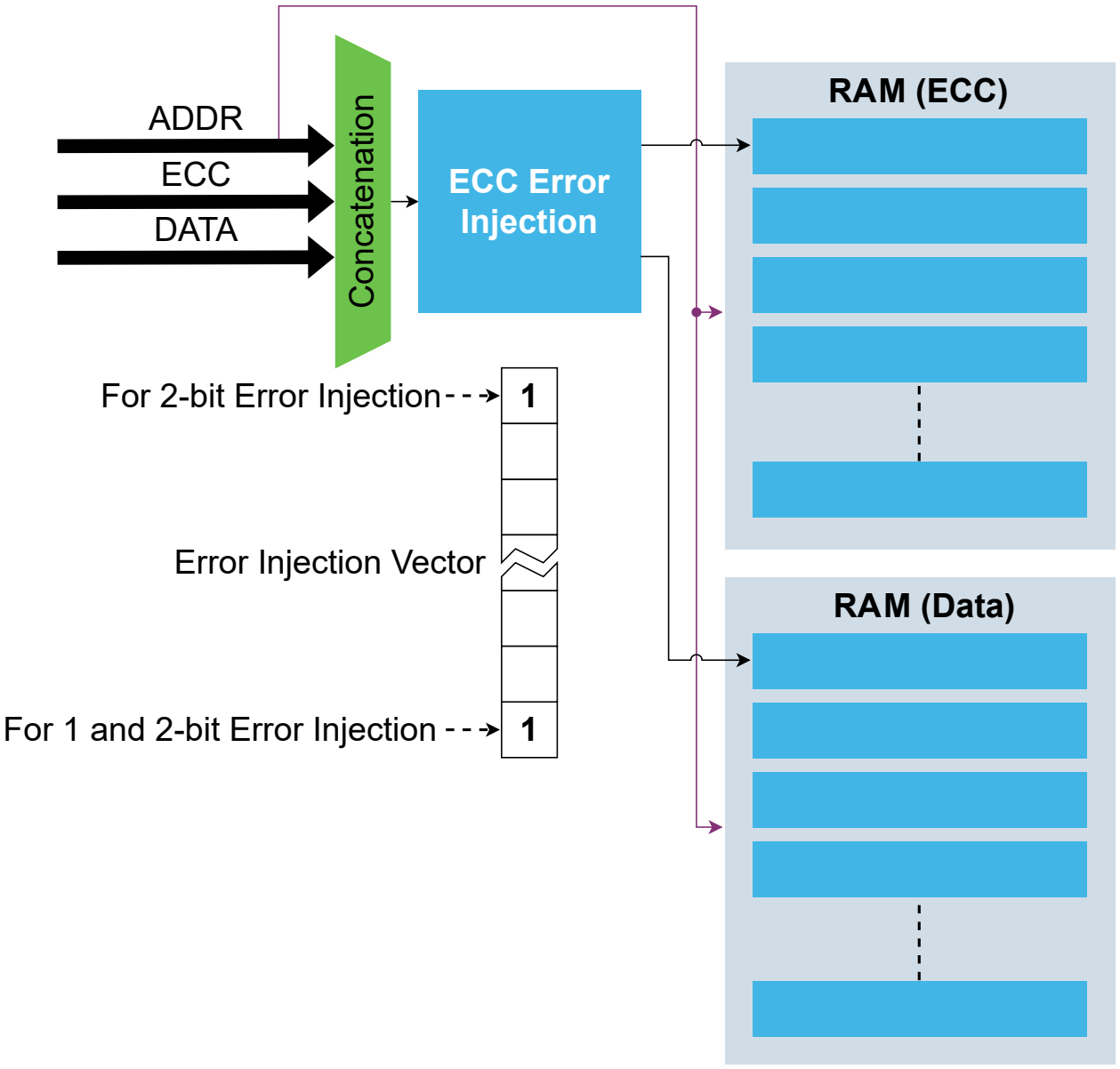
Injecting a single- or double-bit error into the data written to RAM can test the ECC logic. Subsequent RAM read will generate an ECC error, either a single-bit correction, double-bit non-correctable, or an ECC checker comparator error. These errors are flagged in INTFLAGS.

A single-bit or double-bit ECC error is injected by writing a '1' to the ECC1 bit or ECC2 bit in the CTRLA register and then performing a store instruction to an address in RAM. A subsequent load instruction from this address will result in a single-bit or double-bit ECC error being reported in the INTFLAGS. Single-bit errors are injected in the data LSB resulting in a SYNDROME value of 0. With double-bit error injection, the MSB of the ECC parity data is additionally modified. In this case, the SYNDROME value is invalid since the ECC can only detect the presence of double-bit errors, not the location. ECC error injection must be performed with global interrupts disabled to ensure that the error is not accidentally injected in the wrong data. In a system with additional active bus hosts,

such as a DMA controller, these bus hosts must be kept from accessing the RAM during the error injection sequence to protect against errors injected into accesses from unintended bus hosts.

An ECC checker comparator mismatch error injection is enabled by writing a '1' to the COMP bit in the CTRLA register. The following read access will result in a mismatch between the duplicated ECC checkers, resulting in the setting of the COMP bit in the INTFLAGS register.

Figure 17-3. Error Injection Into RAM



17.4.7.2 Parity Error Injection

Parity errors can be injected into data output on the data bus (i.e., a load instruction from the CPU) by writing a '1' to the PARITY bit in the CTRLA register.

To test the parity error on the load address, force the CPU to inject a parity error on the address bus on the next data bus access by writing a '1' to the INJPDA bit in the CPU.CTRLA register.

To test the parity error on the data to be stored in RAM, configure the CPU to inject a parity error on the data bus on the next data bus write access by writing a '1' to the INJPDD bit in the CPU.CTRLA register.

Writing a '1' to the INJPDC bit in the CPU.CTRLA register can also inject an error into the control signals.

Note: See *Error Injection* in the *Error Controller* section for more information on how to do an error injection and configure the Error Controller.

17.4.8 Interrupts

Table 17-3. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
RAMERROR	RAM Error	Generated when an error flag is set in the INTFLAGS register

17.4.9 Events

RAMCTRL does not use or generate events.

17.4.10 Sleep Mode Operation

The RAM and ECC system require a clock to function, so it does not function in sleep modes where no clock is present.

17.4.11 Configuration Change Protection

RAMCTRL has registers under Configuration Change Protection (CCP), a security mechanism to prevent unintentional changes to the RAMCTRL settings. In order to write to these registers, a certain key must be written to the CPU.CCP register first, 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 and returns a Bus Error on the data bus.

The following RAMCTRL registers are under CCP:

Table 17-4. RAMCTRL - Registers under Configuration Change Protection

Register	Key
RAMCTRL.CTRLA	IOREG
RAMCTRL.INTFLAGS	IOREG

17.5 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0					PARITY	COMP	ECC2	ECC1
0x01	INTFLAGS	7:0			PARITYD	PARITYA	PARITYC	COMP	ECC2	ECC1
0x02	ADDR	7:0	ADDR[7:0]							
		15:8	ADDR[15:8]							
0x04	SYNDROME	7:0	SYNDROME[7:0]							

17.6 Register Description

17.6.1 Control A

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

Note: There must be at least one cycle (NOP or other instruction) between writing to this register and accessing the RAM to allow the injection logic to be ready.

Bit	7	6	5	4	3	2	1	0
					PARITY	COMP	ECC2	ECC1
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – PARITY Inject Parity Error
Writing this bit to '1' will inject a data bus parity error on the data returned by the next read from RAM. This bit is automatically cleared after the error has been injected.

Value	Name	Description
0x0	DISABLE	Do not inject error
0x1	ENABLE	Inject error

Bit 2 – COMP Inject ECC Comparator Mismatch Error
Writing this bit to '1' will inject an ECC comparator mismatch error on the next read from RAM. This bit is automatically cleared after the error has been injected.

Value	Name	Description
0x0	DISABLE	Do not inject error
0x1	ENABLE	Inject error

Bit 1 – ECC2 Inject 2-bit ECC Error
Writing this bit to '1' will inject a 2-bit ECC error on the next write to RAM. The error is injected into the LSB and the MSB of the parity data. This bit is automatically cleared after the error has been injected.

Value	Name	Description
0x0	DISABLE	Do not inject error
0x1	ENABLE	Inject error

Bit 0 – ECC1 Inject 1-bit ECC Error
Writing this bit to '1' will inject a 1-bit ECC error on the next write to RAM. The error is injected into the LSB. This bit is automatically cleared after the error has been injected.

Value	Name	Description
0x0	DISABLE	Do not inject error
0x1	ENABLE	Inject error

17.6.2 Interrupt Flags

Name: INTFLAGS
Offset: 0x01
Reset: 0x00
Property: Configuration Change Protection

This register is set by hardware when an error is generated as a response to a hardware error or an error caused by error injection.

Bit	7	6	5	4	3	2	1	0
			PARITYD	PARITYA	PARITYC	COMP	ECC2	ECC1
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – PARITYD Parity Error Detected on Write Data
Set when a data bus parity error is detected on write data. Write this bit to '1' to clear it.

Bit 4 – PARITYA Parity Error Detected on Address
Set when a data bus parity error is detected on the read or write address. Write this bit to '1' to clear it.

Bit 3 – PARITYC Parity Error Detected on Control
Set when a data bus parity error is detected on control signals. Write this bit to '1' to clear it.

Bit 2 – COMP Comparator Mismatch Error Detected
Set when a mismatch between the duplicated ECC checkers is detected. Routed to ERRCTRL. Write this bit to '1' to clear it.

Bit 1 – ECC2 ECC Multibit Error Detected
Set when a 2-bit (or more) non-correctable error is detected. Routed to ERRCTRL. Write this bit to '1' to clear it.

Bit 0 – ECC1 ECC 1-bit Error Detected
Set when a 1-bit correctable error is detected. Routed to ERRCTRL. Write this bit to '1' to clear it.

17.6.3 ECC Address

Name: ADDR
Offset: 0x02
Reset: 0x0000
Property: -

In the event of an ECC error, this register is updated with the failing address. The ADDR0 and ADDR1 register pair represents the 16-bit value, ADDR.

The low byte [7:0] is accessible at the address offset.

The high byte [15:8] can be accessed at the address offset + `0x01`.

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 15:0 – ADDR[15:0] Address
The Address register contains the address of the last memory location accessed. Only the number of bits required to access the memory is used.

17.6.4 ECC Syndrome

Name: SYNDROME
Offset: 0x04
Reset: 0x00
Property: -

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

Bits 7:0 – SYNDROME[7:0] Error Syndrome
The ECC syndrome calculated from the erroneous word read from the RAM. SYNDROME identifies which bit is in error.

18. CLKCTRL - Clock Controller

18.1 Features

- All Clocks and Clock Sources Are Automatically Enabled When Requested by Peripherals
- Internal Oscillators:
  - Up to 20 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
  - 600 kHz Internal Oscillator (OSC600K)
- 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
  - Main clock output to I/O pin:
    - Selection of undivided output or output divided by 32
  - Two Clock Failure Detection mechanisms with automatic clock switching
  - Two Clock Frequency Measurement mechanisms

18.2 Overview

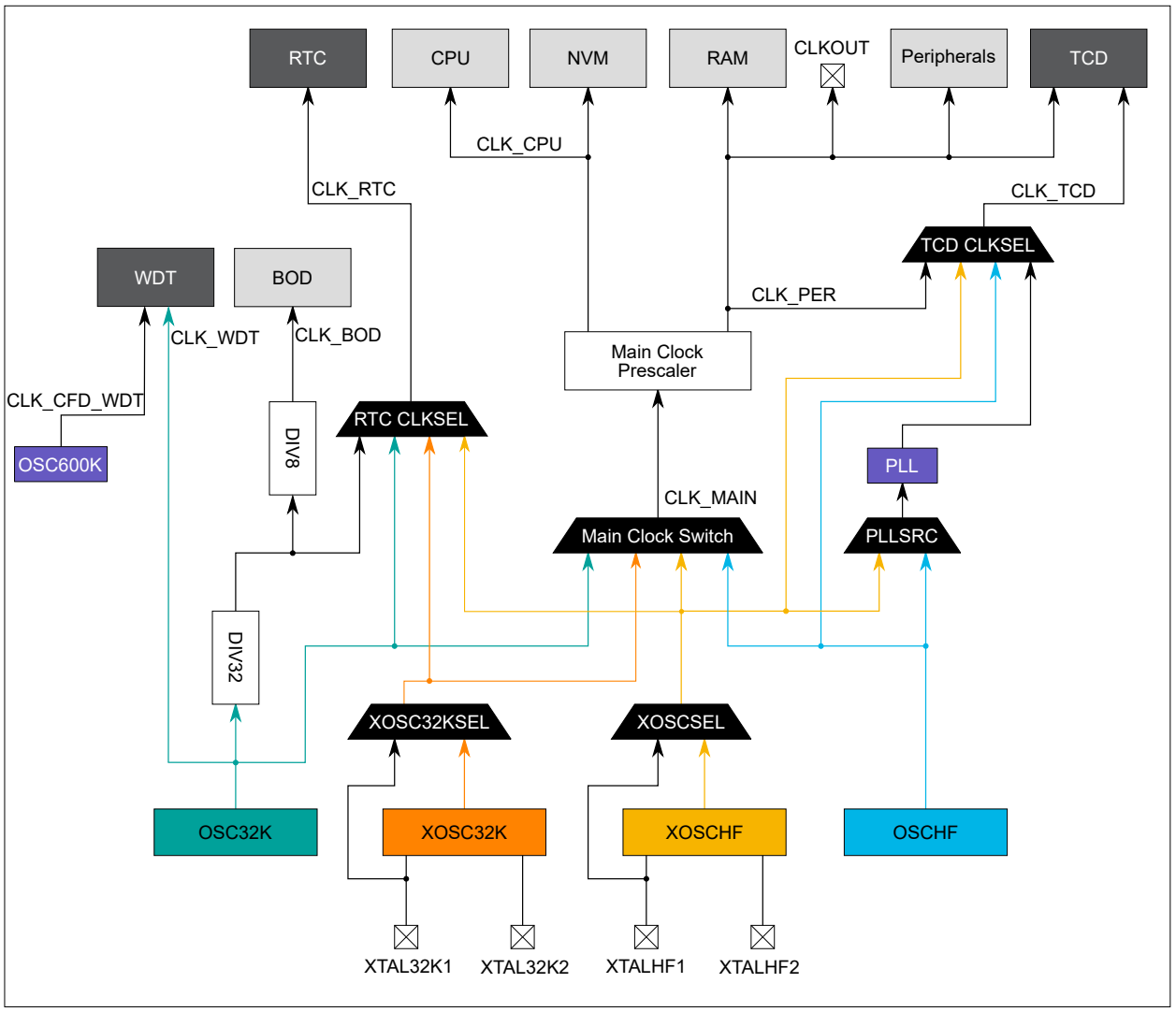
The Clock Controller (CLKCTRL) controls, distributes and prescales the clock signals from the available oscillators and supports internal and external clock sources.

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, Nonvolatile Memory (NVM), SRAM, 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.

18.2.1 Block Diagram

Figure 18-1. 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) peripheral
  - 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 the 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_CFD_WDT is used by the WDT monitor. 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 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. Change the clock source only if the peripheral is disabled.

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.

18.2.2 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.

18.3 Functional Description

18.3.1 Initialization

To initialize a clock source as the main clock, these steps need to 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.
   a. 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.
   b. Select the configured clock source as the main clock in the Clock Select (CLKSEL) bit field in CLKCTRL.MCLKCTRLA.
5. Wait for the 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.

18.3.2 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 (CCP)* section.

The Clock Failure Detection mechanism ensures safe switching to an internal clock source if possible upon clock failure when enabled. If it is not possible to switch to a safe clock source, a Machine Check Reset is generated. See also section Clock Failure Detection (CFD).

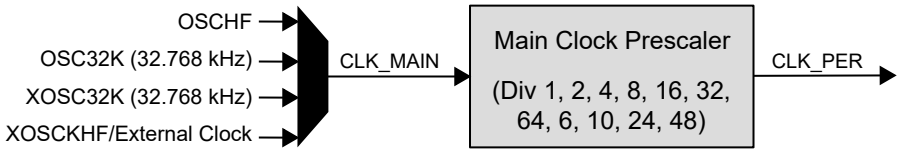
Upon the selection of an external clock source, a switch to the chosen clock source will occur only if clock edges on the external clock are detected. Until a sufficient number of clock edges are detected, the switch will not occur, and it will not be possible to change to another clock source again 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.

Note:
When selecting a 32.768 kHz oscillator as source for the main clock (either OSC32K or XOSC32K), the VMON Sleep Mode (VSLP) flag in the Interrupt Flags Register of the Sleep Controller (SLPCTRL.INTFLAGS) is set. See also section *SLPCTRL - Sleep Controller*.

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 18-2. Main Clock and Prescaler



18.3.3 Main Clock After Reset

After any Reset, the Main Clock (CLK_MAIN) is provided by the OSCHF, running at the default frequency of 4 MHz.

18.3.4 Clock Sources

All the internal clock sources are automatically enabled when requested by a peripheral. The crystal oscillators, based on an external crystal, must be enabled before they 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 running from the internal high-frequency oscillator.

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

18.3.5 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* figure in the *CLKCTRL - Clock Controller* section.

18.3.6 OSC600K - Internal 600 kHz Oscillator

This oscillator is requested and used by the Watchdog Timer (WDT) monitor. It has no configuration options.

18.3.7 Manual Tuning and Auto-Tune

Tune the output frequency of the OSCHF either manually or automatically against an external oscillator.

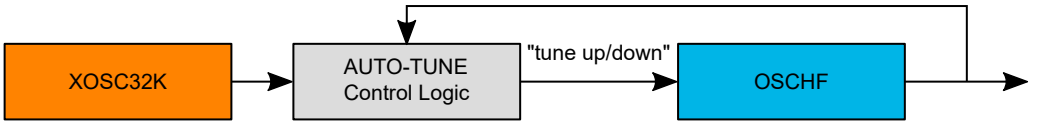
Manual Tuning

Tune the output frequency of the OSCHF up and down by writing the Oscillator Tune (TUNE) bit field in the Frequency Tune (TUNE) register. The Automatic Oscillator Tune (AUTOTUNE) bit field in the CTRLA register must remain zero.

Auto-Tune Against an External Crystal Oscillator

The OSCHF output frequency can be calibrated by automatic tuning against an external 32.768 kHz crystal oscillator. Enable auto-tune by selecting the external oscillator in the Automatic Oscillator Tune (AUTOTUNE) bit field in the CTRLA register. This will lock the TUNE register, and no manual tuning is possible. The autotune hardware periodically updates the TUNE register when AUTOTUNE is enabled.

Figure 18-3. OSCHF Auto-Tune Block Diagram



Refer to the *Electrical Characteristics* section for details.

18.3.8 TIMEBASE

Some peripherals have an internal timer that is used to determine the delay between the peripheral being enabled and the indication that it is ready to use.

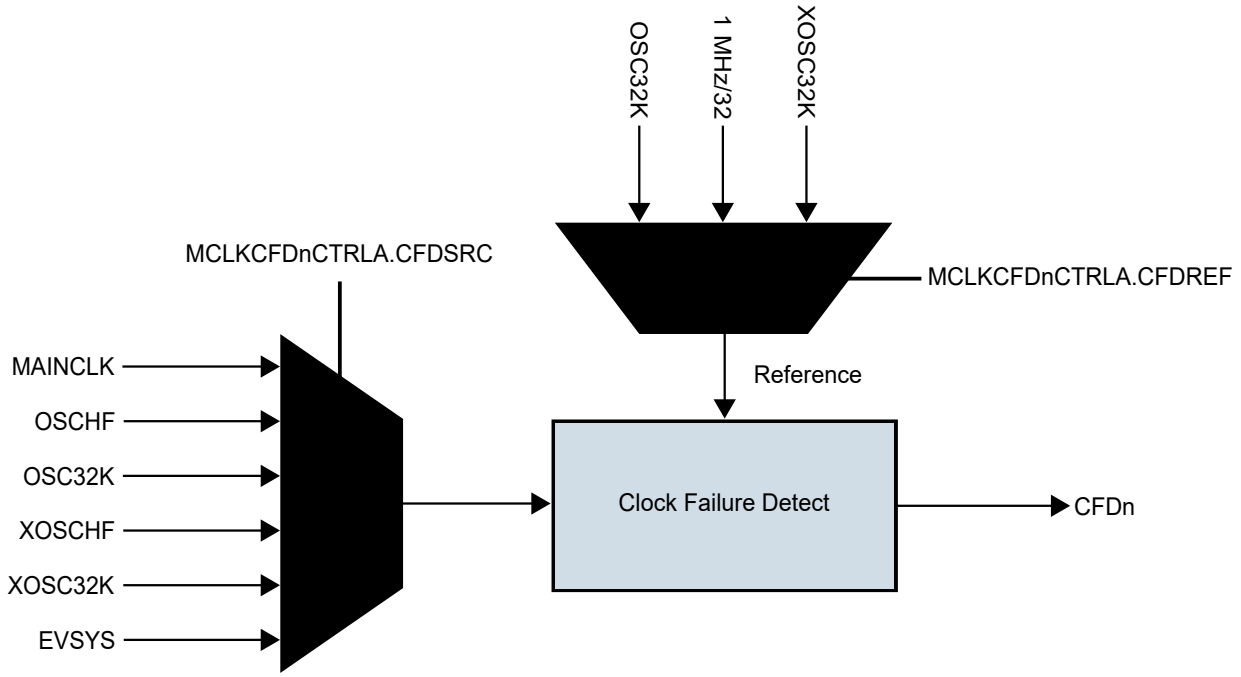
For the internal timer to function correctly, the Timebase (TIMEBASE) bit field in the Timebase (MCLKCTRL.TIMEBASE) register needs to be configured with how many cycles of the peripheral clock (CLK_PER) are equal to 1 µs. If that number is an integer, subtract one from it and write it to the TIMEBASE 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.

18.3.9 Clock Failure Detection (CFD)

18.3.9.1 CFD Operation

The CFD feature detects a failed oscillator or clock source by checking for edges on the selected oscillator/clock during a CFD period. A CFD period is 10 cycles on a reference clock, selected by Clock Failure Detection Reference (CFDREF) in the CLKCTRL.MCLKCFDnCTRLA register. In the first 8 cycles of the CFD period, edge detectors will detect edges on the monitored oscillator/clock. The 2 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 18-4. Clock Failure Detection (CFD) Block Diagram



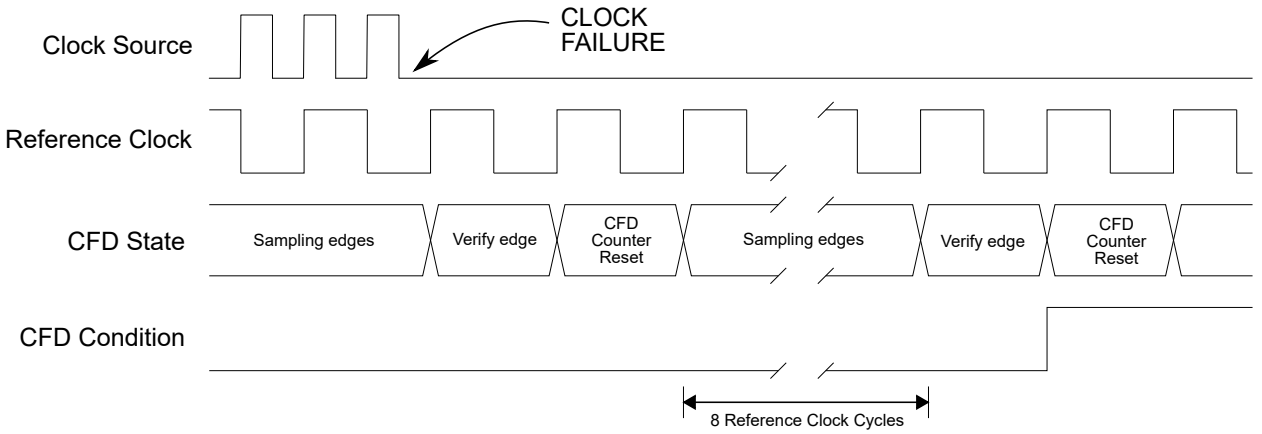
When the CFD feature is enabled, it will monitor the selected source from the Clock Failure Detection Source (CFDSRC) bit field in the CLKCTRL.MCLKCFDnCTRLA register. To avoid a CFD event, disable CFD before disabling a monitored clock, or before going to sleep, if the monitored clock stops in sleep.

If the main clock fails, everything running on it will stop. Therefore, a CFD on main clock (CFDSRC == 0 in CLKCTRL.MCLKCFDnCTRLA) is treated differently from a CFD on any other clock:

Table 18-1. CFD Source Behavior

CFD Source		
Not Main Clock	Main Clock	
	MAINCLK Source: Not XOSCHF/EXTCLK	MAINCLK Source: XOSCHF/EXTCLK
The CFDn interrupt flag in the CLKCTRL.MCLKINTFLAGS register is set, and if the interrupt is enabled, an interrupt request is issued. The Error Controller is informed.	The clock controller will request a Machine Check Reset.	The CFDn interrupt flag in the CLKCTRL.MCLKINTFLAGS register is set, and if the interrupt is enabled, an interrupt request is issued. The Error Controller is informed. The clock controller will switch to OSCHF as main clock. MCLKCTRL.CTRLB will be written to the reset value. The start-up clock source is changed back to its reset frequency. If set, the CLKOUT and CLKOUTDIV bits in CLKCTRL.MCLKCTRLA is cleared automatically

Figure 18-5. Clock Failure Detection (CFD) Timing Diagram



18.3.9.2 Condition Clearing

The CFD condition is cleared:

- After a Reset
- If the monitored source starts toggling again
- If the CFDn flag in the Main Clock Interrupt Flags (CLKCTRL.MCLKINTFLAGS) register is cleared

If it is the main clock that is being monitored and XOSCHF/EXTCLK is the main clock source, changing back to the default start-up clock (OSCHF) will make the main clock start toggling again, clearing the condition.

As long as the failure condition is met, the interrupt will trigger every ten OSC32K cycles. If these repeated interrupts are not wanted, write a '0' to the Clock Failure Detection (CFDn) interrupt enable bit in the Main Clock Interrupt Control (CLKCTRL.MCLKINTCTRL) register.

18.3.9.3 Reset or Wake-up from Sleep

Many clock sources are automatically stopped when entering sleep. If monitoring such a clock source, the monitoring must be disabled before going to sleep, otherwise a CFD event will occur. The monitoring can be re-enabled after wake-up from sleep, after the oscillator has been re-enabled and is stable.

18.3.9.4 CFD Test

The CFD can be tested by writing the corresponding bit in the CLKCTRL.MCLKCTRLC register.

The test facility works by masking all pulses from the monitored clock during one monitor interval. Therefore, the test will require between one and two monitor intervals before the CFD failure is detected, flagged and reported to the Error Controller. The reason for requiring up to two cycles is that the current monitoring cycle must be allowed to complete before a new monitoring cycle, which is ensured to have no edges on the monitored clock, can start.

There are three different modes of testing the CFD:

1. **Testing the clock failure without influencing the main clock:**
   To not influence the main clock when testing CFD, the CFDSRC bit field must be configured to a clock source different than the main clock, i.e. CFDSRC in CLKCTRL.MCLKCFD[n]CTRLA must be different from 0x00. Write the CFDn bit in CLKCTRL.MCLKCTRLC to trigger a CFD condition. The CFD condition will set the CFDn flag in the CLKCTRL.MCLKINTFLAGS register but the main clock will not change to the start-up clock source.
2. **Testing the clock failure and changing the main clock to OSCHF:**
   If the CFDSRC bit field is 0x00, such that the main clock is monitored, and XOSCHF or EXTCLK is the main clock source, writing the CFDn bit in CLKCTRL.MCLKCTRLC triggers a CFD condition.

The CFDn bit in the CLKCTRL.MCLKINTFLAGS register will be set and the main clock source will change to OSCHF.

3. **Testing the clock failure and triggering a Machine Check Reset:**
   If the CFDSRC bit field is `0x00`, such that the main clock is monitored, and XOSCHF or EXTCLK is NOT the main clock source, writing the CFDn bit in CLKCTRL.MCLKCTRLC triggers a CFD condition and a Machine Check Reset will be requested.

When a CFD error is injected on the main clock, the main clock is stopped in a safe way, i.e. not generating any glitches or violations of the required clock properties. This will emulate a real error, and also allow the system to continue in a safe way after the failure has been detected.

18.3.10 Clock Frequency Measurement (CFM)

The Clock Frequency Measure (CFM) feature allows measuring the frequency of a clock source relative to a reference clock.

18.3.10.1 CFM Operation

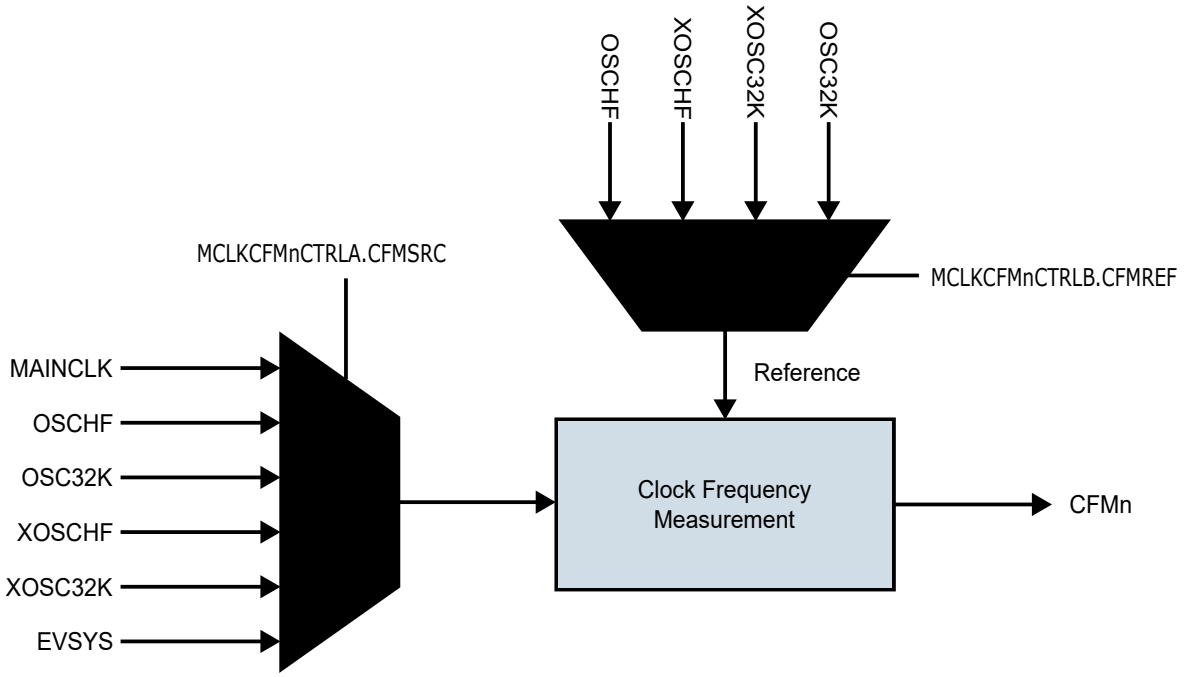
The clock source to be measured is chosen in the CFMSRC bit field in the CLKCTRL.MCLKCFMnCTRLA register, and the reference clock is chosen in the CFMREF bit field in the CLKCTRL.MCLKCFMnCTRLB register. The CFM works by incrementing a counter VALUE in the CLKCTRL.MCLKCFMnVALUE register on every clock period of main clock. The measurement is done over a span of REFNUM reference clock cycles. Once REFNUM reference clock cycles have passed, the number in VALUE is compared to the expected value range given by WINLT in the CLKCTRL.MCLKCFMnWINLT register and WINHT in the CLKCTRL.MCLKCFMnWINHT register. If VALUE is not within the configured window, the frequency is wrong and the Error Controller is notified. If the TYPE bit field in the CLKCTRL.MCLKCFMnCTRLA register is configured to CONTINUOUS mode, this measurement sequence is continuously repeated. Otherwise, only a single measurement period is performed.

If the reference clock is stopped, the measurement period will never complete. This can be detected by observing the reference clock with a CFD.

If VALUE reaches its maximum value, it will stop counting, regardless of how many reference clock cycles has passed. To ensure reliable measurements, REFNUM must be configured to a low enough value such that VALUE never reaches its maximum value.

The CFM will detect if the source clock has stopped (i.e. no transition within the measurement interval) and reports this as a CFM error by setting the CFMDn and CFMn bits in the CLKCTRL.MCLKINTFLAGS register.

Figure 18-6. Clock Frequency Measure (CFM) Block Diagram



18.3.10.2 Condition Clearing

In case of a CFM error, the CFMn flag in the CLKCTRL.MCLKINTFLAGS register will be set. The CFMn flag can be cleared by writing a '1' to it.

The CFMDn flag in CLKCTRL.MCLKINTFLAGS is set when a measurement period in CFM channel n has completed. This is useful in ONE-SHOT measurements.

18.3.10.3 CFM Test

The CFM can be efficiently tested by the following software process:

1. Stop the CFM by writing '0' to the MEN bit in the CLKCTRL.MCLKCFMnCTRLA register.
2. Select the fastest possible reference clock by writing the CFMREF bit field in the CLKCTRL.MCLKCFMnCTRLB register to the fastest of OSCHF or XOSCHF, and write the REFCEN bit to '1'.
3. Write the value in REFNUM to the value 8, so that comparison completes after 8 reference clock cycles. The expected value of the CLKCTRL.MCLKCFMnVALUE register will then be 8 ± 2, i.e. in the range 6 to 10.
4. Set the value in the CLKCTRL.MCLKCFMnWINLT register to the value 5.
5. Set the value in CLKCTRL.MCLKCFMnWINHT register to the value 11.
6. Start a measurement by configuring CLKCTRL.MCLKCFMnCTRLA to:
   a. ONESHOT in the TYPE bit field.
   b. The same source as selected in CLKCTRL.MCLKCFMnCTRLB in the CFMSRC bit field.
   c. ENABLE in the MEN bit.
7. Observe that no clock failure is detected.
8. Repeat this process steps 1-6 but with selecting WINLT=2 and WINHT=5 so that a window is configured below the measurement, and observe that a clock measurement failure is detected.
9. Repeat this process steps 1-6 but with selecting WINLT=12 and WINHT=15 so that a window is configured above the measurement, and observe that a clock measurement failure is detected.

18.3.11 Typical Configurations for CFD and CFM

Table 18-2. Typical Configurations

Main Clock	Reference Clock	CFD0		CFD1		CFM0		CFM1	
		SRC	REF	SRC	REF	SRC	REF	SRC	REF
OSCHF	XOSC32K	CLK_MAIN	OSC32K	OSC32K	XOSC32K	OSCHF	XOSC32K	-	-
OSCHF	XOSC32K	CLK_MAIN	OSC32K	OSC32K	1 MHz/32	XOSCHF	OSCHF	OSC32K	XOSCHF
OSCHF,PLL	XOSC32K	CLK_MAIN	OSC32K	OSC32K	XOSC32K	OSCHF	XOSC32K	PLL	XOSC32K
XOSCHF	XOSC32K	CLK_MAIN	OSC32K	OSC32K	1 MHz/32	OSCHF	XOSC32K	OSC32K	XOSC32K

18.3.12 Interrupts

Table 18-3. Available Interrupt Vectors and Sources

Vector Name	Source Name	Condition	Dependency
CLKCTRL_INT	CFD0	Failure of the monitored clock	-
	CFD1		-
	CFM0	The clock frequency is outside the expected frequency range	-
	CFM1		-
	CFMD0	A clock frequency measurement is completed	-
	CFMD1		-

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags register of the peripheral (CLKCTRL.MCLKINTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (CLKCTRL.MCLKINTCTRL).

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 MCLKINTFLAGS register(s) for details on how to clear interrupt flags.

18.3.13 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 mode and Idle sleep mode. 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 *SLPCTRL - Sleep Controller* section for more details on sleep mode operation.

Many clock sources are automatically stopped when entering sleep. If monitoring such a clock source, the monitoring must be disabled before going to sleep, otherwise a CFD or CFM event will occur. The monitoring can be re-enabled after wake-up from sleep, after the oscillator has been re-enabled and is stable.

18.3.14 Configuration Change Protection (CCP)

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a given 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 it unchanged.

The following registers are under CCP:

Table 18-4. CLKCTRL - Registers Under Configuration Change Protection

Register	Key
CLKCTRL.MCLKCTRLA	IOREG
CLKCTRL.MCLKCTRLB	IOREG
CLKCTRL.MCLKCTRLC	IOREG
CLKCTRL.MCLKINTCTRL	IOREG
CLKCTRL.MCLKINTFLAGS	IOREG
CLKCTRL.MCLKTIMEBASE	IOREG
CLKCTRL.MCLKCFD0CTRLA	IOREG
CLKCTRL.MCLKCFD1CTRLA	IOREG
CLKCTRL.MCLKCFM0CTRLA	IOREG
CLKCTRL.MCLKCFM0CTRLB	IOREG
CLKCTRL.MCLKCFM1CTRLA	IOREG
CLKCTRL.MCLKCFM1CTRLB	IOREG
CLKCTRL.OSCHFCTRLA	IOREG
CLKCTRL.PLLCTRLA	IOREG
CLKCTRL.OSC32KCTRLA	IOREG
CLKCTRL.XOSC32KCTRLA	IOREG
CLKCTRL.XOSCHFCTRLA	IOREG

18.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	MCLKCTRLA	7:0	CLKOUT	CLKOUTDIV			CLKSEL[3:0]			
0x01	MCLKCTRLB	7:0				PDIV[3:0]				PEN
0x02	MCLKCTRLC	7:0							CFD1	CFD0
0x03	MCLKINTCTRL	7:0			CFMD1	CFMD0	CFM1	CFM0	CFD1	CFD0
0x04	MCLKINTFLAGS	7:0			CFMD1	CFMD0	CFM1	CFM0	CFD1	CFD0
0x05	MCLKSTATUS	7:0			PLLS	EXTS	XOSC32KS	OSC32KS	OSCHFS	SOSC
0x06	MCLKTIMEBASE	7:0				TIMEBASE[4:0]				
0x07	Reserved									
0x08	MCLKCFD0CTRLA	7:0			CFDSRC[2:0]			CFDREF[1:0]		CFDEN
0x09	MCLKCFD1CTRLA	7:0			CFDSRC[2:0]			CFDREF[1:0]		CFDEN
0x0A ... 0x0F	Reserved									
0x10	MCLKCFM0VALUE	7:0	VALUE[7:0]							
		15:8	VALUE[15:8]							
0x12	MCLKCFM0WINLT	7:0	WINLT[7:0]							
		15:8	WINLT[15:8]							
0x14	MCLKCFM0WINHT	7:0	WINHT[7:0]							
		15:8	WINHT[15:8]							
0x16	MCLKCFM0REFNUM	7:0	REFNUM[7:0]							
		15:8	REFNUM[15:8]							
0x18	MCLKCFM0CTRLA	7:0				TYPE	CFMSRC[2:0]			MEN
0x19	MCLKCFM0CTRLB	7:0						CFMREF[1:0]		REFCEN
0x1A ... 0x1F	Reserved									
0x20	MCLKCFM1VALUE	7:0	VALUE[7:0]							
		15:8	VALUE[15:8]							
0x22	MCLKCFM1WINLT	7:0	WINLT[7:0]							
		15:8	WINLT[15:8]							
0x24	MCLKCFM1WINHT	7:0	WINHT[7:0]							
		15:8	WINHT[15:8]							
0x26	MCLKCFM1REFNUM	7:0	REFNUM[7:0]							
		15:8	REFNUM[15:8]							
0x28	MCLKCFM1CTRLA	7:0				TYPE	CFMSRC[2:0]			MEN
0x29	MCLKCFM1CTRLB	7:0						CFMREF[1:0]		REFCEN
0x2A ... 0x3F	Reserved									
0x40	OSCHFCTRLA	7:0	RUNSTDBY		FRQSEL[3:0]					AUTOTUNE
0x41	OSCHFTUNE	7:0	TUNE[7:0]							
0x42 ... 0x47	Reserved									
0x48	PLLCTRLA	7:0	RUNSTDBY	SOURCE					MULFAC[1:0]	
0x49 ... 0x4F	Reserved									
0x50	OSC32KCTRLA	7:0	RUNSTDBY							
0x51 ... 0x53	Reserved									
0x54	XOSC32KCTRLA	7:0	RUNSTDBY		CSUT[1:0]			SEL	LPMODE	ENABLE
0x55 ... 0x57	Reserved									
0x58	XOSCHFCTRLA	7:0	RUNSTDBY		CSUTHF[1:0]		FRQRANGE[1:0]		SELHF	ENABLE

18.5 Register Description

18.5.1 Main Clock Control A

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

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

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

Bit 6 – CLKOUTDIV Divide Main Clock Out
Changing prescaler may result in a glitch on the output pin. This bit is cleared by hardware on a CFD event if the failing clock is the system clock.

Value	Description
0	Divide by 1
1	Divide by 32

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

18.5.2 Main Clock Control B

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

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

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	DIV2	Divide by 2
0x1	DIV4	Divide by 4
0x2	DIV8	Divide by 8
0x3	DIV16	Divide by 16
0x4	DIV32	Divide by 32
0x5	DIV64	Divide by 64
0x8	DIV6	Divide by 6
0x9	DIV10	Divide by 10
0xA	DIV12	Divide by 12
0xB	DIV24	Divide by 24
0xC	DIV48	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

18.5.3 Main Clock Control C

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

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

Bit	7	6	5	4	3	2	1	0
							CFD1	CFD0
Access							R/W	R/W
Reset							0	0

Bit 1 – CFD1 Inject error in CFD1
Control error injection. Automatically cleared by HW when error has been injected.

Value	Name	Description
0x0	DISABLE	Do not inject error
0x1	ENABLE	Inject error

Bit 0 – CFD0 Inject error in CFD0
Control error injection. Automatically cleared by HW when error has been injected.

Value	Name	Description
0x0	DISABLE	Do not inject error
0x1	ENABLE	Inject error

18.5.4 Main Clock Interrupt Control

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

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

Bit	7	6	5	4	3	2	1	0
			CFMD1	CFMD0	CFM1	CFM0	CFD1	CFD0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – CFMD1 Clock Frequency Measurement Done Interrupt Enable
This bit controls whether the CFMD1 interrupt is enabled or not.

Value	Description
0	The interrupt is disabled
1	The interrupt is enabled

Bit 4 – CFMD0 Clock Frequency Measurement Done Interrupt Enable
This bit controls whether the CFMD0 interrupt is enabled or not.

Value	Description
0	The interrupt is disabled
1	The interrupt is enabled

Bit 3 – CFM1 Clock Frequency Measurement Failure Interrupt Enable
This bit controls whether the CFM1 interrupt is enabled or not.

Value	Description
0	The interrupt is disabled
1	The interrupt is enabled

Bit 2 – CFM0 Clock Frequency Measurement Failure Interrupt Enable
This bit controls whether the CFM0 interrupt is enabled or not.

Value	Description
0	The interrupt is disabled
1	The interrupt is enabled

Bit 1 – CFD1 Clock Failure Detection Interrupt Enable
This bit controls whether the CFD1 interrupt is enabled or not.

Value	Description
0	The interrupt is disabled
1	The interrupt is enabled

Bit 0 – CFD0 Clock Failure Detection Interrupt Enable
This bit controls whether the CFD0 interrupt is enabled or not.

Value	Description
0	The interrupt is disabled
1	The interrupt is enabled

18.5.5 Main Clock Interrupt Flags

Name: MCLKINTFLAGS
Offset: 0x04
Reset: 0x0
Property: Configuration Change Protection

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

Bit	7	6	5	4	3	2	1	0
			CFMD1	CFMD0	CFM1	CFM0	CFD1	CFD0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – CFMD1 Clock Frequency Measurement Done CFM1
Set when a measurement is done for CFM1.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the interrupt flag.

Bit 4 – CFMD0 Clock Frequency Measurement Done CFM0
Set when a measurement is done for CFM0.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the interrupt flag.

Bit 3 – CFM1 Clock Frequency Measurement Error Detected CFM1
Clock measurement failure on CFM1.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the interrupt flag.

Bit 2 – CFM0 Clock Frequency Measurement Error Detected CFM0
Clock measurement failure on CFM0.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the interrupt flag.

Bit 1 – CFD1 Clock Failure Detected CFD1
Clock failure on CFD1.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the interrupt flag.

Bit 0 – CFD0 Clock Failure Detected CFD0
Clock failure on CFD0.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the interrupt flag.

18.5.6 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'. 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'. 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'. 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'. 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

Value	Description
1	The clock source for CLK_MAIN is undergoing a switch and will change as soon as the new source is stable

18.5.7 Main Clock Timebase

Name: MCLKTIMEBASE
Offset: 0x06
Reset: 0x00
Property: Configuration Change Protection

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

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

Bits 4:0 – TIMEBASE[4:0] Timebase
This bit field selects the number of CLK_PER cycles to get a period equal to or larger than 1 µs, used for timing internal delays such as ADC start-up time.

18.5.8 Main Clock Clock Failure Detect n Control A

Name: MCLKCFDnCTRLA
Offset: 0x08 + n*0x01 [n=0..1]
Reset: 0x00
Property: Configuration Change Protection

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

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

Bits 5:3 – CFDSRC[2:0] CFD Source
Selects the clock to monitor with the CFD channel.

Value	Name	Description
0x0	MCLK	Main Clock
0x1	OSCHF	High frequency internal oscillator
0x2	OSC32K	32.768 kHz internal oscillator
0x3	XOSCHF	High frequency external oscillator
0x4	XOSC32K	32.768 kHz external oscillator
0x5-0x6	-	Reserved
0x7	EVSYS	Event system channel input

Bits 2:1 – CFDREF[1:0] Clock Failure Detect Reference
Reference clock for CFD channel.

Value	Name	Description
0x0	OSC32K	32.768 kHz internal oscillator
0x1	ONEDIV32	1 MHz from OSCHF divided by 32
0x2	XOSC32K	32.768 kHz external crystal oscillator
Other	-	Reserved

Bit 0 – CFDEN Clock Failure Detect Enable
Enable CFD channel.

Value	Name	Description
0x0	DISABLED	CFD disabled
0x1	ENABLED	CFD enabled

18.5.9 Main Clock Clock Frequency Measure n Value

Name: MCLKCFMnVALUE
Offset: 0x10 + n*0x10 [n=0..1]
Reset: 0x00
Property: -

The number of measure clock cycles. This register is cleared by HW at the start of a measurement period. The CLKCTRL.MCLKCFMnVALUEL and CLKCTRL.MCLKCFMnVALUEH registers represent the 16-bit value, CLKCTRL.MCLKCFMnVALUE. 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. The VALUE register is NOT buffered by a TEMP register to ensure a consistent readout. The MSB of the VALUE register is bit 15.

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

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

Bits 15:8 – VALUE[15:8] CFM Value High
These bits hold the MSB of the 16-bit Value register.

Bits 7:0 – VALUE[7:0] CFM Value Low
These bits hold the LSB of the 16-bit Value register.

18.5.10 Main Clock Clock Frequency Measure n Window Low Threshold

Name: MCLKCFMnWINLT
Offset: 0x12 + n*0x10 [n=0..1]
Reset: 0x00
Property: -

The low threshold in the window of acceptable CFM measure values. The CLKCTRL.MCLKCFMnWINLTL and CLKCTRL.MCLKCFMnWINLTH registers represent the 16-bit value, CLKCTRL.MCLKCFMnWINLT. 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. The WINLT register is NOT buffered by a TEMP register to ensure a consistent readout. The MSB of the WINLT register is bit 15. This register is locked when MCLKCFMnCTRLA.MEN = '1' and an attempted write when locked will generate a bus error.

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] CFM Window Low Threshold
These bits hold the MSB of the 16-bit WINLT register.

Bits 7:0 – WINLT[7:0] CFM Window Low Threshold
These bits hold the LSB of the 16-bit WINLT register.

18.5.11 Main Clock Clock Frequency Measure n Window High Threshold

Name: MCLKCFMnWINHT
Offset: 0x14 + n*0x10 [n=0..1]
Reset: 0x00
Property: -

The high threshold in the window of acceptable CFM measure values. The CLKCTRL.MCLKCFMnWINHTL and CLKCTRL.MCLKCFMnWINHTH registers represent the 16-bit value, CLKCTRL.MCLKCFMnWINHT. 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. The WINHT register is NOT buffered by a TEMP register to ensure a consistent readout. The MSB of the WINHT register is bit 15. This register is locked when MCLKCFMnCTRLA.MEN = '1' and an attempted write when locked will generate a bus error.

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] CFM Window High Threshold
These bits hold the MSB of the 16-bit WINHT register.

Bits 7:0 – WINHT[7:0] CFM Window High Threshold
These bits hold the LSB of the 16-bit WINHT register.

18.5.12 Main Clock Clock Frequency Measure n Reference Clock Cycles

Name: MCLKCFMnREFNUM
Offset: 0x16 + n*0x10 [n=0..1]
Reset: 0x00
Property: -

Selects the number of reference clock cycles to be used for counting measurement clock cycles. The CLKCTRL.MCLKCFMnREFNUML and CLKCTRL.MCLKCFMnREFNUMH registers represent the 16-bit value, CLKCTRL.MCLKCFMnREFNUM. 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. The REFNUM register is NOT buffered by a TEMP register to ensure a consistent readout. The MSB of the REFNUM register is bit 15. This register is locked when MCLKCFMnCTRLA.MEN = '1' and an attempted write when locked will generate a bus error.

Bit	15	14	13	12	11	10	9	8
	REFNUM[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
	REFNUM[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 – REFNUM[15:8] CFM Reference Clock Measurement Cycles High
These bits hold the MSB of the 16-bit REFNUM register.

Bits 7:0 – REFNUM[7:0] CFM Reference Clock Measurement Cycles Low
These bits hold the LSB of the 16-bit REFNUM register.

18.5.13 Main Clock Clock Frequency Measure n Control A

Name: MCLKCFMnCTRLA
Offset: 0x18 + n*0x10 [n=0..1]
Reset: 0x00
Property: Configuration Change Protection

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

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

Bit 4 – TYPE Measurement Type
Controls whether the measurement will be done as a single (one-shot) measurement, or a continuous measurement.

Value	Name	Description
0x0	ONESHOT	One-shot measurement
0x1	CONTINUOUS	Continuous measurement

Bits 3:1 – CFMSRC[2:0] CFM Measure Clock Source
Selects the source for the clock to be measured by CFM.

Value	Name	Description
0x0	CLK_MAIN	Main Clock
0x1	OSCHF	High frequency internal oscillator
0x2	OSC32K	32.768 kHz internal oscillator
0x3	XOSCHF	High frequency external oscillator
0x4	XOSC32K	32.768 kHz external oscillator
0x5	EVSYS	Event system channel input
Other	-	Reserved

Bit 0 – MEN Measurement Enable
Controls the frequency measurement process.

Value	Name	Description
0x0	DISABLE	Frequency measurement disabled
0x1	ENABLE	Frequency measurement enabled

18.5.14 Main Clock Clock Frequency Measure n Control B

Name: MCLKCFMnCTRLB
Offset: 0x19 + n*0x10 [n=0..1]
Reset: 0x00
Property: Configuration Change Protection

Attempting to write to this register without following the appropriate CCP unlock sequence will return a bus error.

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

Bits 2:1 – CFMREF[1:0] CFM Reference Source
Selects the source for the reference clock for CFM.

Value	Name	Description
0x0	OSCHF	High frequency internal oscillator
0x1	XOSCHF	High frequency external oscillator
0x2	XOSC32K	32.768 kHz external oscillator
0x3	OSC32K	32.768 kHz internal oscillator

Bit 0 – REFCEN Reference Clock enable
Controls the reference clock.

Value	Name	Description
0x0	DISABLE	Reference clock disabled
0x1	ENABLE	Reference clock enabled

18.5.15 Internal High-Frequency Oscillator Control A

Name: OSCHFCTRLA
Offset: 0x40
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 oscillator (OSCHF).

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

18.5.16 Internal High-Frequency Oscillator Frequency Tune

Name: OSCHFTUNE
Offset: 0x41
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 bits 6 and 7 has no effect, as bit 5 will be mirrored to bits 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. When AUTOTUNE is disabled, it takes up to three µs and three Main Clock cycles before this bit field is updated with the latest tune value from the auto-tune operation.

18.5.17 PLL Control A

Name: PLLCTRLA
Offset: 0x48
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
0x0	OSCHF	The high-frequency internal oscillator as a PLL source
0x1	XOSCHF	High-frequency external clock or external high-frequency oscillator as a 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

18.5.18 Internal 32.768 kHz Oscillator Control A

Name: OSC32KCTRLA
Offset: 0x50
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.

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 mode, Idle sleep mode, Standby sleep mode and Power-Down sleep mode (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.

18.5.19 External 32.768 kHz Crystal Oscillator Control A

Name: XOSC32KCTRLA
Offset: 0x54
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 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 mode and Idle sleep mode (1)
1	The XOSC32K oscillator will always run in Active mode, Idle sleep mode, Standby sleep mode and Power-Down sleep mode (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 32.768 kHz Crystal Oscillator (XOSC32K) start-up time when the Source Select (SEL) bit 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	Name	Description
0	XTAL	External crystal connected to the XTAL32K1 and XTAL32K2 pins
1	EXTCLK	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.

Note: Enabling the Low-Power mode can increase the crystal's start-up time. Mitigate this by altering the crystal implementation to reduce serial resistance and overall capacitance or disabling the Low-Power mode.

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.

Value	Description
0	The XOSC32K oscillator is disabled
1	The XOSC32K oscillator is enabled and overrides ordinary port operation for the respective oscillator pins

18.5.20 External High-Frequency Oscillator Control A

Name: XOSCHFCTRLA
Offset: 0x58
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 higher than the maximum supported CLK_CPU frequency is used, and it is 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	XTAL	External Crystal on the XTALHF1 and XTALHF2 pins
1	EXTCLK	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

19. SLPCTRL - Sleep Controller

19.1 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
- Control of Voltage Regulator Monitor (VMON)
  - Fault Injection into VMON
  - Control of VMON in Sleep
- Functional Safety Features Included to Protect Against Inadvertent Sleep Entry

19.2 Overview

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

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

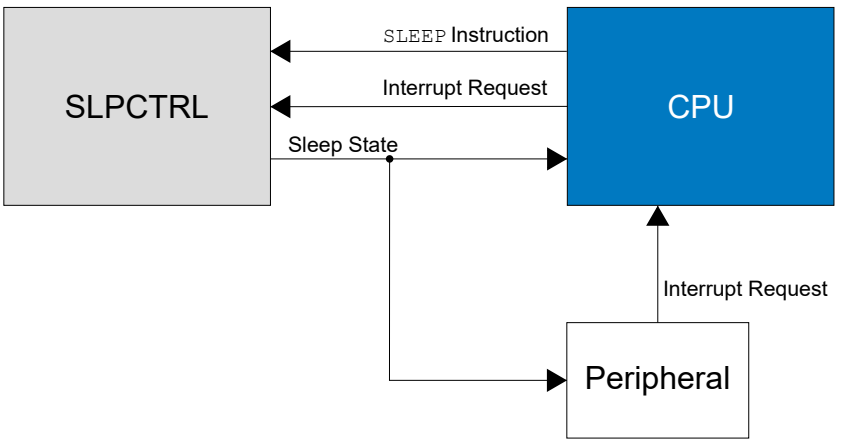
All sleep modes are available and can be entered from the Active mode. In Active mode, the CPU is executing application code. When the device enters one of the sleep modes, the program execution stops. The application code decides which sleep mode to enter and when.

Interrupts will 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.

19.2.1 Block Diagram

Figure 19-1. SLPCTRL Block Diagram



19.3 Functional Description

19.3.1 Initialization

To enter a sleep mode, follow these steps:

1. Configure and enable the interrupts that can wake the device from sleep mode.
2. Enable global interrupts by writing the Global Interrupt Enable (I) bit in the CPU Status Register (CPU.SREG) register to '1'.
3. Select the sleep mode to enter by writing to the Sleep Mode (SMODE) bit field in the Control A (SLPCTRL.CTRLA) register.
4. Enable the Sleep Controller by writing to the Enable (SEN) bit in the Control A (SLPCTRL.CTRLA) register.
5. Execute the SLEEP instruction to make the device enter the selected sleep mode.

Note: If executing a SLEEP instruction when the Sleep Enable (SEN) bit in the Control A (CTRLA) register is '0' (CTRLA.SEN = '0') or the Global Interrupt Enable (I) bit in the CPU Status Register (CPU.SREG) register is '0' (CPU.SREG.I = '0'), the SLEEP instruction is disregarded - and the Sleep Error (SERR) flag in the Interrupt Flags (SLPCTRL.INTFLAGS) register is set.

WARNING When executing the SLEEP instruction, at least one interrupt that can wake the device must be enabled. If no interrupts are enabled, the device cannot return from sleep mode, and only a Reset will allow the device to resume ordinary operation.

19.3.2 Operation

19.3.2.1 Sleep Modes

In Active mode, the device runs as ordinary with all systems active. In addition to Active mode, there are three different sleep modes with decreasing power consumption and functionality.

Table 19-1. Sleep Modes

Mode	Description	Power Consumption
Active	Normal mode, all systems are active	Highest power consumption
Idle	The CPU stops executing code, resulting in reduced power consumption. All peripherals are running. All interrupt sources can wake the device.	
Standby	In this mode, each clock source or peripheral can individually be enabled or disabled using its running in Standby sleep mode, enabled by writing its RUNSTDBY bit to '1'. The power consumption depends on the number of enabled peripherals and clocks. A subset of interrupt sources can wake the device (1).	
Power-Down	All high-frequency clocks are stopped. Only the WDT and the PIT (component of the RTC) are active, resulting in the lowest power consumption. The only wake-up sources are the pin change interrupt, CCL interrupt (if filter and edge detect are disabled), and TWI address match. A subset of the peripherals are running, and a subset of interrupt sources can wake the device(1).	Lowest power consumption

Note:

1. Refer to the *Sleep Mode Activity* tables for further information.

Refer to the *Wake-up Time* section for information on how the three sleep modes affect the wake-up time.

Table 19-2. Sleep Mode Activity Overview for Peripherals

Peripheral	Active in Sleep Mode		
	Idle	Standby	Power-Down
CPU			
RTC	X	X(1,2)	X(2)
WDT	X	X	X
BOD	X	X	X
EVSYS	X	X	X
CCL	X	X(1)	
AC			
ADC			
DAC			
ZCD			
TCA			
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 written to '1'.
2. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY bit to be '1'. In Power-Down sleep mode, only the PIT functionality is available.

Table 19-3. Sleep Mode Activity Overview for Clock Sources

Clock Source	Active in Sleep Mode		
	Idle	Standby	Power-Down
Main clock source	X	X(1)	
RTC clock source	X	X(1,2)	X(2)
WDT oscillator	X	X	X
BOD oscillator(3)	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 written to '1'.
2. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY bit to be '1'. In Power-Down sleep mode, only the PIT functionality is available.
3. The Sampled mode only.

Table 19-4. Sleep Mode Wake-up Sources

Wake-Up Sources	Active in Sleep Mode		
	Idle	Standby	Power-Down
PORT Pin interrupt	X	X	X(1)
BOD VLM interrupt	X	X	X
MVIO interrupts	X	X	X

.....continued

Wake-Up Sources	Active in Sleep Mode		
	Idle	Standby	Power-Down
RTC interrupts	X	X(2,3)	X(3)
TWI Address Match interrupt	X	X	X
Periodic Interrupt	X	X	X(3)
CCL interrupts	X	X(1)	X(4)
USART Start-Of-Frame interrupt	X	X(2)	
TCA interrupts			
TCBn interrupts			
ADC interrupts			
AC interrupts			
ZCD interrupts			
All other interrupts	X		

Notes:

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

19.3.2.2 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 Power Mode (PMODE) bit field in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register is, by default, configured to automatically switch the regulator to low-power mode in sleep mode when OSC32K is the only oscillator enabled and UPDI is inactive. In the low-power mode, the regulator consumes less power but is limited in the amount of current it can provide, which, in turn, limits the clock frequency it can support. A maximum-performance power mode can also be selected using the PMODE bit field. The two modes are described in the following table:

Table 19-5. Voltage Regulator Power Modes Description

Voltage Regulator Power Mode	Description	Condition	Active/Idle	Standby(1)	Power-Down
Normal (AUTO)	Maximum performance in Active mode and Idle mode. Power saving in Standby mode and Power-Down mode.	Internal oscillator with f_{CLK} > 32.768 kHz, external clock, or UPDI active	Maximum Performance	Low Power	Low Power
		f_{CLK} ≤ 32.768 kHz and UPDI inactive	Low Power		
Performance (FULL)	Maximum performance in all modes (Active mode and any Sleep mode) and fast start-up from all sleep modes	Any clock source	Maximum Performance	Maximum Performance	Maximum Performance

Note:

1. If any peripheral requests an External clock/XOSCHF or an internal clock faster than f_{CLK} > 32.768 kHz, or UPDI is active, the power voltage regulator switches to Maximum Performance, also in Normal mode (AUTO).

The VMON in Sleep Mode (VSLP) flag in the Interrupt Flags (INTFLAGS) register will permanently be set when selecting OSC32K as the main clock source with UPDI inactive. Configure the Error Controller so that this does not cause an unwanted response, such as entering a safe state.

The VMON Sleep Mode Enable (VMONSEN) bit in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register can be configured to either monitor the voltage regulator in Standby and Power-Down sleep modes or to be disabled in Standby and Power-Down sleep modes. Enabling VMONSEN in these sleep modes offers protection from Voltage Regulator failure but will increase the power consumption.

19.3.2.3 Wake-Up Time

The ordinary 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 regulator if it has been switched off:

- In Idle sleep mode, the main clock source is kept running to eliminate additional wake-up time
- In Standby sleep mode, the main clock might be running depending on the peripheral configuration
- In Power-Down sleep mode, only the OSC32K oscillator and the Real-Time Clock (RTC) may be running if the clock 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 19-6. Sleep Modes and Start-Up Time

Sleep Mode	Start-Up Time
Idle	Six clock cycles
Standby	Six clock cycles + oscillator start-up time + regulator start-up time
Power-Down	Six clock cycles + oscillator start-up time + regulator start-up time

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

In addition to the ordinary wake-up time, it is possible to make the device wait until the BOD is ready before executing the code. This is controlled by the BOD Operation in the Active and Idle Mode (ACTIVE) bit field in the BOD Configuration (FUSE.BODCFG) fuse. If the BOD is ready before the ordinary wake-up time, the total wake-up time will be the same. If the BOD takes longer than the ordinary wake-up time, the wake-up time will be extended until the BOD is ready, ensuring correct supply voltage whenever code is executed.

19.3.2.4 Voltage Regulator Monitor

SLPCTRL provides the control interface to the Voltage Regulator Monitor (VMON). VMON is an independent monitor for the on-chip voltage regulator. VMON will detect any voltage regulator failure and trigger a Machine Check Reset. VMON monitors the supply voltage when the device is in Active and Idle sleep modes.

The VMON Undervoltage Detected (VUV) flag and the VMON Overvoltage Detected (VOV) flag in the Interrupt Flags (INTFLAGS) register indicate if VMON has detected an under- or overvoltage condition, which is potentially a situation that never will be possible to observe since the reset controller ordinarily will reset the CPU if a VMON event condition occurs, and by doing so will reset the VUV and VOV flags. Such resets can be identified by the Machine Check Reset Flag (MCRF) flag in the Reset Flag Register (RSTCTRL.RSTFR) register and Voltage Regulator Monitor has Tripped (VREG) flag in the Machine Check Flags A (RSTCTRL.MCFLAGSA) register in the Reset Controller. For this reason, the VUV and VOV flags can be considered a redundant mechanism to catch failure modes

in case the reset controller does not work as intended. These flags will also be set by fault injection. See the *Fault Injection* section for details.

VMON will, by default, monitor the voltage in sleep modes, which increases power consumption. Writing the VMON Sleep Mode Enable (VMONSEN) bit in the Voltage Regulator Control (VREGCTRL) register to '0' will disable VMON in Power-Down sleep mode, reducing power consumption but disabling the safety mechanism.

Some sleep modes may cause the voltage regulator to reduce its output voltage to reduce power consumption. The VMON will automatically alter its detection window to match the voltage regulator output target level. The VMON is in Sleep Mode (VSLP) flag in the Interrupt Flags (INTFLAGS) register indicates whether the VMON window has been shifted to match a reduced voltage regulator output voltage.

Note: The VSLP flag in the INTFLAGS register will be permanently set when selecting OSC32K as the main clock source with UPDI inactive. Configure the Error Controller so that this does not cause an unwanted response, such as entering a safe state. If the VSLP flag is set outside a sleep mode, a hardware error has occurred, or OSC32K is the only active clock source.

19.3.2.4.1 Fault Injection

The Control B (CTRLB) register controls fault injection into VMON. The VMON Diagnostic Mode (DMODE) bitfield controls the type of fault injected (correct voltage, undervoltage or overvoltage). Writing DMODE to UNDER or OVER brings VMON into diagnostic mode, triggering a fault injection. Writing DMODE to NO brings VMON out of diagnostic mode. A successful error injection will set the VOV or VUV flags in the INTFLAGS register but will not trigger a Machine Check Reset. When the VMON has entered diagnostic mode, the VMON Has Entered Diagnostic Mode (VDENTER) flag in the INTFLAGS register is set. This flag can also be used to detect if VMON has entered the diagnostic mode in error.

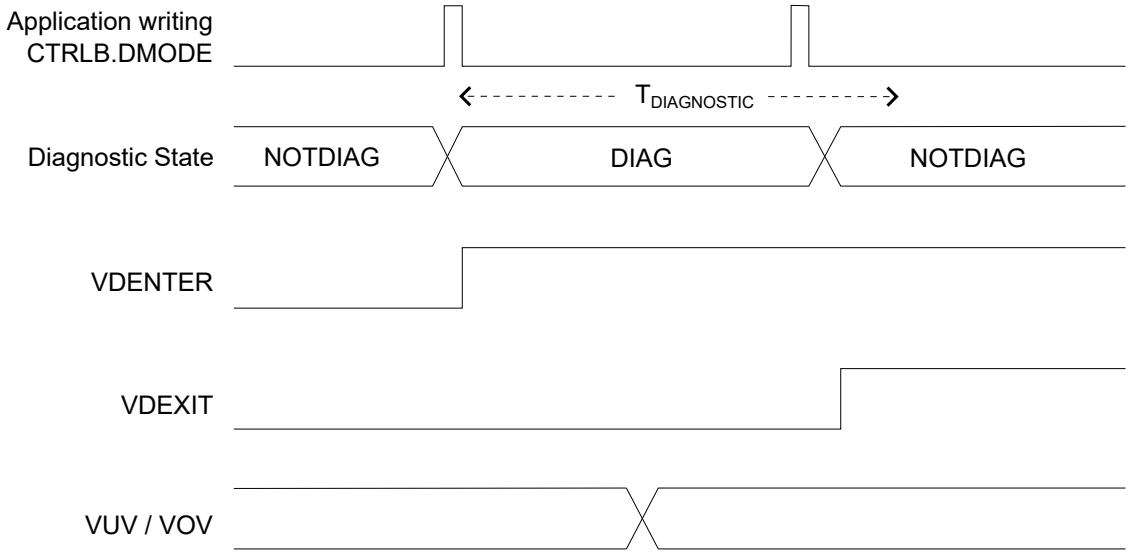
Note: Use the diagnostic mode only when the device runs with the voltage regulator in normal mode (VREGCTRL.PMODE = AUTO), and the power mode is Maximum Performance (see the *Voltage Regulator Configuration* section for details). Do not use the diagnostic mode when OSC32K is the only active clock source.

When VMON enters diagnostic mode, an internal diagnostic timer in VMON starts. If the VMON has not transitioned to normal mode before the timer reaches its timeout value ($T_{DIAGNOSTIC}$ in the figure below), VMON is automatically transitioned to normal mode. To manually end diagnostic mode, write DMODE to NO. The application is expected to manually end diagnostic mode once the diagnostic has been completed with the expected result, enabling maximum system protection as soon as possible. The automatic transition to normal mode upon timeout is intended as a safety mechanism in case a fault in the system somehow prevents manual exit from diagnostic mode.

During the diagnostic cycle, the VUV and VOV flags in the INTFLAGS register are updated as appropriate. The flags can be polled or interrupts enabled for VUV and VOV to detect that the diagnostic is successful. A failed diagnostic, meaning VUV and/or VOV not being set as expected, can be detected by the diagnostic timer reaching its timeout value and the VMON Has Exited Diagnostic Mode (VDEXIT) flag in the INTFLAGS register being set. The VDEXIT flag is also set if VMON transitions to normal mode by writing the DMODE bit field of the CTRLB register to NO.

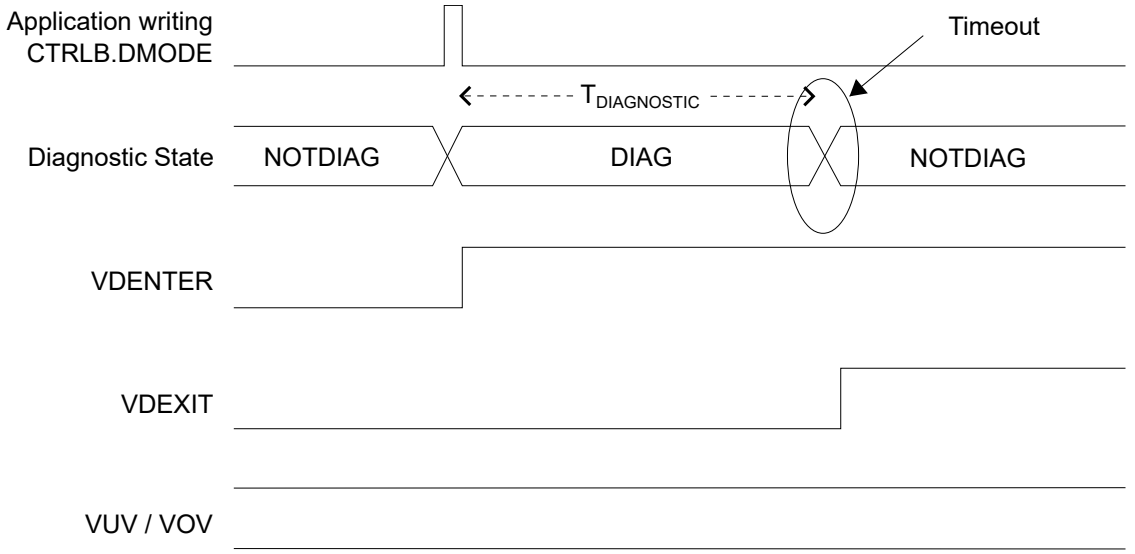
VMON has an internal fault injection state machine with two states: NOTDIAG and DIAG. The following timing diagram shows a manually terminated diagnostic.

Figure 19-2. Manually Terminated Diagnostic



The following timing diagram shows a diagnostic terminated by the timeout mechanism, where VUV and VOV did not get set, potentially due to a failed diagnostic.

Figure 19-3. Diagnostic Terminated by Timeout



19.3.2.5 Data Bus Accesses

Any attempted accesses to unused addresses within the address space of this peripheral will be discarded and return a Bus Error. For more information, see the *BUSMATRIX* section.

19.3.3 Interrupts

Table 19-7. Available Interrupt Vectors and Sources

Interrupt Vector Name	Interrupt Source Name	Description	Conditions
SLPCTRL	VOV	VMON overvoltage detected	VMON detects voltage level above high threshold
	VUV	VMON undervoltage detected	VMON detects voltage level below low threshold
	VDENTER	VMON has entered diagnostic (fault injection) mode	CTRLB.DMODE = 'UNDER' or CTRLB.DMODE = 'OVER'
	VDEXIT	VMON has exited diagnostic (fault injection) mode	Diagnostic mode exited due to timeout or because CTRLB.DMODE = 'NO'
	VERR	VMON has an internal error	VMON is in an illegal state
	VSLP	VMON is in sleep mode	The device has entered sleep mode and VMON thresholds are adjusted to sleep mode voltage levels. This flag will be permanently set when selecting OSC32K as main clock source.
	VDIS	VMON is disabled	Either of these: • A hardware error has occurred erroneously disabling the VMON • VREGCTRL.VMONSEN = 'DISABLE' and Standby or Power-Down sleep mode is entered
	SERR	Sleep error	SLEEP instruction was attempted with CTRLA.SEN = '0' or CPU.SREG.I = '0'

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.

19.3.4 Events

None.

19.3.5 Debug Operation

During run-time debugging, this peripheral will continue ordinary 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.

19.3.6 Configuration Change Protection

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

Writing to a CCP-protected register outside the CCP sequence will be disregarded and return a Bus Error.

The following registers are under CCP:

Table 19-8. SLPCTRL - Registers Under Configuration Change Protection

Register	Key
SLPCTRL.CTRLB	IOREG
SLPCTRL.VREGCTRL	IOREG

19.4 Functional Safety

The following Functional Safety features are included:

- Execution of `SLEEP` automatically clears the Sleep Enable (SEN) bit in the Control A (CTRLA) register to increase robustness against inadvertent sleep
- Execution of `SLEEP` requires the Global Interrupt Enable (I) bit in the CPU Status Register (CPU.SREG) register to be set to '1', otherwise an error is flagged in the Interrupt Flags (INTFLAGS) register

19.5 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0						SMODE[2:0]		SEN
0x01	CTRLB	7:0							DMODE[1:0]	
0x02	VREGCTRL	7:0		VMONSEN				PMODE[2:0]		
0x03	INTCTRL	7:0	VOV	VUV	VDENTER	VDEXIT	VERR	VSLP	VDIS	SERR
0x04	INTFLAGS	7:0	VOV	VUV	VDENTER	VDEXIT	VERR	VSLP	VDIS	SERR

19.6 Register Description

19.6.1 Control A Register

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 executing the SLEEP instruction to make the microcontroller enter the selected sleep mode. It will automatically clear after a successful SLEEP instruction executes.
Note: To avoid unintentional entering of Sleep modes, it is recommended to write SEN = '1' just before executing the SLEEP instruction.

19.6.2 Control B Register

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
							DMODE[1:0]	
Access							W	W
Reset							0	0

Bits 1:0 – DMODE[1:0] VMON Diagnostic Mode
Writing these bits selects the VMON diagnostic mode. Reading this register always return '0'.

Value	Name	Description
0x0	NO	No VMON error is expected. VMON will operate as normal using normal detection thresholds.
0x1	UNDER	VMON Undervoltage error will be injected and must be detected
0x2	OVER	VMON Overvoltage error will be injected and must be detected
Other	—	Reserved

19.6.3 Voltage Regulator Control Register

Name: VREGCTRL
Offset: 0x02
Reset: 0x40
Property: Configuration Change Protection

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

Bit 6 – VMONSEN VMON Sleep Mode Enable
Writing this bit to '0' disables the VMON in Standby or Power-Down sleep modes, which reduces power consumption at the cost of reduced protection.

Value	Name	Description
0	DISABLE	VMON disabled in Standby and Power-Down sleep mode
1	ENABLE	VMON enabled in Standby and Power-Down sleep mode (Reset value)

Bits 2:0 – PMODE[2:0] Power Mode Select
This bit field controls the drive strength of the voltage regulator.

Value	Name	Description
0x0	AUTO	In Active and Idle sleep modes with any clock source other than the 32.768 kHz oscillator, or if UPDI is active, the voltage regulator is in maximum performance mode. In Active and Idle sleep modes with the 32.768 kHz oscillator and UPDI inactive, the voltage regulator is in power-saving mode. In Standby and Power-Down sleep modes, the voltage regulator is in power-saving mode, unless UPDI is active or a peripheral is requesting a clock source faster than 32.768 kHz.
0x1	FULL	The voltage regulator is in maximum performance mode for Active and all sleep modes. A shorter time is needed to wake the device from sleep mode.
Other	—	Reserved

19.6.4 Interrupt Control Register

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

Bit	7	6	5	4	3	2	1	0
	VOV	VUV	VDENTER	VDEXIT	VERR	VSLP	VDIS	SERR
Access	R/W	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 – VOV VMON Overvoltage Detected
Writing a '1' to this bit enables the interrupt.

Bit 6 – VUV VMON Undervoltage Detected
Writing a '1' to this bit enables the interrupt.

Bit 5 – VDENTER VMON Has Entered Diagnostic Mode
Writing a '1' to this bit enables the interrupt.

Bit 4 – VDEXIT VMON Has Exited Diagnostic Mode
Writing a '1' to this bit enables the interrupt.

Bit 3 – VERR VMON Has an Internal Error
Writing a '1' to this bit enables the interrupt.

Bit 2 – VSLP VMON is in Sleep Mode
Writing a '1' to this bit enables the interrupt.

Bit 1 – VDIS VMON is Disabled
Writing a '1' to this bit enables the interrupt.

Bit 0 – SERR Sleep Error
Writing a '1' to this bit enables the interrupt.

19.6.5 Interrupt Flags Register

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

Bit	7	6	5	4	3	2	1	0
	VOV	VUV	VDENTER	VDEXIT	VERR	VSLP	VDIS	SERR
Access	R/W	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 – VOV VMON Overvoltage Detected
This bit is set when VMON has detected an Overvoltage condition. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 6 – VUV VMON Undervoltage Detected
This bit is set when VMON has detected an Undervoltage condition. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 5 – VDENTER VMON Has Entered Diagnostic Mode
This bit is set when VMON has entered diagnostic (fault injection) mode. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 4 – VDEXIT VMON Has Exited Diagnostic Mode
This bit is set when VMON has exited diagnostic (fault injection) mode, either due to timeout or because CTRLB.DMODE was written to 'NO'. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 3 – VERR VMON has an internal error
This bit is set when VMON is in an illegal state. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 2 – VSLP VMON is in Sleep Mode
This bit is set when VMON is in a Sleep state. Voltage thresholds match the voltage regulator sleep mode output voltage. This flag is permanently set when OSC32K is selected as the main clock source and UPDI is inactive. Any error controller present in the system must be configured in a way that does not cause an unwanted response, such as entering a safe state. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 1 – VDIS VMON is Disabled
This bit is set when VMON is disabled. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 0 – SERR Sleep Error
This bit is set when a SLEEP instruction was attempted executed, but was aborted due to an illegal condition. An error is flagged if attempting to enter sleep with CTRLA.SEN = '0' or CPU.SREG.I = '0'. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

20. RSTCTRL - Reset Controller

20.1 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
- Error Controller Reset
- Machine Check Reset:
  - Voltage Monitor
  - CRCSCAN failed
  - WDT clock failure
  - Boot process error
  - Dual-core lockstep CPU error
  - Error Controller internal error
  - Design-For-Test mechanism enabled error
  - On-Chip Debug system enabled error
  - UPDI Illegal Bus Activity Detected
  - Clock Failure Detected

20.2 Overview

The Reset Controller (RSTCTRL) manages the Reset of the device. When receiving a Reset request, the device enters an initial state, which may identify the Reset source by the software. The Reset controller can also be issue a Software Reset (SWRST).

20.2.1 Block Diagram

Figure 20-1. Block Diagram

20.2.2 Signal Description

Signal	Description	Type
$\overline{\text{RESET}}$	External Reset (active-low)	Digital input
UPDI	Unified Program and Debug Interface	Digital input

20.3 Functional Description

20.3.1 Initialization

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

The registers in the device with automatic loading from the Fuses or the Signature Row are updated. The program counter will be set to 0x0000 after a Reset from any source.

20.3.2 Operation

20.3.2.1 Reset Sources

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

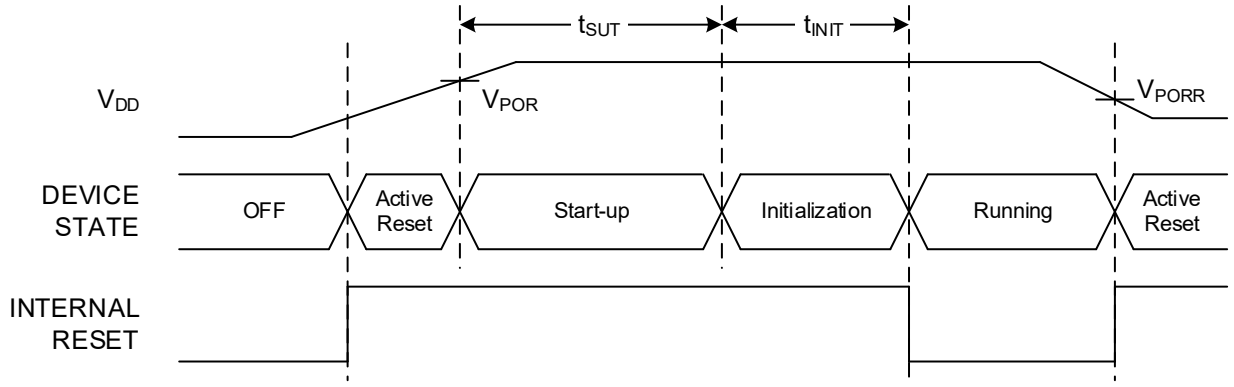
There are four types of Resets based on the source:

- 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
- Error Controller Reset
- Machine Check Reset:
  - Voltage Regulator Monitor
  - CRCSCAN failed
  - WDT clock failure
  - Boot process error
  - Dual-core lockstep CPU error
  - Error Controller internal error
  - Design-For-Test mechanism enabled error
  - On-Chip debug system enabled error
  - UPDI Illegal Bus Activity Detected
  - Clock Failure Detected

20.3.2.1.1 Power-on Reset (POR)

The Power-on Reset (POR) aims to ensure a safe start-up of logic and memories, a process generated by an on-chip detection circuit that is always enabled. The POR is activated when the V_{DD} rises and sets an active Reset as long as the V_{DD} is below the POR threshold voltage (V_{POR}). The Reset will continue until the Start-up and Reset initialization sequence is completed. Fuses determine the Start-Up Time (SUT). Reset is activated again, without delay, when V_{DD} falls below the detection level (V_{PORR}).

Figure 20-2. MCU Start-Up, $\overline{\text{RESET}}$ Tied to V_{DD}

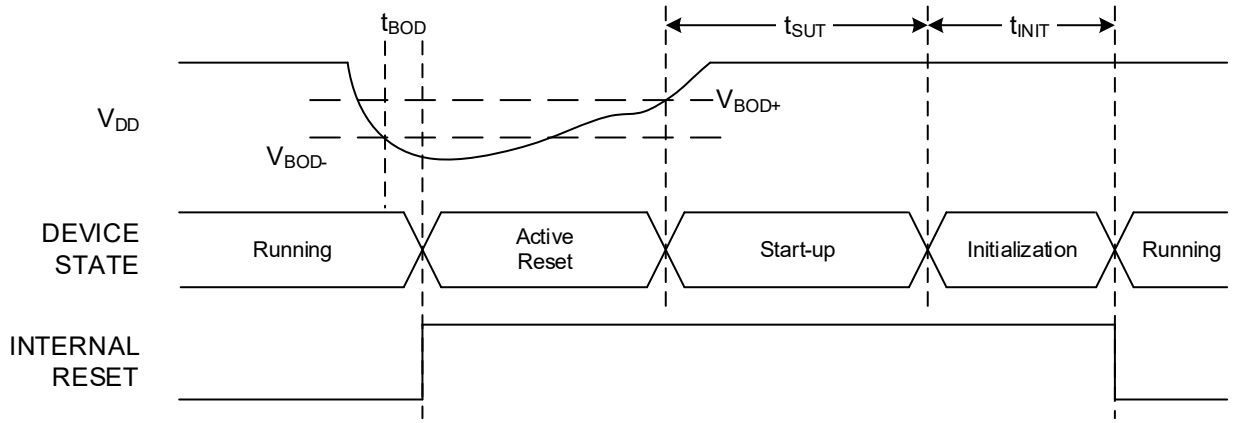


20.3.2.1.2 Brown-out Detector (BOD) Reset

The Brown-out Detector (BOD) can be enabled by the user. The BOD halts code execution when the voltage falls below a specified threshold, ensuring the required voltage for the oscillator to operate at the configured application speed and preventing code corruption caused by low-voltage levels.

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 BOD Configuration (FUSE.BODCFG) fuse must select the trigger level for the BOD.

Figure 20-3. Brown-out Detector Reset



20.3.2.1.3 External Reset ($\overline{\text{RESET}}$)

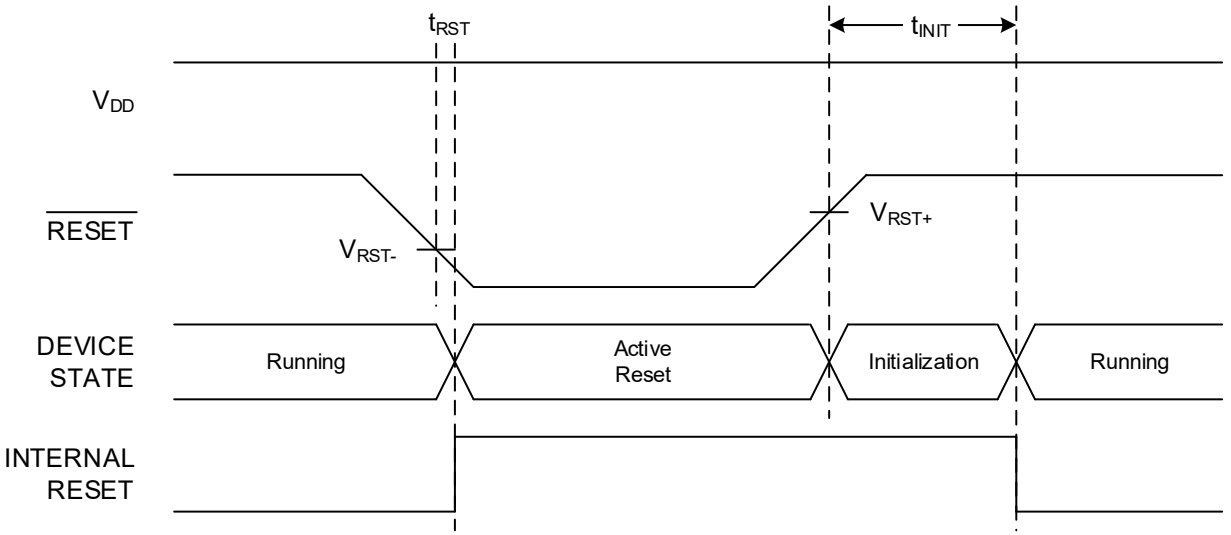
The $\overline{\text{RESET}}$ pin requires a noise filter that eliminates short, low-going pulses. Filtering the input assures that an external Reset event is only issued when the $\overline{\text{RESET}}$ has been low for a minimum amount of time. See the *Electrical Characteristics* section for the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

The external Reset is enabled by configuring the Reset Pin Configuration (RSTPINCFG) bit field 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.

For specific devices (e.g., AVR SD), the $\overline{\text{RESET}}$ pin is permanently allocated to the $\overline{\text{RESET}}$ function and always enabled, and configuration of the Reset Pin is impossible.

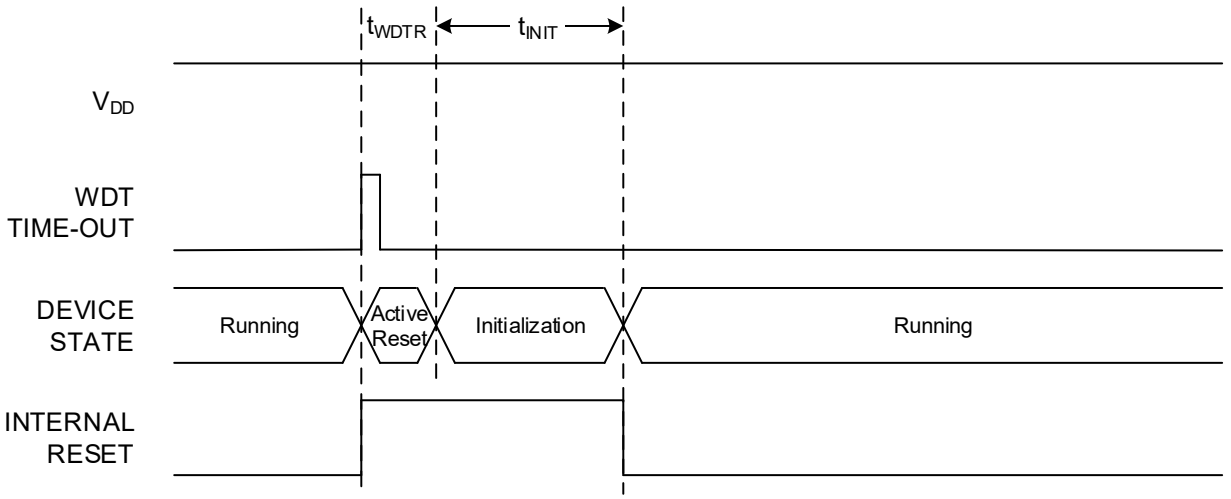
Figure 20-4. External Reset Characteristics



20.3.2.1.4 Watchdog Timer (WDT) Reset

The Watchdog Timer (WDT) is a system function that monitors the program's operation. A Watchdog Reset will be issued if the software does not handle the WDT according to the programmed time-out period. Find more details in the *WDT - Watchdog Timer* section.

Figure 20-5. Watchdog Reset

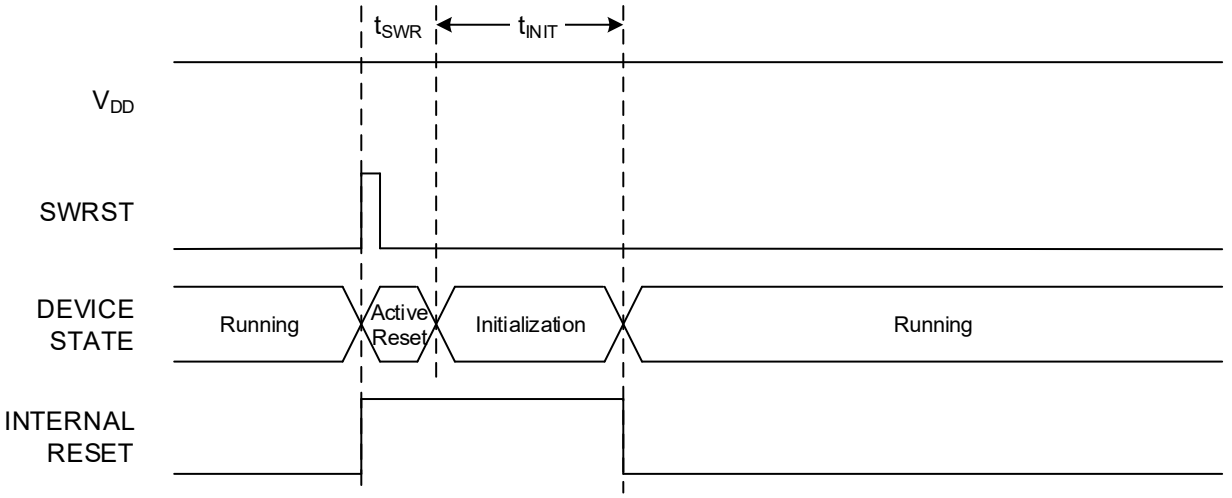


20.3.2.1.5 Software Reset (SWRST)

The Software Reset makes it possible to issue a System Reset from the software. Writing a '1' to the Software Reset (SWRST) bit in the Software Reset (RSTCTRL.SWRR) register generates the Reset.

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

Figure 20-6. Software Reset



20.3.2.1.6 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. Find more details in the *UPDI - Unified Program and Debug Interface* section.

20.3.2.1.7 Error Controller (ERRCTRL) Reset

In a Functional Safety application, the software configures the Error Controller. The ERRCTRL prioritizes various error sources. A failure in an error source assigned CRITICAL priority will reset the device and set the Error Controller Flag (ECRF).

20.3.2.1.8 Machine Check (MC) Reset

The MC Reset is asserted if a critical hardware error is detected. Such an error will reset the device and set the Machine Check Reset Flag (MCRF). The MCFLAGSA and MCFLAGSB registers will identify the cause of the reset.

20.3.2.1.9 RESET Cause

Special Function Registers (SFRs) are provided to identify Reset causes, as shown in the *Interpretation of RESET Flags* table. Multiple Resets can simultaneously occur, leading to the activation of several Reset cause flags.

Table 20-1. Nomenclature

Value	Description
0	Cleared - This flag is cleared in the register
1	Set - This flag is set in the register
U	Undefined - See notes
V	Valid - This flag may be set while another flag is currently set

Table 20-2. Interpretation of RESET Flags

MCFLAGSB		MCFLAGSA								RSTFR								Cause
UPDI	CFD	VREG	CRC	WDT	BOOT	DCLS	EC	DFT	OCD	MCRF	ECRF	UPDIRF	SWRF	WDRF	EXTRF	BORF	PORF	
0	**0**	**U**	V	**0**	V	V	**0**	**0**	**0**	V	**0**	**0**	**0**	**0**	**0**	**0**	**1**	POR[1]

.....continued

MCFLAGSB		MCFLAGSA								RSTFR								Cause
UPDI	CFD	VREG	CRC	WDT	BOOT	DCLS	EC	DFT	OCD	MCRF	ECRF	UPDIRF	SWRF	WDRF	EXTRF	BORF	PORF	
V	V	**U**	V	V	V	V	V	V	V	V	V	V	V	V	V	**1**	**0**	BOR[2,3]
V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	**1**	V	**0**	External Reset[2,3]
V	V	V	V	V	V	V	V	V	V	V	V	V	V	**1**	V	V	**0**	Watchdog Reset[1,2]
V	V	V	V	V	V	V	V	V	V	V	V	V	**1**	V	V	V	**0**	Software Reset[2,3]
V	V	V	V	V	V	V	V	V	V	V	V	**1**	V	V	V	V	**0**	UPDI Reset[2,3]
V	V	V	V	V	V	V	V	V	V	V	**1**	V	V	V	V	V	**0**	Error Controller Reset[2,3]
V	V	V	V	V	V	V	V	V	V	**1**	V	V	V	V	V	V	**0**	Machine Check Reset

Notes:

1. An error may be detected during the boot process after the initial Reset, causing MCRF to be set and MCFLAGS to identify the source. MCRF will also be set if the only MCFLAG set is VREG, which has an UNDEFINED value. In this case, MCRF should also be considered UNDEFINED.
2. An error may be detected during the boot process after the initial Reset, causing MCRF to be set and the MCFLAGS to identify the source.
3. MCFLAGSA and MCFLAGSB are only valid if MCRF in RSTFR is set.

20.3.2.1.10 Domains Affected By Reset

The following logic domains are affected by the various Resets:

Table 20-3. Logic Domains Affected by Various Resets

Reset Type	Fuses Are Reloaded	Reset of BOD Configuration	Reset of UPDI	Reset of Other Volatile Logic
POR	X	X	X	X
BOD	X		X	X
External Reset	X		X	X
Watchdog Reset	X		X	X
Software Reset	X		X	X
UPDI Reset	X			X
Error Controller Reset	X		X	X
Machine Check Reset	X	X	X	X

20.3.2.2 Reset Time

The Reset time can be divided into two parts.

The first part is when any of the Reset sources are active, which 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 when 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. If a Power Supply Reset Source has triggered the Reset, the time will increase by the start-up time specified in the Start-Up Time (SUT) bit field within the System Configuration 1 (FUSE.SYSCFG1) fuse. The internal Reset initialization time will increase if the Cyclic Redundancy

Check Memory Scan (CRCSCAN) is set to run at start-up. It is possible to change this configuration in the CRC-on-Boot (CRCBOOT) bit field section in the System Configuration 0 (FUSE.SYSCFG0) fuse.

20.3.2.3 Data Bus Accesses

Attempted accesses to any unused SFR addresses within the address space will be discarded and will return a Bus Error.

20.3.3 Sleep Mode Operation

The RSTCTRL operates in Active mode and all sleep modes.

20.3.4 Configuration Change Protection (CCP)

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

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves it unchanged and returns a Bus error.

The following registers are under CCP:

Table 20-4. RSTCTRL - Registers Under CCP

Register	Key
RSTCTRL.SWRR	IOREG

20.3.5 Functional Safety

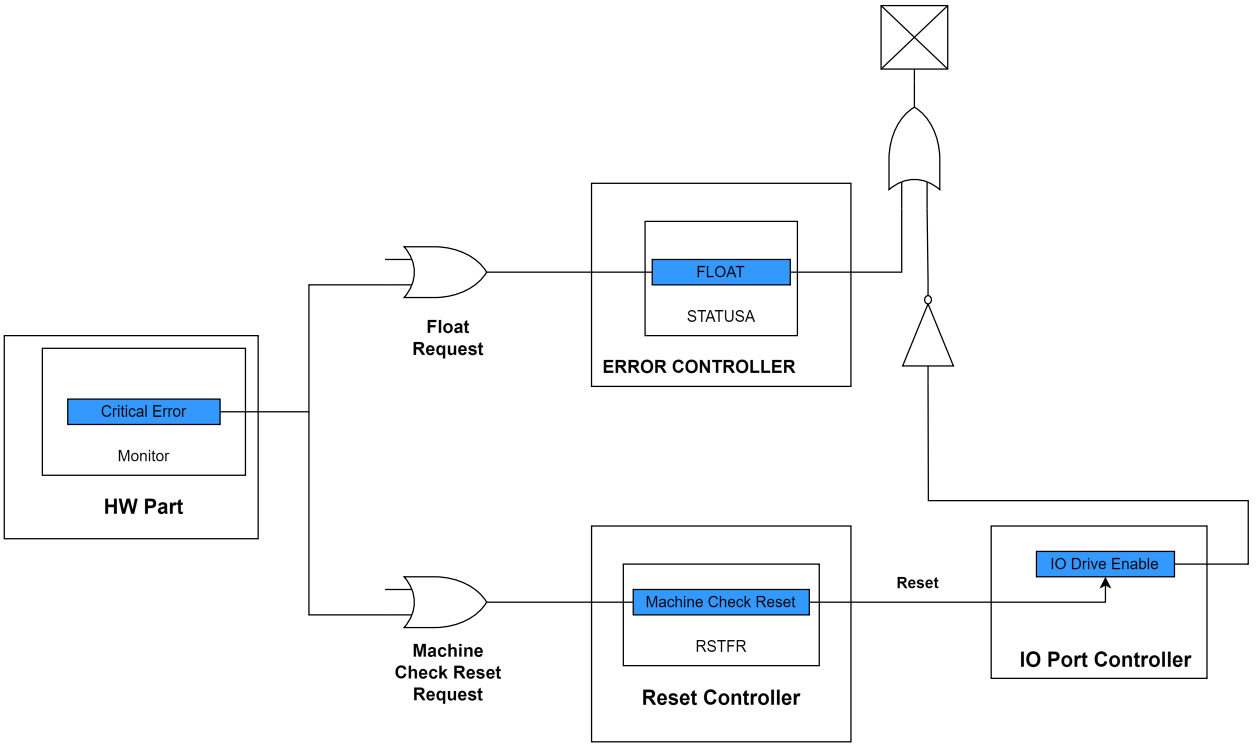
The following Functional Safety features are added:

- Machine Check Reset Flag (RSTFR.MCRF) for unrecoverable HW faults
- Error Controller Reset Flag (RSTFR.ECRF) for HW faults reported by the Error Controller

The Reset controller is essential for initiating a Safe State when the microcontroller (MCU) diagnostics detect a Functional Safety-related failure. Safe State requires the tristating of I/O pins. To avoid having RSTCTRL as a source of latent failures, the Machine Check Reset request is routed both to RSTCTRL and the Error Controller, where it will cause the Error Controller to assert its “Float IO” signal that is routed to each I/O pad, see the figure below.

The Voltage Monitor will detect a failure of VDDCORE and reset the system by requesting a Machine Check Reset. The appropriate Machine Check Reset flag will be set. All the logic in the figure below is powered by VDDCORE, except the VREG flag in MCFLAGSA. Even if there is a VDDCORE failure, the VREG flag will keep its value and receive power from the VDDIO domain.

Figure 20-7. System Connections of Machine Check Error



20.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	RSTFR	7:0	MCRF	ECRF	UPDIRF	SWRF	WDRF	EXTRF	BORF	PORF
0x01	SWRR	7:0								SWRST
0x02	MCFLAGSA	7:0	VREG	CRC	WDT	BOOT	DCLS	EC	DFT	OCD
0x03	MCFLAGSB	7:0							UPDI	CFD

20.5 Register Description

20.5.1 Reset Flag Register

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

A Power-on Reset (POR) will clear all flags except for the Power-on Reset (PORF) flag.

A Brownout Reset (BOR) will clear all flags except for the Brownout Reset (BORF) flag, and the Power-On Reset (PORF) flag will be left unchanged.

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

Bit 7 – MCRF Machine Check Reset Flag
This flag is cleared by writing a '1' to it.
This flag is set if a Machine Check Reset has occurred. The cause for the error is described in the MCFLAGSA and MCFLAGSB registers.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Machine Check Reset flag.

Bit 6 – ECRF Error Controller Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set if an Error Controller Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Error Controller Reset flag.

Bit 5 – UPDIRF UPDI Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set if a UPDI Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the UPDI Reset flag.

Bit 4 – SWRF Software Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set if a Software Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Software Reset flag.

Bit 3 – WDRF Watchdog Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set if a Watchdog Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Watchdog Reset flag.

Bit 2 – EXTRF External Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set if an External Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the External Reset flag.

Bit 1 – BORF Brown-out Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set if a Brown-out Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Brown-out Reset flag.

Bit 0 – PORF Power-on Reset Flag
This flag is cleared by writing a '1' to it.
This bit is set to if a Power-on Reset has occurred.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Power-on Reset flag.
After a POR, only the POR flag is set, and all the other flags are cleared. Errors may be detected during the boot process after the initial Reset, allowing the MCR flag to be set and the MCFLAGSA and MCFLAGSB registers to identify the source.

20.5.2 Software Reset Register

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

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

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will trigger a software reset.

20.5.3 Machine Check Flags A

Name: MCFLAGSA
Offset: 0x02
Reset: 0xXX
Property: -

A Power-on Reset (POR) will clear all flags.

Bit	7	6	5	4	3	2	1	0
	VREG	CRC	WDT	BOOT	DCLS	EC	DFT	OCD
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bit 7 – VREG Voltage Regulator Monitor has Tripped
This flag is cleared by writing a '1' to it.
This flag is set if the VREG monitor generates a system reset.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Voltage Regulator Monitor has Tripped flag.

Bit 6 – CRC CRC Scan Error
This flag is cleared by writing a '1' to it.
This bit is set if the CRC Scan module detects a CRC error while scanning Flash during the boot sequence and generates a system Reset.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the CRC Scan Error flag.

Bit 5 – WDT WDT Clock Failure Monitor has Tripped
This flag is cleared by writing a '1' to it.
This bit is set if the WDT clock failure monitor generates a system Reset.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the WDT Clock Failure Monitor has Tripped flag.

Bit 4 – BOOT Error detected during System Start-up
This flag is cleared by writing a '1' to it.
This bit is set if detecting an error during system startup that generates a system Reset. This includes errors found during the CRC scan of the Boot ROM.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Error detected during System Start-up flag.

Bit 3 – DCLS Dual Core Lockstep Comparator Mismatch
This flag is cleared by writing a '1' to it.
This bit is set if the DCLSC generates a system Reset.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Dual Core Lockstep Comparator Mismatch flag.

Bit 2 – EC Error Controller Internal Error
This flag is cleared by writing a '1' to it.
This bit is set if the Error Controller generates a system Reset.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Error Controller Internal Error flag.

Bit 1 – DFT Design-For-Test Mechanisms Enabled
This flag is cleared by writing a '1' to it.

This bit is set if the internal consistency check detects that the Design-For-Test feature has been erroneously enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Design-For-Test Mechanisms Enabled flag.

Bit 0 – OCD OCD System Enabled
This flag is cleared by writing a '1' to it.
This bit is set if the internal consistency check detects that the OCD system has been erroneously enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the OCD System Enabled flag.

20.5.4 Machine Check Reset Flags B

Name: MCFLAGSB
Offset: 0x03
Reset: 0xXX
Property: -

A Power-on Reset (POR) will clear all flags.

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

Bit 1 – UPDI Illegal Bus Activity Detected
This flag is cleared by writing a '1' to it.
This bit is set if the internal consistency check detects that the UPDI bus controller is active when it should not be, such as when the device is locked.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Illegal Bus Activity Detected flag.

Bit 0 – CFD Clock Failure Detected
This flag is cleared by writing a '1' to it.
This bit is set if a CFD monitor triggers a system Reset.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Clock Failure Detected flag.

21. CPUINT - CPU Interrupt Controller

21.1 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)

21.2 Overview

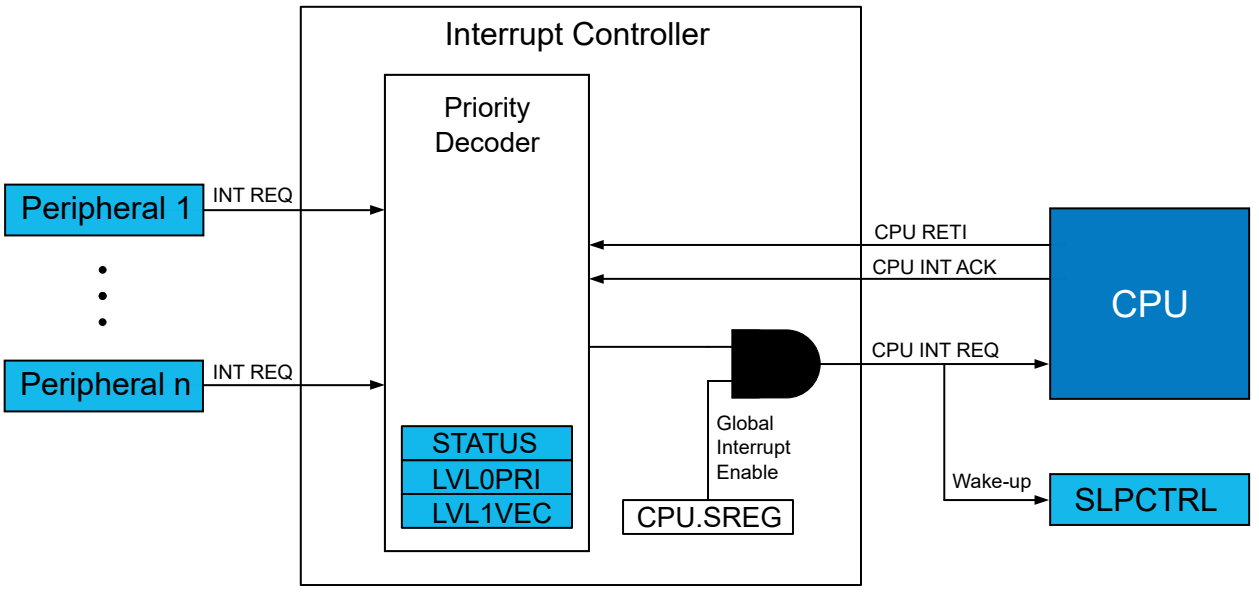
An interrupt request signals a state change inside a peripheral and can be used to alter the program execution. The peripherals can have one or more interrupts. All interrupts are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when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

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 servicing all interrupts within a certain amount of time.

21.2.1 Block Diagram

Figure 21-1. CPUINT Block Diagram



21.3 Functional Description

21.3.1 Initialization

Initialize an interrupt in the following order:

1. Optional: Configure the expected location of the interrupt vectors using the IVSEL bit in the Control A (CPUINT.CTRLA) register.
2. Optional: Enable compact vector table by writing '1' to the CVT bit in the Control A (CPUINT.CTRLA) register.
3. Optional: Enable vector prioritizing by round robin by writing a '1' to the Round Robin Priority Enable (LVL0RR) bit in CPUINT.CTRLA.
4. Optional: Select the Priority Level 1 vector by writing the interrupt vector number to the Interrupt Vector with Priority Level 1 (CPUINT.LVL1VEC) register.
5. Optional: Modify the priority of the LVL0 interrupts by configuring Interrupt Priority Level 0 (LVL0PRI) register.
6. Configure the interrupt conditions within each peripheral and enable the peripheral's interrupt.
7. Enable interrupts globally by writing a '1' to the Global Interrupt Enable (I) bit in the CPU Status (CPU.SREG) register.

21.3.2 Operation

21.3.2.1 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.

21.3.2.2 Interrupt Vector Locations

The expected location of interrupt vectors 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.

21.3.2.3 Interrupt Response Time

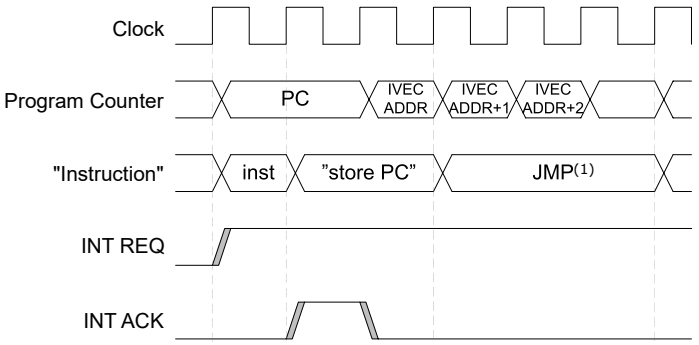
The minimum interrupt response time is represented in the following table.

Table 21-1. 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`)

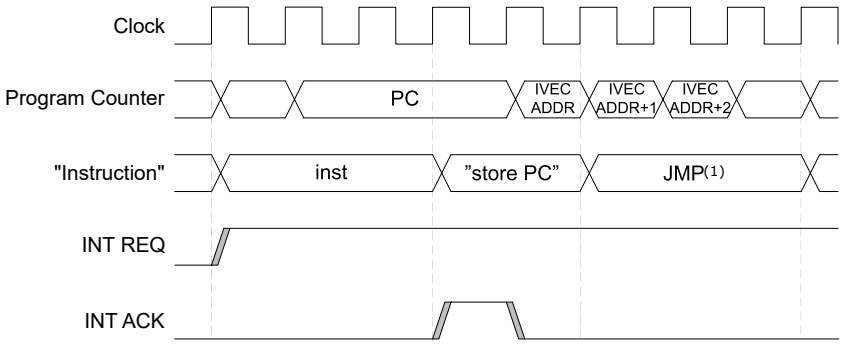
After the Program Counter is pushed on the stack, the program vector for the interrupt is executed. See the following figure.

Figure 21-2. Interrupt Execution of Single-Cycle Instruction



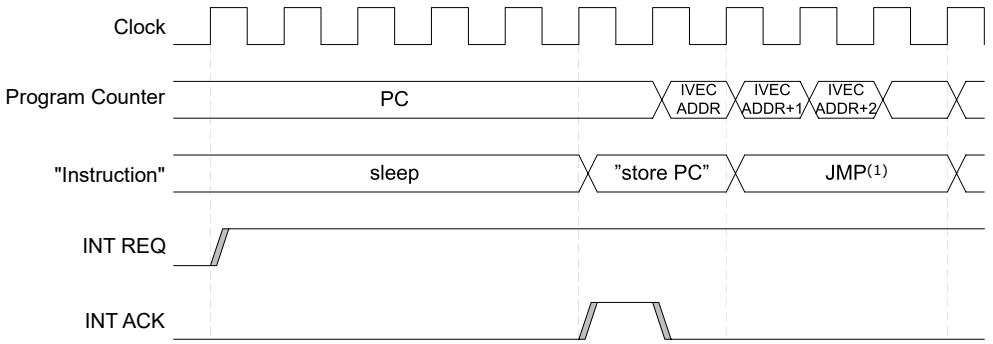
If an interrupt occurs during the execution of a multi-cycle instruction, the instruction is completed before the interrupt is served, as shown in the following figure.

Figure 21-3. 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 21-4. 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.

21.3.2.4 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 21-2. 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

21.3.2.4.1 Non-Maskable Interrupts

A Non-Maskable Interrupt (NMI) will be executed regardless of the I bit setting 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* table of the device for available NMI sources.

21.3.2.4.2 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.

21.3.2.4.3 Normal-Priority Interrupts

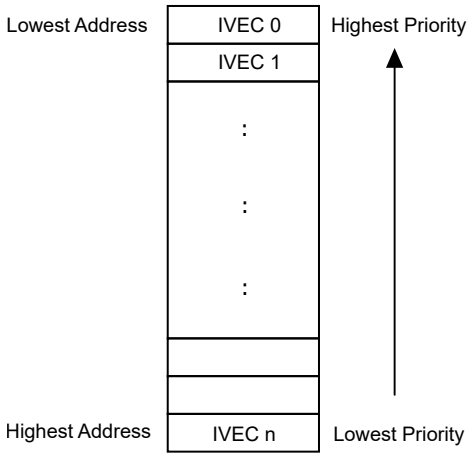
All interrupt vectors other than NMI are assigned to priority level 0 (normal) by default. The user may override this by assigning one of these vectors as a high-priority vector. The device will have many normal-priority vectors, and some of these may be pending at the same time. Two different scheduling 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* section. 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.

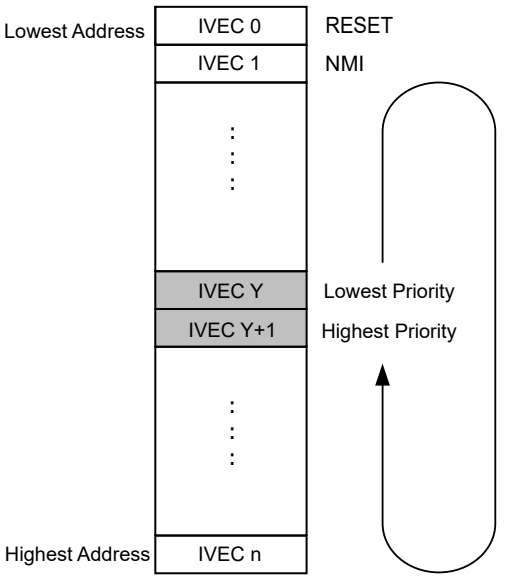
Figure 21-5. Default Static Scheduling



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 21-6. Static Scheduling When CPUINT.LVL0PRI Is Different from Zero



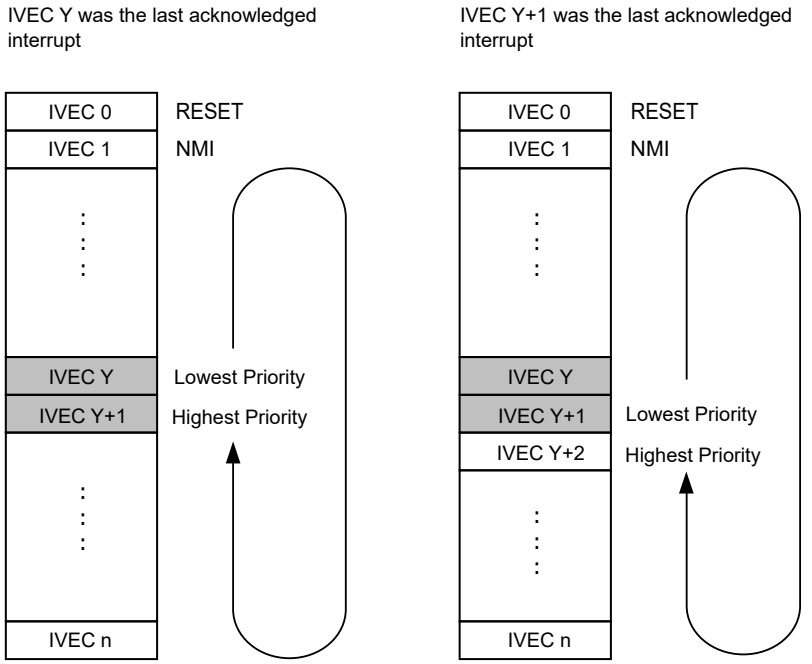
Here, value Y has been written to CPUINT.LVL0PRI so that the interrupt vector Y+1 has the highest priority. Note that, In this case, the priorities will wrap so that the lowest address no longer has the highest priority, not including RESET and NMI, which will always have the highest priority.

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 21-7. 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.

21.3.2.5 Compact Vector Table

The Compact Vector Table (CVT) is a feature to allow the 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 few of interrupt generators.

21.3.3 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.

21.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a given 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 it unchanged.

The following registers are under CCP:

Table 21-3. CPUINT - Registers Under Configuration Change Protection

Register	Key
The IVSEL and CVT bit fields in CPUINT.CTRLA	IOREG

21.4 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]							

21.5 Register Description

21.5.1 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
When the entire Flash is configured as a BOOT section, this bit will be ignored.

Value	Description
0	The expected location of the interrupt vectors is directly after the BOOT section(1)
1	The expected location of the interrupt vectors is at the start of the BOOT section

Note:
1. A system reset will cause the Program Counter to be reset to 0x0000, regardless of the IVSEL bit value.

Bit 5 – CVT Compact Vector Table

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

21.5.2 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.

21.5.3 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.

21.5.4 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.

22. ERRCTRL - Error Controller

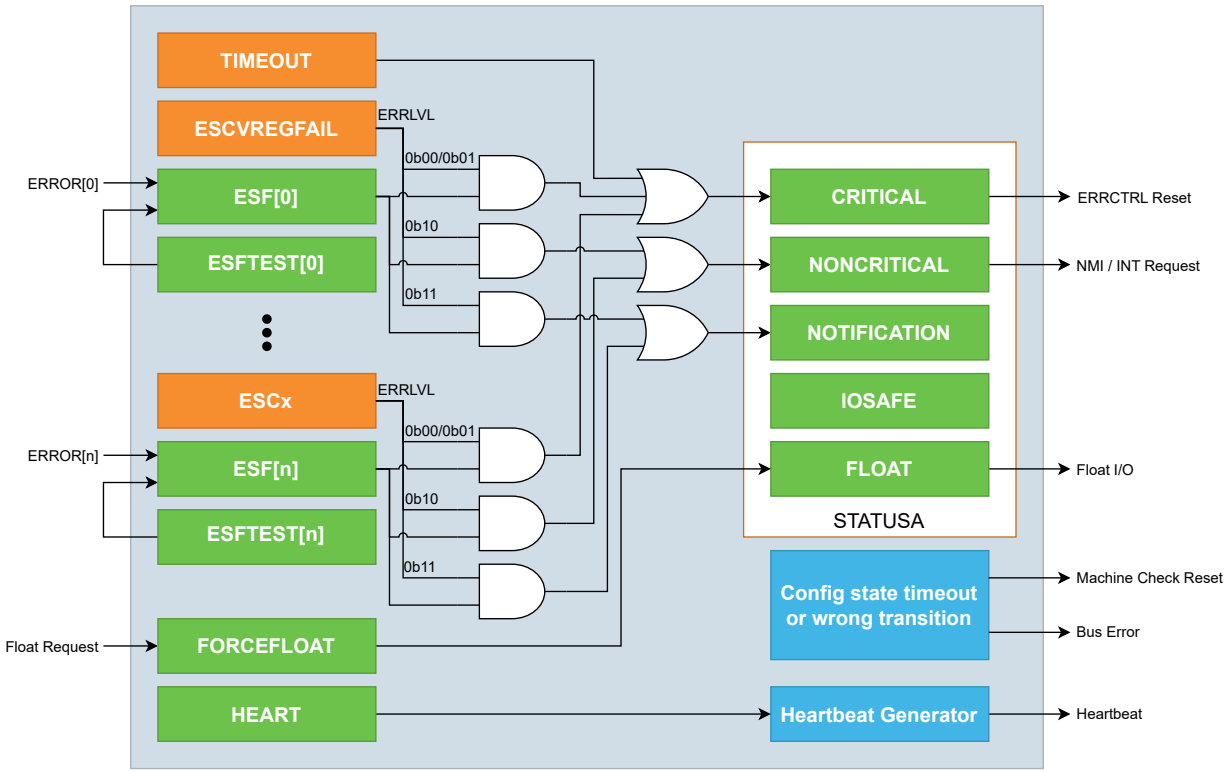
22.1 Features

- Common Interface for Faults Detected in Various Hardware Elements in the Device
- Each Fault Has a Programmable Severity Level:
  - NOTIFICATION
  - NONCRITICAL
  - CRITICAL
- Hardware Handling of These Faults According to Their Severity
- Automatically Transitions the Device to a Safe State for Faults with a Severity Level of CRITICAL
- Heartbeat Signal Optionally Output on I/O Pin to Signal Normal Operation to an External System
- Optional Float (Tri-state) of All I/O Pins When Fault is Detected
- Fault Injection Allows Verification of Correct Operation
- Redundancy and Consistency Checks Incorporated to Detect Internal Faults

22.2 Overview

22.2.1 Block Diagram

Figure 22-1. Error Controller



22.3 Functional Description

22.3.1 Overview

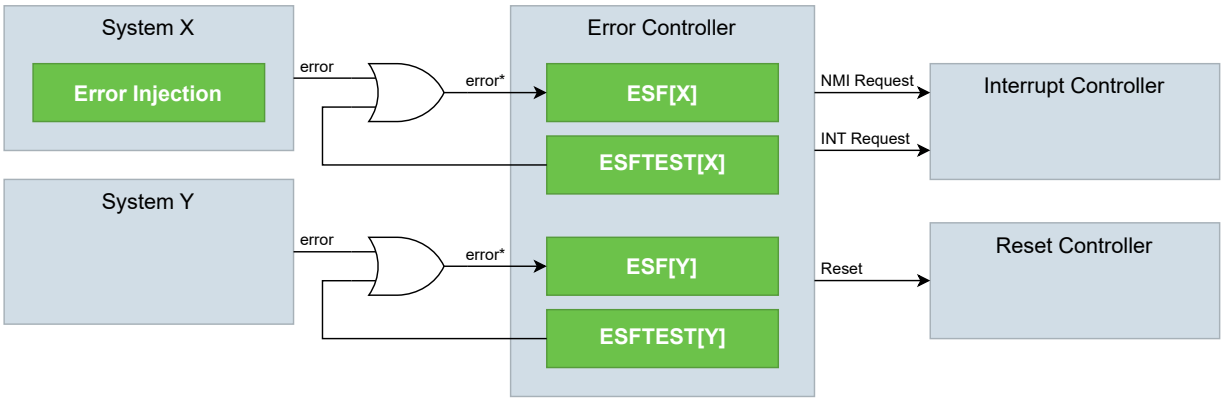
The Error Controller's (ERRCTRL) purpose is to collect hardware error reports from various systems in a functional safety MCU, present them to the application, and take immediate action if needed. The ERRCTRL is always enabled, and a timeout mechanism ensures it can always respond to errors.

The ERRCTRL is an interface or front-end to various MCU subsystems equipped to detect internal hardware errors, as shown in the block diagram above. The application can associate various severity levels with each hardware error by configuring the Error Source Control x (ERRCTRLn.ESCx) registers.

The ERRCTRL can be configured to output a heartbeat signal on an I/O pin. As long as no error is detected, this signal is a square wave with a specific frequency. When an error is detected, the heartbeat signal stays at a fixed level and does not change.

Each error source, such as Instruction ECC error or Stack Pointer Limit, is connected to an Error Channel. Whenever an MCU component detects a hardware error, it signals the ERRCTRL using the associated error channel. This causes the error channel's Error Status Flag to be set in the Error Status Flags (ERRCTRL.ESF) register. The error channel's ERRCTRL.ESCx register determines the resulting severity. The figure below shows how the ERRCTRL is connected to other systems X and Y in the MCU. For additional information, see the *Error Injection* section.

Figure 22-2. System Integration



22.3.2 Severity Levels

The value written to the Error Severity Level (ERRLVL) bit field in the associated Error Source Control x (ERRCTRL.ESCx) register determines the severity of each error source. The severity levels are as follows:

Table 22-1. Severity Levels

ERRLVL bit field in ESCx	Severity	Effect
00	CRITICAL	The system has received a CRITICAL error. • The Critical Error Detected (CRITICAL) bit in the Status A (ERRCTRL.STATUSA) register is set • The appropriate bit in the Error Status Flags (ERRCTRL.ESF) register is set On the subsequent clock edge, the ERRCTRL State (STATE) bit field in the Control A (ERRCTRL.CTRLA) register is set to Fault State (FAULT), thereby transferring the MCU to a safe state: • The Error Controller Reset Request is asserted. The Error Controller Reset Flag (ECRF) in the Reset Flag Register (RSTCTRL.RSTFR) register is set to '1'. • The heartbeat is stopped • The I/O float mechanism is activated and will float all I/O pins
01	RESERVED	
10	NONCRITICAL	The system has received a NONCRITICAL error. • The Noncritical Error Detected (NONCRITICAL) bit in the ERRCTRL.STATUSA register is set • The appropriate bit in ERRCTRL.ESF is set • The STATE bit field in the ERRCTRL.CTRLA register is set to Alarm State (ALARM), and an interrupt or NMI request is asserted • The I/O float mechanism is activated and will float all I/O pins if the Float all I/O Pins (FLOAT) bit in the Error Source Control x (ERRCTRL.ESCx) register is set.
11	NOTIFICATION	The system has received a NOTIFICATION error. • The Notification Error Detected (NOTIFICATION) bit in the ERRCTRL.STATUSA register is set • The appropriate bit in ERRCTRL.ESF is set • The I/O float mechanism is activated and will float all I/O pins if the FLOAT bit in ERRCTRL.ESCx is set.

22.3.3 System States

The ERRCTRL can be in one of four different states:

- **CONFIG**: The timeout timer starts if enabled, and Error Source Control x (ERRCTRL.ESCx) registers can be written. Error Status Flags (ESF) will be set upon receiving signals from the system, but no error action or state transition is performed. The Error Controller will transition to a

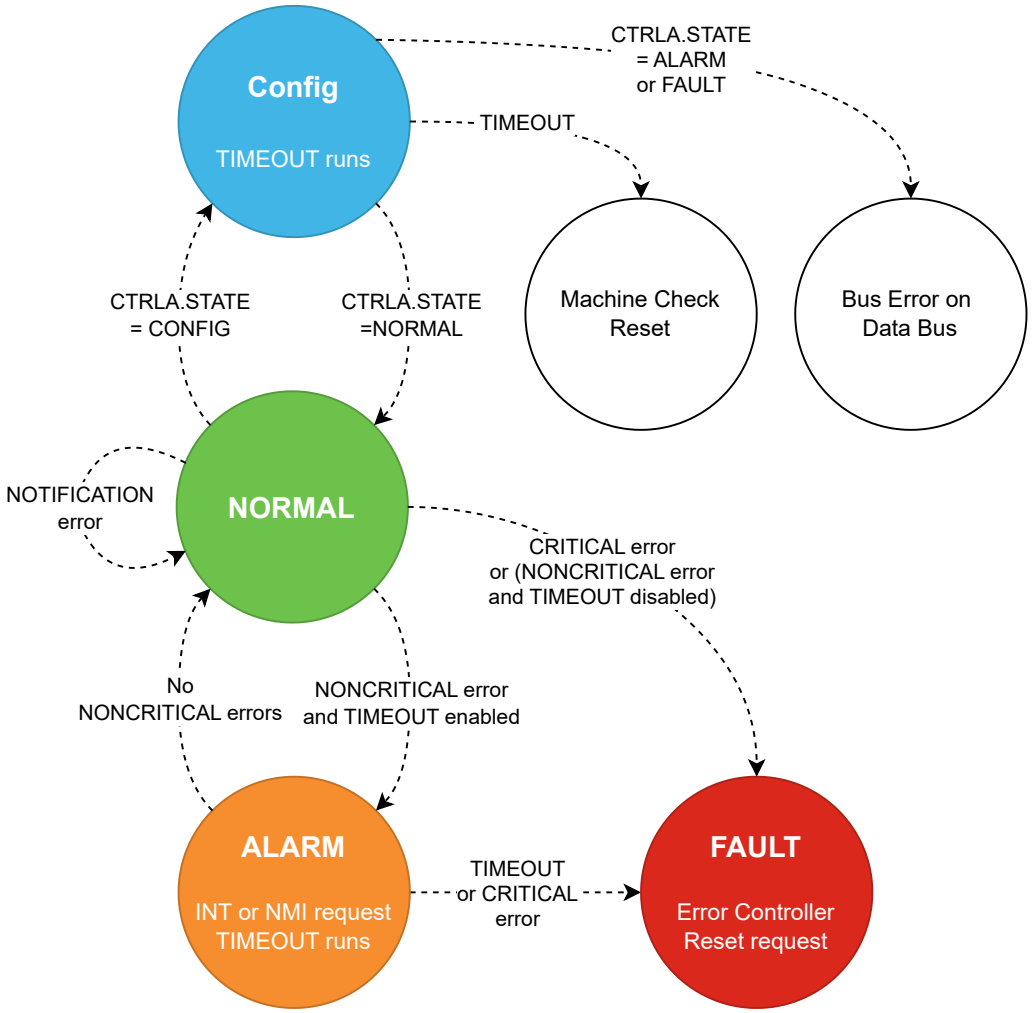
NORMAL state when writing the ERRCTRL State (STATE) bit field in the Control A (ERRCTRL.CTRLA) register to Normal State (NORMAL). If the Timeout Counter (ERRCTRL.TIMECNT) register value exceeds the value in the Timeout Value (ERRCTRL.TIMEOUT) register, a Machine Check Reset is automatically triggered. Any active I/O Float, INT, NMI or Machine Check Reset when the ERRCTRL entered the CONFIG state is left unchanged while in the CONFIG state. New errors will not be serviced when in the CONFIG state. Any new error will set their ESF, but no I/O Float, INT, NMI or Machine Check Reset will occur.

The expected error handling (I/O float, INT/NMI, or Machine Check Reset, depending on the severity of the error) will be postponed until transitioning from the CONFIG to NORMAL state. This may increase the Fault Detection Time Interval (FDTI), depending on how long the ERRCTRL is in the CONFIG state.

Any heartbeat is paused while in the CONFIG state, i.e., no transitions will occur on the heartbeat signal. It is possible to transition to CONFIG by writing the STATE bit field in ERRCTRL.CTRLA to CONFIG. This is only allowed from the NORMAL state, an attempt to transition from any other state will be discarded and will result in a Bus Error. The Force I/O Float (FORCEFLOAT) bit in the ERRCTRL.CTRLA register and external float requests will float I/Os even in the CONFIG state.

- **NORMAL**: ERRCTRL.ESCx registers cannot be written, but bits in the Error Status Flag (ESF) register can be set. It is possible to transition to CONFIG by writing the STATE bit field in ERRCTRL.CTRLA to CONFIG. CRITICAL errors detected while in NORMAL state will automatically transition to FAULT state, and noncritical errors will automatically transition to ALARM state.
- **ALARM**: ERRCTRL.ESCx registers cannot be written. Once entering the ALARM state, the timeout timer starts if the feature is enabled. The state will automatically transition to FAULT if the ERRCTRL.TIMECNT value exceeds the value in ERRCTRL.TIMEOUT and will automatically transition to NORMAL when all noncritical errors have been cleared in the ERRCTRL.ESF register.
  This state allows the system to possibly fix the fault and return to normal operation instead of going directly to FAULT state. This timeout is a safety mechanism in case the ALARM interrupt is never serviced, e.g., due to a software or Interrupt Controller (INTCTRL) error. A CRITICAL error received in the ALARM state will automatically transit to FAULT.
- **FAULT**: An Error Controller Reset is requested immediately. The Error Controller Reset Flag (ECRF) in the Reset Flag Register (RSTCTRL.RSTFR) becomes set. Any heartbeat will stop. The Float I/O Pins (FLOAT) bit in the Status A (ERRCTRL.STATUSA) register is set, and all I/O pins are floated.

Figure 22-3. Module States



22.3.4 Timeout Timer

The timeout timer automatically transitions the system state from ALARM to FAULT when it counts down to zero. It also triggers a Machine Check Reset if the ERRCTRL stays in CONFIG state longer than specified in the Timeout Value (ERRCTRL.TIMEOUT) register. The ERRCTRL.TIMEOUT register is write-protected and can only be written in the CONFIG state.

The timeout value is TIMEOUT*4 clock cycles for a maximum of 1020 such cycles.

A value of zero disables the timeout mechanism. A NONCRITICAL error will directly cause an entry to the FAULT state in this setting. The ALARM state is unused in this case.

Writing to ERRCTRL.TIMEOUT will not reset the timeout counter. A value written to ERRCTRL.TIMEOUT will be reloaded into the timeout counter when the ERRCTRL enters the NORMAL state, preventing the possibility that repetitive writes to ERRCTRL.TIMEOUT due to a software or hardware error cause a timeout to never happen.

Reading the Timeout Counter (ERRCTRL.TIMECNT) register returns the current value of the counter.

⚠WARNING Disabling the timeout function is only recommended at system startup during Error Controller configuration.

22.3.5 CPU Dual Core Lockstep Comparator Error

A mismatch from the Dual Core Lockstep Comparators is not handled by or routed to the ERRCTRL. The Reset Controller handles this error directly and will cause an immediate reset because the CPU is deemed unstable and cannot be expected to perform any operation reliably. Refer to the *RSTCTRL - Reset Controller* section for additional details.

22.3.6 Clearing Error Signals

After an error has been handled (in a way that depends on the source of the error), the error flag must be cleared at its source. The clearing procedure for an error flag is specific to the failing module. Typically, this happens by clearing the Interrupt Flags (peripheral.INTFLAGS) register in the failing module. This may be unnecessary if the error caused the entry to Safe State (i.e., a reset) since a reset will typically clear most interrupt flags. The Error Status Flags (ERRCTRL.ESF) register must also be cleared.

Clearing the failing module error input must happen before clearing the ERRCTRL.ESF register since the ERRCTRL.ESF is just a registered version of the error input from the failing module, which usually means that the failing module's peripheral.INTFLAGS register must be cleared before the ERRCTRL.ESF is cleared.

22.3.7 Heartbeat

A heartbeat signal is enabled by writing '1' to the Heartbeat Enable (HEART) bit in the Control A (ERRCTRL.CTRLA) register. A 1 kHz tap from the OSC32K is used to generate a nominal 1 kHz heartbeat. The heartbeat is asynchronous to the ERRCTRL clock.

The heartbeat signal drives the I/O pin in an open-drain configuration, i.e., it drives hard to ground in the "low" phase but releases the driver in the "high" phase, i.e., a tri-state. An internal or external pull-up is required to pull the pin to a high level.

Once enabled, the heartbeat toggles in all ERRCTRL states except FAULT and CONFIG:

- When the ERRCTRL transitions to the FAULT state, the heartbeat is immediately disabled by tri-stating the I/O pin. This immediate release may cause the last heartbeat period on the pin to be shorter than expected.
- While the ERRCTRL is in the CONFIG state, the heartbeat output signal is forced to a logic level low. Toggling will resume when the CONFIG state is exited, with the following possible behaviors:
  - A low spike on the heartbeat if CONFIG mode is entered when the pin is high (due to forcing it to '0' when in CONFIG) and exited before the next toggle
  - Shorter or longer period if a toggle should have happened
  - No effect if entering CONFIG while the heartbeat is low and exiting before the heartbeat should toggle

The application can explicitly signal a failure to the surrounding system by writing HEART in ERRCTRL.CTRLA to '0', causing the heartbeat to stop.

22.3.8 Automatic Float of I/O Pins

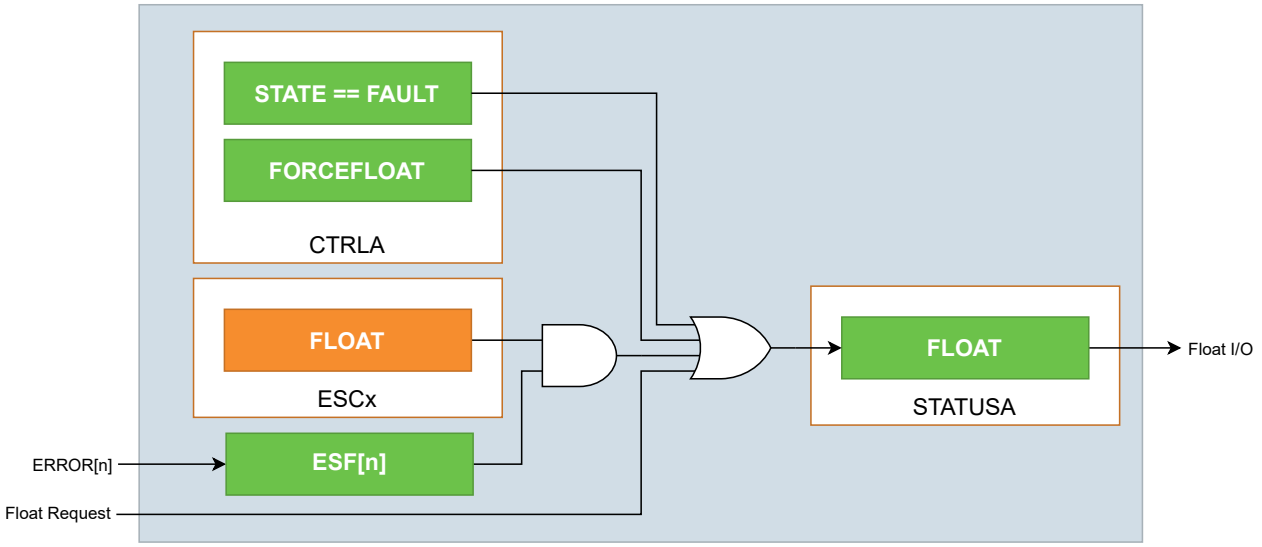
Each Error Source Control (ERRCTRL.ESCx) register has a Float All I/O Pins (FLOAT) bit. If the FLOAT bit is set to ENABLE, and an error is received on that channel, then the Float I/O Pins (FLOAT) bit in the Status A (ERRCTRL.STATUSA) register will be set automatically. All I/O pins are forced to a floating

state (tri-state) when the FLOAT bit in ERRCTRL.STATUSA is set. The FLOAT bit in ERRCTRL.STATUSA is also automatically set when the ERRCTRL enters the FAULT state.

The I/O pins will remain floated when the FLOAT bit in the ERRCTRL.STATUSA register is set. Floating the I/O pins will also float any pin that outputs the heartbeat signal.

Writing a '1' to the Force I/O Float (FORCEFLOAT) bit in the Control A (ERRCTRL.CTRLA) register will immediately cause the I/O pins to float.

Figure 22-4. Drivers of I/O Float



22.3.9 Error Injection

The application can provoke an error using an error injection mechanism to verify that the error detection systems work as intended. Each error channel has a bit in the Error Status Flag Test Injection (ERRCTRL.ESFTEST) register. Error injection is shown in the System Integration figure in the *Overview* section, displaying X and Y systems connected to the ERRCTRL.

- System X has its proprietary mechanism to inject and clear errors, typically through writing to Test Control bit(s) in a register in the system/peripheral. The ECC error channels are examples of this. The ERRCTRL.ESFTEST output from the ERRCTRL is OR'ed with the module's error signal (error) to make the modified error signal (error*) so the ERRCTRL can test its internal circuitry.
- System Y has no mechanism for injecting errors. The ERRCTRL.ESFTEST output from the ERRCTRL is OR'ed with the module's error signal (error) to make the modified error signal (error*) so the ERRCTRL can test its internal circuitry.

Error injection as a diagnostic measure can have severe side effects on the system, depending on the severity level of the injected error. If the error is configured with severity level CRITICAL, injecting an error will cause the MCU to reset, which may not be desirable or possible in the system. This can be mitigated by temporarily reconfiguring the error channel to a severity level that has less severe side effects, such as NOTIFICATION, when performing the error injection, as follows:

1. Place the ERRCTRL into CONFIG mode by writing to the ERRCTRL State (STATE) bit field in the Control A (ERRCTRL.CTRLA) register.
2. Reconfigure the error channel to inject error into by writing its Error Severity Level (ERRLVL) bit field in the Error Source Control (ERRCTRL.ESCx) register to a less severe setting, e.g., NOTIFICATION.
3. Leave CONFIG mode and enter NORMAL mode by writing to the STATE bit field in ERRCTRL.CTRLA.

4. Inject the error and observe the response in the ERRCTRL.
5. Clear any error flags at the source and in the Error Status Flags (ESF) register.
6. Place the ERRCTRL into CONFIG mode by writing to the STATE bit field in ERRCTRL.CTRLA.
7. Reconfigure the error channel to its original severity level by writing the ERRLVL bit field in ERRCTRL.ESCx to its original setting.
8. Leave CONFIG mode and enter NORMAL mode by writing to ERRCTRL.CTRLA.

If the time required to perform the fault injection is less than the Timeout Value (ERRCTRL.TIMEOUT) register, or the timeout mechanism is disabled, the ERRCTRL can be in a CONFIG state during the entire operation, i.e., steps 3 and 6 can be omitted.

22.3.10 Protecting Register Contents

Any attempted write to the Timeout Value (ERRCTRL.TIMEOUT), Error Source Control x (ERRCTRL.ESCx), or Error Status Flag Test Injection (ERRCTRL.ESFTEST) registers when not in the CONFIG state will be discarded and return a Bus Error response on the data bus.

Transition to CONFIG is only allowed from the NORMAL state by writing to the ERRCTRL State (STATE) bit field in the Control A (ERRCTRL.CTRLA) register. An attempted transition from any other state will be discarded and return a Bus Error.

Attempted writes to a CCP-protected register outside of the CCP sequence will be discarded and return a Bus Error response on the data bus. A CCP sequence protects the following registers:

- ERRCTRL.CTRLA
- ERRCTRL.ESFTEST

22.3.11 Data Bus Accesses

Attempted access to any unused register address within the ERRCTRL address space will be discarded and return a Bus Error.

22.3.12 Internal Consistency Checks

The ERRCTRL has internal consistency checks and can detect several internal errors. In particular, the internal state registers are duplicated, and the duplicated copies are monitored continuously for any mismatch. If such an error is detected, the internal state is inconsistent and cannot be trusted, and a Machine Check Reset request is sent to the Reset Controller. The reset request will cause an immediate entry to a safe state. For more information, refer to the *Reset Controller* section.

As a consequence of the immediate Reset, all I/O pins will be floated, and any heartbeat will stop due to the Control A (ERRCTRL.CTRLA) register Reset.

The assumption is that a reset will clear whatever condition caused the internal error. If the error persists, the MCU will enter an endless reset sequence.

22.3.13 Reset Requests

The ERRCTRL can request two types of reset:

Table 22-2. Reset Types from Error Controller

Type	Flag in RSTCTRL	Trigger
Machine Check Reset	Machine Check Reset Flag (MCRF) in the Reset Flag Register (RSTCTRL.RSTFR). Error controller Internal Error (EC) in the Machine Check Flags A (RSTCTRL.MCFLAGSA) register.	TIMEOUT in CONFIG state Internal consistency error
Error Controller Reset	Error Controller Reset flag (ECRF) in RSTCTRL.RSTFR	ERRCTRL in FAULT state

22.3.14 Effect of Reset on Registers

Errors of severity CRITICAL will reset the system. To enable the identification of the reset cause, the Reset Cause (ERRCTRL.CAUSE) register within the ERRCTRL is exclusively reset by a Power-On Reset (POR) but remains unaffected by any other type of reset.

22.3.15 Error Channel Connections

The subsystems that can generate an error are connected to the ERRCTRL channels, as described in the table below. The **CRITICAL Recommended** column identifies which errors are typically recommended to be configured with critical severity. The channels may be configured differently depending on application needs.

Table 22-3. Error Channel Connections

Channel Number	Name	Peripheral	Source	Error	CRITICAL Recommended
0	VREGFAIL	SLPCTRL	VOV flag in INTFLAGS	The Voltage Regulator Monitor has detected high voltage	X
			VUV flag in INTFLAGS	The Voltage Regulator Monitor has detected low voltage	
1	BUSERR	CPU	BUSERR flag in INTFLAGS	Illegal address, parity error or other error response from the bus target	X
			PARITYD flag in INTFLAGS	Parity error on the data bus	
2	RAM2	RAMCTRL	COMP flag in INTFLAGS	ECC comparator error	X
			ECC2 flag in INTFLAGS	RAM ECC multiple-bit error	
3	FLASH2	NVMCTRL	COMP flag in INTFLAGSB	ECC comparator error	X
			FECC2 flag in INTFLAGSB	Flash ECC double-bit error	
4	OPC	CPU	OPC flag in INTFLAGS	Illegal opcode instruction	X
			PARITYI flag in INTFLAGS	Parity error on instruction bus	
5	SPLIM	CPU	SPLIM flag in INTFLAGS	Stack pointer limit exceeded	
6	RAM1	RAMCTRL	ECC1 flag in INTFLAGS	RAM ECC single-bit error	
7	FLASH1	NVMCTRL	FECC1 flag in INTFLAGSB	Flash ECC single-bit error	
8	VREGWARN	SLPCTRL	VDENTER flag in INTFLAGS	VMON has entered diagnostic mode	
			VDEXIT flag in INTFLAGS	VMON has exited diagnostic mode	
			VERR flag in INTFLAGS	Internal VMON error	
			VSLP flag in INTFLAGS	VMON has entered sleep mode	
			VDIS flag in INTFLAGS	VMON is disabled	
			SERR flag in INTFLAGS	Sleep instruction aborted due to an illegal condition	
9	CFD0	CLKCTRL	CFD0 flag in INTFLAGS	Clock failure detected 0	
10	CFD1	CLKCTRL	CFD1 flag in INTFLAGS	Clock failure detected 1	
11	CFM0	CLKCTRL	CFM0 flag in INTFLAGS	Clock frequency measurement error 0	
12	CFM1	CLKCTRL	CFM1 flag in INTFLAGS	Clock frequency measurement error 1	
13	SWDT	SWDT	ERROR flag in INTFLAGS	SWDT error	

.....continued					
Channel Number	Name	Peripheral	Source	Error	CRITICAL Recommended
14	EEPROM	NVMCTRL	EECC2 flag in INTFLAGSB	EEPROM multiple-bit ECC error	
			EECC1 flag in INTFLAGSB	EEPROM single-bit ECC error	
15	EVSYS0	EVSYS0	EVSYS0	Error Controller event system input 0	
16	EVSYS1	EVSYS1	EVSYS1	Error Controller event system input 1	

22.3.16 Imprecise Errors

All error sources (unless explicitly stated otherwise) are imprecise, meaning the error is flagged one or more cycles after it occurred. For the error detection mechanisms timing, refer to the documentation for each error source. For imprecise errors, it may be impossible to know which event or which instruction triggered them. Such errors may be hard or impossible to recover from and may require a severity of CRITICAL, depending on the application.

22.3.17 Errors During System Initialization

Any errors detected during the boot process, such as ECC or bus error, will be handled by the system start-up mechanism and cause the system to never exit from the boot sequence. The I/O pins will be floated.

22.3.18 Error Controller and System Startup

At system start-up, the ERRCTRL is configured as follows:

- The ERRCTRL state is NORMAL
- The Timeout Value (ERRCTRL.TIMEOUT) register contains '0xFF' (255 decimal), so the timeout feature is enabled, and timeout will occur after (255*4) = 1020 clock cycles
- All Error Source Control x (ERRCTRL.ESCx) registers contain 0x82, i.e., FLOAT is '1', and ERRLVL is NONCRITICAL

The consequences are:

- The ERRCTRL is in the NORMAL state when Reset is released, and the MCU starts executing instructions
- Errors detected and reported to the ERRCTRL during this period will lead to the appropriate Error Status Flag (ESF) being set, and the error is treated as a NONCRITICAL error, which will:
  - Transfer the ERRCTRL to the ALARM state
  - Raise an interrupt request since the reset value of the ALARM State Interrupt Type (INTTYPE) bit in the Control A (ERRCTRL.CTRLA) register is INT
  - This request will be left pending since the reset value of the Global Interrupt Enable bit in the CPU's Status Register (CPU.SREG) is '0'
- The timeout mechanism is enabled, and after a period of (255*4) clock cycles, any received and unhandled NONCRITICAL error will be escalated, transferring the ERRCTRL to the FAULT state and causing a safe state entry through a Machine Check Reset
- The start-up and system initialization code needs to reconfigure the ERRCTRL and examine any reported errors detected during the system start-up phase that has set any ESF within (255*4) clock cycles. Reconfiguration will typically require:
  - Configuring the ERRCTRL.ESCx registers to the desired value
  - Writing INTTYPE bit in ERRCTRL.CTRLA to the desired value
  - Writing '1' to the Global Interrupt Enable bit in CPU.SREG so NONCRITICAL errors can be handled by an interrupt, which is only relevant if the INTTYPE bit in ERRCTRL.CTRLA is INT

22.3.19 Interrupts

Table 22-4. Available Interrupt Vectors and Sources

Vector Name	Source Name	Condition	Dependency
NMI	NMINONCRITICAL	The Error Controller has received a Noncritical error	The ALARM State Interrupt Type (INTTYPE) bit in the Control A (ERRCTRL.CTRLA) register is '1' (NMI)
ERRCTRL_INT	NONCRITICAL		The INTTYPE bit in ERRCTRL.CTRLA is '0' (INT)

22.3.19.1 Handling of Interrupts in Systems with Error Controller

A detected error is reported to both the Interrupt Controller and the Error Controller, resulting in a hazard since the interrupt controller may receive two interrupt requests originating from the same error source. These two interrupt requests will reach the interrupt controller in different clock cycles due to register delays in the Error Controller.

Write the interrupt handler to handle this hazard safely so the error is handled correctly and only once. The following pseudocode describes how to do this:

```
;; Interrupt vectors
RESET_VECTOR: jump START_PROGRAM
NMI_VECTOR: jump NMI_HANDLER
ERRCTRL_HANDLER: jump ERROR_HANDLER
<HW_PART_ERR_HANDLER>: jump ERROR_HANDLER
....
NMI_HANDLER:
    Perform desired actions
    Software Reset
ERROR_HANDLER:
    Clear Error Flag in INTFLAGS
    Clear ESF flag in ERRCTRL
    Perform other actions
    Return from interrupt OR Software Reset
START_PROGRAM:
    Execute program
```

The ERRCTRL can be programmed to request an NMI instead of an interrupt, creating another hazard, as the hardware part will request an interrupt while the ERRCTRL will request an NMI. The NMI may arrive later at the Interrupt Controller and possibly disturb the already-started execution of the interrupt. For this reason, if the ERRCTRL is configured to request an NMI instead of an interrupt, the error handler must always exit with a software reset and not a return from interrupt.

22.3.20 Events

Two error channels are connected to the event system as event inputs, allowing events to trigger ERRCTRL handling using the ordinary ERRCTRL configuration mechanisms.

The event system channel multiplexer is disconnected at reset, so any events will not propagate into the ERRCTRL before the event system has been configured and the input to the ERRCTRLis enabled.

22.3.21 Sleep Mode Operation

The Error Controller needs a clock and does not function in sleep modes without a clock. The only sleep mode where the ERRCTRL is clocked is in IDLE. The ERRCTRL is, therefore, not able to register or act on errors detected in POWERDOWN or STANDBY sleep modes. As a result, diagnostic coverage in these two sleep modes is limited or absent. The higher-ranking system is assumed to be designed to accept that the diagnostic coverage from the MCU is lower when the MCU is in a sleep mode.

The ERRCTRL will resume ordinary operation when waking from POWERDOWN or STANDBY.

The heartbeat stops in POWERDOWN and STANDBY sleep modes.

22.3.22 Functional Safety

The Error Controller is designed for use in functional safety systems.

22.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0			FORCEFLOAT	HEART	INTTYPE		STATE[1:0]	
0x01	STATUSA	7:0	FLOAT	IOSAFE				CRITICAL	NONCRITICAL	NOTIFICATION
0x02	TIMEOUT	7:0	TIMEOUT[7:0]							
0x03	TIMECNT	7:0	TIMECNT[7:0]							
0x04	CAUSE	7:0				CAUSE[4:0]				
0x05 ... 0x0F	Reserved									
0x10	ESCVREGFAIL	7:0	FLOAT						ERRLVL[1:0]	
0x11	ESCBUSERR	7:0	FLOAT						ERRLVL[1:0]	
0x12	ESCRAM2	7:0	FLOAT						ERRLVL[1:0]	
0x13	ESCFLASH2	7:0	FLOAT						ERRLVL[1:0]	
0x14	ESCOPC	7:0	FLOAT						ERRLVL[1:0]	
0x15	ESCSPLIM	7:0	FLOAT						ERRLVL[1:0]	
0x16	ESCRAM1	7:0	FLOAT						ERRLVL[1:0]	
0x17	ESCFLASH1	7:0	FLOAT						ERRLVL[1:0]	
0x18	ESCVREGWARN	7:0	FLOAT						ERRLVL[1:0]	
0x19	ESCCFD0	7:0	FLOAT						ERRLVL[1:0]	
0x1A	ESCCFD1	7:0	FLOAT						ERRLVL[1:0]	
0x1B	ESCCFM0	7:0	FLOAT						ERRLVL[1:0]	
0x1C	ESCCFM1	7:0	FLOAT						ERRLVL[1:0]	
0x1D	ESCSWDT	7:0	FLOAT						ERRLVL[1:0]	
0x1E	ESCEEPROM	7:0	FLOAT						ERRLVL[1:0]	
0x1F	ESCEVSYS0	7:0	FLOAT						ERRLVL[1:0]	
0x20	ESCEVSYS1	7:0	FLOAT						ERRLVL[1:0]	
0x21 ... 0x2F	Reserved									
0x30	ESF	7:0	ESF[7:0]							
		15:8	ESF[15:8]							
		23:16								ESF[16]
		31:24								
0x34	ESFTEST	7:0	ESFTEST[7:0]							
		15:8	ESFTEST[15:8]							
		23:16								ESFTEST[16]
		31:24								

22.5 Register Description

22.5.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x01
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
			FORCEFLOAT	HEART	INTTYPE		STATE[1:0]	
Access			R/W	R/W	R/W		R/W	R/W
Reset			0	0	0		0	1

Bit 5 – FORCEFLOAT Force I/O Float
Writing this bit to '1' will force I/O pins to float. The Float I/O Pins (FLOAT) bit in the Status A (ERRCTRL.STATUSA) register will be set as long as FORCEFLOAT is set.

Value	Name	Description
0x0	DISABLE	Disabled
0x1	ENABLE	Enabled

Bit 4 – HEART Heartbeat Enable
Writing this bit to '1' will enable the Heartbeat output.

Value	Name	Description
0x0	DISABLE	Disabled
0x1	ENABLE	Enabled

Bit 3 – INTTYPE ALARM State Interrupt Type
Selects the type of interrupt request asserted when the ALARM state is entered.

Value	Name	Description
0x0	INT	INT
0x1	NMI	NMI

Bits 1:0 – STATE[1:0] ERRCTRL State
The ERRCTRL current state. The software can only write values CONFIG or NORMAL. An attempted write of ALARM or FAULT will be discarded, and a Bus Error response returned on the data bus. An attempted write of CONFIG from any state other than NORMAL will be discarded, and a Bus Error response returned on the data bus.

Value	Name	Description
0x0	CONFIG	Configuration state
0x1	NORMAL	Normal state
0x2	ALARM	Alarm state
0x3	FAULT	Fault state

22.5.2 Status A

Name: STATUSA
Offset: 0x01
Reset: 0xXX
Property: -

Bit	7	6	5	4	3	2	1	0
	FLOAT	IOSAFE				CRITICAL	NONCRITICAL	NOTIFICATION
Access	R	R				R	R	R
Reset	x	x				x	x	x

Bit 7 – FLOAT Float I/O pins
I/O pins are forced to a floating (tri-stated) state when this flag is '1'. This flag is set automatically whenever an error is detected on channel n and Float All I/O Pins (FLOAT) bit in the Error Source Control x (ERRCTRL.ESCx) register is set. This bit is also set automatically if the ERRCTRL enters the FAULT state.

Value	Name	Description
0x0	NORMAL	I/O pins are not forced to floating state
0x1	FLOAT	I/O pins are forced to floating state

Bit 6 – IOSAFE All I/O pins are in safe state
Set if all I/O pins are in a safe state, i.e., output driver disabled with internal pullup and pulldown resistors disabled.

Value	Name	Description
0x0	NOSAFE	One or more I/O pins are not in safe state
0x1	SAFE	All I/O pins are in a safe state

Bit 2 – CRITICAL Critical Error Detected
Set when one or more Critical errors are detected.

Value	Name	Description
0x0	NOERROR	There are no Critical errors in the system
0x1	ERROR	There are one or more Critical errors in the system

Bit 1 – NONCRITICAL Noncritical Error Detected
Set when one or more Noncritical errors are detected.

Value	Name	Description
0x0	NOERROR	There are no Noncritical errors in the system
0x1	ERROR	There are one or more Noncritical errors in the system

Bit 0 – NOTIFICATION Notification Error Detected
Set when one or more Notification errors are detected.

Value	Name	Description
0x0	NOERROR	There are no Notification errors in the system
0x1	ERROR	There are one or more Notification errors in the system

22.5.3 Timeout Value

Name: TIMEOUT
Offset: 0x02
Reset: 0xFF
Property: -

Bit	7	6	5	4	3	2	1	0
	TIMEOUT[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 – TIMEOUT[7:0] Timeout Value
Determines the timeout value for triggering a machine check reset due to overstay in the CONFIG state and automatic transition from ALARM to FAULT state. The actual timeout value is TIMEOUT*4 clock cycles. A zero value disables the timeout mechanism.
The TIMEOUT counter is frozen when the CPU halts in debug mode as long as the device is NOT LOCKED
This register is only writable when the ERRCTRL is in the CONFIG state. A value written to the TIMEOUT register will be loaded into the timeout counter when the ERRCTRL enters the NORMAL state.

22.5.4 Timeout Counter

Name: TIMECNT
Offset: 0x03
Reset: 0xFF
Property: -

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

Bits 7:0 – TIMECNT[7:0] Timeout Counter Value
Reading TIMECNT returns the current counter value.

22.5.5 Reset Cause

Name: CAUSE
Offset: 0x04
Reset: 0xXX
Property: -

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

Bits 4:0 – CAUSE[4:0] Reset Cause
Reading CAUSE returns the channel number that caused the Error Controller Reset. This bit field may report a channel configured to CRITICAL severity or a channel configured to NONCRITICAL severity escalated to CRITICAL due to a timeout from the timeout counter.
This register is reset only by a Power-On Reset (POR); other types of system reset have no effect.

22.5.6 Error Source Control x

Name: ESCx
Offset: 0x10 + x*0x01 [x=0..16]
Reset: 0x82
Property: -

Refer to the *Register Summary* section for all available Error Source Control x registers.

These registers are locked when the ERRCTRL State (STATE) bit field in the Control A (ERRCTRL.CTRLA) register is anything other than CONFIG. An attempted write outside the CONFIG state will return a Bus Error response.

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

Bit 7 – FLOAT Float all I/O pins
Determines if the FLOAT bit in STATUSA is set automatically when an error is detected on channel *n*.

Value	Name	Description
0x0	DISABLE	The Float I/O Pins (FLOAT) bit in the Status A (ERRCTRL.STATUSA) register is not automatically set when an error is reported on this channel
0x1	ENABLE	The FLOAT bit in STATUSA is automatically set when an error is reported on this channel. This value will always be used when ERRLVL = CRITICAL.

Bits 1:0 – ERRLVL[1:0] Error Severity Level
The Severity Level associated with the error channel.

Value	Name	Description
0x0	CRITICAL	Critical error
0x1	RESERVED	Reserved, decoded as CRITICAL
0x2	NONCRITICAL	Noncritical error (reset value)
0x3	NOTIFICATION	Notification error

22.5.7 Error Status Flags

Name: ESF
Offset: 0x30
Reset: 0xXXXXXXXX
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								ESF[16]
Access								R/W
Reset								x

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

Bit	7	6	5	4	3	2	1	0
	ESF[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 16:0 – ESF[16:0] Error Status Flags

Error Status Flags. If a bit is set, the corresponding channel reports an error. Write to '1' to clear the error flag. Writing the flag does not clear the error condition in the error source. If a bit in this register is written to '1' while the error is still present, the flag will remain set.

22.5.8 Error Status Flag Test Injection

Name: ESFTEST
Offset: 0x34
Reset: 0x00
Property: Configuration Change Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								ESFTEST[16]
Access								W
Reset								0

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

Bit	7	6	5	4	3	2	1	0
	ESFTEST[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 16:0 – ESFTEST[16:0] Error Status Flag Test Injection
Writing a bit to '1' outputs a one-clock-wide pulse on the corresponding error injection input for the channel, causing the corresponding Error Status Flag (ESF) to be set.

23. EVSYS - Event System

23.1 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 6 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

23.2 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, the EVSYS peripheral makes it possible to implement Core Independent Peripherals (CIPs). Also, it is a powerful tool for reducing the complexity, size, and execution time of the software.

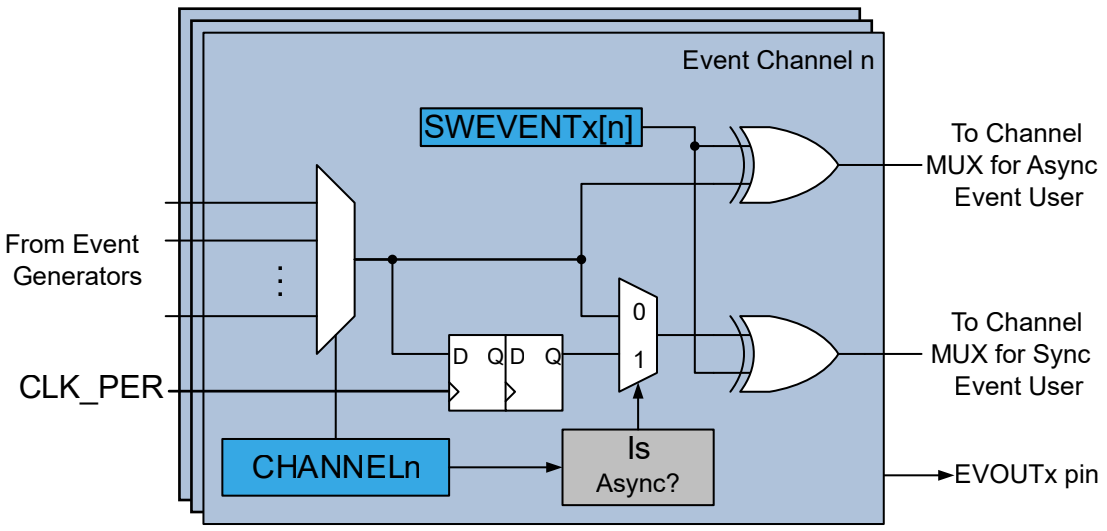
A change of the event generator's state is referred to as an event and usually corresponds to one of the peripheral's interrupt conditions. Events can be forwarded directly to other peripherals using the dedicated event routing network. The routing of each channel is configured in software, including 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.

23.2.1 Block Diagram

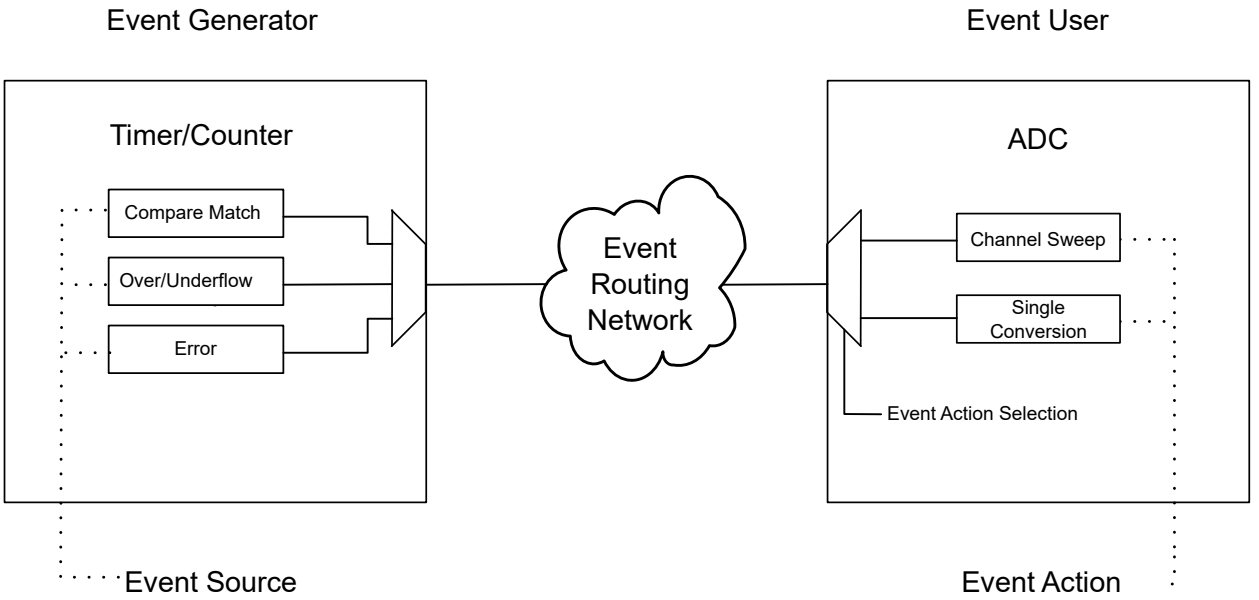
Figure 23-1. EVSYS Block Diagram



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 23-2. Example of Event Source, Generator, User, and Action



23.2.2 Signal Description

Signal	Type	Description
EVOUTx	Digital output	Event output, one output per I/O Port

23.3 Functional Description

23.3.1 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 Selection (EVSYS.USERx) register.

23.3.2 Operation

23.3.2.1 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.

23.3.2.2 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.

23.3.2.3 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 23-1. 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 23-2. 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
WDT	TICK	WDT LSB Tick	Level	CLK_WDT	
SWDT	ERROR	SWDT Error	Pulse	CLK_PER	
MVIO	VDDIO2OK	VDDIO2 is OK	Level	CLK_PER	
RTC	OVF	Overflow	Pulse	CLK_RTC	One CLK_RTC period
	CMP	Compare Match			
	EVGENn	Selectable prescaled RTC event	Level		Prescaled CLK_RTC period
CCL	LUTn	LUT output level	Level	Asynchronous	Depends on the CCL configuration

.....continued

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
ACn	OUT	Comparator output level	Level	Asynchronous	Given by the AC output level
ADCn	RESRDY	Result ready	Pulse	CLK_PER	One CLK_PER period
	SAMPRDY	Sample ready			
	WCMP	Window compare			
ZCDn	OUT	ZCD output level	Level	Asynchronous	Given by the ZCD output level
PORTx	EVGENn	Pin level	Level	Asynchronous	Given by the pin level
USARTn	XCK	USART baud clock	Level	CLK_PER	Minimum two CLK_PER periods
SPIn	SCK	SPI host 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			
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			

23.3.2.4 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 23-3. 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

.....continued		
Input Detection	**Async/Sync**	**Description**
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 23-4. 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 the event signal to the pin	No detection	Async
USARTn	IRDA	IrDA mode input	Level	Sync
TCAn	CNTA	Count on positive event edge	Edge	Sync
		Count on any event edge	Edge	
		Count while the 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			
ERRCTRL	EVENT0	Event input	Level	Sync
	EVENT1			
CLKCTRL	CFD	Clock Failure Detection	Edge	Async
	CFM	Clock Frequency Measurement	Edge	Async

23.3.2.5 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.

23.3.2.6 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.

23.3.3 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.

23.3.4 Debug Operation

This peripheral is unaffected by entering Debug mode.

23.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	SWEVENTA	7:0			CH5	CH4	CH3	CH2	CH1	CH0
0x01 ... 0x0F	Reserved									
0x10	CHANNEL0	7:0	CHANNEL[7:0]							
0x11	CHANNEL1	7:0	CHANNEL[7:0]							
0x12	CHANNEL2	7:0	CHANNEL[7:0]							
0x13	CHANNEL3	7:0	CHANNEL[7:0]							
0x14	CHANNEL4	7:0	CHANNEL[7:0]							
0x15	CHANNEL5	7:0	CHANNEL[7:0]							
0x16 ... 0x1F	Reserved									
0x20	USERCCLLUT0A	7:0	USER[7:0]							
0x21	USERCCLLUT0B	7:0	USER[7:0]							
0x22	USERCCLLUT1A	7:0	USER[7:0]							
0x23	USERCCLLUT1B	7:0	USER[7:0]							
0x24	USERCCLLUT2A	7:0	USER[7:0]							
0x25	USERCCLLUT2B	7:0	USER[7:0]							
0x26	USERCCLLUT3A	7:0	USER[7:0]							
0x27	USERCCLLUT3B	7:0	USER[7:0]							
0x28	USERCCLLUT4A	7:0	USER[7:0]							
0x29	USERCCLLUT4B	7:0	USER[7:0]							
0x2A	USERCCLLUT5A	7:0	USER[7:0]							
0x2B	USERCCLLUT5B	7:0	USER[7:0]							
0x2C	USERADC0START	7:0	USER[7:0]							
0x2D	USERADC1START	7:0	USER[7:0]							
0x2E	USEREVSYSEVOUTA	7:0	USER[7:0]							
0x2F	USEREVSYSEVOUTC	7:0	USER[7:0]							
0x30	USEREVSYSEVOUTD	7:0	USER[7:0]							
0x31	USEREVSYSEVOUTF	7:0	USER[7:0]							
0x32	USERUSART0IRDA	7:0	USER[7:0]							
0x33	USERUSART1IRDA	7:0	USER[7:0]							
0x34	USERUSART2IRDA	7:0	USER[7:0]							
0x35	USERTCA0CNTA	7:0	USER[7:0]							
0x36	USERTCA0CNTB	7:0	USER[7:0]							
0x37	USERTCB0CAPT	7:0	USER[7:0]							
0x38	USERTCB0COUNT	7:0	USER[7:0]							
0x39	USERTCB1CAPT	7:0	USER[7:0]							
0x3A	USERTCB1COUNT	7:0	USER[7:0]							
0x3B	USERTCB2CAPT	7:0	USER[7:0]							
0x3C	USERTCB2COUNT	7:0	USER[7:0]							
0x3D	USERTCB3CAPT	7:0	USER[7:0]							
0x3E	USERTCB3COUNT	7:0	USER[7:0]							
0x3F	USERTCD0INPUTA	7:0	USER[7:0]							
0x40	USERTCD0INPUTB	7:0	USER[7:0]							
0x41	USERERRCTRLEVENT0	7:0	USER[7:0]							
0x42	USERERRCTRLEVENT1	7:0	USER[7:0]							
0x43	USERCLKCTRLCFD	7:0	USER[7:0]							
0x44	USERCLKCTRLCFM	7:0	USER[7:0]							
0x45	USEREVSYSEVOUTB	7:0	USER[7:0]							
0x46	USEREVSYSEVOUTE	7:0	USER[7:0]							

23.5 Register Description

23.5.1 Software Events A

Name: SWEVENTA
Offset: 0x00
Reset: 0x00
Property: -

Refer to the *Peripheral Overview* section for the available Event System channels.

Bit	7	6	5	4	3	2	1	0
			CH5	CH4	CH3	CH2	CH1	CH0
Access			W	W	W	W	W	W
Reset			0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5 – CHn Channel n Select
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will generate a single-pulse event on event channel n by inverting the signal on the event channel for one peripheral clock (CLK_PER) cycle.

23.5.2 Channel n Generator Selection

Name: CHANNELn
Offset: 0x10 + n*0x01 [n=0..5]
Reset: 0x00
Property: -

Each event channel can be connected to a single event generator.

Refer to the *Peripheral Overview* section for the available Event System channels.

Bit	7	6	5	4	3	2	1	0
	CHANNEL[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 – CHANNEL[7:0] Channel Generator Selection
This bit field controls which event generator is connected to this event channel.
Note: Not all generators can be connected to all channels. Refer to the table below for further details.

Value	Name	Description
0x00	OFF	This Event System Channel is disabled.
0x01	UPDI_SYNCH	The rising edge of SYNCH character detection. An asynchronous event, available on all channels.
0x03	WDT_TICK	WDT LSB Tick. An asynchronous event. Available on all channels.
0x04	SWDT_ERROR	SWDT ERROR. A synchronous event. Available on all channels.
0x05	MVIO_VDDIO2OK	V_{DDIO2} OK. An asynchronous event. Available on all channels.
0x06	RTC_OVF	Counter overflow. An asynchronous event. Available on all channels.
0x07	RTC_CMP	Compare match. An asynchronous event. Available on all channels.
0x08	RTC_EVGEN0	Selectable prescaled RTC event. An asynchronous event. Available on all channels.
0x09	RTC_EVGEN1	Selectable prescaled RTC event. An asynchronous event. Available on all channels.
0x10	CCL_LUT0	LUT output level. An asynchronous event. Available on all channels.
0x11	CCL_LUT1	LUT output level. An asynchronous event. Available on all channels.
0x12	CCL_LUT2	LUT output level. An asynchronous event. Available on all channels.
0x13	CCL_LUT3	LUT output level. An asynchronous event. Available on all channels.
0x14	CCL_LUT4	LUT output level. An asynchronous event. Available on all channels.
0x15	CCL_LUT5	LUT output level. An asynchronous event. Available on all channels.
0x20	AC0_OUT	Comparator output level. An asynchronous event. Available on all channels.
0x21	AC1_OUT	Comparator output level. An asynchronous event. Available on all channels.
0x22	AC2_OUT	Comparator output level. An asynchronous event. Available on all channels.
0x24	ADC0_RESRDY	Result ready. A synchronous event. Available on all channels.
0x25	ADC0_SAMPRDY	Sample ready. A synchronous event. Available on all channels.
0x26	ADC0_WCMP	Window comparison match. A synchronous event. Available on all channels.
0x27	ADC1_RESRDY	Result ready. A synchronous event. Available on all channels.
0x28	ADC1_SAMPRDY	Sample ready. A synchronous event. Available on all channels.
0x29	ADC1_WCMP	Window comparison match. A synchronous event. Available on all channels.
0x30	ZCD0_OUT	Given by ZCD output level. An asynchronous event. Available on all channels.
0x33	ZCD3_OUT	Given by ZCD output level. An asynchronous event. Available on all channels.
0x40	PORTA_EVGEN0	Pin level(2). An asynchronous event. Available on all channels.
0x41	PORTA_EVGEN1	Pin level(2). An asynchronous event. Available on all channels.
0x42	PORTB_EVGEN0(1)	Pin level(2). An asynchronous event. Available on all channels.
0x43	PORTB_EVGEN1(1)	Pin level(2). An asynchronous event. Available on all channels.
0x44	PORTC_EVGEN0	Pin level(2). An asynchronous event. Available on all channels.
0x45	PORTC_EVGEN1	Pin level(2). An asynchronous event. Available on all channels.

Value	Name	Description
0x46	PORTD_EVGEN0	Pin level(2). An asynchronous event. Available on all channels.
0x47	PORTD_EVGEN1	Pin level(2). An asynchronous event. Available on all channels.
0x48	PORTE_EVGEN0	Pin level(2). An asynchronous event. Available on all channels.
0x49	PORTE_EVGEN1	Pin level(2). An asynchronous event. Available on all channels.
0x4A	PORTF_EVGEN0	Pin level(2). An asynchronous event. Available on all channels.
0x4B	PORTF_EVGEN1	Pin level(2). An asynchronous event. Available on all channels.
0x60	USART0_XCK	Clock signal in SPI Host mode and synchronous USART Host mode. A synchronous event, available on all channels.
0x61	USART1_XCK	Clock signal in SPI Host mode and synchronous USART Host mode. A synchronous event. Available on all channels.
0x62	USART2_XCK(1)	Clock signal in SPI Host mode and synchronous USART Host mode. A synchronous event. Available on all channels.
0x68	SPI0_SCK	SPI Host clock signal. A synchronous event. Available on all channels.
0x69	SPI1_SCK	SPI Host clock signal. A synchronous event. Available on all channels.
0x80	TCA0_OVF_LUNF	Overflow/Low-byte timer underflow. A synchronous event. Available on all channels.
0x81	TCA0_HUNF	High byte timer underflow. A synchronous event. Available on all channels.
0x84	TCA0_CMP0_LCMP0	Compare channel 0 match/Low-byte timer compare channel 0 match. A synchronous event. Available on all channels.
0x85	TCA0_CMP1_LCMP1	Compare channel 1 match/Low-byte timer compare channel 1 match. A synchronous event. Available on all channels.
0x86	TCA0_CMP2_LCMP2	Compare channel 2 match/Low-byte timer compare channel 2 match. A synchronous event. Available on all channels.
0xA0	TCB0_CAPT	CAPT Interrupt flag set(3). A synchronous event. Available on all channels.
0xA1	TCB0_OVF	Counter overflow. A synchronous event. Available on all channels.
0xA2	TCB1_CAPT	CAPT Interrupt flag set(3). A synchronous event. Available on all channels.
0xA3	TCB1_OVF	Counter overflow. A synchronous event. Available on all channels.
0xA4	TCB2_CAPT	CAPT Interrupt flag set(3). A synchronous event. Available on all channels.
0xA5	TCB2_OVF	Counter overflow. A synchronous event. Available on all channels.
0xA6	TCB3_CAPT	CAPT Interrupt flag set(3). A synchronous event. Available on all channels.
0xA7	TCB3_OVF	Counter overflow. A synchronous event. Available on all channels.
0xB0	TCD0_CMPBCLR	Counter matches CMPBCLR. A synchronous event. Available on all channels.
0xB1	TCD0_CMPASET	Counter matches CMPASET. A synchronous event. Available on all channels.
0xB2	TCD0_CMPBSET	Counter matches CMPBSET. A synchronous event. Available on all channels.
0xB3	TCD0_PROGEV	Programmable event output. An asynchronous event. Available on all channels.

Notes:

1. Not all peripheral instances are available for all pin counts. Refer to the *Peripherals and Architecture* section for details.
2. An event from the 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. Refer to the TCB section for details.

23.5.3 User x Channel Selection

Name: USERx
Offset: 0x20 + x*0x01 [x=0..38]
Reset: 0x00
Property: -

Each event user can be connected to a single channel, and multiple users can be connected to the same channel.

Refer to the *Register Summary* section for all available User x Channel Selection (USERx) registers.

Refer to the *Event Users* section for further details.

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
This bit field controls which event channel this event user is connected to.

Value	Name	Description
0x0	OFF	This Event System user is not connected to any channel
0x1	CHANNEL0	This Event System user is connected to Event Channel 0
0x2	CHANNEL1	This Event System user is connected to Event Channel 1
0x3	CHANNEL2	This Event System user is connected to Event Channel 2
0x4	CHANNEL3	This Event System user is connected to Event Channel 3
0x5	CHANNEL4	This Event System user is connected to Event Channel 4
0x6	CHANNEL5	This Event System user is connected to Event Channel 5
Other	-	Reserved

24. PORTMUX - Port Multiplexer

24.1 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.

24.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	EVSYSROUTEA	7:0			EVOUTF	EVOUTE	EVOUTD	EVOUTC	EVOUTB	EVOUTA
0x01	CCLROUTEA	7:0			LUT5	LUT4	LUT3	LUT2	LUT1	LUT0
0x02	USARTROUTEA	7:0				USART1[1:0]		USART0[2:0]		
0x03	USARTROUTEB	7:0							USART2[1:0]	
0x04	Reserved									
0x05	SPIROUTEA	7:0			SPI1[2:0]			SPI0[2:0]		
0x06	TWIROUTEA	7:0					TWI1[1:0]		TWI0[1:0]	
0x07	TCAROUTEA	7:0						TCA0[2:0]		
0x08	TCBROUTEA	7:0					TCB3	TCB2	TCB1	TCB0
0x09	TCDROUTEA	7:0						TCD0[2:0]		
0x0A	ACROUTEA	7:0						AC2	AC1	AC0
0x0B	ZCDROUTEA	7:0					ZCD3			ZCD 0
0x0C	Reserved									
0x0D	ERRCTRLROUTEA	7:0						HEART[2:0]		

24.3 Register Description

24.3.1 EVSYS Pin Position

Name: EVSYSROUTEA
Offset: 0x00
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
			EVOUTF	EVOUTE	EVOUTD	EVOUTC	EVOUTB	EVOUTA
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – EVOUTF Event Output F
This bit controls the pin position for event output F.

Value	Name	Description
0	DEFAULT	EVOUT on PF2
1	ALT1	-

Bit 4 – EVOUTE Event Output E
This bit controls the pin position for event output E.

Value	Name	Description
0	DEFAULT	EVOUT on PE2
1	ALT1	-

Bit 3 – EVOUTD Event Output D
This bit controls the 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 the 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 the pin position for event output B.

Value	Name	Description
0	DEFAULT	EVOUT on PB2
1	ALT1	-

Bit 0 – EVOUTA Event Output A
This bit controls the pin position for event output A.

Value	Name	Description
0	DEFAULT	EVOUT on PA2
1	ALT1	EVOUT on PA7

24.3.2 CCL LUTn Pin Position

Name: CCLROUTEA
Offset: 0x01
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

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
CCL LUT 5 is not connected to external pins.

Value	Name	Description			
		OUT	IN0	IN1	IN2
0	DEFAULT	-	–	-	-
1	-	Reserved			

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	-	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	ALT1	-	PF0	PF1	PF2

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

24.3.3 USARTn Pin Position

Name: USARTROUTEA
Offset: 0x02
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
				USART1[1:0]		USART0[2:0]		
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 4:3 – 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	ALT2	PD6	PD7	-	-
0x3	NONE	Not connected to any pins			

Bits 2:0 – USART0[2:0] 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	ALT2	PA2	PA3	-	-
0x3	ALT3	PD4	PD5	PD6	PD7
0x4	ALT4	PC1	PC2	PC3	-
0x5	NONE	Not connected to any pins			

24.3.4 USARTn Pin Position

Name: USARTROUTEB
Offset: 0x03
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
							USART2[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – 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	-	-
0x2	-	Reserved			
0x3	NONE	Not connected to any pins			

24.3.5 SPIn Pin Position

Name: SPIROUTEA
Offset: 0x05
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
			SPI1[2:0]			SPI0[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 – SPI1[2:0] 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(2)	Not connected(1)	Not connected(1)	Not connected(1)
0x3 – 0x6	-	Reserved			
0x7	NONE	Not connected to any pins			Not connected(1)

Notes:

1. Not connected to any pin, but the SPI logic is internally set to 1.
2. ALT2 is not supporting proper SPI operation, but the MOSI pin can still serve, e.g., as a digital waveform output.

Bits 2:0 – SPI0[2:0] 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	-	Reserved			
0x3	ALT3	PA0	PA1	PC0	PC1
0x4	ALT4	PD4	PD5	PD6	PD7
0x5	ALT5	PC0	PC1	PC2	PC3
0x6	ALT6	PC1	PC2	PC3	Not connected(1)
0x7	NONE	Not connected to any pins			Not connected(1)

Note:

1. Not connected to any pin, but the SPI logic is set to 1 internally.

24.3.6 TWIn Pin Position

Name: TWIROUTEA
Offset: 0x06
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

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			
		Host/Client		Dual Mode (Client)	
		SDA	SCL	SDA	SCL
0x0	DEFAULT	PF2	PF3	PB2(1)	PB3(1)
0x1	ALT1(1)	PF2	PF3	-	-
0x2	ALT2(1)	PB2	PB3	-	-
0x3	-	Reserved			

Note:

1. Only available on devices with 48 pins.

Bits 1:0 – TWI0[1:0] TWI 0 Signals
This bit field controls the pin positions for TWI 0 signals.

Value	Name	Description			
		Host/Client		Dual Mode (Client)	
		SDA	SCL	SDA	SCL
0x0	DEFAULT	PA2	PA3	PC2	PC3
0x1	ALT1	PA2	PA3	PC6	PC7
0x2	ALT2	PC2	PC3	PC6	PC7
0x3	ALT3	PA0	PA1	PC2	PC3

24.3.7 TCAn Pin Position

Name: TCAROUTEA
Offset: 0x07
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
						TCA0[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bits 2:0 – TCA0[2:0] TCA0 Signals
This bit field controls the pin positions for TCA0 signals.

Value		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	-	-
0x5	PORTF	PF0	PF1	PF2	PF3	PF4	PF5
Others		Reserved					

24.3.8 TCBn Pin Position

Name: TCBROUTEA
Offset: 0x08
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
					TCB3	TCB2	TCB1	TCB0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – TCB3 TCB3 Output
This bit controls the 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 the 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 the 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 the pin position for TCB0 output.

Value	Name	Description
0	DEFAULT	WO on PA2
1	ALT1	WO on PF4

24.3.9 TCDn Pin Position

Name: TCDROUTEA
Offset: 0x09
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

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	-	-
0x2	ALT2	PF0	PF1	PF2	PF3
0x3	-	Reserved			
0x4	ALT4	PA4	PA5	PD4	PD5
Other	-	Reserved			

24.3.10 ACn Pin Position

Name: ACROUTEA
Offset: 0x0A
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

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 AC 2 output
This bit field controls the pin positions for AC 2 output.

Value	Name	Description
0	DEFAULT	PA7
1	ALT1	PC6

Bit 1 – AC1 AC 1 output
This bit field controls the pin positions for AC 1 output.

Value	Name	Description
0	DEFAULT	PA7
1	ALT1	PC6

Bit 0 – AC0 AC 0 output
This bit field controls the pin positions for AC 0 output.

Value	Name	Description
0	DEFAULT	PA7
1	ALT1	PC6

24.3.11 ZCDn Pin Position

Name: ZCDROUTEA
Offset: 0x0B
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

Bit	7	6	5	4	3	2	1	0
					ZCD3			ZCD 0
Access					R/W			R/W
Reset					0			0

Bit 3 – ZCD3 ZCD 3 output
This bit field controls the pin positions for ZCD 3 output.

Value	Name	Description
0	DEFAULT	PA7
1	ALT1	PC7

Bit 0 – ZCD 0 ZCD 0 output
This bit field controls the pin positions for ZCD 0 output.

Value	Name	Description
0	DEFAULT	PA7
1	ALT1	PC7

24.3.12 Error Controller Route A

Name: ERRCTRLROUTEA
Offset: 0x0D
Reset: 0x00
Property: -

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

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

Bits 2:0 – HEART[2:0] Error Controller Heartbeat
This bit field controls the pin positions for the Error Controller Heartbeat.

Value	Name	Description
0x00	DEFAULT	PA2
0x01	ALT1	PA4
0x02	ALT2	PA6
0x03	ALT3	PA7
0x04	ALT4	PD4
0x05	ALT5	PF5

25. PORT - I/O Pin Configuration

25.1 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)

25.2 Overview

The device's I/O pins 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 define the output state. For example, DIR[3] and OUT[3] of the PORTA instance controls pin PA3.

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 pin value can be read whether the pin is configured as input or output.

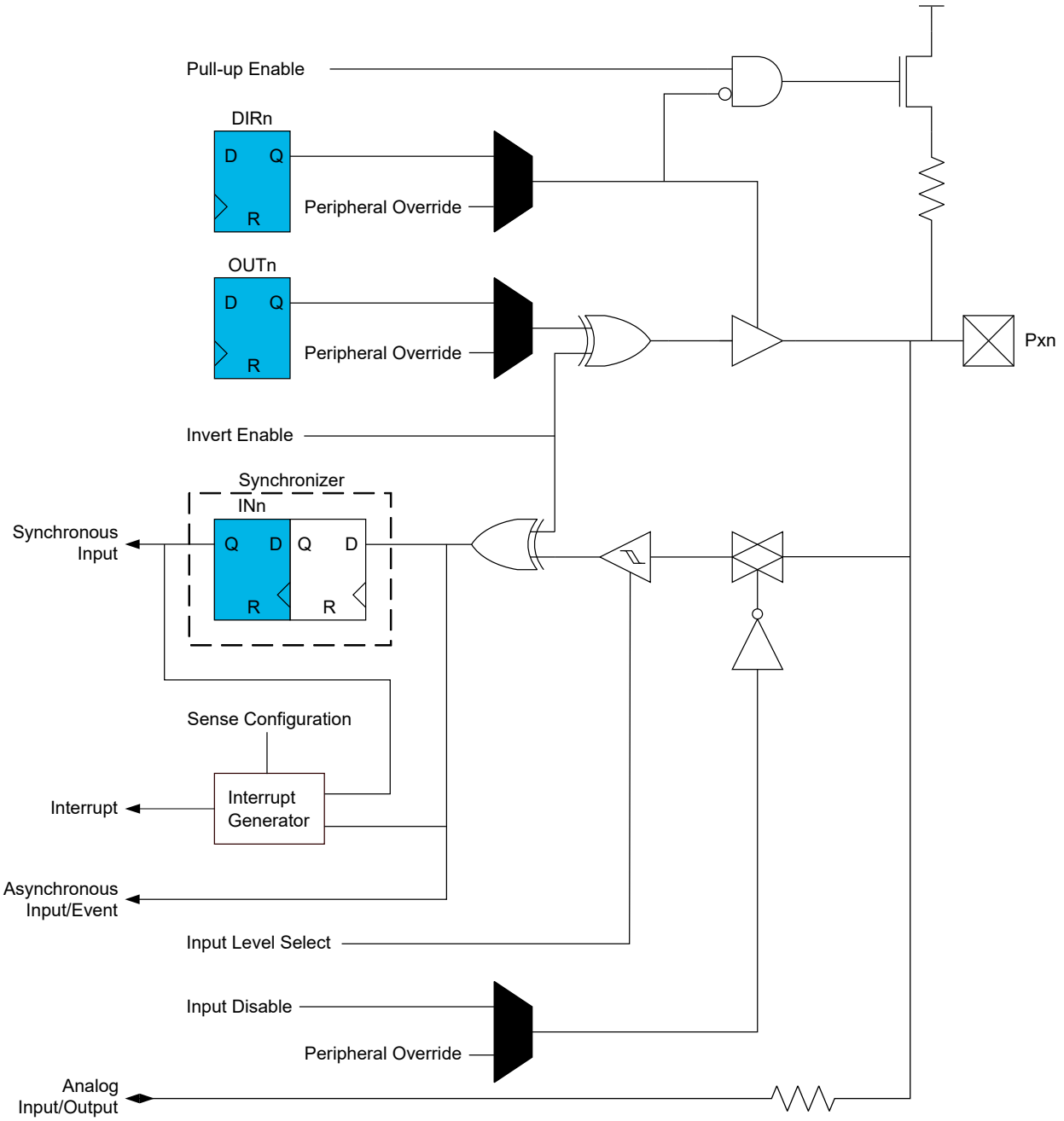
The PORT also supports asynchronous input sensing with interrupts and events for selectable pin change conditions. Asynchronous pin change sensing means that a pin change can trigger an interrupt and wake the device from sleep, including sleep modes where CLK_PER is stopped.

All pin functions are individually configurable per pin. The pins have hardware RMW functionality for a safe and correct change of the drive values and/or input and sense configuration.

The PORT pin configuration controls the input and output selection of other device functions.

25.2.1 Block Diagram

Figure 25-1. PORT Block Diagram



25.2.2 Signal Description

Signal	Type	Description
Pxn	I/O pin	I/O pin n on PORTx

25.3 Functional Description

25.3.1 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 the lowest possible power consumption, disable the digital input buffer of unused pins and pins used as analog inputs or outputs. For pins with the digital input buffer enabled it is recommended to transition between the high and low voltage thresholds as quickly as possible.

Specific pins, such as those used to connect a debugger, may be configured differently, as required by their special function.

25.3.2 Operation

25.3.2.1 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.

25.3.2.2 Port Configuration

The Port Control (PORTx.PORTCTRL) register controls 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.

25.3.2.3 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 compatible thresholds derived from the Low-Voltage Buffer (LVBUF) is 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 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.

25.3.2.4 Multi-Pin Configuration

The multi-pin configuration function can 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, allowing changing the configuration (PORTx.PINnCTRL) for up to eight pins in one write.

Tip: The PORTx.PINCONFIG register is mirrored on all ports, allowing the use of a single setting across multiple ports. The PORTx.PINCTRLUPD/SET/CLR registers are not mirrored and configurations must be written for each port.

For the multi-pin configuration, port pins can be configured and modified by writing to the following registers.

Table 25-1. Multi-Pin Configuration Registers

Register	Description
PORTx.PINCONFIG	PINnCTRL (ISC, PULLUPEN, INLVL and INVEN) setting to prepare simultaneous configuration of 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 PINCTRLCLR 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 nonzero ISC bit fields will result in a bitwise OR with the PINCONFIG and PINnCTRL registers and may give an unexpected setting.
2. Using PINCTRLCLR to configure nonzero ISC bit fields will result in a bitwise inverse AND with the PINCONFIG and PINnCTRL registers and may give an unexpected setting.

The following code snippet demonstrates how to configure multiple PINnCTRL registers of several ports. Note that, because the PINCONFIG register is mirrored across all the ports, it is enough to only write it once, for PORT A, in this example.

```
PORTA.PINCONFIG = PORT_ISC_INPUT_DISABLE_gc; /* The setting to load to the PINnCTRL registers
*/
PORTA.PINCTRLUPD = 0xff;
PORTB.PINCTRLUPD = 0xff;
PORTC.PINCTRLUPD = 0xff;
PORTD.PINCTRLUPD = 0xff;
PORTE.PINCTRLUPD = 0xff;
```

25.3.2.5 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 allowing 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 25-2. Virtual Port Mapping

Regular PORT Register	Mapped to Virtual PORT Register
PORTx.DIR	VPORTx.DIR
PORTx.OUT	VPORTx.OUT
PORTx.IN	VPORTx.IN
PORTx.INTFLAGS	VPORTx.INTFLAGS

Note: Avoid accessing the mapped VPORT register using the single-cycle I/O instructions immediately after accessing the regular PORT register. This may cause a memory collision since the single-cycle I/O access to VPORT is faster than the regular PORT register access.

25.3.2.6 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.

25.3.2.7 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 *MVIO - Multi-Voltage I/O* section for further information.

25.3.3 Interrupts

Table 25-3. 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, consider these points:

- 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 disabling an input by writing to ISC while synchronizing an interrupt, that specific 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

25.3.3.1 Asynchronous Sensing Pin Properties

All PORT pins support fully 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. The pulse width needed to trigger an interrupt is less than one CLK_PER cycle.

25.3.4 Events

PORT can generate the following events:

Table 25-4. Event Generators in PORTx

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
PORTx	EVGEN0	Pin input level	Level	Asynchronous	Given by pin level
PORTx	EVGEN1	Pin input level	Level	Asynchronous	Given by pin level

All PORT pins can be configured as asynchronous Event System generators. Two event generators are available for each port. The output from PORT to the Event System is the value present on the corresponding pin if the digital input buffer is enabled. If a pin input buffer is disabled, the corresponding output to the Event System is zero.

PORT has no event inputs. Refer to the *Event System (EVSYS)* section for more details regarding event types and Event System configuration.

25.3.5 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.

25.3.6 Debug Operation

The PORT continues ordinary operation when halting the CPU in Debug mode. If configuring the PORT in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may occur during debugging.

25.4 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]		
0x18	EVGENCTRLA	7:0		EVGEN1SEL[2:0]				EVGEN0SEL[2:0]		

25.5 Register Description - PORTx

25.5.1 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 table below shows the available configuration for each bit n in this bit field.

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

25.5.2 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.

25.5.3 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.

25.5.4 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.

25.5.5 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 affects the output when the output driver (PORTx.DIR) is enabled for the corresponding pin.
The table below shows the available configuration for each bit n in this bit field.

Value	Description
0	The pin n (Pxn) output is driven low
1	The Pxn output is driven high

25.5.6 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.

25.5.7 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.

25.5.8 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.

25.5.9 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 table below shows the available states of each bit n in this bit field.

Value	Description
0	The voltage level on Pxn is low
1	The voltage level on Pxn is high

25.5.10 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.

25.5.11 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

25.5.12 Multi-Pin Configuration

Name: PINCONFIG
Offset: 0x0B
Reset: 0x00
Property: -

For faster configuration of the port module, the multi-pin configuration write enables the configuration of several port pins in a single cycle. 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.

Value	Name	Description
0	ST	Schmitt Trigger derived from supply level
1	TTL	TTL compatible levels from Low-Voltage Buffer (LVBUF)

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 the pin conditions that will trigger a port interrupt.

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(2)
other	—	Reserved

Notes:

1. If the digital input buffer for pin n is disabled, bit n in the Input Value (PORTx.IN) register will not be updated.
2. The LEVEL interrupt will keep triggering continuously as long as the pin stays low.

25.5.13 Multi-Pin Control Update Mask

Name: PINCTRLUPD
Offset: 0x0C
Reset: 0x00
Property: -

For faster configuration of the port module, the multi-pin configuration write enables the configuration of several port pins in a single cycle. 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.

25.5.14 Multi-Pin Control Set Mask

Name: PINCTRLSET
Offset: 0x0D
Reset: 0x00
Property: -

For faster configuration of the port module, the multi-pin configuration write enables the configuration of several port pins in a single cycle. 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.

25.5.15 Multi-Pin Control Clear Mask

Name: PINCTRLCLR
Offset: 0x0E
Reset: 0x00
Property: -

For faster configuration of the port module, the multi-pin configuration write enables the configuration of several port pins in a single cycle. 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.

25.5.16 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.

Value	Name	Description
0	ST	Schmitt Trigger derived from supply level
1	TTL	TTL compatible levels from Low-Voltage Buffer (LVBUF)

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 the pin conditions that will trigger a port interrupt.

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(2)
other	—	Reserved

Notes:

1. If the digital input buffer for pin n is disabled, bit n in the Input Value (PORTx.IN) register will not be updated.
2. The LEVEL interrupt will keep triggering continuously as long as the pin stays low.

25.5.17 Event Generator Control A

Name: EVGENCTRLA
Offset: 0x18
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		EVGEN1SEL[2:0]				EVGEN0SEL[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bits 0:2, 4:6 – EVGENnSEL Event Generator n Select
This bit field controls which pin is connected to Event Generator n.

Value	Name	Description
0x0	PIN0	Pin 0 used as event generator
0x1	PIN1	Pin 1 used as event generator
0x2	PIN2	Pin 2 used as event generator
0x3	PIN3	Pin 3 used as event generator
0x4	PIN4	Pin 4 used as event generator
0x5	PIN5	Pin 5 used as event generator
0x6	PIN6	Pin 6 used as event generator
0x7	PIN7	Pin 7 used as event generator

25.6 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]							

25.7 Register Description - VPORTx

25.7.1 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 allowing 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 table below shows the available configuration for each bit n in this bit field.

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

25.7.2 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 allowing 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 affects the output when the output driver (PORTx.DIR) is enabled for the corresponding pin.
The table below shows the available configuration for each bit n in this bit field.

Value	Description
0	The pin n (Pxn) output is driven low
1	The Pxn output is driven high

25.7.3 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 allowing 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 table below shows the available states of each bit n in this bit field.

Value	Description
0	The voltage level on Pxn is low
1	The voltage level on Pxn is high

25.7.4 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 allowing 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.

26. MVIO - Multi-Voltage I/O

26.1 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

26.2 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, eliminating the need for having external level shifters for communication or control of external components running on a different voltage level. A voltage applied to the VDDIO2 power pin(s) supplies the MVIO-capable I/O pins, while the voltage applied to the VDD pin(s) supplies the regular I/O pins.

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)

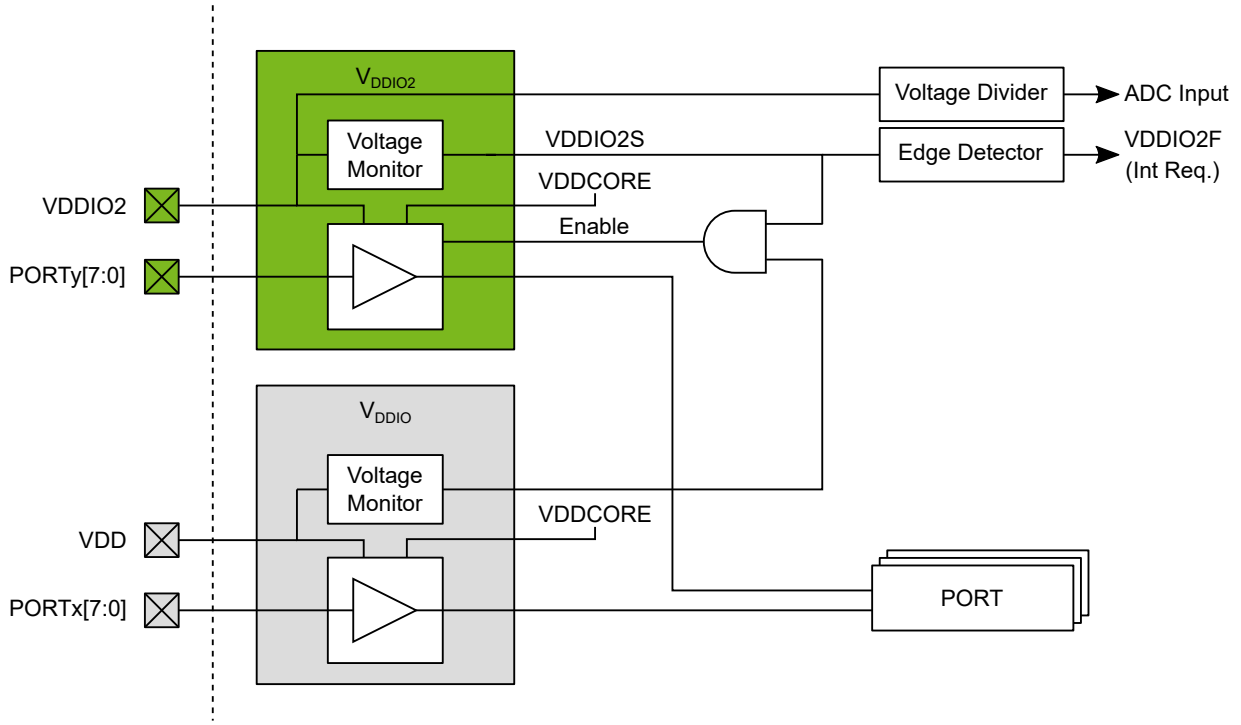
A configuration fuse determines the MVIO supply mode. The loss or gain of power on V_{DDIO2} is signaled by a status register bit. This status bit has corresponding interrupt and event functionality.

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.

26.2.1 Block Diagram

Figure 26-1. MVIO Block Diagram



26.2.2 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

26.3 Functional Description

26.3.1 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'.

26.3.2 Operation

26.3.2.1 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.

Note:
When a peripheral is connected to MVIO pins, it will continue operation when the pins are tri-stated. The VDDIO2S flag must be monitored to ensure the correct operation of the peripheral and I/O pins.

26.3.2.2 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:

$$V_{DDIO2} = \frac{\text{ADC Result} \times V_{REF} \times 10}{\text{ADC Resolution}}$$

26.3.3 Events

The MVIO can generate the following events:

Table 26-1. 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.

26.3.4 Interrupts

Table 26-2. 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 trigger an interrupt. This interrupt 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.

26.3.5 Sleep Mode Operation

When enabled by the Multi-Voltage System Configuration (MVSYSCFG) fuse, the module will operate in all sleep modes.

26.3.6 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.

26.4 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

26.5 Register Description

26.5.1 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

26.5.2 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.

26.5.3 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.

27. BOD - Brown-out Detector

27.1 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

27.2 Overview

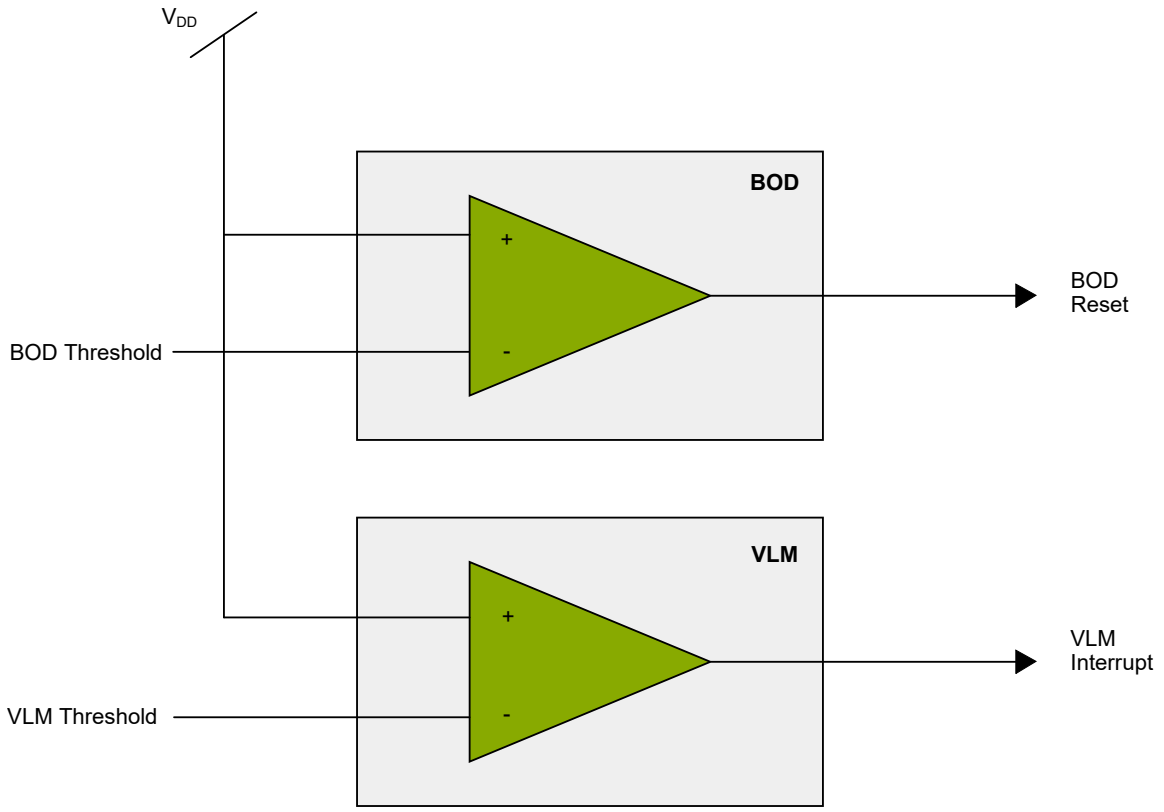
The Brown-out Detector (BOD) monitors the power supply and compares the supply voltage with the programmable brown-out threshold level. The brown-out threshold level defines when to generate a System Reset. The Voltage Level Monitor (VLM) monitors the power supply and compares it to a threshold higher 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.

27.2.1 Block Diagram

Figure 27-1. BOD Block Diagram



27.3 Functional Description

27.3.1 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 the 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.VLMCTRLA) register.

27.3.2 Interrupts

Table 27-1. Available Interrupt Vectors and Sources

Vector Name	Source Name	Condition	Dependency
BOD_VLM	VLM	The supply voltage crosses the voltage level threshold	The Voltage Level Monitor Configuration (VLMCFG) bit field in the Interrupt Control (BOD.INTFLAGS) 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.

27.3.3 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.

27.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a given 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 it unchanged.

The following registers are under CCP:

Table 27-2. Registers Under Configuration Change Protection

Register	Key
The SLEEP and SAMPFREQ bits in the BOD.CTRLA register	IOREG

27.4 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 ... 0x07	Reserved									
0x08	VLMCTRLA	7:0							VLMLVL[1:0]	
0x09	INTCTRL	7:0						VLMCFG[1:0]		VLMIE
0x0A	INTFLAGS	7:0								VLMIF
0x0B	STATUS	7:0								VLMS

27.5 Register Description

27.5.1 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.

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] Active
When the device is in Active or Idle sleep mode, these bits select the BOD operation mode.
The Reset value is loaded from the ACTIVE bits in FUSE.BODCFG.

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] Sleep
When the device is in Standby or Power-Down sleep mode, these bits select the BOD operation mode. The Reset value is loaded from the SLEEP bits in FUSE.BODCFG.

Value	Name	Description
0x0	DISABLE	BOD disabled
0x1	ENABLE	BOD enabled in Continuous mode
0x2	SAMPLE	BOD enabled in Sampled mode
0x3	-	Reserved

27.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0xxx
Property: -

The Reset value is loaded from the BOD Level (LVL) bits in the BOD Configuration Fuse (FUSE.BODCFG).

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

Bits 2:0 – LVL[2:0] BOD Level
This bit field controls the BOD threshold level.

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

Note: BODLEVEL0 will only be enabled during chip erase. During normal operation, writing '0x0' to this bit field will be the same as disabling the BOD.

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

27.5.3 VLM Control

Name: VLMCTRLA
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

27.5.4 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.

27.5.5 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 cleared by writing a '1' to it.
This flag is set when a trigger from the VLM is given. The trigger condition is configured by writing to the VLM Configuration (VLMCFG) bit field in the Interrupt Control (INTCTRL) register. The flag is only update when the BOD is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the VLM Interrupt flag.

27.5.6 VLM Status

Name: STATUS
Offset: 0x0B
Reset: 0x00
Property: -

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

28. VREF - Voltage Reference

28.1 Features

- Programmable Voltage Reference Sources:
  - One reference for Digital-to-Analog Converter 0 (DAC0)
  - One reference for Analog Comparator 0 (AC0)
- Each Reference Source Supports the Following Voltages:
  - 1.024V
  - 2.048V
  - 4.096V
  - 2.500V
  - VDD
  - External reference on VREFx pin

28.2 Overview

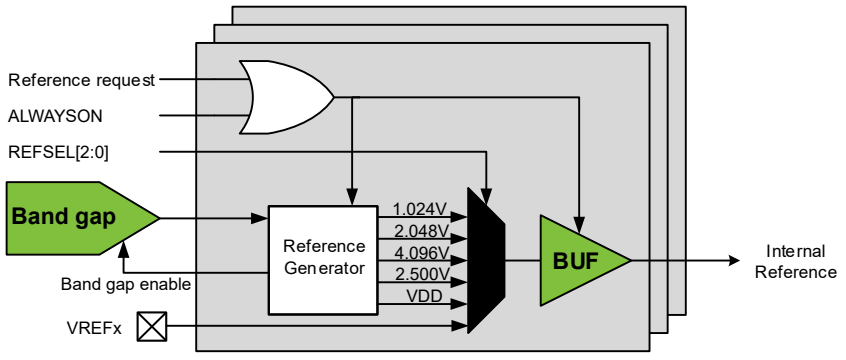
The Voltage Reference (VREF) peripheral provides control registers for the voltage reference sources used by several peripherals. The user can select the reference voltages for the DAC0 and AC by writing to the appropriate registers in the VREF peripheral.

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.DAC0REF and VREF.ACREF. This will decrease the start-up time at the cost of increased power consumption.

Note: The voltage references for ADC0 and ADC1 are configured by writing to the respective Control C register in the ADCn peripheral (ADCn.CTRLC).

28.2.1 Block Diagram

Figure 28-1. VREF Block Diagram



28.2.2 Peripherals Using Voltage References

The devices of the AVR SD family have a number of peripherals that can use a voltage reference.

- The **DAC** peripheral is using the voltage reference controlled by the VREF.DAC0REF register.
- The **AC** peripheral is using input pins, but the negative input can be routed to use the voltage reference controlled by VREF.ACREF. The instances of the AC peripheral share the same voltage reference from the VREF peripheral. In addition, each AC instance has its own, independent voltage divider for the incoming reference voltage, controlled by the respective ACn.DACREF register.

- The **ADC** peripherals are providing their own control register for their internal voltage reference, the Reference Selection (REFSEL) bit field in their respective ADCn.CTRLC registers. Note that ADC0 and ADC1 are accessing two different, independent voltage references. This redundancy can be used to improve functional safety:
  - ADC0 is using a voltage reference residing in the VREF peripheral.
  - ADC1 is using a voltage reference internal to the Voltage Regulator Monitor (VMON). VMON is a part of the Sleep Controller (SLPCTRL) peripheral. The VMON must be enabled when operating ADC1.

28.3 Functional Description

28.3.1 Initialization

DAC, AC The default configuration will enable the respective source when the DAC0 or ACn is 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 DAC0 Reference (DAC0REF) or Analog Comparator (ACREF) register.

ADC The default reference voltage for ADC0 and ADC1 is V_{DD}, and can be further configured by writing to the respective REFSEL bit field in the ADCn.CTRLC register.

Important: Altering the voltage reference for one peripheral may cause undesirable noise in other peripherals dependent on the reference. Disable those peripherals during the reference change to prevent this.

28.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00 ... 0x01	Reserved									
0x02	DAC0REF	7:0	ALWAYSON					REFSEL[2:0]		
0x03	Reserved									
0x04	ACREF	7:0	ALWAYSON					REFSEL[2:0]		

28.5 Register Description

28.5.1 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.
Note:

1. The values given for internal references are only typical. Refer to the *Electrical Characteristics* section for further details.

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

28.5.2 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 AC 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 the analog comparators ACn.

Table 28-1. Voltage Reference

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.

29. WDT - Watchdog Timer

29.1 Features

- Issues a System Reset If Its Counter Is Not Reset Before It Expires
- Asynchronous Operation Relative to the System Clock Using an Independent Oscillator
- Clocked by the 32.768 kHz Ultra-Low Power Internal Oscillator (OSC32K)
- 11 Selectable Time-out Intervals
  - From 244 µs to 250 ms
- Two Operation Modes
  - Normal mode
  - Window mode
- Configuration Lock to Prevent Unwanted Changes
- Closed Interval Timer Activation After First WDT Instruction for Easy Setup
- Diagnostic Features for Functional Safety
  - WDT clock monitor using independent clock
  - WDT counter LSB routed as an event system source
  - WDT counter is readable

29.2 Overview

The Watchdog Timer (WDT) is a system function for monitoring correct program operation. When enabled, the WDT is a constantly running counter with a configurable time-out interval. If the WDT is not reset within the time-out interval, it will issue a system reset allowing the system to recover from situations such as runaway or deadlocked code. Executing the `WDR` (Watchdog Reset) instruction from the software resets the WDT.

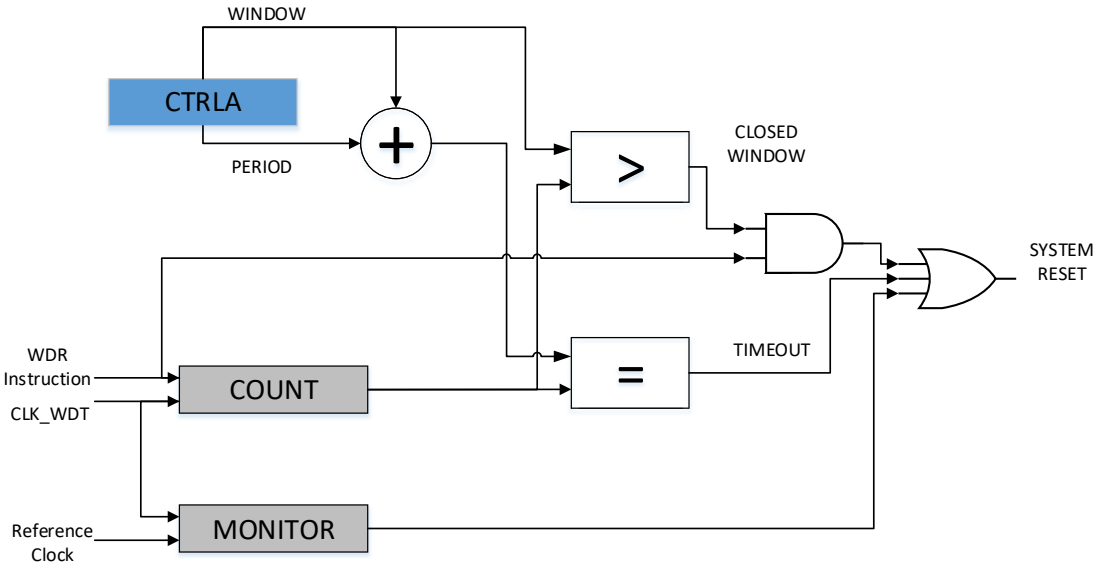
In addition to the Normal mode, the WDT has a Window mode as described above. The Window mode defines a time slot or “window” inside the time-out interval during which the WDT must be reset. If the WDT is reset outside this window, either too early or too late, a system reset will be issued. Compared to the Normal mode, the Window mode can catch situations where a code error causes frequent repeated execution of the `WDR` instruction.

When enabled, the WDT will run in Active mode and all sleep modes. Since it is asynchronous (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 accidentally be changed.

29.2.1 Block Diagram

Figure 29-1. WDT Block Diagram



29.3 Functional Description

29.3.1 Initialization

1. The WDT is enabled when a non-zero value is written to the Period (PERIOD) bit field in the Control A (CTRLA) register.
2. Optional: Write a non-zero value to the Window (WINDOW) bit field in the CTRLA register to enable the Window mode operation.

All bits in the Control A register and the Lock (LOCK) bit in the Status (STATUS) register are write-protected by the Configuration Change Protection (CCP) mechanism.

A fuse (FUSE.WDTCFG) defines the Reset value of the 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 STATUS register is set at boot time.

29.3.2 Clocks

The WDT's clock is sourced from the internal Ultra-Low Power Oscillator. This oscillator is optimized for power consumption, not frequency accuracy. The exact time-out interval will vary from device to device. This variation must be considered when designing software that uses the WDT to ensure that the time-out intervals 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 Control A (CTRLA) register will require synchronization between the clock domains.

The WDT has a clock monitor to detect failures in CLK_WDT. For more information, see the WDT *Clock Monitor* section.

29.3.3 Operation

29.3.3.1 Normal Mode

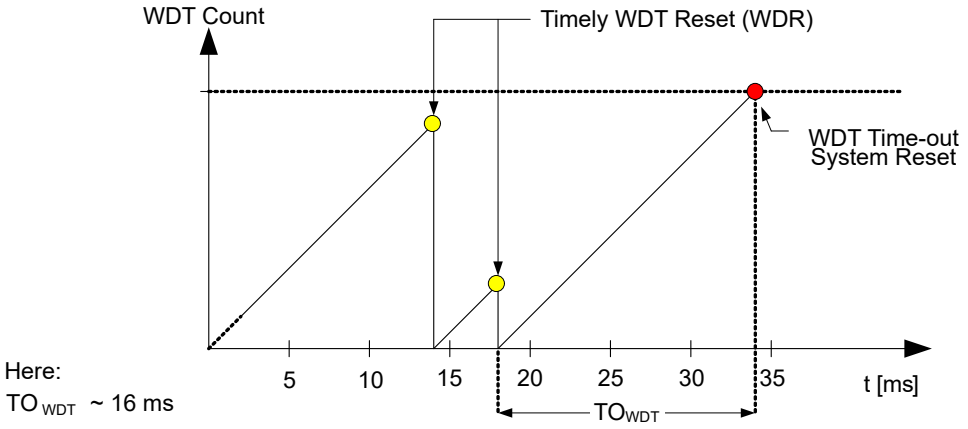
A single time-out interval is configured for the WDT in the Normal mode operation. If the WDT is not reset from the software using the `WDR` instruction during the configured time-out interval, the WDT will issue a system reset.

Each time the WDT is reset by software using the `WDR` instruction, a new WDT time-out interval starts.

There are 11 possible WDT time-out intervals (TO_{WDT}), selectable from 244 µs to 250 ms by writing to the Period (PERIOD) bit field in the Control A (CTRLA) register.

The figure below shows a typical timing scheme for the WDT operating in Normal mode.

Figure 29-2. Normal Mode Operation



The Normal mode is enabled as long as the Window (WINDOW) bit field in the CTRLA register is '`0x0`'.

29.3.3.2 Window Mode

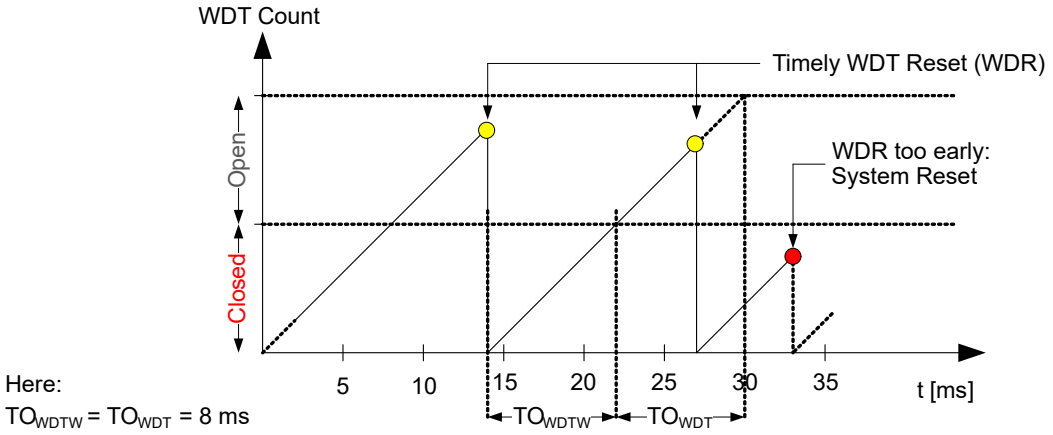
In Window mode operation, the WDT uses two different time-out intervals, a closed window time-out interval (TO_{WDTW}) and an open window time-out interval (TO_{WDT}):

- TO_{WDTW} defines an interval ranging from 244 µs to 250 ms where WDT reset is not allowed. If the `WDR` instruction is executed during this interval, the WDT will issue a system reset.
- TO_{WDT}, which is also configurable from 244 µs to 250 ms, defines the duration of the open interval during which the WDT can (and needs to) be reset. The open interval will always follow the closed interval, so the total duration of the time-out interval is the sum of the closed window and the open window time-out intervals.

When enabling the Window mode or leaving 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 29-3. Window Mode Operation



The Window mode is enabled by writing a non-zero value to the WINDOW bit field in the Control A (CTRLA) register and disabled by writing it to `0x0`.

29.3.3.3 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 *Configuration Change Protection* section for further details.
- Locking the configuration by writing a '1' to the Lock (LOCK) bit in the Status (STATUS) register. When this bit is '1', the Control A (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 write it to '0'. Consequently, the software cannot disable the WDT.

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 the STATUS register.

29.3.3.4 Timing of WDR Instruction

For either Normal or Window modes, the reception of a WDR instruction while a previous WDR instruction is being synchronized into the WDT domain is considered an error and causes the WDT to request a reset of the MCU as if a watchdog timeout had happened.

29.3.3.5 WDT Clock Monitor

A clock from an oscillator independent of CLK_WDT is used to detect a failure of CLK_WDT. The WDT is an important safety mechanism in a Functional Safety system, and a failure of CLK_WDT would render the WDT unable to fulfill its function. If the CLK_WDT monitor detects that CLK_WDT has failed, the monitor will immediately request entry to a safe state, implemented by a Machine Check Reset request.

The WDTMON bit field in the CTRLB register controls the behavior of the WDT clock monitor feature in sleep. This read-only bit field is initialized from a fuse value during system initialization.

29.3.4 Sleep Mode Operation

The WDT will continue to operate in any sleep mode where the source clock is active. The behavior of the Clock Monitor is determined by the WDTMON bit field in the CTRLB register.

29.3.5 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 resets.

When starting the CPU and when the WDT is operating in Window mode, the first closed window time-out interval will be disabled, and a Normal mode time-out interval is executed.

29.3.6 Synchronization

The Control A (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 (STATUS) register indicates if there is an ongoing synchronization.

Writing to the CTRLA register while SYNCBUSY = `1` is not allowed.

The following bit fields must be synchronized when written:

- The Period (PERIOD) bit field in Control A (CTRLA) register
- The Window (WINDOW) bit field in Control A (CTRLA) register

Synchronizing the `WDR` instruction requires two to three WDT clock cycles.

The CTRLB register is in the peripheral clock domain and does not need synchronization when read or written.

The CNT register is synchronized to the peripheral clock domain and can be read whenever the CNTBUSY bit is not set. This 14-bit register is not buffered by a TEMP register, so reads of the two register bytes may be inconsistent. The intended use of this register is reading it at periodic intervals, e.g., every millisecond or on every iteration of the main()-loop, and verifying that the CNTL register has approximately the expected value. Alternatively, verify the entire register during the system startup test by polling CNTBUSY until it transitions from 1 to 0 and then immediately reading CNTH and CNTL, all in an atomic operation.

29.3.7 Accesses to Reserved Addresses

Accesses to RESERVED addresses will be aborted and return a Bus Error response.

The CTRLB and CNT registers are read-only. An attempted write will be discarded and return a bus error response.

29.3.8 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a given 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 it unchanged.

The following registers are under CCP:

Table 29-1. WDT - Registers Under Configuration Change Protection

Register	Key
WDT.CTRLA	IOREG
LOCK bit in WDT.STATUS	IOREG

29.3.9 Events

An asynchronous event is generated from the LSB of the WDT counter. The event will be identical to the waveform of the LSB of the WDT counter. This feature is useful to verify that the WDT is running. This event is active whenever the WDT clock is active, i.e., also in STANDBY and POWERDOWN sleep modes.

Note: This event will not have a constant period as the LSB of the counter may not change on every WDT clock tick in the case of the window going from OPEN to CLOSED in Window mode, or in the case of a WDR instruction. In these cases, up to three CLK_WDT cycles may pass without the LSB toggling. Any monitoring system must be tolerant of this irregularity.

29.4 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	CTRLB	7:0	WDTMON[1:0]							
0x02	STATUS	7:0	LOCK					CAUSE	CNTBUSY	SYNCBUSY
0x03	Reserved									
0x04	CNT	7:0	CNT[7:0]							
		15:8			CNT[13:8]					

29.5 Register Description

29.5.1 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 this bit field enables the Window mode and selects the duration of the closed window.
The bits are optionally lock-protected:

- If the LOCK bit in the STATUS register is '1', all bits are change-protected (Access = R)
- If the LOCK bit in the STATUS register is '0', all bits can be changed (Access = R/W)

Value	Name	Description
0x0	OFF	-
0x1	8CLK	244 µs
0x2	16CLK	488 µs
0x3	32CLK	977 µs
0x4	64CLK	1.95 ms
0x5	128CLK	3.91 ms
0x6	256CLK	7.81 ms
0x7	512CLK	15.6 ms
0x8	1KCLK	31.2 ms
0x9	2KCLK	62.5 ms
0xA	4KCLK	125 ms
0xB	8KCLK	250 ms
Other	-	Reserved

Note: The timing values are provided for an Ultra-Low Power Oscillator frequency of precisely 32.768 kHz. Refer to the *Electrical Characteristics* section for specific information regarding the accuracy of the Ultra-Low Power Oscillator.

Bits 3:0 – PERIOD[3:0] Period
Writing a non-zero value to this bit field enables the WDT. In Normal mode, this bit field selects the time-out interval. In Window mode, this bit field selects the open window duration.
The bits are optionally lock-protected:

- If the LOCK bit in the STATUS register is '1', all bits are change-protected (Access = R)
- If the LOCK bit in the STATUS register is '0', all bits can be changed (Access = R/W)

Value	Name	Description
0x0	OFF	-
0x1	8CLK	244 µs
0x2	16CLK	488 µs
0x3	32CLK	977 µs
0x4	64CLK	1.95 ms
0x5	128CLK	3.91 ms
0x6	256CLK	7.81 ms

Value	Name	Description
0x7	512CLK	15.6 ms
0x8	1KCLK	31.2 ms
0x9	2KCLK	62.5 ms
0xA	4KCLK	125 ms
0xB	8KCLK	250 ms
Other	-	Reserved

Note: The timing values are provided for an Ultra-Low Power Oscillator frequency of precisely 32.768 kHz. Refer to the *Electrical Characteristics* section for specific information regarding the accuracy of the Ultra-Low Power Oscillator.

29.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x40

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

Bits 7:6 – WDTMON[1:0] WDT Clock Monitor
This bit field controls the WDT Clock Monitor mechanism. This bit field is initialized from fuses by the system initialization process.

Value	Name	Description
0x0	OFF	The WDT Clock Monitor is disabled
0x1	ON	The WDT Clock Monitor is always on (Reset value)
0x2	SLEEP	The WDT Clock Monitor is always on in normal and IDLE mode but automatically disabled when the MCU is in the POWERDOWN and STANDBY sleep modes
0x3	RESERVED	Reserved

29.5.3 Status

Name: STATUS
Offset: 0x02
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	LOCK					CAUSE	CNTBUSY	SYNCBUSY
Access	R/W					R	R	R
Reset	0					0	0	0

Bit 7 – LOCK Lock
Writing this bit to '1' write-protects the CTRLA register.
It is only possible to write this bit to '1'. This bit can be cleared in Debug mode only.
If the PERIOD value in the WDTCFG fuse is non-zero, the lock will automatically be set.
This bit is under CCP.

Bit 2 – CAUSE Reset Cause
The last WDT reset source. Reset only by POR/BOR.

Value	Name	Description
0x0	OUTSIDE	Outside window. This also covers the case where one WDR is received while a prior one is processed.
0x1	TIMEOUT	Timeout

Bit 1 – CNTBUSY CNT Synchronization Busy
This bit is set while synchronizing the counter value from the WDT clock domain to the system clock domain.
This bit is cleared by the system after the synchronization is finished.
CNT may become out-of-sync with the WDT counter when CLK_PER stops, e.g., in sleep mode.
This bit is not under CCP.

Bit 0 – SYNCBUSY Synchronization Busy
This bit is set after writing to the CTRLA register while synchronizing the data from the peripheral clock domain to the WDT clock domain.
This bit is cleared after finishing the synchronization.
This bit is not under CCP.

29.5.4 Counter

Name: CNT
Offset: 0x04
Reset: 0x00
Property: -

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

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

Bits 13:0 – CNT[13:0] Counter
The WDT counter is synchronized to the peripheral clock. This register is not buffered by any TEMP registers, so a readout may result in inconsistent bytes.

30. SWDT - Synchronous Watchdog Timer

30.1 Features

- Issues a System Reset If Its Counter Is Not Cleared Before It Expires
- 24-bit Configurable Expiration Value Based Upon Either
  - Counting executed processor instructions
  - Counting system clock cycles
- Software Enabled
- 16-bit Configurable Window Determines When The Counter Can Be Cleared
- Command Sequence Required to Clear Counter
- Operates Synchronously With System Clock Using The Same Clock Source
- Configuration Lock to Prevent Unwanted Changes
- Test Features for Functional Safety
  - Counter is readable
  - To speed up testing, expiration can be accelerated

30.2 Overview

The primary function of the Synchronous Watchdog Timer (SWDT) is to generate an interrupt or notify the Error Controller in the event of a software malfunction, potentially issuing a system reset. The SWDT has a 24-bit counter that counts down when the SWDT is enabled. Two different count modes are selectable using the MODE bit in the Control A (CTRLA) register:

- MODE=0 (CLOCK): The SWDT counts clock cycles, decrementing the counter by one for every clock cycle until the count reaches zero. This mode is similar to a high-frequency synchronous WDT.
- MODE=1 (INSTRUCTION): The SWDT counts instructions, decrementing the counter by one whenever an instruction is completed until the count reaches zero. Instructions are not fetched when the processor is in sleep mode. This mode is helpful for ensuring exact control over the execution time of specific loops and system activities. This mode is also useful in systems with multiple bus hosts that compete for bus resources, possibly preventing the CPU from using the bus 100%, which would cause cycle inaccuracy in program execution. The number of CPU instructions between two points in the code would still be deterministic.

The 24-bit counter can be read by software at any time to verify that it works. Although the counter is only 24 bits, the register for reading it (CNT) is 32 bits (with the upper byte fixed at zero) for straightforward use with 32-bit data types. Because there is no mechanism to simultaneously read all four bytes, the individual bytes of the counter will be read at slightly different times and will be inconsistent. This poses no problem, however, for verifying that the counter is changing.

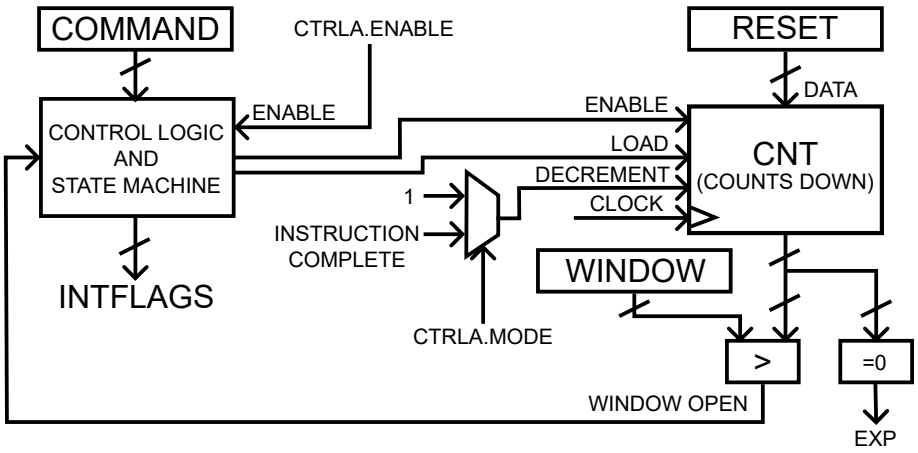
An SWDT expiration (EXP) error occurs if the counter reaches zero, which is considered a software error. The software must periodically reset the SWDT counter by writing a sequence of commands to the COMMAND register to prevent this from happening. The correct sequence of commands will load the counter with the value in the Reset Value register. Like the counter, the Reset Value is only 24 bits, but it is accessed via a 32-bit RESET register (with the upper byte fixed at zero) for straightforward use with 32-bit data types.

The SWDT is used in safety-critical applications where software functionality and sequencing failure must be detected. The main benefit of using the SWDT instead of the WDT is that the SWDT runs on CLK_PER, making it much easier to configure a tight window because measuring the expected number of counts between SWDT resets can be done with high accuracy. In contrast, the WDT is clocked independently of and asynchronously to the CPU. The WDT requires a broad window due to synchronization jitter and the wide frequency tolerance of the WDT clock source.

The SWDT cannot work when the system is asleep or the main clock has stopped, unlike the WDT. The SWDT is thus a supplement to the WDT, not a replacement.

30.2.1 Block Diagram

Figure 30-1. SWDT Block Diagram



30.3 Functional Description

30.3.1 Initialization

The following steps are recommended to initialize the SWDT for basic operation:

- Configure the Reset Value of the SWDT by writing to the RESET register
- Configure the WINDOW register. As shown in the block diagram above, this value is compared with CNT to determine when the window is open for clearing the counter.
- Enable the SWDT by simultaneously writing a '1' to the ENABLE bit and either a '0' or a '1' to the MODE selection bit (to choose count mode) in the CTRLA register

30.3.2 Clocks

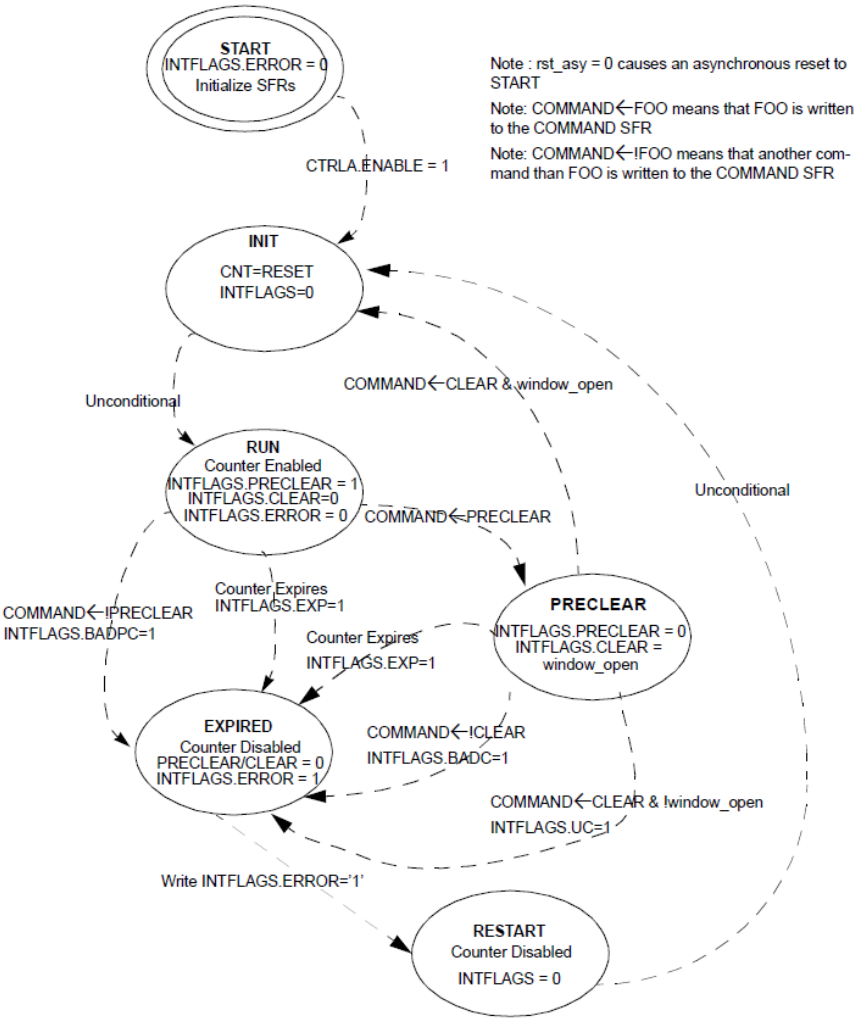
The SWDT logic is clocked by the peripheral clock, CLK_PER. The SWDT counter decrements by one for each peripheral clock or completed instruction depending on the MODE bit in the CTRLA register.

30.3.3 Operation

Before the SWDT is enabled, its counter (CNT) is zero, and all bits in the Interrupt Flags (INTFLAGS) register are zero. Once the SWDT is enabled by writing a '1' to the ENABLE bit in the CTRLA register, the value in the RESET register is loaded into the counter (CNT) and begins decrementing in accordance with the MODE bit selection.

The counter (CNT) register may be read at any time, but any attempt to write to the CNT register will have no effect. The software can reload the RESET value into the CNT register before it reaches zero and generates an expiration (EXP) event, but it must do so using a specific two-step, timed sequence. This two-step sequence requires the software to issue a PRECLEAR command followed by a CLEAR command using the COMMAND register, as described in further detail below.

Figure 30-2. SWDT States and Transitions



30.3.3.1 Step 1 (PRECLEAR)

The software must first perform a PRECLEAR action by writing PRECLEAR to the COMMAND register. This action places the SWDT into a state that permits it to be cleared by step 2.

A PRECLEAR command can be successfully accepted by the SWDT whether or not the window is open. It must be received in the correct sequence, i.e., before a CLEAR command, and not after another PRECLEAR. The PRECLEAR bit in the INTFLAGS register indicates when a PRECLEAR command can legally be received.

Writing a value other than PRECLEAR as the first step will cause an error, indicated by the ERROR bit in the INTFLAGS register. Additional bits in the INTFLAGS register indicate why an error occurred.

30.3.3.2 Step 2 (CLEAR)

After a PRECLEAR command has been accepted by the SWDT without errors, the software must perform a CLEAR action by writing CLEAR to the COMMAND register. The CLEAR command is accepted only if one and only one PRECLEAR command has already been received and the window is open, meaning that the CNT value is less than the WINDOW value. The CLEAR bit in the INTFLAGS register indicates when a CLEAR command can legally be received. If the CLEAR command is successful, the CLEAR bit in the INTFLAGS register is cleared, and the value in the RESET register is loaded into the counter (CNT).

If the CLEAR command was written to COMMAND without a prior PRECLEAR command, the BADPC and ERROR flags are set in the INTFLAGS register, indicating that a non-PRECLEAR command was received while expecting a PRECLEAR.

If the CLEAR command was written to COMMAND after a PRECLEAR command, but the window was not open when writing the CLEAR command, the UC (Unexpected Command) and ERROR flags are set in the INTFLAGS register.

The two-step clearing sequence of the SWDT depends upon the appropriate code placement of each action of the two-step sequence. The code placement of the PRECLEAR action will typically be part way through the initial portion of the application's main loop, while the code placement of the CLEAR action will be near the end of the application's main loop just before it returns to the start of the loop.

The reasoning for this placement and the requirement for a two-step action to clear the SWDT are based on the single-point fault model. In this model, the failure of the software is assumed to be caused by a single fault under the assumption that the software was designed correctly.

One fault pattern that might occur would be the repeated clearing of the SWDT counter in a tight loop. For example, a response input may be required for the application. If that response does not occur, the counter should expire, yet a single fault could defeat that detection if that fault caused a loop of just a single action that clears the SWDT counter.

A second separate action is required to clear the counter to overcome this potential failure. Separating the second action spatially and temporally reduces the possibility that the two sequential actions could occur because of a single fault.

The second action is further constrained to occur during a window interval so that it may not happen too early.

30.3.3.3 Restart After Error

If an incorrect sequence of commands is written to the COMMAND register, the ERROR flag and other flags will be set in the INTFLAGS register to indicate an error and what caused it, and the SWDT counter will stop counting. The error can be cleared by writing a '1' to the ERROR bit in the INTFLAGS register; this will clear all flags in the INTFLAGS register. Clearing the error will also restart the SWDT, meaning that the RESET value will be loaded into the CNT register and counting will begin.

30.3.3.4 Counter Expiration

When the SWDT counter expires (reaches zero), counting stops and the ERROR and EXP (Counter Expiration) flags are set in the INTFLAGS register. Writing a '1' to the ERROR bit in the INTFLAGS register will clear the error in addition to all flags in the INTFLAGS register. Clearing the error will also restart the SWDT, meaning that the RESET value will be loaded into the CNT register and counting will begin.

30.3.3.5 Test Speedup

Writing '1' to the TEST bit in the CTRLB register will immediately load the SWDT counter (CNT) with a value of one. Counter operation is otherwise as normal. This feature allows speedup of testing the SWDT functionality, which is helpful in the Functional Safety applications.

30.3.4 Configuration Change Protection

This peripheral has registers that can be placed under Configuration Change Protection (CCP) by writing a '1' to the CCP bit in the CTRLA register. CCP is a security mechanism to prevent unintentional changes to the SWDT settings. To write to one of these protected registers, write a given key to the CPU's CCP register and follow it by a write access to the protected register within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged and returns a Bus Error on the data bus.

Writing a '1' to the CCP bit in the CTRLA register sets the following registers under CCP.

Table 30-1. Registers Under Configuration Change Protection

Register	Key
SWDT.CTRLA	IOREG
SWDT.CTRLB	IOREG
SWDT.INTCTRL	IOREG
SWDT.INTFLAGS	IOREG
SWDT.RESET	IOREG
SWDT.WINDOW	IOREG
SWDT.COMMAND	IOREG

30.3.5 Lock Protection

This peripheral has registers that can be placed under Lock Protection, a security mechanism to prevent unintentional changes, by writing a '1' to the LOCK bit in the CTRLA register. Attempting to write to these registers when the LOCK bit is '1' leaves the protected register unchanged and returns a Bus Error on the data bus.

The ENABLE bit in the CTRLA register can never be cleared from software, regardless of whether Lock Protection is active. Once the SWDT is enabled, it can be disabled only by device reset.

The following registers are under Lock Protection:

Table 30-2. Registers Under Lock Protection

Register	Key
SWDT.CTRLA	LOCK
SWDT.INTCTRL	LOCK
SWDT.RESET	LOCK
SWDT.WINDOW	LOCK

When Lock Protection is active, the following registers are automatically placed under Configuration Change Protection regardless of the status of the CCP bit in the CTRLA register:

Table 30-3. Registers Automatically Under Configuration Change Protection When Lock Protection Is Active

Register	Key
SWDT.CTRLB	IOREG
SWDT.INTFLAGS	IOREG
SWDT.COMMAND	IOREG

The following table summarizes the interaction between CCP and Lock Protection:

Table 30-4. Summary of CCP and Lock Protection

CCP	LOCK	Protection of CTRLB, INTFLAGS, COMMAND	Protection of CTRLA, INTCTRL, RESET, WINDOW	Comment
0	0	No protection. Can be written anytime	No protection. Can be written anytime	ENABLE bit in the CTRLA register can only be written to '1', but not cleared by writing it to '0'
1	0	CCP protected. Any write requires a CCP unlock sequence	CCP protected. Any write requires a CCP unlock sequence	
X	1	CCP protected. Any write requires a CCP unlock sequence	Locked. cannot be written, except for the ENABLE bit in the CTRLA register	

30.3.6 Accesses to Reserved Addresses

Accesses to reserved addresses will be aborted and return a Bus Error response.

30.3.7 Events

The SWDT can generate the following events:

Table 30-5. Event Generators in SWDT

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
SWDT	ERROR	SWDT Error	Pulse	CLK_PER	One CLK_PER period

Note: The conditions for generating an event are identical to those that will raise the corresponding flag in the Interrupt Flags (INTFLAGS) register.

The SWDT does not use any events.

30.3.8 Interrupts

Table 30-6. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
SWDT	SWDT Error	Generated on SWDT Error

30.3.9 Sleep Mode Operation

The SWDT will not count in any sleep modes. When the device wakes up, the counter will continue from the value it had when entering sleep.

30.3.10 Debug Operation

The SWDT behaves identically if the CPU executes a sequence of instructions in normal mode or is single-stepped in debug mode.

30.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0					CCP	LOCK	MODE	ENABLE
0x01	CTRLB	7:0								TEST
0x02	INTCTRL	7:0					ERROR			
0x03	INTFLAGS	7:0	BADPC	BADC	UC	EXP	ERROR		PRECLEAR	CLEAR
0x04	CNT	7:0	CNT[31:0]							
		15:8	CNT[31:0]							
		23:16	CNT[31:0]							
		31:24	CNT[31:0]							
0x08	RESET	7:0	RESET[31:0]							
		15:8	RESET[31:0]							
		23:16	RESET[31:0]							
		31:24	RESET[31:0]							
0x0C	WINDOW	7:0	WINDOW[15:0]							
		15:8	WINDOW[15:0]							
0x0E	COMMAND	7:0	COMMAND[7:0]							

30.5 Register Description

30.5.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: Configuration Change Protection (when enabled by CCP bit in CTRLA)

Bit	7	6	5	4	3	2	1	0
					CCP	LOCK	MODE	ENABLE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – CCP SWDT CCP Mode
This bit enables CCP protection of most SWDT registers. It can be written only when the LOCK bit is '0'.

Value	Name	Description
0x0	DISABLE	SWDT registers are not CCP-Protected
0x1	ENABLE	SWDT registers are CCP-Protected

Bit 2 – LOCK SWDT Register Lock
This bit write-protects many of the registers in the SWDT and enables CCP protection of several other SWDT registers. After it is written to '1', it cannot be changed back to '0'.

Value	Name	Description
0x0	UNLOCKED	SWDT registers are not Lock Protected
0x1	LOCKED	SWDT registers are Lock Protected

Bit 1 – MODE SWDT Mode
This bit selects the operation mode of the SWDT. It can be written only when the LOCK bit is '0'.

Value	Name	Description
0x0	CLOCK	SWDT counts clock cycles
0x1	INSTRUCTION	SWDT counts executed instructions

Bit 0 – ENABLE Enable SWDT
Writing this bit to '1' enables the SWDT. This bit can be written to '1' even when the LOCK bit is '1'. Once written to '1' , only a reset can clear this bit.

30.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: Configuration Change Protection (when enabled by CCP bit or LOCK bit in CTRLA)

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

Bit 0 – TEST SWDT Test
Writing this bit to '1' loads the SWDT counter (CNT) with a value of one. This can be used to speed up SWDT expiration testing in Functional Safety applications. This bit will always read as '0'.

30.5.3 Interrupt Control

Name: INTCTRL
Offset: 0x02
Reset: 0x00
Property: Configuration Change Protection (when enabled by CCP bit in CTRLA)

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

Bit 3 – ERROR SWDT Error has occurred
Writing a '1' to this bit enables the interrupt. This register is writable only when the LOCK bit in the CTRLA register is '0'.

30.5.4 Interrupt Flags

Name: INTFLAGS
Offset: 0x03
Reset: 0x00
Property: Configuration Change Protection (when enabled by CCP bit or LOCK bit in the CTRLA register)

Bit	7	6	5	4	3	2	1	0
	BADPC	BADC	UC	EXP	ERROR		PRECLEAR	CLEAR
Access	R	R	R	R	R/W		R	R
Reset	0	0	0	0	0		0	0

Bit 7 – BADPC Non-PRECLEAR command received while expecting PRECLEAR
Set if a command other than PRECLEAR was received while the PRECLEAR bit in the INTFLAGS register is set. Cleared by writing a '1' to the ERROR bit.

Bit 6 – BADC Non-CLEAR command received while expecting CLEAR
Set if a successful PRECLEAR has been received earlier, and a command other than CLEAR is received. Cleared by writing a '1' to the ERROR bit.

Bit 5 – UC Unexpected Command
Set if a successful PRECLEAR has been received earlier, and a subsequent CLEAR is received before the window has become open. Cleared by writing a '1' to the ERROR bit.

Bit 4 – EXP Counter has expired
Set if the counter has expired. Cleared by writing a '1' to the ERROR bit.

Bit 3 – ERROR SWDT Error has occurred
This bit is set when BADPC, BADC, UC or EXP are set. It is cleared by writing a '1' to it. The BADPC, BADC, UC and EXP flags will be cleared in the same operation that clears ERROR.

Bit 1 – PRECLEAR PRECLEAR can be received
This flag is set when a PRECLEAR command can be received.

Bit 0 – CLEAR CLEAR can be received
This bit is set when a CLEAR command can be received.

30.5.5 Counter

Name: CNT
Offset: 0x04
Reset: 0x00
Property: -

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

Bit	23	22	21	20	19	18	17	16
	CNT[31:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT[31:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[31:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CNT[31:0] SWDT Counter

The CNT register represents the 24-bit counter value as a 32-bit quantity (with the upper byte fixed at zero) for straightforward use with 32-bit data types. The CNT register is not buffered by any TEMP registers so a readout may result in inconsistent bytes.

30.5.6 Reset Value

Name: RESET
Offset: 0x08
Reset: 0x00
Property: Configuration Change Protection (when enabled by CCP bit in the CTRLA register)

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

Bit	23	22	21	20	19	18	17	16
	RESET[31:0]							
Access	R/W	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
	RESET[31:0]							
Access	R/W	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
	RESET[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – RESET[31:0] SWDT Reset Value

The RESET register represents the 24-bit reset value as a 32-bit quantity (with the upper byte fixed at zero) for straightforward use with 32-bit data types.
This register is writable only when the LOCK bit in the CTRLA register is '0'.

30.5.7 Window

Name: WINDOW
Offset: 0x0C
Reset: 0x00
Property: Configuration Change Protection (when enabled by CCP bit in the CTRLA register)

Bit	15	14	13	12	11	10	9	8
	WINDOW[15:0]							
Access	R/W	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
	WINDOW[15:0]							
Access	R/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 – WINDOW[15:0] SWDT Window

The WINDOW register represents the 16-bit window value that determines when a CLEAR command can legally be received.
This register is writable only when the LOCK bit in the CTRLA register is '0'.

30.5.8 Command

Name: COMMAND
Offset: 0x0E
Reset: 0x00
Property: Configuration Change Protection (when enabled by the CCP bit or the LOCK bit in the CTRLA register)

Bit	7	6	5	4	3	2	1	0
	COMMAND[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – COMMAND[7:0] SWDT Command
This bit field loads either a PRECLEAR command or a CLEAR command into the SWDT. All other commands are illegal. To successfully clear the SWDT, the commands must arrive in the correct sequence and at expected times.

Value	Name	Description
0x08	PRECLEAR	Step 1 in the SWDT clearing sequence
0x10	CLEAR	Step 2 in the SWDT clearing sequence

31. TCA - 16-bit Timer/Counter Type A

31.1 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

31.2 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 can use a period setting for timing. The compare channels can be used with the base counter to perform a 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.

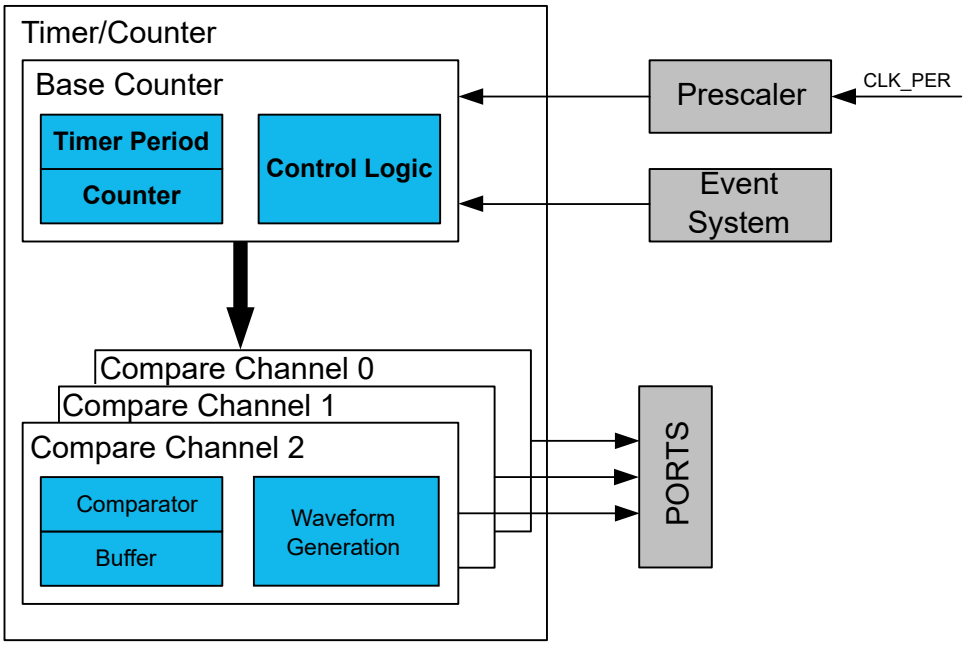
A timer/counter can be clocked and timed from the peripheral clock, with optional prescaling, or from the Event System. The Event System can also be used for direction control or synchronizing operations.

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. Depending on the used mode, addressing registers or using bit masks and group configurations is done as follows: Either TCAn.SINGLE.REGISTER or TCAn.SPLIT.REGISTER for the registers and TCA_SINGLE_CLKSEL_DIV1_gc or TCA_SPLIT_CLKSEL_DIV1_gc as an example for the bit masks and group configurations.

In this section the registers will be addressed as TCAn.REGISTER.

The figure below shows a block diagram of the 16-bit timer/counter with closely related peripheral modules (in gray).

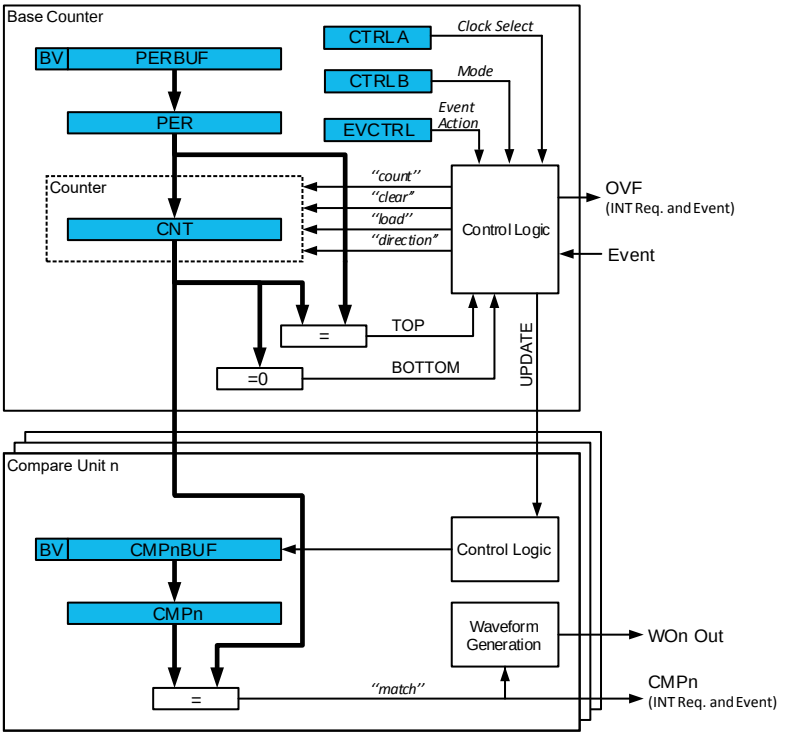
Figure 31-1. 16-Bit Timer/Counter and Closely Related Peripherals



31.2.1 Block Diagram

The figure below shows a detailed block diagram of the timer/counter.

Figure 31-2. Timer/Counter Block Diagram



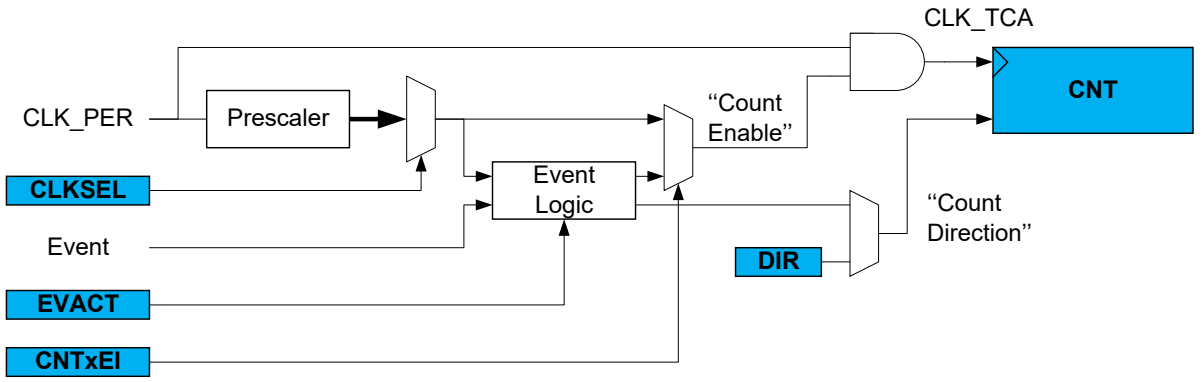
The Counter (TCAn.CNT) register, Period and Compare (TCAn.PER and TCAn.CMPn) registers, and their corresponding buffer registers (TCAn.PERBUF and TCAn.CMPnBUF) are 16-bit registers. All buffer registers have a Buffer Valid (BV) flag indicating when the buffer contains a new value.

During ordinary 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. After the triggering, the interrupt requests, events, or waveform output changes will occur on the next CLK_TCA cycle.

CLK_TCA is either the prescaled peripheral clock or events from the Event System, as shown in the figure below.

Figure 31-3. Timer/Counter Clock Logic



31.2.2 Signal Description

Signal	Description	Type
WOn	Digital output	Waveform output

31.3 Functional Description

31.3.1 Definitions

The following definitions are used throughout the documentation:

Table 31-1. 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.

31.3.2 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.

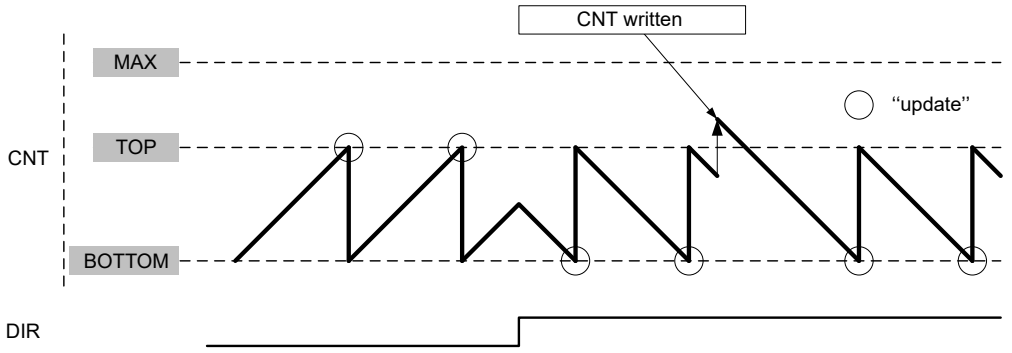
31.3.3 Operation

31.3.3.1 Normal Operation

In ordinary operation, the counter counts 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 peripheral clock (CLK_PER), prescaled according to the Clock Select (CLKSEL) bit field in the Control A (TCAn.CTRLA) register, gives the clock ticks.

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 the BOTTOM is reached.

Figure 31-4. Normal Operation



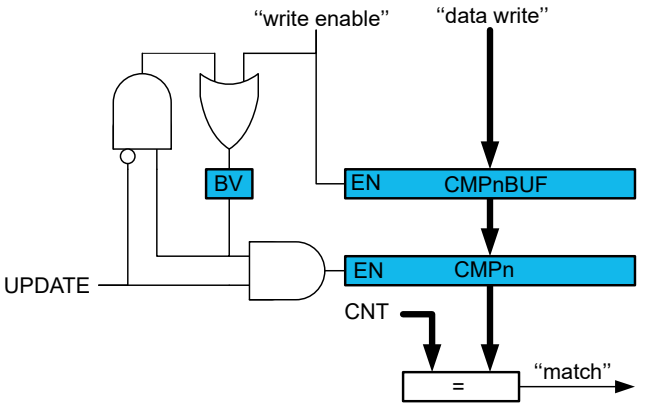
It is possible to change the counter value in the Counter (TCAn.CNT) register when the counter is running. The write access to TCAn.CNT register has higher priority than count, clear or reload, and will be immediate. The direction of the counter can also be changed during ordinary operation by writing to the Direction (DIR) bit in the Control E (TCAn.CTRLE) register.

31.3.3.2 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. The figure below shows this for a Compare (CMPn) register.

Figure 31-5. Period and Compare Double Buffering



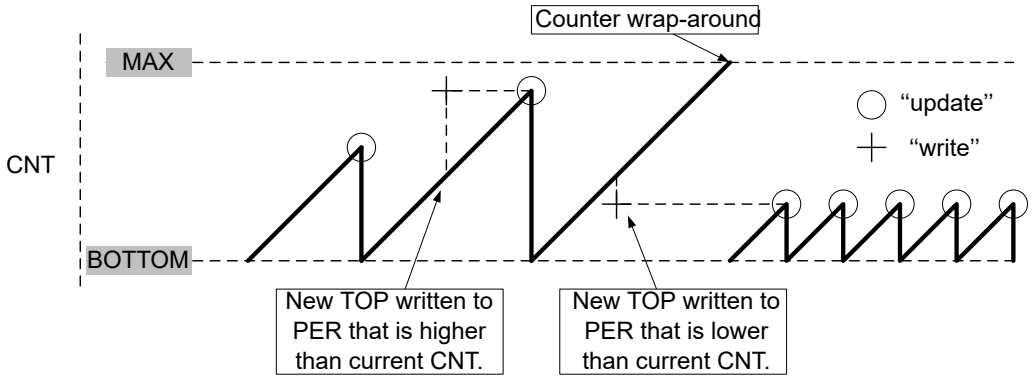
Both the TCAn.CMPn and TCAn.CMPnBUF registers are available as I/O registers, allowing the initialization and bypassing of the buffer register and the double-buffering function.

31.3.3.3 Changing the Period

The Counter period is changed by writing a new TOP value to the Period (TCAn.PER) register.

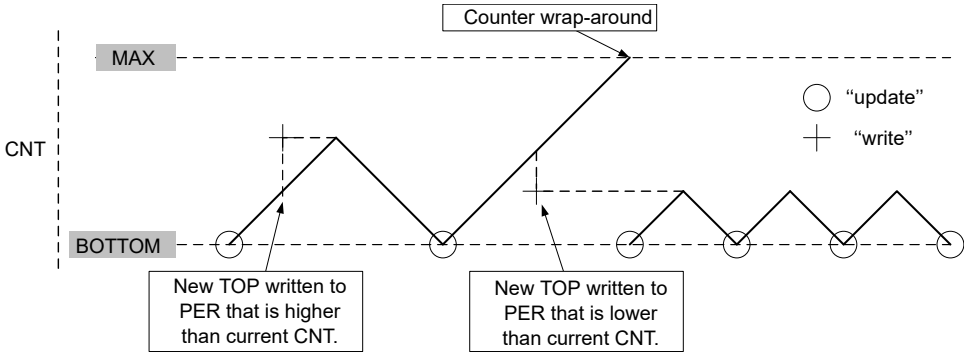
No Buffering: Any period update is immediate if not using double-buffering.

Figure 31-6. Changing the Period Without Buffering



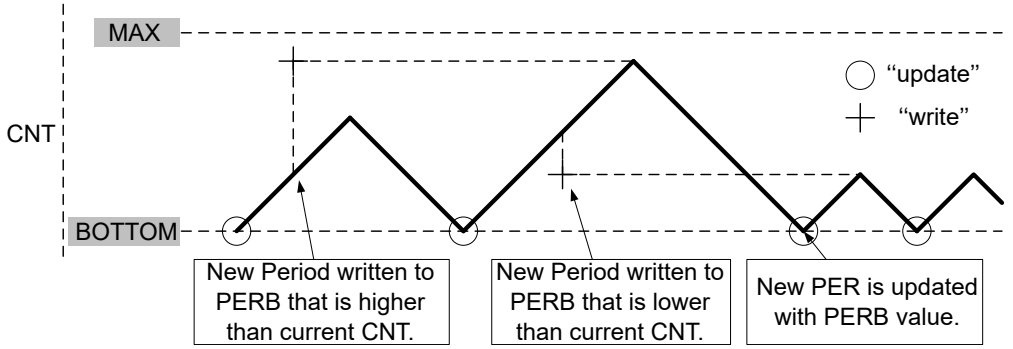
A counter wrap-around can occur in any mode of operation when counting up without buffering, as the TCAn.CNT and TCAn.PER registers are continuously compared. If writing a new TOP value to TCAn.PER lower than the current TCAn.CNT, the counter will wrap first before a compare match occurs.

Figure 31-7. Unbuffered Dual-Slope Operation



With Buffering: When using double-buffering, the buffer can be written at any time and still maintain the correct operation. TCAn.PER is always updated on the UPDATE condition, as shown for dual-slope operation in the figure below. This prevents wrap-around and the generation of odd waveforms.

Figure 31-8. Changing the Period Using Buffering



Note: Buffering is used in figures illustrating TCA operation if not otherwise specified.

31.3.3.4 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 a double-buffer capability equivalent to the one 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 compared to the counter value (TCAn.CNT) from the next count.

31.3.3.4.1 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:

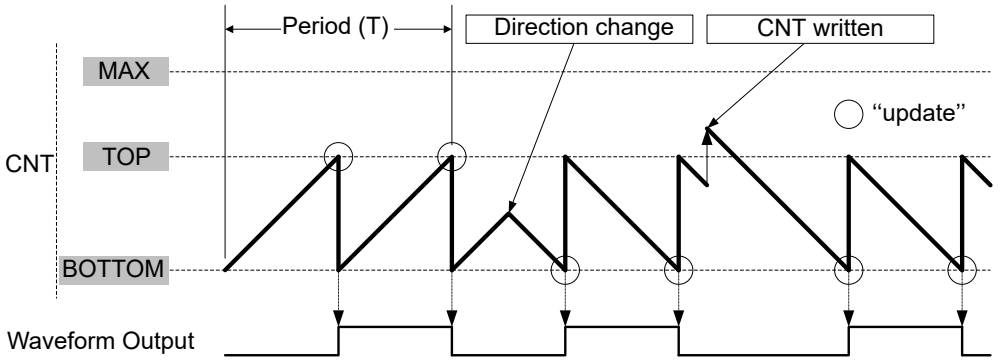
1. A Waveform Generation mode must be selected by writing the Waveform Generation Mode (WGMODE) bit field in the TCAn.CTRLB register.
2. The used compare channels must be enabled (CMPnEN = `1` in TCAn.CTRLB), which will override the output value for the corresponding pin. An alternative pin can be selected by configuring the Port Multiplexer (PORTMUX). Refer to the *PORTMUX - Port Multiplexer* 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 - I/O Pin Configuration* section for details.

Note: In Normal mode, WO0-2 are the only waveform outputs available. Split mode must be enabled to use WO3-5.

31.3.3.4.2 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 31-9. Frequency Waveform Generation



The following equation defines the waveform frequency (f_{FRQ}):

$$f_{FRQ} = \frac{f_{CLK\_PER}}{2N(CMP0+1)}$$

where *N* represents the prescaler divider used (see the 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).

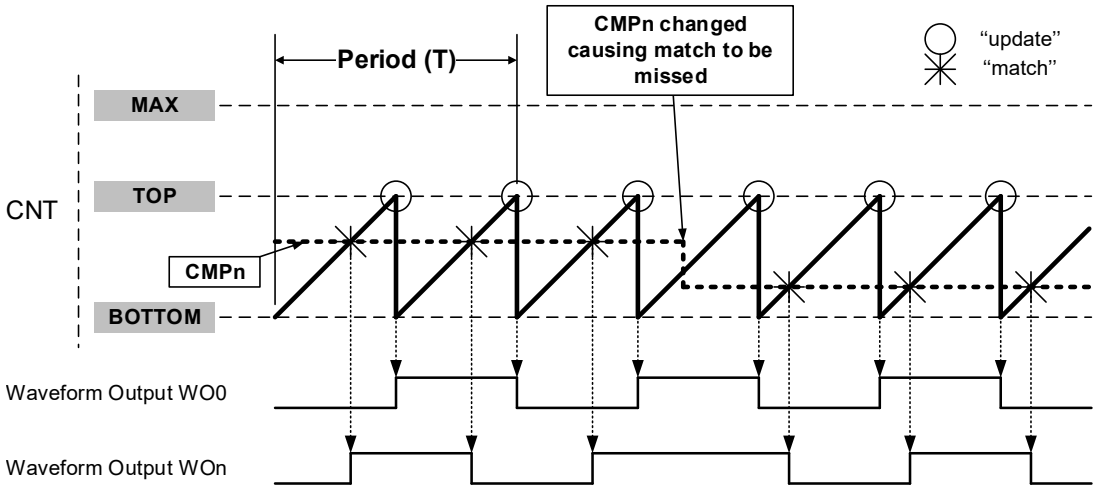
Use the TCAn.CMP1 and TCAn.CMP2 registers to get additional waveform outputs WOn. The waveforms WOn can either be identical or offset to WO0. The offset can be influenced by TCAn.CMPn, TCAn.CNT and the count direction. The offset in seconds t_{Offset} can be calculated using the equations in the table below. The equations are only valid when CMPn < CMP0.

Table 31-2. Offset Equation Overview

Equation	Count Direction	CMPn vs. CNT State	Offset
$t_{Offset} = \left(\frac{CMP0 - CMPn}{CMP0 + 1}\right)\left(\frac{T}{2}\right)$	UP	CMPn ≥ CNT	WOn leading WO0
	DOWN	CMP0 ≤ CNT	WOn trailing WO0
		CMP0 > CNT and CMPn >CNT	WOn trailing WO0
$t_{Offset} = \left(\frac{CMPn + 1}{CMP0 + 1}\right)\left(\frac{T}{2}\right)$	UP	CMPn < CNT	WOn trailing WO0
	DOWN	CMPn ≤ CNT	WOn leading WO0

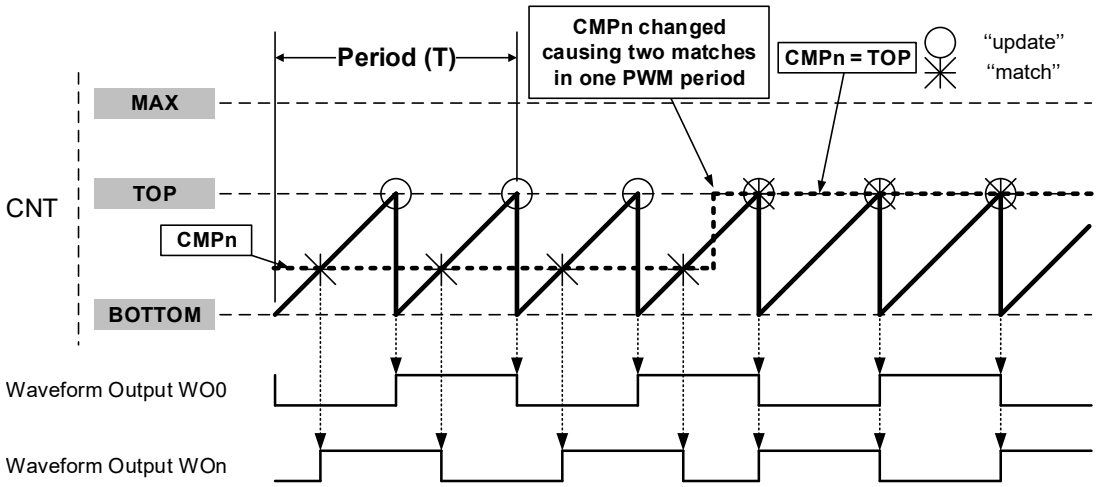
The figure below shows the leading and trailing offset for WOn, where both equations can be used. The correct equation is determined by count direction, and the state of CMPn vs. CNT when the timer is enabled or CMPn is changed.

Figure 31-10. Offset When Counting Up



The figure below shows how changing CMPn during run-time can invert the waveform.

Figure 31-11. Inverting Waveform Output

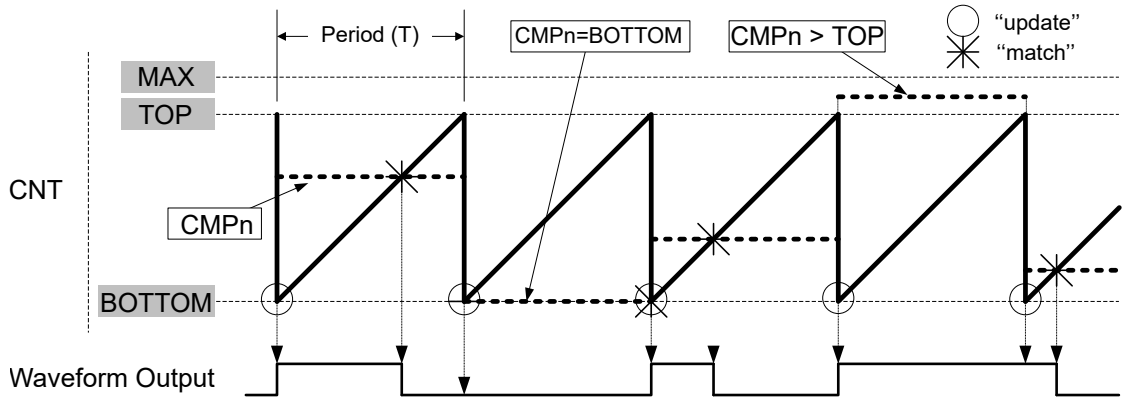


31.3.3.4.3 Single-Slope PWM Generation

For single-slope Pulse-Width Modulation (PWM) generation, the TCAn.PER register controls the period (T), while the TCAn.CMPn register values 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 31-12. Single-Slope Pulse-Width Modulation



Notes:

1. The representation in the figure above is valid when CMPn is updated using CMPnBUF.
2. For single-slope Pulse-Width Modulation (PWM) generation, the counter counting from TOP to BOTTOM is not supported.

The Period (TCAn.PER) register defines the PWM resolution. The minimum resolution is two bits (TCAn.PER = `0x0003`), and the maximum resolution is 16 bits (TCAn.PER = MAX).

The following equation calculates the exact resolution in bits for single-slope PWM (R_{PWM_SS}):

$$R_{PWM\_SS} = \frac{\log(PER+1)}{\log(2)}$$

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:

$$f_{PWM\_SS} = \frac{f_{CLK\_PER}}{N(PER+1)}$$

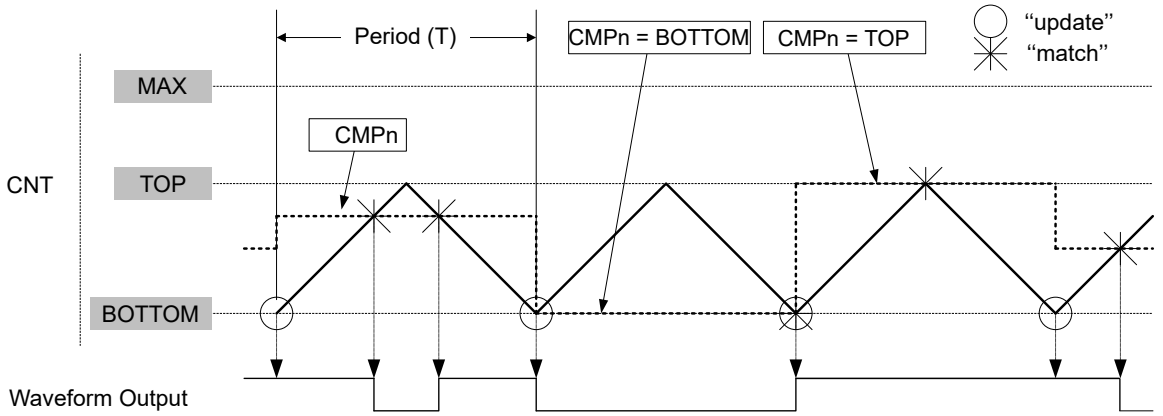
31.3.3.4.4 Dual-Slope PWM Generation

For the dual-slope PWM generation, the TCAn.PER controls the period (T), while the TCAn.CMPn register values control the duty cycle of the WG output.

The figure below shows how, for dual-slope PWM, the counter repeatedly counts 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 produces a static low signal on WOn, while CMPn = TOP produces a static high signal on WOn.

Figure 31-13. Dual-Slope Pulse-Width Modulation



Note: The representation in the figure above is valid when CMPn is updated using CMPnBUF.

The Period (TCAn.PER) register defines the PWM resolution. The minimum resolution is two 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}):

$$R_{PWM\_DS} = \frac{\log(PER+1)}{\log(2)}$$

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:

$$f_{PWM\_DS} = \frac{f_{CLK\_PER}}{2N \cdot PER}$$

N represents the prescaler divider used.

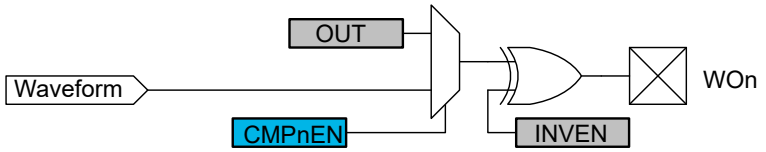
Using dual-slope PWM results in approximately half the maximum operation frequency compared to single-slope PWM operation due to twice the number of timer increments per period.

31.3.3.4.5 Port Override for Waveform Generation

The corresponding port pin direction must be set as output (PORTx.DIR[n] = `1`) to make the waveform generation available on the port pins. The TCA will override the port pin values when the compare channel is enabled (CMPnEN = `1` in the TCAn.CTRLB register), 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 (PORTx.OUT) on the corresponding port pin. Enabling inverted I/O on the port pin (INVEN = `1` in the PORTx.PINnCTRL register) inverts the corresponding WG output.

Figure 31-14. Port Override for Timer/Counter Type A



31.3.3.5 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 and all waveform 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).

31.3.3.6 Split Mode - Two 8-Bit Timer/Counters

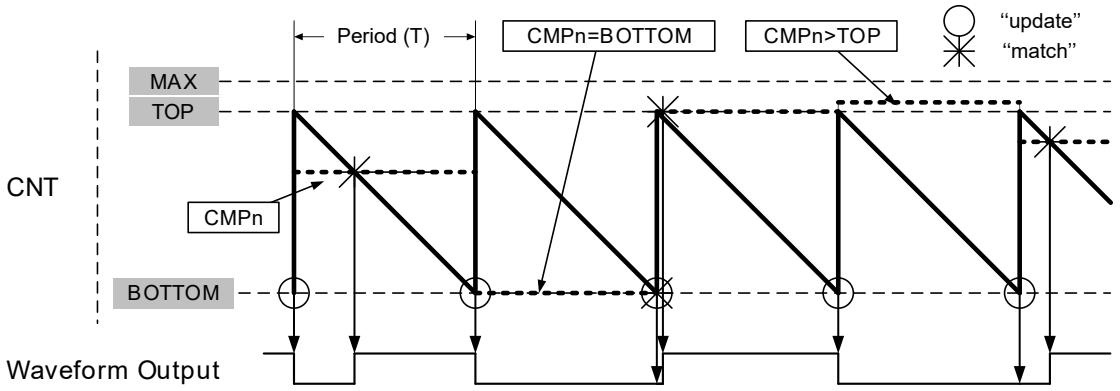
Split Mode Overview

A Split mode is provided to double the number of timers and PWM channels in the TCA. 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.

The figure below shows single-slope PWM generation in Split mode. The waveform generator output is cleared at BOTTOM and set on the compare match between the counter value (TCAn.CNT) and the Compare n (TCAn.CMPn) register.

CMPn = BOTTOM or CMPn > TOP will produce a static low signal on WOn.

Figure 31-15. Single-Slope Pulse-Width Modulation in Split mode



Note: The maximum duty-cycle of the waveform output is TOP/(TOP+1).

Activating Split mode changes the functionality of some registers and register bits. The modifications are described in a separate register map (see Register Summary - Split Mode).

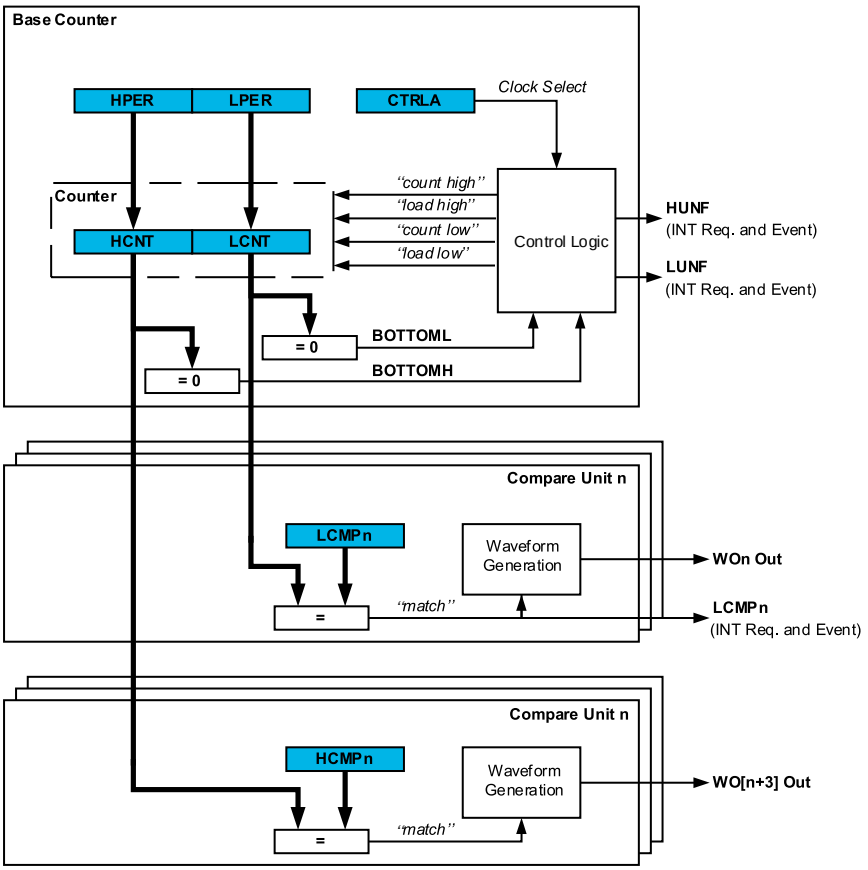
Split Mode Differences Compared to Normal Mode

- 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 the TCAn.CTRLB register)
- 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 31-16. 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 the TCAn.CTRLA register) and doing a hard Reset (CMD = RESET in the TCAn.CTRLESET register) 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.

31.3.4 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. 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 31-3. Event Generators in TCA

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
TCAn	OVF_LUNF	Normal mode: Overflow Split mode: Low byte timer underflow	Pulse	CLK_PER	One CLK_PER period
	HUNF	Normal mode: Not available Split mode: High byte timer underflow	Pulse	CLK_PER	One CLK_PER period
	CMP0_LCMP0	Normal mode: Compare Channel 0 match Split mode: Low byte timer Compare Channel 0 match	Pulse	CLK_PER	One CLK_PER period
	CMP1_LCMP1	Normal mode: Compare Channel 1 match Split mode: Low byte timer Compare Channel 1 match	Pulse	CLK_PER	One CLK_PER period
	CMP2_LCMP2	Normal mode: Compare Channel 2 match Split mode: Low byte timer Compare Channel 2 match	Pulse	CLK_PER	One CLK_PER period

Note: 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 31-4. 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.

Notes:

1. Event inputs are not used in Split mode.
2. Event actions with level input detection only work reliably if the event frequency is less than the timer's frequency.

Refer to the *EVSYS - Event System* section for more details regarding event types and Event System configuration.

31.3.5 Interrupts

Table 31-5. 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 31-6. 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.

31.3.6 Sleep Mode Operation

TCA is by default disabled in Standby sleep mode. It will be halted as soon as entering sleep mode.

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 operations halt in Power-Down sleep mode.

31.4 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]							

31.5 Register Description - Normal Mode

31.5.1 Control A - Normal Mode

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

31.5.2 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.

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', the LUPD bit will be set to '1' until the Buffer Valid (CMPnBV) bits of all enabled compare channels are '1'. This condition will clear the LUPD bit.
It will remain cleared until the following UPDATE condition, where the buffer values will be transferred to the CMPn registers, and the LUPD bit 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 bit in the TCAn.CTRLE register is not altered by the system
1	LUPD bit in the TCAn.CTRLE register is set and cleared automatically

Bits 2:0 – WGMODE[2:0] Waveform Generation Mode
This bit field selects the Waveform Generation mode and controls 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. The waveform generator output will only be directed to the port pins if setting the corresponding CMPnEN bit for all other modes. The port pin direction must be set as output.

Table 31-7. 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.

31.5.3 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 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.

Note: When connecting the output to the pad, overriding these bits will not work unless the CMPnEN bits in the TCAn.CTRLB register have been set. The CMPnEN bits in the TCAn.CTRLB register are bypassed when connecting the output to CCL.

31.5.4 Control D - Normal Mode

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 and will work as two 8-bit timer/counters. The register map will change compared to the normal 16-bit mode.

31.5.5 Control Register E Clear - Normal Mode

Name: CTRLECLR
Offset: 0x04
Reset: 0x00
Property: -

Use this register 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
This bit field is used for software control of update, restart, and Reset of the timer/counter. The command bit field is 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 performing an update.

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. This setting will not prevent an update issued by the Command bit field.

Bit 0 – DIR Counter Direction
Usually, this bit is controlled in hardware by the Waveform Generation mode or by event actions but can also be changed from the software.

Value	Description
0	The counter is counting up (incrementing)
1	The counter is counting down (decrementing)

31.5.6 Control Register E Set - Normal Mode

Name: CTRLESET
Offset: 0x05
Reset: 0x00
Property: -

Use this register 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
This bit field is used for software control of update, restart, and Reset of the timer/counter. The command bit field always reads 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 performing an update.

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. This setting will not prevent an update issued by the Command bit field.

Bit 0 – DIR Counter Direction
Usually, this bit is controlled in hardware by the Waveform Generation mode or by event actions but can also be changed from the software.

Value	Description
0	The counter is counting up (incrementing)
1	The counter is counting down (decrementing)

31.5.7 Control Register F Clear

Name: CTRLFCLR
Offset: 0x06
Reset: 0x00
Property: -

Use this register 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 automatically clear 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 automatically clears on an UPDATE condition.

31.5.8 Control Register F Set

Name: CTRLFSET
Offset: 0x07
Reset: 0x00
Property: -

Use this register 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 automatically clear 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 automatically clears on an UPDATE condition.

31.5.9 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	Counts the prescaled clock cycles or counts the matching events according to the setting for event input A. The event signal controls the count direction, up when low and down when high. The direction is latched when the counter counts.
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. The direction is latched when the counter counts.
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

31.5.10 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.

31.5.11 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 (TCAn.CNT) register and the corresponding Compare n (TCAn.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.

31.5.12 Debug Control Register - Normal Mode

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

31.5.13 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 *AVR CPU* 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 Bits for 16-bit Access

31.5.14 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`.

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
This bit field holds the MSB of the 16-bit Counter register.

Bits 7:0 – CNT[7:0] Counter Low Byte
This bit field holds the LSB of the 16-bit Counter register.

31.5.15 Period Register - Normal Mode

Name: PER
Offset: 0x26
Reset: 0xFFFF
Property: -

The TCAn.PER register 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
This bit field holds the MSB of the 16-bit Period register.

Bits 7:0 – PER[7:0] Periodic Low Byte
This bit field holds the LSB of the 16-bit Period register.

31.5.16 Compare n Register - Normal Mode

Name: CMPn
Offset: 0x28 + n*0x02 [n=0..2]
Reset: 0x00
Property: -

This register continuously compares to the counter value. Usually, the outputs from the comparators are used to generate waveforms.

The 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
This bit field holds the MSB of the 16-bit Compare register.

Bits 7:0 – CMP[7:0] Compare Low Byte
This bit filed holds the LSB of the 16-bit Compare register.

31.5.17 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
This bit field holds the MSB of the 16-bit Period Buffer register.

Bits 7:0 – PERBUF[7:0] Period Buffer Low Byte
This bit field holds the LSB of the 16-bit Period Buffer register.

31.5.18 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
This bit field holds the MSB of the 16-bit Compare Buffer register.

Bits 7:0 – CMPBUF[7:0] Compare Low Byte
This bit field holds the LSB of the 16-bit Compare Buffer register.

31.6 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]							

31.7 Register Description - Split Mode

31.7.1 Control A - Split Mode

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

31.7.2 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.

31.7.3 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 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 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.

Note: When the output is connected to the pad, overriding these bits will not work unless the xCMPnEN bits in the TCAn.CTRLB register have been set. If the output is connected to CCL, the xCMPnEN bits in the TCAn.CTRLB register are bypassed.

31.7.4 Control D - Split Mode

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 and will work as two 8-bit timer/counters. The register map will change compared to the normal 16-bit mode.

31.7.5 Control Register E Clear - Split Mode

Name: CTRLECLR
Offset: 0x04
Reset: 0x00
Property: -

Use this register 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
This bit field is used for software control of restart and reset of the timer/counter. The command bit field always reads 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
This bit field configures what timer/counters the command given by the CMD-bits will apply to.

Value	Name	Description
0x0	NONE	None
0x1	-	Reserved
0x2	-	Reserved
0x3	BOTH	Command (CMD) will apply to both low byte and high byte timer/counter

31.7.6 Control Register E Set - Split Mode

Name: CTRLESET
Offset: 0x05
Reset: 0x00
Property: -

Use this register 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 is used for software control of restart and reset of the timer/counter. The command bit field always reads as '0'. The CMD bit field must be used together with the Command Enable (CMDEN) bits. Using the RESET command requires CMDEN to be selected with both low byte and high byte timer/counter.

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
This bit field configures what timer/counters the command given by the CMD-bits will apply to.

Value	Name	Description
0x0	NONE	None
0x1	-	Reserved
0x2	-	Reserved
0x3	BOTH	Command (CMD) will apply to both low byte and high byte timer/counter

31.7.7 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.

31.7.8 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. Software must clear the LCMPn flag as it will not be cleared automatically. Writing a '1' to its bit location will do this.

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. Writing a '1' to its bit location will do this.

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. Writing a '1' to its bit location will do this.

31.7.9 Debug Control Register - Split Mode

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

31.7.10 Low Byte Timer Counter Register - Split Mode

Name: LCNT
Offset: 0x20
Reset: 0x00
Property: -

The TCAn.LCNT register 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
This bit field defines the counter value of the low byte timer.

31.7.11 High Byte Timer Counter Register - Split Mode

Name: HCNT
Offset: 0x21
Reset: 0x00
Property: -

The TCAn.HCNT register 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
This bit field defines the counter value in high byte timer.

31.7.12 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
This bit field holds the TOP value for the low byte timer.

31.7.13 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
This bit field holds the TOP value for the high byte timer.

31.7.14 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
This bit field holds the compare value of channel n that is compared to TCAn.LCNT.

31.7.15 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
This bit field holds the compare value of channel n that is compared to TCAn.HCNT.

32. TCB - 16-Bit Timer/Counter Type B

32.1 Features

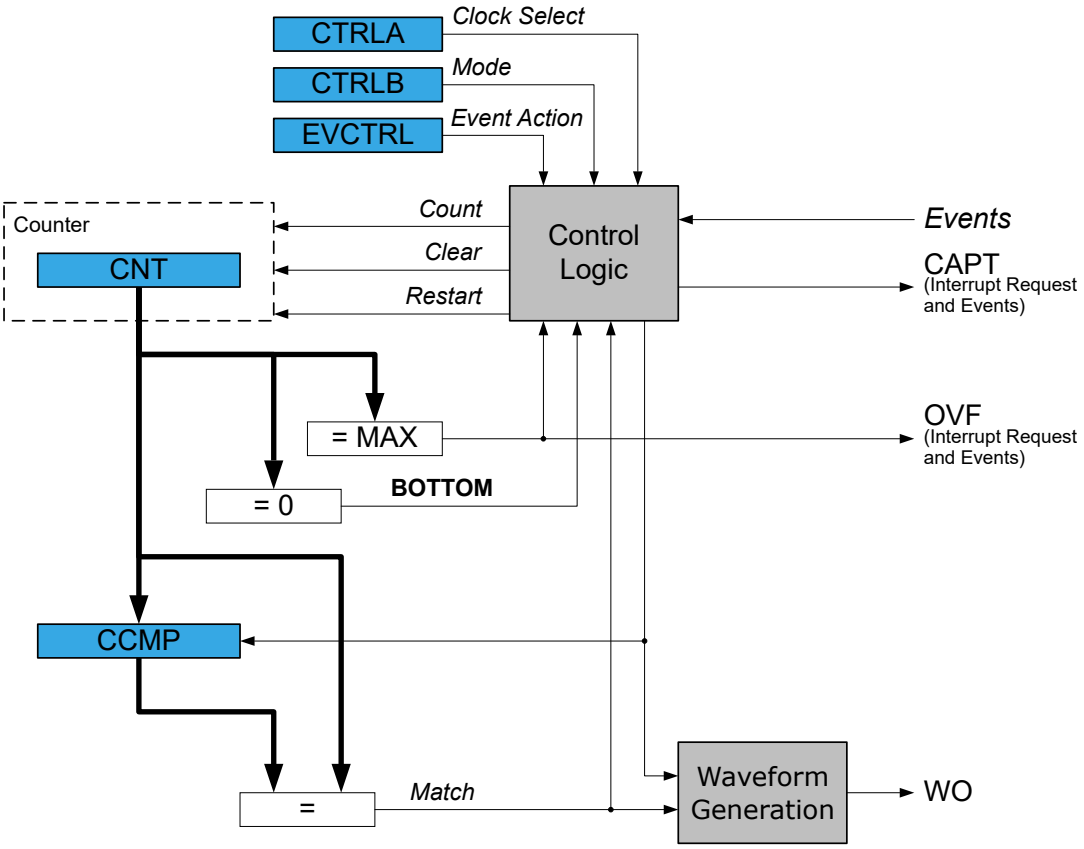
- 16-bit Counter Operation Modes:
  - Periodic interrupt
  - Time-out check
  - Input capture
    - On event
    - Frequency measurement
    - Pulse-width measurement
    - Frequency and pulse-width measurement
    - 32-bit capture
  - Single-shot
  - 8-bit Pulse-Width Modulation (PWM)
- Noise Canceler on Event Input
- Synchronize Operation with TCAn

32.2 Overview

The 16-bit Timer/Counter type B (TCB) capabilities include frequency and waveform generation and input capture on event with time and frequency measurement of digital signals. The TCB peripheral consists of a base counter and control logic that can be set in one of eight different modes, each providing unique functionality. The base counter is clocked by the peripheral clock with optional prescaling.

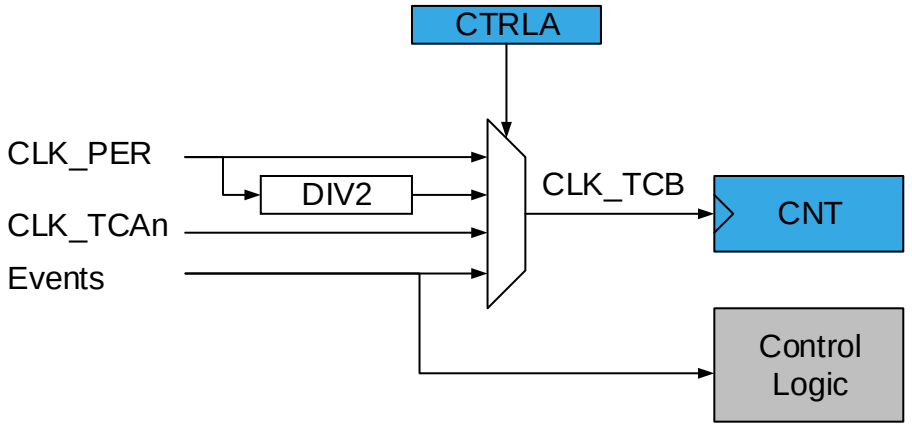
32.2.1 Block Diagram

Figure 32-1. TCB 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 32-2. 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.

32.2.2 Signal Description

Signal	Description	Type
WO	Digital Asynchronous Output	Waveform Output

32.3 Functional Description

32.3.1 Definitions

The following definitions are used throughout the data sheet:

Table 32-1. 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	Counter (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.

32.3.2 Initialization

By default, the TCB peripheral is in Periodic Interrupt mode. Follow these steps to start using it:

1. Write a TOP value to the Capture/Compare (TCBn.CCMP) register.
2. 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.

3. The counter value can be read from the Counter (TCBn.CNT) register. The peripheral will generate a CAPT interrupt and event when the CNT value reaches TOP.
   a. If the Compare/Capture register is modified to a value lower than the current CNT, the peripheral will count to MAX and wrap around.
   b. At MAX, an OVF interrupt and event will be generated.

32.3.3 Operation

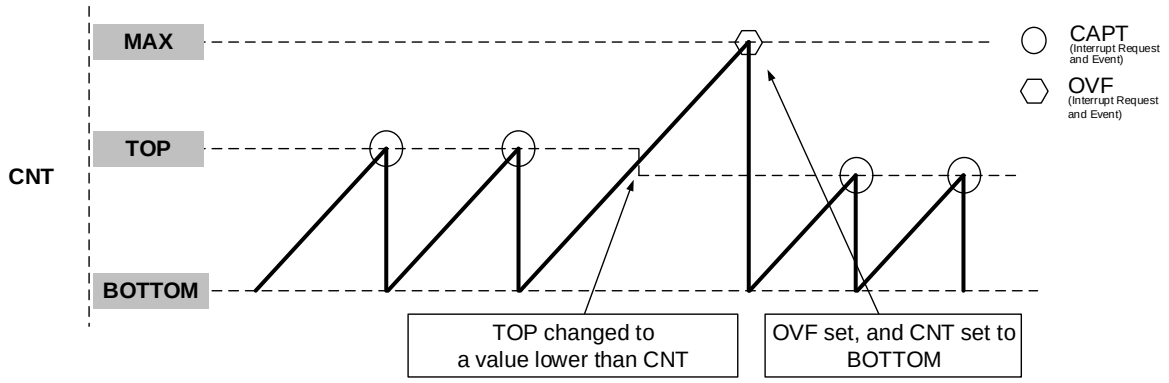
32.3.3.1 Modes

The TCB peripheral can be configured to run in one of the eight different modes described in the sections below. The event pulse must be longer than one system clock cycle to ensure edge detection.

32.3.3.1.1 Periodic Interrupt Mode

In the Periodic Interrupt mode, the counter counts to the capture value and restarts from BOTTOM. A CAPT interrupt and event are generated when the CNT equals TOP. If TOP is updated to a value lower than CNT when reaching MAX, an OVF interrupt and event are generated, and the counter restarts from BOTTOM.

Figure 32-3. Periodic Interrupt Mode



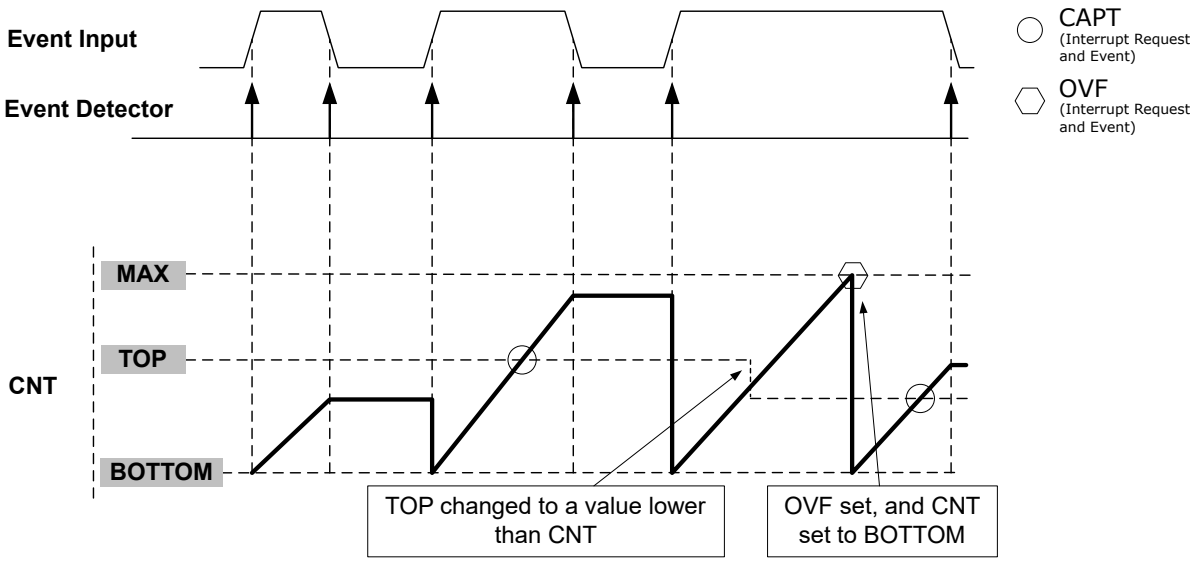
32.3.3.1.2 Time-Out Check Mode

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 the Freeze state, the counter restarts on a new Start edge.

This mode requires TCB to be configured as an event user and is explained in the Events section.

The 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 are generated, and the counter restarts from BOTTOM. In the Freeze state, reading the Counter (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 32-4. Time-Out Check Mode



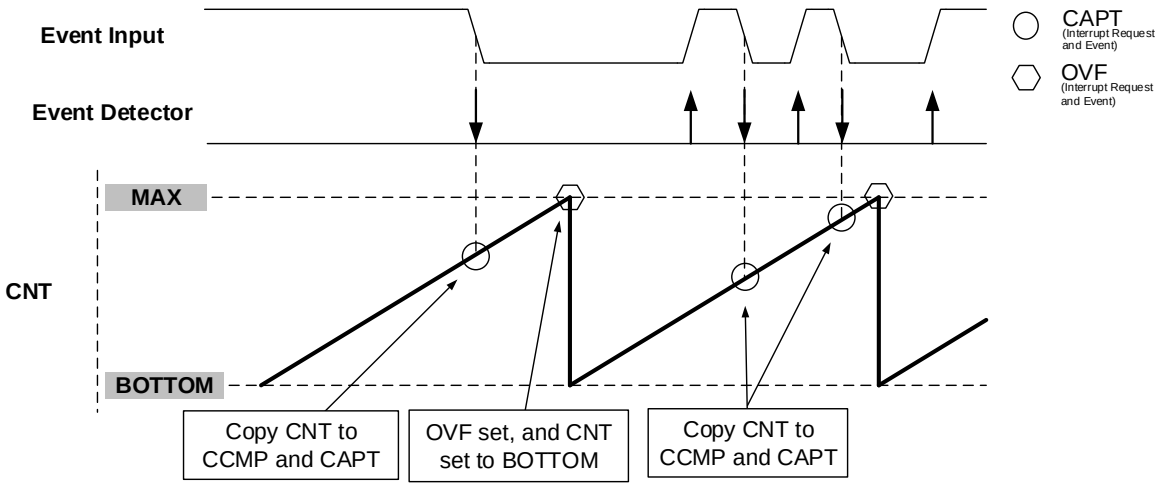
32.3.3.1.3 Input Capture on Event Mode

The counter counts from BOTTOM to MAX in the Input Capture on Event mode. When an event is detected, the Counter (TCBn.CNT) register value is transferred to the Capture/Compare (TCBn.CCMP) register, and a CAPT interrupt and event are 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 the Events section.

The figure below shows the input capture unit configured to capture the falling edge of the event input signal. The CAPT interrupt flag is cleared automatically after reading the low byte of the Capture/Compare (TCBn.CCMP) register. An OVF interrupt and event are generated when the CNT is MAX.

Figure 32-5. Input Capture on Event



Important: It is recommended to write `0x0000` to the Counter (TCBn.CNT) register when entering this mode from any other mode.

32.3.3.1.4 Input Capture Frequency Measurement Mode

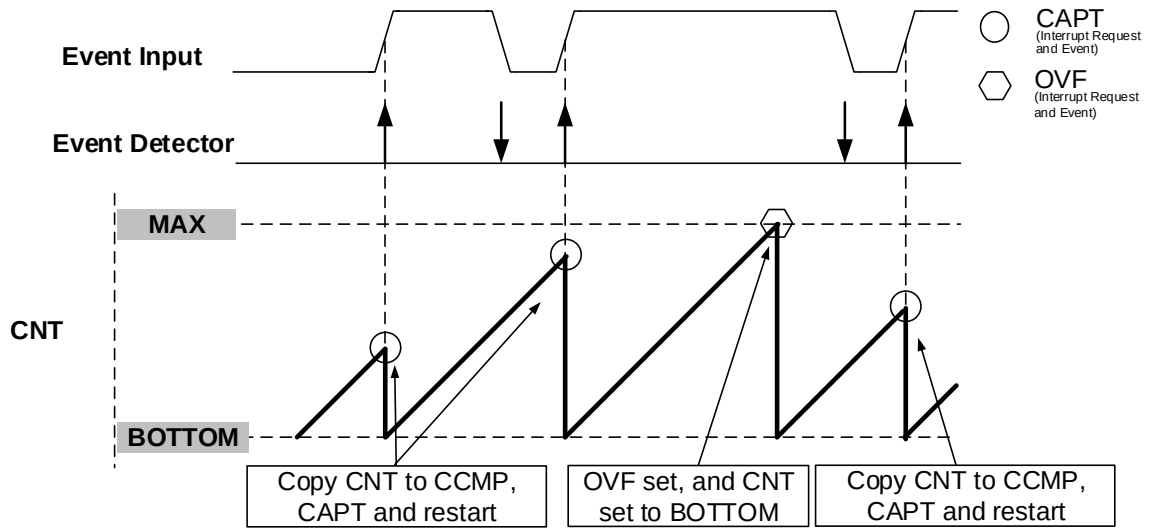
In the Input Capture Frequency Measurement mode, the TCB peripheral captures the counter value and restarts on either a positive or negative edge of the event input signal.

The CAPT interrupt flag is automatically cleared after reading the low byte of the Capture/Compare (TCBn.CCMP) register. When the CNT value is MAX, an OVF interrupt and event are generated.

This mode requires TCB to be configured as an event user and is explained in the Events section.

The figure below illustrates this mode configured to act on a rising edge.

Figure 32-6. Input Capture Frequency Measurement

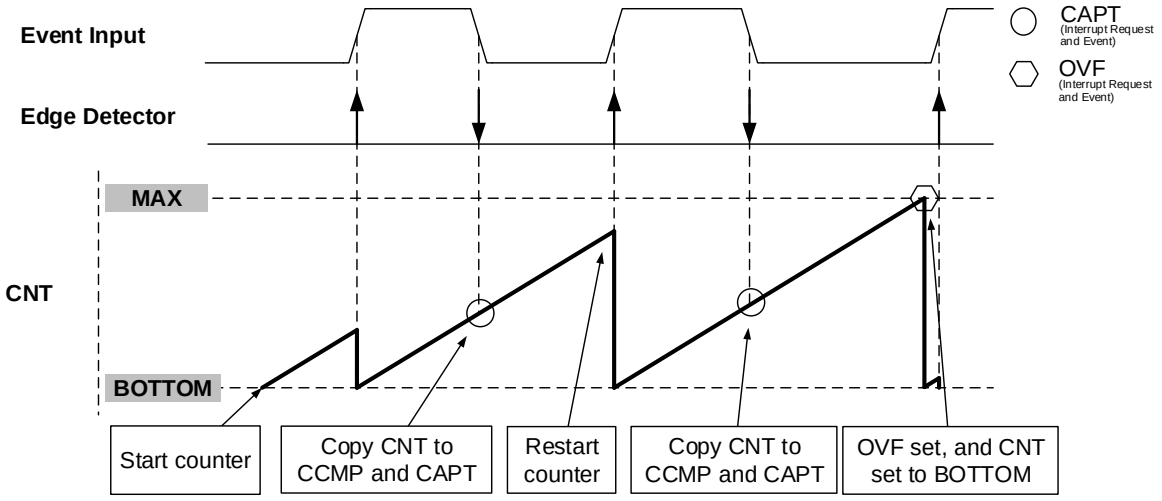


32.3.3.1.5 Input Capture Pulse-Width Measurement Mode

In the Input Capture Pulse-Width Measurement mode, the input capture pulse-width measurement restarts the counter on a positive edge and captures the next falling edge before an interrupt request is generated. The CAPT interrupt flag is cleared automatically after reading the low byte of the Compare/Capture (TCBn.CCMP) register. An OVF interrupt and event are generated when the CNT is MAX. The TCB peripheral 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 the Events section.

Figure 32-7. Input Capture Pulse-Width Measurement



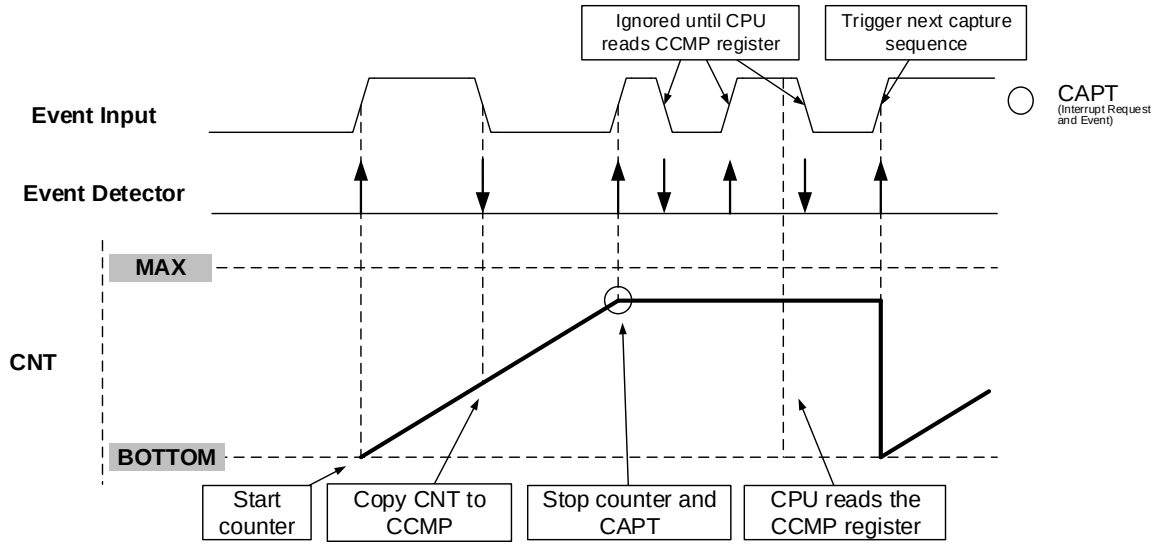
32.3.3.1.6 Input Capture Frequency and Pulse-Width Measurement Mode

In the Input Capture Frequency and Pulse-Width Measurement mode, the TCB peripheral starts 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 and will set the CAPT interrupt flag.

This mode requires TCB to be configured as an event user and is explained in the Events section.

The CAPT interrupt flag is cleared automatically after reading the low byte of the Compare/Capture (TCBn.CCMP) register, and the TCB peripheral is ready for a new capture sequence. Therefore, read the Counter (TCBn.CNT) register before the Capture/Compare (TCBn.CCMP) register since it is reset to BOTTOM at the next positive edge of the event input signal. An OVF interrupt and event are generated when the CNT value is MAX.

Figure 32-8. Input Capture Frequency and Pulse-Width Measurement



32.3.3.1.7 Single-Shot Mode

Use the Single-Shot mode to generate a pulse with a duration defined by the Capture/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 the Events section.

When the counter stops, the output pin is set low. If an event is detected on the connected event channel, the TCB peripheral will reset and start counting from BOTTOM to TOP while driving its output high. Read the Run (RUN) bit in the Status (TCBn.STATUS) register to see if the counter is counting. Once the value of CNT reaches the TCBn.CCMP register, the counter ceases counting. Simultaneously, the output pin transitions to a low state for at least one counter-clock cycle (TCB_CLK). During this period, any new event that occurs is disregarded. Following this, there is a two peripheral clock cycles (PER_CLK) delay before the output is set high after receiving a new event.

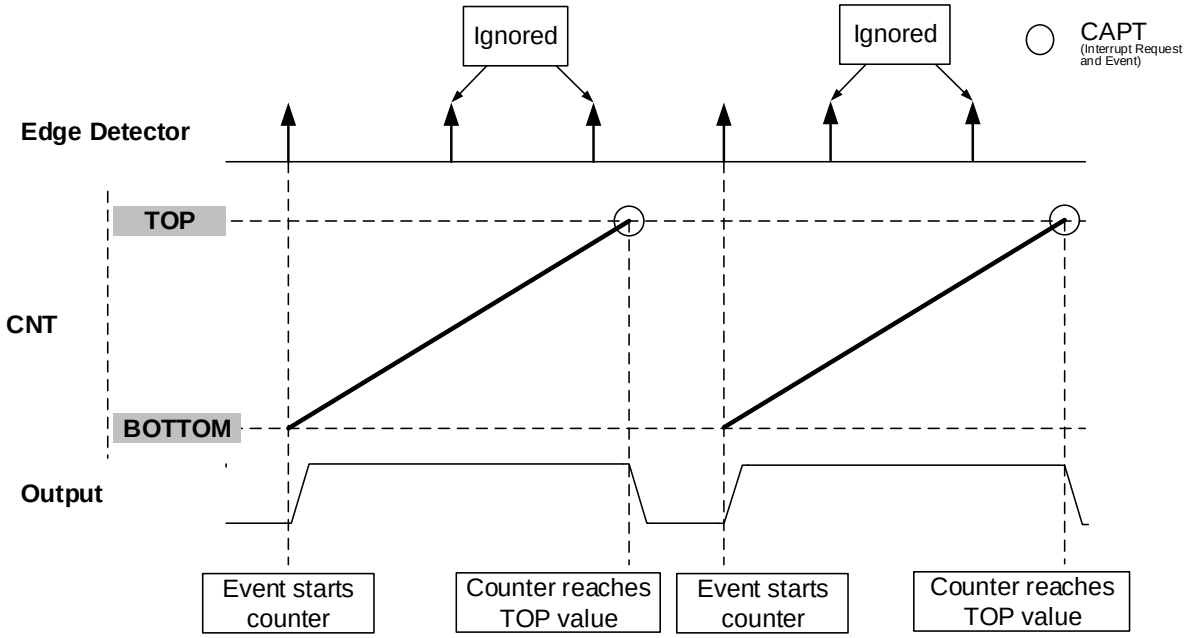
Writing a '1' to the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register triggers any edge to start the counter.

Writing a '0' to the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register triggers only positive edges to start the counter.

The counter starts 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, which is prevented by writing TOP to the Counter (TCBn.CNT) register. A 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 (TCBn.CNT) register prevents this.

Writing a '1' to the Asynchronous Enable (ASYNC) bit in the Control B (TCBn.CTRLB) register, causes the TCB peripheral to 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 complete clock cycles after receiving the event, resulting in an observed delay of two to three clock cycles.

Figure 32-9. Single-Shot Mode

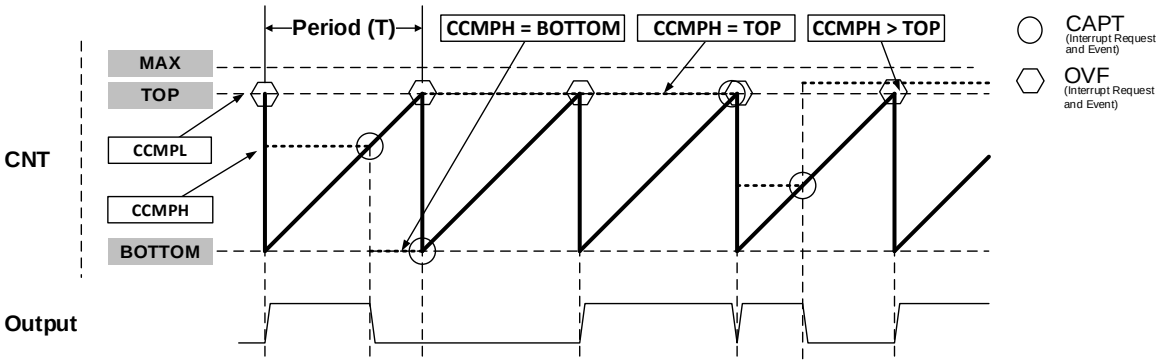


32.3.3.1.8 8-Bit PWM Mode

The TCB peripheral can be configured to run in 8-bit PWM mode, where each register pair in the 16-bit Capture/Compare (TCBn.CCMPH and TCBn.CCMPL) registers are used as individual Compare registers. CCMPL controls the period (T), while CCMPH controls the waveform duty cycle. 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 output pulse period, the +1 resulting in an observed delay of one clock cycle.

Figure 32-10. 8-Bit PWM Mode



32.3.3.2 Output

The TCB peripheral synchronization and output logic level depend on the selected Timer Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) register. In the Single-Shot mode, the peripheral can be configured so that the signal generation happens asynchronously to an incoming event by

writing a '1' to the Asynchronous Enable (ASYNC) bit in the Control B (TCBn.CTRLB) register. Then, the output signal is set immediately at the incoming event instead of being synchronized to the TCB clock. Due to the synchronization delay for the counter, the waveform output will be set high for three to four CLK_TCB cycles more than what is defined by the TOP value.

Writing a '1' to the Capture/Compare Output Enable (CCMPEN) bit in the Control B (TCBn.CTRLB) register enables and makes the waveform output available on the corresponding pin, overriding the value in the corresponding PORT output register.

The table below lists the different configurations and their impact on the output.

Table 32-2. Output Configuration

CCMPEN	CNTMODE	ASYNC	Output
1	Single-Shot mode	0	The output is high when the counter starts and low when the counter stops
		1	The output is high when the event arrives and low when the counter stops
	8-bit PWM mode	Not applicable	8-bit PWM mode
	Other modes	Not applicable	The output is static as configured in the Capture/Compare Pin Initial Value (CCMPINIT) bit in the Control B (TCBn.CTRLB) register.
0	Not applicable	Not applicable	No output

Changing modes while the peripheral is enabled is not recommended, as this can produce an unpredictable output. An interrupt flag may be set during the timer configuration. Clearing the TCB Interrupt Flags (TCBn.INTFLAGS) register is recommended after configuring this peripheral.

32.3.3.3 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.

The selected event source for capture will be delayed by one peripheral clock cycle. This compensates for the carry propagation delay that occurs when cascading two TCBs via the Event System.

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 MSB TCB (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 one of the Input Capture modes
- The Cascade Two Timer/Counters (CASCADE) bit in CTRLA must be '0'

Configuration of the MSB Counter

- Enable the 32-bit mode by writing the Cascade Two Timer/Counters (CASCADE) bit 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 same Input Capture mode as the LSB TCB

Capturing a 32-Bit Counter Value

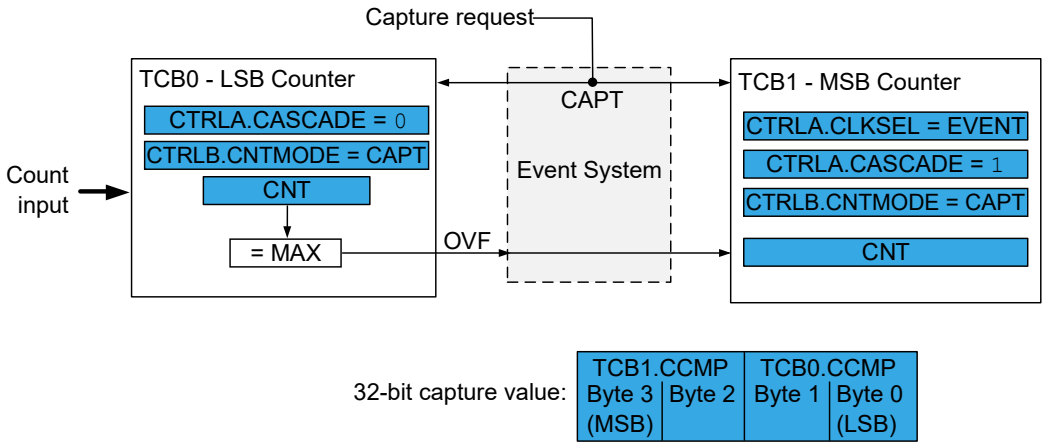
To acquire a 32-bit counter value, send a CAPT event to both TCBs. Both TCBs are running in the same Capture mode, so each will capture the current counter value (CNT) in the respective Capture/Compare (CCMP) register. The 32-bit capture value is formed by concatenating the two CCMP registers.

Example 32-1. 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. Both TCBs are in Input Capture on Event mode.

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 a correct timing of the CAPT event.

The captured 32-bit value is concatenated from TCB1.CCMP (MSB) and TCB0.CCMP (LSB).



32.3.3.4 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.

32.3.3.5 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 (CLKSEL) bit field in the Control A (TCBn.CTRLA) register. 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.

32.3.4 Events

The TCB can generate the events described in the following table:

Table 32-3. 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 *EVSYS - Event System* section for more details regarding event users and configuration.

The TCB can receive the events described in the following table:

Table 32-4. 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 must 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.

32.3.5 Interrupts

Table 32-5. Available Interrupt Vectors and Sources

Vector Name	Source Name	Condition	Dependency
TCBn_INT	CAPT	A capture condition occurs	The Timer Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) 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.

32.3.6 Sleep Mode Operation

The TCB peripheral is disabled by default when in the Standby sleep mode.

The peripheral can stay fully operational in the Standby sleep mode by writing a '1' to the Run Standby (RUNSTDBY) bit in the Control A (TCBn.CTRLA) register.

All operations are halted in the Power-Down sleep mode.

32.4 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 ... 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]							

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	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 enables the peripheral to run in Standby sleep mode.

Value	Description
0	The peripheral will not run in Standby
1	The peripheral will run in Standby

Bit 5 – CASCADE Cascade Two Timer/Counters
This bit controls whether the peripheral will be used for the Most Significant Bytes (MSBs) in cascading of two 16-bit Timer/Counters type B (TCBn) for 32-bit operation through the Event System.

Value	Description
0	The peripheral is used for 16-bit operation or used for the two Least Significant Bytes (LSBs) in 32-bit operation
1	The peripheral is used for the two MSBs in 32-bit operation

Bit 4 – SYNCUPD Synchronize Update
This bit controls whether the peripheral is synchronized with a TCA. Which TCA instance is controlled by CLKSEL, for all other clock selections than TCAn the peripheral will be synchronized with TCA0.

Value	Description
0	The peripheral is not synchronized
1	The peripheral will restart whenever TCAn is restarted or overflows. This can be used to synchronize capture with the PWM period

Bits 3:1 – CLKSEL[2:0] Clock Select
This bit field controls the clock source for this peripheral.

Value	Name	Description
0x0	DIV1	CLK_PER
0x1	DIV2	CLK_PER / 2
0x2	TCA0	CLK_TCA from TCA0
0x3-0x6	-	Reserved
0x7	EVENT	Positive edge on event input

Bit 0 – ENABLE Enable
Writing a '1' to this bit enables the Timer/Counter type B peripheral.

32.5.2 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 a '1' to this bit allows asynchronous updates of the TCB output signal in Single-Shot mode.

Value	Description
0	Pulse generation starts when event detected and after the peripheral clock is synchronized
1	Pulse generation starts immediately when event is detected

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 a '1' to this bit enables the waveform output. This will make the waveform output available on the corresponding pin regardless of the direction that is set on the pin, and overriding the value in the corresponding PORT output register.

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
This bit field controls 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

32.5.3 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 a '1' to this bit 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 Timer Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) register. "—" means an event or edge has no effect in this mode.

Timer 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 • On the following Negative: Input Capture • On the 2nd Positive: Stop counter, interrupt	
	1	• On the 1st Negative: Clear and restart counter • On the following Positive: Input Capture • On the 2nd Negative: Stop counter, interrupt	
Single-Shot mode	0	Start counter	—
	1	Start counter	Start counter
8-bit PWM mode	0	—	—
	1	—	—

Bit 0 – CAPTEI Capture Event Input Enable
Writing a '1' to this bit enables the input capture event.

32.5.4 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 a '1' to this bit enables interrupt on overflow.

Value	Description
0	The Overflow interrupt is disabled
1	The Overflow interrupt is enabled

Bit 0 – CAPT Capture Interrupt Enable
Writing a '1' to this bit enables interrupt on capture.

Value	Description
0	The Capture interrupt is disabled
1	The Capture interrupt is enabled

32.5.5 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 when the timer/counter wraps from MAX to BOTTOM.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

Bit 0 – CAPT Capture Interrupt Flag
This bit is set when a capture interrupt occurs. The interrupt conditions depend on the Timer Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) register.
This bit is cleared by writing a '1' or when the Capture register is read in Capture mode.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Capture interrupt flag.

Table 32-6. Interrupt Sources Set Conditions by Timer Mode

Timer 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

32.5.6 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

This bit is set to '1' when the counter is running. This bit is cleared to '0' when the counter is stopped.
This bit is cleared when the timer/counter stops.
This bit is set continuously when the timer/counter is running.
The bit is read-only.

32.5.7 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

32.5.8 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 *AVR CPU* 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

32.5.9 Counter

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] Counter Value High
This bit field holds the MSB of the 16-bit Counter register.

Bits 7:0 – CNT[7:0] Counter Value Low
This bit field holds the LSB of the 16-bit Counter register.

32.5.10 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 the Capture operation, this register contains the captured value of the counter at the time the capture occurs
- In Periodic Interrupt, Time-Out Check, 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
This bit field holds the MSB of the 16-bit compare, capture and top value.

Bits 7:0 – CCMP[7:0] Capture/Compare Value Low Byte
This bit field holds the LSB of the 16-bit compare, capture and top value.

33. TCD - 12-Bit Timer/Counter Type D

33.1 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 the hardware
- Half-Bridge and Full-Bridge Output Support

33.2 Overview

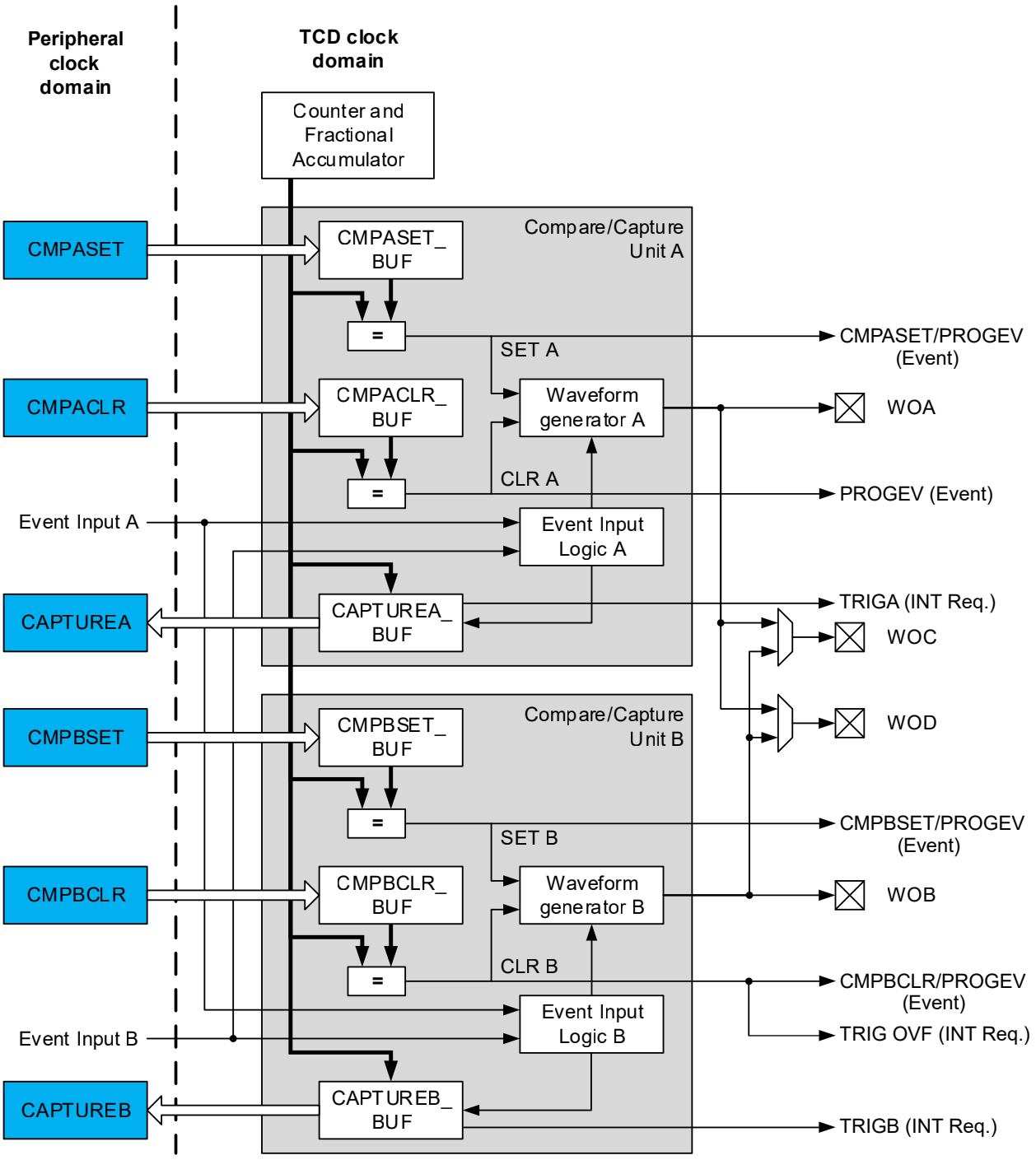
The Timer/Counter type D (TCD) is a high-performance waveform generator that consists of an asynchronous counter, a prescaler, and compare, capture and control logic.

The TCD contains a counter that can run on a clock that 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.

33.2.1 Block Diagram

Figure 33-1. TCD Block Diagram



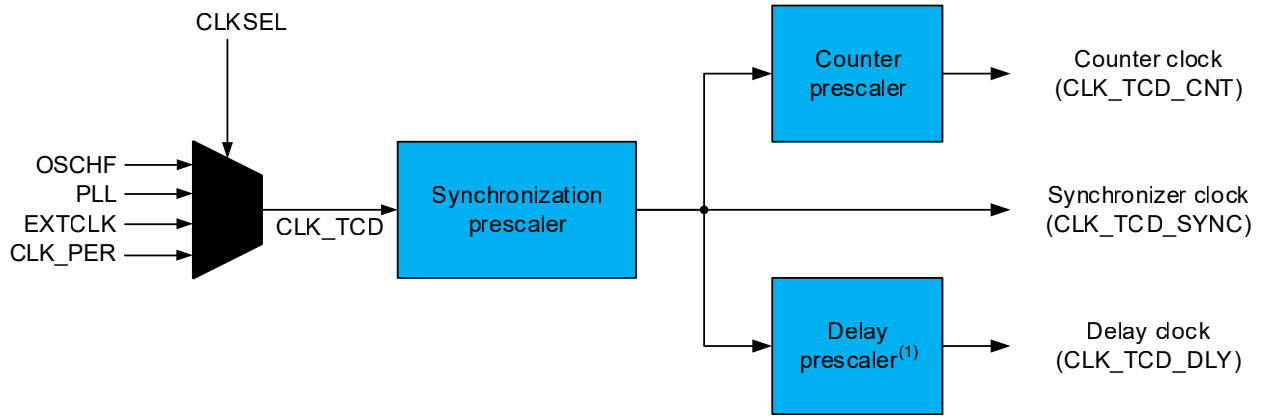
The TCD core is asynchronous to the peripheral clock. The timer/counter consists of two compare/capture units, each with a separate waveform output. There are also two extra waveform outputs, which can be equal to the output from one of the units. For each compare/capture unit, there is a pair of compare registers which is stored in the respective peripheral (TCDn.CMPASET, TCDn.CMPACLR, TCDn.CMPBSET, TCDn.CMPBCLR) registers.

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 33-2. 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:

$$SYNCPRESC\_division\_factor \times CNTPRESC\_division\_factor$$

The delay prescaler is used to prescale the clock utilized 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.

33.2.2 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

33.3 Functional Description

33.3.1 Definitions

The following definitions are used throughout the documentation:

Table 33-1. 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

33.3.2 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.

33.3.3 Operation

33.3.3.1 Register Synchronization Categories

Most of the I/O registers need to be synchronized to the TCD core clock domain, which is done differently for different register categories.

Table 33-2. 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 the TCDn.STATUS register is '1' to ensure that it is possible to issue a new command. The 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 the TCD is enabled, and no synchronization between the two clock domains is ongoing. Check that the CMDRDY bit in the TCDn.STATUS register 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 the TCD is enabled.

Table 33-3. Issuing Synchronization Command

Synchronization Issuing Bit	Double Register Update
CTRLC.AUPDATE	Every time the TCDn.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 the synchronization is already ongoing, the action has no effect.

Static Registers

Static registers cannot be updated while the TCD is enabled. Therefore, these registers must be configured before enabling the TCD. To see if the TCD is enabled, check if the ENABLE bit in the TCDn.CTRLA register 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.

33.3.3.2 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)

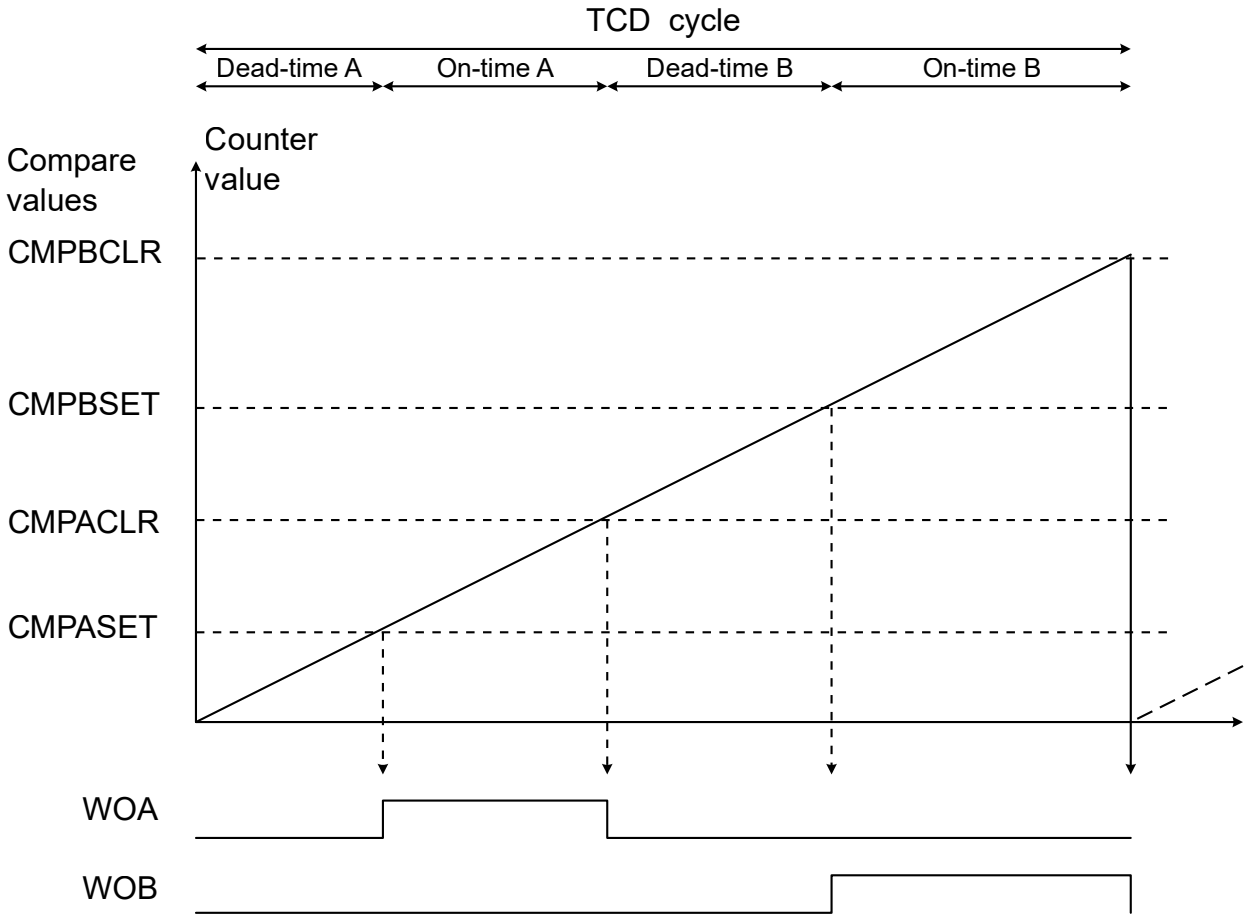
The Compare A Set (CMPASET), Compare A Clear (CMPACLR), Compare B Set (CMPBSET), and Compare B Clear (CMPBCLR) compare values define when each state ends and the next begins.

33.3.3.2.1 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:

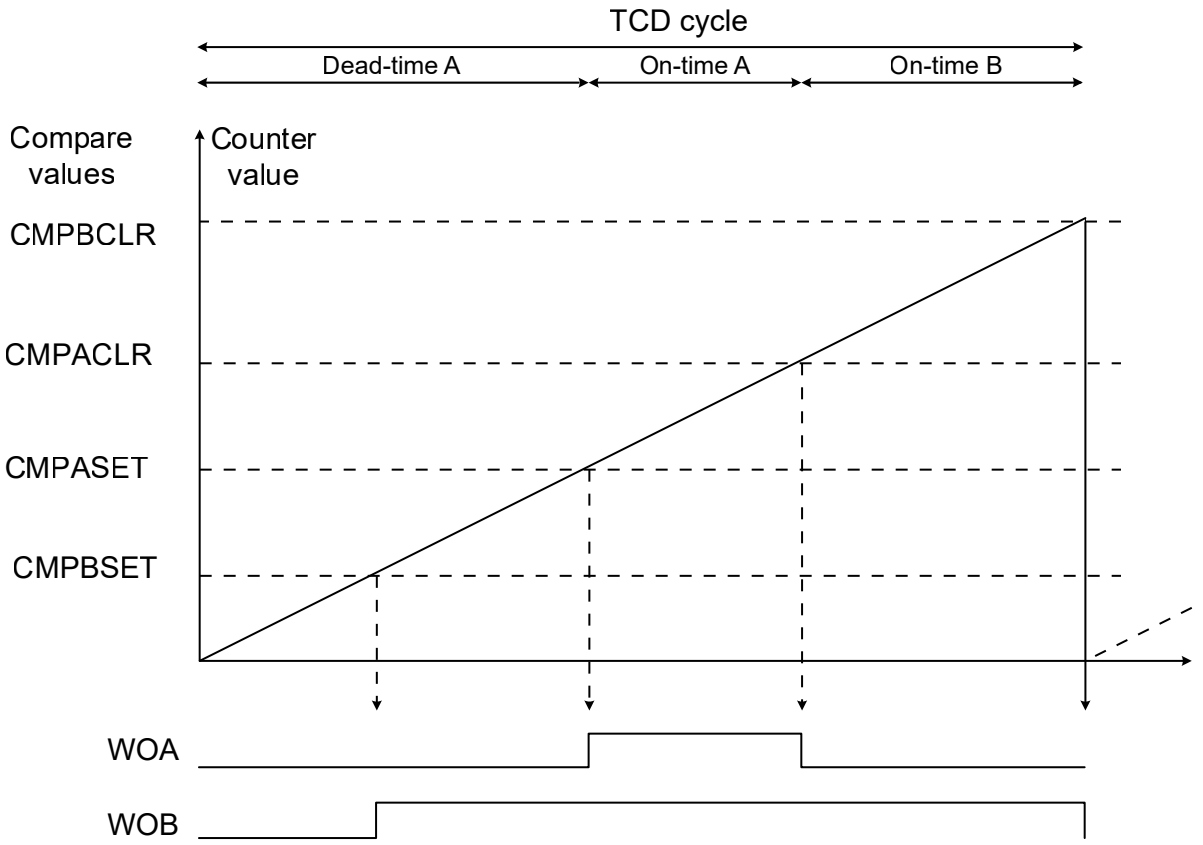
$$T_{\text{TCD\_cycle}} = \frac{(\text{CMPBCLR} + 1)}{f_{\text{CLK\_TCD\_CNT}}}$$

Figure 33-3. 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 33-4. One Ramp Mode with *CMPBSET < CMPASET*



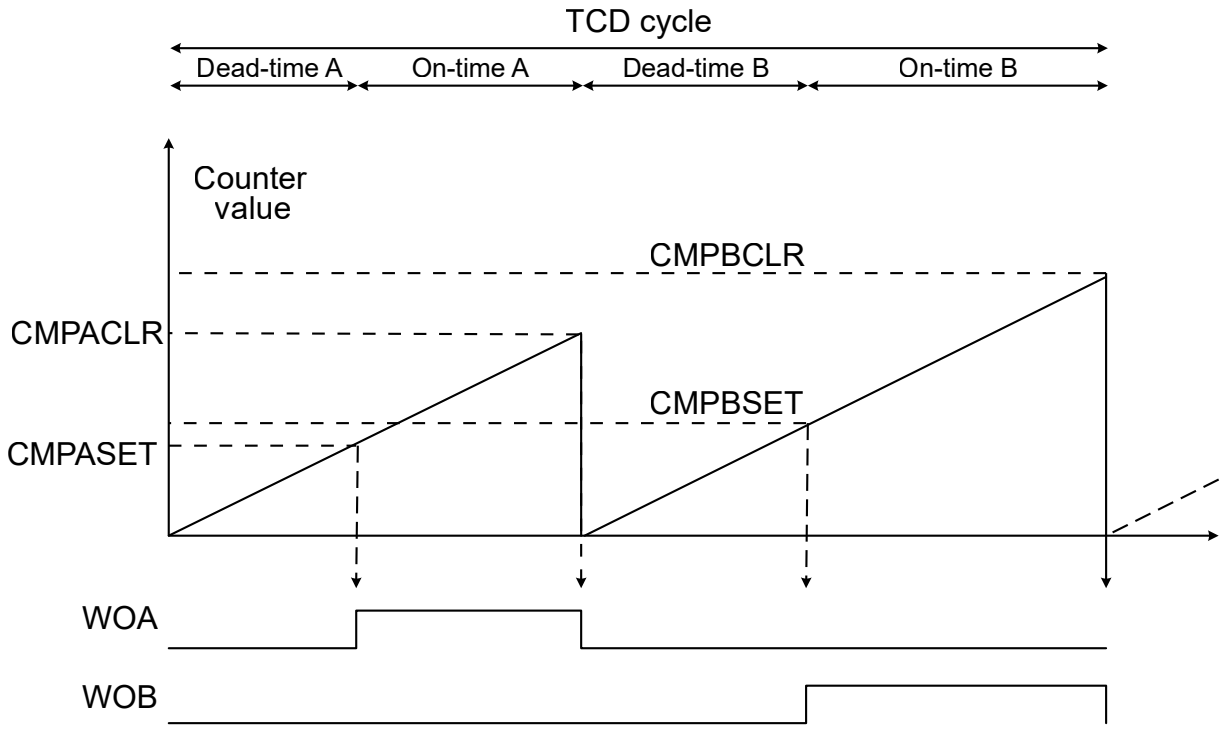
A match with CMPBCLR will always result in all outputs being cleared. If any of the other compare values are bigger than CMPBCLR, their associated effect will never occur. If the CMPACLR is smaller than the CMPASET value, the clear value will not have any effect.

33.3.3.2.2 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 33-5. 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.

33.3.3.2.3 Four Ramp Mode

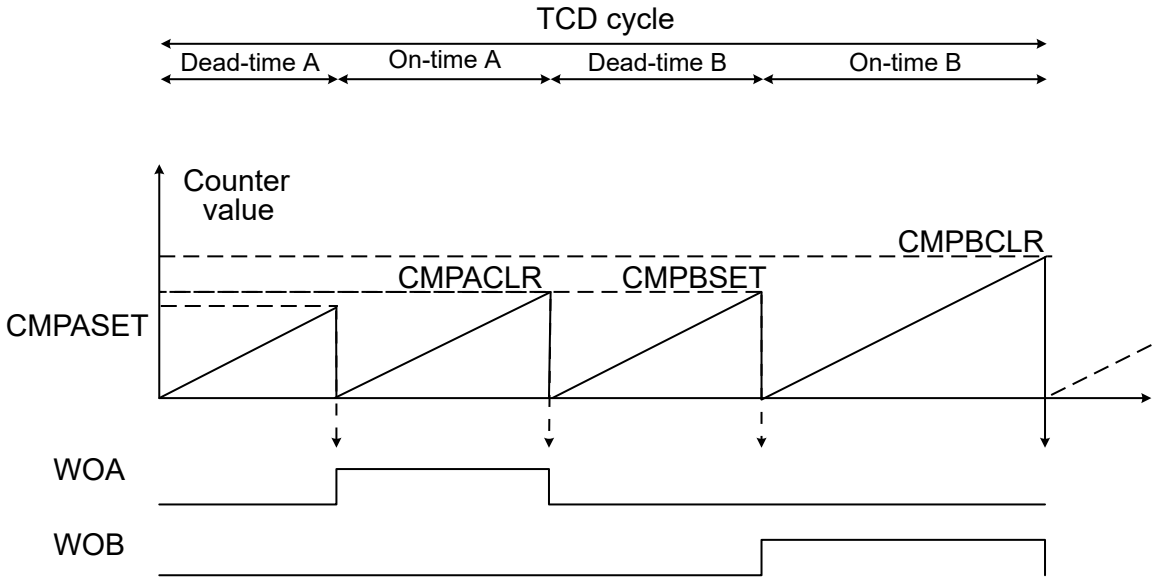
In Four Ramp mode, the TCD cycle follows this pattern:

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 it 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 33-6. 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.

33.3.3.2.4 Dual Slope Mode

In Dual Slope mode, a TCD cycle consists of the TCD counter counting down from the CMPBCLR value to zero and up again to the CMPBCLR value, which gives a TCD cycle period:

$$T_{\mathrm{TCD\_cycle}} = \frac{2 \times (\mathrm{CMPBCLR} + 1)}{f_{\mathrm{CLK\_TCD\_CNT}}}$$

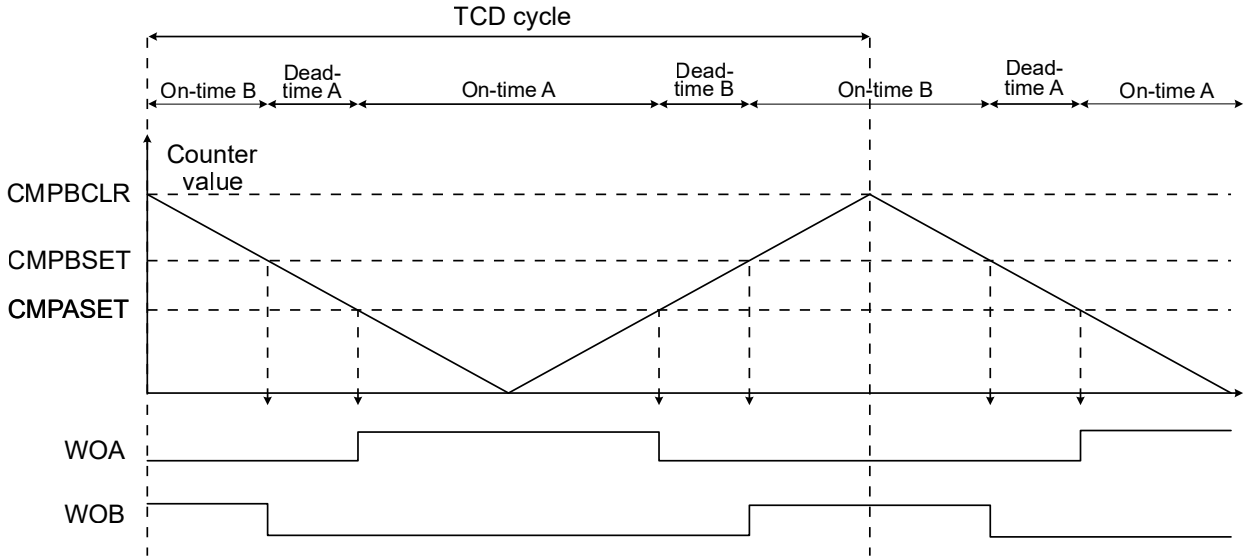
The WOA output is set when the TCD counter counts down and matches the CMPASET value. WOA is cleared when the TCD counter counts up and matches the CMPASET value.

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.

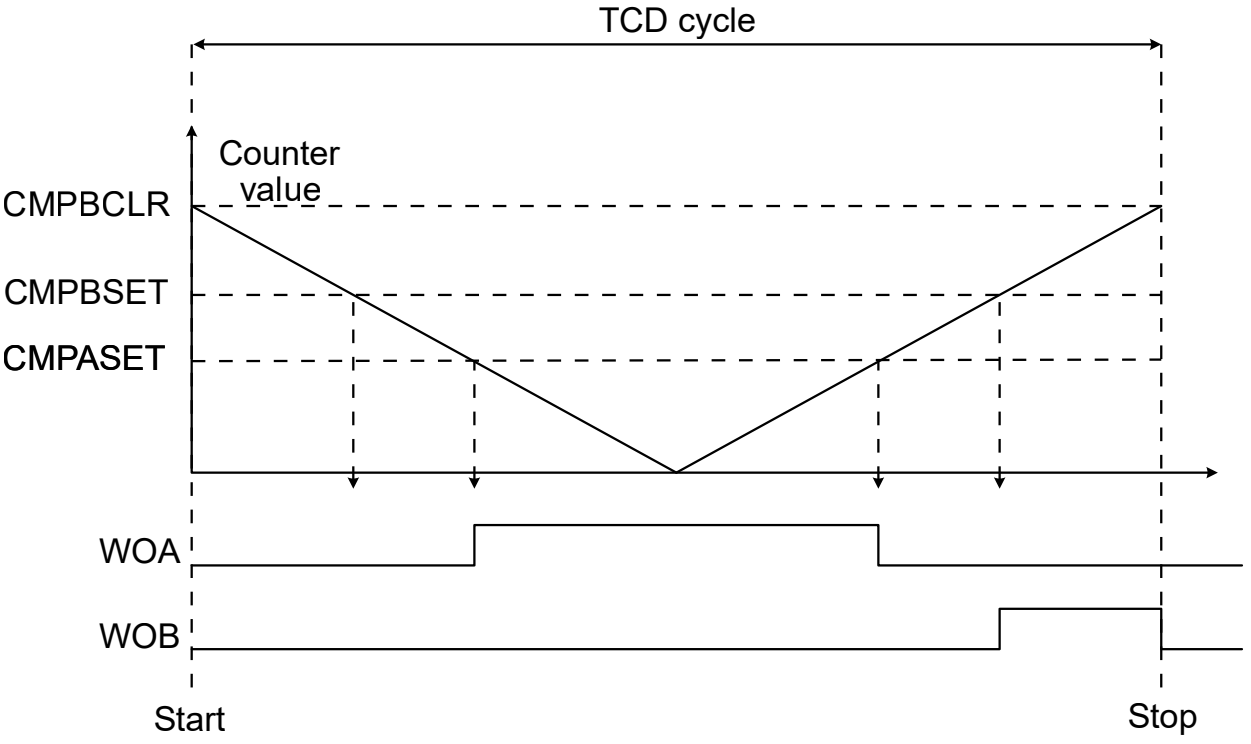
Figure 33-7. Dual Slope Mode



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 33-8. Dual Slope Mode Starting and Stopping



33.3.3.3 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.

33.3.3.4 TCD Inputs

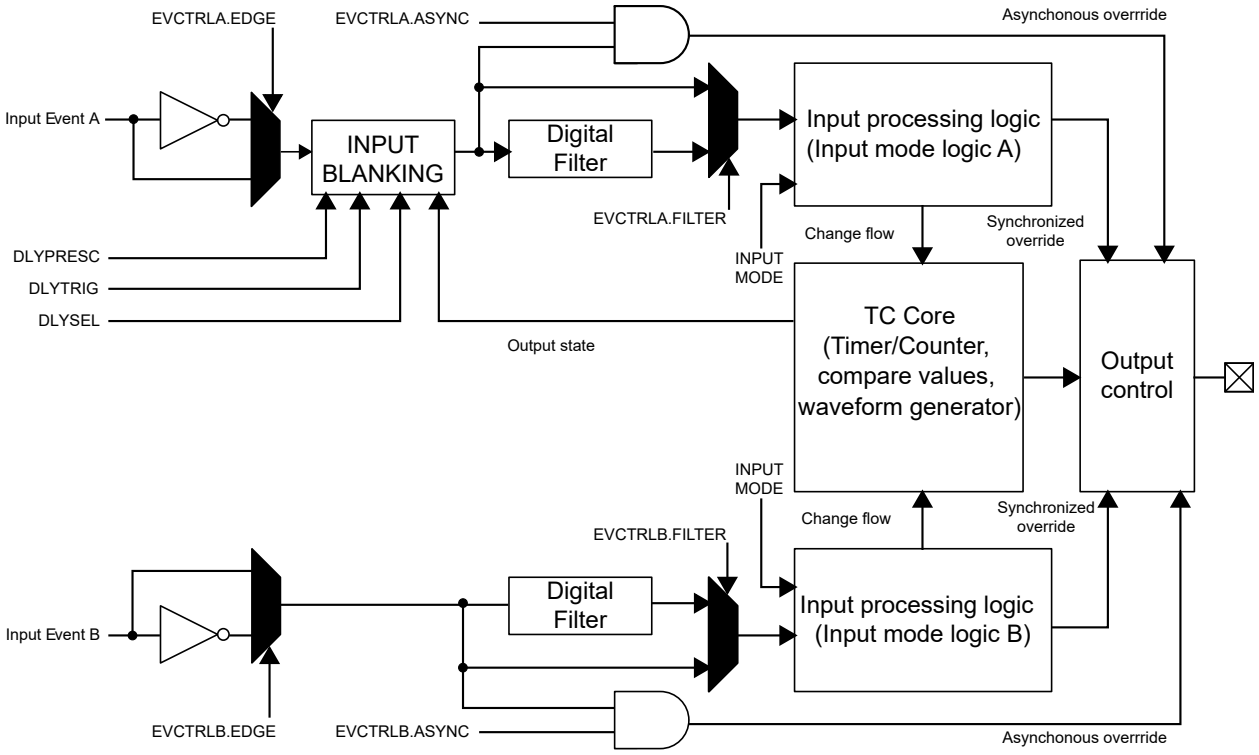
The TCD has two inputs connected to the Event System: Input A and input B. Each input has a functionality connected to the corresponding output (WOA and WOB). This functionality is controlled by the Event Control (TCDn.EVCTRLA and TCDn.EVCTRLB) registers and the Input Control (TCDn.INPUTCTRLA and TCDn.INPUTCTRLB) 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. The INPUTMODE bit field in the corresponding Input Control (TCDn.INPUTCTRLA or TCDn.INPUTCTRLB) register must be written to '0' to disable Fault detect.

There are ten different input modes for 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 33-9. 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.

33.3.3.4.1 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.

33.3.3.4.2 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. Therefore, any change on the incoming event will take four counter clock cycles before it affects the input processing logic.

33.3.3.4.3 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.

33.3.3.4.4 Software Commands

The following table displays the commands for the TCD module.

Table 33-4. 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

33.3.3.4.5 Input Modes

The user can select between ten input modes. The selection is made 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.

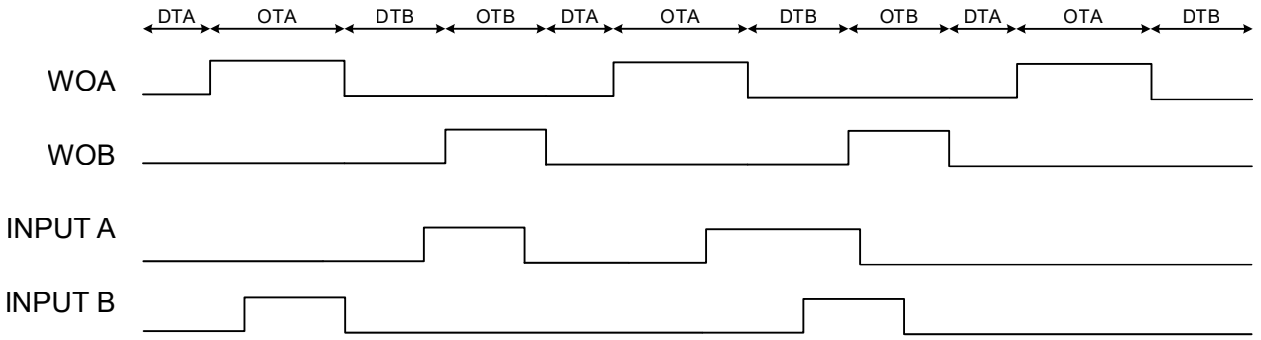
Table 33-5. Input Modes Validity

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 33-10. Input Mode 0

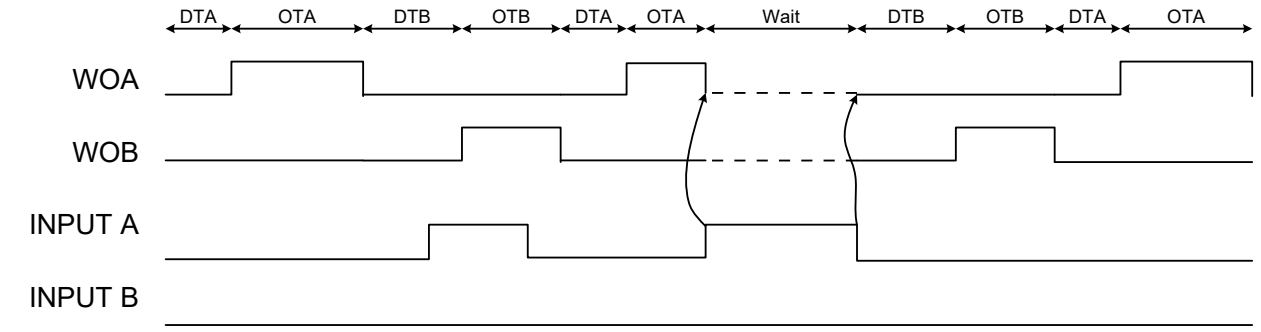


Input Mode 1: Stop Output, Jump to the Opposite Compare Cycle, and Wait

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.

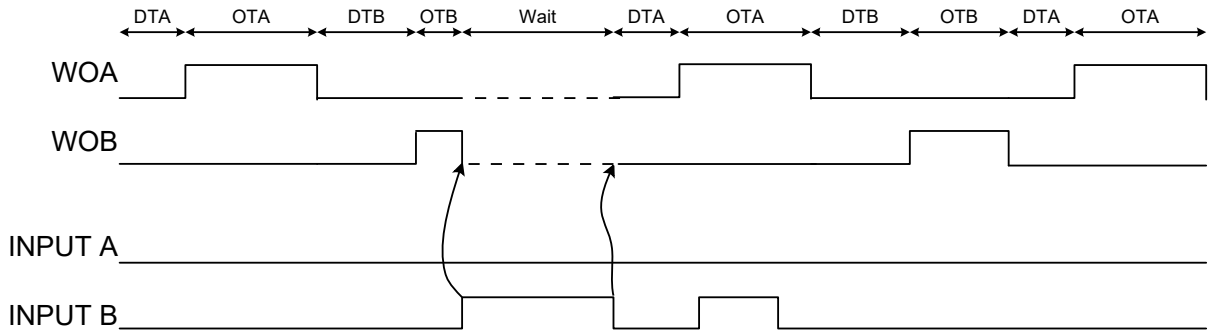
If Input mode 1 is used on input A, an event will only affect if the TCD is in dead-time A or on-time A, and it will affect only the WOA output. When the event is done, the TCD counter starts at dead-time B.

Figure 33-11. Input Mode 1 on Input A



If Input mode 1 is used on input B, an event will only affect if the TCD is in dead-time B or on-time B, and it will affect only the WOB output. When the event is done, the TCD counter starts at dead-time A.

Figure 33-12. Input Mode 1 on Input B

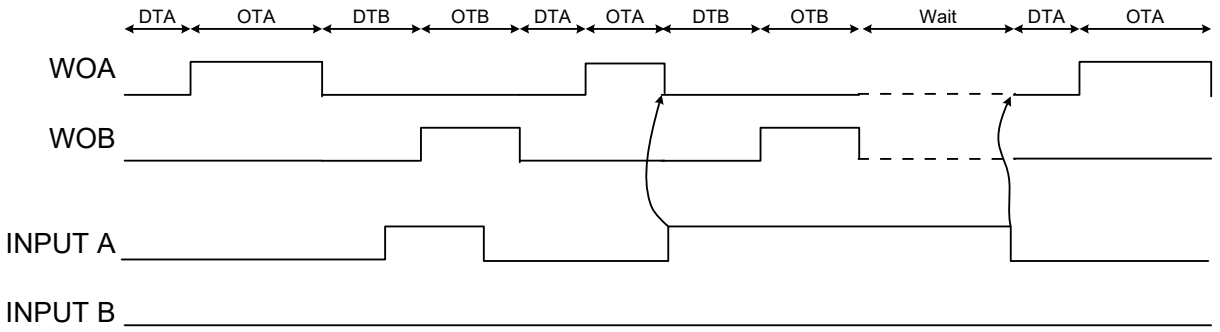


Input Mode 2: Stop Output, Execute Opposite Compare Cycle, and Wait

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.

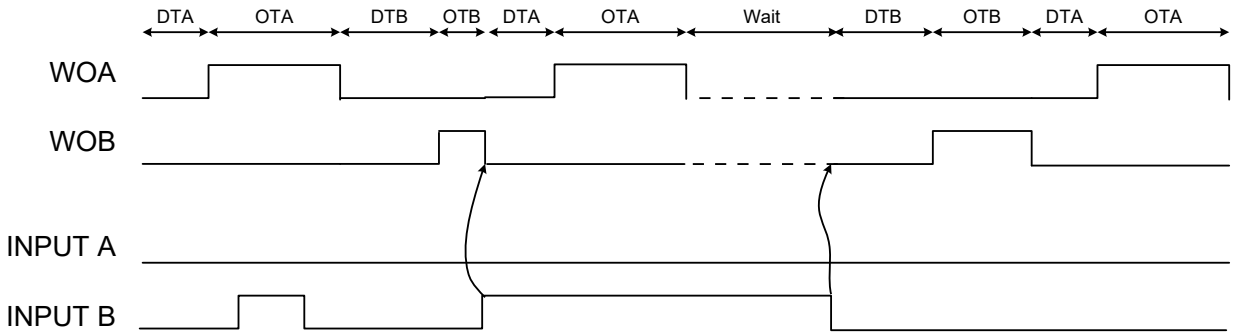
If Input mode 2 is used on input A, an event will only affect if the TCD is in dead-time A or on-time A, and will affect only the WOA output.

Figure 33-13. Input Mode 2 on Input A



If Input mode 2 is used on input B, an event will only affect if the TCD is in dead-time B or on-time B, and it will affect only the WOB output.

Figure 33-14. Input Mode 2 on Input B

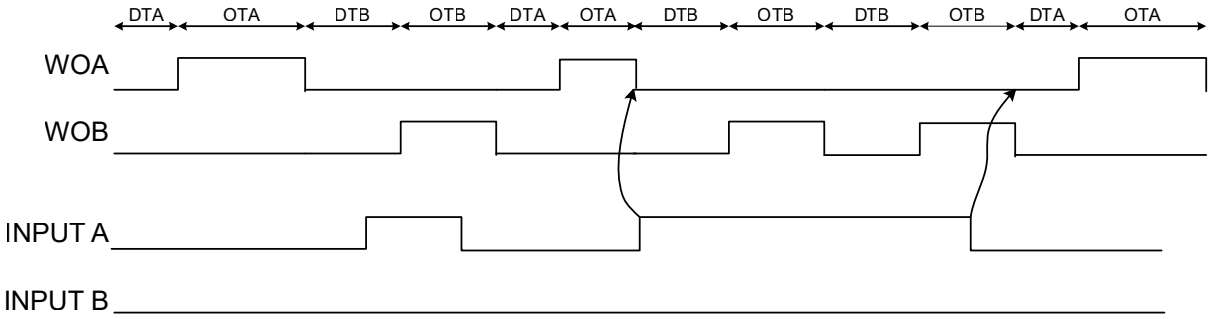


Input Mode 3: Stop Output, Execute Opposite Compare Cycle while Fault Active

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.

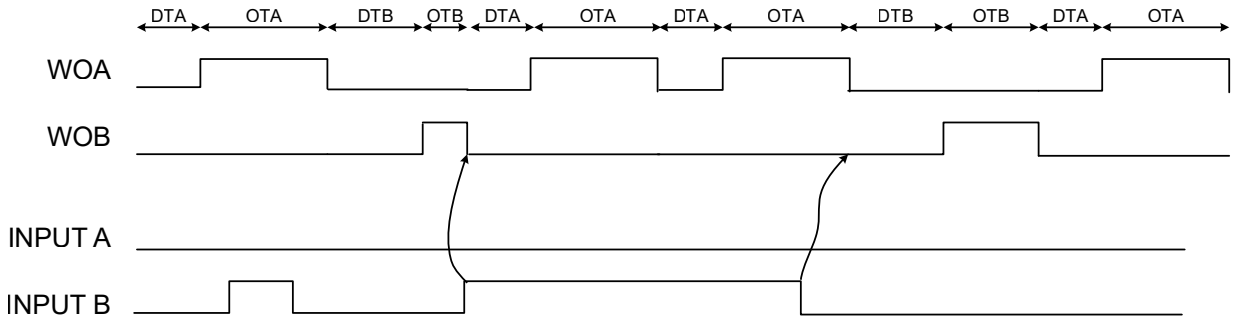
If Input mode 3 is used on input A, an event will only affect if the TCD is in dead-time A or on-time A.

Figure 33-15. Input Mode 3 on Input A



If Input mode 3 is used on input B, an event will only affect if the TCD is in dead-time B or on-time B.

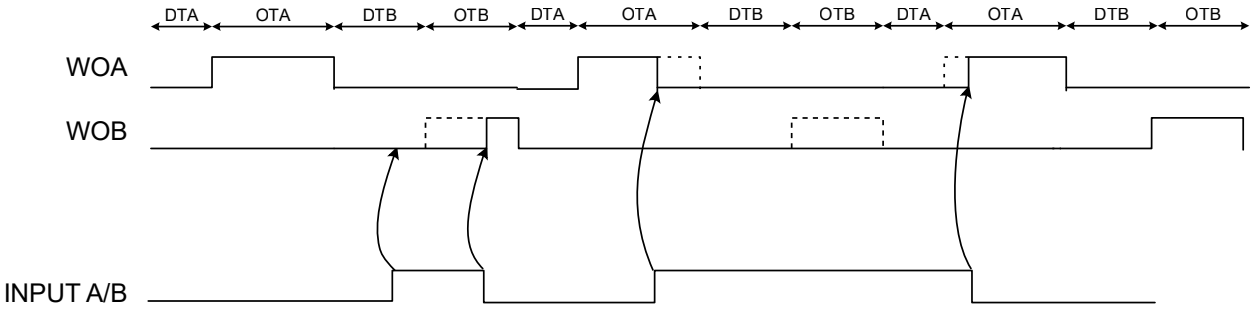
Figure 33-16. Input Mode 3 on Input B



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 33-17. Input Mode 4

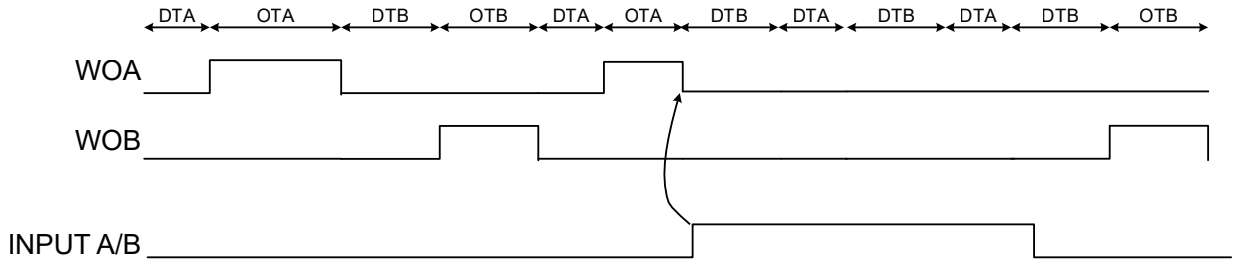


Input Mode 5: Stop all Outputs, Execute Dead-Time while Fault Active

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.

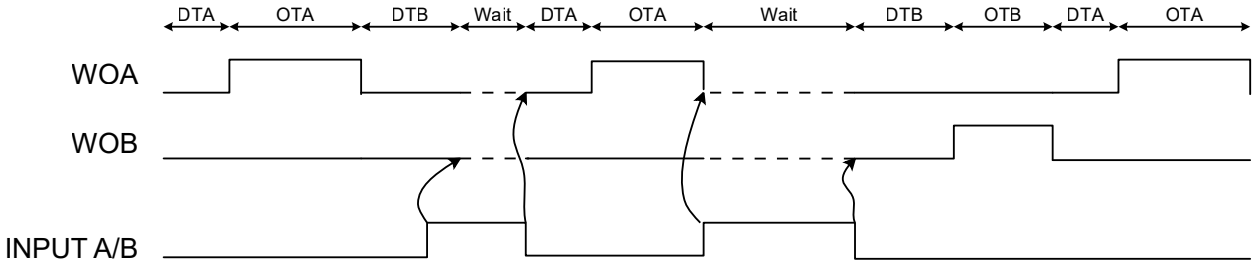
Figure 33-18. Input Mode 5



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.

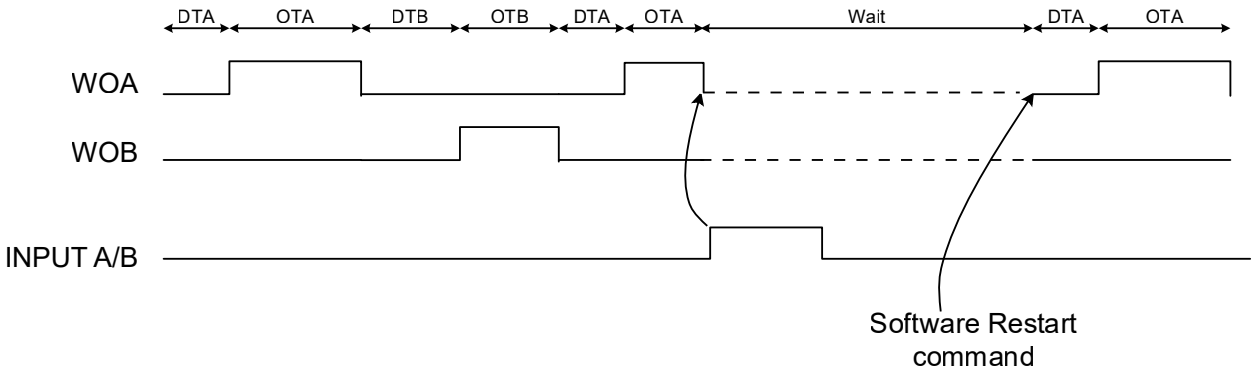
Figure 33-19. Input Mode 6



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 33-20. Input Mode 7

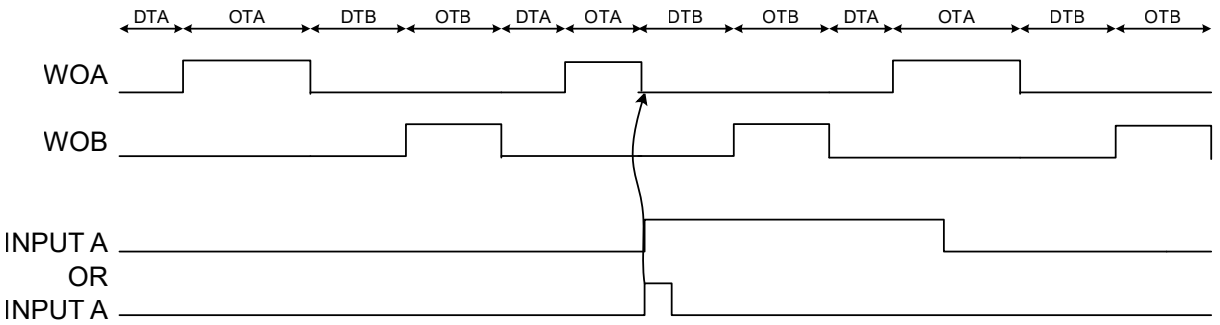


Input Mode 8: Stop Output on Edge, Jump to Next Compare Cycle

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.

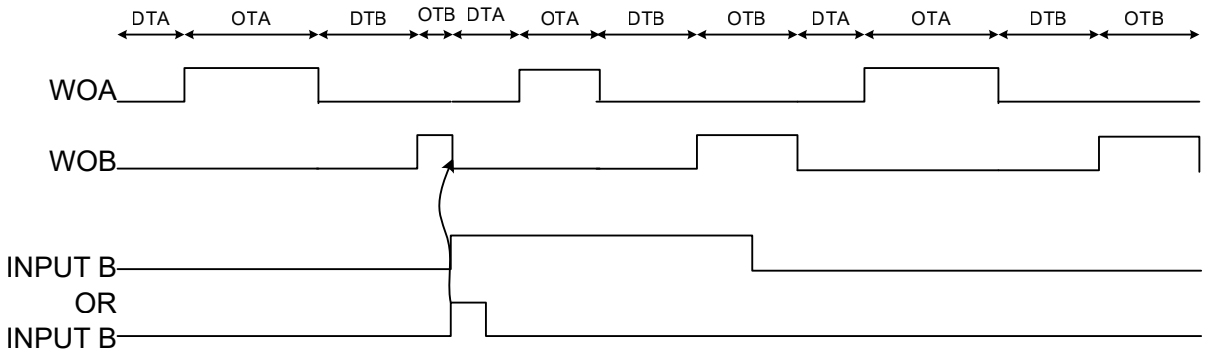
If Input mode 8 is used on input A and a positive edge on the input event occurs while in on-time A, the TCD counter jumps to dead-time B.

Figure 33-21. 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 33-22. Input Mode 8 on Input B

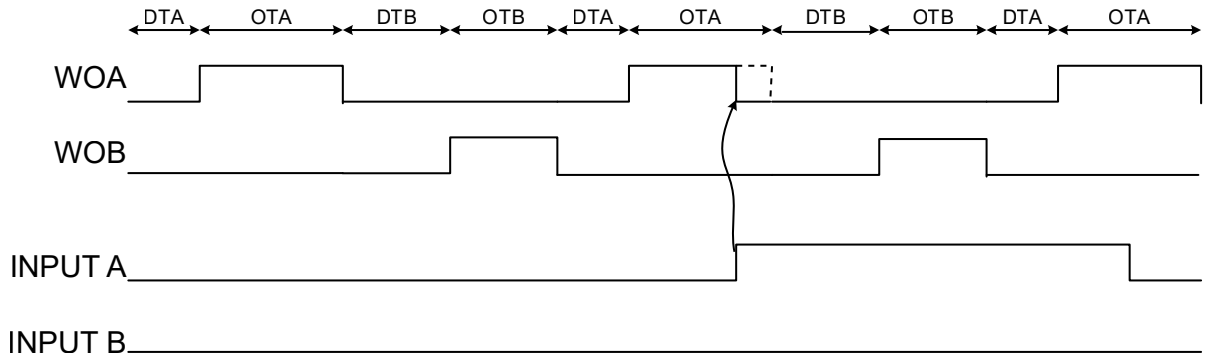


Input Mode 9: Stop Output on Edge, Maintain Frequency

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.

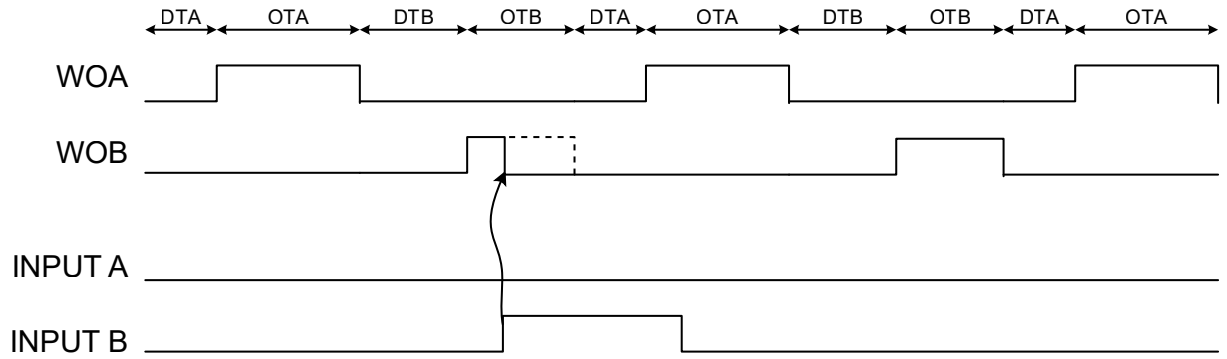
If Input mode 9 is used on input A and a positive edge on the input event occurs while in on-time A, the output will be OFF for the rest of the on-time.

Figure 33-23. 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 33-24. Input Mode 9 on Input B

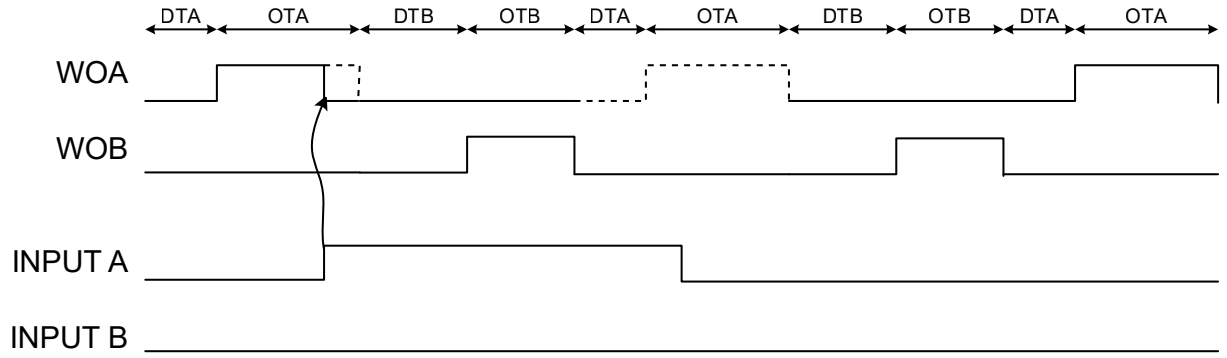


Input Mode 10: Stop Output at Level, Maintain Frequency

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.

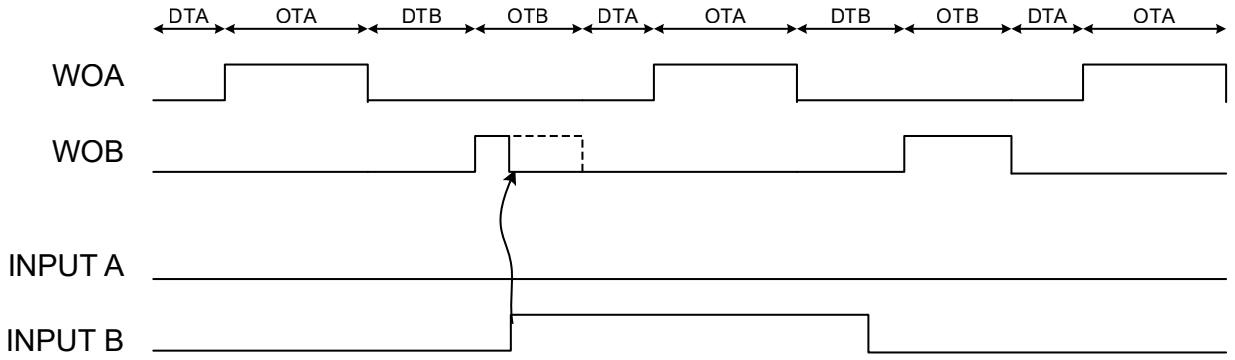
If Input mode 10 is used on input A and an input event occurs, the WOA will be OFF as long as the event lasts. If released during an on-time, it will be OFF for the rest of the on-time.

Figure 33-25. 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 33-26. Input Mode 10 on Input B



Input Mode Summary

Table 33-6 summarizes the conditions, as illustrated in the timing diagrams of the preceding sections.

Table 33-6. 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}		
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, consider possible conflicts, keeping in mind that the TCD has a single counter, to avoid unexpected results.

33.3.3.5 Dithering

If it is impossible 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.

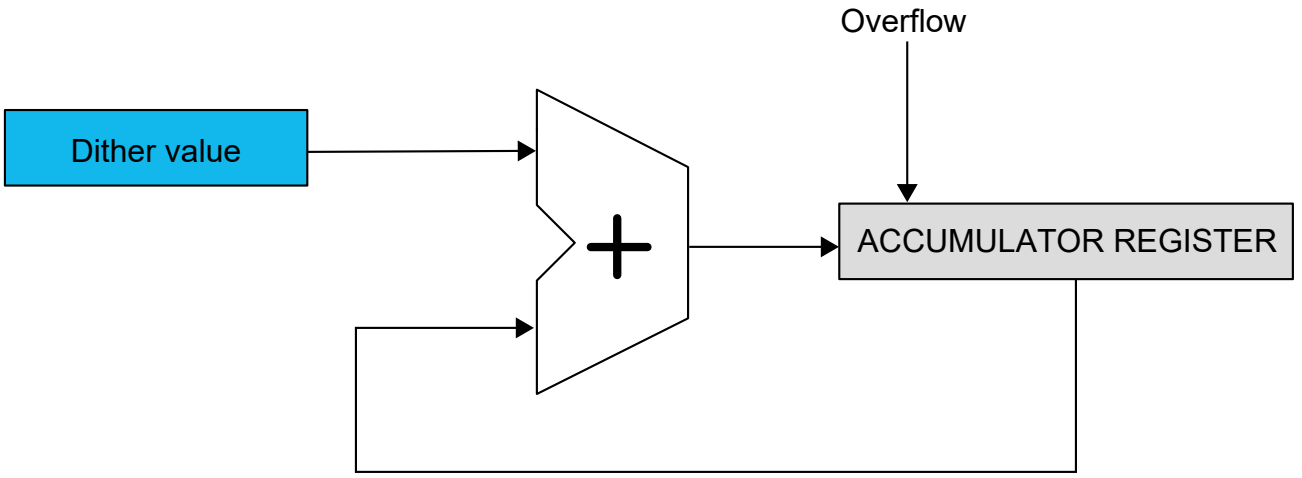
The dither accumulates the fractional error of the counter clock for each cycle. When the fractional error overflows, an additional clock cycle is added to the selected part of the TCD cycle.

Example 33-1. 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 13,333 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 33-27. Dither Logic



The user can select where in the TCD cycle the dither will be added by writing to the Dither Selection (DITHERSEL) bit field 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 33-7). Dithering is not supported in Dual Slope mode.

Table 33-7. 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 33-2. 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.

33.3.3.6 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 the TCDn.CAPTUREAL/TCDn.CAPTUREBL registers and then the TCDn.CAPTUREAH/TCDn.CAPTUREBH registers.

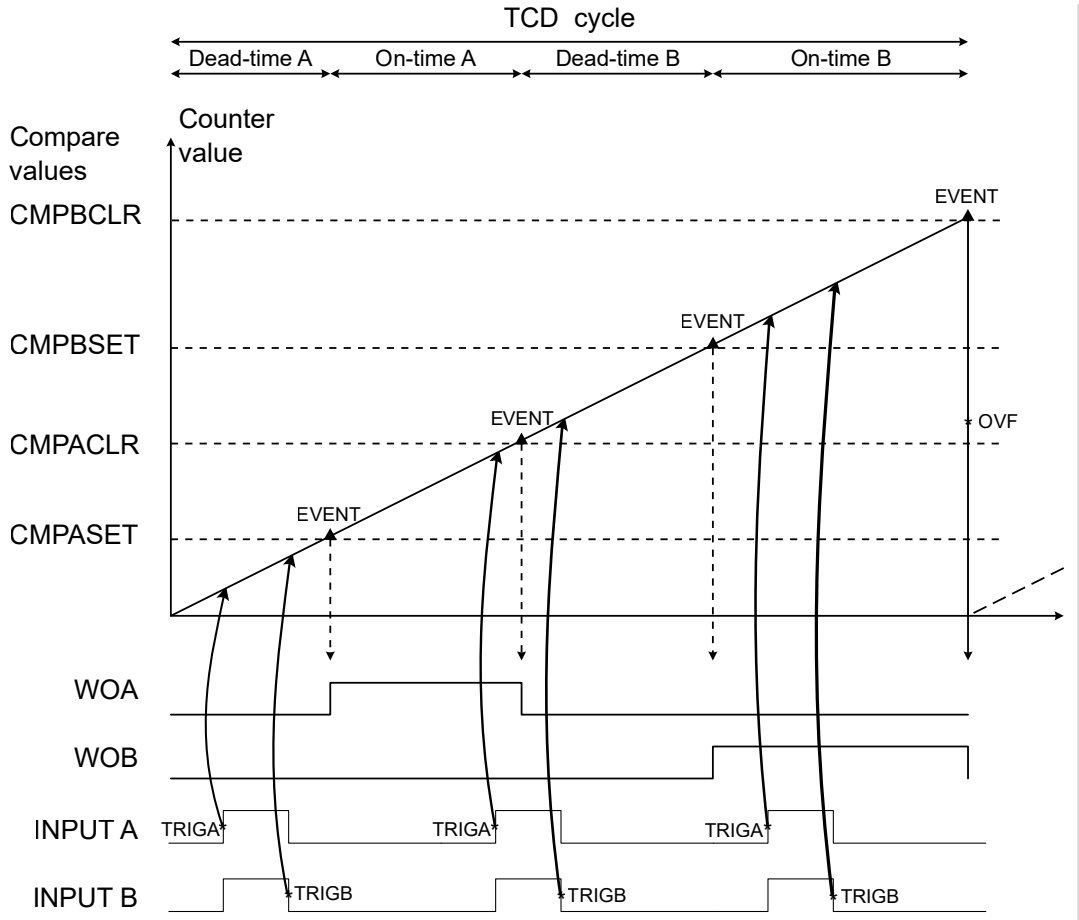
Captures Triggered by Input Events

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 33-3. 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

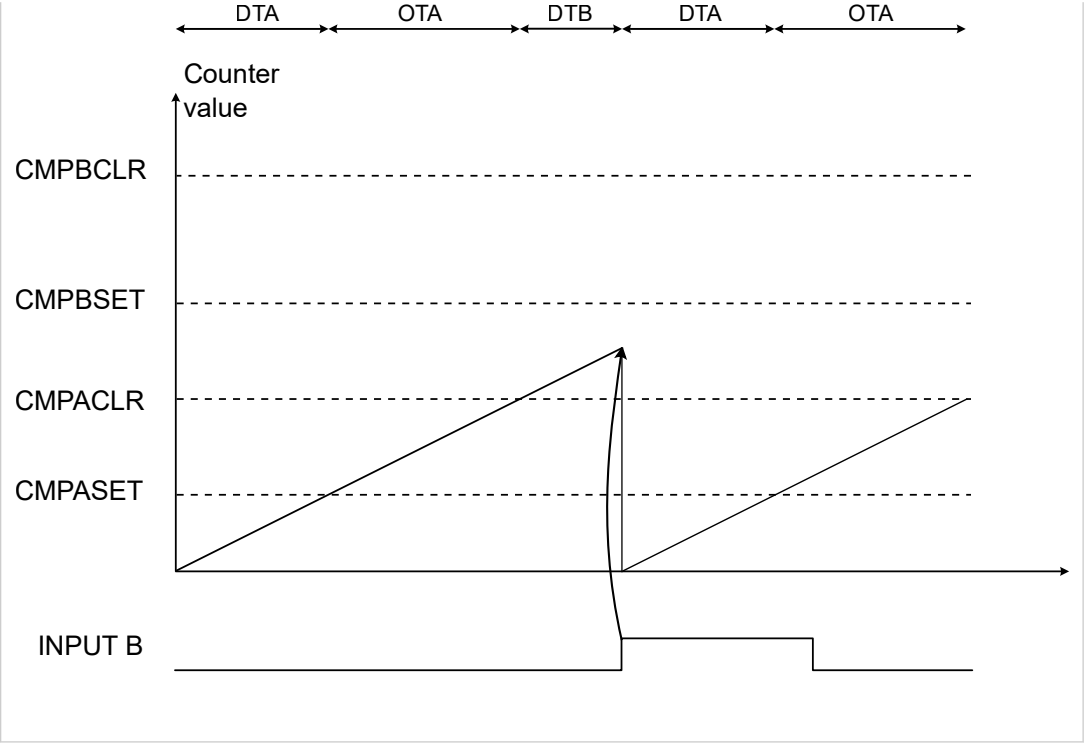
The software can capture the TCD value by writing a '1' to the respective Software Capture A/B Strobe (SCAPTUREx) bit in the Control E (TCDn.CTRLE) register. When this command is executed, and the Command Ready (CMDRDY) bit in the Status (TCDn.STATUS) register reads '1' again, the CAPTUREA/B value is available. It can now be read by reading first the TCDn.CAPTUREAL or TCDn.CAPTUREBL register and then the TCDn.CAPTUREAH or the TCDn.CAPTUREBH register.

Using Capture Together with Input Modes

The capture functionality can be used together with the 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 33-4. 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.



33.3.3.7 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 Fault 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.

33.3.4 Events

The TCD can generate the events described in the following table:

Table 33-8. 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 33-9. 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 TCD Inputs section. Refer to the *EVSYS - Event System* section for more details regarding event types and Event System configuration.

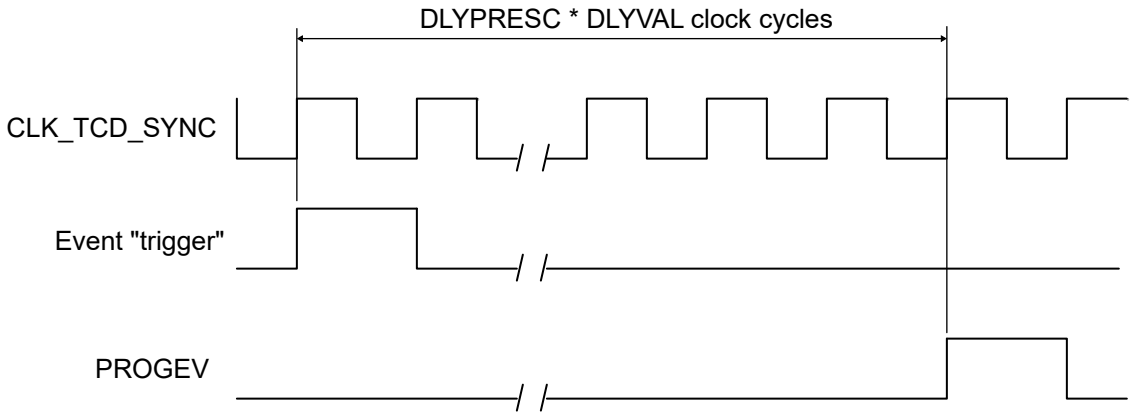
33.3.4.1 Programmable Output Events

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 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 is delayed by $n = \text{DLYPRESC} \times \text{DLYVAL}$ CLK_TCD_SYNC clock cycles, which lead to a delay time of:

$$t_{DELAY} = \frac{\text{DLYPRESC} \times \text{DLYVAL}}{f_{CLK\_TCD\_SYNC}}$$

Figure 33-28. Programmable Output Event Timing



33.3.5 Interrupts

Table 33-10. 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 • 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.

33.3.6 Sleep Mode Operation

The TCD operates in Idle sleep mode and is stopped when entering Standby and Power-Down sleep modes.

33.3.7 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.

33.3.8 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a given 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 it unchanged.

The following registers are under CCP:

Table 33-11. Registers Under Configuration Change Protection in TCD

Register	Key
TCDn.FAULTCTRL	IOREG

33.4 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	CMPDEN	CMPCEN	CMPBEN	CMPAEN	CMPD	CMPC	CMPB	CMPA
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]			

33.5 Register Description

33.5.1 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 bit field selects the clock source of the TCD clock.

Value	Name	Description
0x0	OSCHF	Internal high-frequency oscillator
0x1	PLL	PLL
0x2	EXTCLK	External clock
0x3	CLKPER	Peripheral clock, i.e., main clock after pre-scaler (CLK_PER)

Bits 4:3 – CNTPRES[1:0] Counter Prescaler
The Counter Prescaler bit field selects 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 bit field selects 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

33.5.2 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
This bit field selects 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

33.5.3 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
A write to either TCDn.CMPASET or TCDn.CMPBSET will be written to both registers if FIFTY = '1' The same is the case for TCDn.CMPACLR and TCDn.CMPBCLR.

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 (FIFTY = '1'), writing to either the Compare A Clear High or the Compare B Clear High register will request a synchronization at the end of the TCD cycle.

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 the Active state bit fields in the Control D register. See the CTRLD register description for more details.

33.5.4 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)
This bit field sets 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 33-12. 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 33-13. One Ramp Mode

CMPxVAL	DTA	OTA	DTB	OTB
PWMA	CMPAVAL[1]	CMPAVAL[0]	-	-
PWMB	-	-	CMPBVAL[3]	CMPBVAL[2]

33.5.5 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 only affects 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 only affects 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 only affects 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 only affects 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 only affects 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 only affects if there is no ongoing synchronization of a command. See also the CMDRDY bit in TCDn.STATUS.

33.5.6 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. Therefore, the input capture is 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 input triggers a Fault
1	CAPTURE	Event input triggers a Fault and a capture

Bit 0 – TRIGEI Trigger Event Input Enable
Writing this bit to '1' enables the event as the trigger for input A.

33.5.7 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. Therefore, the input capture is 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 input triggers a Fault
1	CAPTURE	Event input triggers a Fault and a capture

Bit 0 – TRIGEI Trigger Event Input Enable
Writing this bit to '1' enables the event as the trigger for input B.

33.5.8 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.

33.5.9 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.

33.5.10 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'.

33.5.11 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

33.5.12 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

33.5.13 Fault Control

Name: FAULTCTRL
Offset: 0x12
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	CMPDEN	CMPCEN	CMPBEN	CMPAEN	CMPD	CMPC	CMPB	CMPA
Access	R/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 – CMPEN Compare Enable
These bits enable the waveform output from the Compare (CMPx) bit field on pin x.

Bits 0, 1, 2, 3 – CMP Compare Value
These bits define the default state of the Fault state. When a fault occurs and the respective Compare Enable (CMPENx) bit is enabled, each waveform output pin WOx will assume the same state as CMPx.

33.5.14 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
This bit field controls 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
This bit field controls 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
This bit field selects which function is controlled by the delay trigger, delay prescaler, or delay value.

Value	Name	Description
0x0	OFF	Delay functionality not used
0x1	INBLANK	Input blanking
0x2	EVENT	Programmable output event
0x3	-	Reserved

33.5.15 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
This bit field configures the blanking/output event delay time or event output synchronization delay in several prescaled TCD cycles.

33.5.16 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 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

33.5.17 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
This bit field configures the fractional adjustment of the on-time or off-time, according to the Dither Selection (DITHERSEL) bit field 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 bit field is double-buffered, so the new value is copied when an update condition occurs.

33.5.18 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 the TCD is stopped in Debug mode
1	FAULT	A Fault is generated, and both trigger flags are set if the 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

33.5.19 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 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

33.5.20 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 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

33.5.21 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
This bit field holds the value of the compare register.

33.5.22 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
This bit field holds the value of the compare register.

33.5.23 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
This bit field holds the value of the compare register.

33.5.24 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
This bit field holds the value of the compare register.

34. RTC - Real-Time Counter

34.1 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

34.2 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 value 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 kHz Internal Oscillator (OSC32K), or the OSC32K divided by 32.

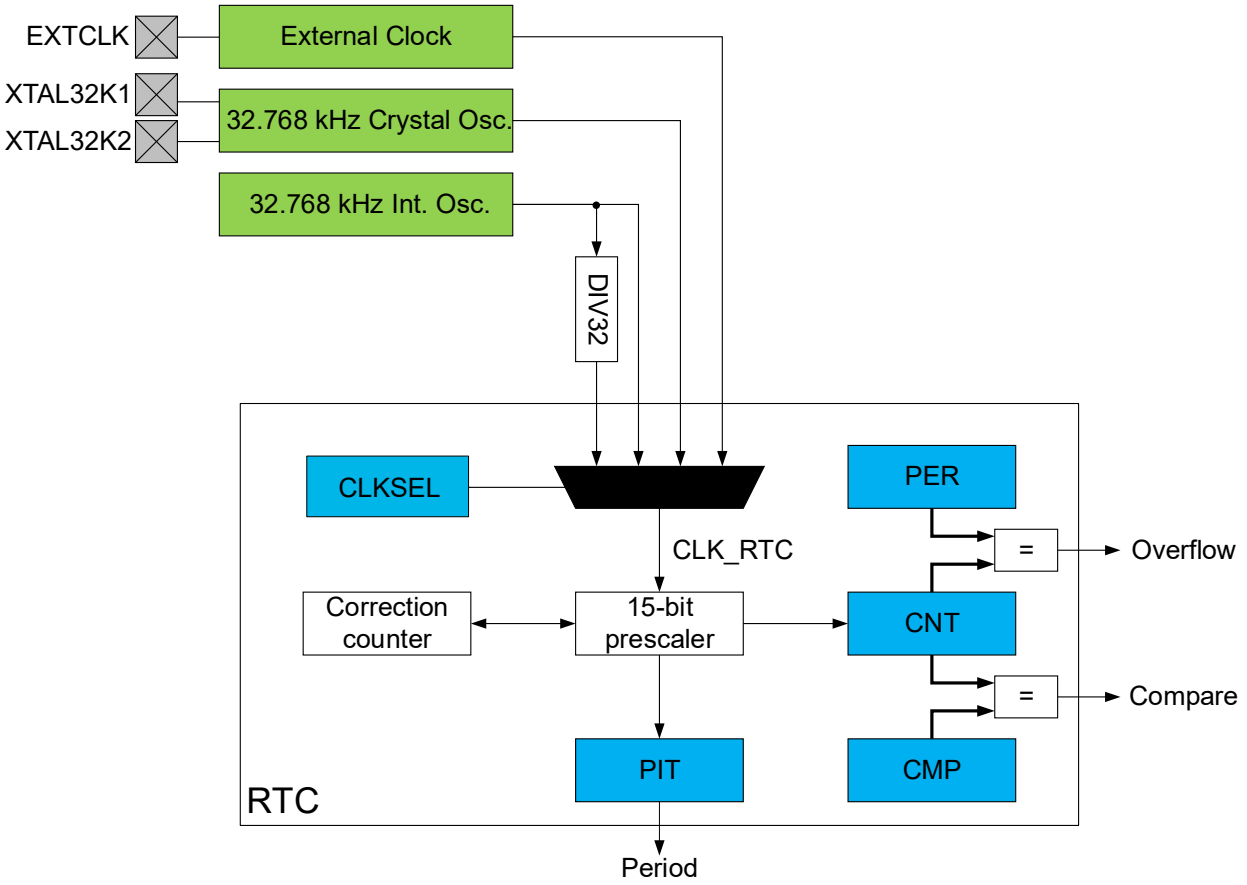
The RTC peripheral includes a 15-bit programmable prescaler that can scale down the reference clock before it reaches the counter. A wide range of resolutions and time-out periods can be configured for the RTC. With a 32.768 kHz clock source, the maximum resolution is 30.5 µs, and time-out periods can be up to two seconds. With a resolution of 1s, the maximum time-out period is more than 18 hours (65536 seconds).

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.

34.2.1 RTC Block Diagram

Figure 34-1. RTC Block Diagram



34.3 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.

34.4 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.

34.4.1 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.

34.4.1.1 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 functionalities.

34.4.1.2 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.

34.4.2 Operation - RTC

34.4.2.1 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'.

34.5 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.

34.5.1 Initialization

To operate the PIT, follow these steps:

1. Configure the RTC clock CLK_RTC as described in section 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. Enable the PIT by writing a '1' to the Periodic Interrupt Timer Enable (PITEN) bit in the RTC.PITCTRLA register.
4. 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.

34.5.2 Operation - PIT

34.5.2.1 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'.

34.5.2.2 PIT Interrupt Timing

Timing of the First Interrupt

Both PIT and RTC functions 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

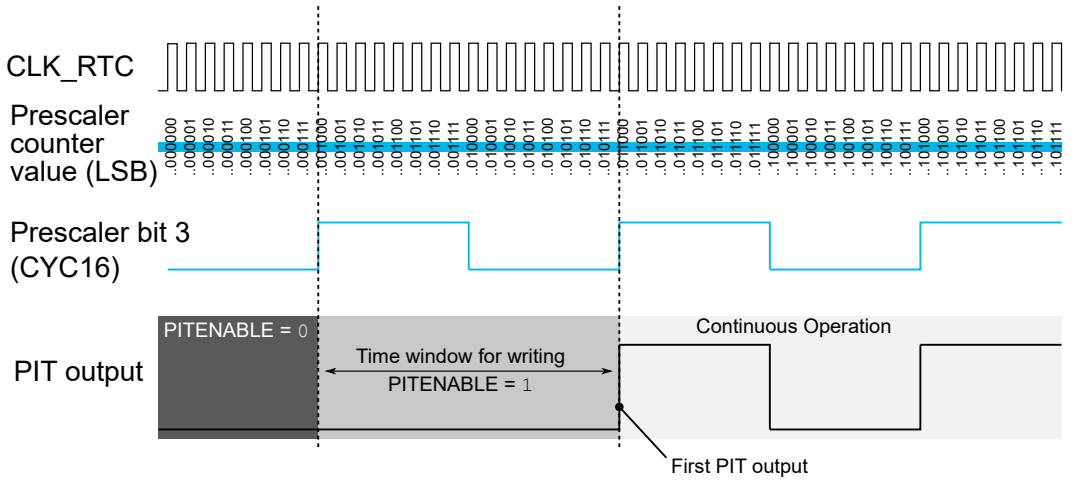
After the first interrupt, the PIT will continue toggling every ½ PIT period resulting in a full PIT period signal.

Example 34-1. 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 34-2. Timing Between PIT Enable and First Interrupt



34.6 Events

The RTC can generate the events described in the following table:

Table 34-1. RTC Event Generators

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
	EVGEN0	Selectable prescaled RTC event	Level		Prescaled CLK_RTC period
	EVGEN1				

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.

34.7 Interrupts

Table 34-2. 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 • 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.

34.8 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.

34.9 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 (CMPBUSY, PERBUSY, CNTBUSY, CTRLABUSY) flag is available in the Status (RTC.STATUS) register.

For the RTC.PITCTRLA register, a Synchronization Busy (CTRLBUSY) flag is available in the Periodic Interrupt Timer Status (RTC.PITSTATUS) register.

Check these flags before writing to the mentioned registers.

34.10 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 does not happen during normal operation. If the PIT output is low at the break, the PIT will resume low without additional interrupt.

34.11 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY	PRESCALER[3:0]						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	Reserved									
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
0x16	PITEVGENCTRLA	7:0	EVGEN1SEL[3:0]				EVGEN0SEL[3:0]			

34.12 Register Description

34.12.1 Control A

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

Application software needs to check that the CTRLABUSY flag in the RTC.STATUS register is cleared before writing to this register.

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	PRESCALER[3:0]						RTCEN
Access	R/W	R/W	R/W	R/W	R/W			R/W
Reset	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.

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 0 – RTCEN RTC Peripheral Enable

Value	Description
0	RTC peripheral is disabled
1	RTC peripheral is enabled

34.12.2 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.

34.12.3 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

34.12.4 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.

34.12.5 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 *AVR CPU* 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.

34.12.6 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

34.12.7 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
0x00	OSC32K	32.768 kHz from OSC32K
0x01	OSC1K	1.024 kHz from OSC32K
0x02	XTAL32K	32.768 kHz from XOSC32K or external clock from XTAL32K1 pin
0x03	EXTCLK	External clock from either the XOSCHF or from the XTALHF1 pin

34.12.8 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 reading or writing 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. Application software needs to check that the CNTBUSY flag in the RTC.STATUS register is cleared before writing to this register.

Bits 7:0 – CNT[7:0] Counter Low Byte
These bits hold the LSB of the 16-bit Counter register. Application software needs to check that the CNTBUSY flag in the RTC.STATUS register is cleared before writing to this register.

34.12.9 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. Application software needs to check that the PERBUSY flag in the RTC.STATUS register is cleared before writing to this register.

Bits 7:0 – PER[7:0] Period Low Byte
These bits hold the LSB of the 16-bit Period register. Application software needs to check that the PERBUSY flag in the RTC.STATUS register is cleared before writing to this register.

34.12.10 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`.

Application software needs to check that the CMPBUSY flag in the RTC.STATUS register is cleared before writing to this register.

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. Application software needs to check that the CMPBUSY flag in the RTC.STATUS register is cleared before writing to this register.

34.12.11 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.
Note: Application software needs to check that the CTRLBUSY flag in the RTC.PITSTATUS register is cleared before writing to this register.

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
Note: Application software needs to check that the CTRLBUSY flag in the RTC.PITSTATUS register is cleared before writing to this register.

Value	Description
0	Periodic Interrupt Timer disabled
1	Periodic Interrupt Timer enabled

Important: 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. Application software needs to check that the CTRLBUSY flag in the RTC.PITSTATUS register is cleared before writing to this register.

34.12.12 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.

34.12.13 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

34.12.14 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.

34.12.15 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

34.12.16 Periodic Timer Event Generation Control A

Name: PITEVGENCTRLA
Offset: 0x16
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	EVGEN1SEL[3:0]				EVGEN0SEL[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 – EVGENnSEL Event Generator n Select

Value	Name	Description
0x0	OFF	No event generated
0x1	DIV4	CLK_RTC divided by 4
0x2	DIV8	CLK_RTC divided by 8
0x3	DIV16	CLK_RTC divided by 16
0x4	DIV32	CLK_RTC divided by 32
0x5	DIV64	CLK_RTC divided by 64
0x6	DIV128	CLK_RTC divided by 128
0x7	DIV256	CLK_RTC divided by 256
0x8	DIV512	CLK_RTC divided by 512
0x9	DIV1024	CLK_RTC divided by 1024
0xA	DIV2048	CLK_RTC divided by 2048
0xB	DIV4096	CLK_RTC divided by 4096
0xC	DIV8192	CLK_RTC divided by 8192
0xD	DIV16384	CLK_RTC divided by 16384
0xE	DIV32768	CLK_RTC divided by 32768
other	-	Reserved

35. USART - Universal Synchronous and Asynchronous Receiver and Transmitter

35.1 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
- Host SPI Mode
- Multiprocessor Communication Mode
- Start-of-Frame Detection
- IRCOM Module for IrDA® Compliant Pulse Modulation/Demodulation
- LIN Client Support

35.2 Overview

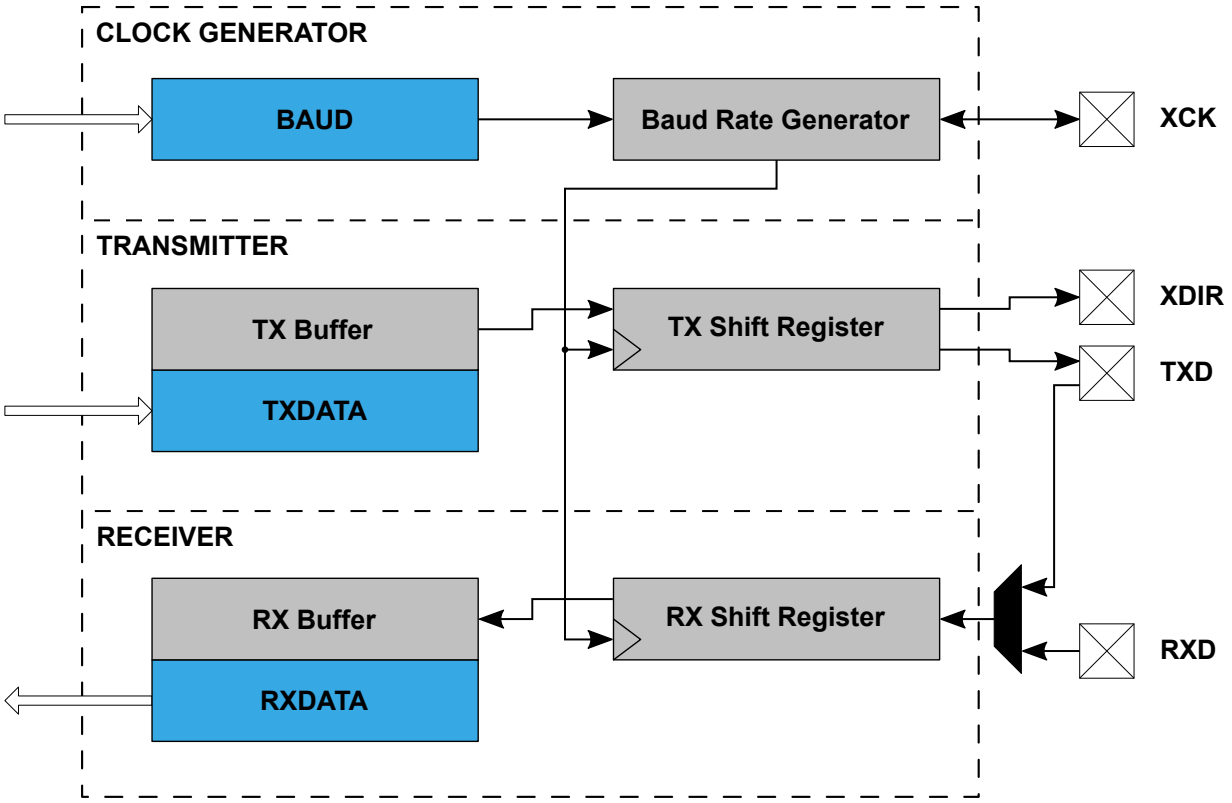
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a fast and flexible serial communication peripheral. The USART supports several different modes of operation that can accommodate multiple types of applications and communication devices. For example, the One-Wire Half-Duplex 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 two-level 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.

35.2.1 Block Diagram

Figure 35-1. USART Block Diagram



35.2.2 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

35.3 Functional Description

35.3.1 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

35.3.2 Operation

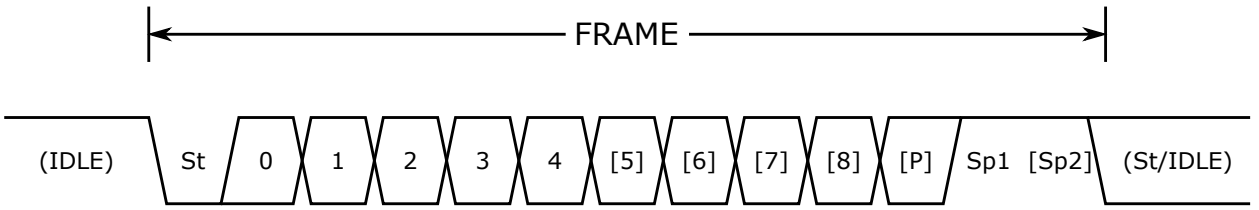
35.3.2.1 Frame Formats

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 35-2. Frame Formats



St Start bit, always low

(n) Data bits (0 to 8)

P Parity bit, may be odd or even

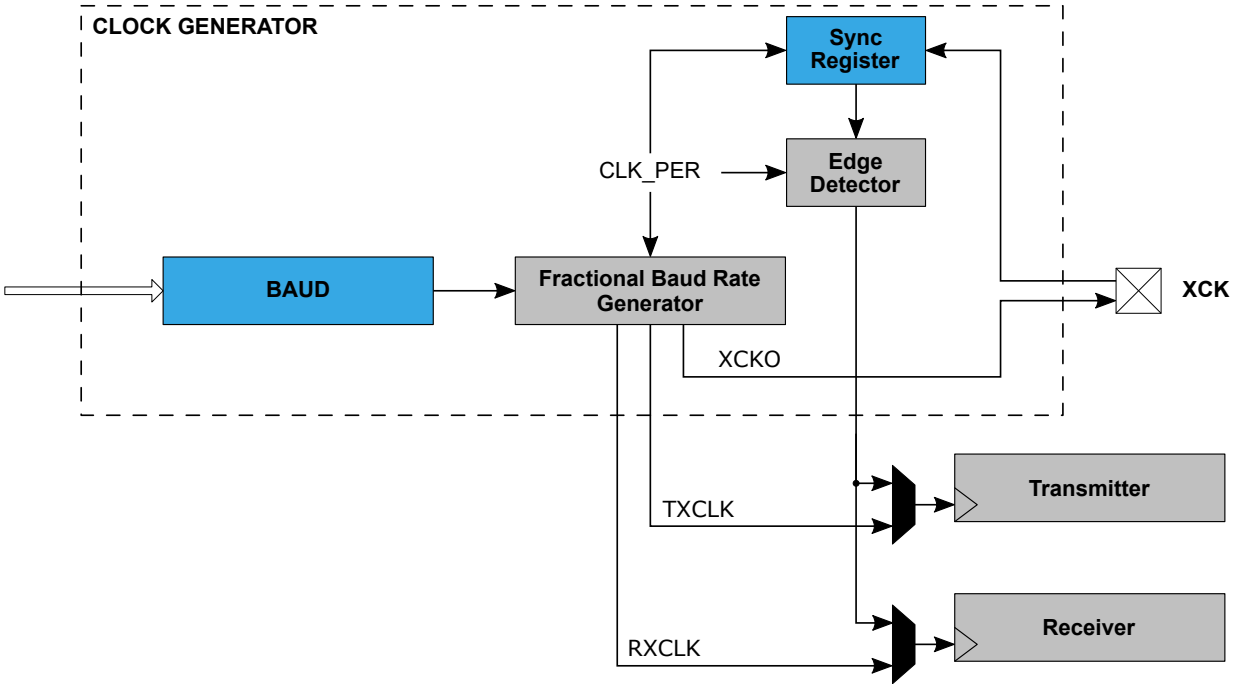
Sp Stop bit, always high

IDLE No transfer on the communication line (RxD or TxD). The Idle state is always high.

35.3.2.2 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 35-3. Clock Generation Logic Block Diagram



35.3.2.2.1 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 35-1. The six Least Significant bits (LSbs) will then hold the fractional part of the desired divisor. Use the fractional part of the BAUD register 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 consider fractional interpretation, so the BAUD values calculated with these equations can be written directly to USARTn.BAUD without any additional scaling.

Table 35-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Conditions	Baud Rate (Bits Per Seconds)	USART.BAUD Register Value Calculation
Asynchronous	$f_{BAUD} \leq \frac{f_{CLK\_PER}}{S}$ $USART.BAUD \geq 64$	$f_{BAUD} = \frac{64 \times f_{CLK\_PER}}{S \times BAUD}$	$BAUD = \frac{64 \times f_{CLK\_PER}}{S \times f_{BAUD}}$
Synchronous Host	$f_{BAUD} \leq \frac{f_{CLK\_PER}}{S}$ $USART.BAUD \geq 64$	$f_{BAUD} = \frac{f_{CLK\_PER}}{S \times BAUD[15:6]}$	$BAUD[15:6] = \frac{f_{CLK\_PER}}{S \times f_{BAUD}}$

S is the number of samples per bit

- Asynchronous Normal mode: S = 16
- Asynchronous Double-Speed mode: S = 8
- Synchronous mode: S = 2

35.3.2.3 Data Transmission

The USART transmitter sends data by periodically driving the transmission line low. The data transmission is initiated by loading the Transmit Data (USARTn.TXDATAL and USARTn.TXDATAH) registers with the data to be sent. The data in the Transmit Data registers are moved to the TX Buffer once it is empty and onwards 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 Data registers or the TX 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.

The Transmit Data registers can only be written when the Data Register Empty Interrupt Flag (the DREIF bit in the USARTn.STATUS register) is set, indicating that they are empty and ready for new data.

When using frames with fewer than eight bits, the Most Significant bits (MSbs) written to the Transmit Data registers are ignored. When the Character Size (CHSIZE) bit field in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), the Transmit Data Register Low Byte (TXDATAL) must be written before the Transmit Data Register High Byte (TXDATAH). When CHSIZE is configured to 9-bit (high byte first), TXDATAH must be written before TXDATAL.

35.3.2.3.1 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, Transmit Data (USARTn.TXDATAL and USARTn.TXDATAH) registers, and TX 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.

35.3.2.4 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 Data Direction (PORTx.DIR) register.

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 RXDATAH is read, depending on the configuration. The register, which does not lead to data being shifted, must be read first to be able to read both bytes before shifting. When the Character Size (CHSIZE) bit field 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.

35.3.2.4.1 Receiver Error Flags

The USART receiver features error detection mechanisms that uncover any 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 (CHSIZE) bit field in the Control C (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.

35.3.2.4.2 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.

35.3.2.4.3 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.

35.3.3 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 host, 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 host or a client 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.

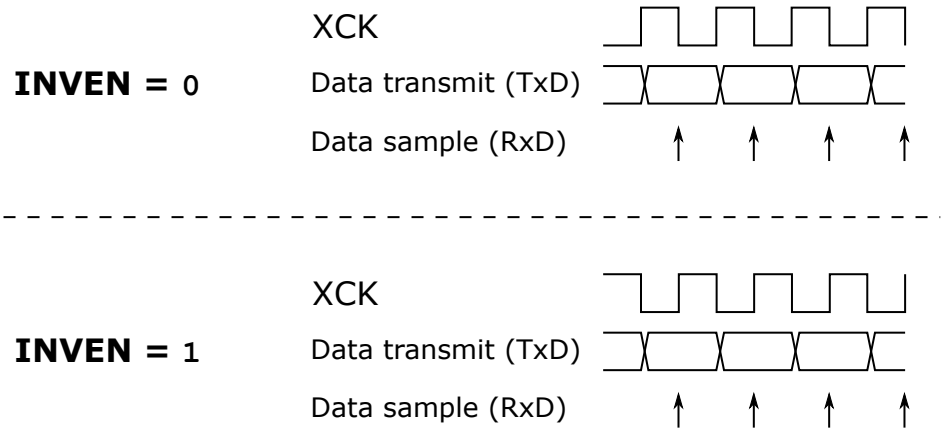
35.3.3.1 Synchronous Operation

35.3.3.1.1 Clock Operation

The XCK pin direction controls whether the transmission clock is an input (Client mode) or an output (Host mode). The corresponding port pin direction must be set to output for Host mode or input

for Client 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 35-4. 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). When 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.

35.3.3.1.2 External Clock Limitations

When the USART is configured in Synchronous Client mode, the XCK signal must be provided externally by the host 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_{Client_XCK}, is therefore limited by:

$$f_{Client\_XCK} < \frac{f_{CLK\_PER}}{4}$$

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.

35.3.3.1.3 USART in Host 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 work as the host device. The SPI is a four-wire interface that enables a host device to communicate with one or multiple clients.

Frame Formats

The serial frame for the USART in Host 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 (UDORD) bit in the Control C (USARTn.CTRLC) register.

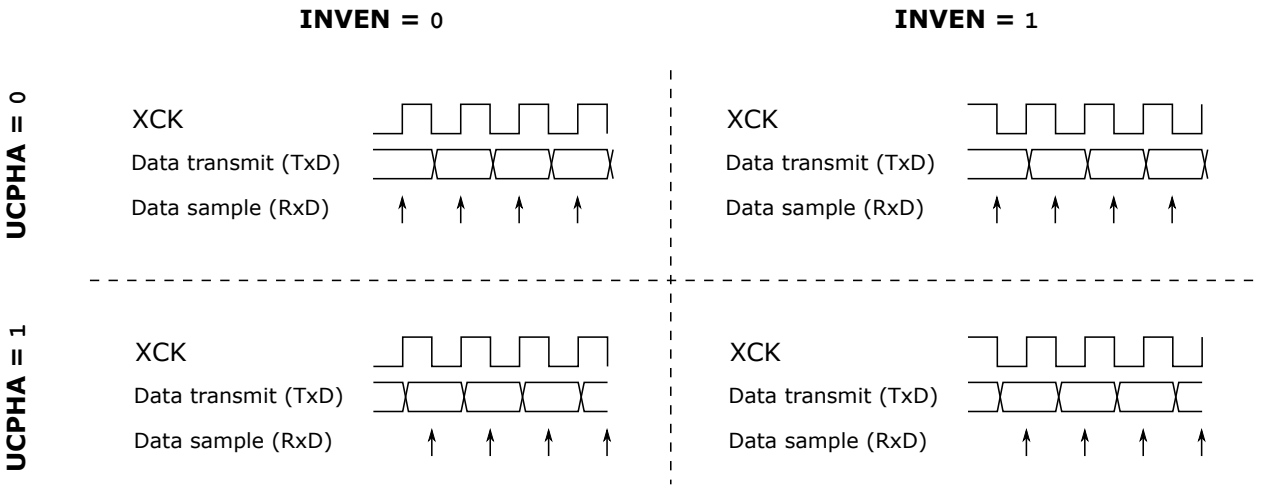
SPI does not use Start, Stop, or Parity bits, so the transmission frame can only consist of the Data bits.

Clock Generation

Being a host device in a synchronous communication interface, the USART in Host SPI mode must generate the interface clock to be shared with the client devices. The interface clock is generated using the fractional Baud Rate Generator, which is described in 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 35-5. Data Transfer Timing Diagrams



The table below further explains the figure above.

Table 35-2. Functionality of the 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 Host SPI mode is functionally identical to the general USART operation, as described in the *Operation* section. The transmitter interrupt flags and corresponding USART interrupts are also identical. See Data Transmission for further description.

Data Reception

Data reception in Host 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, except for the receiver error flags that are not in use and always read as '0'. See Data Reception for further description.

USART in Host SPI Mode vs. SPI

The USART in Host 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 Host SPI mode:

- Write Collision Flag Protection
- Double-Speed mode
- Multi-Host support

A comparison of the pins used with USART in Host SPI mode and with SPI is shown in the table below.

Table 35-3. Comparison of USART in Host SPI Mode and SPI Pins

USART	SPI	Comment
TXD	MOSI	Host out
RXD	MISO	Host in
XCK	SCK	Functionally identical
-	$\overline{SS}$	Not supported by USART in Host SPI mode(1)

Note:

1. For the stand-alone SPI peripheral, this pin is used with the Multi-Host function or as a dedicated Client Select pin. The Multi-Host function is not available with the USART in Host SPI mode, and no dedicated Client Select pin is available.

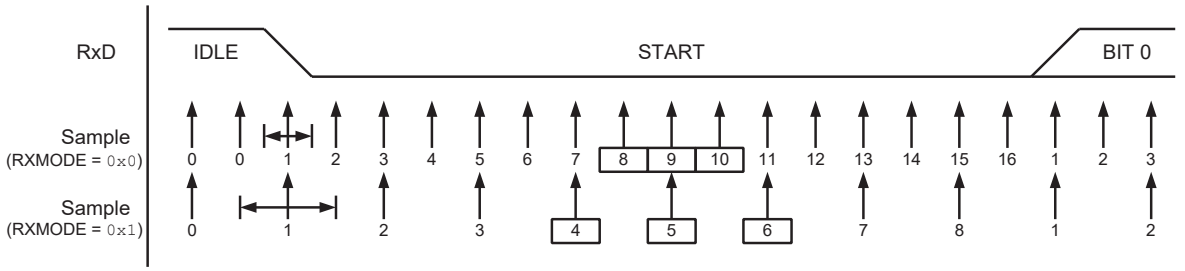
35.3.3.2 Asynchronous Operation

35.3.3.2.1 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 bit field 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 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 35-6. Start Bit Sampling

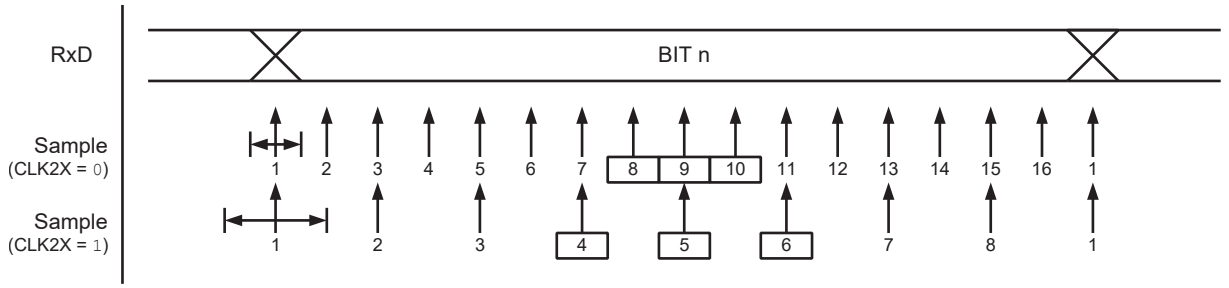


When the clock recovery logic detects a falling edge from the Idle (high) state to the Start bit (low), the Start bit detection sequence is initiated. In the figure above, sample 1 denotes the first sample reading '0'. The clock recovery logic then uses three subsequent samples (samples 8, 9, and 10 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.

35.3.3.2.2 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.

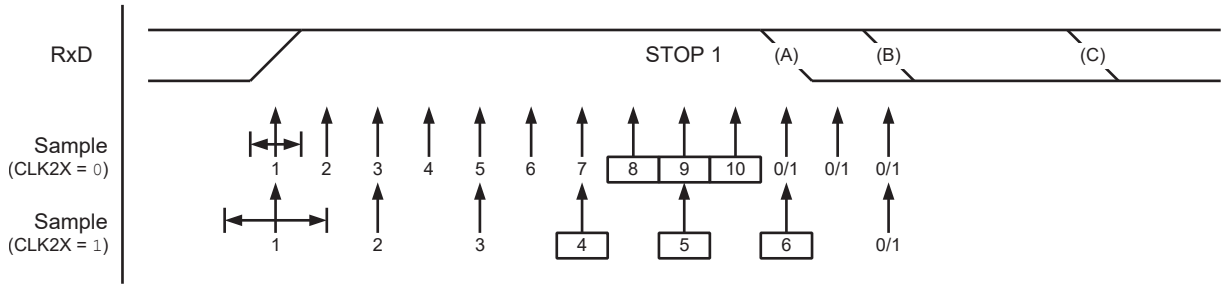
Figure 35-7. Sampling of Data and Parity Bits



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 35-8. 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.

35.3.3.2.3 Error Tolerance

The speed of the internally generated baud rate and the externally received data rate has to be identical, but, due to natural clock source error, this is usually 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 35-4. 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 35-5. 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

35.3.3.2.4 Double-Speed Operation

The 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 bit field in the Control B (USARTn.CTRLB) register to `0x01`.

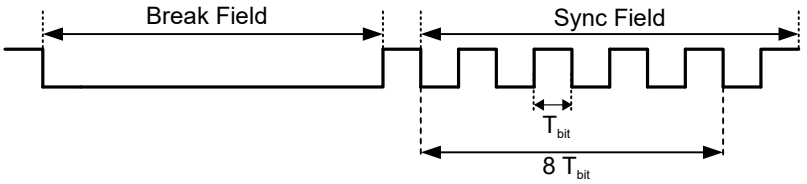
When enabled, the baud rate for a given asynchronous baud rate setting will be doubled, as shown in the equations in The Fractional Baud Rate Generator. In this mode, the receiver will use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery. This requires a more accurate baud rate setting and peripheral clock. See Error Tolerance for more details.

35.3.3.2.5 Auto-Baud

The auto-baud feature lets the USART configure its BAUD register based on input from a communication device, which 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 35-9. 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 bit field 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 bit field 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, which 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`. If the received synchronization field character is not `0x55`, the Inconsistent Sync Field Error Flag (the ISFIF bit in the USARTn.STATUS register) is set, and the baud rate is unchanged.

35.3.3.2.6 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 can 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.

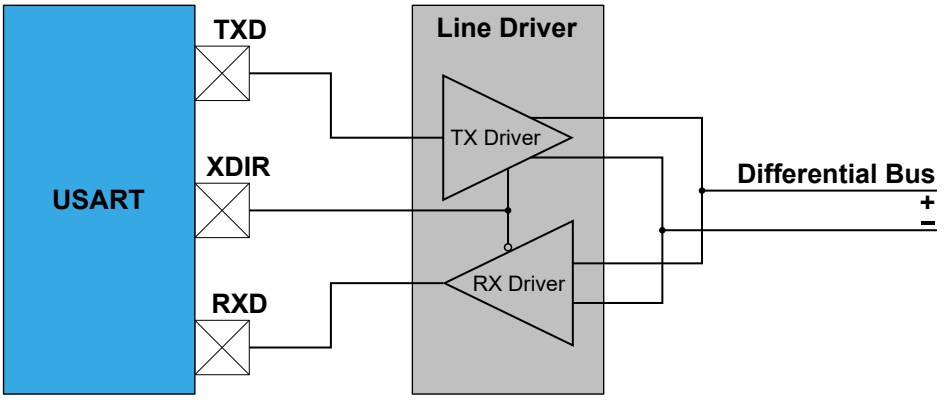
When the USART is transmitting to the TxD/RxD line, it will also receive its transmission. This can be used to detect overlapping transmissions by checking if the received data are the same as the transmitted data.

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 in the USARTn.CTRLA register to '1'.

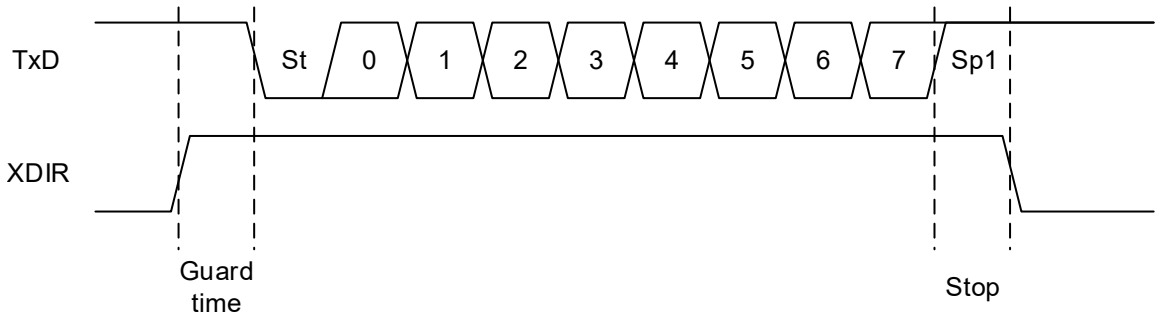
The RS-485 mode supports external line driver devices that convert a single USART transmission into corresponding differential pair signals. It implements automatic control of the XDIR pin that can be used to enable transmission or reception for the line driver device. The USART automatically drives 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 35-10. RS-485 Bus Connection



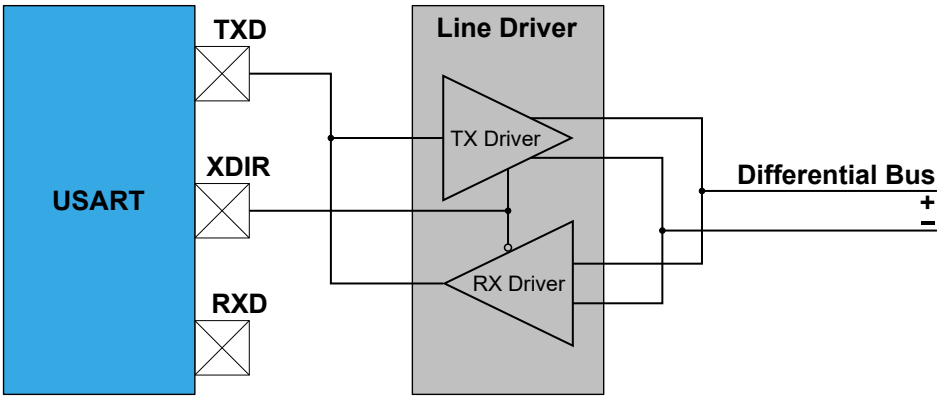
The XDIR pin goes high one baud clock cycle in advance of data being shifted out to allow some guard time to enable the external line driver. The XDIR pin will remain high for the complete frame, including Stop bit(s).

Figure 35-11. 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.

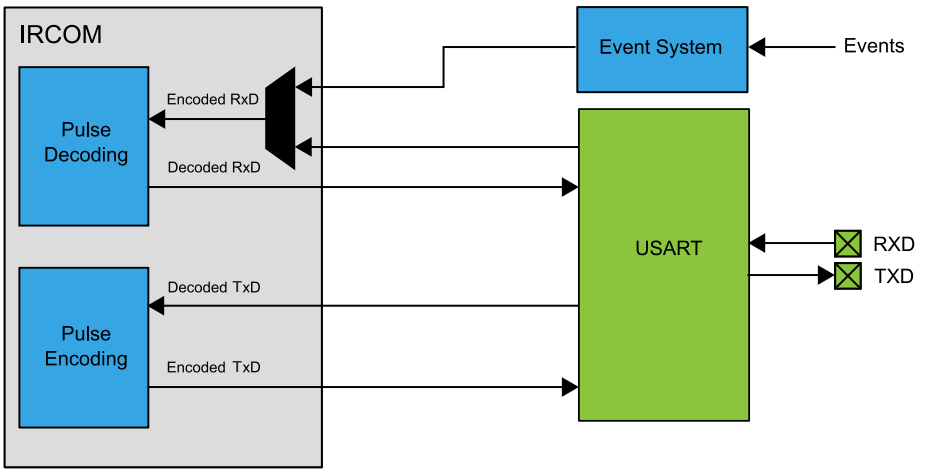
Figure 35-12. RS-485 with Loop-Back Mode Connection



35.3.3.2.7 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 35-13. Block Diagram



The USART is set in IRCOM mode by writing `0x02` to the CMODE bit field 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, which 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.

Double-Speed mode cannot be used for the USART when IRCOM mode is enabled.

35.3.4 Additional Features

35.3.4.1 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) bit field 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).

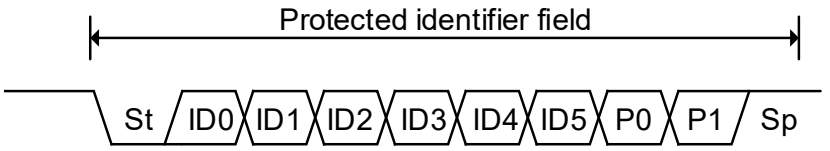
When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the Parity bit of the corresponding frame. If a parity error is detected, the Parity Error flag (the PERR bit in the USARTn.RXDATAH register) is set.

If LIN Constrained Auto-Baud mode is enabled (RXMODE = 0x03 in the USARTn.CTRLB register), a parity check is performed only 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.

$$P0 = ID0 \text{ XOR } ID1 \text{ XOR } ID2 \text{ XOR } ID4$$

$$P1 = \text{ NOT } (ID1 \text{ XOR } ID3 \text{ XOR } ID4 \text{ XOR } ID5)$$

Figure 35-14. Protected Identifier Field and Mapping of Identifier and Parity Bits



35.3.4.2 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 concerning 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 35-6. 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

.....continued			
SFDEN	RXSIF Interrupt	RXCIF Interrupt	Comment
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.

35.3.4.3 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 (USARTn.CTRLB) register. 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 client MCU has been addressed, it will receive the following data frames as usual, while the other client MCUs will ignore the frames until another address frame is received.

35.3.4.3.1 Using Multiprocessor Communication

Use the following procedure to exchange data in Multiprocessor Communication mode (MPCM):

1. All client MCUs are in Multiprocessor Communication mode.
2. The host MCU sends an address frame, and all clients receive and read this frame.
3. Each client MCU determines if it has been selected.
4. The addressed MCU will disable MPCM and receive all data frames. The other client 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 host.

The process then repeats from step 2.

35.3.5 Events

The USART can generate the events described in the table below.

Table 35-7. Event Generators in USART

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
USARTn	XCK	The clock signal in SPI Host mode and Synchronous USART Host mode	Pulse	CLK_PER	One XCK period

The table below describes the event user and its associated functionality.

Table 35-8. Event Users in USART

User Name		Description	Input Detection	Async/Sync
Peripheral	Input			
USARTn	IREI	USARTn IrDA event input	Pulse	Sync

35.3.6 Interrupts

Table 35-9. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
RXC	Receive Complete interrupt	• There is unread data in the receive buffer (RXCIE) • Receive of Start-of-Frame detected (RXSIE) • 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 (USARTn.STATUS) register.

An interrupt source is enabled or disabled by writing to the corresponding bit in the Control A (USARTn.CTRLA) 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 USARTn.STATUS register for details on how to clear Interrupt flags.

35.4 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]						

35.5 Register Description

35.5.1 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, must be read first to be able to read both bytes before shifting.

When the Character Size (CHSIZE) bit field in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a read of RXDATAH shifts the receive buffer, or else, 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

35.5.2 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, must 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, or else, 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 Host 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 Host 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, or else, this flag cleared. For details on parity calculation, refer to Parity.
This flag is not used in the Host 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) bit field 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.

35.5.3 Transmit Data Register Low Byte

Name: TXDATAL
Offset: 0x02
Reset: 0x00
Property: -

The data written to this register is automatically loaded into the TX Buffer and through to 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) must be written to the Transmit Data Register High Byte (USARTn.TXDATAH). In that case, the buffer shifts data either when the Transmit Data Register Low Byte (USARTn.TXDATAL) or the Transmit Data Register High Byte (USARTn.TXDATAH) is written, depending on the configuration. The register, which does not lead to data being shifted, must be written first to be able to write both registers before shifting.

When the Character Size (CHSIZE) bit field in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a write of the Transmit Data Register High Byte shifts the transmit buffer. Otherwise, the Transmit Data Register Low Byte 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 Low Byte

35.5.4 Transmit Data Register High Byte

Name: TXDATAH
Offset: 0x03
Reset: 0x00
Property: -

The data written to this register is automatically loaded into the TX Buffer and through to 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) must be written to the Transmit Data Register High Byte (USARTn.TXDATAH). In that case, the buffer shifts data either when the Transmit Data Register Low Byte (USARTn.TXDATAL) or the Transmit Data Register High Byte (USARTn.TXDATAH) is written, depending on the configuration. The register, which does not lead to data being shifted, must be written first to be able to write both registers before shifting.

When the Character Size (CHSIZE) bit field in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a write of the Transmit Data Register High Byte shifts the transmit buffer. Otherwise, the Transmit Data Register Low Byte 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 High Byte

35.5.5 USART Status Register

Name: STATUS
Offset: 0x04
Reset: 0x20
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 Data (USARTn.TXDATAL and USARTn.TXDATAH) registers are empty and cleared when they contain data not yet 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 Host 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 the *Auto-Baud* section 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 are received. It can also be cleared by writing a '1' to it. See the *Auto-Baud* section for more information.

Bit 0 – WFB Wait For Break
Setting this bit enables the Wait For Break feature for the following incoming frame. After this frame, the feature is automatically disabled.

35.5.6 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

35.5.7 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 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 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 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 the One-Wire Mode section 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 this bit field selects the receiver mode of the USART.

- Writing the bits to 0x00 enables Normal-Speed (NORMAL) mode. When the USART Communication Mode (CMODE) bit field in the Control C (USARTn.CTRLC) register is configured to Asynchronous USART (ASYNCHRONOUS) or Infrared Communication (IRCOM), always write the RXMODE bit field to 0x00.
- Writing the bits to 0x01 enables Double-Speed (CLK2X) mode. Refer to Double-Speed Operation for more information.
- Writing the bits to 0x02 enables Generic Auto-Baud (GENAUTO) mode. Refer to the *Auto-Baud* section for more information.

- Writing the bits to 0x03 enables Lin Constrained Auto-Baud (LINAUTO) mode. Refer to the *Auto-Baud* section 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 Multiprocessor Communication for more information.

Value	Description
0	Multi-Processor Communication mode is disabled
1	Multi-Processor Communication mode is enabled

35.5.8 Control C - Normal Mode

Name: CTRLC
Offset: 0x07
Reset: 0x03
Property: -

This register description is valid for all modes except the Host SPI mode. When the USART Communication Mode (CMODE) bit field in this register is written to 'MSPI', see CTRLC - Host 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
This bit field selects the communication mode of the USART.
Writing a `0x03` to these bits alters the available bit fields in this register. See CTRLC - Host SPI mode.

Value	Name	Description
0x00	ASYNCHRONOUS	Asynchronous USART
0x01	SYNCHRONOUS	Synchronous USART
0x02	IRCOM	Infrared Communication
0x03	MSPI	Host SPI

Bits 5:4 – PMODE[1:0] Parity Mode
This bit field enables and selects the type of parity generation. See 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
This bit field selects 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)

Value	Name	Description
0x07	9BITH	9-bit (High byte first)

35.5.9 Control C - Host SPI Mode

Name: CTRLC
Offset: 0x07
Reset: 0x02
Property: -

This register description is valid only when the USART is in Host SPI mode (CMODE written to MSPI). For other CMODE values, see CTRLC - Normal Mode.

See USART in Host SPI Mode for a full description of the Host 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	1	

Bits 7:6 – CMODE[1:0] USART Communication Mode
This bit field selects 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	Host 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 the Clock Generation section for more information.

Value	Description
0	Data are sampled on the leading (first) edge
1	Data are sampled on the trailing (last) edge

35.5.10 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 35-1.

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
This bit field holds the MSB of the 16-bit Baud register.

Bits 7:0 – BAUD[7:0] USART Baud Rate Low Byte
This bit field holds the LSB of the 16-bit Baud register.

35.5.11 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 must 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)

35.5.12 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

35.5.13 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 IRCOM Mode of Operation for more information.

Value	Description
0	IrDA Event input is disabled
1	IrDA Event input is enabled

35.5.14 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 only have an effect 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.

35.5.15 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 an 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.

36. SPI - Serial Peripheral Interface

36.1 Features

- Full Duplex, Three-Wire Synchronous Data Transfer
- Host or Client 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) Host SPI Mode

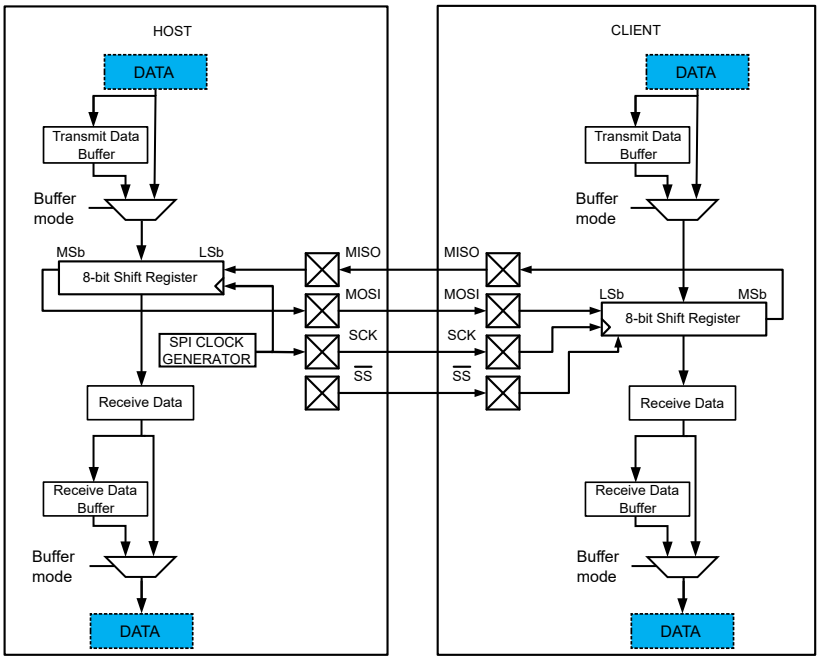
36.2 Overview

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 host or client. The host initiates and controls all data transactions.

The interconnection between host and client devices with SPI is shown in the block diagram below. The system consists of two shift registers and a server clock generator. The SPI host initiates the communication cycle by pulling the desired client's Client Select ($\overline{SS}$) signal low. The host and client prepare the data to be sent to their respective shift registers, and the host generates the required clock pulses on the SCK line to exchange data. Data are always shifted from host to client on the host output, client input (MOSI) line, and from client to host on the host input, client output (MISO) line.

36.2.1 Block Diagram

Figure 36-1. 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 Host mode, the SPI has a clock generator to generate the SCK clock. In Client mode, the received SCK clock is synchronized and sampled to trigger the shifting of data in the shift register.

36.2.2 Signal Description

Table 36-1. Signals in Host and Client Mode

Signal	Description	Pin Configuration	
		Host Mode	Client Mode
MOSI	Host Out Client In	User defined(1)	Input
MISO	Host In Client Out	Input	User defined(1,2)
SCK	Serial Clock	User defined(1)	Input
$\overline{SS}$	Client 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 Client 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 36-1 is overridden.

36.3 Functional Description

36.3.1 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 host/client operation by writing the Host/Client Select (MASTER) bit in the Control A (SPIn.CTRLA) register.
3. In Host 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-host support in Host mode, write ‘1’ to the Client Select Disable (SSD) bit in SPIn.CTRLB.
8. Enable the SPI by writing a ‘1’ to the ENABLE bit in SPIn.CTRLA.

36.3.2 Operation

36.3.2.1 Host Mode Operation

When the SPI is configured in Host mode, a write to the SPIn.DATA register will start a new transfer. The SPI host can operate in two modes, Normal and Buffer, as explained below.

36.3.2.1.1 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 been 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 been 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.

36.3.2.1.2 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 does not affect Host 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 in 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.

36.3.2.1.3 $\overline{SS}$ Pin Functionality in Host Mode - Multi-Host Support

In Host mode, the Client 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 Host to Client mode. This allows multiple SPI hosts 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. 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 Host SPI operation. A low level will be interpreted as another host is trying to take

control of the bus. This will switch the SPI into Client mode, and the hardware of the SPI will perform the following actions:

1. The Host (MASTER) bit in the SPI Control A (SPIn.CTRLA) register is cleared, and the SPI system becomes a client. The direction of the SPI pins will be switched when the conditions in Table 36-2 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 36-2. 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	Host activated (selected)
	Low	Host deactivated, switched to Client mode
Output	High	Host activated (selected)
	Low	

Note: If the device is in Host mode and it cannot be ensured that the $\overline{SS}$ pin will stay high between two transmissions, the status of the Host (MASTER) bit in SPIn.CTRLA has to be checked before a new byte is written. After the Host 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 Host mode.

36.3.2.2 Client Mode

In Client mode, the SPI peripheral receives the SPI clock and Client Select from a Host. Client mode supports three operational modes: One Normal mode and two configurations for the Buffered mode. In Client mode, the control logic will sample the incoming signal on the SCK pin.

36.3.2.2.1 Normal Mode

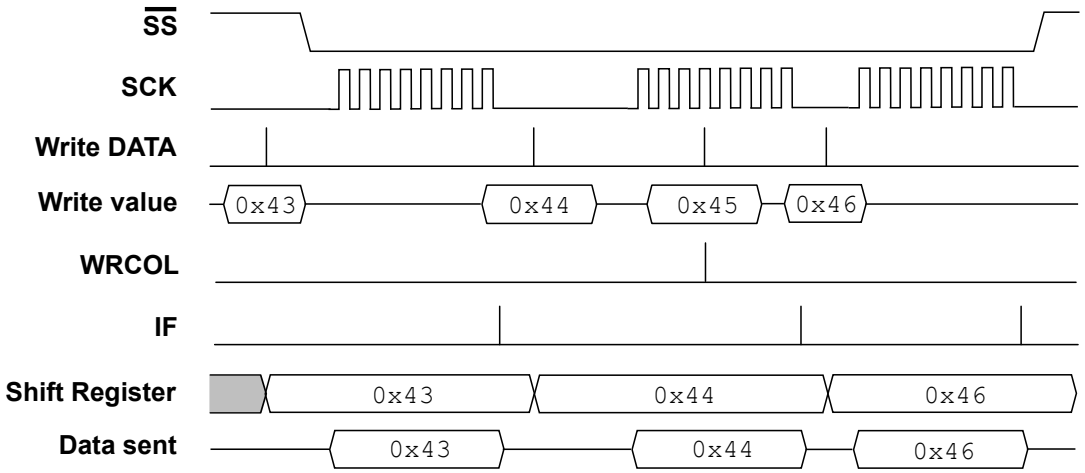
In Normal mode, the SPI peripheral will remain Idle as long as the $\overline{SS}$ pin is driven high. In this state, the software may update the contents of the DATA register, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. If the $\overline{SS}$ pin is driven low, the client 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 been 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 client will not receive any new data. Any partially received packet in the shift register will be lost.

Figure 36-2 shows a transmission sequence in Normal mode. Notice how the value `0x45` is written to the DATA register but never transmitted.

Figure 36-2. 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.

36.3.2.2.2 Buffer Mode

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 to avoid data collisions.

This mode will enable two receive buffers and one transmit buffer. Both will have separate interrupt flags, transmit complete and receive complete. Figure 36-1 shows the extra buffers.

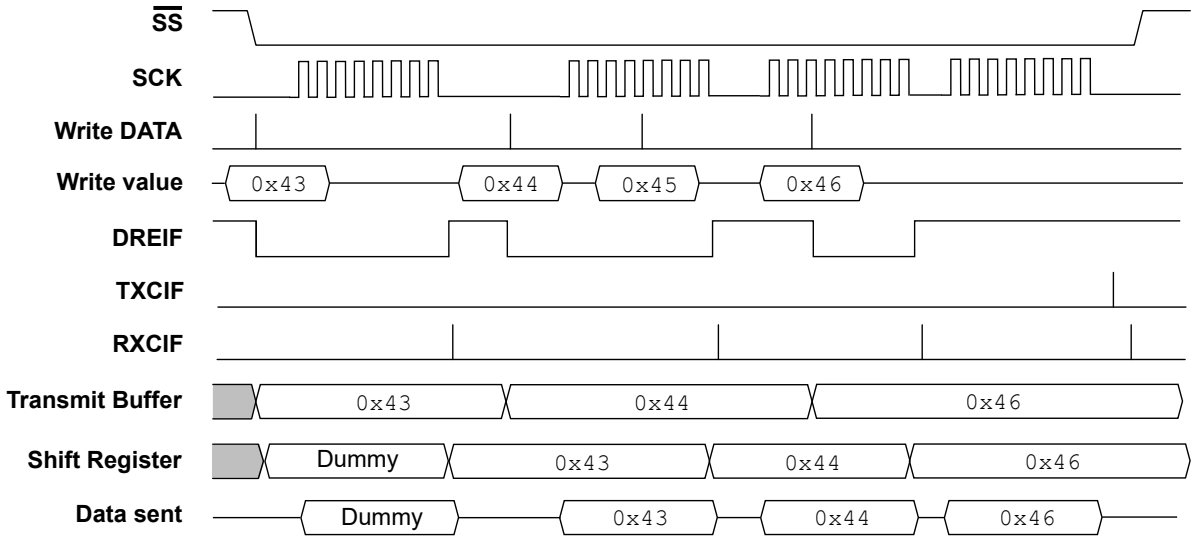
When Buffer mode is enabled it can work in two different ways. The Buffer Mode Wait for Receive (BUFWR) bit in the Control B (SPIn.CTRLB) register controls how the Buffer mode works. The details of how they work including timing diagrams are described below.

Note: When operating as a client in Buffered mode and the SPI clock is close to maximum frequency, the client may not be able to set up data in time for the first sample edge during back-to-back transfers. Refer to the *Electrical Characteristics - SPI* section for details.

Client Buffer Mode with Wait for Receive Bit Written to '0'

In Client 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 36-3 shows a transmission sequence with this configuration. Notice how the value 0x45 is written to the Data (SPIn.DATA) register but never transmitted.

Figure 36-3. SPI Timing Diagram in Buffer Mode with BUFWR in SPIn.CTRLB Written to '0'



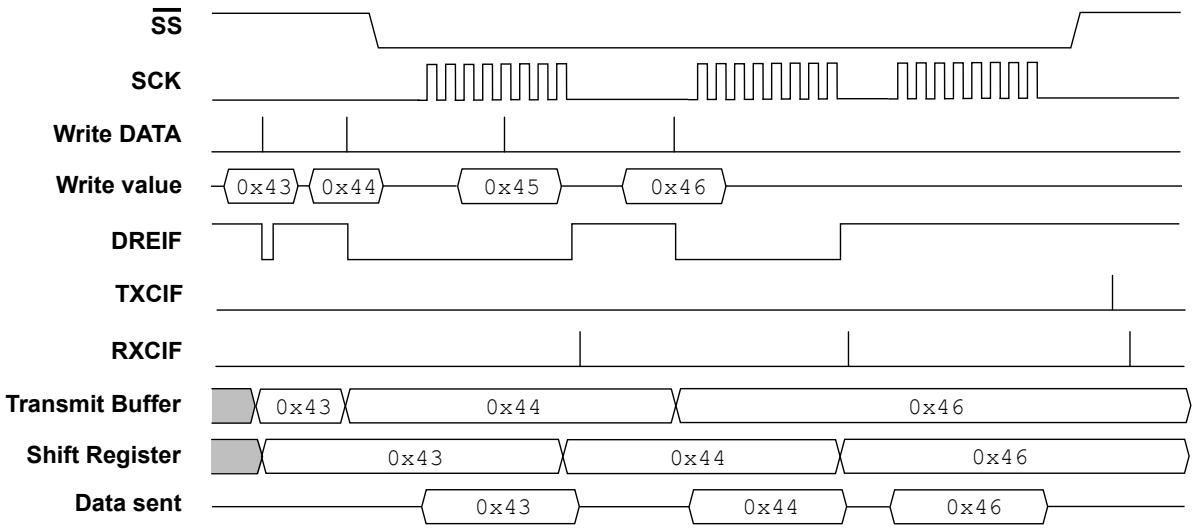
When the Wait for Receive (BUFWR) bit in SPIn.CTRLB is written to '0', all writes to the Data (SPIn.DATA) register go to the Transmit Data Buffer register. The figure above shows that the value 0x43 is written to the Data (SPIn.DATA) register but not immediately transferred to the shift register, so 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 same 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 in the Transmit Data Buffer register has been sent.

Client Buffer Mode with Wait for Receive Bit Written to '1'

In Client 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 36-4 shows a transmission sequence with this configuration. Notice how the value 0x45 is written to the Data (SPIn.DATA) register but never transmitted.

Figure 36-4. 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. 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'.

36.3.2.2.3 $\overline{SS}$ Pin Functionality in Client Mode

The Client Select ($\overline{SS}$) pin plays a central role in the operation of the SPI. Depending on the SPI mode 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 Client mode, the $\overline{SS}$, 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 an 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 an output, the MISO pin will be tri-stated. Table 36-3 shows an overview of the $\overline{SS}$ pin functionality.

Table 36-3. 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	Client deactivated (deselected)	Tri-stated	Input
	Low	Client activated (selected)	Output	Input

Note: In Client 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 Client bit counter synchronized with the host clock generator.

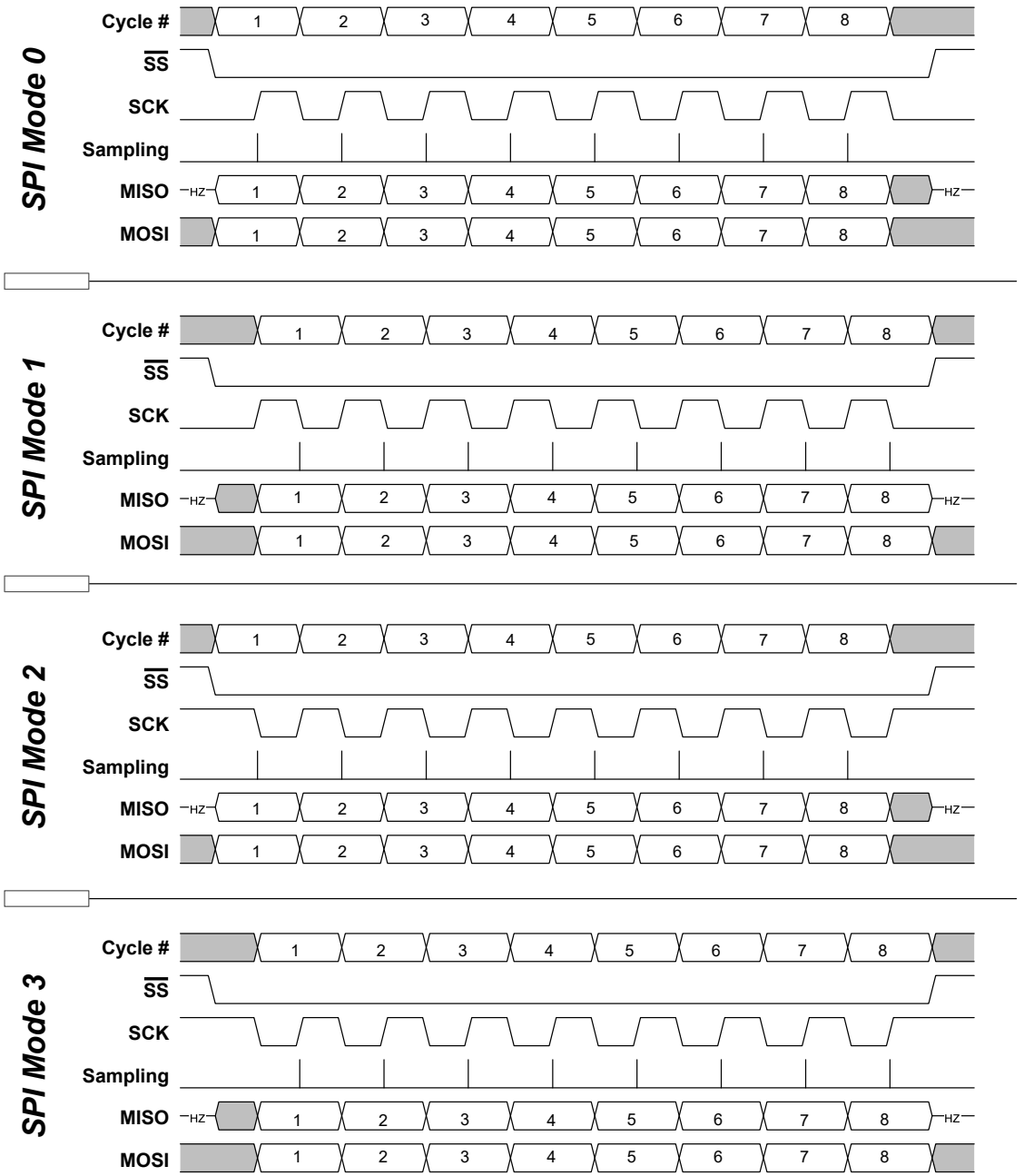
36.3.2.3 Data Modes

There are four combinations of SCK phase and polarity concerning the 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 36-5. SPI Data Transfer Modes



36.3.2.4 Events

The SPI can generate the following events:

Table 36-4. Event Generators in SPI

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Module	Event				
SPIn	SCK	SPI Host clock	Level	CLK_PER	Minimum two CLK_PER periods

The SPI has no event users.

Refer to the *EVSYS - Event System* section for more details regarding event types and Event System configuration.

36.3.2.5 Interrupts

Table 36-5. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions	
		Normal Mode	Buffer Mode
SPIn	SPI interrupt	• IF: Interrupt Flag interrupt • WRCOL: Write Collision interrupt	• SSI: Client Select Trigger Interrupt • DRE: Data Register Empty interrupt • TXC: Transfer Complete interrupt • 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.

36.4 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]							

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
		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 Host/Client 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 Host mode.
This behavior is controlled by the Client Select Disable (SSD) bit in SPIn.CTRLB.

Value	Description
0	SPI Client mode selected
1	SPI Host 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 Host mode.

Value	Description
0	SPI speed (SCK frequency) is not doubled
1	SPI speed (SCK frequency) is doubled in Host mode

Bits 2:1 – PRESC[1:0] Prescaler

This bit field controls the SPI clock rate configured in Host mode. These bits have no effect in Client 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

36.5.2 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 Client Select Disable
If this bit is set when operating as SPI Host (MASTER = 1 in SPIn.CTRLA), $\overline{SS}$ does not disable Host mode.

Value	Description
0	Enable the Client Select line when operating as SPI host
1	Disable the Client Select line when operating as SPI host

Bits 1:0 – MODE[1:0] Mode
These bits select the Transfer mode. The four combinations of SCK phase and polarity concerning 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 Trailing edge: Falling, setup
0x1	1	Leading edge: Rising, setup Trailing edge: Falling, sample
0x2	2	Leading edge: Falling, sample Trailing edge: Rising, setup
0x3	3	Leading edge: Falling, setup Trailing edge: Rising, sample

36.5.3 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 Client Select Trigger Interrupt Enable
In Buffer mode, this bit enables the Client 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.

36.5.4 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 Host mode, this will also set this flag. The IF is cleared by writing a '1' to it. 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.

36.5.5 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 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 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 Client Select Trigger Interrupt Flag
This flag indicates that the SPI has been in Host mode, and the $\overline{SS}$ pin has been pulled low externally, so the SPI is now working in Client mode. The flag will only be set if the Client 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.

36.5.6 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 Host mode while preparing data for sending in Client 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.

37. TWI - Two-Wire Interface

37.1 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
  - Client arbitration allows support for the Address Resolution Protocol (ARP) in software
  - Configurable SMBus Layer 1 time-outs in hardware
  - Independent time-outs for Dual mode
- Independent Host and Client Operation
  - Combined (same pins) or Dual mode (separate pins)
  - Single or multi-host bus operation with full arbitration support
- Hardware Support for Client 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

37.2 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).

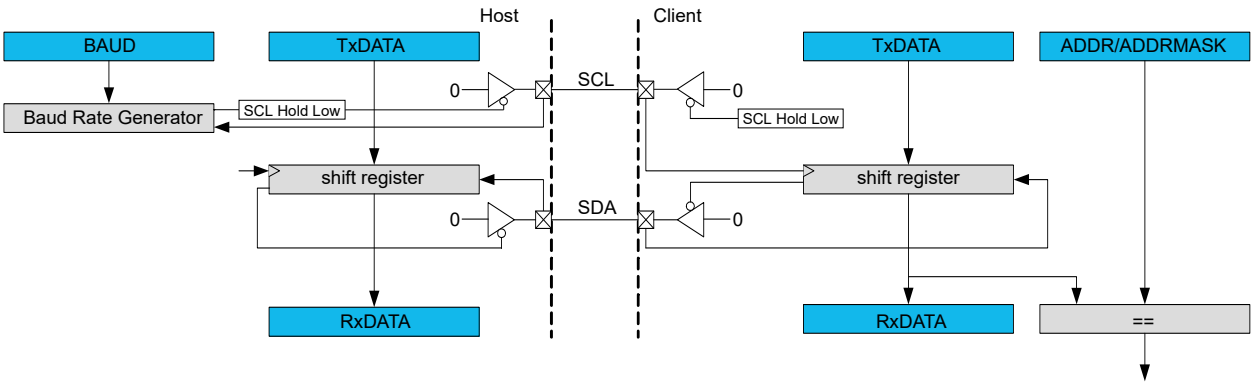
The TWI bus connects one or several client devices to one or several host devices. Any device connected to the bus can act as a host, a client, or both. The host generates the SCL using a Baud Rate Generator (BRG) and initiates data transactions by addressing one client and telling whether it wants to transmit or receive data. The BRG can generate the Standard mode (Sm) and Fast mode (Fm, Fm+) bus frequencies from 100 kHz 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 the Host and Client modes.

The TWI supports multi-host bus operations and arbitration. An arbitration scheme handles cases where more than one host tries to transmit data simultaneously. The TWI also supports Smart mode, which can auto-trigger operations and thus reduce software complexity. The TWI supports Dual mode with simultaneous host and client operations implemented as independent units with separate enabling and configuration. The TWI supports Quick Command mode, where the host can address a client without exchanging data.

37.2.1 Block Diagram

Figure 37-1. TWI Block Diagram



37.2.2 Signal Description

Signal	Description	Type
SCL	Serial Clock Line	Digital I/O
SDA	Serial Data Line	Digital I/O

37.3 Functional Description

37.3.1 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 connecting multiple devices on a serial bus. A device connected to the bus can be a host or a client. Only host devices can control the bus and the bus communication.

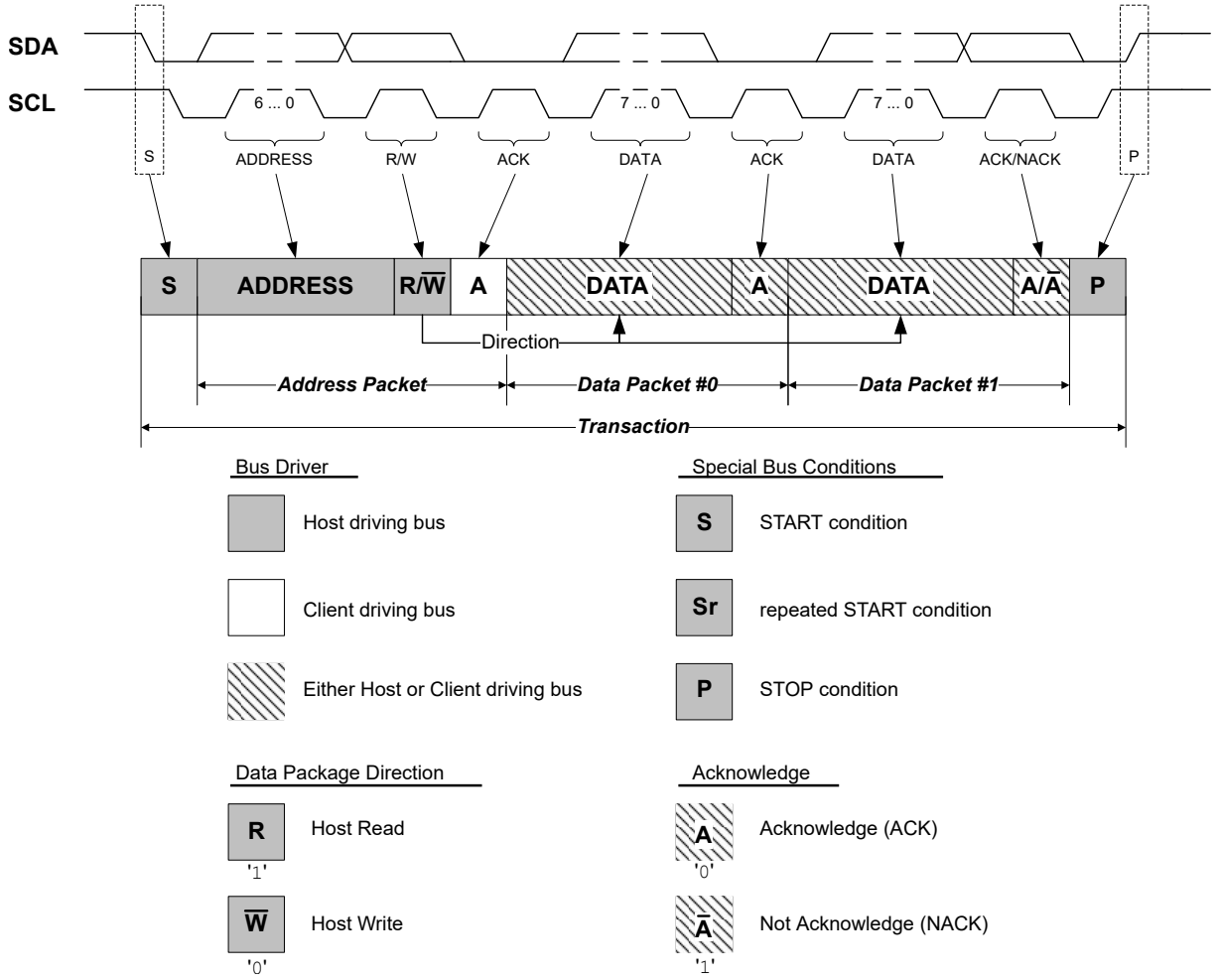
A unique address is assigned to each client device connected to the bus, and the host will use it to control the client and initiate a transaction. Several hosts can connect to the same bus, called a multi-host environment. An arbitration mechanism is provided for resolving bus ownership among hosts since only one host device may own the bus at any given time.

A host indicates the start of a transaction by issuing a Start condition (S) on the bus. The host provides the clock signal for the transaction. An address packet with a 7-bit client address (ADDRESS) and a direction bit, representing whether the host wishes to read or write data (R/$\overline{W}$), are then sent.

The addressed I^2C client 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 transferring all the data packets (DATA), the host issues a Stop condition (P) on the bus to end the transaction.

Figure 37-2. Basic TWI Transaction Diagram Topology for a 7-Bit Address Bus



37.3.2 TWI Basic Operation

37.3.2.1 Initialization

If used, configure the following bits before enabling the TWI peripheral:

- 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

37.3.2.1.1 Host Initialization

Write the Host Baud Rate (TWIn.MBAUD) register to a value that will result in a valid TWI bus clock frequency. Writing a '1' to the Enable TWI Host (ENABLE) bit in the Host Control A (TWIn.MCTRLA) register will enable the TWI host. The Bus State (BUSSTATE) bit field from the Host Status (TWIn.MSTATUS) register must be set to 0x1 to force the bus state to Idle.

37.3.2.1.2 Client Initialization

Follow these steps to initialize the client:

1. Before enabling the TWI client, configure the SDA Setup Time (SDASETUP) bit in the Control A (TWIn.CTRLA) register.
2. Write the client address to the Client Address (TWIn.SADDR) register.
3. Write a '1' to the Enable TWI Client (ENABLE) bit in the Client Control A (TWIn.SCTRLA) register to enable the TWI client.

The TWI client will now wait for a host device to issue a Start condition and the matching client address.

37.3.2.2 TWI Host Operation

The TWI host is byte-oriented, with an optional interrupt after each byte. There are separate interrupt flags for the host write and read operation. Interrupt flags can also be used for polled operations. Dedicated status flags indicate 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, giving the host 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 automatically handling most conditions.

37.3.2.2.1 Clock Generation

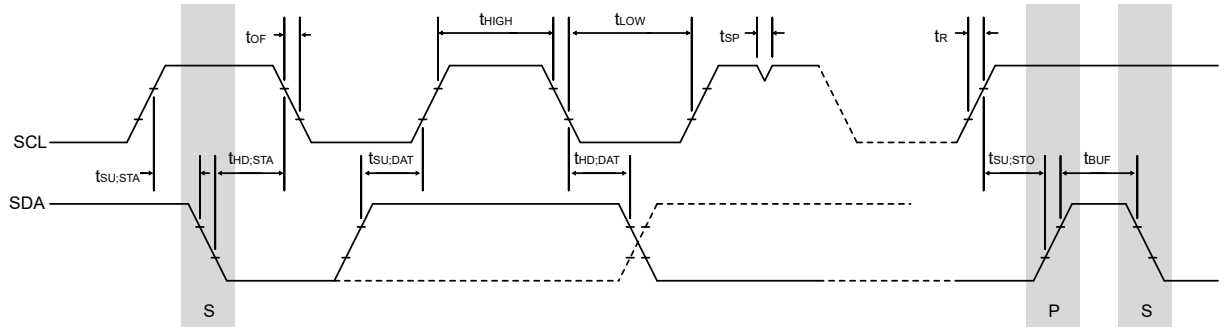
The TWI supports several transmission modes with different frequency limitations:

- Standard mode (Sm) up to 100 kHz
- Fast mode (Fm) up to 400 kHz
- Fast mode Plus (Fm+) up to 1 MHz

The Host 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 Host Baud Rate (TWIn.MBAUD) register, while the rise (t_R) and fall (t_{OF}) times are determined by the bus topology.

Figure 37-3. SCL Timing



- t_{LOW} is the low period of the SCL clock
- t_{HIGH} is the high period of the 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:

$$f_{SCL} = \frac{1}{t_{LOW} + t_{HIGH} + t_{OF} + t_R}[\text{Hz}]$$

The SCL clock is designed to have a 50/50 duty cycle, where the low period of the duty cycle comprises 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:

$$f_{SCL} = \frac{f_{CLK\_PER}}{10 + 2 \times BAUD + f_{CLK\_PER} \times t_R} \quad (1)$$

Equation 1 can be transformed to express BAUD:

$$BAUD = \frac{f_{CLK\_PER}}{2 \times f_{SCL}} - \left(5 + \frac{f_{CLK\_PER} \times t_R}{2}\right) \quad (2)$$

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:

$$t_{LOW\_Fm} = t_{LOW\_Fm+} = \frac{BAUD + 6 + min(SCLDUTY, BAUD)}{f_{CLKPER}} - t_{OF} \quad (3.1)$$

$$t_{LOW} = \frac{BAUD + 6}{f_{CLK\_PER}} - t_{OF} \quad (3.2)$$

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:

$$BAUD = f_{CLK\_PER} \times (t_{LOW\_mode} + t_{OF}) - 3 \quad (4)$$

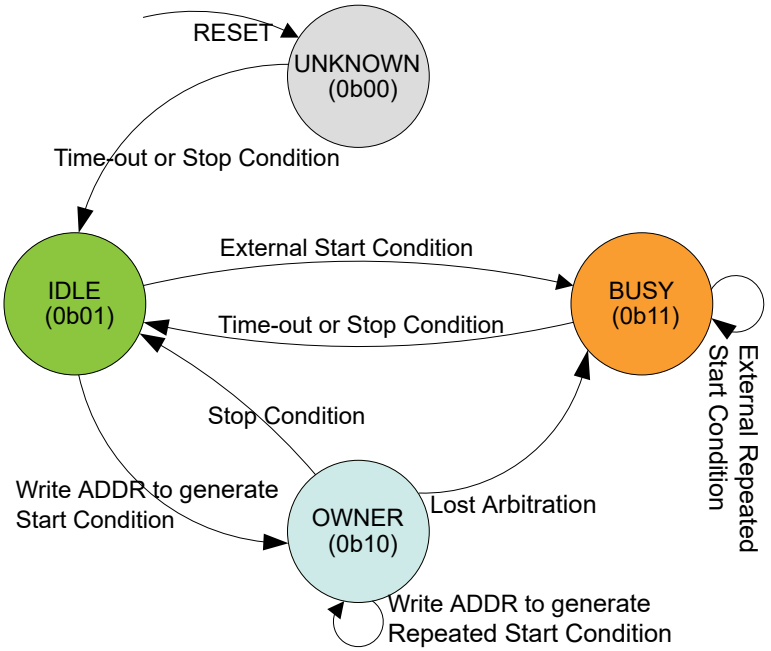
37.3.2.2.2 TWI Bus State Logic

The bus state logic continuously monitors the activity on the TWI bus when the host 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 Host 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 37-4. Bus State Diagram



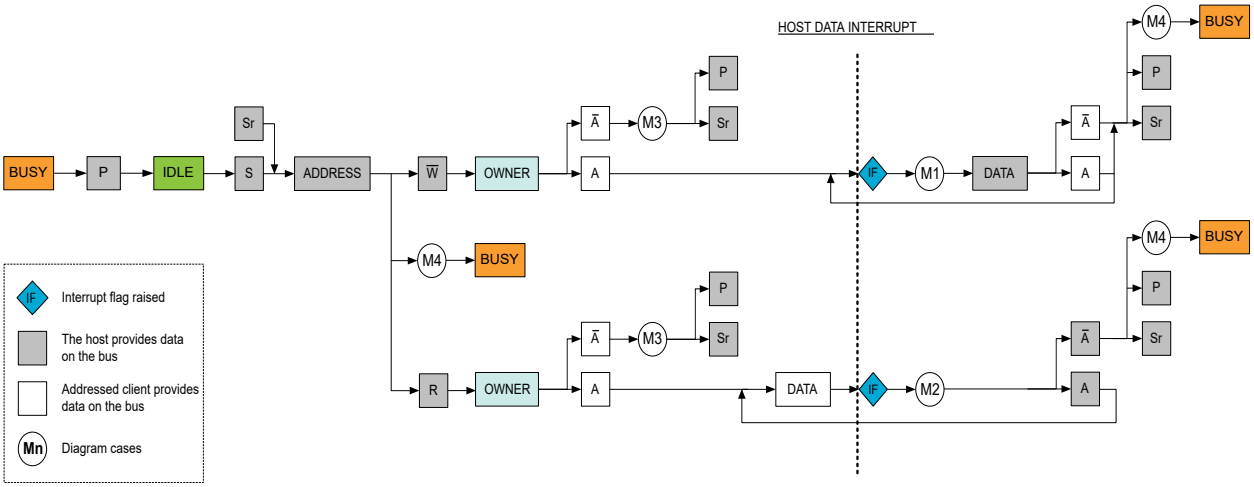
1. **Unknown**: The bus state machine is active when the TWI host is enabled. After enabling the TWI host, performing a system Reset, or disabling the TWI host, the bus state is Unknown.
2. **Idle**: The bus state machine can be forced to enter the Idle state by writing `0x1` to the Bus State (BUSSTATE) bit field. The bus state logic cannot be forced into any other state. If no state is set by the application software when the first Stop condition is detected, the bus state will become Idle. If the Inactive Bus Time-Out (TIMEOUT) bit field from the Host 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 host 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.

37.3.2.2.3 Transmitting Address Packets

The host starts performing a bus transaction when the Host Address (TWIn.MADDR) register is written with the client address and the R/$\overline{W}$ direction bit. The value of the MADDR register is then copied into the Host Data (TWIn.MDATA) register. If the bus state is Busy, the TWI host 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 37-5. TWI Host Operation



Case M1: Address Packet Transmit Complete - Direction Bit Set to '0'

If a client 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 Host Status (TWIn.MSTATUS) register.

The clock hold is active at this point, forcing the SCL low, which 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 client

Case M2: Address Packet Transmit Complete - Direction Bit Set to '1'

If a client device responds to the address packet with an ACK, the RXACK flag is set to '0', and the client can start sending data to the host without any delays because the client 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 client

Case M3: Address Packet Transmit Complete - Address not Acknowledged by Client

If no client 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 client is busy with other tasks or is in a sleep mode and cannot 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 Host Control B (TWIn.MCTRLB) register, which is the recommended action

Case M4: Arbitration Lost or Bus Error

If the arbitration is lost, the WIF and the Arbitration Lost (ARBLOST) flags in the Host 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 host 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 Host 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 Host Status (TWIn.MSTATUS) register

37.3.2.2.4 Transmitting Data Packets

Assuming the M1 case above, the TWI host can start transmitting data by writing to the Host Data (TWIn.MDATA) register, which also clears the Write Interrupt Flag (WIF). The host continuously monitors the bus for collisions and errors during the data transfer. After completing the data packet transfer, the WIF flag will be set to '1'.

If the transmission is successful and the host receives an ACK bit from the client, the Received Acknowledge (RXACK) flag will be set to '0', meaning that the client 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 Host Control B (TWIn.MCTRLB) register

If the transmission is successful and the host receives a NACK bit from the client, the RXACK flag will be set to '1', meaning that the client cannot 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 Host 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 host will lose the arbitration, the Arbitration Lost (ARBLOST) flag will be set to '1', and the bus state changes to Busy. An arbitration lost during the data packet transfer is treated the same way as the above M4 case.

The WIF, ARBLOST, BUSERR and RXACK flags are all located in the Host Status (TWIn.MSTATUS) register.

37.3.2.2.5 Receiving Data Packets

Assuming the M2 case above, the clock is released for one byte, allowing the client to put one byte of data on the bus. The host will receive one data byte from the client, 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 Host 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 execute successfully, as the arbitration can be lost during the transmission. If a collision is detected, the host loses the arbitration, the Arbitration Lost (ARBLOST) flag is set to '1', and the bus state changes to Busy. The Host 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 while transferring the data packet is treated as the above M4 case.

The RIF, CLKHOLD, ARBLOST and WIF flags are all located in the Host Status (TWIn.MSTATUS) register.

Note: The RIF and WIF flags are mutually exclusive and cannot be set simultaneously.

37.3.2.3 TWI Client Operation

The TWI client is byte-oriented with optional interrupts after each byte. There are separate interrupt flags for the client data and address/Stop recognition. Interrupt flags can also be used for polled operations. Dedicated status flags indicate ACK/NACK received, clock hold, collision, bus error, and R/$\overline{W}$ direction.

When an interrupt flag is set to '1', the SCL is forced low, giving the client 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 automatically handling most conditions.

The Address Recognition Mode (PMEN) bit in the Client Control A (TWIn.SCTRLA) register can be configured to allow the client to respond to all received addresses.

37.3.2.3.1 Receiving Address Packets

When the TWI is configured as a client, 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 client will ACK a correct address and store the address in the Client Data (TWIn.SDATA) register. If the received address is not a match, the client will not acknowledge or save the address but wait for a new Start condition.

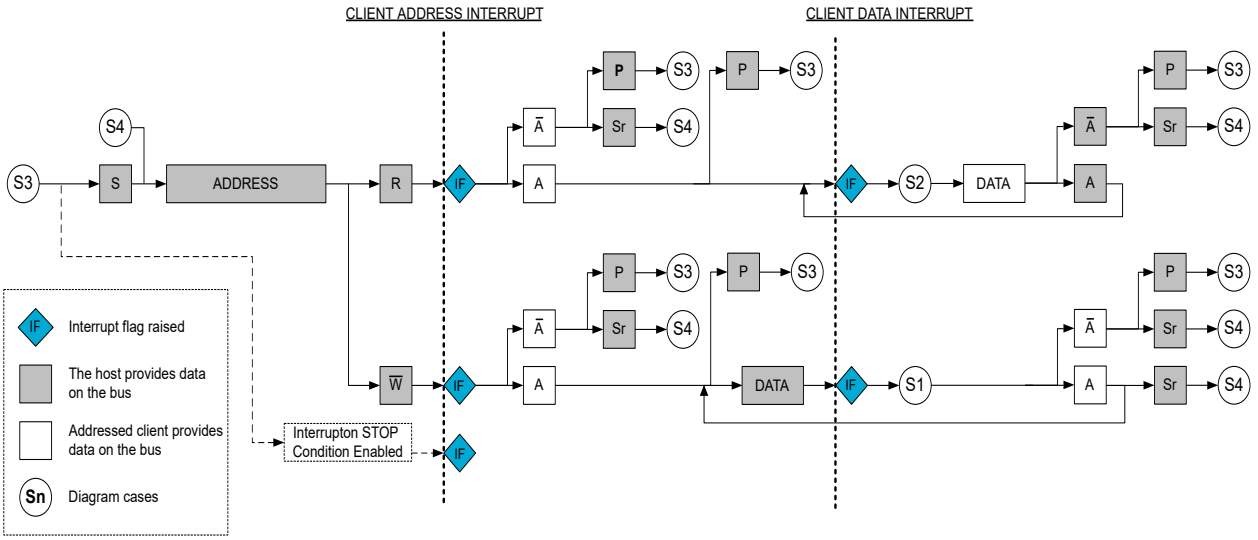
The Address or Stop Interrupt Flag (APIF) in the Client Status (TWIn.SSTATUS) register is set to '1' when a Start condition is followed by:

- A valid address matches the address stored in the Address (ADDR[7:1]) bit field in the Client Address (TWIn.SADDR) register
- The General Call Address (0x00) and the Address (ADDR[0]) bit in the Client Address (TWIn.SADDR) register is set to '1'
- A valid address matches the secondary address stored in the Address Mask (ADDRMASK) bit field, and the Address Mask Enable (ADDREN) bit is set to '1' in the Client Address Mask (TWIn.SADDRMASK) register
- Any address if the Address Recognition Mode (PMEN) bit in the Client 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 Client Status (TWIn.SSTATUS) register is set.

Depending on the Read/Write Direction (DIR) bit in the Client Status (TWIn.SSTATUS) register and the bus condition, one of four cases (S1 to S4) arises after the reception of the address packet.

Figure 37-6. TWI Client Operation



Case S1: Address Packet Accepted - Direction Bit Set to '0'

If an ACK is sent by the client after the address packet is received, and the Read/Write Direction (DIR) bit in the Client Status (TWIn.SSTATUS) register is set to '0', the host indicates a write operation.

The clock hold is active at this point, forcing the SCL low and stretching 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 host

Case S2: Address Packet Accepted - Direction Bit Set to '1'

If an ACK is sent by the client after the address packet is received, and the DIR bit is set to '1', the host indicates a read operation, and the Data Interrupt Flag (DIF) in the Client 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 host

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 Client Status (TWIn.SSTATUS) register.

The software can prepare to:

- Wait until a new address packet has been addressed to it

Case S4: Collision

If the client cannot send a high-level data bit or a NACK, the Collision (COLL) bit in the Client Status (TWIn.SSTATUS) register is set to '1'. The client will commence ordinary operation, except no low values will be shifted out on the SDA. The data and acknowledge output from the client 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.

37.3.2.3.2 Receiving Data Packets

Assuming the S1 case above, the client must be ready to receive data. When a data packet is received, the Data Interrupt Flag (DIF) in the Client Status (TWIn.SSTATUS) register is set to '1'.
The action selected by the Acknowledge Action (ACKACT) bit in the Client 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 client is ready to receive more data
- Respond with a NACK by writing '1' to the ACKACT bit, indicating that the client cannot receive any more data and the host must issue a Stop or repeated Start condition

37.3.2.3.3 Transmitting Data Packets

Assuming the S2 case above, the client can start transmitting data by writing to the Client Data (TWIn.SDATA) register. When a data packet transmission is completed, the Data Interrupt Flag (DIF) in the Client Status (TWIn.SSTATUS) register is set to '1'.

The software can prepare to take one of the following actions:

- Check if the host responded with an ACK by reading the Received Acknowledge (RXACK) bit from the Client Status (TWIn.SSTATUS) register, and start transmitting new data packets
- Check if the host responded with a NACK by reading the RXACK bit and stop transmitting data packets. The host must send a Stop or repeated Start condition after the NACK.

37.3.3 Additional Features

37.3.3.1 SMBus

The Inactive Bus Time-Out (TIMEOUT) bit field from the Host Control A (TWIn.MCTRLA) register must be configured if the TWI is used in an SMBus environment. It is necessary to configure the TIMEBASE bit field in the CLKCTRL.TIMEBASE register to comply with the SMBus standard. Refer to the CLKCTRL - Clock Controller section for more information.

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.

37.3.3.1.1 Compliance to SMBus Specifications

Hardware-Specific Restrictions

Section 2 of the SMBus 2.0 specifications states that powered-down devices must provide no leakage path to ground. There are ESD diodes placed between V_{DD} and the pads used for SCL and SDA on this device. Assuming V_{DD} is equivalent to ground when powered down, these ESD diodes provide a path to ground.

Implementation in Software

The following elements of the SMBus 2.0 specifications are not implemented in hardware:

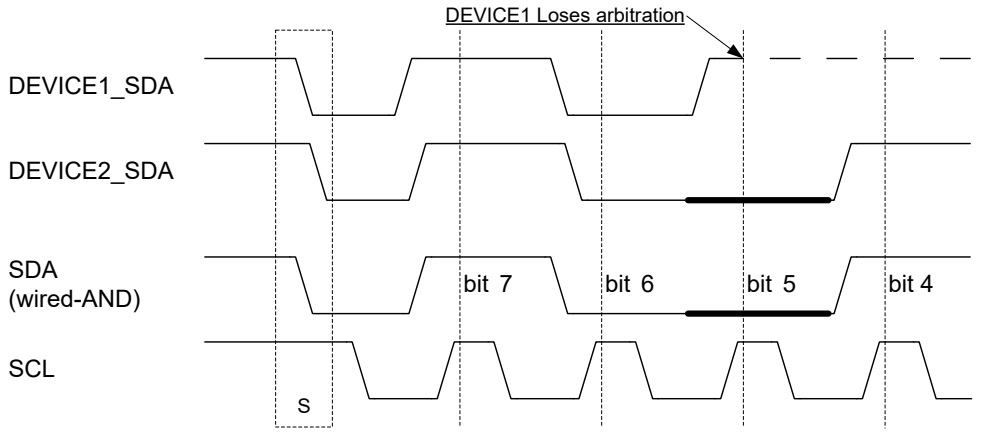
- Table 1 of the SMBus 2.0 specifications gives a maximum clock low timeout ($T_{timeout}$) of 25-35 ms, which can be implemented by connecting the SCL pin to the TCB peripheral using the Event System. Configure the TCB in Time-Out Check mode with the desired timeout value.
- Layer 3 (network layer) features such as packet error checking (PEC), address resolution protocol (ARP). These can be implemented in software if required.

37.3.3.2 Multi-Host

A host can start a bus transaction only if it has detected that the bus is in the Idle state. If multiple hosts are on the bus, other devices may try to initiate a transaction simultaneously, resulting in multiple hosts owning the bus. The TWI solves this problem by using an arbitration scheme where the host 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 Host Status (TWIn.MSTATUS) register will change to Busy. The hosts that lose the arbitration must wait until the bus becomes Idle before attempting to reacquire 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 37-7. TWI Arbitration



37.3.3.3 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 host, the Smart mode will automatically send the ACK action as soon as the Host Data (TWIn.MDATA) register is read. This feature is only active when the Acknowledge Action (ACKACT) bit in the Host Control B (TWIn.MCTRLB) register is set to ACK. The TWI host will not generate a NACK after the MDATA register is read if the ACKACT bit is set to NACK. This feature is enabled when the Smart Mode Enable (SMEN) bit in the Host Control A (TWIn.MCTRLA) register is set to '1'.

For the TWI client, the Smart mode will automatically send the ACK action as soon as the Client Data (TWIn.SDATA) register is read. The Smart mode will automatically set the Data Interrupt Flag (DIF) to '0' in the Client 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 Client Control A (TWIn.SCTRLA) register is set to '1'.

37.3.3.4 Dual Mode

The TWI supports Dual mode operation where the host and the client will operate simultaneously and independently. In this case, the Control A (TWIn.CTRLA) register will configure the TWI host, and the Dual Mode Control (TWIn.DUALCTRL) register will configure the TWI client. See the Initialization section for more details about the host configuration.

If used, the following bits must be configured before enabling the TWI Dual mode:

- The SDA Hold Time (SDAHOLD) bit field in the DUALCTRL register
- 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.

37.3.3.5 Quick Command Mode

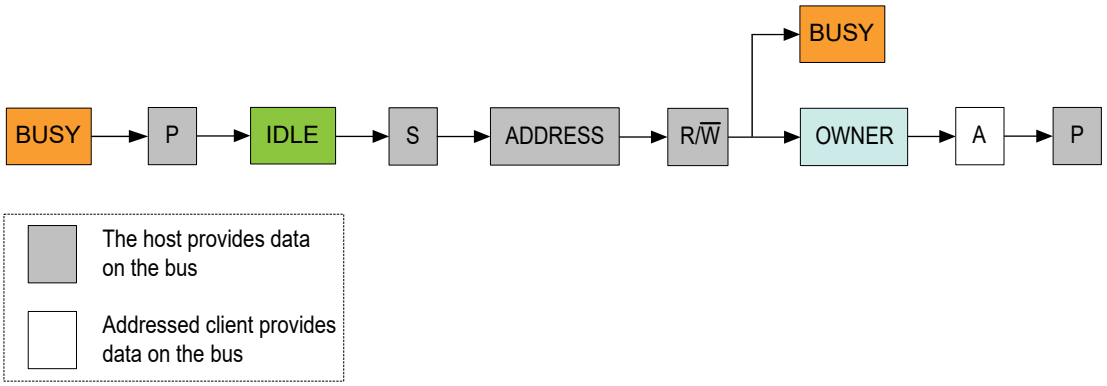
In Quick Command mode, the R/$\overline{W}$ bit from the address packet denotes the command. This mode is enabled by writing '1' to the Quick Command Enable (QCEN) bit in the Host Control A (TWIn.MCTRLA) register. There are no data sent or received.

The Quick Command mode is SMBus specific, using the R/$\overline{W}$ bit to turn a device function on/off or enable/disable a low-power Standby mode. This mode can be enabled to auto-trigger operations and reduce software complexity.

After the host receives an ACK from the client, 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 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 Host 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 Host Status (TWIn.MSTATUS) register.

Figure 37-8. Quick Command Frame Format



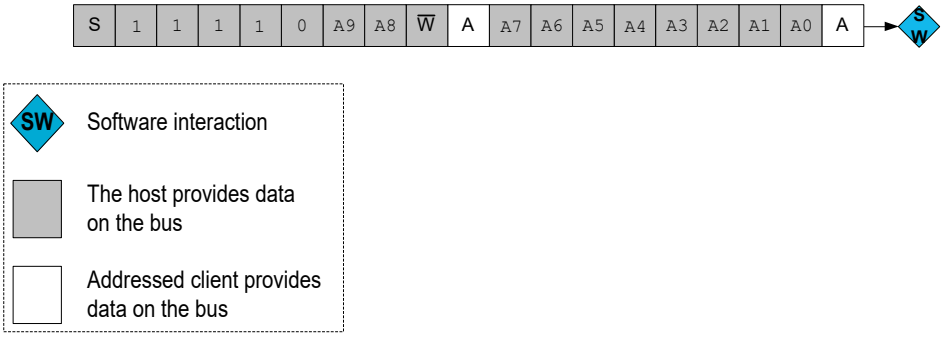
37.3.3.6 10-Bit Address

Regardless of whether the transaction is a read or write, the host must start by sending the 10-bit address with the R/$\overline{W}$ direction bit set to '0'.

The client address match logic supports recognition of 7-bit addresses and General Call Address. The Client Address (TWIn.SADDR) register is used by the client address match logic to determine if a host device has addressed the TWI client.

The TWI client address match logic only supports the 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 Client Address (TWIn.SADDR) register are 0b11110. Thus, the first byte will consist of five indication bits, the two Most Significant bits (MSbs) of the 10-bits address, and the R/$\overline{W}$ direction bit. The Least Significant Byte (LSB) of the address that follows from the host will come in the form of a data packet.

Figure 37-9. 10-Bit Address Transmission



37.3.4 Interrupts

Table 37-1. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
Client	TWI Client interrupt	• DIF: Data Interrupt Flag in TWIn.SSTATUS is set to '1' • APIF: Address or Stop Interrupt Flag in TWIn.SSTATUS is set to '1'
Host	TWI Host interrupt	• RIF: Read Interrupt Flag in TWIn.MSTATUS is set to '1' • WIF: Write Interrupt Flag in TWIn.MSTATUS is set to '1'

When an interrupt condition occurs, the corresponding interrupt flag is set in the Host Status (TWIn.MSTATUS) register or the Client 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.

37.3.5 Sleep Mode Operation

The bus state logic and the address recognition hardware continue to operate in all sleep modes. If the TWI client is in a sleep mode and a Start condition followed by the client address is detected, clock stretching is active during the wake-up period until the main clock is available. The TWI host will stop operation in all sleep modes. When the Dual mode is active, the TWI peripheral will wake up only when the Start condition is received by the TWI client.

37.3.6 Debug Operation

During run-time debugging, the TWI will continue its ordinary operation. Halting the CPU in Debugging mode will stop the normal operation of the TWI. The TWI can be forced to operate with a 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 Host Data (TWIn.MDATA) register or the Client 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.

37.4 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

37.5 Register Description

37.5.1 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 selects 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
This bit controls the number of cycles the SCL is stretched to ensure sufficient setup time on the SDA out signal. This bit is used when operating in client mode.

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 Fast-mode Plus Enable
Writing a '1' to this bit selects the 1 MHz bus speed for the TWI in default configuration or the TWI host in Dual mode configuration.

Value	Name	Description
0	OFF	Operating in Standard mode or Fast mode
1	ON	Operating in Fast mode Plus

37.5.2 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 selects between I^2C and SMBUS. This bit is ignored if the Dual mode is not enabled.

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 client. See also the *Electrical Characteristics* section. 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 client. This bit is ignored if the Dual mode is not enabled.

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.

37.5.3 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
Refer to the *Debug Operation* section for 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

37.5.4 Host 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 host read interrupt will only be generated 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 Host Status (TWIn.MSTATUS) register. The RIF flag is set to '1' when the host read interrupt occurs.

Bit 6 – WIEN Write Interrupt Enable
A TWI host write interrupt will only be generated 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 Host Status (TWIn.MSTATUS) register. The WIF flag is set to '1' when the host write interrupt occurs.

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 client acknowledges the address, the corresponding Read Interrupt Flag (RIF) or Write Interrupt Flag (WIF) will be set depending on the value of the R/$\overline{W}$ bit.
The software must issue a Stop command by writing to the Command (MCMD) bit field in the Host Control B (TWIn.MCTRLB) register.

Bits 3:2 – TIMEOUT[1:0] Inactive Bus Time-Out
Setting this bit field to a non-zero 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
0x2	100US	100 µs
0x3	200US	200 µs

Bit 1 – SMEN Smart Mode Enable
Writing a '1' to this bit enables the Host Smart mode. When the Smart mode is enabled, the existing value in the Acknowledge Action (ACKACT) bit from the Host Control B (TWIn.MCTRLB) register is sent immediately after reading the Host Data (TWIn.MDATA) register.

Bit 0 – ENABLE Enable TWI Host
Writing a '1' to this bit enables the TWI as host.

37.5.5 Host 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 host and the bus states changes to Idle. The TWI will transmit invalid data if the Host Data (TWIn.MDATA) register is written before the Host Address (TWIn.MADDR) register. Writing to Host Address (TWIn.MADDR) and Host Data (TWIn.MDATA) after a Flush will cause a transaction to start as soon as hardware detects SCL bus free.
Writing a '1' to this bit generates a strobe for one clock cycle, disabling the host and then re-enabling the host. Writing a '0' to this bit has no effect.

Bit 2 – ACKACT Acknowledge Action

The ACKACT[1] bit represents the behavior in the Host mode under certain conditions defined by the bus state and the software interaction. If the Smart Mode Enable (SMEN) bit in the Host Control A (TWIn.MCTRLA) register is set to '1', the acknowledge action is performed when the Host Data (TWIn.MDATA) register is read. Otherwise a command must be written to the Command (MCDM) bit field in the Host Control B (TWIn.MCTRLB) register.
The acknowledge action is not performed when the Host Data (TWIn.MDATA) register is written since the host 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 host operation, as defined by the table below.

Table 37-2. 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 simultaneously.
2. For a host write operation, the TWI will wait for new data to be written to the Host Data (TWIn.MDATA) register.

37.5.6 Host 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 host byte read operation is completed.
The RIF flag can generate a host read interrupt. Find more information in the description of the Read Interrupt Enable (RIEN) bit in the Host Control A (TWIn.MCTRLA) register.
This flag automatically clears when some TWI registers are accessed. Any of the following methods can be used to clear the RIF flag:

1. Writing a '1' to it.
2. Writing to the Host Address (TWIn.MADDR) register.
3. Writing/Reading the Host Data (TWIn.MDATA) register.
4. Writing to the Command (MCMD) bit field from the Host Control B (TWIn.MCTRLB) register.

Bit 6 – WIF Write Interrupt Flag
This flag is set to '1' when a host address transmit or byte write operation is completed, regardless of any occurrence of a bus error or arbitration lost condition.
The WIF flag can generate a host write interrupt. Find more information in the description of the Write Interrupt Enable (WIEN) bit in the Host Control A (TWIn.MCTRLA) register.
This flag can be cleared using any of the methods described above for the RIF flag.

Bit 5 – CLKHOLD Clock Hold
When this bit is read as '1', it indicates that the host currently holds the SCL low, stretching the TWI clock period.
This bit can be cleared using any of the methods described above 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 client was ACK, and the client is ready for more data.
When this flag is read as '1', it indicates that the most recent Acknowledge bit from the client was NACK, and the client 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 host 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 Host Address (TWIn.MADDR) register.

The TWI bus error detector is part of the TWI host circuitry. For bus errors to be detected, the TWI host 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. Writing 0x1 to this bit field will force the bus state to IDLE. All other values will be ignored.

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

37.5.7 Host 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 host is disabled. The host can be disabled by writing '0' to the Enable TWI Host (ENABLE) bit from the Host Control A (TWIn.MCTRLA) register.
Refer to the *Clock Generation* section for more information on how to calculate the frequency of the SCL.

37.5.8 Host 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 client 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 read access does not trigger the host logic to perform any bus protocol-related operations.
The host control logic uses bit 0 of this register as the R/$\overline{W}$ direction bit.

37.5.9 Host 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 host'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 to the MDATA register will command the host to perform a byte transmit operation on the bus, directly followed by receiving the Acknowledge bit from the client. This is independent of the Acknowledge Action (ACKACT) bit from the Host 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 Host Control A (TWIn.MCTRLA) register is set to '1', read access to the MDATA register will command the host to perform an acknowledge action. This is dependent on the setting of the Acknowledge Action (ACKACT) bit from the Host Control B (TWIn.MCTRLB) register.

Notes:

1. The WIF and RIF flags are automatically cleared 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 Host Status (TWIn.MSTATUS) register.

37.5.10 Client 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 Client Status (TWIn.SSTATUS) register.
A TWI client data interrupt will only be generated if this bit, the DIF flag, and the Global Interrupt Enable (I) bit in the 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 Client Status (TWIn.SSTATUS) register.
A TWI client address or stop interrupt will only be generated 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 client stop interrupt shares the interrupt flag and vector with the client address interrupt.
2. The Stop Interrupt Enable (PIEN) bit in the Client 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 Client 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 Client Status (TWIn.SSTATUS) register to be set when a Stop condition occurs. The main clock frequency must be at least four times the SCL frequency to use this feature.

Bit 2 – PMEN Address Recognition Mode
If this bit is written to '1', the client address match logic responds to all received addresses.
If this bit is written to '0', the address match logic uses the Client Address (TWIn.SADDR) register to determine which address to recognize as the client's address.

Bit 1 – SMEN Smart Mode Enable
Writing this bit to '1' enables the client Smart mode. When the Smart mode is enabled, issuing a command by writing to the Command (SCMD) bit field in the Client Control B (TWIn.SCTRLB) register or accessing the Client Data (TWIn.SDATA) register resets the interrupt, and the operation continues. If the Smart mode is disabled, the client always waits for a new client command before continuing.

Bit 0 – ENABLE Enable TWI Client
Writing this bit to '1' enables the TWI client.

37.5.11 Client 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 TWI client under certain conditions defined by the bus protocol state and the software interaction. If the Smart Mode Enable (SMEN) bit in the Client Control A (TWIn.SCTRLA) register is set to '1', the acknowledge action is performed when the Client Data (TWIn.SDATA) register is read. Otherwise a command must be written to the Command (SCMD) bit field in the Client Control B (TWIn.SCTRLB) register.
The acknowledge action is not performed when the Client Data (TWIn.SDATA) register is written since the client 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 client operation as defined by the table below.

Table 37-3. Command Settings

Value	Name	DIR	Description	
0x0	NOACT	X	No action	
0x1	—	X	Reserved	
0x2	COMPTRANS	$\overline{W}$	Execute Acknowledge Action succeeded by waiting for any Start (S/Sr) condition	Used to complete a transaction
		R	Wait for any Start (S/Sr) condition	
0x3	RESPONSE	$\overline{W}$	Execute Acknowledge Action succeeded by the reception of the next byte	
		R	Used in response to an address interrupt (APIF): Execute Acknowledge Action succeeded by client data interrupt.	
			Used in response to a data interrupt (DIF): Execute a byte read operation followed by Acknowledge Action.	

Note: 1. The ACKACT bit and the SCMD bit field can be written simultaneously. The ACKACT will be updated before the command is triggered.

37.5.12 Client 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 client byte transmit or receive operation is completed without bus errors. This flag can be set to '1' with an unsuccessful transaction in case of collision detection. Find more information in the description of the Collision (COLL) bit.
The DIF flag can generate a client data interrupt. Find more information in the description of the Data Interrupt Enable (DIEN) bit in the Client Control A (TWIn.SCTRLA) register.
This flag automatically clears when some TWI registers are accessed. Any of the following methods can be used to clear the DIF flag:

1. Writing/Reading the Client Data (TWIn.SDATA) register.
2. Writing to the Command (SCMD) bit field in the Client Control B (TWIn.SCTRLB) register.

Bit 6 – APIF Address or Stop Interrupt Flag
This flag is set to '1' when the client address has been received or by a Stop condition.
The APIF flag can generate a client address or stop interrupt. Find more information in the description of the Address or Stop Interrupt Enable (APIEN) bit in the Client Control A (TWIn.SCTRLA) register.
This flag can be cleared using any of the methods described for the DIF flag.

Bit 5 – CLKHOLD Clock Hold
When this bit is read as '1', it indicates that the client 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 host was ACK.
When this flag is read as '1', it indicates that the most recent Acknowledge bit from the host was NACK.

Bit 3 – COLL Collision
When this bit is read as '1', it indicates that the client 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 because of the internal completion of an unsuccessful transaction.
2. Transmit the NACK bit. The collision occurs because the client 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 host circuitry. For the bus errors to be detected by the client, the TWI Dual mode or the TWI host 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 host 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 host TWI device.

When this bit is read as '1', it indicates that a host read operation is in progress.

When this bit is read as '0', it indicates that a host write operation is in progress.

Bit 0 – AP Address or Stop

When the TWI client Address or Stop Interrupt Flag (APIF) is set to '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

37.5.13 Client 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 Client Address (TWIn.SADDR) register is used by the client address match logic to determine if a host device has addressed the TWI client. If an address packet is received, the Address or Stop Interrupt Flag (APIF) and the Address or Stop (AP) bit in the Client Status (TWIn.SSTATUS) register are set to '1'.

The upper seven bits (ADDR[7:1]) of the TWIn.SADDR register represent the main client address. The TWIn.SADDR register's Least Significant bit (ADDR[0]) is used for recognition of the General Call Address (0x00) of the I^2C protocol. This feature is enabled when this bit is set to '1'.

37.5.14 Client 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 client data register.
Reading valid data or writing data to be transmitted can only be achieved when the SCL is held low by the client (i.e., when the client CLKHOLD bit is set to '1'). It is unnecessary to check the Clock Hold (CLKHOLD) bit from the Client 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 Client Control A (TWIn.SCTRLA) register is set to '1', read access to the SDATA register, when the clock hold is active, auto-triggers bus operations and commands the client to perform an acknowledge action. This is dependent on the setting of the Acknowledge Action (ACKACT) bit from the Client Control B (TWIn.SCTRLB) register.

37.5.15 Client 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 Client Address mask. Each of the bits in the Client Address Mask (TWIn.SADDRMASK) register can mask (disable) the corresponding address bits in the TWI Client 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 Client Address (TWIn.SADDR) register. In other words, masked bits will always match, making it possible to recognize the ranges of addresses.
If the ADDREN bit is written to '1', the Client Address Mask (TWIn.SADDRMASK) register can be loaded with a second client address in addition to the Client Address (TWIn.SADDR) register. In this mode, the client will have two unique addresses -- one in the Client Address (TWIn.SADDR) register and the other in the Client 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 client address match logic responds to the two unique addresses in the client TWIn.SADDR and TWIn.SADDRMASK registers.

38. CRCSCAN - Cyclic Redundancy Check Memory Scan

38.1 Features

- CRC-16-CCITT or CRC-32 (IEEE 802.3)
- Scan of the boot section, application code section, or application data section
- Scan of the Boot ROM during system start-up
- User-configurable scan of the boot section of the flash during system start-up
- Scanning is performed when the CPU is in IDLE sleep mode only
- Optional periodic interrupt after every 32 words scanned
- Optional interrupt when scanning is completed
- Selectable Non-Maskable Interrupt (NMI) Trigger on Failure
- Manual mode with application-supplied input data
- Error injection to test functionality

38.2 Overview

A Cyclic Redundancy Check (CRC) takes a data stream of bytes from the Flash (either the boot section, application code section, or application data section) and generates a checksum. The CRC peripheral (CRCSCAN) can be used to detect errors in the program memory.

The last location in the section being checked must contain a correct pre-calculated 16- or 32-bit checksum for comparison. If the checksum calculated by the CRCSCAN and the pre-calculated checksums match, a Status bit is set. If they do not match, the Status A (CRCSCAN.STATUSA) register will indicate that it failed. The CRCSCAN can be configured to generate a Non-Maskable Interrupt (NMI) if there is a checksum mismatch.

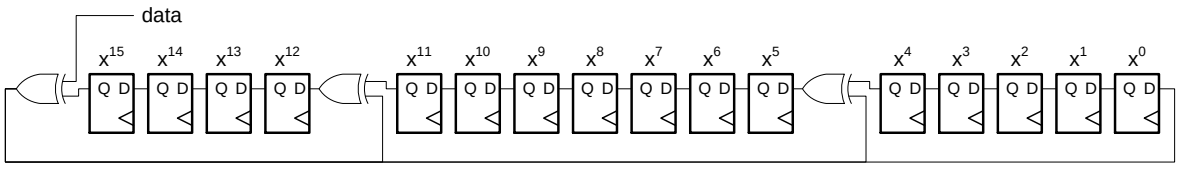
The CRC generator supports CRC-16-CCITT and CRC-32 (IEEE 802.3).

Polynomial:

- CRC-16-CCITT: $x^{16} + x^{12} + x^5 + 1$, initial value = `0xFFFF`
- CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$, initial value = `0xFFFF_FFFF`

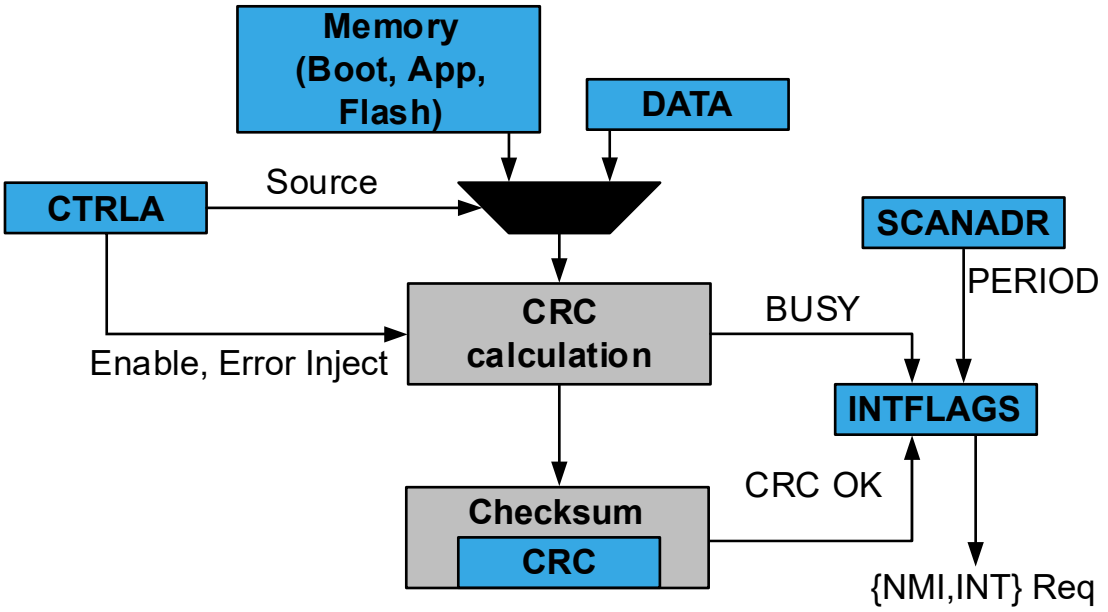
The CRC reads byte-by-byte the content of the section(s) it is set up to check, starting with byte 0, and generates a new checksum for each byte. The byte is sent through a shift register, as depicted below, starting with the most significant bit. The CRC passes if the last bytes in the section contain the correct checksum. See the *Checksum* section for instruction on how to place the checksum.

Figure 38-1. CRC Implementation Description



38.2.1 Block Diagram

Figure 38-2. Cyclic Redundancy Check Block Diagram



38.3 Functional Description

38.3.1 Initialization

To enable a CRC scan in software (or via the debugger):

Sleep Mode

1. Select between CRC32 and CRC16 modes with the CRC Mode Select (CRCSEL) bit in the Control A (CRCSCAN.CTRLA) register.
2. Write the CRC Source (SRC) bit field in the CRCSCAN.CTRLA register to select the desired region of the Flash to be scanned.
3. Enable the CRCSCAN by writing a '1' to the ENABLE bit in the CRCSCAN.CTRLA register. The CRC Busy (BUSY) bit in the Status A (CRCSCAN.STATUSA) register will be set.
4. The CRC will begin scanning when the CPU enters the Idle sleep mode.

Note: If sectioning the Flash and using interrupt-driven CRCSCAN, the interrupt vector table may need to be moved from its default location at the start of the Application Code (APPCODE) section of the Flash. For information on changing the expected location of the vector table, refer to the *CPU Interrupt Controller* section.

Manual Mode

1. Select CRC32 or CRC16 mode with the CRCSEL bit in the CRCSCAN.CTRLA register.
2. Write the SRC bit field in the CRCSCAN.CTRLA register to MANUAL.
3. Enable the CRCSCAN by writing a '1' to the ENABLE bit in the CRCSCAN.CTRLA register.
4. Write the byte to the Input Data (CRCSCAN.DATA) register for the CRC calculation.
5. Read the CRC result from the CRC Result (CRCSCAN.CRC) register.

38.3.2 Operation

38.3.2.1 Operation Modes

The CRCSCAN peripheral supports Scan-In-Sleep mode and the Manual mode operation. The mode is set in the CRC Source (SRC) bit field of the Control A (CRCSCAN.CTRLA) register. For Idle sleep mode operation, configure SRC with the desired section of the Flash to scan (BOOT, CODE or DATA). For the Manual mode operation, set SRC to MANUAL.

Note: The CRCSCAN can be set up to scan the BOOT section in Flash before the device exits Reset. If this scan fails, the CPU is not allowed to start normal code execution. This feature is enabled and controlled by the CRC Boot (CRCBOOT) and CRC Select (CRCSEL) bits in the Fuse System Configuration 0 (FUSE.SYSCFG0) fuse. Refer to the *Fuse* section for more information.

Scan-In-Sleep Mode

In Scan-In-Sleep mode, the CRCSCAN peripheral has access to the Flash only when the CPU is in Idle sleep mode. If a scan is enabled, the scan will start when the CPU enters Idle sleep mode, and the scan will pause when the CPU wakes up. If the CPU goes to sleep again, the scan will resume at the address where it left off. This operation will continue until the entire selected memory range has been scanned. An interrupt can optionally be triggered when the scan is complete.

The CRCSCAN peripheral has a built-in counter that, when enabled, will set an interrupt flag every 32 scanned words. This counter is enabled by writing to the Enable Periodic Timer (PEREN) bit in the CRCSCAN.CTRLA register. Its interrupt can be enabled by writing to the Scan Period Done Interrupt Enable (PERIOD) bit in the Interrupt Control (CRCSCAN.INTCTRL) register. If enabled, the interrupt will cause the CPU to leave Idle sleep and allow the application to do various housekeeping tasks between scanning. Continue scanning by clearing the Scan Period Done (PERIOD) bit in the Interrupt Flags (CRCSCAN.INTFLAGS) register and re-enter the Idle sleep mode.

Other sources can wake up the CPU even when the Period Interrupt is enabled. In this case, the periodic timer is frozen while the CPU is awake, and will continue where it froze after re-entering Idle sleep mode.

If the periodic timer is enabled without the period interrupt enabled, the CRC will pause scanning of more Flash words when the flag is set without waking up the CPU. Upon waking up from a different source clearing the PERIOD flag in INTFLAGS will cause the scan to continue.

A CRC error detected during the scan will set an NMI request if the Enable NMI Trigger (NMIEN) bit in the CRCSCAN.CTRLA register is '1'.

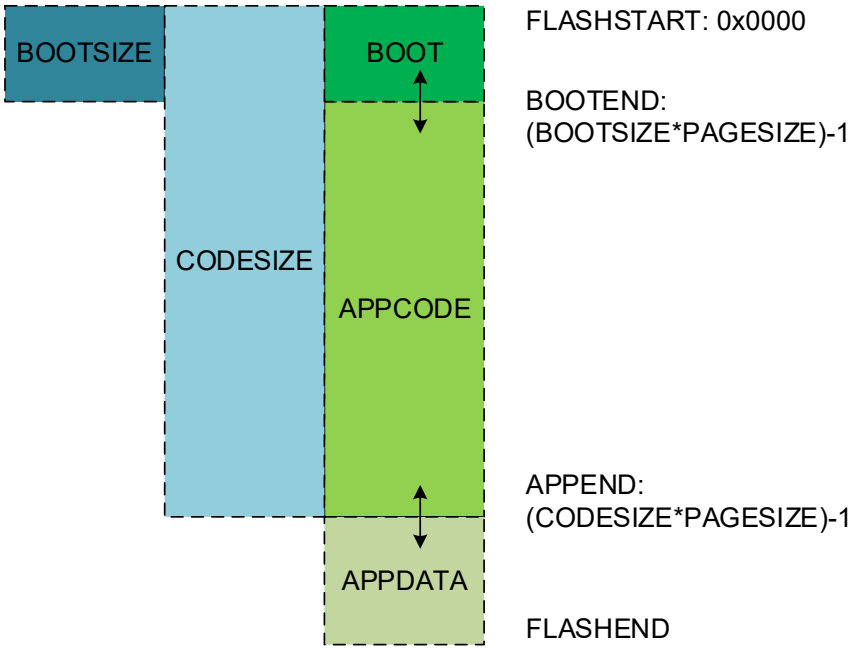
Manual Mode

The Manual mode allows the application to calculate CRC on a data stream by writing the bytes in the stream to the Input Data (CRCSCAN.DATA) register. CRC-16 or CRC-32 is used in the MANUAL mode, depending on the value of the CRC Source Select (CRCSEL) bit in the CRCSCAN.CTRLA register. One clock cycle is required to calculate CRC on a written byte, so data can be written continuously to the CRCSCAN.DATA register without polling the CRC Busy (BUSY) bit in the Status A (CRCSCAN.STATUSA) register between writes. The result of the CRC operation can be read from the CRC Result (CRCSCAN.CRC) register.

38.3.2.2 Configuration of Scan Region

The Flash can be divided into sections, as illustrated in the figure below.

Figure 38-3. Division of Flash Into Sections



Determine the section sizes in blocks corresponding to the Flash memory page size. The sizes of these sections are set by the Boot Size (FUSE.BOOTSIZE) and Code Size (FUSE.CODESIZE) fuses. Refer to the *Fuse* section for more information. The Boot Loader Code (BOOT) section stretches from FLASHSTART to BOOTEND, the Application Code (APPCODE) section stretches from BOOTEND to APPEND, and the remaining is the Application Data (APPDATA) section.

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

Which region to scan is configured in the CRC Source (SRC) bit field in the CRCSCAN.CTRLA register.

Starting a scan of an unconfigured Flash region will set the ERROR bit in the CRCSCAN.STATUSA register. For starting a scan with SRC set to CODE or DATA when BOOTSIZE='0' (the whole Flash is BOOT) or SRC set to DATA when CODESIZE='0' (no DATA section) will both result in CRC failure.

Table 38-1. Configuration of Scan Regions

CRC Source	Scan Region	BOOTSIZE	CODESIZE
BOOT	BOOT	> 0	-
	The entire Flash	0	-
CODE	APPCODE	> 0	-
DATA	APPDATA	> 0	BOOTEND + APPEND < FLASHEND

38.3.2.3 Checksum

The pre-calculated checksum must be present at the end of the section to be scanned. The checksum must be stored in the last bytes of the BOOT section to check the BOOT section and similarly for the APPCODE and APPDATA sections. The checksum for CRC32 has opposite endianness compared to CRC16. *Table 1-2* and *Table 1-3* show explicitly how to store the checksum for the different sections. For additional information on partitioning the Flash section. Refer to the *Configuration of Scan Region* section.

Table 38-2. Placement of the Pre-Calculated Checksum for CRC16 in Flash

Section to Check	CHECKSUM[15:8]	CHECKSUM[7:0]
BOOT	BOOTEND - 1	BOOTEND
APPCODE	APPEND - 1	APPEND
APPDATA	FLASHEND - 1	FLASHEND

Table 38-3. Placement of the Pre-Calculated Checksum for CRC32 in Flash

Section to Check	CHECKSUM[7:0]	CHECKSUM[15:8]	CHECKSUM[23:16]	CHECKSUM[31:24]
BOOT	BOOTEND - 3	BOOTEND - 2	BOOTEND - 1	BOOTEND
APPCODE	APPEND - 3	APPEND - 2	APPEND - 1	APPEND
APPDATA	FLASHEND - 3	FLASHEND - 2	FLASHEND - 1	FLASHEND

Defining the Pre-Calculated Checksum

The pre-calculated checksum can be determined by using a CRC algorithm and the parameters specified in *Table 1-4*. The calculation should start at the region start and end at the address before the pre-calculated checksum's placement. Manual mode can be used to compare the checksum at the end of a region with known data, ensuring correctly configured calculation.

Table 38-4. Parameters for Checksum Calculation

CRC Selected	Polynomial	Initial Value	Width	XOR Value
CRC16	0x1021	0xFFFF	2	N/A
CRC32	0x4C11DB7	0xFFFF_FFFF	4	0xFFFF_FFFF

For more information on Checksum calculation and use, refer to the *Hexmate* section in the *MPLAB® XC8 C Compiler User's Guide for PIC® MCU* document.

38.3.2.4 Error Injection

The CRCSCAN error injection triggers a rapid CRC scan, resulting in a CRC error.

Writing a '1' to the Inject CRC Error (EINJ) bit in the Control B (CRCSCAN.CTRLB) register will trigger a scan that starts when the CPU enters Idle sleep mode.

No error injection will be performed if the BUSY bit in the Status A (CRCSCAN.STATUSA) register is '1' or if CRCSCAN is in the Manual mode (determined by the CRC Source (SRC) bit field in the Control A (CRCSCAN.CTRLA) register).

The error-injected scan is performed by only scanning the last four words in the section selected by the SRC bit in the CRCSCAN.CTRLA register and comparing the CRC with the pre-calculated checksum described in the *Checksum* section. As the checksums will differ, the ERROR bit in the CRCSCAN.STATUSA register will be set to '1' at the end of the scan.

The status flags in the Interrupt Flags (CRCSCAN.INTFLAGS) register are updated (as after an ordinary scan), and an interrupt is generated if the DONE bit in the Interrupt Control (CRCSCAN.INTCTRL) register is set. Error injection when the Enable NMI Trigger (NMIEN) bit in the CRCSCAN.CTRLA register is '1' will trigger an NMI request.

Error injection does not start in sleep unless the Enable CRCSCAN (ENABLE) bit in the CRCSCAN.CTRLA register is '1'. Writing a '1' to the Reset CRCSCAN (RESET) bit in the CRCSCAN.CTRLA register will abort the error injection, just as it would for an ordinary scan.

38.3.2.5 Data Bus Accesses

Access attempts to unused Special Function Register (SFR) addresses within the CRCSCAN peripheral's address space will be discarded and return Bus Error to the CPU.

Attempted writes to read-only SFRs will be discarded and return Bus Error. A bus error received during the scan will cause the CRC Scan to fail, and a parity error on read data will cause the CRC Scan to fail.

38.3.3 Configuration Change Protection

This peripheral has registers under Configuration Change Protection (CCP), a security mechanism to avoid unintentional changes to the CRCSCAN settings. To write to these, first write a specific key to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to an SFR protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged and returns a Bus Error response on the data bus.

The following register is under CCP:

Table 38-5. CRCSCAN - Registers under Configuration Change Protection

Register	Key
CRCSCAN.CTRLB	IOREG

38.3.4 Interrupts

Table 38-6. Available Interrupt Vectors and Sources

Interrupt Vector Name	Interrupt Source Name	Description	Condition
NMI	ERROR	Non-Maskable Interrupt from a CRC Scan failure	ERROR in CRCSCAN.STATUSA and NMIEN in CRCSCAN.CTRLA is '1'
CRCSCAN	PERIOD	The scan period is done	PERIOD in CRCSCAN.INTFLAGS and PEREN in CRCSCAN.CTRLA is '1'
CRCSCAN	DONE	The scan of an entire Flash region is done	DONE in CRCSCAN.INTFLAGS is '1'

When the CRCSCAN interrupt condition occurs, the DONE or PERIOD flag in the Interrupt Flags (CRCSCAN.INTFLAGS) register is set to '1'. The flag is cleared by writing a '1' to the bit position.

Enable the Enable Periodic Timer (PEREN) bit in the Control A (CRCSCAN.CTRLA) register to get a PERIOD interrupt.

A Non-Maskable Interrupt (NMI) is enabled by writing a '1' to the Enable NMI Trigger (NMIEN) bit in the CTRLA register. An NMI is generated when the CRC Error (ERROR) bit in the Status A (CRCSCAN.STATUSA) register is '1', and the NMIEN bit in the CRCSCAN.CTRLA register is '1'. The NMI request remains active until a System Reset or by writing a '1' to the Reset CRCSCAN (RESET) bit in the CRCSCAN.CTRLA register.

38.3.5 Scan-Sleep Mode Operation

In Scan-Sleep mode, the CRCSCAN scans while the CPU is in Idle mode. For more information, see the *Operation* section.

38.3.6 Debug Operation

When the debugger sends a break command, the CRC will finish an ongoing scan before releasing the CPU. So, when stepping through a sequence that starts the CRC, it will finish the CRC during one step.

38.4 Functional Safety

The CRCSCAN peripheral is designed for use in Functional Safety systems. Primarily to check the Flash and boot ROM at boot time, but also to do periodic scans, in systems both with and without ECC, as CRC scan detects other errors than ECC. Error injection is present to test correct behavior.

38.5 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RESET	SRC[1:0]		PEREN	CRCSEL		NMIEN	ENABLE
0x01	CTRLB	7:0								EINJ
0x02	INTCTRL	7:0							PERIOD	DONE
0x03	INTFLAGS	7:0							PERIOD	DONE
0x04	STATUSA	7:0						OK	BUSY	ERROR
0x05	SCANADR	7:0	SCANADR[7:0]							
0x06	DATA	7:0	DATA[7:0]							
0x07	Reserved									
0x08	CRC	7:0	CRC[7:0]							
		15:8	CRC[15:8]							
		23:16	CRC[23:16]							
		31:24	CRC[31:24]							

38.6 Register Description

38.6.1 Control A

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

The bits in this register are not writable when the CRC is busy (the BUSY bit in the Interrupt Flags (CRCSCAN.INTFLAGS) register is '1'), except for the RESET bit, which is writable at any time.

Bit	7	6	5	4	3	2	1	0
	RESET	SRC[1:0]		PEREN	CRCSEL		NMIEN	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 – RESET Reset CRCSCAN
Writing this bit to '1' resets the CRCSCAN peripheral. The CRCSCAN Control and Status registers (CTRLA, INTFLAGS, INTCTRL, STATUSA) will be cleared one clock cycle after the RESET bit is written to '1'.
The RESET bit is a strobe bit.

Bits 6:5 – SRC[1:0] CRC Source
The SRC bit field selects which section of the Flash the CRCSCAN peripheral should check. To set up section sizes, refer to the fuse description.
The CRCSCAN can be enabled during internal Reset initialization to verify the Boot section before letting the CPU start (see the *Fuses* section).

Value	Name	Description
0x0	BOOT	The CRC is performed on the boot section of the Flash
0x1	CODE	The CRC is performed on the application code section of the Flash
0x2	DATA	The CRC is performed on the application data section of the Flash
0x3	MANUAL	Manual mode, CRC is performed on data written to the Data (DATA) register

Bit 4 – PEREN Enable Periodic Timer
This bit enables the periodic timer mechanism, which sets the Scan Period Done (PERIOD) flag in the Interrupt Flags (CRCSCAN.INTFLAGS) register every 32 scanned Flash words.

Value	Name	Description
0	DISABLE	The periodic timer is disabled
1	ENABLE	The periodic timer is enabled

Bit 3 – CRCSEL CRC Mode Select
This bit determines whether to use CRC32 or CRC16.

Value	Name	Description
0	CRC16	The CRC is performed using CRC16
1	CRC32	The CRC is performed using CRC32

Bit 1 – NMIEN Enable NMI Trigger
When this bit is written to '1', a CRC error during the scan will trigger an NMI.

Value	Name	Description
0	DISABLE	The CRC error during scan will not trigger a NMI
1	ENABLE	The CRC CRC error during scan will trigger a NMI

Bit 0 – ENABLE Enable CRCSCAN

Writing this bit to '1' enables the CRCSCAN peripheral with the current settings. The ENABLE bit is cleared by hardware after a scan when the DONE bit in the CRCSCAN.INTFLAGS register is set.
Writing this bit to '0' has no effect.
To see whether the CRCSCAN peripheral is busy with an ongoing scan, poll the BUSY bit in the Status A (CRCSCAN.STATUSA) register.

38.6.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: Configuration Change Protection

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

Bit 0 – EINJ Inject CRC Error
Writing this bit to '1' starts a short scan resulting in a CRC error. It is used to test the error system in a Functional Safety application.
This bit always reads as '0'.

38.6.3 Interrupt Control

Name: INTCTRL
Offset: 0x02
Reset: 0x00
Property: -

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

Bit 1 – PERIOD Scan Period Done Interrupt Enable
Writing this bit to '1' enables the interrupt. The Periodic Interrupt Timer (PEREN) bit in the Control A (CRCSCAN.CTRLA) register needs to be enabled for the timer to run, which eventually causes the corresponding interrupt flag to be set.

Bit 0 – DONE Scan Done Interrupt Enable
Writing this bit to '1' enables the interrupt.

38.6.4 Interrupt Flags

Name: INTFLAGS
Offset: 0x03
Reset: 0x00
Property: -

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

Bit 1 – PERIOD Scan Period Done

This bit is set when a scan period has expired, i.e., 32 Flash words have been scanned. CRC scan is halted when the PERIOD bit in the CRCSCAN.INTFLAGS register is set, regardless of the value of the PERIOD bit in the CRCSCAN.INTCTRL register. Clear the PERIOD flag and enter the IDLE sleep mode to resume the scan. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

Bit 0 – DONE Scan Done

This bit is set when the scan has completed, i.e., the entire region selected by SRC bit field in the Control A (CRCSCAN.CTRLA) register has been scanned. The result of the scan can be read from the ERROR bit in the CRCSCAN.STATUSA register. The flag is cleared by writing a '1' to the bit position. Writing a '0' to this bit has no effect.

The Boot ROM may be configured to execute a scan of the flash on start up in which case the application will read the flag as '1'. The DONE flag must then be cleared before starting a scan.

38.6.5 Status A

Name: STATUSA
Offset: 0x04
Reset: 0x04
Property: -

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

Bit 2 – OK CRC OK
When this bit is read as '1', the previous CRC has completed successfully.
This bit is only valid when the BUSY bit is read as '0'.

Bit 1 – BUSY CRC Busy
When this bit is read as '1', the CRCSCAN peripheral is busy. BUSY will remain '1' until completion of the entire scan, i.e., all addresses from the start to the end address have been scanned. As long as the module is busy, the access to the control registers is limited.

Bit 0 – ERROR CRC Error
When this bit is read as '1', the previous CRC completed with failure, `ERROR = (!BUSY & !OK)`.
This bit is connected as an NMI source and is set when a scan results in an error, issuing an NMI request if the NMIEN bit in the Control A (CRCSCAN.CTRLA) register is set to '1'.

38.6.6 Scan Address

Name: SCANADR
Offset: 0x05
Reset: 0x00
Property: -

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

Bits 7:0 – SCANADR[7:0] Address Being Scanned
The least significant byte (LSB) of the address being scanned. This allows the software to monitor if a scan is ongoing by confirming the address changes.
The contents of this register are undefined if the CRC Source (SRC) bit field configuration of the Control A (CRCSCAN.CTRLA) register is MANUAL.
The SCANADR register contains byte addresses, i.e., a scan of 32 words will cause SCANADR to increment by 64.

38.6.7 Input Data

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

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

Bits 7:0 – DATA[7:0] Input Data
Input data written by the user application in manual mode (when the CRC Source (SRC) bit field configuration in the CRCSCAN.CTRLA register is MANUAL). Data can be written every clock cycle.

38.6.8 CRC Result

Name: CRC
Offset: 0x08
Reset: 0xFFFFFFFF
Property: -

Bit	31	30	29	28	27	26	25	24
	CRC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	CRC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	CRC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

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

Bits 31:0 – CRC[31:0] CRC Result

The CRC result register represents the 32-bit result value. The low byte ([7:0], suffix 0) is accessible at the original offset, and the n higher bytes can be accessed at offset +n. The temporary registers do not buffer the CRC result register. The Most Significant bit (MSb) of the CRC result is bit 31. If CRCSCAN is in 16-bit CRC mode (the CRC Mode Select (CRCSEL) bit in the CRCSCAN.CTRLA register is '0'), only bit field [15:0] is used, and bit field [31:16] will read as '0'.

This register is the actual linear-feedback shift register (LFSR) used in the CRC calculation. This register is reset by writing a '1' to the RESET bit in the CRCSCAN.CTRLA register. Such a reset must be performed between each data block in MANUAL mode so that the CRC calculation on each block starts with the correct initial value.

This register is only readable if the CRC Source (SRC) bit field configuration in the CRCSCAN.CTRLA register is in MANUAL mode to prevent information leakage. Any other configuration of the SRC bit field will cause all reads to return '0x0000_0000'.

If reading the reset value of the SFR in the CRC32 mode (CRCSEL=CRC32 in CRCSCAN.CTRLA), the value read will be '0x0000_0000' due to post-processing of the read result.

39. CCL - Configurable Custom Logic

39.1 Features

- Glue Logic for General Purpose PCB Design
- Six 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

39.2 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.

39.2.1 Block Diagram

Figure 39-1. CCL Block Diagram

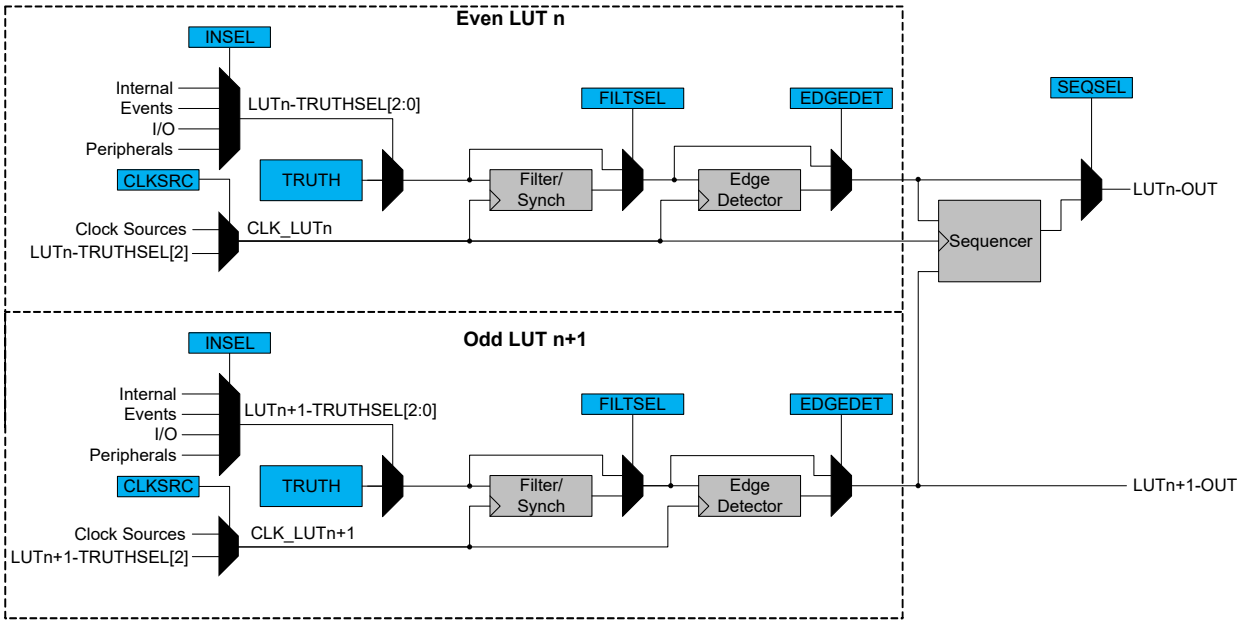


Table 39-2. Sequencer and LUT Connection

Sequencer	Even and Odd LUT
SEQ0	LUT0 and LUT1
SEQ1	LUT2 and LUT3
SEQ2	LUT4 and LUT5

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

39.2.2.1 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	None		
0x01	FEEDBACK	LUTn		
0x02	LINK	LUT[n+1]		
0x03	EVENTA	EVENTA		
0x04	EVENTB	EVENTB		
0x05	INn	IN0	IN1	IN2
0x06	AC	AC0 OUT	AC1 OUT	AC2 OUT
0x07	ZCD	ZCD0 OUT	ZCD3 OUT	
0x08	USART	USART0 TXD	USART1 TXD	USART2 TXD
0x09	SPI	SPI0 MOSI	SPI0 MOSI	SPI0 SCK

.....continued				
Value	Input Source	INSEL0[3:0]	INSEL1[3:0]	INSEL2[3:0]
0x0A	TCA0	WO0	WO1	WO2
0x0B	TCB	TCB0 WO	TCB1 WO	TCB2 WO
0x0C	TCD0	WOA	WOB	WOC

Notes:

- SPI connections to the CCL work only in SPI Host mode
- USART connections to the CCL work only in asynchronous/synchronous USART Host mode

39.3 Functional Description

39.3.1 Operation

39.3.1.1 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 (CCL.LUTnCTRLA) register). 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 (CCL.SEQCTRLn) registers
- LUT n Control x (CCL.LUTnCTRLx) registers, except the ENABLE bit in the CCL.LUTnCTRLA register
- TRUTHn (CCL.TRUTHn) registers

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.

39.3.1.2 Enabling, Disabling, and Resetting

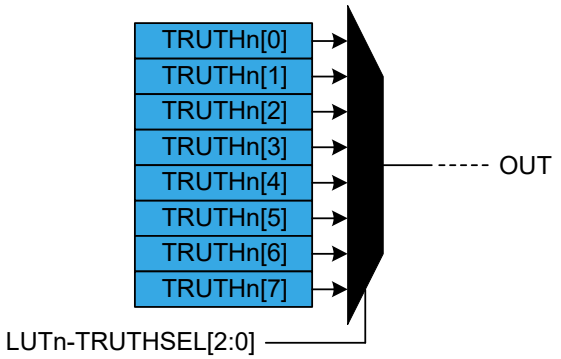
The CCL is enabled by writing a '1' to the ENABLE bit in the Control A (CCL.CTRLA) register. The CCL is disabled by writing a '0' to that ENABLE bit.

Each LUT is enabled by writing a '1' to the LUT Enable (ENABLE) bit in the CCL.LUTnCTRLA register. Each LUT is disabled by writing a '0' to the ENABLE bit in the CCL.LUTnCTRLA register.

39.3.1.3 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]). It is possible to realize any 3-input Boolean logic function using one LUT.

Figure 39-2. Truth Table Output Value Selection of an LUT



Configure the truth table inputs (LUTn-TRUTHSEL[2:0]) by writing the Input Source Selection bit fields in the LUT Control registers:

- INSEL0 in CCL.LUTnCTRLB
- INSEL1 in CCL.LUTnCTRLB
- INSEL2 in CCL.LUTnCTRLC

Each combination of the input bits (LUTn-TRUTHSEL[2:0]) corresponds to one bit in the CCL.TRUTHn register, as shown in the table below:

Table 39-3. Truth Table of an LUT

LUTn-TRUTHSEL[2]	LUTn-TRUTHSEL[1]	LUTn-TRUTHSEL[0]	OUT
0	0	0	TRUTHn[0]
0	0	1	TRUTHn[1]
0	1	0	TRUTHn[2]
0	1	1	TRUTHn[3]
1	0	0	TRUTHn[4]
1	0	1	TRUTHn[5]
1	1	0	TRUTHn[6]
1	1	1	TRUTHn[7]

Important: Consider the unused inputs turned off (tied low) when logic functions are created.

Example 39-1. LUT Output for CCL.TRUTHn = `0x42`

If CCL.TRUTHn is configured to '`0x42`', the LUT output will be '`1`' when the inputs are '`0x1`' or '`0x6`', and will be '`0`' for any other combination of inputs.

39.3.1.4 Truth Table Inputs Selection

Input Overview

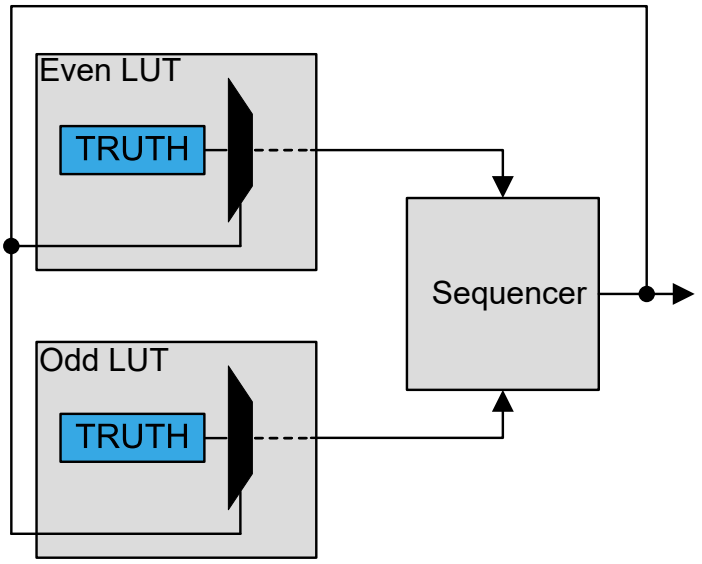
The inputs can be individually:

- OFF
- Driven by peripherals
- Driven by internal events from the Event System
- Driven by I/O pin inputs
- Driven by other LUTs

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 39-3. Feedback Input Selection



When selected (INSELy = FEEDBACK in LUTnCTRLx), the sequencer (SEQ) output is used as input for the corresponding LUTs.

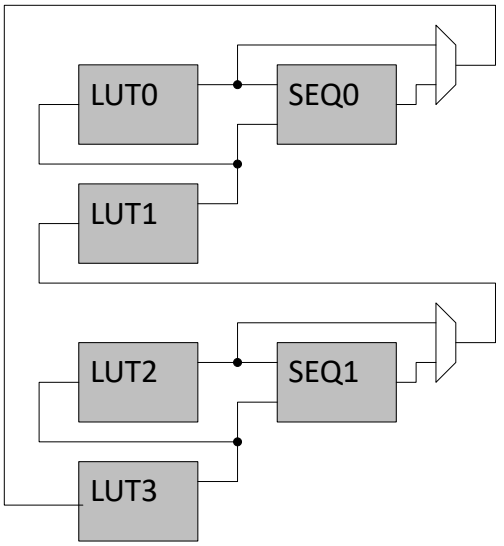
Linked LUT (LINK)

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 39-2. 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 39-4. 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 and Considerations* 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 (LUTnCTRLB and LUTnCTRLC) registers.

39.3.1.5 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.

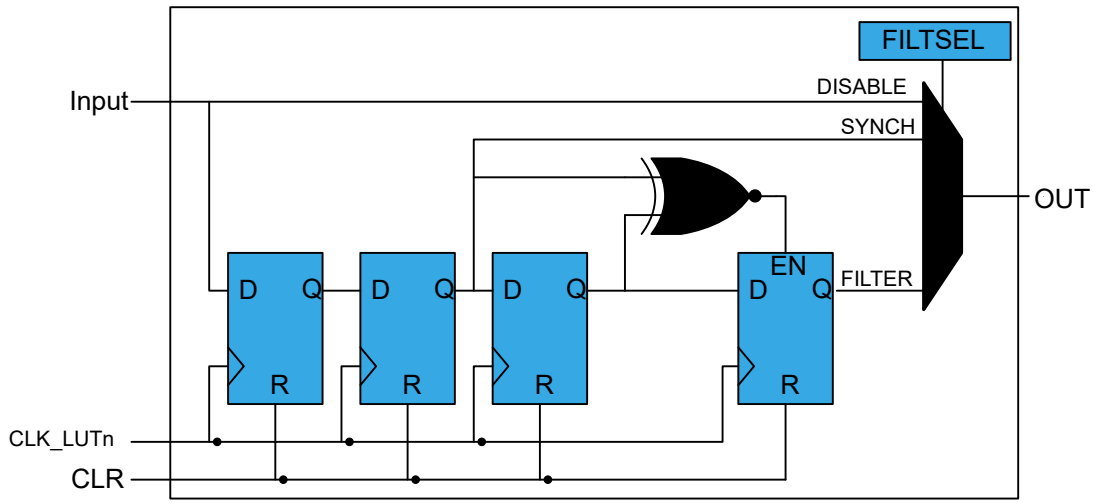
The Filter Selection (FILTSEL) bits in the LUT n Control A (CCL.LUTnCTRLA) registers define the digital filter options.

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 39-5. Filter



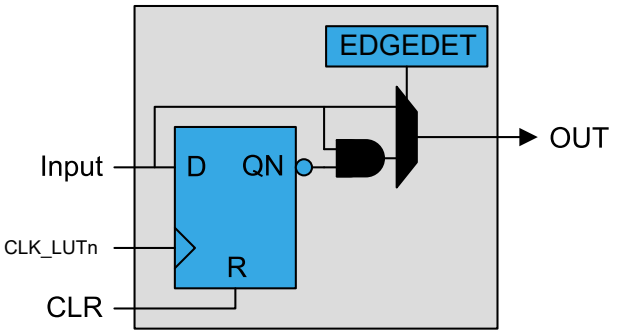
39.3.1.6 Edge Detector

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 an inverted output.

The edge detector is enabled by writing '1' to the Edge Detection (EDGEDET) bit in the LUTn Control A (CCL.LUTnCTRLA) register. 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 an LUT, the corresponding internal edge detector logic is cleared one clock cycle later.

Figure 39-6. Edge Detector



39.3.1.7 Sequencer Logic

Each LUT pair can be connected to a sequencer. The sequencer can function as either gated 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 (CCL.SEQCTRLn) register.

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 (CCL.LUTnCTRLA) register.

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 (GDFF)

The D input is driven by the even LUT output, and the G input is driven by the odd LUT output.

Figure 39-7. Gated D Flip-Flop

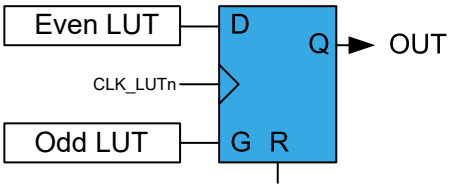


Table 39-4. GDFF Characteristics

R	G	D	OUT
1	X	X	0
0	1	1	1
0	1	0	0
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 39-8. JK Flip-Flop

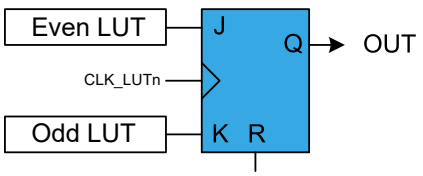


Table 39-5. JK Characteristics

R	J	K	OUT
1	X	X	0
0	0	0	Hold state (no change)
0	0	1	0
0	1	0	1
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 39-9. D Latch

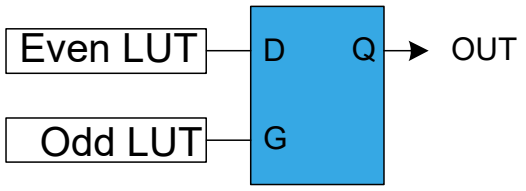


Table 39-6. D Latch Characteristics

G	D	OUT
0	X	Hold state (no change)
1	0	0
1	1	1

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 39-10. RS Latch

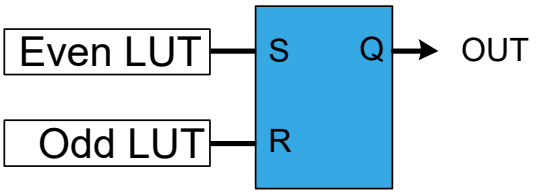


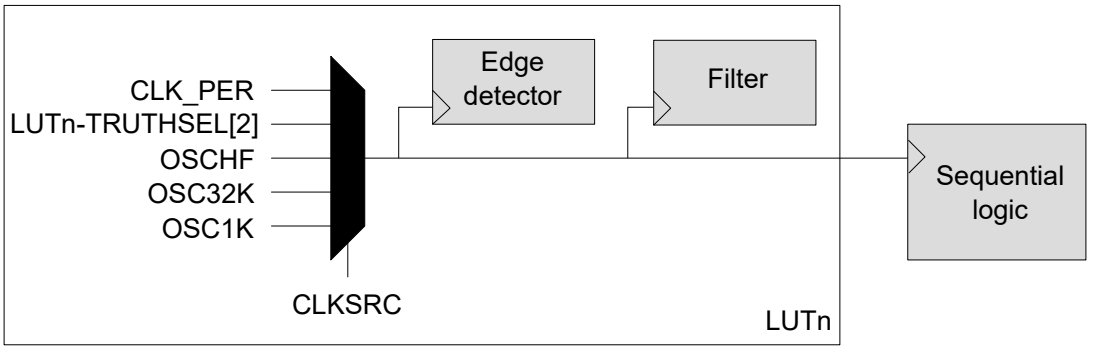
Table 39-7. RS Latch Characteristics

S	R	OUT
0	0	Hold state (no change)
0	1	0
1	0	1
1	1	Forbidden state

39.3.1.8 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 39-11. 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.

39.3.2 Interrupts

Table 39-8. 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.

39.3.3 Events

The CCL can generate the events shown in the table below.

Table 39-9. 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 39-10. 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 (CCL.LUTnCTRLB or LUTnCTRLC) registers.

Refer to the *EVSYS - Event System* section for more details regarding the event types and the EVSYS configuration.

39.3.4 Sleep Mode Operation

Writing the Run In Standby (RUNSTDBY) bit in the Control A (CCL.CTRLA) register 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 (CLKSRC) bit in the LUT n Control A (CCL.LUTnCTRLA) register 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.

39.4 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]							

39.5 Register Description

39.5.1 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

39.5.2 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

39.5.3 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

39.5.4 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

39.5.5 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 – INTMODEn
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

39.5.6 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

39.5.7 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.

39.5.8 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 an LUT. The CLK_LUTn of the even LUT is used for clocking the sequencer of an LUT pair.

Value	Name	Description
0x0	CLKPER	CLK_PER is clocking the LUT
0x1	IN2	IN2 is clocking the LUTn
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

39.5.9 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 Host 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 host

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	IN1 input source
0x6	AC1	AC1 OUT input source
0x7	ZCD3	ZCD3 OUT input source
0x8	USART1	USART1 TXD input source
0x9	SPI0	SPI0 MOSI input source
0xA	TCA0	TCA0 WO1 input source
0xB	TCB1	TCB1 WO input source
0xC	TCD0	TCD0 WOB 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	IN0 input source
0x6	AC0	AC0 OUT input source
0x7	ZCD0	ZCD0 OUT input source
0x8	USART0	USART0 TXD input source
0x9	SPI0	SPI0 MOSI input source
0xA	TCA0	TCA WO0 input source
0xB	TCB0	TCB0 WO input source
0xC	TCD0	TCD0 WOA input source

.....continued		
Value	Name	Description
Other	-	Reserved

39.5.10 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	IN2 input source
0x6	AC2	AC2 OUT input source
0x7	ZCD3	ZCD3 OUT input source
0x8	USART2	USART2 TXD input source
0x9	SPI0	SPI0 SCK input source
0xA	TCA0	TCA0 WO2 input source
0xB	TCB2	TCB2 WO input source
0xC	TCD0	TCD0 WOC input source
Other	-	Reserved

39.5.11 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 determine the output of LUTn according to the LUTn-TRUTHSEL[2:0] inputs.

Bit Name	Value	Description
TRUTH[0]	0	The output of LUTn is '0' when the inputs are '0x0'
	1	The output of LUTn is '1' when the inputs are '0x0'
TRUTH[1]	0	The output of LUTn is '0' when the inputs are '0x1'
	1	The output of LUTn is '1' when the inputs are '0x1'
TRUTH[2]	0	The output of LUTn is '0' when the inputs are '0x2'
	1	The output of LUTn is '1' when the inputs are '0x2'
TRUTH[3]	0	The output of LUTn is '0' when the inputs are '0x3'
	1	The output of LUTn is '1' when the inputs are '0x3'
TRUTH[4]	0	The output of LUTn is '0' when the inputs are '0x4'
	1	The output of LUTn is '1' when the inputs are '0x4'
TRUTH[5]	0	The output of LUTn is '0' when the inputs are '0x5'
	1	The output of LUTn is '1' when the inputs are '0x5'
TRUTH[6]	0	The output of LUTn is '0' when the inputs are '0x6'
	1	The output of LUTn is '1' when the inputs are '0x6'
TRUTH[7]	0	The output of LUTn is '0' when the inputs are '0x7'
	1	The output of LUTn is '1' when the inputs are '0x7'

40. AC - Analog Comparator

40.1 Features

- Selectable Response Time
- Selectable Hysteresis
- Analog Comparator Output Available on Pin
- Comparator Output Inversion Available
- Flexible Input Selection:
  - 5 positive pins
  - 3 negative pins
  - Internal reference voltage generator (DACREF)
- Interrupt Generation on:
  - Rising edge
  - Falling edge
  - Both edges
- Window Function Interrupt Generation on:
  - Signal above the window
  - Signal inside the window
  - Signal below the window
  - Signal outside the window
- Event Generation:
  - Comparator output
  - Window function

40.2 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.

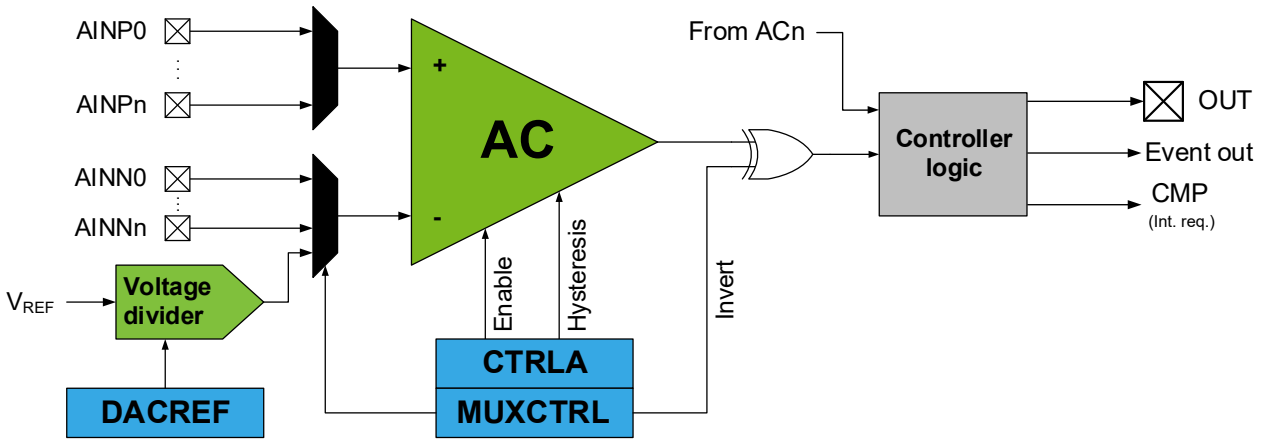
The input selection includes analog port pins and internally generated inputs. The AC digital output goes through controller logic, enabling customization of the signal for use internally with the Event System or externally on the pin.

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).

40.2.1 Block Diagram

Figure 40-1. AC Block Diagram



40.2.2 Signal Description

Signal	Description	Type
AINNn	Negative input n	Analog
AINPn	Positive input n	Analog
OUT	Comparator output of AC	Digital

40.3 Functional Description

40.3.1 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.

40.3.2 Operation

40.3.2.1 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.

40.3.2.2 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.

40.3.2.3 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.

40.3.2.4 Power Modes

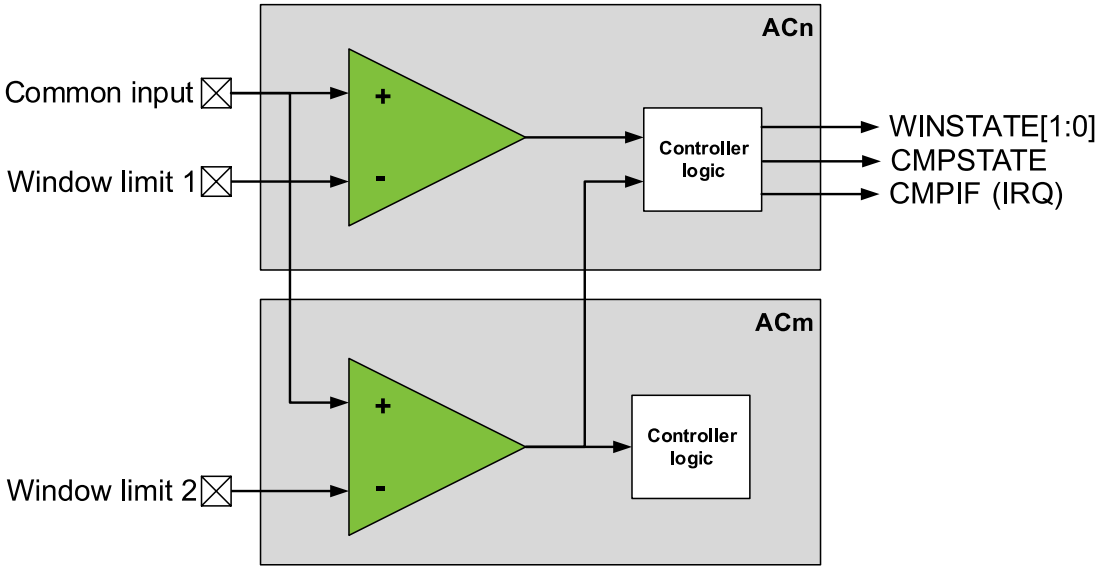
For power sensitive applications, the AC provides multiple power modes with balance power consumption and response time. A mode is selected by writing to the Power Profile (POWER) bit field in the Control A (ACn.CTRLA) register.

40.3.2.5 Window Mode

Each AC (i.e., ACn) can be configured to work together with another comparator (i.e., ACm) 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 ACn instance is connected to the current comparator (ACm) to create the window comparator. The user is responsible for configuring the MUXPOS and MUXNEG bit fields in the MUX Control (MUXCTRL) register for ACn and ACm, so they match the setup in the figure below. Note that the MUXPOS bit field in the MUXCTRL register of both ACs must be configured to the same pin.

Figure 40-2. 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 the window - the input signal is above the upper limit
- Inside the window - the input signal is between the lower and upper limits
- Below the 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 the window - the interrupt/event is issued when the input signal is above the upper limit
- Inside the window - the interrupt/event is issued when the input signal is between the lower and upper limits
- Below the window - the interrupt/event is issued when the input signal is below the lower limit
- Outside the window - the interrupt/event is issued when the input signal is not between the lower and upper limits

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.

40.3.3 Events

The AC can generate the following events:

Table 40-1. 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 *EVSYS - Event System* section for more details regarding event types and Event System configuration.

40.3.4 Interrupts

Table 40-2. 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.

40.3.5 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.

40.4 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 ... 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

40.5 Register Description

40.5.1 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.

40.5.2 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

40.5.3 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
0x4	AINP4	Positive pin 4
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	-	Reserved
0x2	AINN2	Negative pin 2
0x3	AINN3	Negative pin 3
0x4	DACREF	DAC Reference
Other	-	Reserved

40.5.4 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:

$$V_{\mathrm{DACREF}} = \frac{\mathrm{DACREF}[7:0]}{256} \times V_{\mathrm{REF}}$$

40.5.5 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 40-3. 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 40-4. Interrupt Generation with Single Comparator

Value	Name	Description
0x0	BOTHEDGE	Positive and negative inputs crosses
0x1	-	Reserved
0x2	NEGEDGE	Positive input goes below negative input
0x3	POSEDGE	Positive input goes above 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.

40.5.6 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 40-5. 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, and 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.

41. ADC - Analog-to-Digital Converter

41.1 Features

- 10-Bit Single-Ended Analog-to-Digital Converter (ADC)
  - Up to 13 bits with oversampling
- Conversion Rate Up to 170 ksps ksps, at 10-bit Resolution
- Up to 23 I/O Pin Inputs, in Addition to Internal Inputs from Sensors and References
- Multiple Internal ADC Reference Voltages
  - V_{DD}
  - 1.024V
  - 2.048V
  - 2.500V
  - 4.096V
- Independent Voltage Reference Sources For Each ADC Instance
- External Reference Input
- Single and Free-Running Conversions
- Event Triggered Conversion
- Series and Burst Accumulation Modes
- Accumulation of Up to 64 Conversions
- Left or Right Adjusted Result
- Interrupts on Conversion Complete
- Configurable Window Comparator

41.2 Overview

The Analog-to-Digital Converter (ADC) peripheral is a single-ended ADC with a 10-bit resolution. It is connected to an analog input multiplexer to select between multiple inputs. The ADC measures the voltage between the selected input and 0V (GND). The ADC inputs can be internal (for example, the internal temperature sensor) 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, which makes it possible to sample input signals periodically, trigger an ADC conversion on a particular condition, and trigger ADC conversions in Standby sleep mode. A window compare feature is available to monitor the input signal. It can be configured to trigger an interrupt if the sample is under or over a user-defined threshold or inside or outside a user-defined window.

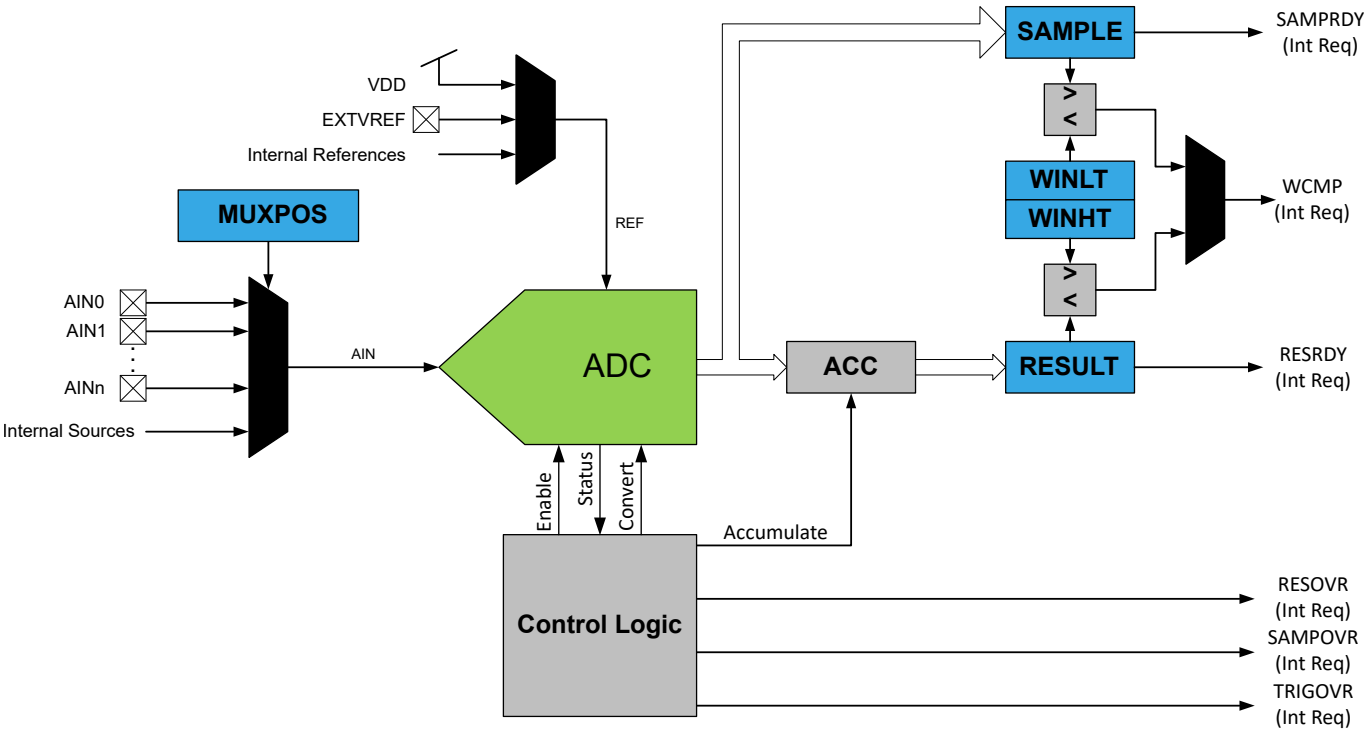
The ADC input signal is fed through a sample-and-hold circuit that ensures that the input voltage to the ADC is kept at a constant level during the conversion.

The ADC supports sampling in bursts where a configurable number of samples are accumulated into a single ADC result (Sample Accumulation).

The ADC reference voltage can be either internal or supplied from the external analog reference pin (EXTVREF).

41.2.1 Block Diagram

Figure 41-1. Block Diagram



41.2.2 Signal Description

Signal	Type	Description
AIN	Analog input	Analog input
REF	Analog input	Voltage reference

41.3 Functional Description

41.3.1 Definitions

- Conversion: The operation where analog values on the selected ADC inputs are transformed into a digital representation.
- Sample: The value placed in the Sample (ADCn.SAMPLE) register, that is, the outcome of a conversion operation.
- Result: The value placed in the Result (ADCn.RESULT) register. Depending on the ADC configuration, this value is a single sample or the sum of multiple accumulated samples.

41.3.2 Initialization

The following steps are recommended to initialize and run the ADC in the basic operation:

1. Configure the timebase by writing to the TIMEBASE bit field in the Main Clock Timebase (MCLKTIMEBASE) register in the Clock Controller (CLKCTRL).
2. Configure the Prescaler (PRESC) bit field in the Control B (ADCn.CTRLB) register.
3. Configure the Reference Select (REFSEL) bit field in the Control C (ADCn.CTRLC) register.
4. Configure the Sample Duration (SAMPDUR) bit field in the Control E (ADCn.CTRLE) register.
5. Optional: Configure the number of samples to be accumulated by writing the Sample Accumulation Number Select (SAMPNUM) bit field in the Control F (ADCn.CTRLF) register.

6. Optional: Enable the Free-Running mode by writing a '1' to the Free-Running (FREERUN) bit in the Control F (ADCn.CTRLF) register.
7. Configure a positive input by writing to the MUXPOS bit field in the Positive Input Multiplexer (ADCn.MUXPOS) register.
8. Configure the mode of operation for the ADC by writing to the MODE bit field in the Command (ADCn.COMMAND) register.
9. Configure how an ADC conversion will start by writing to the START bit field in the Command (ADCn.COMMAND) register.
10. Enable the ADC by writing a '1' to the ENABLE bit in the Control A (ADCn.CTRLA) register.

41.3.3 Operation

41.3.3.1 Operation Modes

The ADC supports four operation modes, configured in the Command (ADCn.COMMAND) register.

The operation modes can be split into three groups:

- Single mode - Single conversion per trigger, with 8- or 10-bit conversion output
- Series Accumulation mode - One conversion per trigger, with an accumulation of n samples
- Burst Accumulation mode - A burst with n samples accumulated as fast as possible after a single trigger

Series and Burst modes utilize 10-bit conversions. The SAMPNUM bit field in the Control F (ADCn.CTRLF) register controls the number of samples to accumulate. The accumulator is always reset to zero when a new Series or Burst accumulation is started.

The table below shows an overview of the available operation modes.

Table 41-1. Operation Modes

Operation Mode	COMMAND Mode	Conversions Per Trigger	RESULT Update
Single 8-bit	0	1	Every conversion
Single 10-bit	1		
Series Accumulation	2	1	After SAMPNUM conversions
Burst Accumulation	3	SAMPNUM	After SAMPNUM conversions

In addition to the ordinary operating modes, an Accumulator Test mode is available. See section Accumulator Test for more information.

41.3.3.2 Conversion Triggers

A conversion is started by one of the following triggers, depending on the configuration of the START bit field in the Command (ADCn.COMMAND) register:

- Writing the IMMEDIATE value to the START bit field in the Command (ADCn.COMMAND) register
- Receiving an event input
- Writing to the Positive Input Multiplexer (ADCn.MUXPOS) register

Continuously repeating Single conversion or Burst accumulation can be enabled by writing a '1' to the Free-Running (FREERUN) bit in the Control F (ADCn.CTRLF) register before starting the first conversion. The Free-Running mode is unsupported in Series mode.

Attempting to trigger a new conversion before the ongoing conversion is finished will set the Trigger Overrun Interrupt (TRIGOVR) flag in the Interrupt Flags (ADCn.INTFLAGS) register, and the trigger will be ignored.

The Result Ready and Sample Ready (RESRDY and SAMPRDY) interrupt flags in the Interrupt Flags (ADCn.INTFLAGS) register show if a conversion or accumulation has finished. These flags also trigger the corresponding interrupts if enabled in the Interrupt Control (ADCn.INTCTRL) register.

41.3.3.3 Aborting a Conversion

These actions will abort an ongoing conversion:

- Writing STOP to the START bit field in the Command (ADCn.COMMAND) register
- Writing a new value that changes the Reference Selection bit field (REFSEL) in the Control C (ADCn.CTRLC) register during a conversion
- Writing a new value that changes Positive Input Multiplexer (ADCn.MUXPOS) register

This will result in undefined values in the Result (ADCn.RESULT) and Sample (ADCn.SAMPLE) registers.

41.3.3.4 Output Formats

The resolution of an ADC output is given by the number of bits used in the conversion:

- An **8-bit** ADC means 256 discrete levels (from 0 to 255)
- A **10-bit** ADC means 1024 discrete levels (from 0 to 1023)

A **single-ended** conversion measures the voltage difference between one input voltage and ground. The result of an N-bit single-ended conversion is an unsigned (positive) integer between 0 and the maximum value 2^N - 1:

```
ADC result ∈ [0, 2^N-1]
```

The output from an ADC conversion is given by the equations below:

Equation	Explanation
Single-Ended 8-bit conversion:	
$RESULT = \frac{V_{IN}}{V_{REF}} \times 255$	• The multiplication factor is 255 (2^8 - 1 = 255) • The result (RES) will be an integer between 0 and 255
Single-Ended 10-bit conversion:	
$RESULT = \frac{V_{IN}}{V_{REF}} \times 1023$	• The multiplication factor is 1023 (2^{10} - 1 = 1023) • The result (RES) will be an integer between 0 and 1023

Explanation to the equations:

- $\mathbf{V_{IN}}$ is the input voltage
- $\mathbf{V_{REF}}$ is the selected reference voltage

The ADC has two output registers, the Sample (ADCn.SAMPLE) and Result (ADCn.RESULT) registers. The 16-bit Sample register will always be updated with the latest ADC conversion output (one sample). In series and burst accumulation mode, samples are added in an internal sample accumulator, configured by the Sample Accumulation Number Select (SAMPNUM) bit field in the Control F (ADCn.CTRLF) register. The accumulated result will automatically be transferred to the 16-bit Result register when all samples are completed. In single conversion modes, the Result register will be updated with the latest sample, identical to the Sample register.

The Left Adjust (LEFTADJ) bit in the Control F (ADCn.CTRLF) register enables the left-shift of the output data. If enabled, this will left shift the output from both the Result and Sample registers. Left adjusting the result will effectively scale the result such that however many samples are accumulated, the MSb of the result is always situated at the leftmost bit position in the Result register.

The data format for a sample is an unsigned number, where `0x000` represents zero, and `0x3FF` represents the largest number (full scale). If the analog input is higher than the reference level of the

ADC, the 10-bit ADC output will be equal to the maximum value of 0x3FF. Likewise, if the input is below 0V, the ADC output will be 0x000.

The following table shows the Result register output formats by mode of operation and left adjustment.

Table 41-2. RESULT Register

MODE(1)	LEFTADJ	RESULT[15:10]	RESULT[9:8]	RESULT[7:0]
0	X(2)	0x00		Conversion[7:0]
1	0	0x00	Conversion[9:0]	
	1	Conversion[9:0] << 6		
2, 3	0	Accumulation[15:0]		
2, 3	1	Accumulation[15:0] << (6 - SAMPNUM)		

Notes:

1. See section Operation Modes.
2. Left adjust is not available in 8-bit mode.

The following table shows the Sample register output formats by mode of operation and left adjustment.

Table 41-3. SAMPLE Register

MODE(1)	LEFTADJ	SAMPLE[15:10]	SAMPLE[9:8]	SAMPLE[7:0]
0	X	0x00		Conversion[7:0]
Other	0	0x00	Conversion[9:0]	
	1	Conversion[9:0] << 6		

Note:

1. See section Operation Modes.

41.3.3.5 ADC Clock

The ADC clock (CLK_ADC) is down-scaled from the peripheral clock (CLK_PER), which can be configured by the Prescaler (PRESC) bit field in the Control B (ADCn.CTRLB) register. Refer to the ADC section of the Electrical Characteristics section for details on the minimum and maximum allowed values for the ADC clock (CLK_ADC) period.

Some of the internal timings in the ADC are independent of CLK_ADC. To ensure correct internal timing regardless of the ADC clock frequency, a 1 µs timebase, given in CLK_PER cycles, must be written in the Main Clock Timebase (MCLKTIMEBASE) register in the Clock Controller (CLKCTRL) peripheral. For more details, refer to the MCLKTIMEBASE register description in the *CLKCTRL - Clock Controller* section.

41.3.3.6 Input and Reference Selection

The input selection to the ADC is controlled by the Positive Input Multiplexer (ADCn.MUXPOS) register.

The reference voltage for the ADC (V_{REF}) controls the conversion range of the ADC. V_{REF} can be selected by writing the Reference Selection (REFSEL) bit field in the Control C (ADCn.CTRLC) register. Except for V_{DD}, the internal reference voltages are generated from an internal band gap reference.

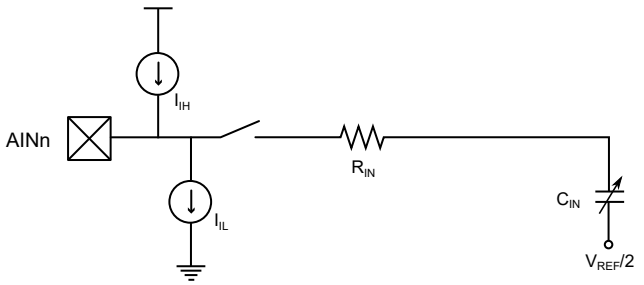
Changing input and reference selections while a conversion is ongoing will corrupt the output. Reading the ADC Busy (ADCBUSY) bit in the Status (ADCn.STATUS) register can detect an ongoing conversion. Writing STOP to the Start Conversion (START) bit field in the Command (ADCn.COMMAND) register will stop an ongoing conversion.

After switching input or reference, the ADC requires time to settle. Refer to the *Electrical Characteristics* section for further details.

41.3.3.6.1 Analog Input Circuit

The figure below illustrates the analog input circuit. An analog source connected to an analog input (AINn) is subject to the pin capacitance and input leakage of that pin (represented by I_H and I_L). When the input is selected, the source must also drive the Sample-Hold capacitor (C_{IN}) through the combined resistance of the input path (represented by R_{IN}). Refer to the *Electrical Characteristics* section for details on the input characteristics of the ADC.

Figure 41-2. Analog Input Schematic



If a source with high impedance is used, the sampling time may need to be increased. The required sample time will depend on how long the source needs to charge the C_{IN} capacitor and can be configured using the Sample Duration (SAMPDUR) bit field in the Control E (ADCn.CTRLE) register.

41.3.3.7 Conversion Timing

Some of the analog modules in the ADC are disabled between conversions and require time to initialize before conversion starts. Only the modules used by the current ADC configuration are enabled, and as the initializations run in parallel, the limiting factor is the module with the slowest initialization time.

By writing the Low Latency (LOWLAT) bit in the Control A (ADCn.CTRLA) register to '1', the latency of the ADC peripheral can be reduced. This will keep the configured modules continuously enabled, effectively removing all initialization time at the start of a conversion. The initialization time is still needed when enabling the ADC for the first time and reconfiguring the ADC to use an input or reference that requires initialization. The ADC Busy (ADCBUSY) bit in the Status (ADCn.STATUS) register can be used to check if initialization is ongoing.

The following table shows the different initialization times needed by the analog modules.

Table 41-4. ADC Initialization Timing

Analog Module	LOWLAT	Initialization Time (µs)
ADC	0	6
	1 (1)	0
Settling of internal references	0	40
	1 (1)	2 (2)
Settling of internal references after changing from one internal reference voltage to another internal reference voltage	X	20
Settling internal Temperature Sensor input	0	40
	1 (1)	0
Settling of internal Analog Comparator Reference DAC input	X	20

Notes:

1. The LOWLAT timing values are valid between two conversions that both have the LOWLAT bit written to '1'.
2. When changing from one internal reference voltage to another internal reference voltage. The full 40 µs delay will be used if changing from a non-internal reference to an internal reference.

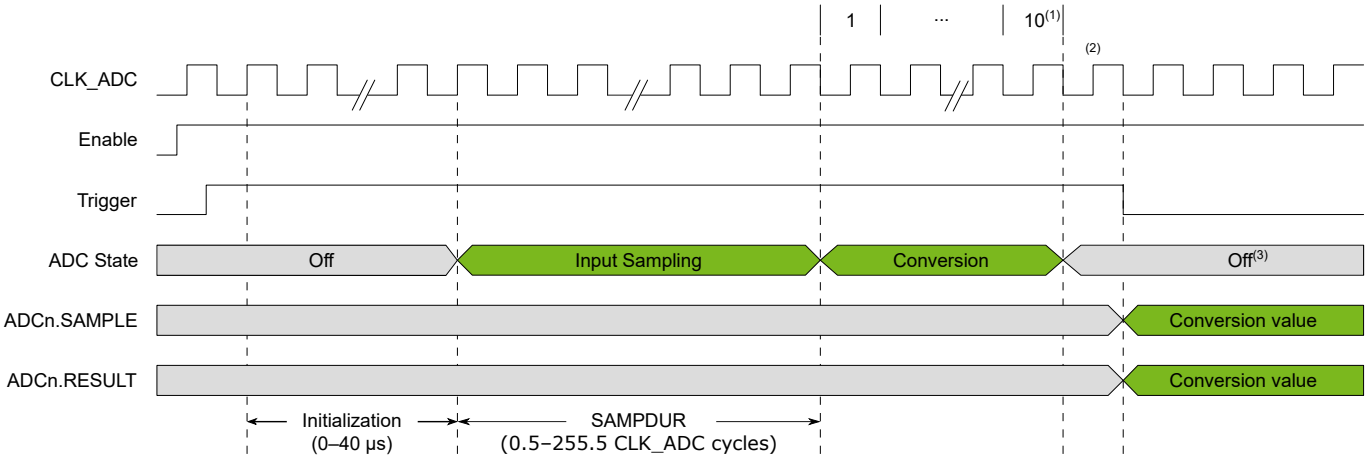
The sampling period of the input to the ADC is configured through the Sample Duration (SAMPDUR) bit field in the Control E (ADCn.CTRLE) register as *SAMPDUR* + ½ CLK_ADC cycles.

If using an internal reference, the value of the SAMPDUR bit field in the Control E (ADCn.CTRLE) register must be ≥ (4 µs * f_{CLK_ADC}) - 1.5.

41.3.3.7.1 Single Conversion

The figure below shows the timing diagram for the ADC when running in Single 8- or 10-bit mode.

Figure 41-3. Timing Diagram - Single Conversion



Notes:

1. In Single 8-bit mode, the length of the Conversion state is eight CLK_ADC cycles. In all other modes, it is ten cycles.
2. The time from the conversion has finished to the outputs are available in the registers is 0.5 CLK_ADC cycles followed by one CLK_MAIN cycle. With a minimum prescaler of two, this sums up to the maximum of one CLK_ADC cycle.
3. If the Low Latency (LOWLAT) bit is set to '1' in the Control A (ADCn.CTRLA) register, the analog modules in the ADC will not turn OFF at the end of the conversion, which will eliminate the initialization time when triggering the following conversion.

The total conversion time for a single result is calculated by:

$$t_{conv}\,(\text{10-bit}) = t_{initialization} + \frac{\text{SAMPDUR}+½ + 10 + 1}{f_{\text{CLK\_ADC}}}$$

$$t_{conv}\,(\text{8-bit}) = t_{initialization} + \frac{\text{SAMPDUR}+½ + 8 + 1}{f_{\text{CLK\_ADC}}}$$

If the Free-Running (FREERUN) bit is set to '1' in the Control F (ADCn.CTRLF) register, a new conversion will be started immediately after a result is available in the Result (ADCn.RESULT) register. The Free-Running conversion rate (f_{conv}) is calculated by:

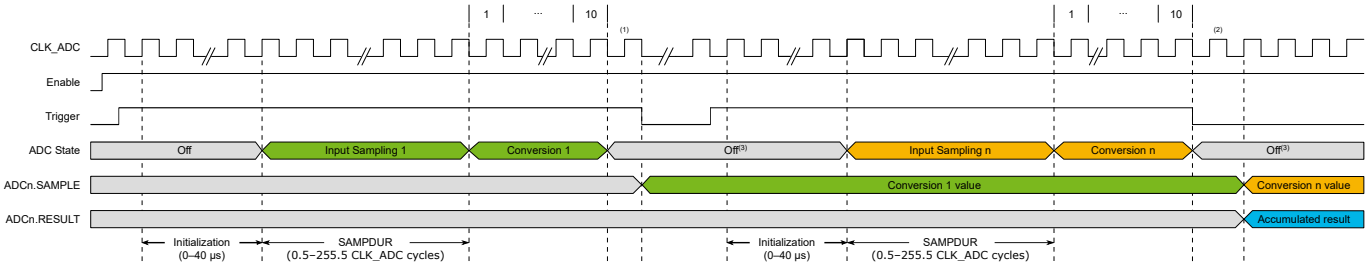
$$f_{\text{conv}}\,(\text{10-bit}) = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR}+½+10 + 1}$$

$$f_{\text{conv}}\,(8\text{-bit}) = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR}+½+8+1}$$

41.3.3.7.2 Series Accumulation

The figure below shows the timing diagram for the ADC when running in Series Accumulation mode.

Figure 41-4. Timing Diagram - Series Accumulation



Notes:

1. The time from the conversion has finished to the outputs are available in the registers is 0.5 CLK_ADC cycles followed by one CLK_MAIN cycle. With a minimum prescaler of two, this sums up to the maximum of one CLK_ADC cycle.
2. The time from the final conversion has finished to the Result (ADCn.RESULT) register is updated is 0.5 CLK_ADC cycles followed by two CLK_MAIN cycles. With a minimum prescaler of two, this sums up to the maximum of 1.5 CLK_ADC cycles.
3. If the Low Latency (LOWLAT) bit is set to '1' in the Control A (ADCn.CTRLA) register, the analog modules in the ADC will not turn OFF at the end of the conversion, which will eliminate the initialization time when triggering the following conversion.

The number of samples to accumulate is set by the Sample Number (SAMPNUM) bit field in the Control F (ADCn.CTRLF) register.

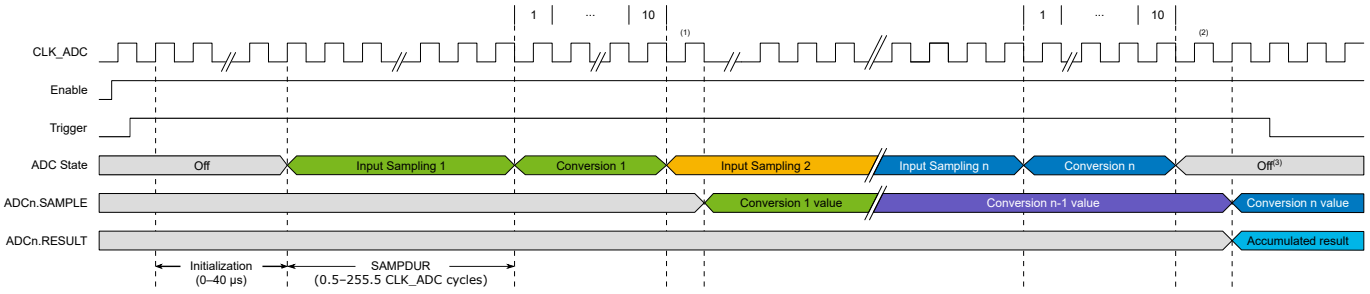
The total conversion time for each separate sample is calculated by:

$$t_{conv} = t_{initialization} + \frac{\text{SAMPDUR}+½+10+1}{f_{\text{CLK\_ADC}}}$$

41.3.3.7.3 Burst Accumulation

The figure below shows the timing diagram for the ADC when running in Burst Accumulation mode.

Figure 41-5. Timing Diagram - Burst Accumulation



Notes:

1. The time from the conversion has finished to the outputs are available in the registers is 0.5 CLK_ADC cycles followed by one CLK_MAIN cycle. With a minimum prescaler of two, this sums up to the maximum of one CLK_ADC cycle.
2. The time from the final conversion has finished to the Result (ADCn.RESULT) register is updated is 0.5 CLK_ADC cycles followed by two CLK_MAIN cycles. With a minimum prescaler of two, this sums up to the maximum of 1.5 CLK_ADC cycles.
3. If the Low Latency (LOWLAT) bit is set to '1' in the Control A (ADCn.CTRLA) register, the analog modules in the ADC will not turn OFF at the end of the conversion, which will eliminate the initialization time when triggering the following conversion.

The number of samples to accumulate is set by the Sample Number (SAMPNUM) bit field in the Control F (ADCn.CTRLF) register.

The total conversion time for a Burst Accumulation is calculated by:

$$t_{conv} = t_{initialization} + \frac{(\text{SAMPDUR} + ½ + 10) \times \text{SAMPNUM} \ + \ 1.5}{f_{\text{CLK\_ADC}}}$$

The Burst Accumulation conversion rate (f_{conv}) is calculated by:

$$f_{\text{conv}} = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR} + ½ + 10}$$

41.3.3.8 Temperature Measurement

An on-chip temperature sensor is available. To do a temperature measurement, follow these steps:

1. Configure the voltage reference to internal 2.048V by writing to the Reference Selection (REFSEL) bit field in the Control C (ADCn.CTRLC) register.
2. Select the temperature sensor as input in the Positive Input Multiplexer (ADCn.MUXPOS) register.
3. Configure the ADC Sample Duration by writing a value ≥ 32 µs × $f_{\text{CLK\_ADC}}$ to the Sample Duration (SAMPDUR) bit field in the Control E (ADCn.CTRLE) register.
4. Acquire the temperature sensor output voltage by running a 10-bit Single-Ended conversion.
5. Process the measurement result, as described below.

The measured voltage has a linear relationship to the temperature. The relationship between the ADC conversion result and temperature in kelvin is tested in production and is available in the Signature Row:

- SIGROW.TEMPSENSE0: gain/slope correction
- SIGROW.TEMPSENSE1: offset correction

Use the following equation to calculate the temperature in kelvin:

$$T = \frac{(\text{Offset Correction} - \text{ADC Result}) \times \text{Gain Correction}}{1024}$$

It is possible to perform further calibration to achieve even more accurate results. For example, multi-point calibration is performed by imposing known temperatures on the device and recording the ADC results. This helps create a precise calibration curve, which can be implemented in firmware to adjust the output result.

41.3.3.9 Window Comparator

The ADC features a window comparator, which automatically compares the conversion to two configurable thresholds. The comparison result is reflected by the WCMP bit in the Interrupt Flags (ADCn.INTFLAGS) register and optionally triggers the corresponding interrupt. The available modes are:

- The value is above a threshold
- The value is below a threshold
- The value is inside a window (above the lower threshold and below the upper threshold)
- The value is outside a window (either below the lower threshold or above the upper threshold)

The thresholds are set by writing to the Window Comparator Low and High Threshold (ADCn.WINLT and ADCn.WINHT) registers. The Window mode to use is selected by the Window Comparator mode (WINCM) bit field in the Control D (ADCn.CTRLD) register.

The Window Mode Source (WINSRC) bit in the Control D (ADCn.CTRLD) register selects if the comparison is done on the Result (ADCn.RESULT) register or the Sample (ADCn.SAMPLE) register. If an interrupt request is enabled for the WCMP flag, WINSRC selects which interrupt vector to request, RESRDY or SAMPRDY.

When accumulating multiple samples, if the Window Comparator source is the Result register, the comparison between the result and the threshold(s) will happen after the last conversion is complete. If the source is the Sample register, the comparison will happen after every conversion.

Assuming the ADC is already configured to run, follow these steps to use the Window Comparator mode:

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 by writing the WINSRC bit field and a non-zero value to the WINCM bit field in the Control D (ADCn.CTRLD) register.

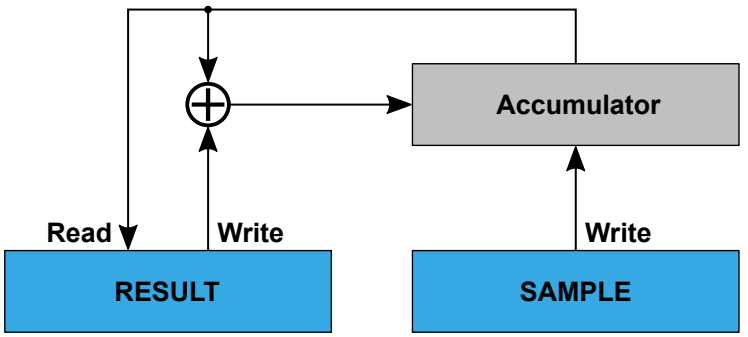
41.3.3.10 Accumulator Test

The Accumulator Test mode aids in diagnostics of the accumulator and the result formatting hardware and can test the integrity of the logic. It is enabled by configuring the MODE bitfield in the Command (ADCn.COMMAND) register to ACCTEST.

The Accumulator Test mode is used as follows:

- Writing to the Sample (ADCn.SAMPLE) register writes directly to the internal 16-bit accumulator
- Writing to the Result (ADCn.RESULT) register accumulates the 10 LSb of the written value in the internal 16-bit accumulator
- Reading the Result register will read the internal 16-bit accumulator

Figure 41-6. Accumulator Test



Notes:

1. If the Left Adjust (LEFTADJ) bit in the Control F (ADCn.CTRLF) register is '1', a value written to the Result register will be left adjusted 6-SAMPNUM bits before accumulated.
2. One peripheral clock (CLK_PER) clock cycle must be allowed between writing to the Result register and reading it back again. This can be achieved by executing any instruction between the write and the read operations, such as a NOP instruction.

41.3.4 Events

The ADC can generate the following events:

Table 41-5. ADC Event Generators

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
ADCn	RESRDY	Result ready	Pulse	CLK_PER	One CLK_PER period
ADCn	SAMPRDY	Sample ready	Pulse	CLK_PER	One CLK_PER period
ADCn	WCMP	Window compare match	Pulse	CLK_PER	One CLK_PER 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 41-6. ADC Event Users and Available Event Actions

User Name		Description	Input Detection	Async/Sync
Peripheral	Event			
ADCn	START	ADC start on event	Edge	Async

The START event action can be triggered if the EVENT_TRIGGER setting is written to the START bit field in the Command (ADCn.COMMAND) register.

41.3.5 Interrupts

Table 41-7. Available Interrupt Vectors and Sources

Vector Name	Source Name	Condition	Dependency
ADCn_ERROR	TRIGOVR	A new conversion is triggered while another is in progress	
	SAMPOVR	A new conversion overwrites an unread sample in the Sample (ADCn.SAMPLE) register	
	RESOVR	A new conversion or accumulation overwrites an unread result in the Result (ADCn.RESULT) register	
ADCn_RESRDY	RESRDY	A new result is available in the Result (ADCn.RESULT) register	
	WCMP	A conversion or accumulation matches the conditions set by the window comparator	The Window Source (WINSRC) bit in the Control D (ADCn.CTRLD) register is '0'
ADCn_SAMPRDY	SAMPRDY	A new sample is available in the Sample (ADCn.SAMPLE) register	
	WCMP	A sample matches the conditions set by the window comparator	The WINSRC bit in ADCn.CTRLD is '1'

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.

41.3.6 Sleep Mode Operation

The ADC can start conversions in Idle sleep mode if the START bit field in the Command (ADCn.COMMAND) register is configured to start a conversion on an event trigger. This is also possible in Standby sleep mode if the RUNSTDBY bit is set in the Control A (ADCn.CTRLA) register. For both cases, the ADC will finish any ongoing conversion or burst accumulation when transitioning to sleep.

If both the LOWLAT and RUNSTDBY bits in the Control A register are set, the ADC will keep all required modules ON during Standby sleep mode to start a conversion faster, at the expense of increased power consumption during sleep. A clock startup delay may be experienced as described in the *SLPCTRL - Sleep Controller* and *Electrical Characteristics* sections.

When the system enters POWERDOWN, the ADC will abort an ongoing conversion and enter sleep mode immediately. Make sure conversions have completed before entering Power-Down mode.

41.3.7 Debug Operation

If the Run in Debug mode (DBGRUN) bit in the Debug Control (ADCn.DBGCTRL) register is written to '1', the ADC will continue operating when the CPU is halted in Debug mode.

If DBGRUN is '0' when the CPU halts, an ongoing conversion will finish before the ADC halts.

41.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY		LOWLAT					ENABLE
0x01	CTRLB	7:0					PRESC[3:0]			
0x02	CTRLC	7:0						REFSEL[2:0]		
0x03	CTRLD	7:0					WINSRC	WINCM[2:0]		
0x04	CTRLE	7:0	SAMPDUR[7:0]							
0x05	CTRLF	7:0			FREERUN	LEFTADJ		SAMPNUM[2:0]		
0x06	INTCTRL	7:0			TRIGOVR	SAMPOVR	RESOVR	WCMP	SAMPRDY	RESRDY
0x07	INTFLAGS	7:0			TRIGOVR	SAMPOVR	RESOVR	WCMP	SAMPRDY	RESRDY
0x08	STATUS	7:0								ADCBUSY
0x09	DBGCTRL	7:0								DBGRUN
0x0A	COMMAND	7:0		MODE[2:0]				START[2:0]		
0x0B	MUXPOS	7:0		MUXPOS[6:0]						
0x0C	RESULT	7:0	RESULT[7:0]							
		15:8	RESULT[15:8]							
0x0E	SAMPLE	7:0	SAMPLE[7:0]							
		15:8	SAMPLE[15:8]							
0x10	WINLT	7:0	WINLT[7:0]							
		15:8	WINLT[15:8]							
0x12	WINHT	7:0	WINHT[7:0]							
		15:8	WINHT[15:8]							
0x14	TEMP	7:0	TEMP[7:0]							

41.5 Register Description

41.5.1 Control A

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

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

Bit 7 – RUNSTDBY Run in Standby
This bit controls whether the ADC will run in Standby sleep mode or not.

Value	Description
0	The ADC will not run in Standby sleep mode. An ongoing conversion will finish before the ADC enters sleep mode.
1	The ADC will run in Standby sleep mode. The main clock will be requested when the ADC is triggered to perform a conversion.

Bit 5 – LOWLAT Low Latency
This bit controls whether the analog modules required by the ADC are enabled continuously or only when needed.

Value	Description
0	The ADC enables the required analog modules only when starting a conversion, which reduces the overall power consumption of the ADC and increases the initialization time when starting an ADC conversion.
1	The analog modules stay enabled when selected as input to the ADC. Using this setting will minimize the initialization time of the ADC.

Bit 0 – ENABLE ADC Enable
This bit controls whether the ADC is enabled or not.

Value	Description
0	The ADC is disabled
1	The ADC is enabled

41.5.2 Control B

Name: CTRLB
Offset: 0x01
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 controls 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	DIV6	CLK_PER divided by 6
0x3	DIV8	CLK_PER divided by 8
0x4	DIV10	CLK_PER divided by 10
0x5	DIV12	CLK_PER divided by 12
0x6	DIV14	CLK_PER divided by 14
0x7	DIV16	CLK_PER divided by 16
0x8	DIV20	CLK_PER divided by 20
0x9	DIV24	CLK_PER divided by 24
0xA	DIV28	CLK_PER divided by 28
0xB	DIV32	CLK_PER divided by 32
0xC	DIV40	CLK_PER divided by 40
0xD	DIV48	CLK_PER divided by 48
0xE	DIV56	CLK_PER divided by 56
0xF	DIV64	CLK_PER divided by 64

41.5.3 Control C

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

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

Bits 2:0 – REFSEL[2:0] Reference Selection
This bit field controls the voltage reference for the ADC. Changing to one of the internal references will require a 40 µs initialization time.

Value	Name	Description
0x0	VDD	V_{DD}
0x1	-	Reserved
0x2	EXTVREF	External Reference VREFA for ADC0 and VREFB for ADC1
0x3	-	Reserved
0x4	1V024	Internal reference 1.024V
0x5	2V048	Internal reference 2.048V
0x6	4V096	Internal reference 4.096V
0x7	2V500	Internal reference 2.500V

41.5.4 Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: -

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

Bit 3 – WINSRC Window Mode Source
This bit controls which source is used by the Window Comparator.

Value	Name	Description
0	RESULT	ADCn.RESULT is used as the Window Comparator source
1	SAMPLE	ADCn.SAMPLE is used as the Window Comparator source

Bits 2:0 – WINCM[2:0] Window Comparator Mode
This bit field controls whether the Window Comparator is enabled and which thresholds will set the Window Comparator (WCMP) interrupt flag.
In the table below, OUTPUT is the 16-bit result or sample selected by WINSRC. WINLT and WINHT are the 16-bit low and high threshold registers, respectively.

Value	Name	Description
0x0	NONE	Window Comparator disabled
0x1	BELOW	*OUTPUT < WINLT*
0x2	ABOVE	*OUTPUT > WINHT*
0x3	INSIDE	*WINLT < OUTPUT < WINHT*
0x4	OUTSIDE	*OUTPUT < WINLT or OUTPUT >WINHT*
Other	-	Reserved

41.5.5 Control E

Name: CTRLE
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	SAMPDUR[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 – SAMPDUR[7:0] Sample Duration
This bit field selects the ADC sample duration in ADC clock (CLK_ADC) cycles. The sample duration is *SAMPDUR* + ½ CLK_ADC cycles.

41.5.6 Control F

Name: CTRLF
Offset: 0x05
Reset: 0x00
Property: -

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

Bit 5 – FREERUN Free-Running
This bit controls whether the ADC Free-Running mode is enabled or not.
Free-Running mode is not supported in Series mode.

Value	Name	Description
0x0	DISABLE	The ADC Free-Running mode is disabled
0x1	ENABLE	The ADC Free-Running mode is enabled. The first conversion is started by writing the IMMEDIATE setting to the START bit field in the Command (ADCn.COMMAND) register. In Free-Running mode, a new conversion is started as soon as the previous conversion or accumulation has been completed, which is signaled by RESRDY in the Interrupt Flags (ADCn.INTFLAGS) register.

Bit 4 – LEFTADJ Left Adjust
This bit controls whether the ADC output is left adjusted or not.

Value	Name	Description
0x0	DISABLE	The ADC output left adjustment is disabled
0x1	ENABLE	The ADC output left adjustment is enabled

Bits 2:0 – SAMPNUM[2:0] Sample Accumulation Number Select
This bit field selects the number of consecutive ADC samples accumulated automatically into the ADC Result (ADCn.RESULT) register. The most recent sample will be available in the ADC Sample (ADCn.SAMPLE) register.

Value	Name	Description
0x0	NONE	No accumulation, single sample per conversion result
0x1	ACC2	2 samples accumulated
0x2	ACC4	4 samples accumulated
0x3	ACC8	8 samples accumulated
0x4	ACC16	16 samples accumulated
0x5	ACC32	32 samples accumulated
0x6	ACC64	64 samples accumulated
Other	-	Reserved

41.5.7 Interrupt Control

Name: INTCTRL
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			TRIGOVR	SAMPOVR	RESOVR	WCMP	SAMPRDY	RESRDY
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – TRIGOVR Trigger Overrun Interrupt Enable
This bit controls whether the interrupt for a trigger overrun is enabled or not.

Value	Description
0	The Trigger Overrun interrupt is disabled
1	The Trigger Overrun interrupt is enabled

Bit 4 – SAMPOVR Sample Overwrite Interrupt Enable
This bit controls whether the interrupt for a sample overwrite is enabled or not.

Value	Description
0	The Sample Overwrite interrupt is disabled
1	The Sample Overwrite interrupt is enabled

Bit 3 – RESOVR Result Overwrite Interrupt Enable
This bit controls whether the interrupt for a result overwrite is enabled or not.

Value	Description
0	The Result Overwrite interrupt is disabled
1	The Result Overwrite interrupt is enabled

Bit 2 – WCMP Window Comparator Interrupt Enable
This bit controls whether the interrupt for the Window Comparator is enabled or not.

Value	Description
0	The Window Comparator interrupt is disabled
1	The Window Comparator interrupt is enabled

Bit 1 – SAMPRDY Sample Ready Interrupt Enable
This bit controls whether the Sample Ready interrupt is enabled or not.

Value	Description
0	The Sample Ready interrupt is disabled
1	The Sample Ready interrupt is enabled

Bit 0 – RESRDY Result Ready Interrupt Enable
This bit controls whether the Result Ready interrupt is enabled or not.

Value	Description
0	The Result Ready interrupt is disabled
1	The Result Ready interrupt is enabled

41.5.8 Interrupt Flags

Name: INTFLAGS
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			TRIGOVR	SAMPOVR	RESOVR	WCMP	SAMPRDY	RESRDY
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – TRIGOVR Trigger Overrun Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when a start trigger is received while a conversion is ongoing.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Trigger Overrun interrupt flag.

Bit 4 – SAMPOVR Sample Overwrite Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when an unread sample is overwritten in the Sample (ADCn.SAMPLE) register.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Sample Overwrite interrupt flag.

Bit 3 – RESOVR Result Overwrite Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when an unread result is overwritten in the Result (ADCn.RESULT) register.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Result Overwrite interrupt flag.

Bit 2 – WCMP Window Comparator Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when the conversion or accumulation is complete, and the thresholds match the selected window comparator source and mode, as set by WINSRC and WINCM in the Control D (ADCn.CTRLD) register.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Window Comparator interrupt flag.

Bit 1 – SAMPRDY Sample Ready Interrupt Flag
This flag is cleared by writing a '1' to it or by reading the Sample (ADCn.SAMPLE) register.
This flag is set when a conversion is complete, and a new sample is ready.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Sample Ready interrupt flag.

Bit 0 – RESRDY Result Ready Interrupt Flag
This flag is cleared by writing a '1' to it or by reading the Result (ADCn.RESULT) register.
This flag is set when a conversion or accumulation is complete, and a new result is ready.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Result Ready interrupt flag.

41.5.9 Status

Name: STATUS
Offset: 0x08
Reset: 0x00
Property: -

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

Bit 0 – ADCBUSY ADC Busy

This bit is cleared when an ADC conversion is complete and settling times related to configuration changes are finished.
This bit is set when the ADC is doing a conversion or waiting for settling times related to configuration changes.

41.5.10 Debug Control

Name: DBGCTRL
Offset: 0x09
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 the ADC will continue operation or not when in Debug mode and the CPU is halted.

Value	Description
0	The ADC will not continue operating in Debug mode when the CPU is halted. An ongoing conversion or burst accumulation will finish before the ADC stops.
1	The ADC will continue operating in Debug mode when the CPU is halted

41.5.11 Command

Name: COMMAND
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		MODE[2:0]				START[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 – MODE[2:0] Mode
This bit field controls the conversion mode for the ADC. Switching from one of the accumulation modes to a Single mode will reset the accumulator.
For more information on the operation modes, see sections *Operation Modes* and *Accumulator Test*.

Value	Name	Description
0x0	SINGLE_8BIT	Single conversion with 8-bit resolution
0x1	SINGLE_10BIT	Single conversion with 10-bit resolution
0x2	SERIES	Series with accumulation, separate trigger for every 10-bit conversion
0x3	BURST	Burst with accumulation. One trigger will run SAMPNUM 10-bit conversions in one sequence.
0x7	ACCTEST	Accumulator diagnostics mode for Functional Safety applications
Other	-	Reserved

Bits 2:0 – START[2:0] Start Conversion
This bit field starts or stops an ADC conversion, or controls how an ADC conversion will start.

Value	Name	Description
0x0	STOP	Stop an ongoing conversion
0x1	IMMEDIATE	Start a conversion immediately. The bitfield value will be set back to STOP after the conversion is completed unless Free-Running mode is enabled
0x2	MUXPOS_WRITE	Start when a write to the MUXPOS register is done
0x4	EVENT_TRIGGER	Start when an event is received by the ADC
Other	-	Reserved

Note: If the ENABLE bit in ADCn.CTRLA is '0' when writing the START bit field to IMMEDIATE, it will automatically be set to STOP.

41.5.12 Positive Input Multiplexer

Name: MUXPOS
Offset: 0x0B
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] Positive Input Multiplexer
This bit field controls which analog input is connected to the positive input of the ADC. Changing this setting may require some settling time. Refer to the *Electrical Characteristics* section for further details.

Value	Name	Description
0x00	AIN0	ADC input pin 0(1)
0x01	AIN1	ADC input pin 1(1)
0x02	AIN2	ADC input pin 2(1)
0x03	AIN3	ADC input pin 3(1)
0x04	AIN4	ADC input pin 4
0x05	AIN5	ADC input pin 5
0x06	AIN6	ADC input pin 6
0x07	AIN7	ADC input pin 7
0x10	AIN16	ADC input pin 16(1)
0x11	AIN17	ADC input pin 17(1)
0x12	AIN18	ADC input pin 18(1)
0x13	AIN19	ADC input pin 19(1)
0x14	AIN20	ADC input pin 20(1)
0x15	AIN21	ADC input pin 21(1)
0x16	AIN22	ADC input pin 22(1)
0x17	AIN23	ADC input pin 23(1)
0x18	AIN24	ADC input pin 24
0x19	AIN25	ADC input pin 25
0x1A	AIN26	ADC input pin 26
0x1B	AIN27	ADC input pin 27
0x1C	AIN28	ADC input pin 28(1)
0x1D	AIN29	ADC input pin 29(1)
0x1E	AIN30	ADC input pin 30(1)
0x1F	AIN31	ADC input pin 31(1)
0x40	GND	Ground
0x41	VDDCOREDIV4	Regulated core voltage divided by 4
0x42	TEMPSENSE	Temperature sensor(1)
0x44	VDDDIV10	V_{DD} divided by 10
0x45	VDDIO2DIV10	V_{DDIO2} divided by 10
0x48	DAC0	Digital-to-Analog Converter 0
0x49	DACREF0	AC0 DAC voltage reference
0x4A	DACREF1	AC1 DAC voltage reference
0x4B	DACREF2	AC2 DAC voltage reference
Other	-	Reserved

Note:

1. Only available in ADC0.

41.5.13 Result

Name: RESULT
Offset: 0x0C
Reset: 0x0000
Property: -

The ADCn.RESULTL and ADCn.RESULTH registers represent the 16-bit value, ADCn.RESULT. 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 is also used and writable in the Functional Safety test mode ACCTEST.

Refer to the *Output Formats* section for details on the output from this register.

Bit	15	14	13	12	11	10	9	8
	RESULT[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
	RESULT[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 – RESULT[15:8] Result byte 1
This bit field constitutes the MSB of the 16-bit register.

Bits 7:0 – RESULT[7:0] Result byte 0
This bit field constitutes the LSB of the 16-bit register.

41.5.14 Sample

Name: SAMPLE
Offset: 0x0E
Reset: 0x0000
Property: -

The ADCn.SAMPLEL and ADCn.SAMPLEH register pair represents the 16-bit value, ADCn.SAMPLE. 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 *Output Formats* section for details on the output from this register.

Bit	15	14	13	12	11	10	9	8
	SAMPLE[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
	SAMPLE[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 – SAMPLE[15:8] Sample high byte
This bit field constitutes the MSB of the 16-bit register.

Bits 7:0 – SAMPLE[7:0] Sample low byte
This bit field constitutes the LSB of the 16-bit register.

41.5.15 Window Comparator Low Threshold

Name: WINLT
Offset: 0x10
Reset: 0x0000
Property: -

This register is the 16-bit Low Threshold for the digital comparator monitoring the ADC Result or Sample (ADCn.RESULT or ADCn.SAMPLE) registers. The data format must be according to Conversion mode and left 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`.

When monitoring the ADC Result register in an accumulation mode, the window comparator thresholds are applied to the result after all accumulation, and optionally scaling, is done.

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.

41.5.16 Window Comparator High Threshold

Name: WINHT
Offset: 0x12
Reset: 0x0000
Property: -

This register is the 16-bit High Threshold for the digital comparator monitoring the ADC Result or Sample (ADCn.RESULT or ADCn.SAMPLE) registers. The data format must be according to Conversion mode and left 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`.

When monitoring the ADC Result register in an accumulation mode, the window comparator thresholds are applied to the result after all accumulation, and optionally scaling, is done.

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.

41.5.17 Temporary

Name: TEMP
Offset: 0x14
Reset: 0x00
Property: -

Temporary register for read/write operations in the (ADCn.RESULT and ADCn.SAMPLE) registers. 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 *AVR CPU* 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 bit field for read/write operations in 16-bit registers.

42. DAC - Digital-to-Analog Converter

42.1 Features

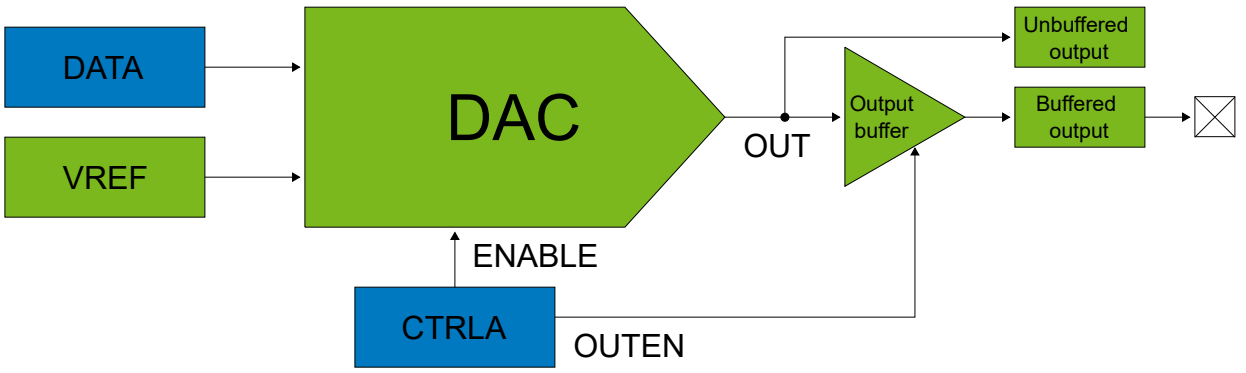
- 10-Bit Resolution
- High Drive Capabilities
- The DAC Output can be Used as Input to Other Analog Peripherals

42.2 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. Start the DAC conversion from the application by writing to the Data (DACn.DATA) register.

42.2.1 Block Diagram

Figure 42-1. DAC Block Diagram



42.2.2 Signal Description

Signal	Description	Type
OUT	DAC output	Analog

42.3 Functional Description

42.3.1 Initialization

The following steps are required to operate the DAC:

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.

42.3.2 Operation

42.3.2.1 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.

42.3.2.2 Starting a Conversion

When the ENABLE bit in the Control A (DACn.CTRLA) register is '1', a conversion starts as soon as the Data (DACn.DATA) register is written.

When the ENABLE bit in DACn.CTRLA is '0', writing to the Data register does not trigger a conversion. Instead, the conversion starts when the ENABLE bit in DACn.CTRLA is '1'.

42.3.2.3 DAC Output Voltage

The analog output voltage from the DAC peripheral is found on the DACn.OUT pin. See the *I/O Multiplexing and Considerations* section for details.

The following equation defines the DAC output voltage:

$$DACn\ Output\ Voltage = \left(\frac{DACn.DATA}{1024}\right) * (DAC\ voltage\ reference)$$

The 10-bit value entered into the DATA register must be divided by the maximum value of the register (10-bit max. value is $2^{10} = 1024$) to get the correct voltage fraction.

The maximum value we can get is when DATA = 2^{10}, the resulting voltage is equal to the selected DAC voltage reference.

The minimum result is when DATA = '0', then the output is '0'.

The step size for the voltage output is $\frac{DAC\ voltage\ reference}{1024}$ V.

42.3.2.4 DAC Output

The DAC can be used as an output to a pin and as an input to the peripherals in the table below.

DAC Output	Peripheral Input	Notes
Unbuffered	Analog-to-Digital Converter (ADC)	The peripheral is connected to the unbuffered DAC output. See section Unbuffered Output as Source For Internal Peripherals.
Buffered	-	The peripheral is connected to the DAC output pin. See section Buffered Output.

42.3.2.4.1 Unbuffered Output as Source For Internal Peripherals

The unbuffered analog output of the DAC can be used as a source for internal peripherals when the ENABLE bit in the Control A (DACn.CTRLA) register is '1'.

When using only the unbuffered analog output of the DAC, the Output Buffer Enable (OUTEN) bit in DACn.CTRLA can be '0', freeing the DAC output pin to be used by other peripherals.

42.3.2.4.2 Buffered Output

Enable the buffered analog output of the DAC by writing a '1' to the Output Buffer Enable (OUTEN) bit in the Control A (DACn.CTRLA) register. Refer to the *Electrical Characteristics* section for information about the drive capabilities of the DAC output buffer.
Note: To reduce power consumption, the DAC output pin must have its input disabled from the PORT peripheral. Refer to the *I/O Multiplexing and Considerations* section to determine which pin is used by the DAC output.

42.3.3 Sleep Mode Operation

If the Run in Standby (RUNSTDBY) bit in the Control A (DACn.CTRLA) register is '1', the DAC will continue to operate in Standby sleep mode. If the RUNSTDBY bit is '0', 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 leaves Standby sleep mode, the DAC and the output buffer (if the OUTEN bit in the Control A (DACn.CTRLA) register is '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.

42.4 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]							

42.5 Register Description

42.5.1 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.

42.5.2 DATA

Name: DATA
Offset: 0x02
Reset: 0x00
Property: -

The DACn.DATAL and DACn.DATAH register pair represents the 10-bit value DACn.DATA in the following way:

- DACn.DATA[9:2] = DACn.DATAH (can be accessed at offset + `0x01`)
- DACn.DATA[1:0] = DACn.DATAL (can be accessed at original offset)

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 that the DAC peripheral will convert to an analog value and send to the DAC output.

43. ZCD - Zero-Cross Detector

43.1 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

43.2 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.

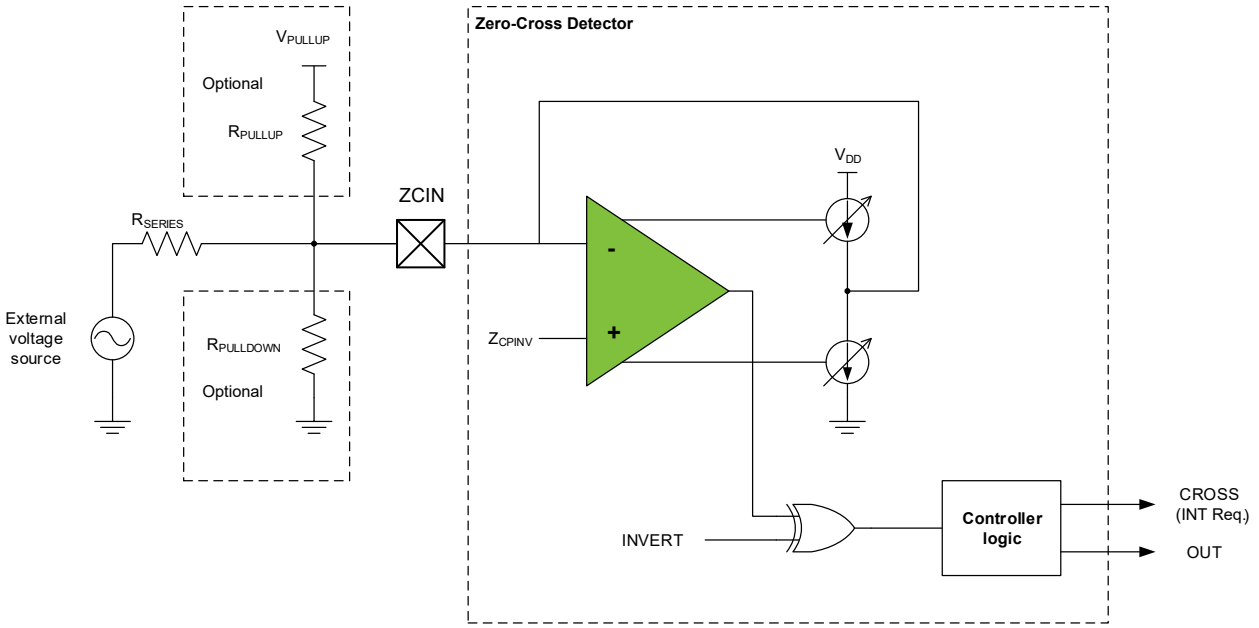
The connection from the ZCD input pin (ZCIN) to the alternating voltage must be made through a series current-limiting resistor (R_{SERIES}). The ZCD applies either a current source or sink to the ZCD input pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward 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

43.2.1 Block Diagram

Figure 43-1. ZCD Block Diagram



43.2.2 Signal Description

Signal	Description	Type
ZCIN	Input	Analog
OUT	Output	Digital

43.3 Functional Description

43.3.1 Initialization

For basic operation, follow these steps:

1. Configure the desired input pin in the PORT peripheral as an analog pin with the digital input buffer disabled. Internal pull-up and pull-down resistors must also be disabled.
2. Optional: Enable the output pin by writing a '1' to the Output Enable (OUTEN) bit in the Control A (ZCDn.CTRLA) register.
3. 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.

43.3.2 Operation

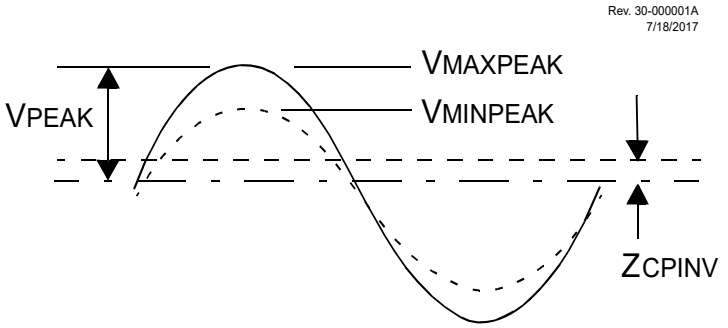
43.3.2.1 External Resistor Selection

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 must 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 *Handling V_{PEAK} Variations* section below).

Equation 43-1. External Resistor

$$R_{SERIES} = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

Figure 43-2. External Voltage Source



43.3.2.2 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. Writing the INVERT bit to '1' in the ZCDn.CTRLA register can reverse the STATE flag polarity. The INVERT bit will also affect ZCD interrupt polarity.

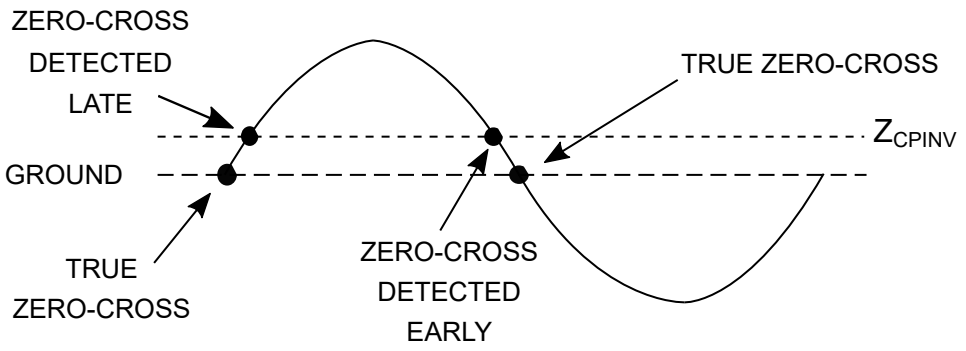
43.3.2.3 Correction for Z_{CPINV} Offset

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.

43.3.2.3.1 Correction By Offset Current

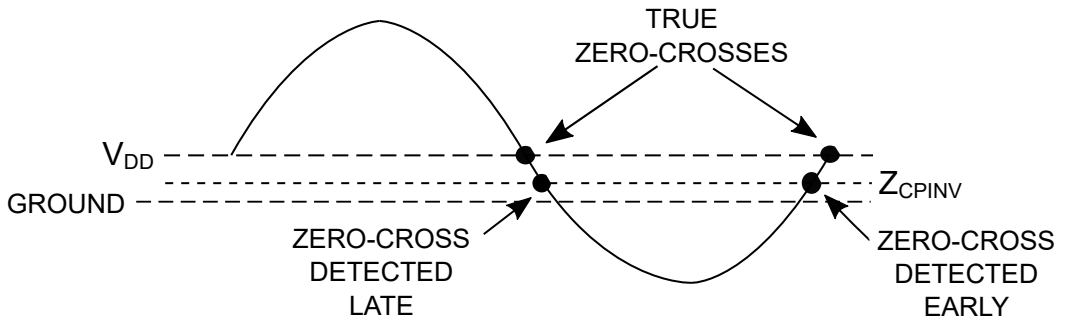
When the alternating waveform is referenced to the ground, as shown in the figure below, the zero-cross is detected too late as the waveform rises and too early as the waveform falls.

Figure 43-3. 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 43-4. 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 43-2. ZCD Event Offset

When the External Voltage source is referenced to ground:

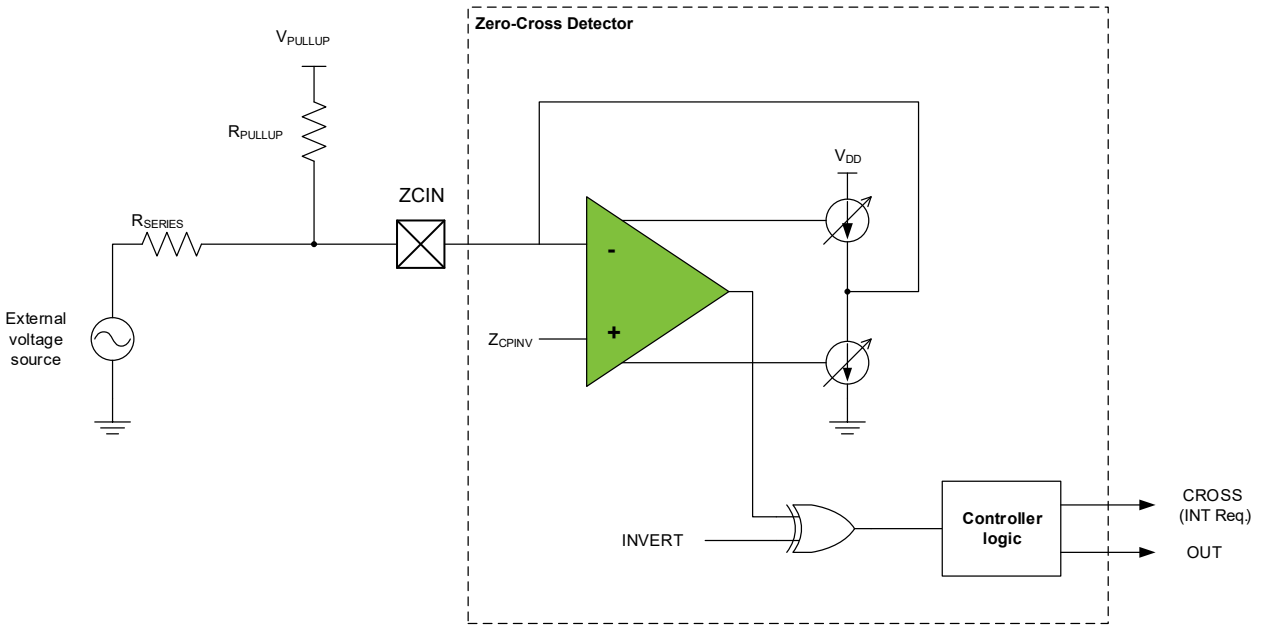
$$T_{offset} = \frac{\sin^{-1}\left(\frac{Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

When the External Voltage source is referenced to V_{DD}:

$$T_{offset} = \frac{\sin^{-1}\left(\frac{V_{DD} - Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

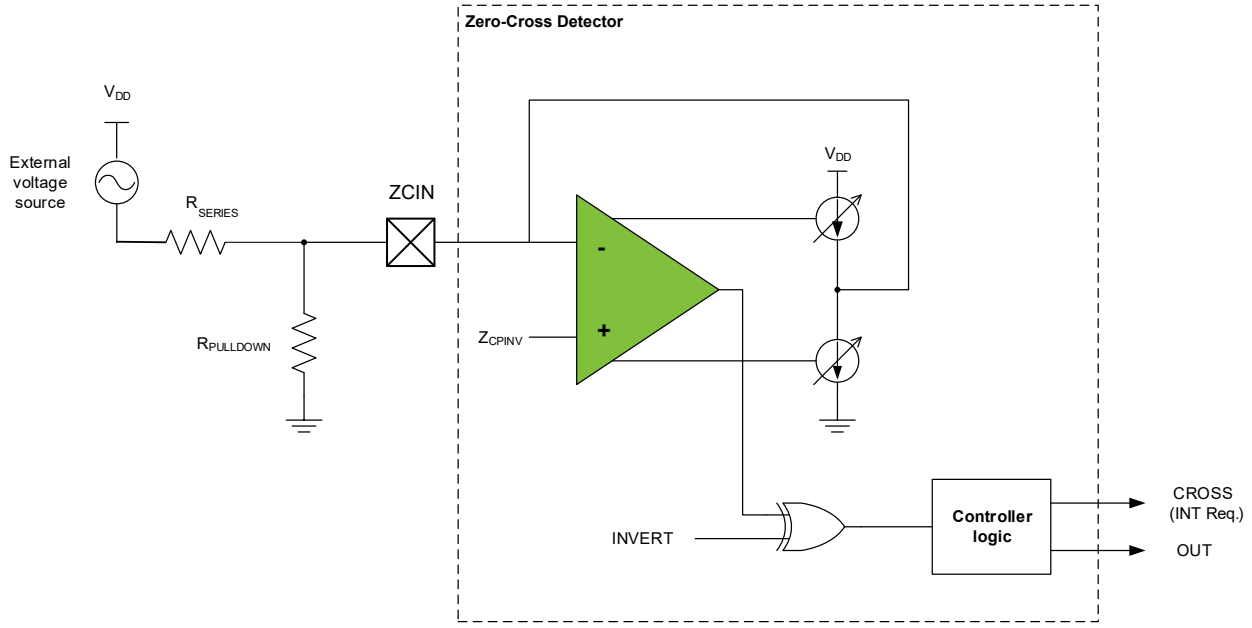
This offset time can be compensated by adding a pull-up or pull-down biasing resistor to the ZCD input pin. A pull-up resistor is used when the external voltage source is referenced to ground, as shown in the figure below.

Figure 43-5. 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 43-6. 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 43-3. ZCD Pull-Up/Pull-Down Resistor

When the External Voltage source is referenced to ground:

$$R_{pullup} = \frac{R_{SERIES}\left(V_{pullup} - Z_{CPINV}\right)}{Z_{CPINV}}$$

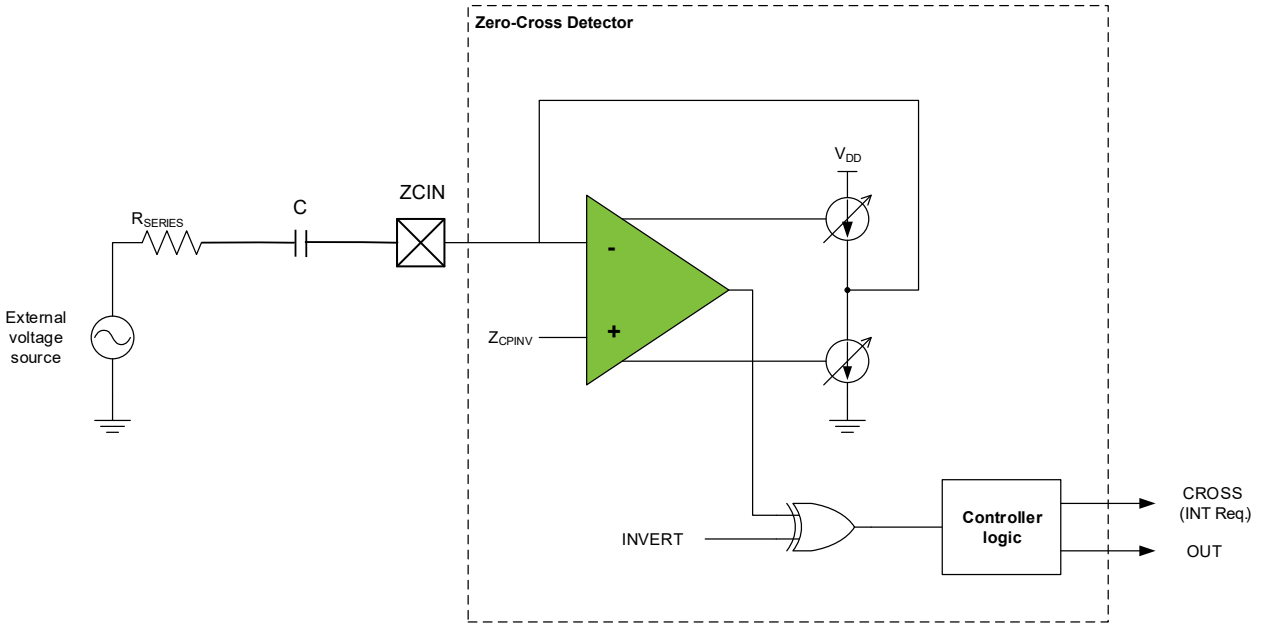
When the External Voltage source is referenced to V_{DD}:

$$R_{pulldown} = \frac{R_{SERIES}(Z_{CPINV})}{(V_{DD} - Z_{CPINV})}$$

43.3.2.3.2 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 43-7. 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 43-4. 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

$T_Φ$ = 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 43-5. R-C Calculation Example

$$V_{rms} = 120$$

$$V_{PEAK} = V_{rms} \times \sqrt{2} = 169.7$$

$$f = 60\ Hz$$

$$C = 0.1\ \mu F$$

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}} = \frac{169.7}{3 \times 10^{-4}} = 565.7\ k\Omega$$

$$X_C = \frac{1}{2\pi f C} = \frac{1}{2\pi \times 60 \times 10^{-7}} = 26.53\ k\Omega$$

$$R = \sqrt{Z^2 - {X_C}^2} = 565.1\ k\Omega\ (computed)$$

$$R_a = 560\ k\Omega\ (used)$$

$$Z_R = \sqrt{{R_a}^2 + {X_C}^2} = 560.6\ k\Omega$$

$$I_{PEAK} = \frac{V_{PEAK}}{Z_R} = 302.7 \times 10^{-6} A$$

$$V_C = X_C \times I_{PEAK} = 8.0\ V$$

$$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right) = 0.047 radians$$

$$T_\Phi = \frac{\Phi}{2\pi f} = 125.6 \mu s$$

43.3.2.4 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 must 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 43-6. Series Resistor for External Voltage Range

$$R_{SERIES} = \frac{V_{MAXPEAK} + V_{MINPEAK}}{7 \times 10^{-4}}$$

43.3.3 Events

The ZCD can generate the following events:

Table 43-1. 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 *EVSYS - Event System* section for more details about event types and Event System configuration.

43.3.4 Interrupts

Table 43-2. 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.

43.3.5 Sleep Mode Operation

In Idle sleep mode, the ZCD will continue to operate as ordinary.

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.

43.4 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

43.5 Register Description

43.5.1 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.

43.5.2 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

43.5.3 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, including a three-cycle synchronizer delay.

Bit 0 – CROSSIF Cross Interrupt Flag
This bit 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.

44. UPDI - Unified Program and Debug Interface

44.1 Features

- UPDI One-Wire Interface for External Programming and On-Chip-Debugging (OCD)
  - Uses the $\overline{\text{RESET}}$ pin to enable the UPDI function, and one UPDI pin of the device for programming
  - 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
- Programming and Debug Interface Disable (PDID) Security Functionality
  - UPDI access can be limited by PDICFG and LOCK fuses
  - Bootloader still negotiating firmware updates

44.2 Overview

The Unified Program and Debug Interface (UPDI) is a proprietary interface for external programming and OCD of a device.

The UPDI supports the 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 UPDI pin for data reception and transmission. The clocking of the PHY layer is done by a dedicated internal oscillator.

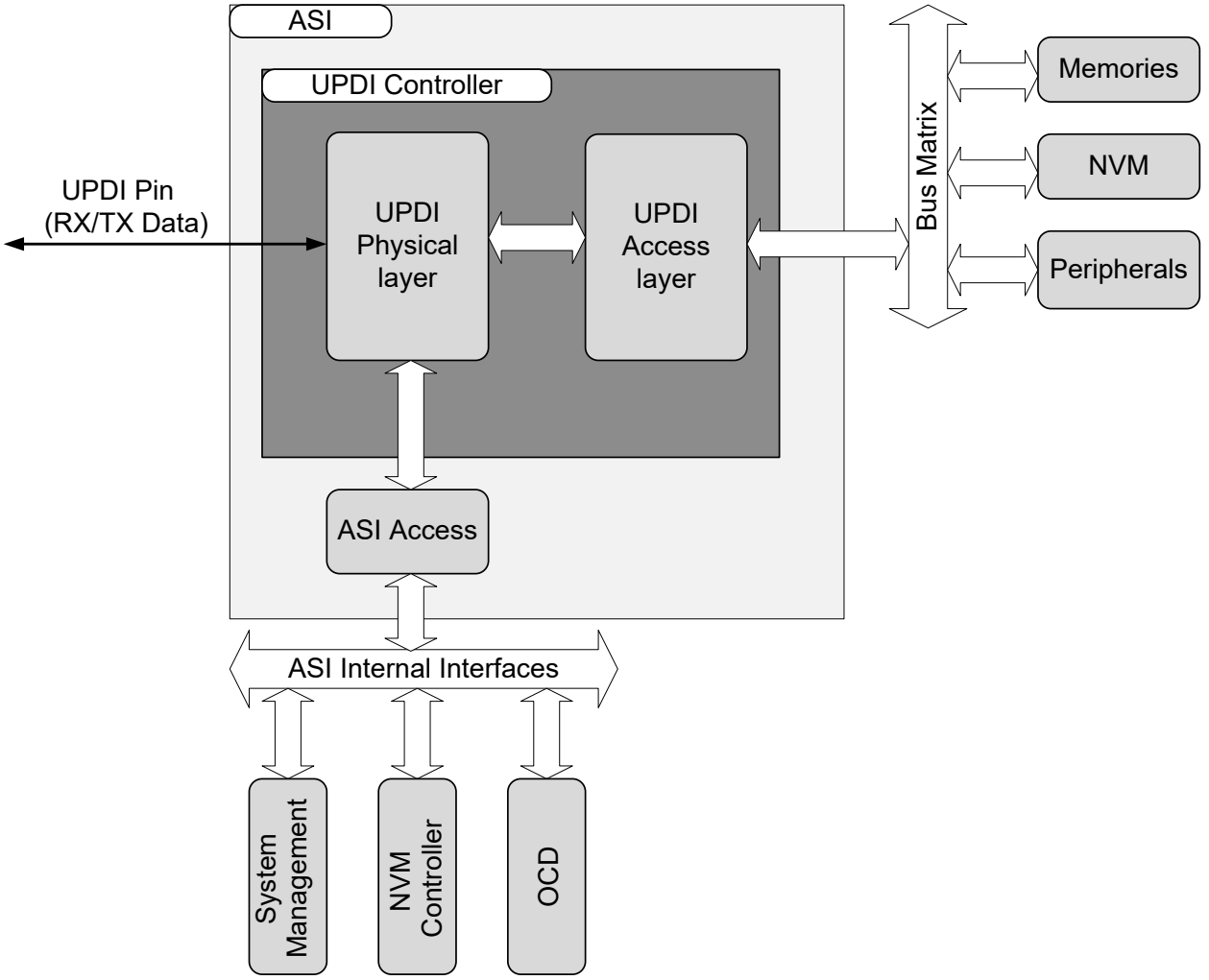
The ACC layer is the interface between the UPDI and the connected bus matrix. This layer grants access via the UPDI interface to the bus matrix with memory-mapped access to system blocks such as memories, NVM, and peripherals.

The Asynchronous System Interface (ASI) provides direct interface access to select features in the OCD, NVM, and System Management systems, which gives the debugger direct access to system information without requesting bus access.

The UPDI access can be limited by PDICFG and LOCK fuses, while the bootloader can still negotiate firmware updates.

44.2.1 Block Diagram

Figure 44-1. UPDI Block Diagram



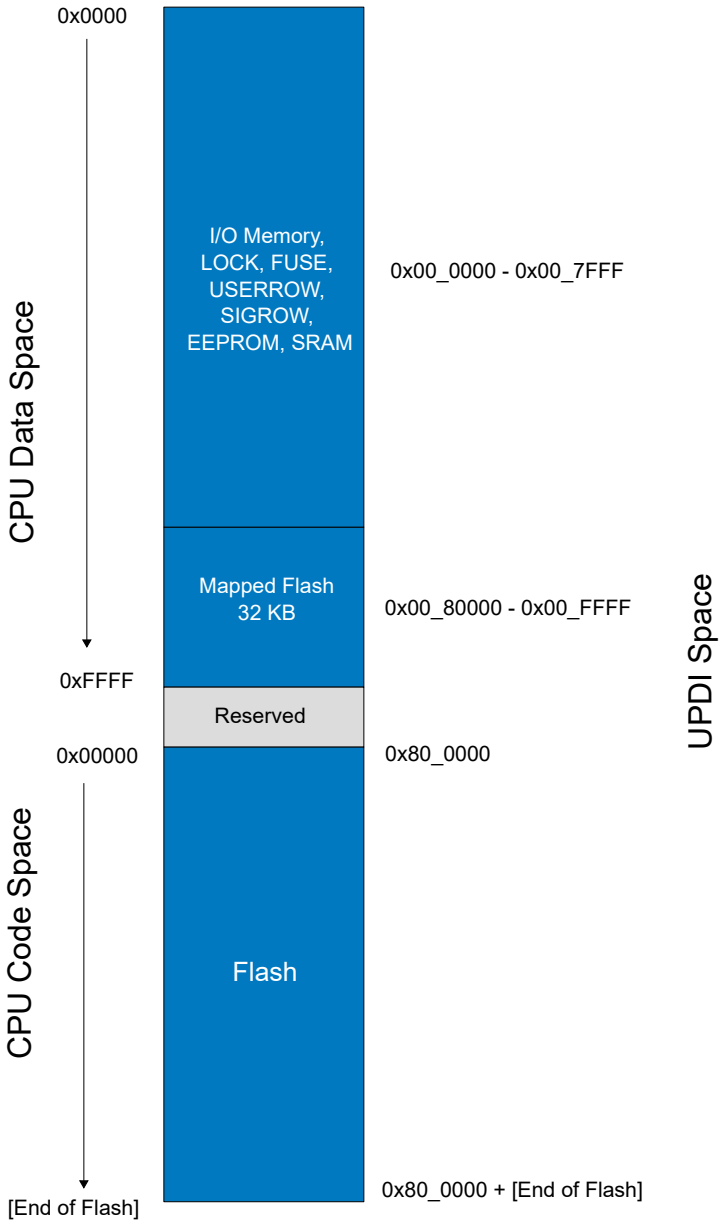
44.2.2 Addressing the Program Memory Space

In the CPU data space, the I/O memory, the fuses, EEPROM and SRAM are located at addresses from 0x0000 to 0x7FFF. In addition, a section of the Flash memory (up to 32 KB) can be mapped into the addresses from 0x8000 to 0xFFFF. These addresses (0x0000 - 0xFFFF) are also valid for access by the UPDI peripheral.

The CPU code space, i.e., the *entire* Flash memory, can be accessed by the CPU using the `LPM/SPM` instructions, starting at the relative address 0x0000. For access by UPDI, the CPU data space and the

CPU code space are virtually one continuous address space, and the code space always starts at the offset address 0x80_0000.

Figure 44-2. Memory Map, As Seen From The UPDI

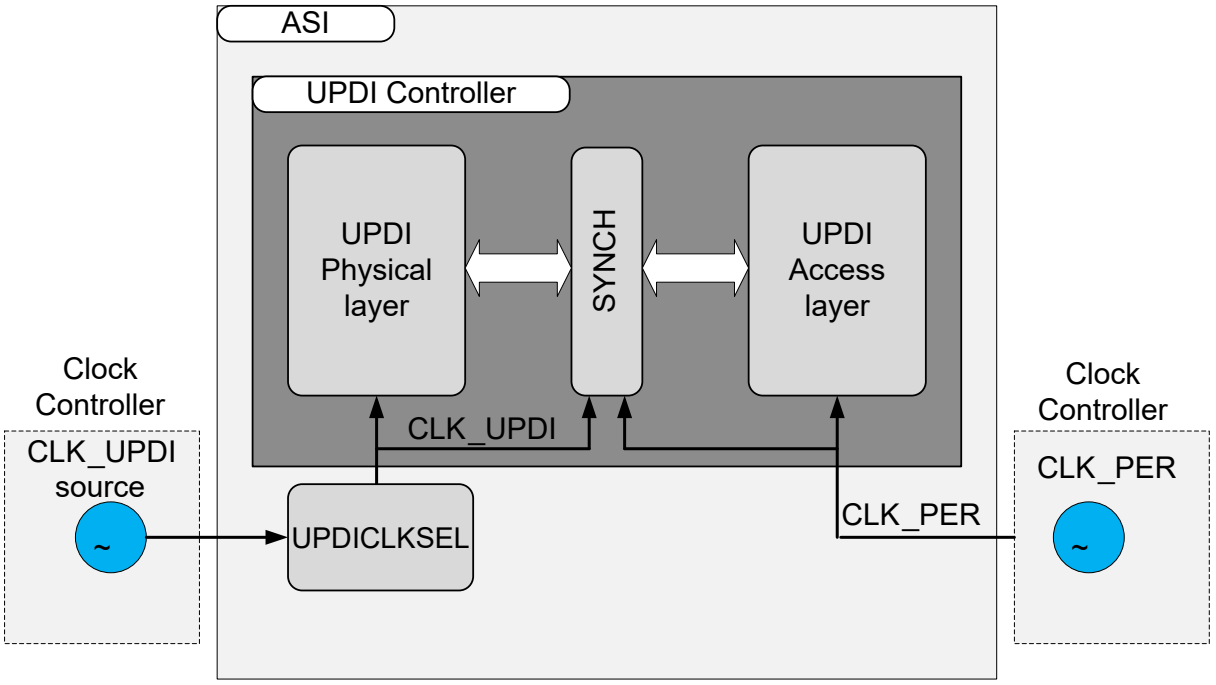


See the *Memories* sections for more details and exact addresses of the memory areas in a given device.

44.2.3 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 (UPDICLKSEL) bit field in the ASI Control A (UPDI.ASI_CTRLA) register.

Figure 44-3. UPDI Clock Domains



44.2.4 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:

- 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

44.2.5 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.

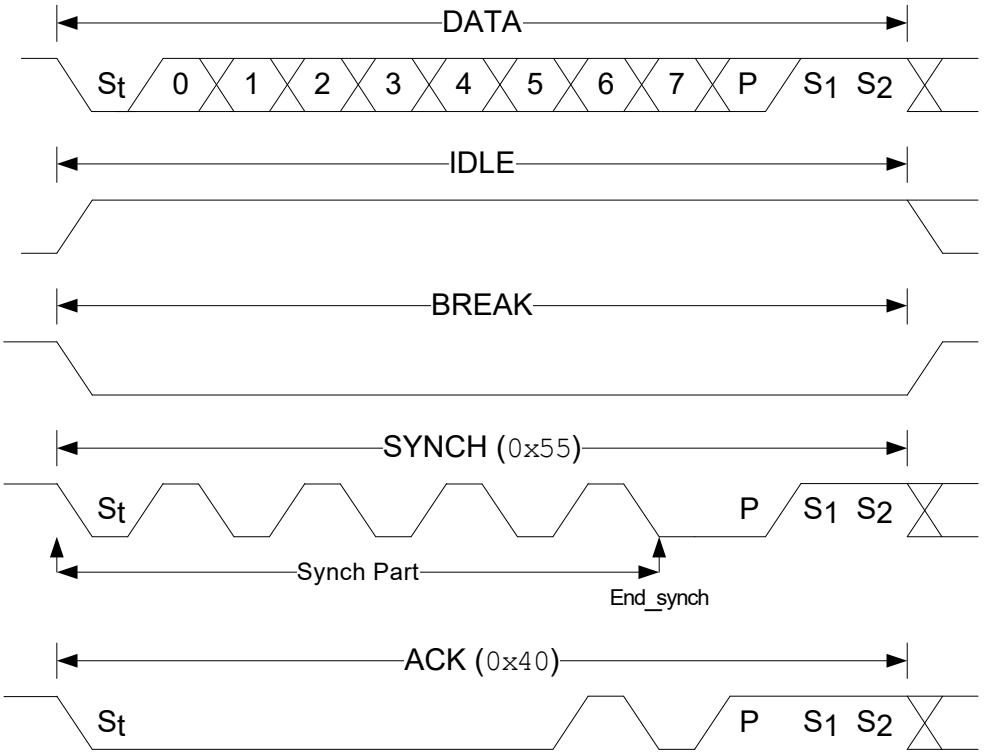
Function	Pin Name
UPDI	UPDI / PF7

44.3 Functional Description

44.3.1 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 44-4. 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	IDLE is a specific frame that consists of at least 12 high bits, which is the same as keeping the transmission line in an Idle state
BREAK	BREAK is a specific frame that consists of at least 12 low bits. It is used to reset the UPDI back to its default state and is typically used for error recovery.
SYNCH	The Baud Rate Generator uses the SYNCH frame 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 `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.

44.3.1.1 UPDI UART

The communication is initiated from the debugger/programmer side. 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 *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 (UPDICLKSEL) 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. It is recommended that the clock frequency does not run faster than

the required frequency for the desired baud rate. The default UPDICLKSEL setting after Reset and enable is 4 MHz. Any other clock output selection is only recommended when the BOD is at the highest level. For all other BOD settings, the default 4 MHz selection is recommended.

Table 44-1. Recommended UART Baud Rate Based on UPDICLKSEL Setting

UPDICLKSEL[1:0]	Max. Recommended Baud Rate	Min. Recommended Baud Rate
0x0 (32 MHz)	1.8 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 Table 44-2.

Table 44-2. 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

44.3.1.2 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 an 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, which changes the baud rate range of the UPDI, according to the *Recommended UART Baud Rate Based on UPDICLKSEL Setting* table above.

44.3.1.2.1 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
$65535 \times 250\ \text{ns} = 16.4\ \text{ms/byte} = 16.4\ \text{ms}/8\ \text{bits} = 2.05\ \text{ms/bit}.$

This gives a worst-case BREAK frame duration of $2.05\ \text{ms} \times 12\text{bits} \approx 24.6\ \text{ms}$ 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 Table 44-3.

Table 44-3. Recommended BREAK Character Duration

UPDICLKSEL[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

44.3.1.3 SYNCH Character

The SYNCH character has eight bits and follows the regular UPDI frame format. It has a fixed 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.

44.3.1.3.1 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.

44.3.1.4 Special Considerations for Programming

Any reset source will also reset the UPDI on a locked device. The reset may be triggered during boot of the device, which results in a repeating reset loop. Use a UPDI handshake mechanism to access a device in this state.

Important: Both the UPDI pin and the $\overline{\text{RESET}}$ pin must be connected to the programmer/debugger to use the UPDI handshake mechanism.

The UPDI handshake procedure:

1. Drive the UPDI line low and hold it low.
2. Pull the $\overline{\text{RESET}}$ line low for the minimum reset pulse width and release it.
3. Wait for at least 40 ms so the device can reach an initial state.
4. Release the UPDI line to its UART IDLE state.
5. Issue a negative pulse to the UPDI line.
6. Send a CHIP ERASE key to unlock the device and poll for successful completion.
7. Toggle power on the device.

Important: Steps 4 to 6 must be completed within a 64 ms time window.

NOTICE After performing the chip erase, the lock bits on the device will be cleared. However, any reset source will continue to reset the UPDI until the next Power-On Reset happens.

44.3.2 Operation

The UPDI must be enabled before the UART communication can start.

However, PDICFG and LOCK fuses can limit the UPDI access, while the bootloader can still negotiate firmware updates.

44.3.2.1 UPDI Enabling

44.3.2.1.1 One-Wire Enable

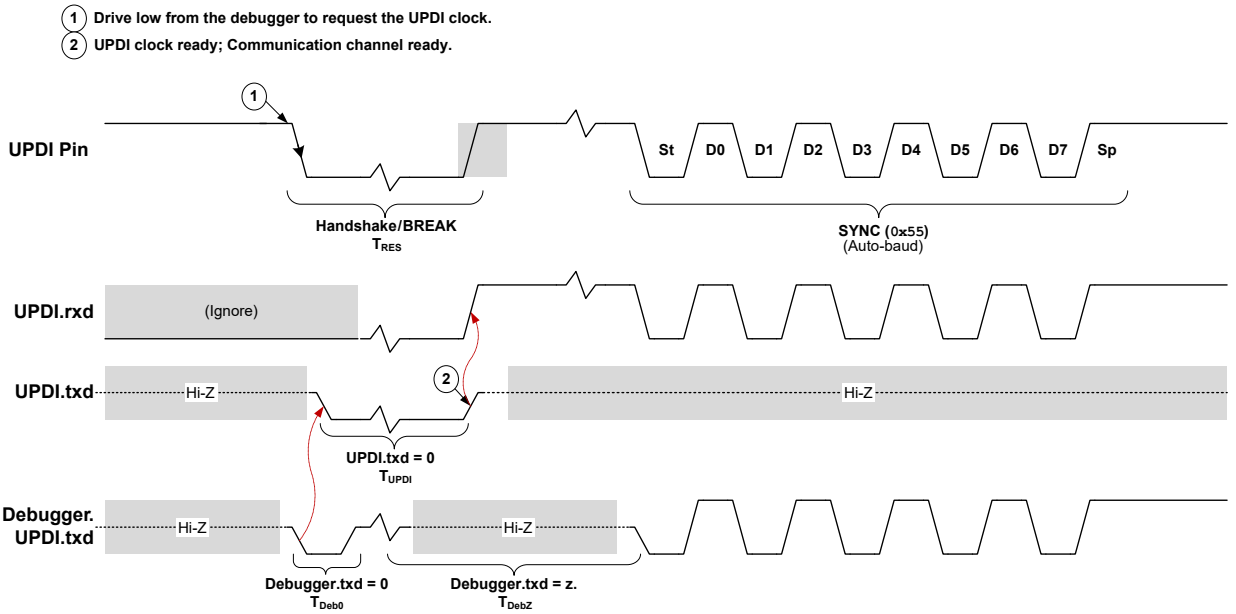
The UPDI pin has a constant pull-up when enabled.

Follow this sequence to enable the UPDI:

1. Drive the UPDI pin low for more than 200 ns, and release it.
   By this, a connected programmer will initiate the start-up sequence:
   - An edge detector starts driving the UPDI pin low, so when the programmer releases the line, it will stay low
   - The UPDI clock is started. The UPDI will continue to drive the line low until the clock is stable and ready for the UPDI to use
     The expected arrival time for the clock will depend on the oscillator implementation regarding the accuracy, overshoot, and readout of the oscillator calibration
   - The data line will be released by the UPDI and pulled high when the oscillator is ready and stable
2. Poll the UPDI pin to detect when the pin transitions to high again.
   This transition indicates that the edge detector has released the pin (pull-up), and the UPDI can receive a SYNCH character.
3. Send a SYNCH character `0x55`.
   After a successful SYNCH character transmission, the first instruction frame can be transmitted.
4. Send the NVMPROG key using the `KEY` instruction.
   Sending this key clears the lock bits, and the Programming Start (PROGSTART) bit in the ASI_SYS_STATUS is set. The device is now prepared for programming.
5. After the programming is finished, reset the UPDI by writing the UPDI Disable (UPDIDIS) bit in the Control B (UPDI.CTRLB) register to '1' using the `STCS` instruction.
   Disabling the UPDI and hence, the accompanying clock request, will reduce power consumption.

The timing of the enable sequence is shown in Figure 44-5, 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.

Figure 44-5. UPDI Enable Sequence



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.

When the debugger detects that the line is high, the initial SYNCH character `0x55` must be transmitted to synchronize the UPDI communication data rate. If the Start bit of the SYNCH character is not sent within maximum T_{DebZ}, the UPDI will disable itself, and the UPDI enabling sequence must be reinitiated. If the timing is violated, the UPDI is disabled to avoid unintentional enabling of the UPDI. See Disable During Start-Up for more details.

Note: The actual values for T_{RES}, T_{UPDI}, T_{Deb0}, and T_{DebZ} can be found in the *Electrical Characteristics* section.

44.3.2.2 UPDI Disabling

44.3.2.2.1 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.

44.3.2.2.2 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, which causes increased power consumption for the application.

44.3.2.3 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,

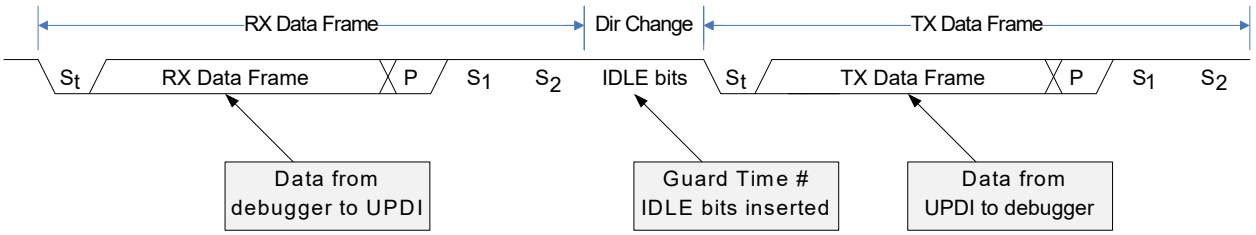
except when a BREAK character is received. The following procedure must always be applied when recovering from an Error condition.

1. Send a BREAK character. See the *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.

44.3.2.4 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 the 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 44-6. UPDI Direction Change by Inserting Idle Bits



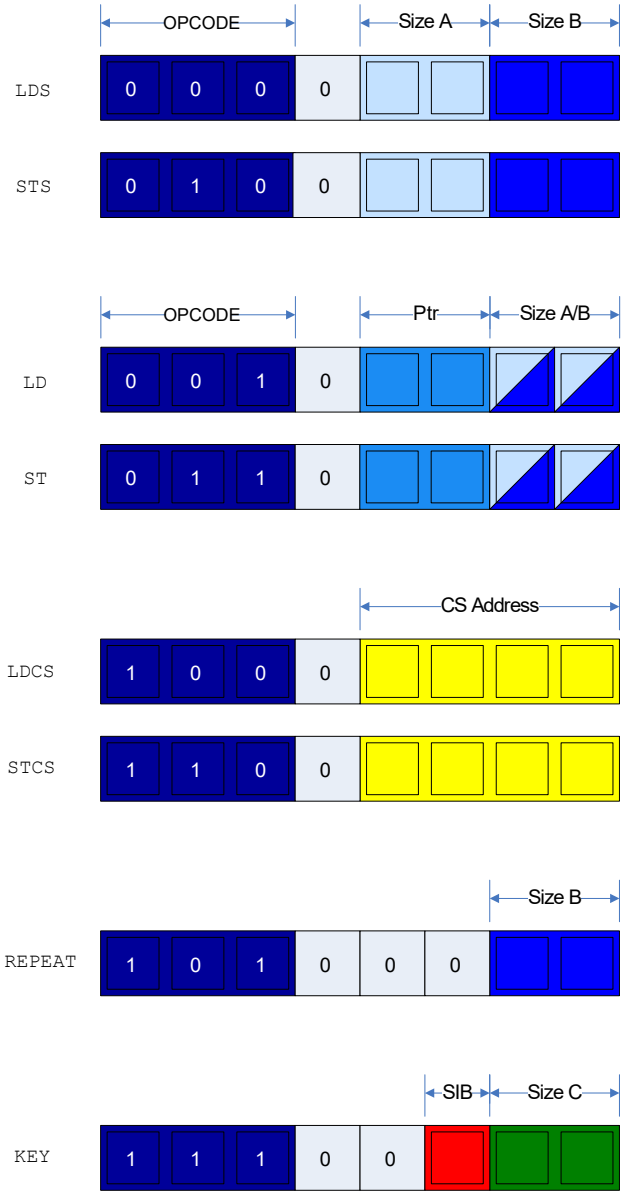
The UPDI guard time is the minimum Idle time that the connected debugger will experience when waiting for data from the UPDI. The maximum Idle time is the same as time-out. When the synchronization time plus the data bus accessing time is longer than the guard time, the Idle time before a transmission will be more than the expected guard time.

It is recommended to always use the insertion of a minimum of two Guard Time bits on the UPDI side and one guard time cycle insertion from the debugger side.

44.3.3 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 the *UPDI UART* section for information about setting the baud rate for the transmission. Figure 44-7 gives an overview of the UPDI instruction set.

Figure 44-7. 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	1 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	*(ptr--)

Size B - Data Size		
0	0	1 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

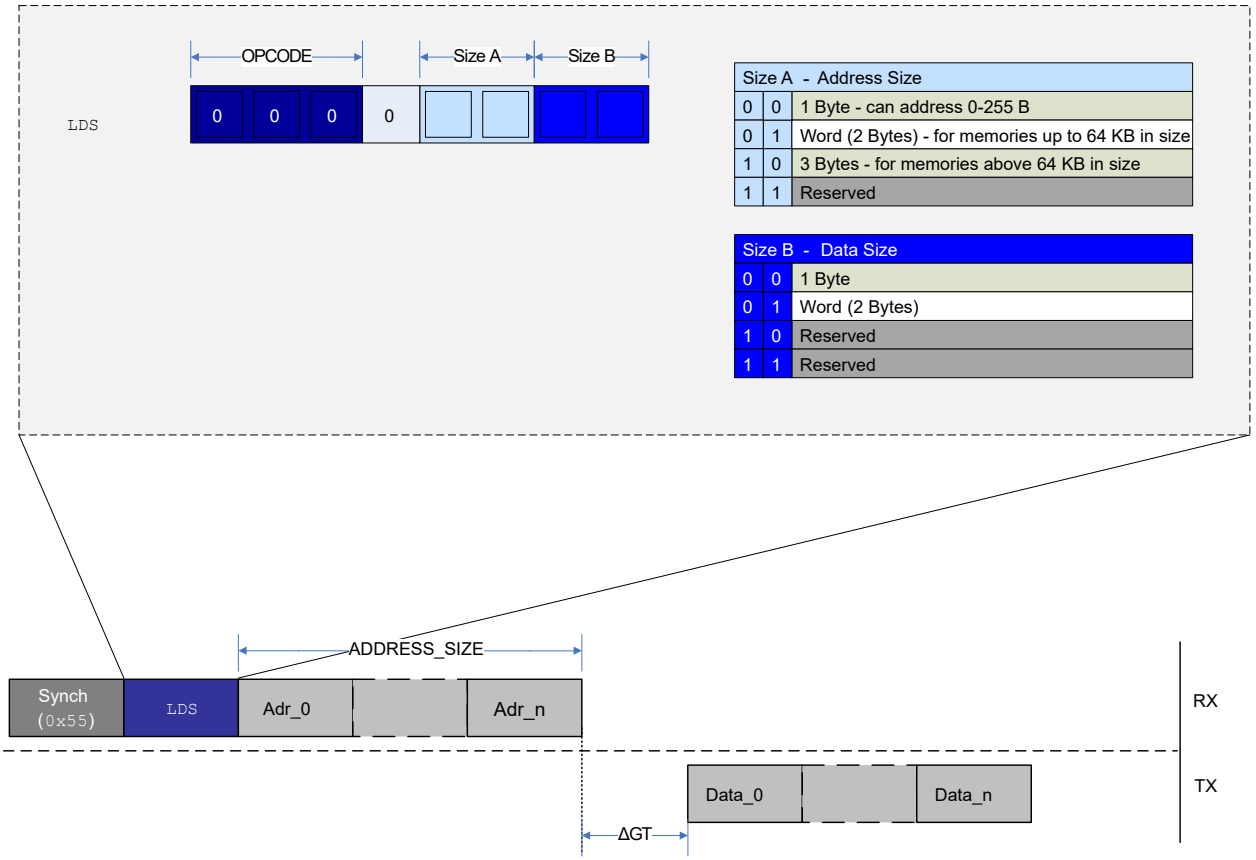
44.3.3.1 LDS - Load Data from Data Space Using Direct Addressing

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 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 44-8. LDS Instruction Operation



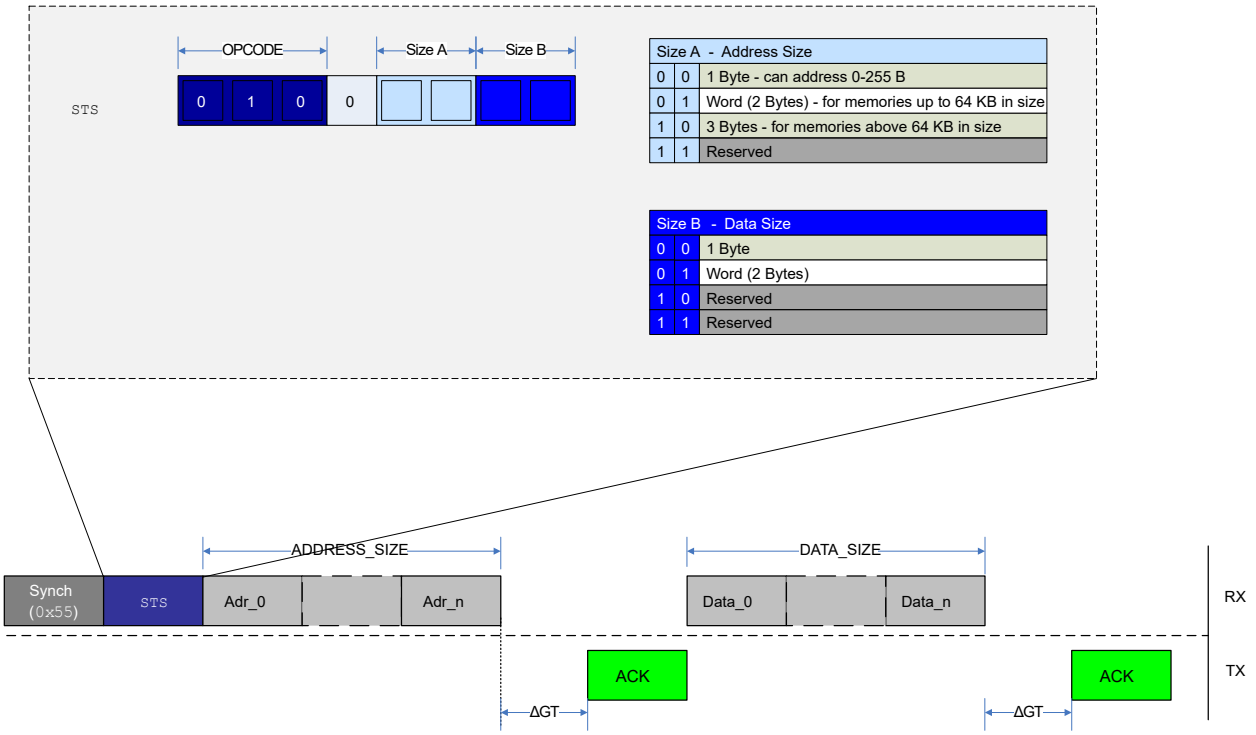
When the instruction is decoded and the address byte(s) are received as dictated by the decoded instruction, the DL layer will synchronize all required information to the ACC layer. This will handle the bus request and synchronize data buffered from the bus back to the DL layer, which will create a synchronization delay that must be taken into consideration upon receiving the data from the UPDI.

44.3.3.2 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 is given by the size fields presented in Figure 44-9. The maximum size for both address and data is 32 bits.

The STS supports repeated memory access when combined with the REPEAT instruction.

Figure 44-9. `STS` Instruction Operation



The transfer protocol for an `STS` instruction is depicted in Figure 44-9, 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.

44.3.3.3 `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 44-10. LD Instruction Operation

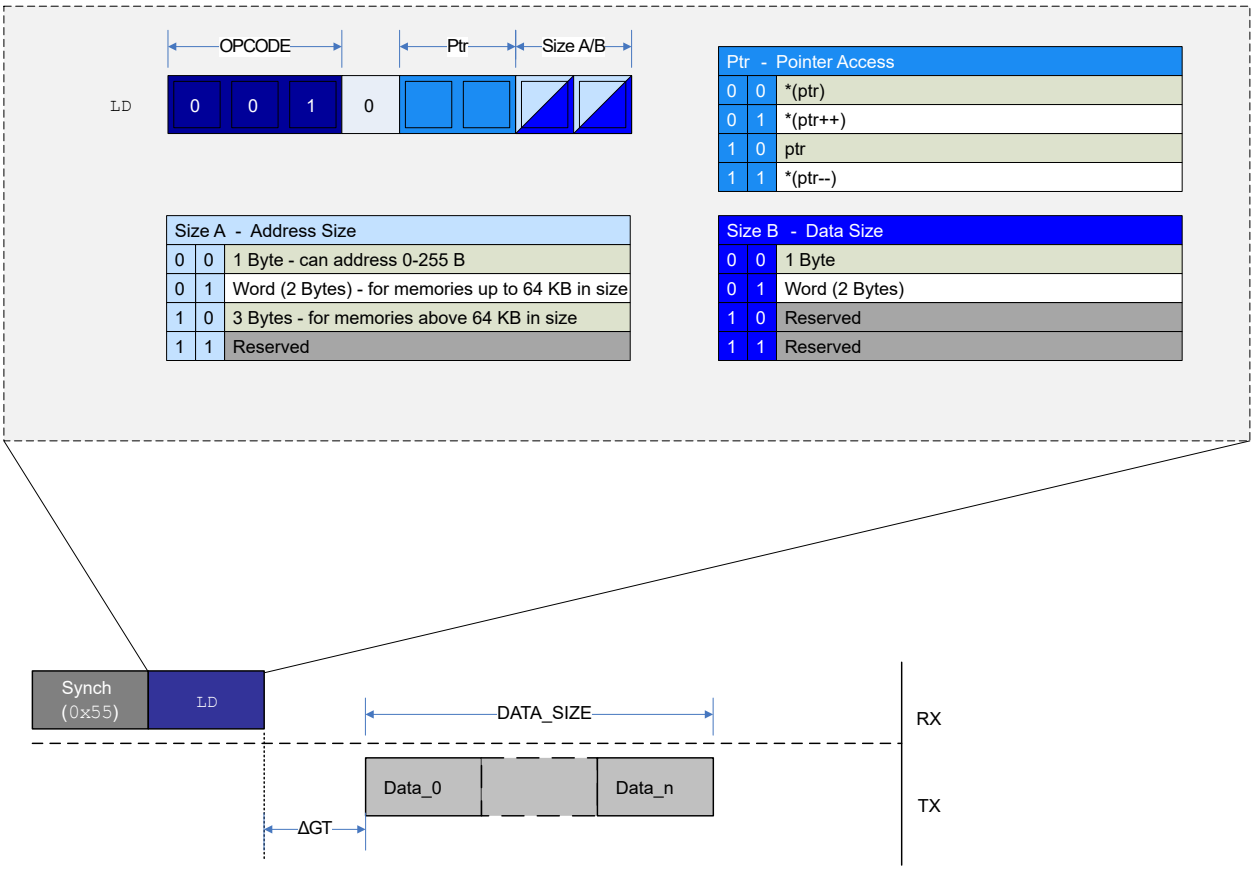


Figure 44-10 shows an example of a typical LD sequence, where the data are received after the Guard Time (GT) period. Loading data from the UPDI Pointer register follows the same transmission protocol.

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.

44.3.3.4 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 44-11. ST Instruction Operation

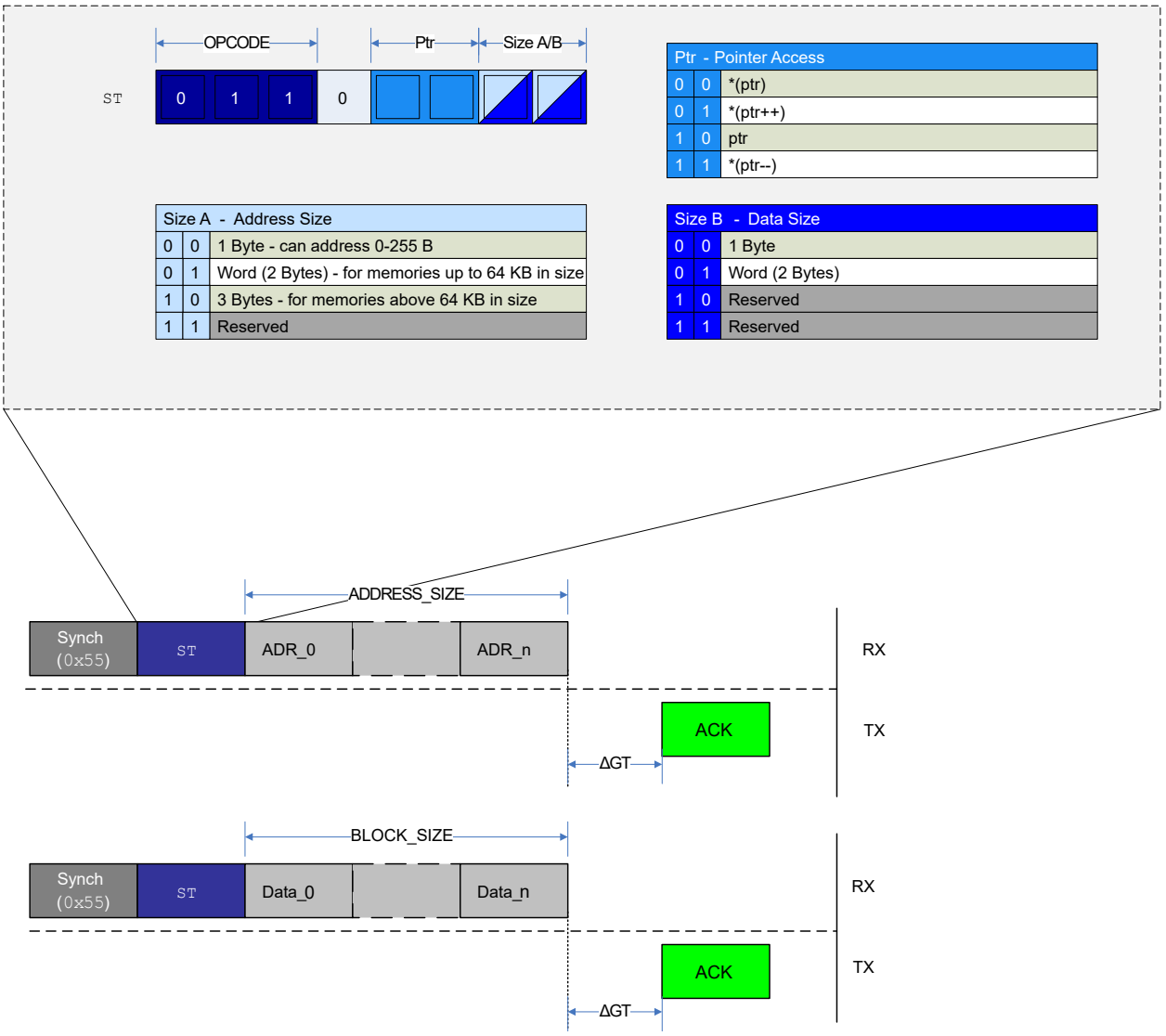


Figure 44-11 gives an example of an ST instruction to the UPDI Pointer register and the storage of regular data. A SYNCH character is sent before each instruction. In both cases, an Acknowledge (ACK) is sent back by the UPDI if the ST instruction was successful.

The next procedure has to be followed to write the UPDI Pointer register:

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.

44.3.3.5 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 44-12. LDCS Instruction Operation

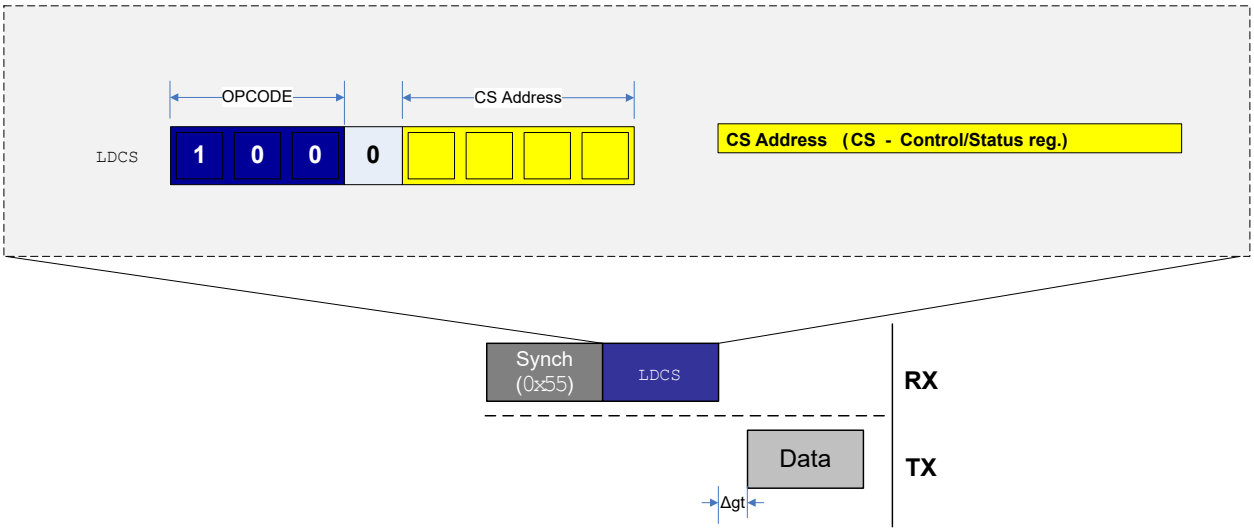


Figure 44-12 shows a typical example of LDCS data transmission. A data byte from the LDCS is transmitted from the UPDI after the guard time is completed.

44.3.3.6 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 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 44-13. STCS Instruction Operation

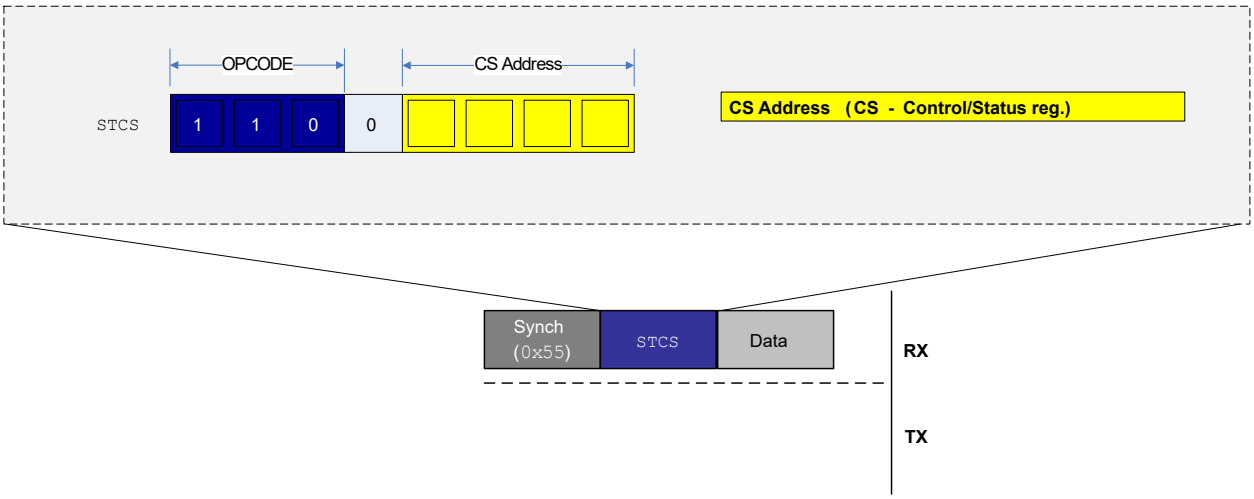


Figure 44-13 shows the data frame transmitted after the SYNCH character and the instruction frames. The STCS instruction byte can immediately be followed by the data byte. There is no response generated from the STCS instruction, as is the case for the ST and STS instructions.

44.3.3.7 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 for 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 44-14. REPEAT Instruction Operation Used with ST Instruction

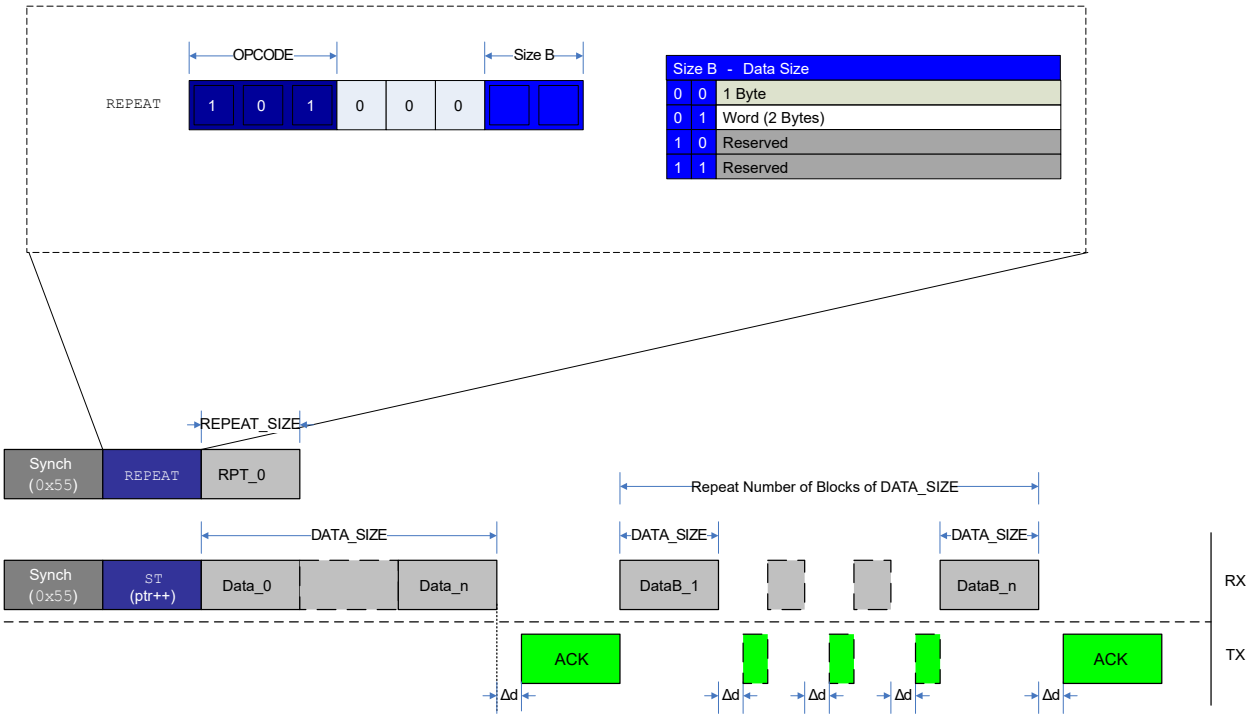
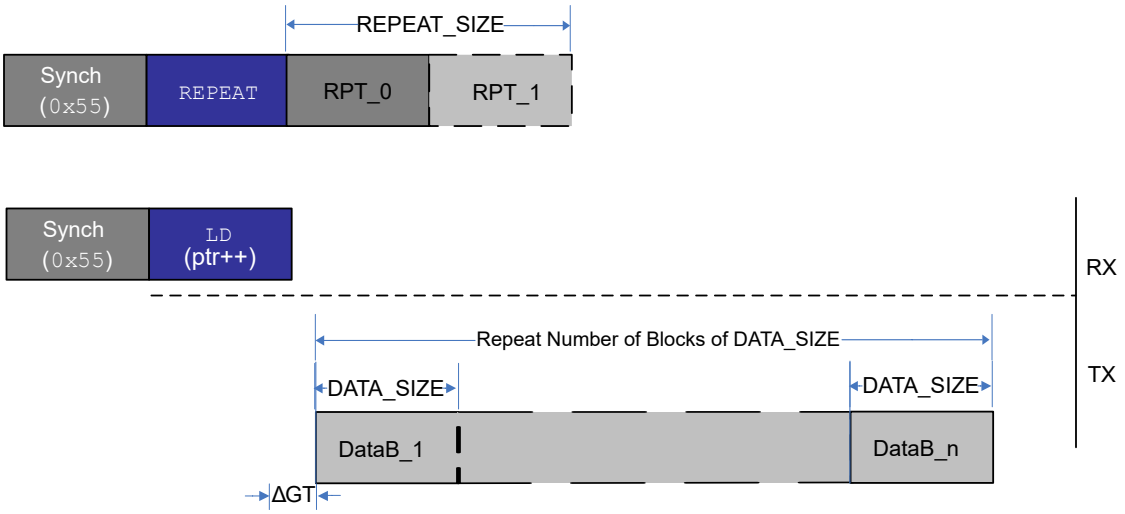


Figure 44-14 gives an example of a repeat operation with an ST instruction using pointer post-increment operation. After the REPEAT instruction is sent with RPT_0 = *n*, the first ST instruction is issued with SYNCH and instruction frame. The next *n* ST instructions are executed by only sending data bytes according to the ST operand DATA_SIZE and maintaining the Acknowledge (ACK) handshake protocol.

Figure 44-15. 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).

44.3.3.8 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 the *Key Activation Overview* table in the *Enabling of Key Protected Interfaces* section 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 44-16. KEY Instruction Operation

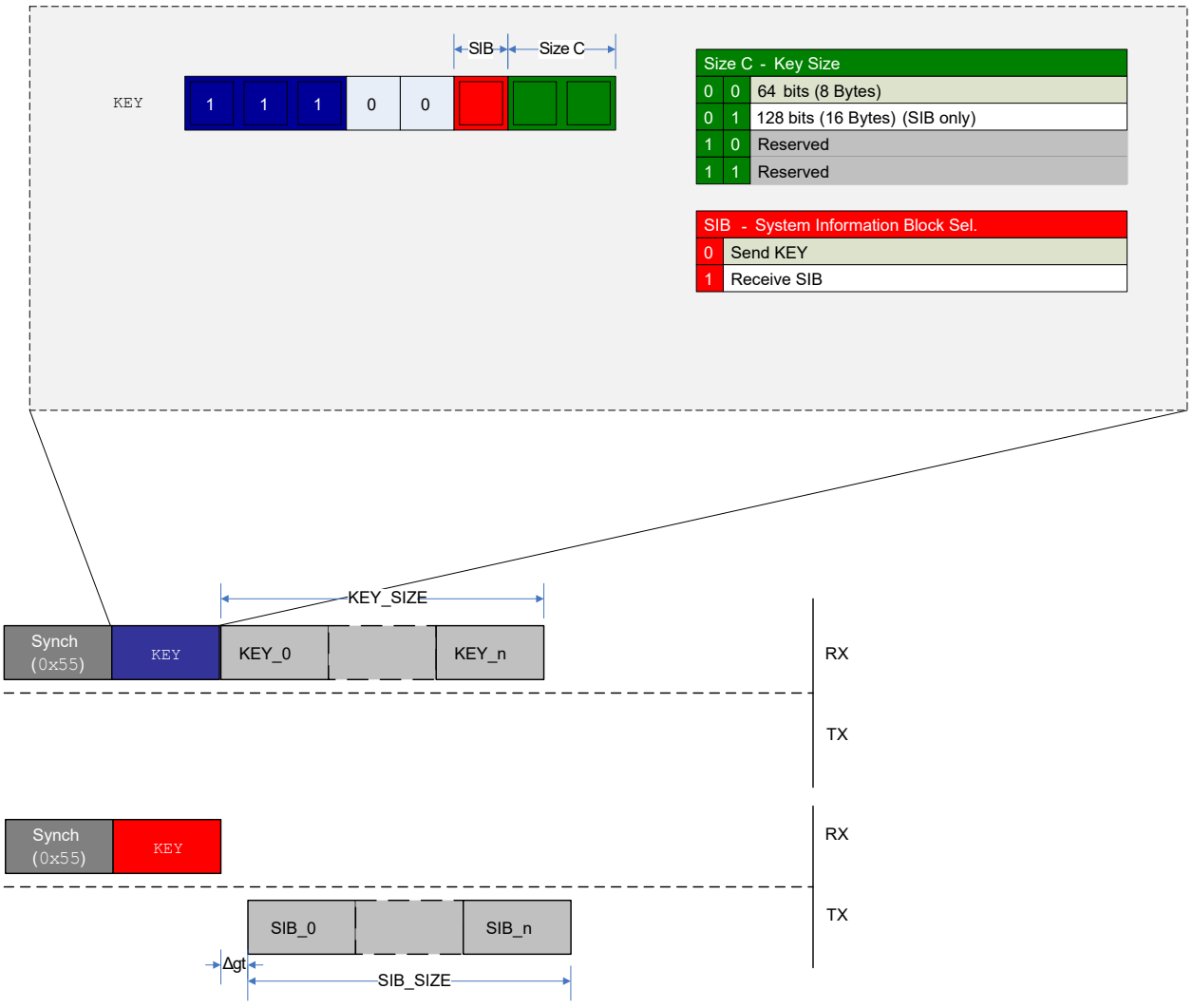


Figure 44-16 shows the transmission of a key and the reception of a SIB. In both cases, the Size C (SIZE_C) field in the operand determines the number of frames being sent or received. There is no response after sending a KEY to the UPDI. When requesting the SIB, data will be transmitted from the UPDI according to the current guard time setting.

44.3.4 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

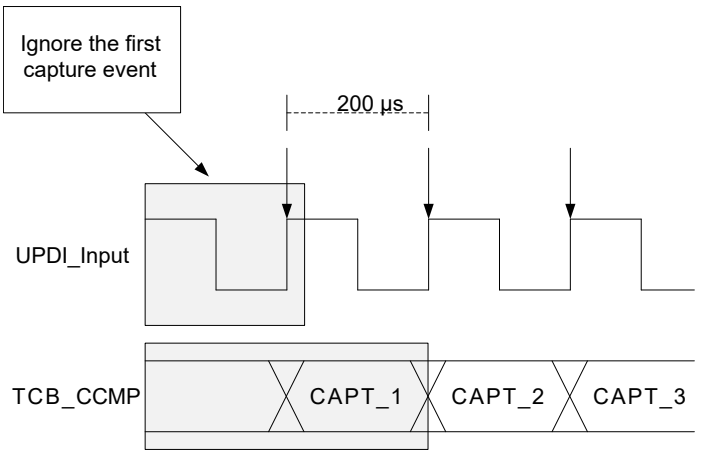
from the ASI_CRC_STATUS register. Refer to the *CRCSCAN - Cyclic Redundancy Check Memory Scan* section in the device data sheet for more information on this feature.

44.3.5 System Clock Measurement with UPDI

It is possible to use the UPDI to get an accurate measurement of the system clock frequency by utilizing the UPDI event connected to TCB with Input Capture capabilities. A recommended setup flow for this feature is given by the following steps:

- Set up TCBn.CTRLB with setting CNTMODE = `0x3`, Input Capture Frequency Measurement mode
- Write CAPTEI = `1` in TCBn.EVCTRL to enable Event Interrupt. Keep EDGE = `0` in TCBn.EVCTRL
- Configure the Event System to route the UPDI SYNCH event (generator) to the TCB (user)
- For the SYNCH character used to generate the UPDI events, it is recommended to use a slow baud rate in the range of 10-50 kbps to get a more accurate measurement of the value captured by the timer between each UPDI event. One particular thing is that if the capture is set up to trigger an interrupt, the first captured value must be ignored. The second captured value based on the input event must be used for the measurement. See Figure 44-17 for an example using 10 kbps UPDI SYNCH character pulses, giving a capture window of 200 µs for the timer.
- It is possible to read out the captured value directly after the SYNCH character by reading the TCBn.CCMP register, or the value can be written to memory by the CPU once the capture is done. For more details, refer to the *TCB - 16-bit Timer/Counter Type B* section.

Figure 44-17. UPDI System Clock Measurement Events



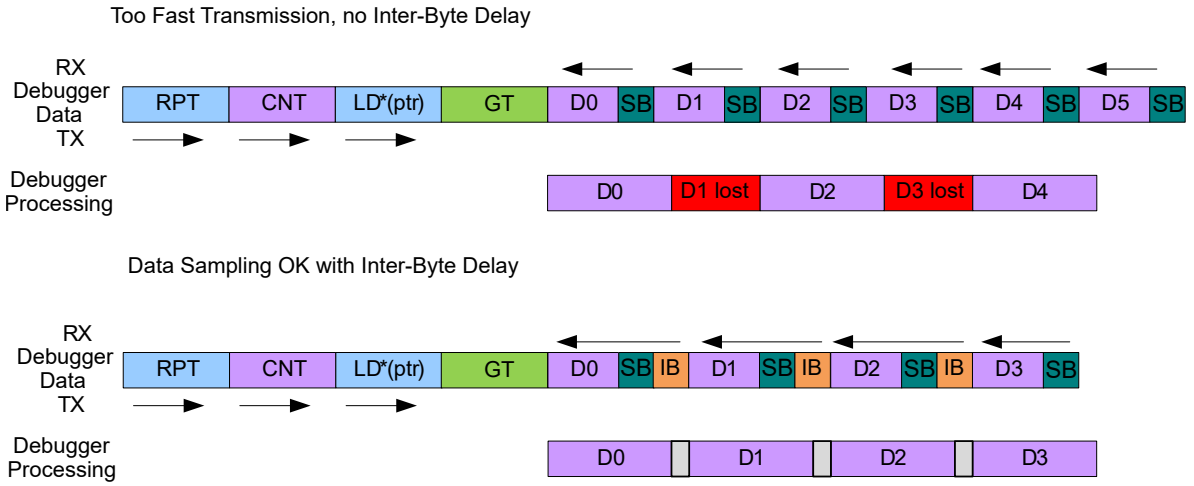
44.3.6 Inter-Byte Delay

When performing a multibyte 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, the data might come out too fast on the receiver side, 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 multibyte 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 44-18. 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.

44.3.7 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 the *KEY - Set Activation Key or Send System Information Block* section. 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 Figure 44-19 for SIB format description and which data are available at different readout sizes.

Figure 44-19. 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

44.3.8 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 KEY - Set Activation Key or Send System Information Block. Table 44-4 describes the available keys and the condition required for starting the operation after the key has been loaded.

Table 44-4. 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
NVMPROG	Activate NVM programming	Lock bits cleared. ASI_SYS_STATUS.PROGSTART set.	Programming done/UPDI Reset
USERROW-Write	Program the user row on the locked device	ASI_SYS_STATUS.UROWSTART set	Write to key Status bit/UPDI Reset

Table 44-5 gives an overview of the available key signatures that must be shifted in to activate the interfaces.

Table 44-5. 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

44.3.8.1 Chip Erase

The next steps must be followed to issue a chip erase:

1. Enter the Chip Erase key by using the `KEY` instruction. See the *Key Activation Signatures* table for the CHIPERASE signature.
2. Enter the NVM Programming key by using the `KEY` instruction. See the *Key Activation Signatures* table for the NVMPROG signature. This will prevent a freshly erased device from failing the CRC (if activated).
3. Read the ASI Key Status (UPDI.ASI_KEY_STATUS) register to verify that both the Chip Erase Key Status (CHER) and the NVM Programming Key Status (NVMPROG) bits are set.
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 Lock Status (LOCKSTATUS) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
7. The chip erase is done when the LOCKSTATUS bit is '`0`'. If the LOCKSTATUS bit is '`1`', return to step 5.
8. Check the Chip Erase Key Failed (ERASEFAIL) bit in the ASI System Status (UPDI.ASI_SYS_STATUS) register to verify if the chip erase was successful.
9. If the ERASEFAIL 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 - Brown-out Detector* section for more details.

44.3.8.2 NVM Programming

If the device is unlocked, it is possible to write directly to the NVM Controller or 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 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 44-5 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 Start NVM Programming (PROGSTART) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
7. NVM programming can start when the PROGSTART bit is '1'. If the PROGSTART 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.

44.3.8.3 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 next sequence must be followed:

1. Enter the USERROW-Write key located in Table 44-5 by using the `KEY` instruction. See Table 44-5 for the USERROW-Write signature.
2. **Optional:** Read the User Row Write Key Status (UROWWR) 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 (UROWSTART) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
6. The user row programming can start when the UROWSTART bit is '1'. If UROWSTART 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 512 bytes, and it is only possible to write user row data to the first 512 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 the 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 (UROWSTART) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.

10. The user row programming is completed when the UROWSTART bit is '0'. If the UROWSTART bit is '1', return to step 9.
11. Write to the User Row Write Key Status (UROWWR) 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 512 bytes of the RAM are allowed.

44.3.9 Events

The UPDI can generate the following events:

Table 44-6. 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 *EVSYS - Event System* section for more details regarding event types and Event System configuration.

44.3.10 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 cannot 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's 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.

44.4 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 ... 0x06	Reserved									
0x07	ASI_KEY_STATUS	7:0			UROWWR	NVMPROG	CHER			
0x08	ASI_RESET_REQ	7:0	RSTREQ[7:0]							
0x09	ASI_CTRLA	7:0							UPDICLKSEL[1:0]	
0x0A	ASI_SYS_CTRLA	7:0							UROWDONE	CLKREQ
0x0B	ASI_SYS_STATUS	7:0	BDEF	ERASEFAIL	SYSRST	INSLEEP	PROGSTART	UROWSTART	BOOTDONE	LOCKSTATUS
0x0C	ASI_CRC_STATUS	7:0						CRC_STATUS[2:0]		

44.5 Register Description

These registers are readable only through the UPDI with special instructions and are not readable through the CPU.

44.5.1 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	1	1				

Bits 7:4 – UPDIREV[3:0] UPDI Revision
This bit field contains the revision of the current UPDI implementation.

44.5.2 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 44-7. 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

44.5.3 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 multibyte 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 and 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, a loss of data may occur.

Bits 2:0 – GTVAL[2:0] Guard Time Value
This bit field selects the guard time value 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

44.5.4 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 contention detection. Writing a '0' to this bit enables 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.

44.5.5 ASI Key Status

Name: ASI_KEY_STATUS
Offset: 0x07
Reset: 0x00
Property: -

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

Bit 5 – UROWWR 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 PROGSTART bit in ASI_SYS_STATUS is set.

Bit 3 – CHER 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 Chip Erase section.

44.5.6 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	-	Reserved

44.5.7 ASI Control A

Name: ASI_CTRLA
Offset: 0x09
Reset: 0x03
Property: -

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

Bits 1:0 – UPDICLKSEL[1:0] UPDI Clock Divider Select
Writing these bits selects the UPDI clock output frequency. The default setting after Reset and enable is 4 MHz. See the *Electrical Characteristics* section for more information on possible UPDI oscillator frequencies.

Value	Description
0x0	32 MHz UPDI clock
0x1	16 MHz UPDI clock
0x2	8 MHz UPDI clock
0x3	4 MHz UPDI clock (default setting)

44.5.8 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
Write this bit when the user row data is 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's 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.

44.5.9 ASI System Status

Name: ASI_SYS_STATUS
Offset: 0x0B
Reset: 0x01
Property: -

Bit	7	6	5	4	3	2	1	0
	BDEF	ERASEFAIL	SYSRST	INSLEEP	PROGSTART	UROWSTART	BOOTDONE	LOCKSTATUS
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit 7 – BDEF Boot Sequence Done or Chip Erase Failed
This bit is set to '1' if the chip erase has failed (ERASEFAIL bit is '1') or the boot sequence is complete (BOOTDONE bit is '1').

Bit 6 – ERASEFAIL 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 Request (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 – PROGSTART 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 ASI Reset Request (ASI_RESET_REQ) register.

Bit 2 – UROWSTART 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 1 – BOOTDONE Boot Sequence Done
This bit is set to '1' when the CPU is done with the boot sequence. The UPDI will not have access to the ACC layer until this bit is set to '1'.
Check also that SYSRST is '0' before proceeding.

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'.

44.5.10 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	Description
0x0	Not enabled
0x1	CRC enabled, busy
0x2	CRC enabled, done with OK signature
0x4	CRC enabled, done with FAILED signature
Other	Reserved

45. Instruction Set Summary

The instruction set summary is 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.

46. Electrical Characteristics

46.1 Disclaimer

Unless otherwise specified, all typical values are measured at T_A = 25°C and V_{DD} = 3.0V. All minimum and maximum values are valid across operating temperature and voltage unless otherwise specified.

The given typical values must be considered for design guidance only, and part variation around the values is expected.

46.2 Absolute Maximum Ratings

Stresses beyond those listed in this section can cause permanent damage to the device. These are stress ratings only. Functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 46-1. Absolute Maximum Ratings

Parameter	Condition	Rating	Units
Ambient temperature under bias		-40 to +125	°C
Maximum junction temperature		+145	°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 $\overline{\text{RESET}}$ pin		-0.3 to +9.0	V
On the MVIO pins		-0.3 to (V_{DDIO2} + 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's power dissipation characterizations. See the *Thermal Characteristics* section to calculate device specifications.
2. Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OL} \times I_{OL})$.

46.3 Standard Operating Conditions

For all other device characteristics to be valid, the device must operate within the ratings listed in this section.

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 follows:

Table 46-2. Standard Operating Conditions

Parameter		Ratings	Units
V_{DD} — Operating Supply Voltage(1)			
Industrial and Extended temperature	V_{DDMIN}	2.7	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 the Supply Voltage parameter in the *Supply Voltage* section.

46.4 Supply Voltage

Table 46-3. Supply Voltage

Symbol	Min.	Typ. †	Max.	Units	Conditions
Supply Voltage(1)					
V_{DD}	2.7	-	5.5	V	
V_{DDIO2}	1.71	-	5.5	V	
V_{DD} Slew Rate	-	-	0.25	V/µs	
RAM Data Retention(2)					
V_{DR}	1.7	-	-	V	Device in Power-Down mode
Power-on Reset Release Voltage(4)					
V_{POR}	-	1.6	1.8	V	BOD disabled(3)
t_{POR}	-	1	-	µs	BOD disabled(3)
Power-on Reset Re-Arm Voltage(4)					
V_{PORR}		1.3	-	V	BOD disabled(3)
t_{PORR}	-	2.7	-	µs	BOD disabled(3)
V_{DD} Rise Rate to Ensure Internal Power-on Reset Signal(4)					
S_{VDD} *	0.05	-	-	V/ms	BOD disabled(3)

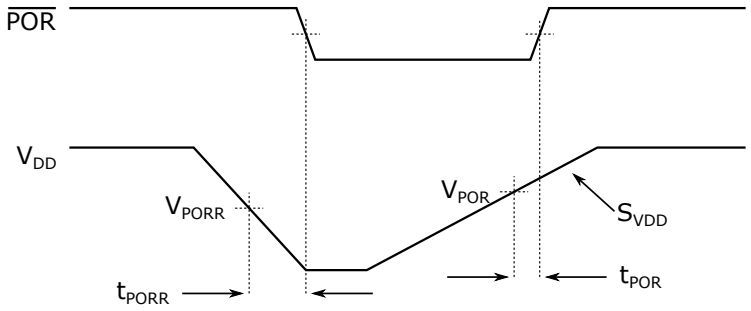
† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

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. This is the limit to which V_{DD} can be lowered in sleep mode without losing RAM data.
3. Refer to the *RSTCTRL* section for BOD trip point information.
4. Refer to the *POR and PORR with Slow-Rising V_{DD}* figure below.

Figure 46-1. POR and PORR with Slow-Rising V_{DD}



Note:

- The device is held in Reset when $\overline{POR}$ is low

46.5 Power Consumption

Table 46-4. Device Power Consumption

Operating Conditions: System power consumption measured with peripherals disabled and I/O ports driven low with inputs disabled $V_{DD} = V_{DDIO2} = 3.0V$							
Symbol	**Description**	**Min.**	**Typ. †**	**Max. 85°C**	**Max. 125°C**	**Units**	**Conditions**
I_{DD}	Active power consumption	-	6	-	-	mA	OSCHF = 20 MHz
		-	1.5	-	-	mA	OSCHF = 4 MHz
		-	28	-	-	µA	OSC32K = 32.768 kHz
		-	5.9	-	-	mA	EXTCLK = 20 MHz
		-	1.4	-	-	mA	EXTCLK = 4 MHz
		-	30	-	-	µA	XOSC32K = 32.768 kHz (High Power)
		-	28	-	-	µA	XOSC32K = 32.768 kHz (Low Power)
		-	6	-	-	mA	XOSCHF = 20 MHz XOSCHFCTRLA.FRQRANGE = 0x2
		-	1.6	-	-	mA	XOSCHF = 4 MHz XOSCHFCTRLA.FRQRANGE = 0x0
I_{DD_IDLE}	Idle power consumption	-	2.3	-	-	mA	OSCHF = 20 MHz
		-	720	-	-	µA	OSCHF = 4 MHz
		-	22.5	-	-	µA	OSC32K = 32.768 kHz
		-	2.2	-	-	mA	EXTCLK = 20 MHz
		-	620	-	-	µA	EXTCLK = 4 MHz
		-	22	-	-	µA	XOSC32K = 32.768 kHz, XOSC32KCTRLA.LPMODE = 0
		-	24	-	-	µA	XOSC32K = 32.768 kHz, XOSC32KCTRLA.LPMODE = 1
		-	2.3	-	-	mA	XOSCHF = 20 MHz XOSCHFCTRLA.FRQRANGE = 0x2
		-	720	-	-	µA	XOSCHF = 4 MHz XOSCHFCTRLA.FRQRANGE = 0x0

.....continued

Operating Conditions: System power consumption measured with peripherals disabled and I/O ports driven low with inputs disabled $V_{DD} = V_{DDIO2} = 3.0V$							
Symbol	**Description**	**Min.**	**Typ. †**	**Max. 85°C**	**Max. 125°C**	**Units**	**Conditions**
I_{DD_BASE}[1]	Minimum power consumption in different sleep modes	-	20	-	-	µA	Power-Down or Standby mode, all peripherals disabled, VMONSEN = 1 VREGCTRL.PMODE = 0 (AUTO)
		-	184	-	-	µA	Power-Down mode, all peripherals disabled, VMONSEN = 1 VREGCTRL.PMODE = 1 (FULL)
		-	1.4	-	-	µA	Power-Down or Standby mode, all peripherals disabled, VMONSEN = 0 VREGCTRL.PMODE = 0 (AUTO)
I_{RST}	Reset power consumption	-	180	-	-	µA	$\overline{RESET}$ pulled to GND

† Unless otherwise specified, data in the "Typ." column is at $T_A = 25°C$ and $V_{DD} = 3.0V$. These parameters are not tested and are for design guidance only.

Note:

1. Single supply mode.

46.6 Peripherals Power Consumption

Use the table below when calculating the additional current consumption for the different I/O peripherals in 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 46-5. Peripheral Power Consumption[1]

Operating Conditions: $-40°C \le T_A \le +125°C$ (typ. +25°C) $V_{DD} = V_{DDIO2} = 3.0V$ OSCHF at 4 MHz used as clock source Device in Standby sleep mode						
Symbol	**Description**	**Min.**	**Typ. †**	**Max. 85°C**	**Units**	**Conditions**
I_{DD_WDT}	Watchdog Timer (WDT)	-	0.5	-	µA	With 32.768 kHz Internal Oscillator
I_{DD_MVIO}	Multi-Voltage I/O (MVIO)	-	0.5	-	µA	Dual supply configuration
I_{DD_VREF}	Voltage Reference (VREF)	-	175	-	µA	ADC0REF enabled, V_{REF} = 2.048V Excluding bandgap
		-	36	-	µA	ACREF enabled, V_{REF} = 2.048V Excluding bandgap
		-	28	-	µA	DACREF enabled, V_{REF} = 2.048V Excluding bandgap
I_{DD_BOD}	Brown-out Detector (BOD)	-	17	-	µA	Brown-out Detect (BOD) continuous Including bandgap
		-	1.6	-	µA	Brown-out Detect (BOD) sampling @128 Hz, including I_{DD_OSC32K} Including bandgap
		-	0.95	-	µA	Brown-out Detect (BOD) sampling @32 Hz, including I_{DD_OSC32K} Including bandgap
I_{DD_BG}	Bandgap	-	13	-	µA	
I_{DD_VMON}		-	18	-	µA	

.....continued

Operating Conditions:

-40°C ≤ T_A ≤ +125°C (typ. +25°C)

V_{DD} = V_{DDIO2} = 3.0V

OSCHF at 4 MHz used as clock source

Device in Standby sleep mode

Symbol	Description	Min.	Typ. †	Max. 85°C	Units	Conditions
I_{DD_TCA}	16-bit Timer/Counter Type A (TCA)	-	6	-	µA	CLK_PER = HFOSC/4 = 1 MHz
I_{DD_TCB}	16-bit Timer/Counter Type B (TCB)	-	3.6	-	µA	
I_{DD_TCD}	12-bit Timer/Counter Type D (TCD)	-	4.5	-	µA	
I_{DD_RTC}	Real-Time Counter (RTC)	-	0.7	-	µA	CLK_RTC = 32.768 kHz Internal Oscillator
		-	3.9	-	µA	RTC running at 1.024 kHz from XOSC32K, XOSC32KCTRLA.LPMODE = 0
		-	2.1	-	µA	RTC running at 1.024 kHz from XOSC32K, XOSC32KCTRLA.LPMODE = 1
I_{DD_OSCHF}	Internal High-Frequency Oscillator (OSCHF)	-	160	-	µA	Internal Oscillator running at 4 MHz
I_{DD_XOSCHF}	High Frequency Crystal Oscillator (XOSCHF)	-	360	-	µA	24 MHz XTAL, C_L = 15 pF
I_{DD_OSC32K}	32.768 kHz Internal Oscillator (OSC32K)	-	0.3	-	µA	
$I_{DD\_XOSC32K}$	32.768 kHz Crystal Oscillator (XOSC32K)	-	2.5	-	µA	XOSC32KCTRLA.LPMODE = 0, C_L = 9 pF
		-	0.5	-	µA	XOSC32KCTRLA.LPMODE = 1, C_L = 9 pF
I_{DD_ADC}	Analog-to-Digital Converter (ADC)	-	-	-	µA	ADC - Non-converting
		-	320	-	µA	ADC @80 ksps(2)
I_{DD_AC}	Analog Comparator (AC)	-	70	-	µA	CTRLA.POWER = 0x0
		-	17	-	µA	CTRLA.POWER = 0x1
		-	12	-	µA	CTRLA.POWER = 0x2
I_{DD_DAC}	Digital-to-Analog Converter (DAC)	-	120	-	µA	DAC + DACOUT, V_{DACREF} = V_{DD}
		-	8	-	µA	DAC, V_{DACREF} = V_{DD}
I_{DD_USART}	Universal Synchronous and Asynchronous Receiver and Transmitter (USART)	-	8.2	-	µA	USART Enabled @9600 Baud
I_{DD_SPI}	Serial Peripheral Interface (SPI)	-	4	-	µA	SPI Host @100 kHz
I_{DD_TWI}	Two-Wire Interface (TWI)	-	8	-	µA	TWI Host @100 kHz
		-	6	-	µA	TWI Client @100 kHz
$I_{DD\_NVM\_ERASE}$	Flash Programming Erase	-	5.1	-	mA	
$I_{DD\_NVM\_WRITE}$	Flash Programming Write	-	6	-	mA	
I_{DD_ZCD}	Zero Cross Detector (ZCD)	-	10	-	µA	Excluding sink/source currents

.....continued						
Operating Conditions: -40°C ≤ T_A ≤ +125°C (typ. +25°C) V_{DD} = V_{DDIO2} = 3.0V OSCHF at 4 MHz used as clock source Device in Standby sleep mode						
Symbol	**Description**	**Min.**	**Typ. †**	**Max. 85°C**	**Units**	**Conditions**

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Notes:

1. Current consumption of the module only. To calculate the total internal power consumption of the microcontroller, add the power consumption values of all the peripherals and the clock sources used to the base power consumption given in the *Power Consumption* section.
2. Average power consumption with ADC active in Free Running mode.

46.7 I/O Pins

Table 46-6. I/O Pin Specifications(1)

Symbol	Description	Min.	Typ. †	Max. 85°C	Units	Conditions
Input Low Voltage						
V_{IL}	I/O PORT:					
	• With Schmitt Trigger buffer	-	-	0.2×V_{DD}	V	PINnCTRL.INLVL = 0x00
	• With LVBUF (TTL compatible)	-	-	0.8	V	PINnCTRL.INLVL = 0x01
	$\overline{\text{RESET}}$ Pin	-	-	0.2×V_{DD}	V	
Input High Voltage						
V_{IH}	I/O PORT:					
	• With Schmitt Trigger buffer	0.8×V_{DD}	-	-	V	PINnCTRL.INLVL = 0x00
	• With LVBUF (TTL compatible)	2	-	-	V	PINnCTRL.INLVL = 0x01
	$\overline{\text{RESET}}$ Pin	0.8×V_{DD}	-	-	V	
Input Leakage Currents(2)						
I_{IL}	I/O PORTS(3)	-	±5	±125	nA	GND ≤ V_{PIN} ≤ V_{DD}, pin at high-impedance, T_A= 85°C
		-	±5	±1000	nA	GND ≤ V_{PIN} ≤ V_{DD}, pin at high-impedance, T_A= 125°C
	$\overline{\text{RESET}}$ Pin(4) *	-	±50	±200	nA	GND ≤ V_{PIN} ≤ V_{DD}, pin at high-impedance, T_A= 85°C
Pull-up Current						
I_{PUR}		90	150	200	µA	V_{DD} = 3.0V, V_{PIN} = GND
Output Low Voltage						
V_{OL}		-	-	0.6	V	I_{OL} = 10 mA, V_{DD} = 3.0V
Output High Voltage						
V_{OH}		V_{DD} - 0.7	-	-	V	I_{OH} = 6 mA, V_{DD} = 3.0V
I/O Slew Rate*						

.....continued						
Symbol	Description	Min.	Typ. †	Max. 85°C	Units	Conditions
	Rising slew rate	-	22	-	ns	PORTCTRL.SRL = 0x00
		-	45	-	ns	PORTCTRL.SRL = 0x01
	Falling slew rate	-	16	-	ns	PORTCTRL.SRL = 0x00
		-	30	-	ns	PORTCTRL.SRL = 0x01
Pin Capacitance*						
C_{IO}	VREF pin	-	7	-	pF	
	XTAL pins	-	4	-	pF	
	Other pins	-	4	-	pF	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

Notes:

1. These figures are valid for all I/O ports regardless of whether they are connected to the V_{DD} or V_{DDIO2} power domain.
2. The negative current is defined as the current sourced by the pin.
3. The leakage current numbers for I/O PORTS are also valid when the pin is used as 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. A higher leakage current may be measured at different input voltages.
5. The input voltage threshold is relative to V_{DDIO2} on MVIO pins (PORTC) and V_{DD} on other pins.

46.8 Memory Programming Specifications

Table 46-7. Memory Programming Specifications

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	
V_{D_RW}	V_{DD} for Read or Erase/Write operation	V_{DDMIN}	-	V_{DDMAX}	V	
N_{D_REF}*	Total Erase/Write cycles before refresh(2)	1M	4M	-	Erase/Write cycles	-40°C ≤ T_A ≤ +85°C
t_{D_CE}	Byte/Multibyte/Full EEPROM Erase time	-	10	11.7	ms	
t_{D_WRE}	Byte Write time	-	70	75	µs	
t_{D_BEW}	Byte Erase and Write time	-	10.07	-	ms	
Program Flash Memory Specifications						
E_P*	Flash memory cell endurance	1k	-	-	Erase/Write cycles	
t_{P_RET}	Characteristic retention	-	40	-	Year	
V_{P_RD}	V_{DD} for Read operation	V_{DDMIN}	-	V_{DDMAX}	V	
V_{P_REW}	V_{DD} for Erase/Write operation	V_{DDMIN} (1)	-	V_{DDMAX}	V	
t_{P_CE}	Chip Erase time	-	11	-	ms	
t_{P_PE}	Page Erase time	-	10	-	ms	
t_{P_WRD}	Byte/Word Write time	-	70	-	µs	

.....continued						
Symbol	**Description**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

Notes:

1. During Chip Erase, the Brown-out Detector (BOD) configured with BODLEVEL0 is forced ON. The erase attempt will fail if the supply voltage V_{DD} is below V_{BOD} for BODLEVEL0.
2. The number of times a separate location may be erased/written before a full refresh (erase/write) of the EEPROM array is required.

46.9 Thermal Characteristics

Table 46-8. Thermal Characteristics

Symbol	Description	Typ.	Unit	Conditions
θ_{JA}	Thermal resistance, junction to ambient	61	°C/W	20-pin SSOP package ()
		52	°C/W	28-pin VQFN package ()
		48	°C/W	28-pin PDIP package (SP)
		52	°C/W	28-pin SSOP package (SS)
		33	°C/W	32-pin VQFN package (RXB)
		52	°C/W	32-pin TQFP package (PT)
T_{JMAX}	Maximum junction temperature			Refer to the *Absolute Maximum Ratings* section

Notes:

- Thermal resistance (θ_{JA}) highly depends on environmental conditions such as airflow, pressure, and PCB conductivity. The given values are based on a four-layer standard PCB with two power planes (2S2P) and without any airflow (refer to EIA/JESD 51-7).
- Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OL} \times I_{OL})$.
- Internal Power Dissipation is calculated as $P_{INTERNAL} = I_{DD} \times V_{DD}$, where I_{DD} is the current to run the chip alone without driving any load on the output pins.
- Derated Power is calculated as $P_{DER} = P_{DMAX}\ (T_J - T_A)/\theta_{JA}$, where T_A = Ambient Temperature and T_J = Junction Temperature.

46.10 CLKCTRL

46.10.1 System Clock

Table 46-9. System Clock Timing Characteristics

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{CLK_MAIN}	Main clock frequency(1,2)			20	MHz	
f_{CY}	Instruction clock frequency		f_{CLK_MAIN}		MHz	
T_{CY}	Instruction period(3)	50	$1/f_{CY}$		ns	

.....continued

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Notes:

1. The main clock frequency (CLK_MAIN) is configured by the Clock Select (CLKSEL) bit field, as described in the *CLKCTRL - Clock Controller* section.
2. The main clock frequency (CLK_MAIN) must meet the voltage requirements defined in the *Standard Operating Conditions* section.
3. Instruction Cycle Period (T_{CY}) equals the input oscillator time base period. Exceeding these limits may result in incorrect code execution and/or higher-than-expected current consumption. All devices are tested to operate at 'min' values with an external clock applied to the EXTCLK pin. When using an external clock input, the 'max' cycle time limit is 'DC' (no clock) for all devices.

46.10.2 Internal Oscillators

Table 46-10. Internal Oscillator Specifications(1)

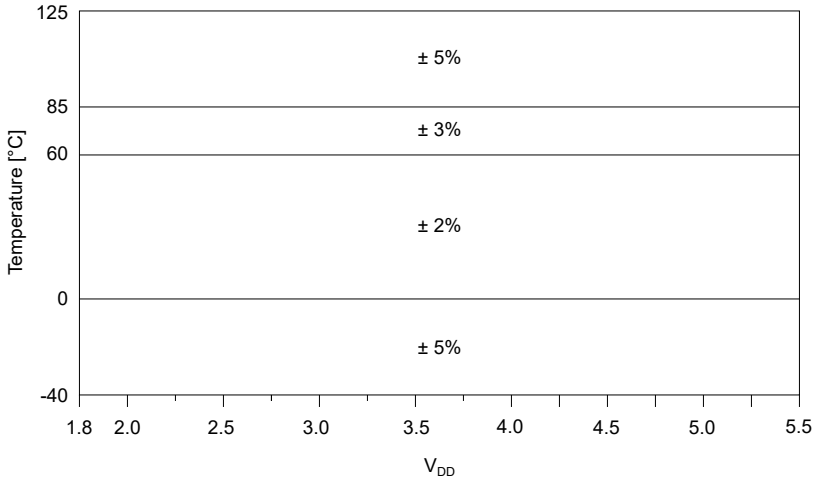
Internal Oscillators						
Operating Conditions: V_{DD} = V_{DDMIN} to V_{DDMAX} Temperature = -40°C to +125°C						
Symbol	**Description**	**Min.**	**Typ. †**	**Max.**	**Unit**	**Conditions**
f_{OSCHF}	OSCHF frequency(2)	-	1	-	MHz	FRQSEL = 1M
		-	2	-		FRQSEL = 2M
		-	3	-		FRQSEL = 3M
	Precision calibrated OSCHF frequency	-	4	-	MHz	FRQSEL = 4M
		-	8	-		FRQSEL = 8M
		-	12	-		FRQSEL = 12M
		-	16	-		FRQSEL = 16M
		-	20	-		FRQSEL = 20M
	Precision calibrated OSCHF accuracy at calibration point	-1.8%	-	+1.8%		FRQSEL = 4M to 20M, T_A = +25°C, V_{DD} = 3.0V
$\%_{CAL}$	OSCHF tune step size	-	0.4	-	%	
t_{OSCHF_ST} (3)	OSCHF wake-up from sleep startup time	-	24	-	µs	Device in Idle or Standby sleep mode, VREGCTRL.PMODE = FULL
		-	220	-	µs	Device in Power-Down sleep mode, VREGCTRL.PMODE = AUTO
f_{OSC32K}	Internal OSC32K frequency	-	32.768	-	kHz	
t_{OSC32K_ST} (3)	OSC32K wake-up from sleep start-up time	-	950	-	µs	Device in Power-Down sleep mode, VREGCTRL.PMODE = AUTO

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Notes:

1. To ensure these oscillator frequency tolerances, V_{DD} and GND must be capacitively decoupled as close to the device as possible. Recommend using 100 nF and 1 µF values in parallel.
2. These parameters are not calibrated.
3. Wake-up times are measured from the wake-up event to code execution.

Figure 46-2. Precision Calibrated OSCHF (f_{OSCHF} ≥ 4 MHz) Frequency Accuracy Over Device V_{DD} and Temperature(1)



Note:

1. The figure is applicable only for f_{OSCHF} ≥ 4 MHz.

46.10.3 XOSC32K

Table 46-11. 32.768 kHz Crystal Oscillator (XOSC32K) Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{XOSC32}	Frequency	-	32.768	-	kHz	
$C_{XTAL1/XTAL2}$*	Parasitic pin capacitance	-	5	-	pF	
C_L*	Crystal load capacitance	-	-	18	pF	XOSC32KCTRLA.LPMODE = 0
		-	-	8	pF	XOSC32KCTRLA.LPMODE = 1
ESR*	Crystal Equivalent Series Resistance	-	100	-	kΩ	XOSC32KCTRLA.LPMODE = 0
		-	50	-	kΩ	XOSC32KCTRLA.LPMODE = 1
t_{XOSC32_ST} *(1)	XOSC32 start-up time	-	<300	-	ms	XOSC32KCTRLA.LPMODE = 0
		-	<1000	-	ms	XOSC32KCTRLA.LPMODE = 1

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

Note:

1. t_{XOSC32_ST} depends on crystal type and external components.

46.10.4 XOSCHF

Table 46-12. High Frequency Crystal Oscillator (XOSCHF) Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{XOSCHF} *	Frequency	4		8	MHz	XOSCHFCTRLA.FRQRANGE = 0x0
		4		16	MHz	XOSCHFCTRLA.FRQRANGE = 0x1
		4		32	MHz	XOSCHFCTRLA.FRQRANGE = 0x2
$C_{XTAL1/XTAL2}$ *	Parasitic pin capacitance		5		pF	
C_L *	Crystal load capacitance		12		pF	

.....continued						
Symbol	**Description**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
ESR *	Equivalent Series Resistance		200		kΩ	XTALHF = 8 MHz, XOSCHFCTRLA.FRQRANGE = 0x0
			60		kΩ	XTALHF = 16 MHz, XOSCHFCTRLA.FRQRANGE = 0x1
			40		kΩ	XTALHF = 24 MHz, XOSCHFCTRLA.FRQRANGE = 0x2
t_{XOSCHF} *	XOSCHF start-up time		<5		ms	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

* These parameters are characterized but not tested in production.

46.10.5 External Clock

Figure 46-3. External Clock Waveform

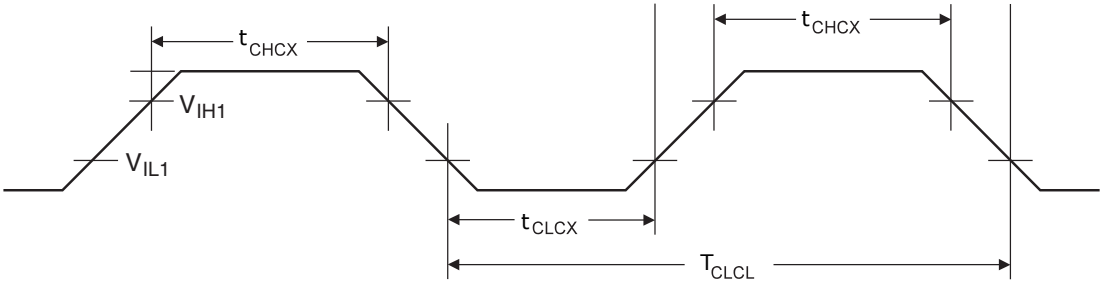


Table 46-13. External Clock Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{CLCL}	Clock frequency			24	MHz	
T_{CLCL}	Clock period	41.7			ns	
t_{CHCX}	High time		40		%	
t_{CLCX}	Low time		40		%	
ΔT_{CLCL}	Change in period from cycle to cycle time		20		%	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

46.10.6 PLL

Table 46-14. PLL Timing Characteristics

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	
t_{PLLST}	PLL lock time		10		µs	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

* These parameters are characterized but not tested in production.

46.11 Voltage Regulator Monitor

Table 46-15. Voltage Regulator Monitor Characteristics

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
$T_{DIAGNOSTIC}$	Duration of diagnostic timer	-	90	-	µs	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

46.12 RSTCTRL

Table 46-16. Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-out Detector Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
t_{RST} *	$\overline{RESET}$ pin pulse-width low to ensure a Reset	2.5			µs	
R_{RST_UP} *	$\overline{RESET}$ pin pull-up resistor		35		kΩ	
T_{WDT} *	Watchdog Timer time-out period		15.6		ms	1:512 Prescaler
T_{SUT} *	Power-up timer period		64		ms	SUT=7
V_{BOD}	Brown-out Detect Voltage(1)		1.9		V	BODLEVEL0
		2.3	2.45	2.65	V	BODLEVEL1
		2.55	2.7	2.9	V	BODLEVEL2
		2.7	2.85	3.05	V	BODLEVEL3
V_{BOD_HYS}	Brown-out Detect hysteresis		70		mV	BODLEVEL1
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	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

Note:

1. To ensure these voltage tolerances, V_{DD} and V_{DDIO2} must be capacitively decoupled to GND as close to the device as possible. See the *Connection for Power Supply* section of the *Hardware Guidelines* section for details.

Table 46-17. Voltage Level Monitor Threshold Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
V_{DET} *	Voltage detection threshold	1	5	10	% of BOD threshold	VLMLVL = 0x01
		9	15	22		VLMLVL = 0x02
		19	25	32		VLMLVL = 0x03

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

46.13 VREF

Table 46-18. V_{REF} Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
VDD_{VREF_1V024}	V_{DD} for 1.024V Internal VREF	2.7		5.5	V	
VDD_{VREF_2V048}	V_{DD} for 2.048V Internal VREF	2.7		5.5	V	
VDD_{VREF_4V096}	V_{DD} for 4.096V Internal VREF	4.55		5.5	V	
VDD_{VREF_2V5}	V_{DD} for 2.5V Internal VREF	2.95		5.5	V	
VDD_{VREF_EXT}	V_{DD} for External VREF	2.7		5.5	V	
V_{VREF_EXT} *	External V_{REF} Input Pin Voltage	1.024		V_{DD}	V	
V_{VREF_1V024} (1)	Internal V_{REF} Accuracy 1.024V	-4		4	%	-40°C ≤ T_A ≤ +85°C
V_{VREF_2V048} (1)	Internal V_{REF} Accuracy 2.048V	-4		4	%	-40°C ≤ T_A ≤ +85°C
V_{VREF_4V096} (1)	Internal V_{REF} Accuracy 4.096V	-4		4	%	-40°C ≤ T_A ≤ +85°C
V_{VREF_2V500} (1)	Internal V_{REF} Accuracy 2.5V	-4		4	%	-40°C ≤ T_A ≤ +85°C
t_{VREF_ST} *	V_{REF} start-up time		10		µs	CLKCTRL.MCLKCTRLA = 0x00 or 0x03
			200		µs	CLKCTRL.MCLKCTRLA = 0x01 or 0x02
t_{INTREF} *	V_{REF} settling time (2)		2		µs	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

* These parameters are characterized but not tested in production.

Notes:

1. The symbol V_{VREF_xVxxx} refers to the respective values of the REFSEL bit fields in the ADC0.CTRLC, ADC1.CTRLC, VREF.DAC0REF and VREF.ACREF registers.
2. Settling time will be needed after changing the REFSEL setting.

46.14 TCD

Table 46-19. TCD Timing Characteristics

Operating Conditions: 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.**	**Units**	**Conditions**
$f_{CLK\_TCD\_SYNC}$ *	CLK_TCD_SYNC maximum frequency			48	MHz	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

* These parameters are characterized but not tested in production.

46.15 USART

Figure 46-4. USART in SPI Mode - Timing Requirements in Host Mode

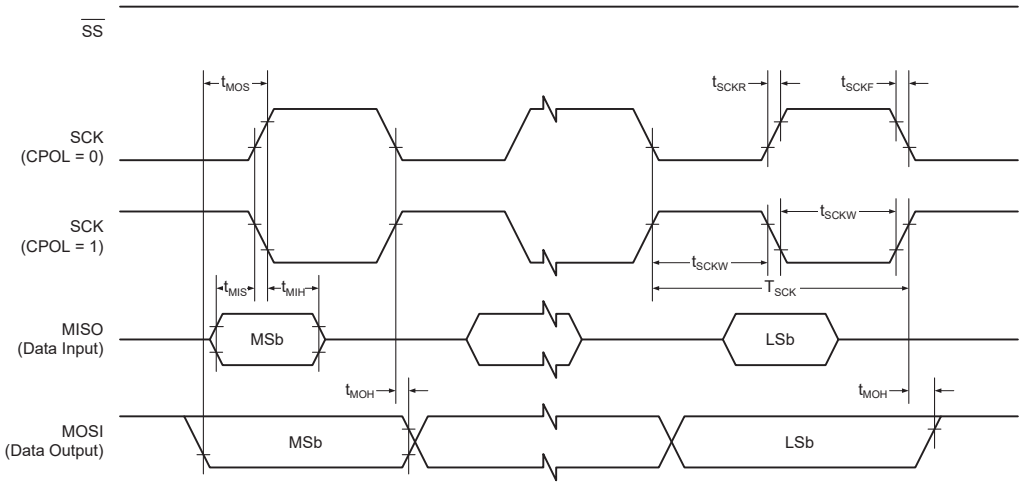


Table 46-20. USART in SPI Host Mode - Timing Characteristics

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{SCK}(1)	SCK clock frequency	—	—	$f_{CLK\_PER}/2$	MHz	
T_{SCK}(1)	SCK period	$2\times T_{CLK\_PER}$	—	—	ns	
t_{SCKW}	SCK high/low width	—	$0.5\times T_{SCK}$	—	ns	
t_{MOS}	MOSI valid before SCK	—	$0.5\times T_{SCK}$		ns	
t_{MOH}	MOSI hold after SCK	—	$0.5\times T_{SCK}$	—	ns	
t_{MIS}	MISO setup to SCK	—	T_{CLK_PER}	—	ns	
t_{MIH}	MISO hold after SCK	—	0	—	ns	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Note:

1. These parameters are characterized but not tested in production.

46.16 SPI

Figure 46-5. SPI - Timing Requirements in Host Mode

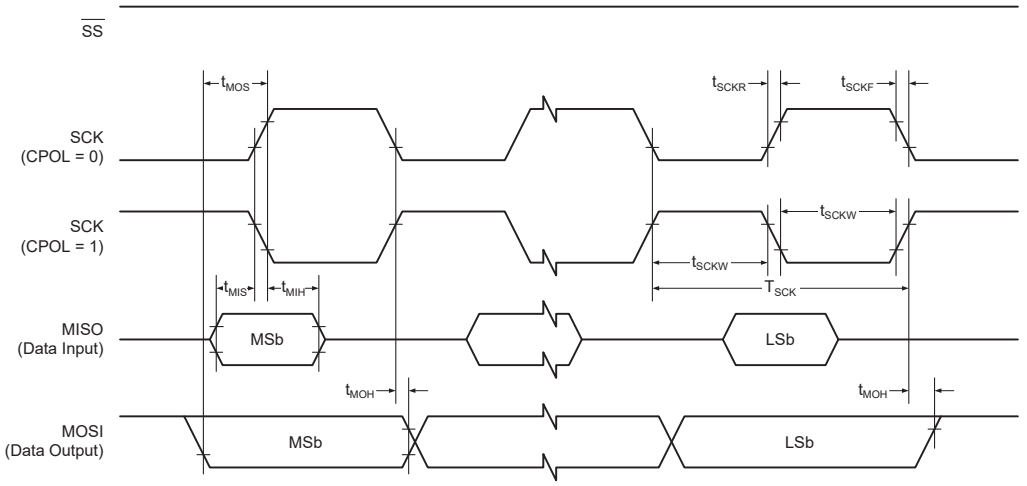


Table 46-21. SPI - Timing Characteristics in Host Mode

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{SCK}(1)	SCK clock frequency	—	—	$f_{CLK\_PER}/2$	MHz	
T_{SCK}(1)	SCK period	$2\times T_{CLK\_PER}$	—	—	ns	
t_{SCKW}	SCK high/low width	—	$0.5\times T_{SCK}$	—	ns	
t_{MOS}	MOSI valid before SCK	—	$0.5\times T_{SCK}$	—	ns	
t_{MOH}	MOSI hold after SCK	—	$0.5\times T_{SCK}$	—	ns	
t_{MIS}	MISO setup to SCK	—	T_{CLK_PER}	—	ns	
t_{MIH}	MISO hold after SCK	—	0	—	ns	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Note:

1. These parameters are characterized but not tested in production.

Figure 46-6. SPI - Timing Requirements in Client Mode

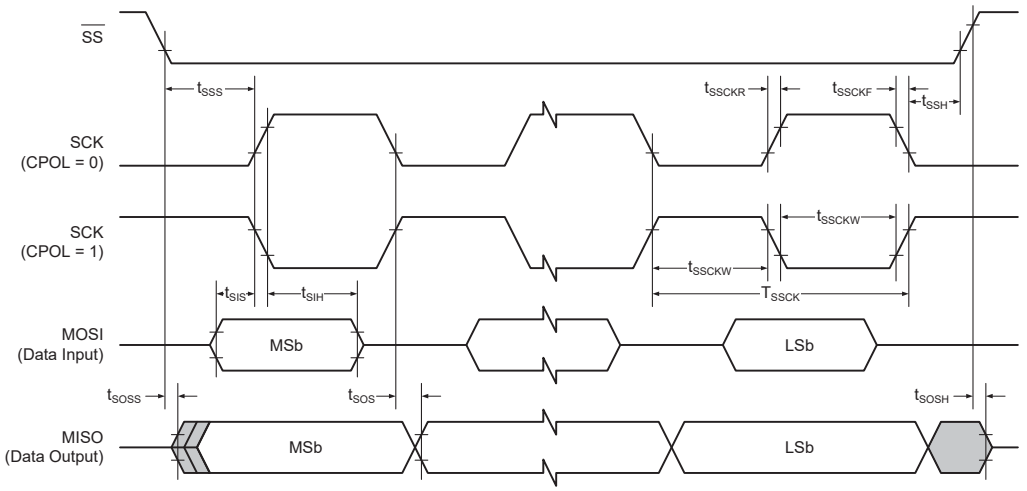


Table 46-22. SPI - Timing Characteristics in Client Mode

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{SSCK}(1)	Client SCK clock frequency	—	—	$f_{CLK\_PER}/4$	MHz	
T_{SSCK}(1)	Client SCK period	$4\times T_{CLK\_PER}$	—	—	ns	
t_{SSCKW}(1)	SCK high/low width	$2\times T_{CLK\_PER}$	—	—	ns	
t_{SIS}(1)	MOSI setup to SCK	—	T_{CLK_PER}	—	ns	
t_{SIH}(1)	MOSI hold after SCK	—	T_{CLK_PER}	—	ns	
t_{SSS}(1)	$\overline{SS}$ low before SCK	—	T_{CLK_PER}	—	ns	
t_{SSH}(1)	$\overline{SS}$ high after SCK	—	T_{CLK_PER}	—	ns	
t_{SOS}	MISO Valid after SCK	—	t_{SR}(2)	—	ns	$f_{SSCK}<f_{CLK\_PER}/6$
		—	$T_{CLK\_PER}+t_{SR}$(2)	—	ns	$f_{SSCK}\geq f_{CLK\_PER}/6$
t_{SOSS}	MISO setup after $\overline{SS}$ low	—	t_{SR}(2)	—	ns	
t_{SOSH}	MISO hold after $\overline{SS}$ high	—	t_{SR}(2)	—	ns	
t_{SDLY}	Interbyte delay	T_{CLK_PER}	—	—	ns	$f_{SSCK}<f_{CLK\_PER}/5$
		0	—	—	ns	$f_{SSCK}\geq f_{CLK\_PER}/5$

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Notes:

1. These parameters are characterized but not tested in production.
2. t_{SR} is the I/O pin rise/fall time.

46.17 TWI

Figure 46-7. TWI - Timing Requirements

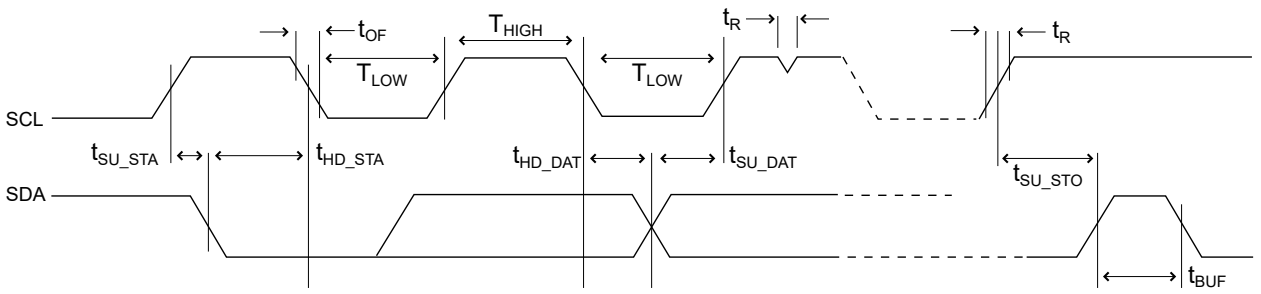


Table 46-23. TWI - Timing Characteristics

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
f_{SCL}	SCL clock frequency(1)	—	—	1000	kHz	Fm+
		—	—	400	kHz	Fm
		—	—	100	kHz	Sm
V_{IL}	LOW-level input voltage	-0.5	—	$0.3\times V_{DD}$	V	INPUTLVL=0x0 (I^2C)
		-0.5	—	0.8		INPUTLVL=0x1 (SMBUS)
V_{IH}	HIGH-level input voltage(2)	$0.7\times V_{DD}$	—	$V_{DD}+0.5V$	V	INPUTLVL=0x0 (I^2C)
		1.35	—	$V_{DD}+0.5V$		INPUTLVL=0x1 (SMBUS)
V_{HYS}*	Hysteresis of Schmitt Trigger inputs	$0.05\times V_{DD}$	—	—	V	INPUTLVL=0x0 (I^2C)
		—	0.2	—		INPUTLVL=0x1 (SMBUS)

.....continued						
Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
V_{OL}	Output low voltage(3)	—	—	0.6	V	I_{LOAD} = 6 mA
t_{SP} *	Spikes suppressed by the input filter	0	—	t_{SPM}	ns	
t_{SPM} *	Input filter delay	50	—	200	ns	
t_{HD_DAT} *	Data hold time	—	0	—	ns	SDAHOLD[1:0] = 0x0 (for most applications)
		—	50	—		SDAHOLD[1:0] = 0x1
		—	300	—		SDAHOLD[1:0] = 0x2
		300	500	900		SDAHOLD[1:0] = 0x3 (for SMBus 2.0)

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

* These parameters are not tested and are for design guidance only.

Notes:

1. The system clock frequency must be at least 10 times faster than the TWI bus clock ($f_{CLK\_PER} \geq 10 \times f_{SCL}$), but additional limitations may apply depending on the baud rate. Refer to the *TWI* section for more detailed information.
2. If a TWI pin is above V_{DD}+0.5V, current will flow from the TWI pin to V_{DD}.
3. I^2C Fm+ full 20 mA drive capability is not supported.

46.18 AC

Table 46-24. Analog Comparator Specifications

Operating Conditions: V_{DD} = 3.0V, T_A = 25°C						
Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
V_{IN} *	Input voltage range	-0.2		V_{DD}	V	
I_L	Input leakage current		5		nA	
V_{OFF}	Input offset voltage	-15	±5	15	mV	0.1V < V_{IN} < (V_{DD} - 0.1V)
CMRR	Common mode input rejection ratio		70		dB	0.1V < V_{IN} < (V_{DD} - 0.1V)
V_{HYST}	Hysteresis		20		mV	CTRLA.HYSMODE = 0x1, CTRLA.POWER = 0x0
			28			CTRLA.HYSMODE = 0x2, CTRLA.POWER = 0x0
			43			CTRLA.HYSMODE = 0x3, CTRLA.POWER = 0x0
t_{RESP} (1)*	Response time, rising edge		85		ns	CTRLA.POWER = 0x0, V_{CM} = V_{DD}/2
	Response time, falling edge		85		ns	
	Response time, rising edge		250		ns	CTRLA.POWER = 0x1, V_{CM} = V_{DD}/2
	Response time, falling edge		220		ns	
	Response time, rising edge		460		ns	CTRLA.POWER = 0x2, V_{CM} = V_{DD}/2
	Response time, falling edge		430		ns	

* These parameters are characterized but not tested.

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V.

Note:

1. Response time is measured with one comparator input at V_{DD}/2 while the other input transitions from GND to V_{DD}.

46.19 ADC

Table 46-25. ADC Accuracy Specifications

Operating Conditions: V_{DD} = 3.0V, T_A = 25°C, V_{ADCREF} = 3.0V, f_{CLK_ADC} = 500 kHz						
Symbol	**Description**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
N_R	Resolution	-	-	10	bit	
E_{INL}	Integral nonlinearity error	-	0.1	-	LSb	
E_{DNL}	Differential nonlinearity error(1)	-	0.1	-	LSb	
E_{OFF}	Offset error	-	0.5	-	LSb	
E_{GAIN}	Gain error	-	0.2	-	LSb	
E_{ABS}	Absolute error	-	0.8	-	LSb	
V_{ADCREF} *	ADC reference voltage	1	-	V_{DD}	V	
V_{AIN}	Full-scale range	GND	-	V_{ADCREF}	V	
Z_{AIN}	Recommended impedance of ADC input voltage source	-	<10	-	kΩ	
R_{VREFA}	ADC voltage reference ladder resistance(2)	-	50	-	kΩ	
$E_{VDD/10}$	V_{DD}/10 divider (VDDDIV10/VDDIO2DIV10) accuracy	-	±10	-	%	Measured with ADC using on-chip internal reference

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

Notes:

1. The ADC conversion result never decreases with an increase in the input and has no missing codes.
2. This is the resistance seen by the VREFA pin when the external reference is selected.

Table 46-26. ADC Conversion Timing Specifications

Symbol		Min.	Typ. †	Max.	Units	Conditions
T_{CLK_ADC} *	ADC clock period	0.5		8	µs	
t_{CNV}	Conversion time		$11.5 \times T_{CLK\_ADC} + 2 \times T_{CLK\_PER}$			
f_{ADC} *	Sample rate	8		170	ksps	
t_{SENSE} *	Delay for changing MUXPOS to TEMP		40		µs	
t_{ADC_INIT} *	Initialization time		6		µs	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

46.20 DAC

Table 46-27. DAC Electrical Specifications

Operating Conditions: V_{DD} = 3.0V, T_A = 25°C, V_{REF} = 3.0V						
Symbol	**Description**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
V_{LSB}	Resolution	-	-	10	Bit	
V_{OUT}	Output voltage range	0.1		V_{DD}-0.1	V	I_{OH} max. (current sourcing) = 1 mA I_{OL} max. (current sinking) = 0.001 mA(2)
V_{ACC}	Absolute accuracy	-10	1	10	LSb	
E_{INL}	Integral nonlinearity error	-2.3	1	2.3	LSb	0x030 ≤ DAC.DATA < 0x3D0
E_{DNL}	Differential nonlinearity error	-0.7	0.2	0.7	LSb	0x030 ≤ DAC.DATA < 0x3D0
E_{OFF}	Offset error	-5	2.8	5	LSb	
E_{GAIN}	Gain error	-3.3	-1.1	1.3	LSb	
t_{ST}	Settling time(1)	-	7	-	µs	V_{DACREF} = V_{DD} = 3.0V, 50 pF Load
		-	10	-	µs	V_{DACREF} = V_{DD} = 5.0V, 50 pF Load

† Unless otherwise specified, data found in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

Notes:

1. Settling time is measured while DAC.DATA[9:0] transitions from '0x000' to '0x3FF'.
2. The DAC output has a limited current sinking capability. It is designed to drive resistive loads connected to ground. If the DAC output needs to sink current, it is recommended to increase the sinking capability by placing a suitable resistor between the DAC output pin and ground.

46.21 ZCD

Table 46-28. Zero-Cross Detector Specifications

Operating Conditions: V_{DD} = 3.0V, T_A = 25°C						
Symbol	**Description**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
V_{PINZC}	Voltage on the ZCD pin		0.95		V	
I_{ZCD_MAX}	Maximum source or sink current			600	µA	
t_{RESPH}	Response time, rising edge		500		ns	
t_{RESPL}	Response time, falling edge		350		ns	

† Unless otherwise specified, data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V. These parameters are not tested and are for design guidance only.

46.22 UPDI

Figure 46-8. UPDI Enable Sequence

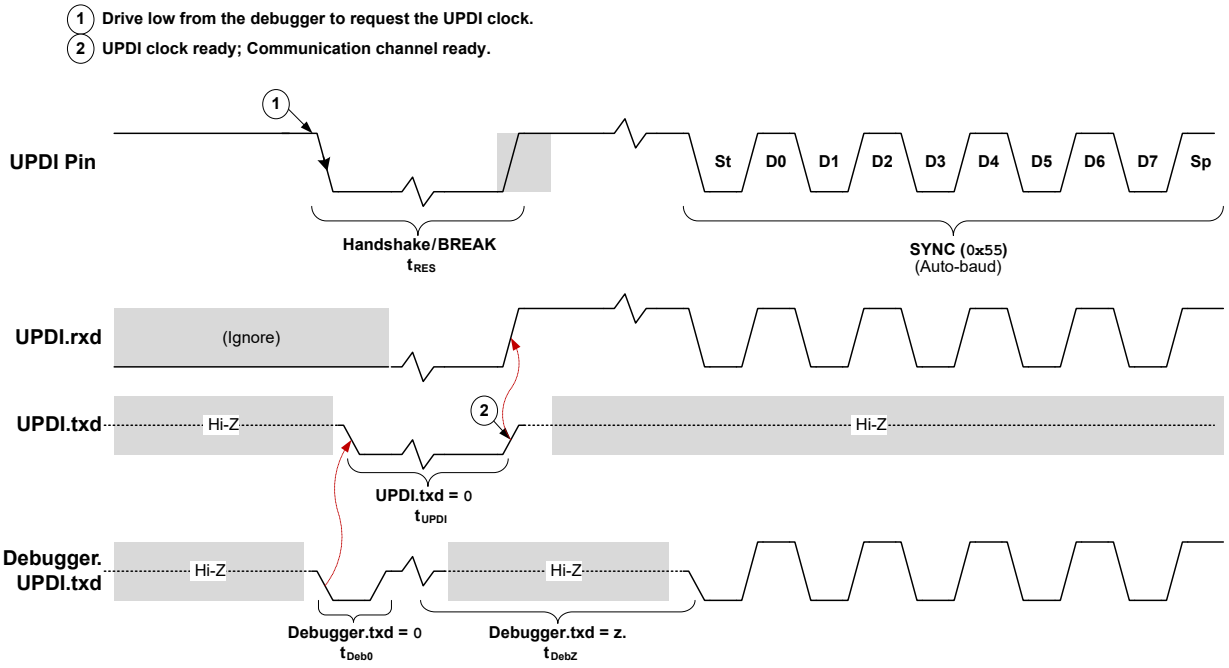


Table 46-29. UPDI Timing Specifications

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
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	
f_{UPDI} *	UPDI baud rate			1.8	Mbps	0°C ≤ T_A ≤ +50°C
				0.9	Mbps	T_A < 0°C or T_A > +50°C
* These parameters are characterized but not tested in production.						

47. Characteristics Graphs

Characteristics Graphs are not available at this time.

48. Ordering Information

- Available ordering options can be found by:
  - Clicking on one of the following product page links:
    - AVR32SD20
    - AVR32SD28
    - AVR32SD32
  - Searching by product name at microchipdirect.com
  - Contacting the local sales representative

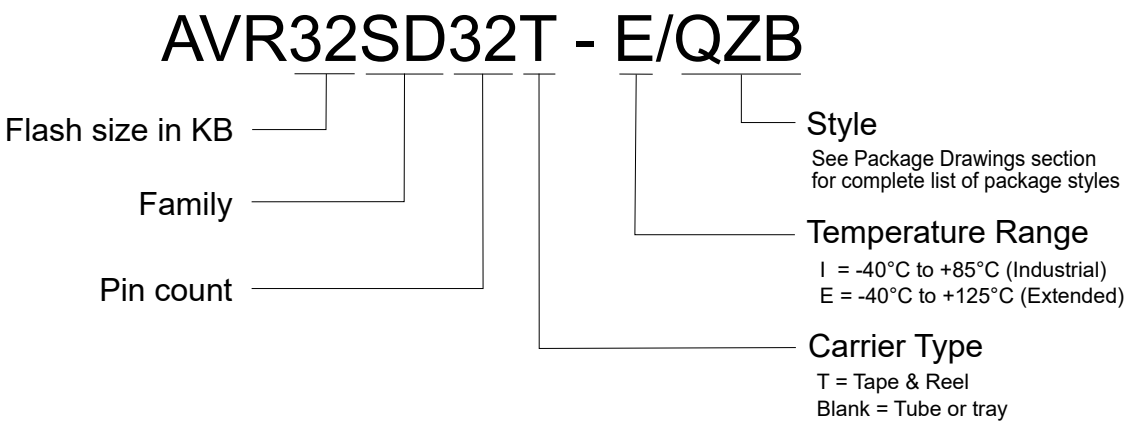
Note: Automotive-grade ordering codes (VAO suffix) are set up on request and not listed in *Available Product Numbers*. Contact your local Microchip sales representative to request VAO ordering codes not on the product page.

Table 48-1. Available Ordering Codes

Ordering Code	Flash / SRAM	Pin Count	Package Type	Supply Voltage	Package Code	Temperature Range	Carrier Type
AVR32SD20-I/SS	32 KB/4 KB	20	SSOP	2.7-5.5V	G3X	-40°C to +85°C	Tube
AVR32SD20-E/SS	32 KB/4 KB	20	SSOP	2.7-5.5V	G3X	-40°C to +125°C	Tube
AVR32SD20T-I/SS	32 KB/4 KB	20	SSOP	2.7-5.5V	G3X	-40°C to +85°C	Tape & Reel
AVR32SD20T-E/SS	32 KB/4 KB	20	SSOP	2.7-5.5V	G3X	-40°C to +125°C	Tape & Reel
AVR32SD28-I/3LW	32 KB/4 KB	28	VQFN	2.7-5.5V	3LW	-40°C to +85°C	Tube
AVR32SD28-I/SS	32 KB/4 KB	28	SSOP	2.7-5.5V	N2X	-40°C to +85°C	Tube
AVR32SD28-I/SP	32 KB/4 KB	28	SPDIP	2.7-5.5V	M3X	-40°C to +85°C	Tube
AVR32SD28-E/3LW	32 KB/4 KB	28	VQFN	2.7-5.5V	3LW	-40°C to +125°C	Tube
AVR32SD28-E/SS	32 KB/4 KB	28	SSOP	2.7-5.5V	N2X	-40°C to +125°C	Tube
AVR32SD28-E/SP	32 KB/4 KB	28	SPDIP	2.7-5.5V	M3X	-40°C to +125°C	Tube
AVR32SD28T-I/3LW	32 KB/4 KB	28	VQFN	2.7-5.5V	3LW	-40°C to +85°C	Tape & Reel
AVR32SD28T-I/SS	32 KB/4 KB	28	SSOP	2.7-5.5V	N2X	-40°C to +85°C	Tape & Reel
AVR32SD28T-E/3LW	32 KB/4 KB	28	VQFN	2.7-5.5V	3LW	-40°C to +125°C	Tape & Reel
AVR32SD28T-E/SS	32 KB/4 KB	28	SSOP	2.7-5.5V	N2X	-40°C to +125°C	Tape & Reel
AVR32SD32-I/QZB	32 KB/4 KB	32	VQFN	2.7-5.5V	QZB	-40°C to +85°C	Tray
AVR32SD32-I/PT	32 KB/4 KB	32	TQFP	2.7-5.5V	T5X	-40°C to +85°C	Tray
AVR32SD32-E/QZB	32 KB/4 KB	32	VQFN	2.7-5.5V	QZB	-40°C to +125°C	Tray
AVR32SD32-E/PT	32 KB/4 KB	32	TQFP	2.7-5.5V	T5X	-40°C to +125°C	Tray
AVR32SD32T-I/QZB	32 KB/4 KB	32	VQFN	2.7-5.5V	QZB	-40°C to +85°C	Tape & Reel
AVR32SD32T-I/PT	32 KB/4 KB	32	TQFP	2.7-5.5V	T5X	-40°C to +85°C	Tape & Reel
AVR32SD32T-E/QZB	32 KB/4 KB	32	VQFN	2.7-5.5V	QZB	-40°C to +125°C	Tape & Reel
AVR32SD32T-E/PT	32 KB/4 KB	32	TQFP	2.7-5.5V	T5X	-40°C to +125°C	Tape & Reel

Figure 48-1. Product Identification System

To order or obtain information, for example on pricing or delivery, refer to the factory or the listed sales office.



Note: The Tape & Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with the Microchip Sales Office for package availability using the Tape & Reel option.

49. Package Drawings

49.1 Online Package Drawings

For the most recent package drawings:

1. Go to www.microchip.com/packaging.
2. Search for the package code to find the most recent package drawing.

Table 49-1. Drawing Numbers

Pin Count	Package Type(1)	Drawing Number	Style	Package Code
20	SSOP	C04-00072	SS	G3X
28	SPDIP	C04-00079	SP	M3X
28	SSOP	C04-00073	SS	N2X
28	VQFN (WF)	C04-00565	3LW	3LW
32	VQFN (WF)	C04-21511	QZB	QZB
32	TQFP	C04-00074	PT	T5X

Note:

1. WF = Wettable Flanks

49.2 Package Marking Information

Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC® designator for Matte Tin (Sn)
Note:	In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.	

49.2.1 20-Pin SSOP

Figure 49-1. General

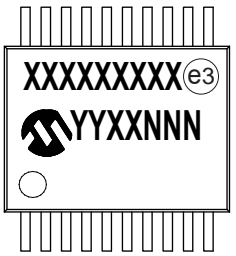
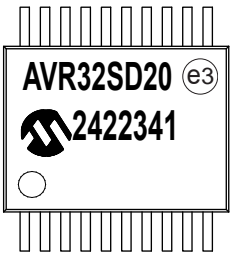


Figure 49-2. Example



49.2.2 28-Pin SSOP

Figure 49-3. General

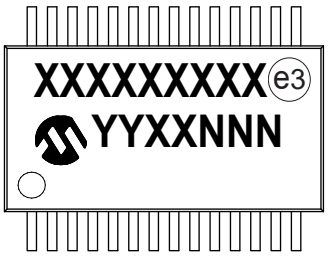
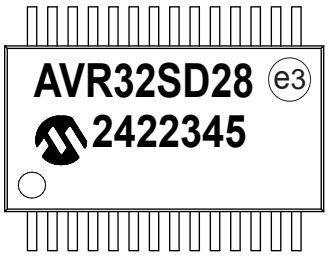


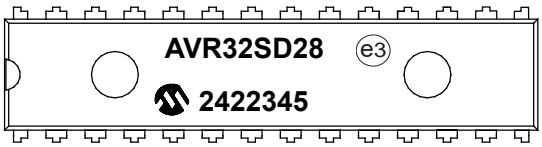
Figure 49-4. Example



49.2.3 28-Pin SPDIP

Figure 49-5. General

Figure 49-6. Example



49.2.4 28-Pin VQFN Wettable Flanks

Figure 49-7. General

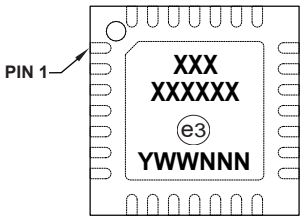
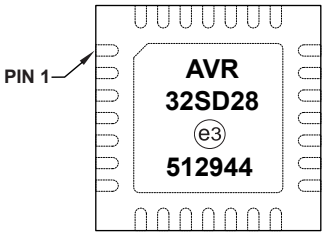


Figure 49-8. Example



49.2.5 32-Pin VQFN Wettable Flanks

Figure 49-9. General

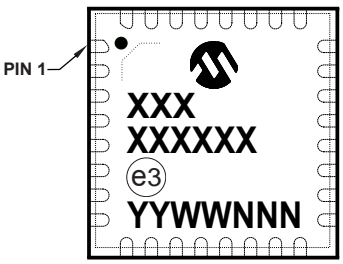
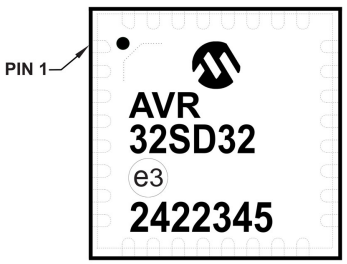


Figure 49-10. Example



49.2.6 32-Pin TQFP

Figure 49-11. General

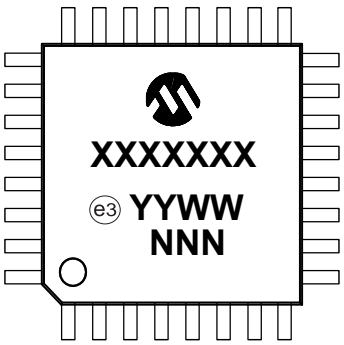
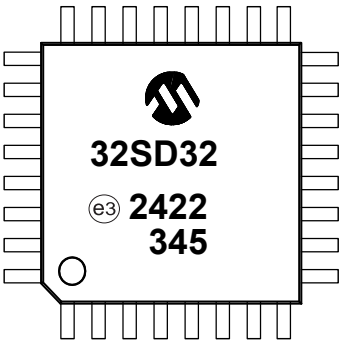


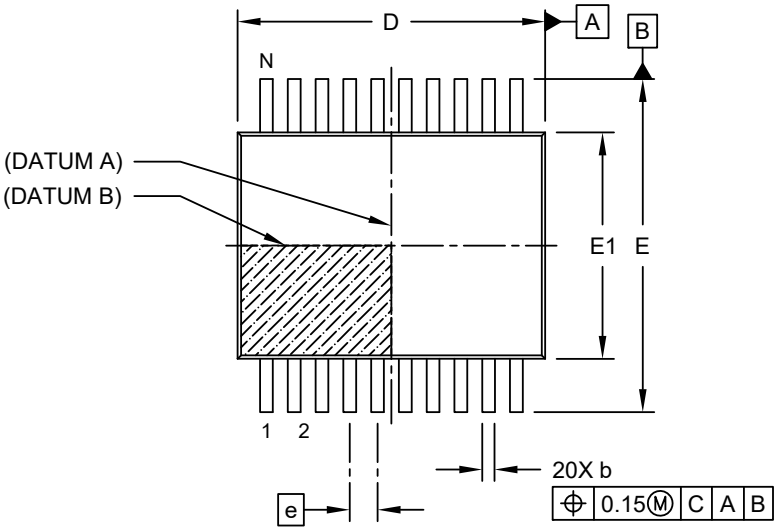
Figure 49-12. Example



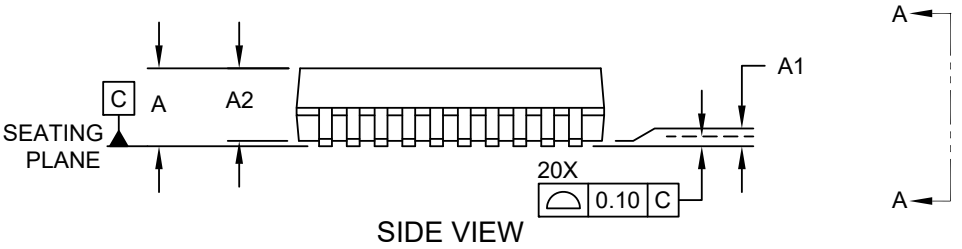
49.3 20-Pin SSOP

20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

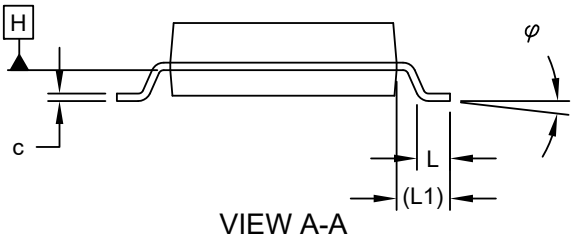
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

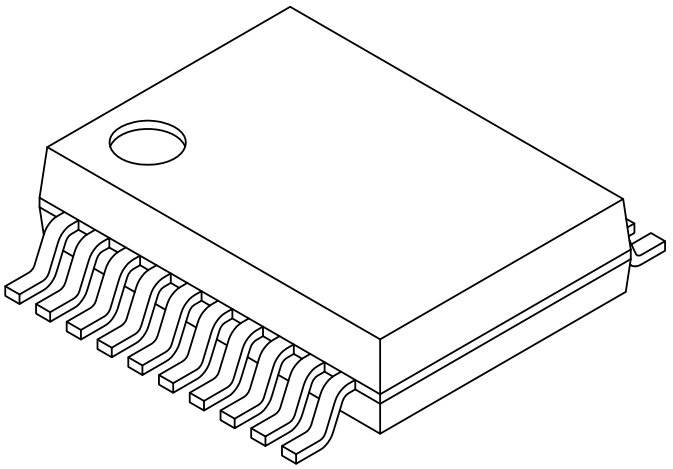


VIEW A-A

Microchip Technology Drawing C04-072 Rev C Sheet 1 of 2

20-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		20	
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	6.90	7.20	7.50
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

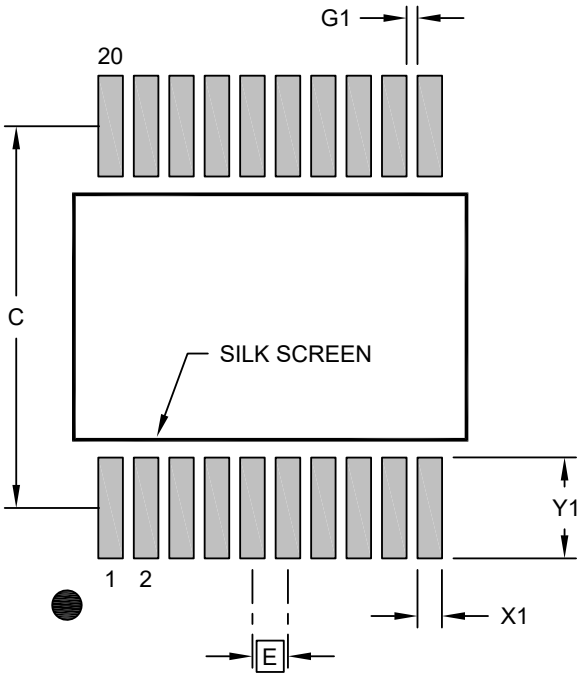
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072 Rev C Sheet 2 of 2

20-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 (X20)	X1			0.45
Contact Pad Length (X20)	Y1			1.85
Contact Pad to Center Pad (X18)	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 C04-2072 Rev C

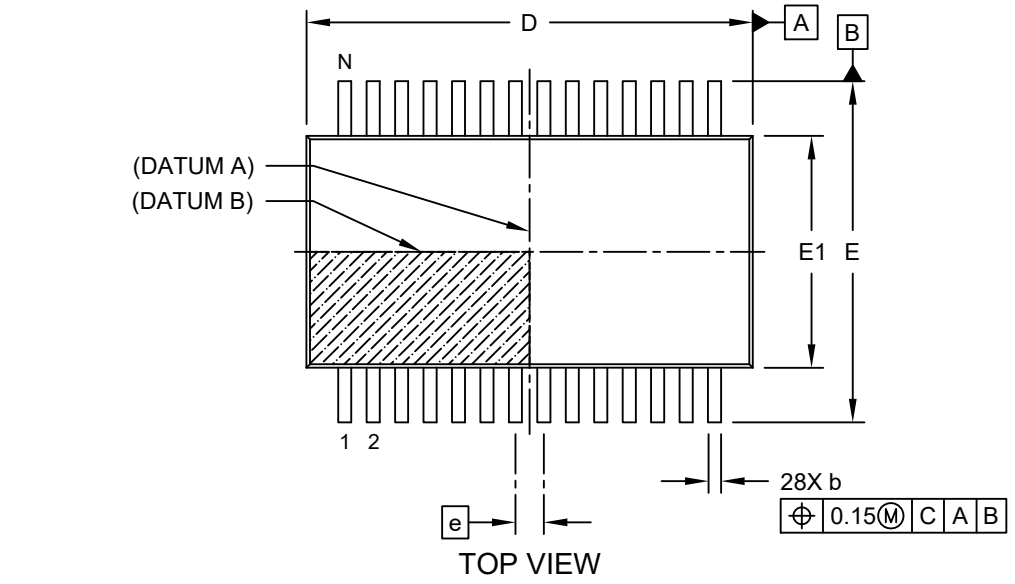
Table 49-2. Package Code

G3X

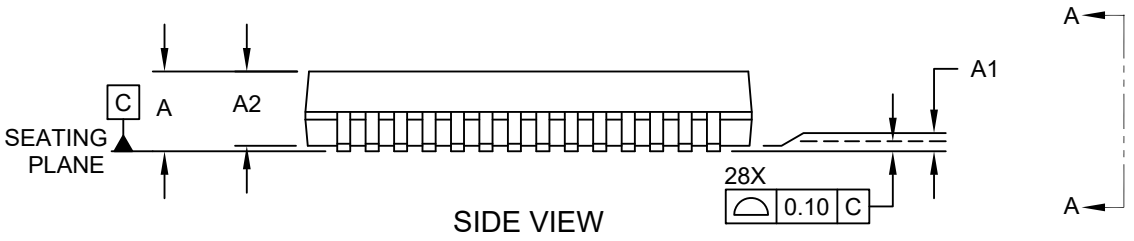
49.4 28-Pin SSOP

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

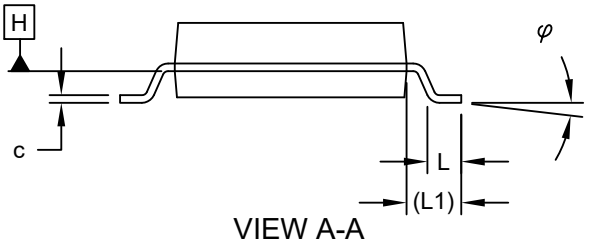
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

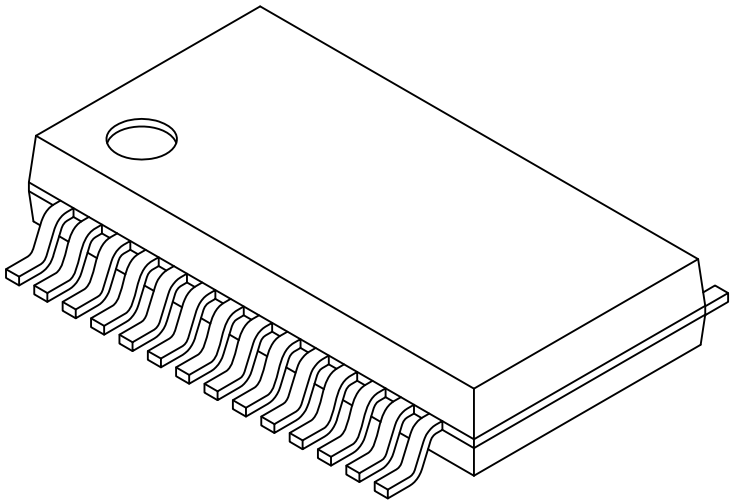


VIEW A-A

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

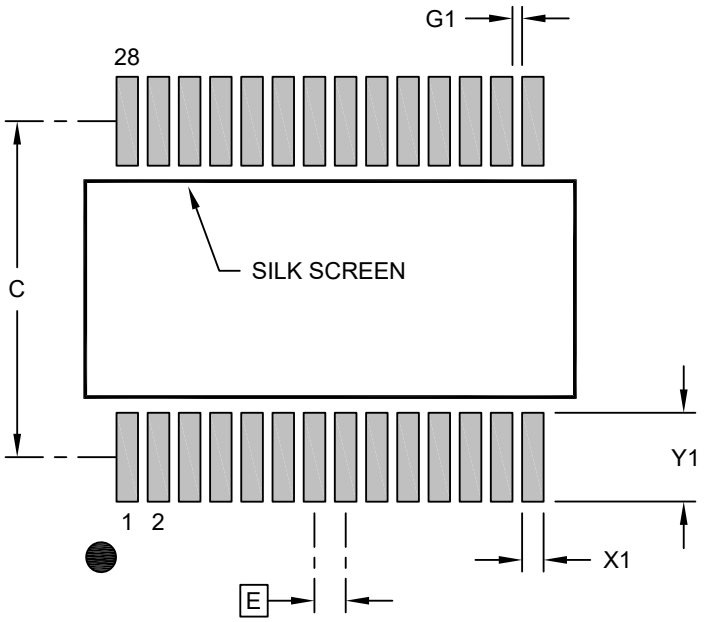
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   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

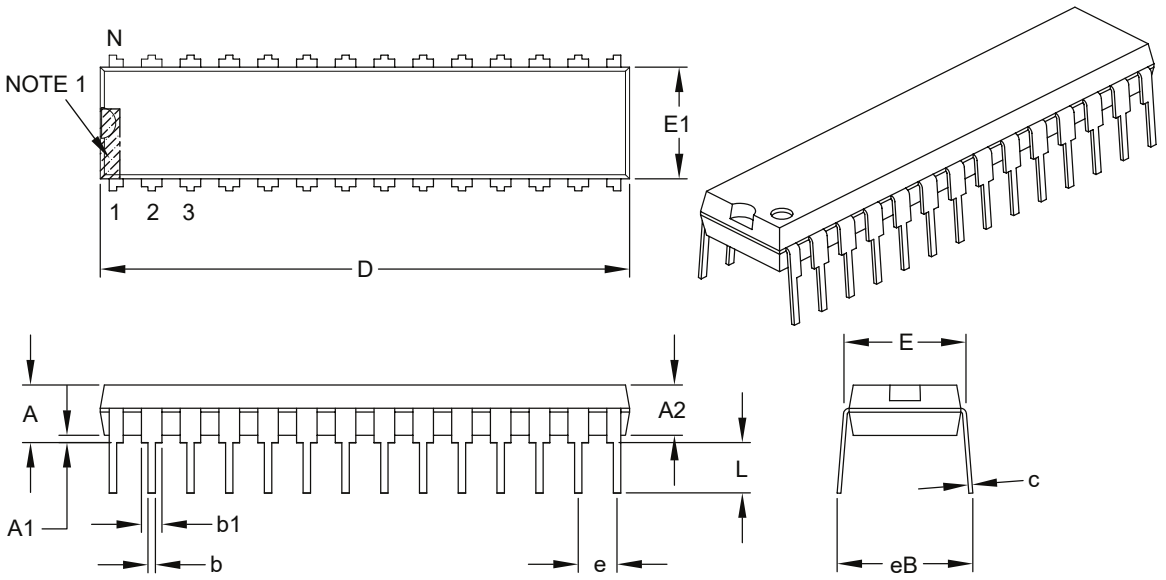
Table 49-3. Package Code

N2X

49.5 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

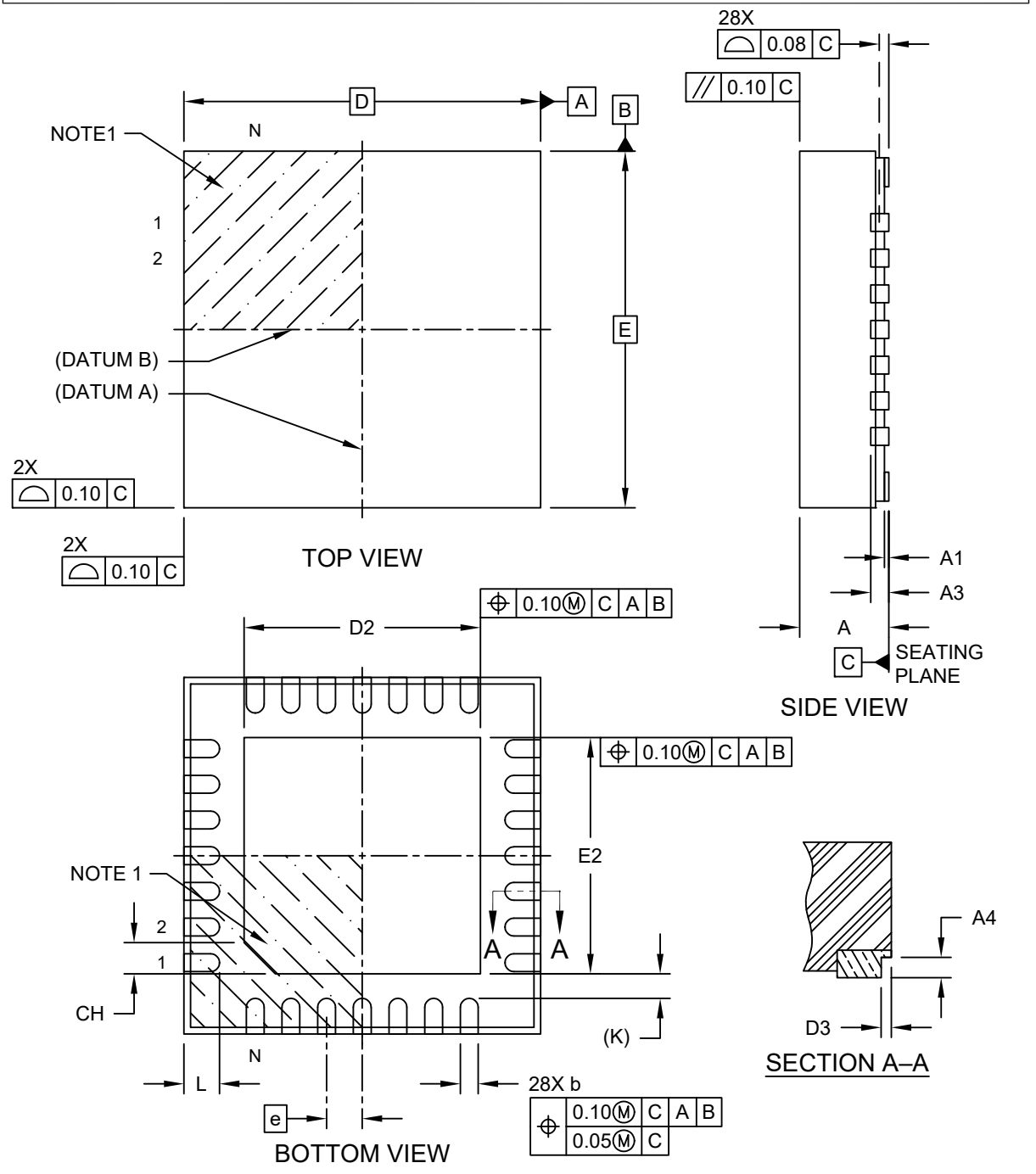
Table 49-4. Package Code

M3X

49.6 28-Pin VQFN Wettable Flanks

28-Lead Very Thin Plastic Quad Flat, No Lead Package (3LW) – 4x4x1.0 mm Body [VQFN] With 2.65 mm Exposed Pad and Stepped Wettable Flanks

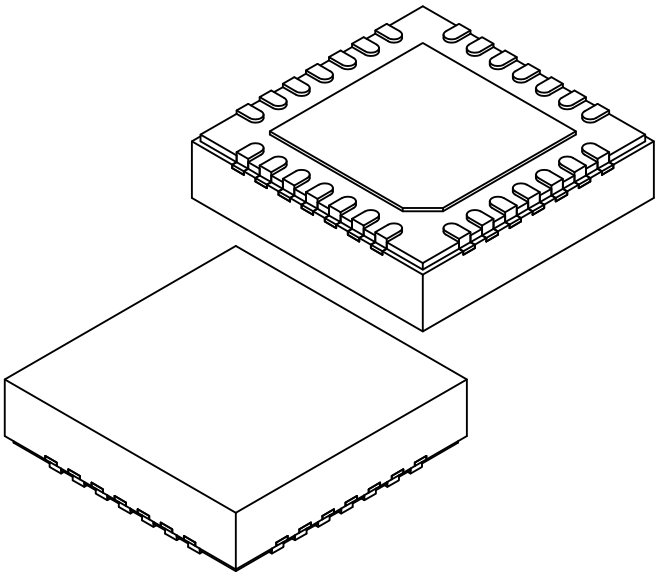
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-565-3LW Rev B Sheet 1 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead Package (3LW) – 4x4x1.0 mm Body [VQFN] With 2.65 mm Exposed Pad and Stepped Wettable Flanks

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		28	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		4.00 BSC	
Exposed Pad Length	D2	2.55	2.65	2.75
Overall Width	E		4.00 BSC	
Exposed Pad Width	E2	2.55	2.65	2.75
Exposed Pad Index 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		0.275 REF	
Wettable Flank Step Cut Length	D3	–	–	0.085
Wettable Flank Step Cut Height	A4	0.10	–	0.19

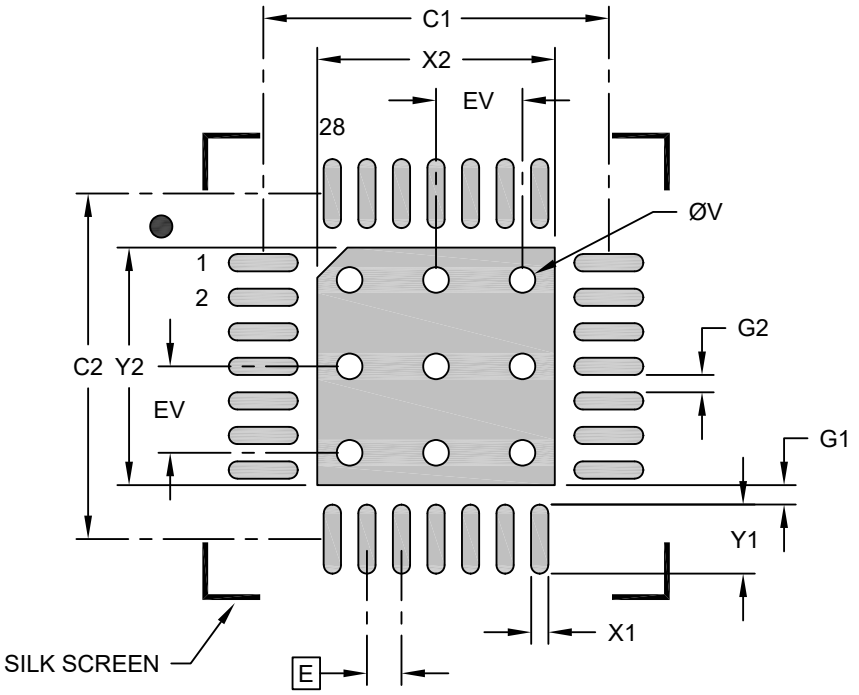
Notes:

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 C04-565-3LW Rev B Sheet 2 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead Package (3LW) – 4x4x1.0 mm Body [VQFN] With 2.65 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.40 BSC	
Center Pad Width	X2			2.75
Center Pad Length	Y2			2.75
Contact Pad Spacing	C1		4.00	
Contact Pad Spacing	C2		4.00	
Contact Pad Width (X28)	X1			0.20
Contact Pad Length (X28)	Y1			0.80
Contact Pad to Center Pad (X28)	G1	0.23		
Contact Pad to Contact Pad (X24)	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 C04-2565-3LW Rev B

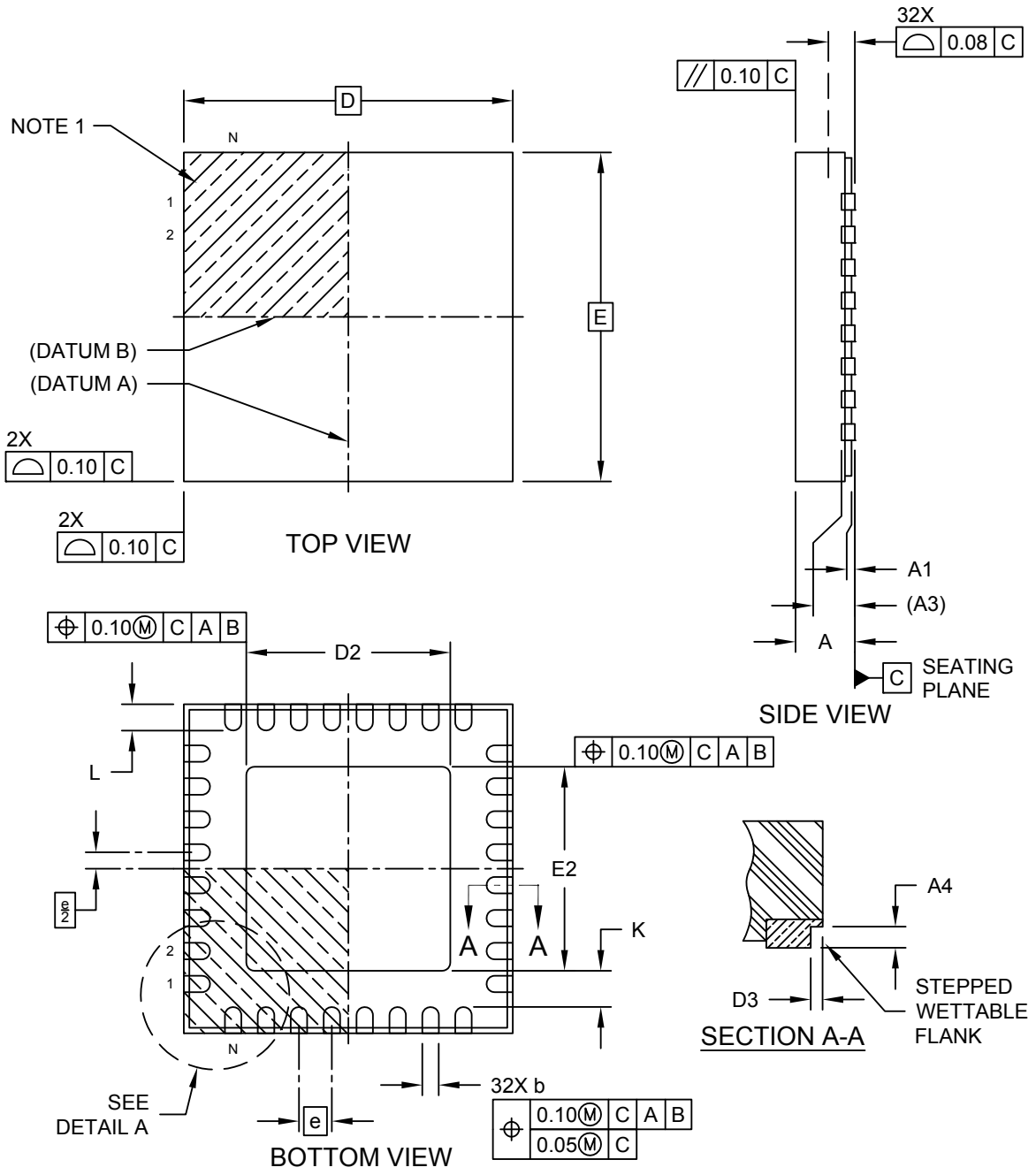
Table 49-5. Package Code

3LW

49.7 32-Pin VQFN Wettable Flanks

32-Lead Ultra Thin Plastic Quad Flat, No Lead Package (QZB) - 5x5x0.9 mm Body [VQFN] With 3.1 mm Exposed Pad and Stepped Wettable Flanks

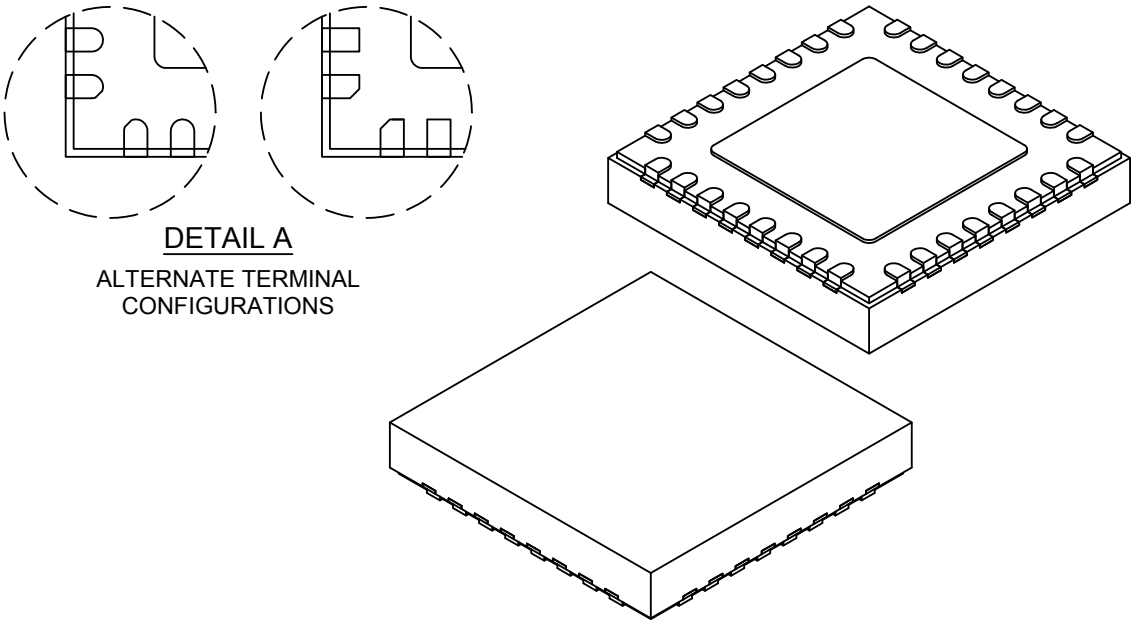
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-21511 Rev A Sheet 1 of 2

32-Lead Ultra Thin Plastic Quad Flat, No Lead Package (QZB) - 5x5x0.9 mm Body [VQFN] With 3.1 mm Exposed Pad and Stepped Wettable Flanks

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		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	-	-
Wettable Flank Step Length	D3	-	-	0.085
Wettable Flank Step Height	A4	0.10	-	0.19

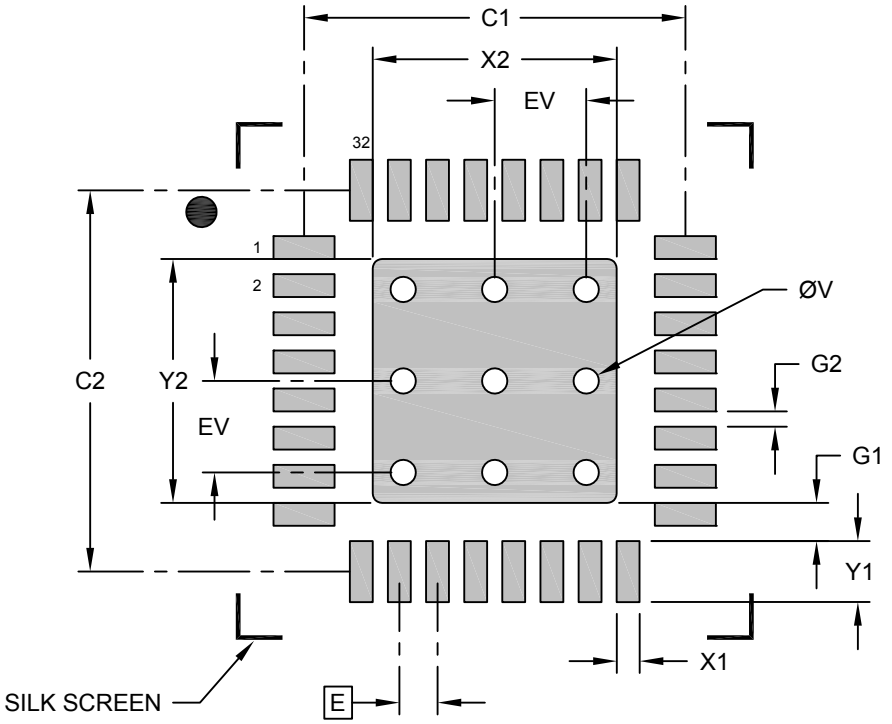
Notes:

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 C04-21511 Rev A Sheet 1 of 2

32-Lead Ultra Thin Plastic Quad Flat, No Lead Package (QZB) - 5x5x0.9 mm Body [VQFN] With 3.1 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

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 C04-23511 Rev A

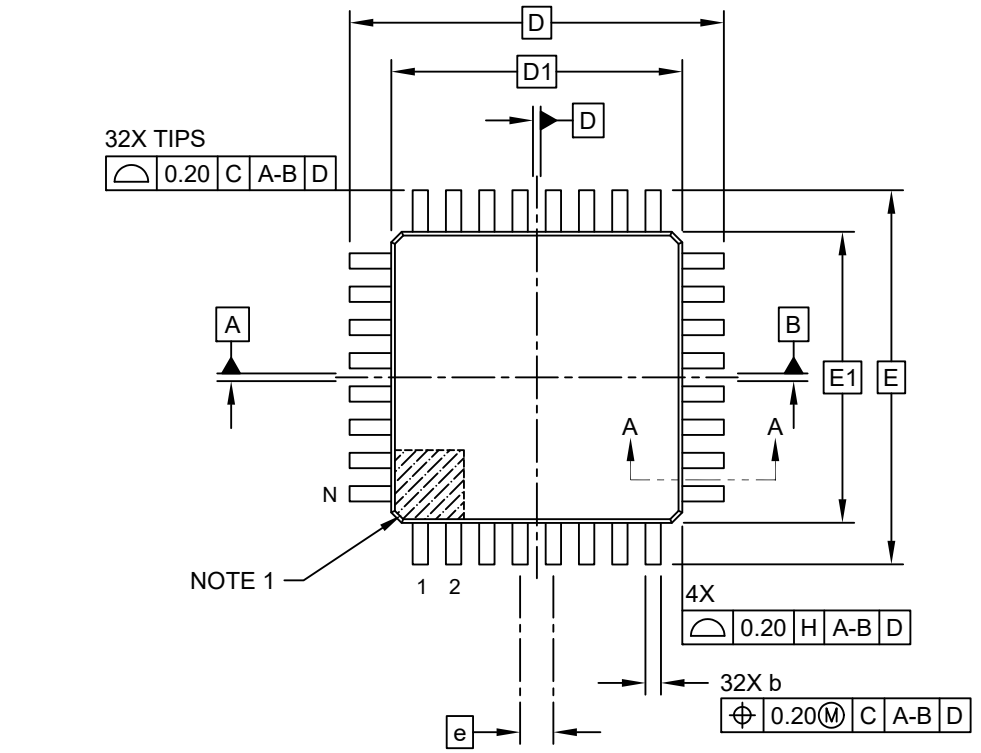
Table 49-6. Package Code

QZB

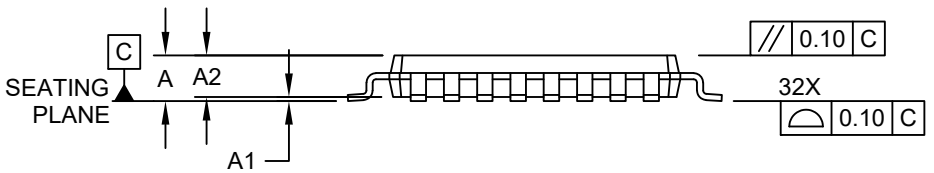
49.8 32-Pin TQFP

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

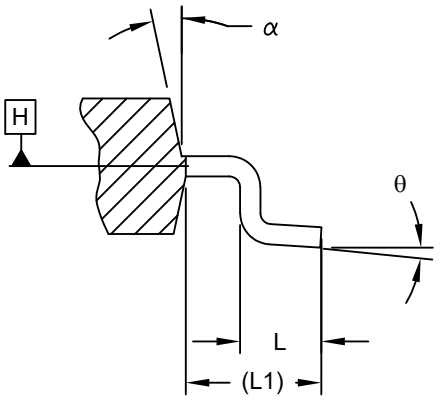


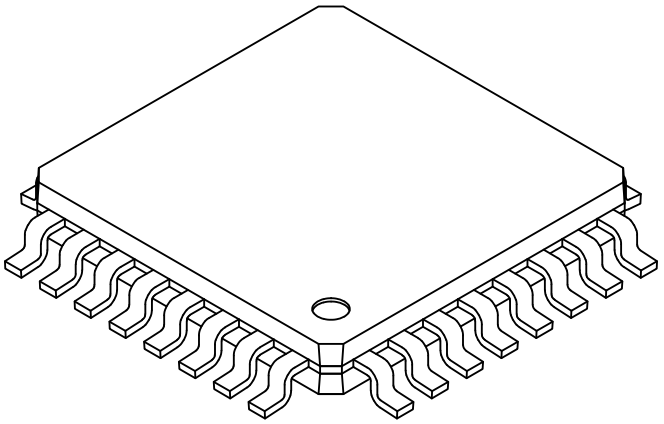
SIDE VIEW

Microchip Technology Drawing C04-074-PT Rev D 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:

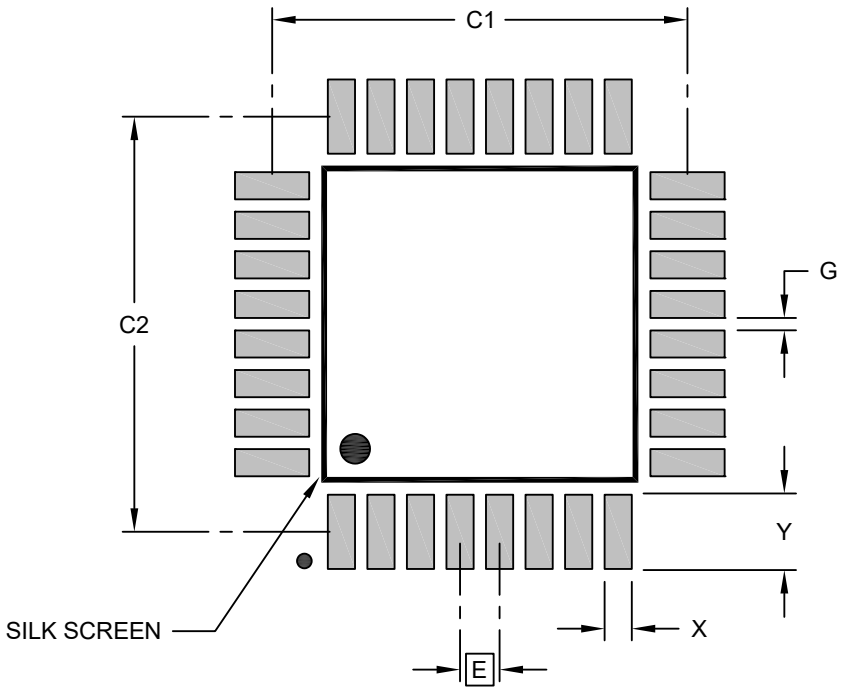
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-074-PT Rev D Sheet 2 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



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 (X32)	X			0.55
Contact Pad Length (X32)	Y			1.55
Contact Pad to Contact Pad (X28)	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-PT Rev D

Table 49-7. Package Code

T5X

50. 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).

50.1 Revision History

Document Rev.	Date	Changes
A	02/2025	Initial release of preliminary data sheet

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-0576-5

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications 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. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

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.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- 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, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- 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. Microchip is committed to continuously improving the code protection features of our products.