AVR64DU28/32 Preliminary Data Sheet

AVR64DU28/32

Introduction

The AVR® DU Family of microcontrollers uses the AVR® CPU with a hardware multiplier running at clock speeds up to 24 MHz. They include 16/32/64 KB of Flash, 2/4/8 KB of SRAM, and 256 bytes of EEPROM. The microcontrollers are available in 14-, 20-, 28-, and 32-pin packages. The family uses the latest technologies from Microchip Technology, with a flexible and low-power architecture, including an Event System, intelligent analog features, and advanced digital peripherals such as a USB 2.0 full-speed device.

Features

- AVR® CPU
  - Running at up to 24 MHz
  - Single-cycle I/O access
  - Two-level interrupt controller
  - Two-cycle hardware multiplier
  - Program and Debug Interface Disable (PDID)
  - Supply voltage range: 1.8-5.5V
- Memories
  - 16/32/64 KB in-system self-programmable Flash memory with a genuine read-while-write operation
  - 2/4/8 KB SRAM
  - 256B EEPROM
  - 512B of user row in nonvolatile memory that can keep data during chip erase and be programmed while the device is locked
  - 256B of boot row
  - Write/erase endurance
    - Flash 1,000 cycles
    - EEPROM 100,000 cycles
  - Data retention: 40 years at 55°C
- System
  - Power-on Reset (POR) circuit
  - Brown-out Detector (BOD)
  - Voltage Level Monitor (VLM) with interrupt at a programmable level above the BOD
  - Clock options
    - High-precision internal high-frequency oscillator with selectable frequency up to 24 MHz (OSCHF)
      - Auto-tuning for improved internal oscillator accuracy
    - 32.768 kHz internal oscillator (OSC32K)
    - 32.768 kHz external crystal oscillator (XOSC32K)
    - External clock input
    - External high-frequency crystal oscillator (XOSCHF) with clock failure detection
  - Single-pin Unified Program and Debug Interface (UPDI)
  - Three sleep modes
    - Idle with all peripherals running for immediate wake-up
    - Standby with a configurable operation of selected peripherals
    - Power-Down with full data retention
  - Automated Cyclic Redundancy Check (CRC) Flash memory scan
  - Watchdog Timer (WDT) with Window mode, with a separate on-chip oscillator
  - External interrupt on all general purpose pins
- Peripherals
  - One 16-bit Timer/Counter type A (TCA) with three compare channels for Pulse-Width Modulation (PWM) and waveform generation

  - Two 16-bit Timer/Counter type B (TCB) with input capture and signal measurements
  - One 16-bit Real-Time Counter (RTC) that can run from an external crystal or internal oscillator
  - One USB 2.0 full-speed (12 Mbps) device-compliant interface[1]
    - Optional internal 3.3V voltage regulator
    - OSCHF oscillator can be tuned to the USB Start-of-Frames (SOFs) for crystal-less operation
    - 16 endpoint addresses each, with one input and one output endpoint for up to 32 endpoints
    - Multipacket transfer for reduced interrupt load and software intervention
  - Two USARTs
    - Operation modes: RS-485, LIN client, SPI host, and IrDA
    - Fractional baud rate generator, auto-baud, and start-of-frame detection
  - One SPI with host/client operation modes
  - One Two-Wire Interface (TWI) with dual address match
    - Simultaneous host/client operation (dual mode)
    - Philips Inter-Integrated Circuit (I^2C) compatible
    - Standard mode (Sm, 100 kHz)
    - Fast mode (Fm, 400 kHz)
    - Fast mode plus (Fm+, 1 MHz)[2]
  - Event System for CPU independent and predictable inter-peripheral signaling
  - Configurable Custom Logic (CCL) with four programmable Look-up Tables (LUTs)
  - One 10-bit 170 ksps Analog-to-Digital Converter (ADC)
  - One Analog Comparator (AC)
  - Internal 1.024V, 2.048V, 2.500V and 4.096V voltage references and external reference option (VREF)
- I/O and Packages:
  - Up to 25 (24) programmable GPIO (I/O) pins
  - 32-pin VQFN 5x5 mm and TQFP 7x7 mm
  - 28-pin VQFN 4x4 mm, SPDIP and SSOP
  - 20-pin VQFN 3x3 mm and SSOP
  - 14-pin SOIC
- Temperature Ranges:
  - Industrial: -40°C to +85°C

Notes:

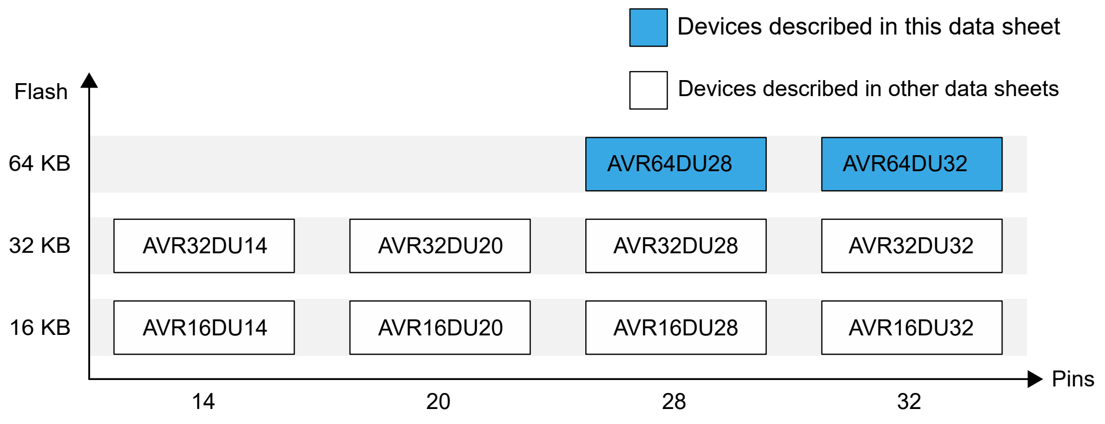
1. USB function is only available for VDD above 3.0V.
2. I^2C Fm+ is only supported for 2.7V and above.

AVR® DU Family Overview

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

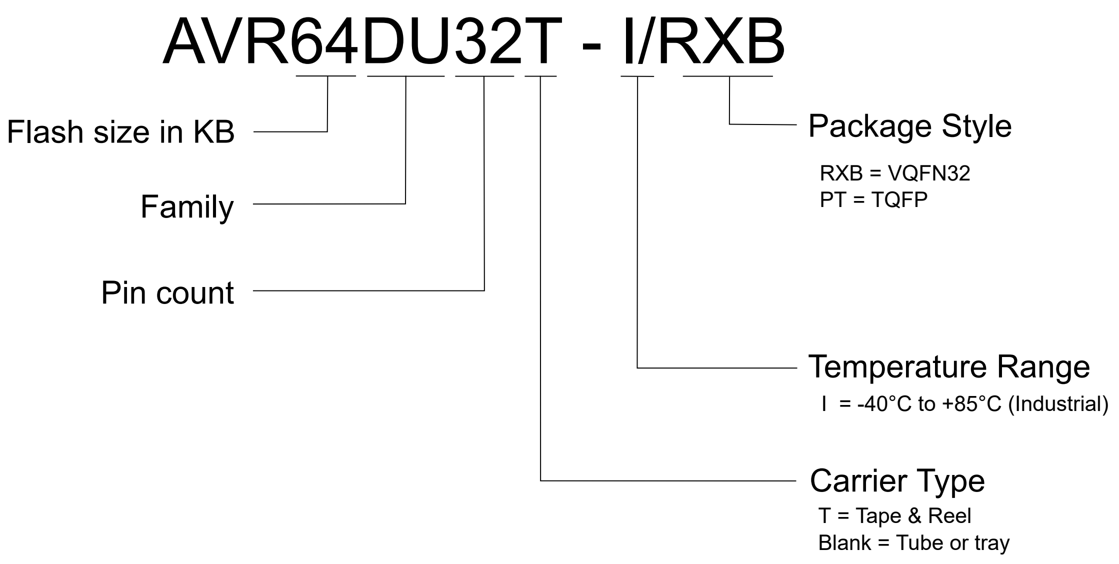
- Vertical migration is possible without code modification, as these devices are entirely pin and feature compatible
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1. AVR® DU Family Overview



The name of a device in the AVR DU Family is decoded as follows:

Figure 2. AVR® DU Device Designations



Memory Overview

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

Table 1. Memory Overview

Devices	AVR16DU14 AVR16DU20 AVR16DU28 AVR16DU32	AVR32DU14 AVR32DU20 AVR32DU28 AVR32DU32	AVR64DU28 AVR64DU32
Flash memory	16 KB	32 KB	64 KB
Flash read-while-write sectioning - bottom NRWW /upper RWW	4 KB/12 KB	4 KB/28 KB	8 KB/56 KB
SRAM	2 KB	4 KB	8 KB
EEPROM	256B	256B	256B
User row	512B	512B	512B
Boot row	256B	256B	256B

Peripheral Overview

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

Table 2. Peripheral Overview

Feature	AVR16DU14 AVR32DU14	AVR16DU20 AVR32DU20	AVR16DU28 AVR32DU28 AVR64DU28	AVR16DU32 AVR32DU32 AVR64DU32
Pins	**14**	**20**	**28**	**32**
Max. frequency (MHz)	24	24	24	24
16-bit Timer/Counter type A (TCA)	1	1	1	1
16-bit Timer/Counter type B (TCB)	2	2	2	2
Real-Time Counter (RTC)	1	1	1	1
USB 2.0 full-speed device	1	1	1	1
USART	2	2	2	2
SPI	1	1	1	1
TWI/I^{2}C(1)	1(1)	1(1)	1(1)	1(1)
10-bit ADC (channels)	1 (5)	1 (11)	1 (17)	1 (21)
Analog Comparator (AC)	1	1	1	1
Peripheral Touch Controller (PTC)	-	-	-	-
Operational amplifier (OP)	-	-	-	-
Configurable Custom Logic Look-up Table (CCL LUT)	4	4	4	4
Watchdog Timer (WDT)	1	1	1	1
Event System (EVSYS) channels	6	6	6	6
General Purpose I/O(2)	9/8(2)	15/14(2)	21/20(2)	25/24(2)
PORT	PA[1:0], PC[3], PD[7:4], PF[7:6]	PA[7:0], PC[3], PD[7:4], PF[7:6]	PA[7:0], PC[3], PD[7:0], PF[7,6,1,0]	PA[7:0], PC[3], PD[7:0], PF[7:0]
External interrupts	9	15	21	25
CRCSCAN	1	1	1	1
Unified Program and Debug Interface (UPDI)	1	1	1	1

Notes:

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

Security Concept

The AVR DU 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 Unified Program and Debug Interface (UPDI). After activating the PDID as described in the section *Memories*, the UPDI is prevented from making any changes to the device. The 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 UPDI.

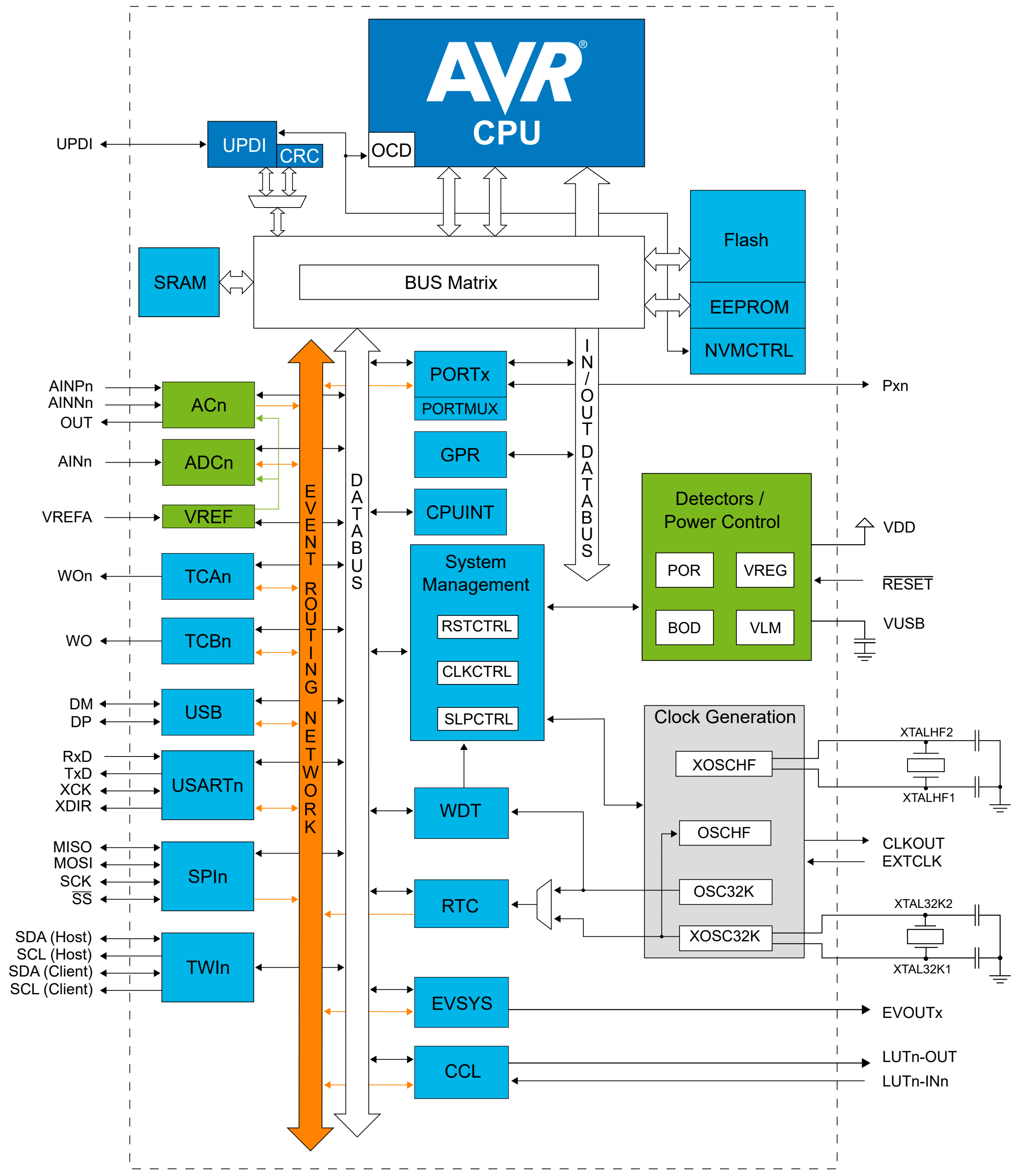
In addition, there is a separate storage space accessible only by code in the Boot Code section, which can hold any data intended to be accessible only from the Boot Code section. One example of this is a cryptographic key to be used to validate data that are sent to a bootloader to update the application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

This creates a two-layer security: The device is prevented from being erased or reprogrammed over the UPDI, and 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.

Using the *Program and Debug Interface Disable (PDID)* in software requires cryptographic competencies to ensure conformity to cyber-security standards such as ISO/SAE DIS 21434.

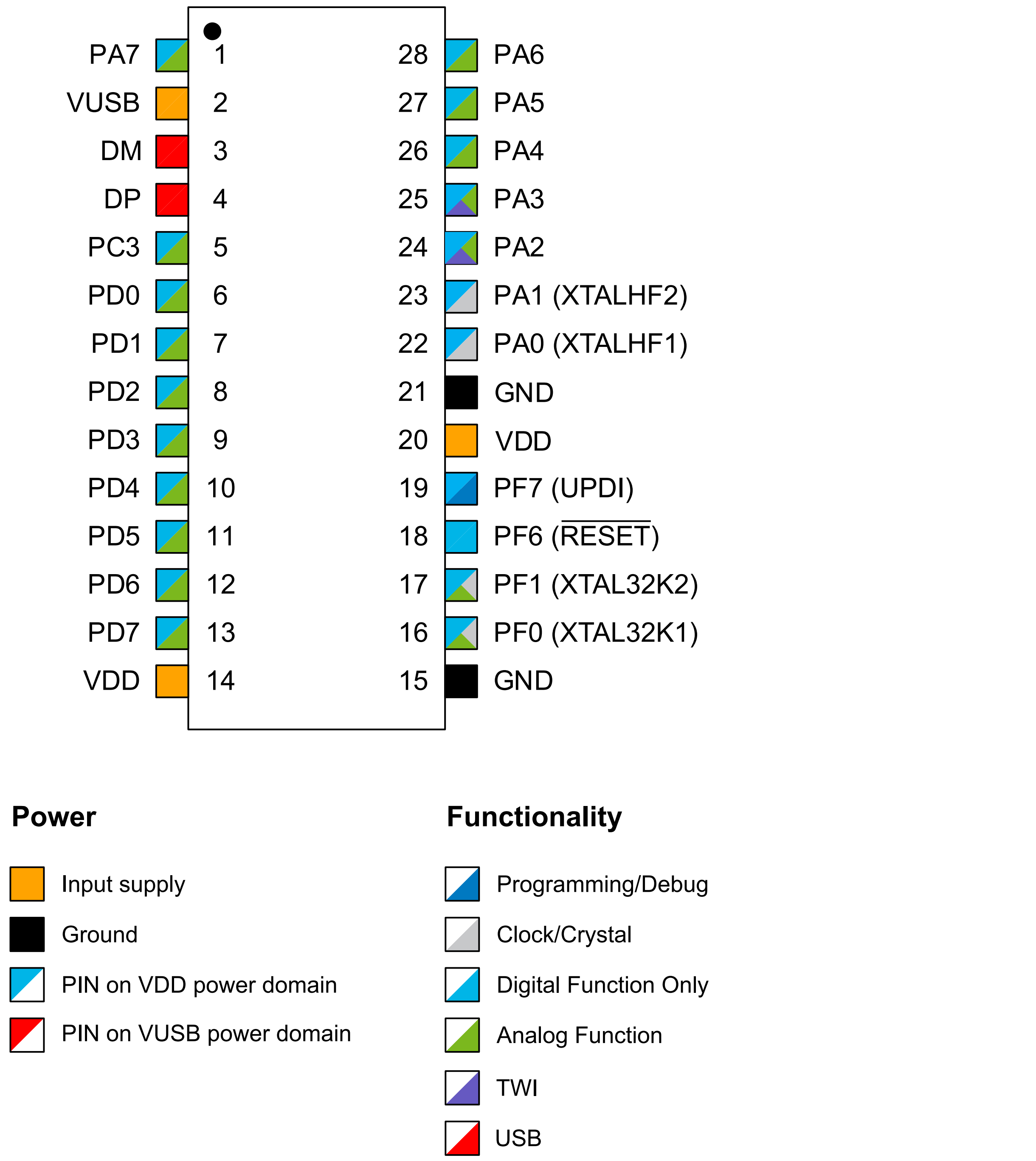
Table of Contents

1. Block Diagram

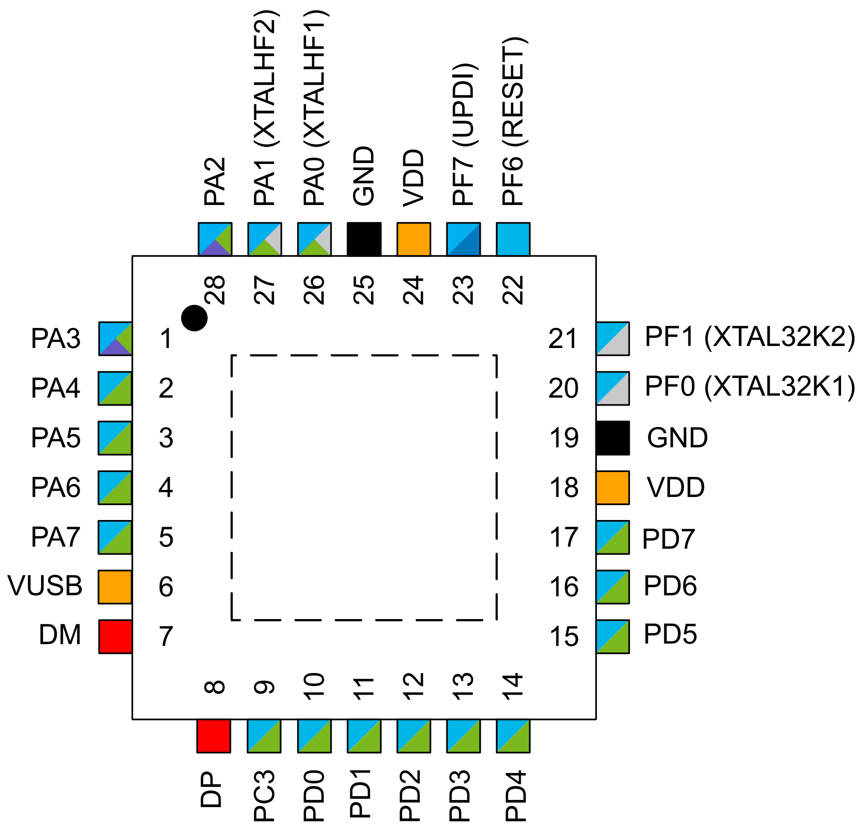


2. Pinout

2.1 28-Pin SPDIP and SSOP



2.2 28-Pin VQFN



Power

🟧 Input supply

⬛ Ground

◩ PIN on VDD power domain

◩ PIN on VUSB power domain

Functionality

◩ Programming/Debug

◩ Clock/Crystal

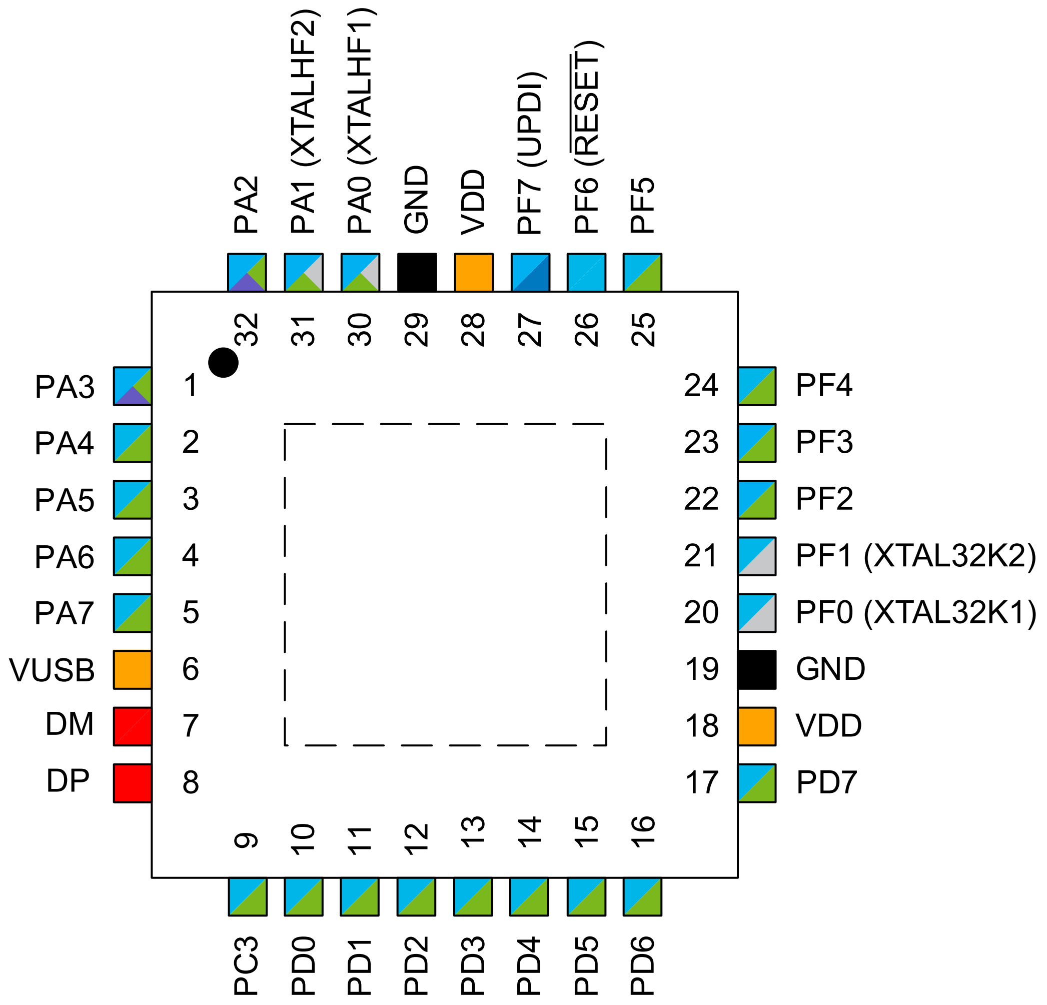
◩ Digital Function Only

◩ Analog Function

◩ TWI

◩ USB

2.3 32-Pin TQFP and VQFN



Power		**Functionality**	
🟧	Input supply	◩	Programming/Debug
⬛	Ground	◩	Clock/Crystal
◩	PIN on VDD power domain	◩	Digital Function Only
◩	PIN on VUSB power domain	◩	Analog Function
		◩	TWI
		◩	USB

3. I/O Multiplexing and Considerations

3.1 I/O Multiplexing

VQFN32 TQFP32	SPDIP28 SSOP28	VQFN28	Pin name (1,2)	Special	ADC0	AC0	USB	USARTn	SPI0	TWI0(4)	TCA0	TCBn	EVSYS	CCL
30	22	26	PA0	XTALHF1 EXTCLK				0,TxD		SDA(HC)(3)	WO0			LUT0,IN0
31	23	27	PA1	XTALHF2				0,RxD		SCL(HC)(3)	WO1			LUT0,IN1
32	24	28	PA2		AIN22			0,XCK 0,TxD(3)		SDA(HC)	WO2	0,WO	EVOUTA	LUT0,IN2
1	25	1	PA3		AIN23			0,XDIR 0,RxD(3)		SCL(HC)	WO3	1,WO		LUT0,OUT
2	26	2	PA4		AIN24			0,TxD(3)	MOSI		WO4			
3	27	3	PA5		AIN25			0,RxD(3)	MISO		WO5			
4	28	4	PA6		AIN26			0,XCK(3)	SCK					LUT0,OUT(3)
5	1	5	PA7	CLKOUT	AIN27	OUT		0,XDIR(3)	$\overline{SS}$				EVOUTA(3)	
6	2	6	VUSB				VUSB							
7	3	7	DM				DM							
8	4	8	DP				DP							
9	5	9	PC3		AIN31	AINP4					WO3(3)			LUT1,OUT
10	6	10	PD0		AIN0	AINN1					WO0(3)			LUT2,IN0
11	7	11	PD1		AIN1						WO1(3)			LUT2,IN1
12	8	12	PD2		AIN2	AINP0					WO2(3)		EVOUTD	LUT2,IN2
13	9	13	PD3		AIN3	AINN0					WO3(3)			LUT2,OUT
14	10	14	PD4		AIN4			0,TxD(3)	MOSI(3)		WO4(3)			
15	11	15	PD5		AIN5			0,RxD(3)	MISO(3)		WO5(3)			
16	12	16	PD6		AIN6	AINP3		0,XCK(3) 1,TxD(3)	SCK(3)					LUT2,OUT(3)
17	13	17	PD7	VREFA	AIN7	AINN2		0,XDIR(3) 1,RxD(3)	$\overline{SS}$(3)				EVOUTD(3)	
18	14	18	VDD											
19	15	19	GND											
20	16	20	PF0	XTAL32K1	AIN16						WO0(3)			LUT3,IN0
21	17	21	PF1	XTAL32K2	AIN17						WO1(3)			LUT3,IN1
22			PF2		AIN18						WO2(3)		EVOUTF	LUT3,IN2
23			PF3		AIN19						WO3(3)			LUT3,OUT
24			PF4		AIN20						WO4(3)	0,WO(3)		
25			PF5		AIN21						WO5(3)	1,WO(3)		
26	18	22	PF6(5)	$\overline{RESET}$										
27	19	23	PF7	UPDI									EVOUTF(3)	
28	20	24	VDD											
29	21	25	GND											

Notes:

1. The pin names are P*xn* type, with *x* being the PORT instance (A, B, C, ...) and *n*, the pin number. The notation for signals is PORT*x*_PIN*n*. All pins can be used as event input.
2. All pins can be used for external interrupt.
3. Alternate pin positions. For selecting the alternate positions, refer to the *PORTMUX - Port Multiplexer* section.
4. The TWI pins that can be used as host or client are marked *H*. The pins with client-only are marked *C*.
5. Input-only.

4. Conventions

4.1 Numerical Notation

Table 4-1. 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

4.2 Memory Size and Type

Table 4-2. 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

4.3 Frequency and Time

Table 4-3. 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

4.4 Registers and Bits

Table 4-4. 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.

4.4.1 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 4-1. 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:

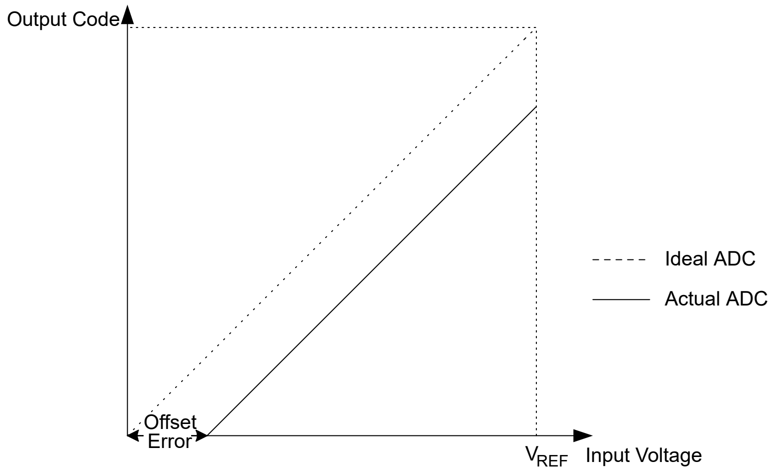
```
// TCA0 in Normal Mode (SINGLE) uses waveform generator in frequency mode
TCA0.SINGLE.CTRL=TCA_SINGLE_WGMODE_FRQ_gc;
```

4.5 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:

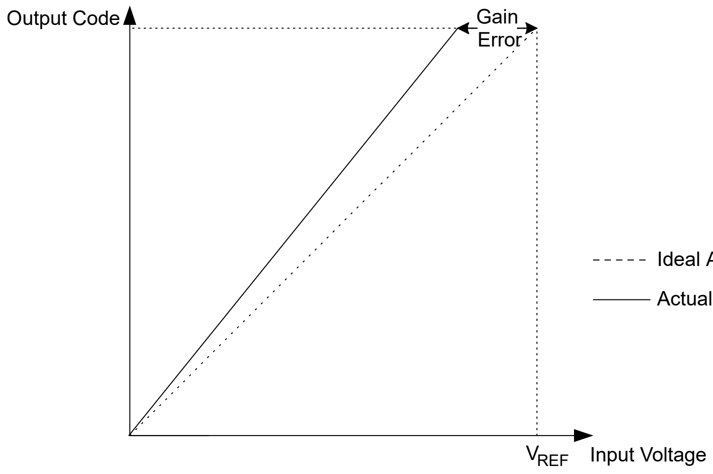
Offset Error The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSb). Ideal value: 0 LSb.

Figure 4-1. Offset Error



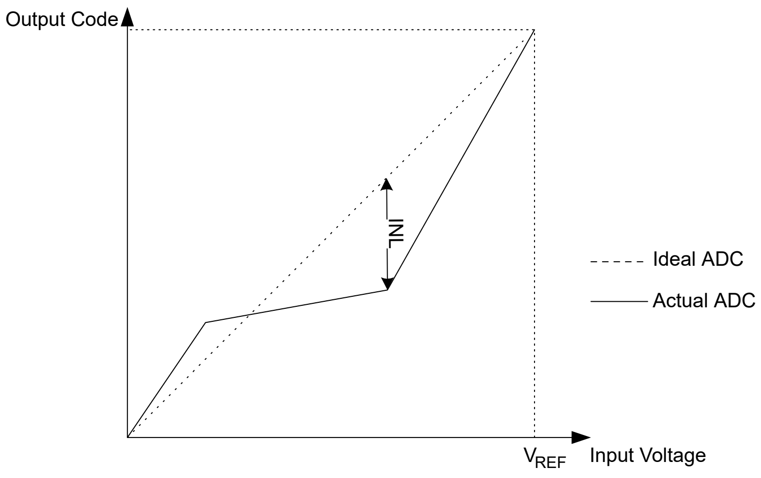
Gain Error After adjusting for offset, the gain error is found as the deviation of the last transition (e.g., 0x3FE to 0x3FF for a 10-bit ADC) compared to the ideal transition (at 1.5 LSb below maximum). Ideal value: 0 LSb.

Figure 4-2. Gain Error



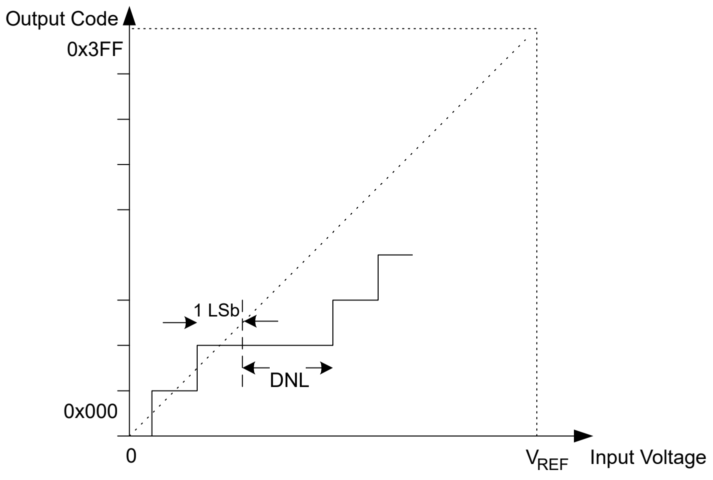
Integral Nonlinearity (INL) After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSb.

Figure 4-3. 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 4-4. 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.

5. Hardware Guidelines

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

This section covers the following topics:

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

5.1 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*

5.1.1 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).

5.2 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.

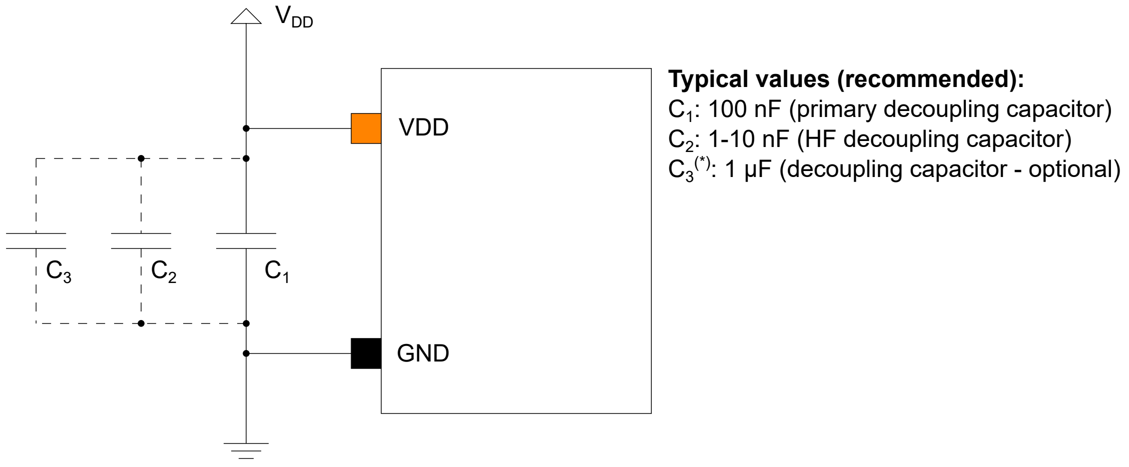
As mentioned at the beginning of this section, all values used in examples are typical values. The actual design may require other values.

5.2.1 Digital 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 5-1. Recommended VDD Connection Circuit Schematic


V_{DD}
VDD
GND
C_3
C_2
C_1
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)

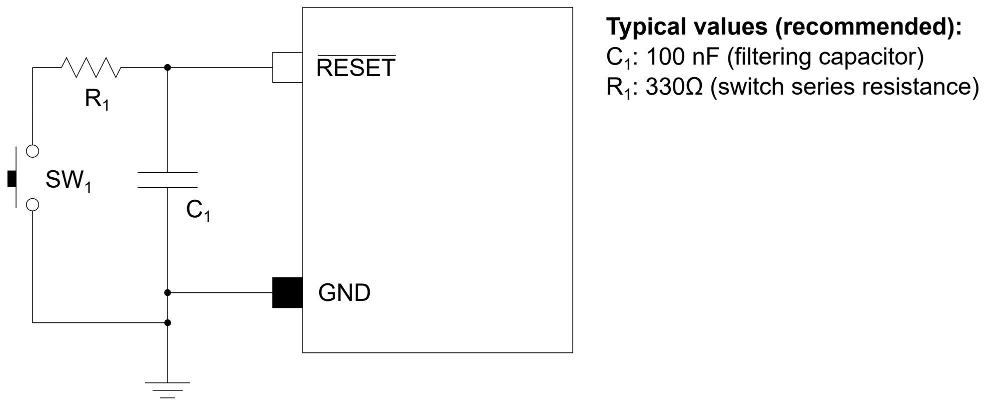
Important: For systems that frequently cycle V_{DD} or experience fast V_{DD} transients, it is recommended to add a decoupling capacitor (C_3) if the power supply slew rate exceeds the slew rate limits. Refer to the *Supply Voltage* section in the *Electrical Characteristics* for details about the power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

5.3 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 5-2. Recommended External Reset Circuit Schematic


Typical values (recommended):
C_1: 100 nF (filtering capacitor)
R_1: 330Ω (switch series resistance)
$\overline{\text{RESET}}$
GND
R_1
SW_1
C_1

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.

UPDI Enable with High-Voltage Override

It is possible to enable a disabled UPDI by applying a high-voltage pulse on the $\overline{\text{RESET}}$ pin. Take care with the reset circuit design and any components connected to the $\overline{\text{RESET}}$ pin to prevent damage if such a high-voltage pulse may be applied.

See the *Connection for UPDI Programming* sub-section and the *UPDI* section for more details.

5.4 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.

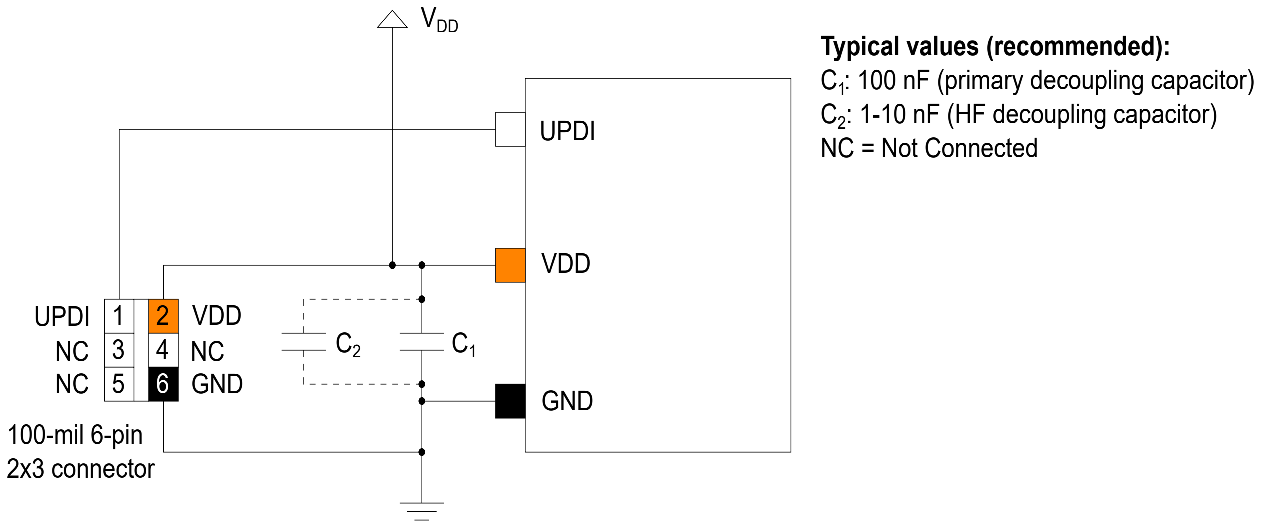
5.4.1 UPDI Connection v1

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 5-3. Recommended UPDI Programming Circuit Schematic


V_{DD}
Typical values (recommended):
C_1: 100 nF (primary decoupling capacitor)
C_2: 1-10 nF (HF decoupling capacitor)
NC = Not Connected
UPDI
VDD
GND
UPDI 1
2 VDD
NC 3
4 NC
NC 5
6 GND
C_2
C_1
100-mil 6-pin
2x3 connector

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.

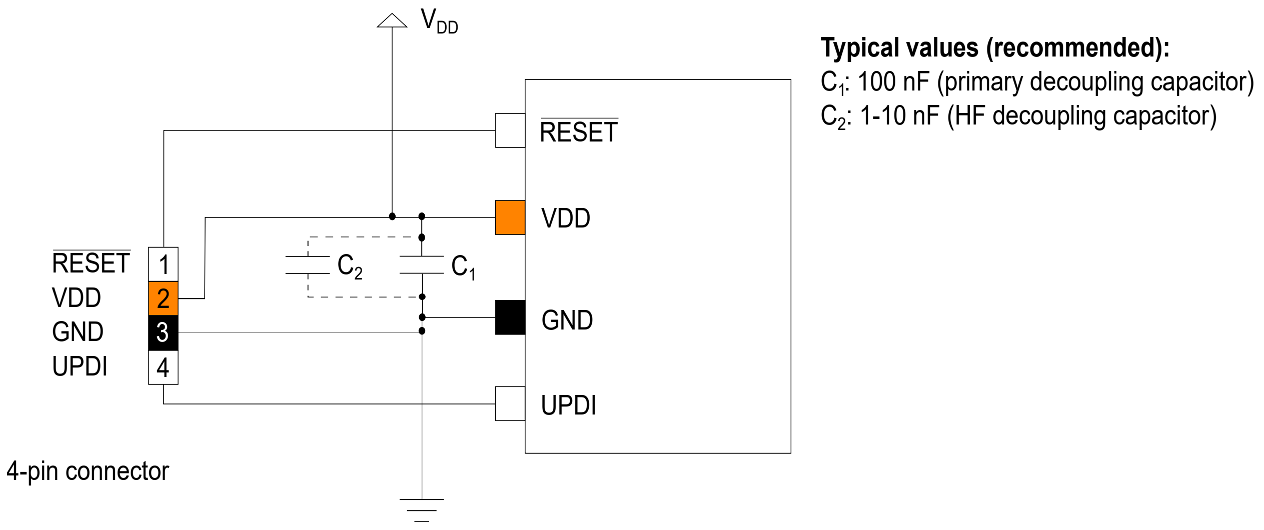
5.4.2 UPDI Connection v2

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 5-4. Recommended UPDI Programming Circuit Schematic


V_{DD}
Typical values (recommended):
C_1: 100 nF (primary decoupling capacitor)
C_2: 1-10 nF (HF decoupling capacitor)
$\overline{\text{RESET}}$
VDD
GND
UPDI
$\overline{\text{RESET}}$ 1
VDD 2
GND 3
UPDI 4
C_2
C_1
4-pin connector

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 some devices, it is not possible 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.

5.5 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*

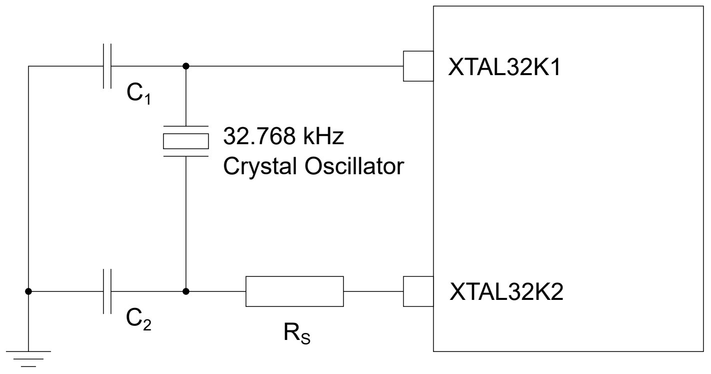
5.5.1 Connection for XTAL32K (External 32.768 kHz Crystal Oscillator)

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:

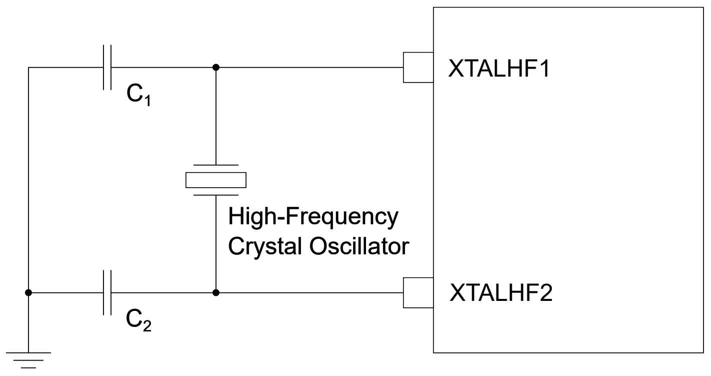
Figure 5-5. Recommended External 32.768 kHz Oscillator Connection Circuit Schematic



5.5.2 Connection for XTALHF (External HF Crystal Oscillator)

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

Figure 5-6. Recommended External High-Frequency Oscillator Connection Circuit Schematic

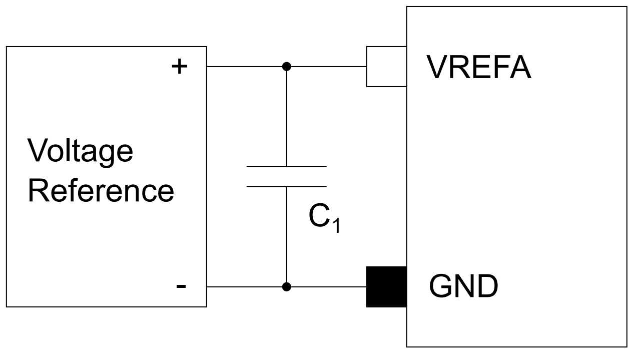


5.6 Connection for External Voltage Reference

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 5-7. Recommended External Voltage Reference Connection



6. Power Supply

6.1 Power Domains

The AVR DU Family devices have several power domains with the following power supply pins:

VDD	Powers I/O lines, XOSCHF and the internal voltage regulator
VUSB	Powers the USB peripheral's internal transceiver (connected to pins DM and DP)

The ground pins, GND, are mutual to VDD and VUSB.

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

6.2 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 adjusts the power consumption when the CPU is active or in a sleep mode. Refer to the *SLPCTRL - Sleep Controller* section for further information.

6.3 Power-Up

The Power-On Reset (POR) and the Brown-out Detector (BOD) will monitor V_{DD} and keep the system 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.

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

6.4 USB Power Supply Configurations

A USB host or hub supplies nominal 5V power on the VBUS wire for use by USB devices that are directly connected. In addition, any USB device may have its own power supply. As stated in the USB 2.0 specification, USB devices that rely solely on power from the USB cable are called *bus-powered* devices. In contrast, those having a separate power source are called *self-powered* devices.

One important design consideration is the maximum allowed slew rate on the device's VDD pin(s). Exceeding this value may cause the device to lock up or even damage the device. Because the slew rate of V_{BUS} can be more than 0.5 V/µs at the moment the USB is connected, it may be necessary to include components in the design to reduce the slew rate on the device's VDD pin(s). These components will be referred to generically as Slew Rate Limiting Components or SRLC. One example of an SRLC is the MIC94165 high-side load switch with reverse blocking and soft start.

The USB peripheral's internal transceiver requires a nominal 3.3V power supply voltage on the VUSB pin for successful data transfer operations on USB, provided by an external source or generated by an internal V_{DD}-to-V_{USB} (3.3V) voltage regulator called USB VREG. For operating the USB VREG, a minimum supply voltage $V_{DD} \geq V_{DDUMIN}$ is required. The USB VREG has these configuration bits:

- The USBVREG bit in the V_{USB} Control (VUSBCTRL) register of the SYSCFG peripheral enables/disables the V_{USB} regulation.
- The USBSINK bit in the System Configuration 1 (SYSCFG1) fuse controls whether the USB VREG can sink current. It is recommended to leave this feature enabled.

Refer to the *Electrical Characteristics* section for standard operating conditions.

The flexibility of external or internally generated V_{USB} allows the following seven power configurations to be supported (with mnemonics in parentheses):

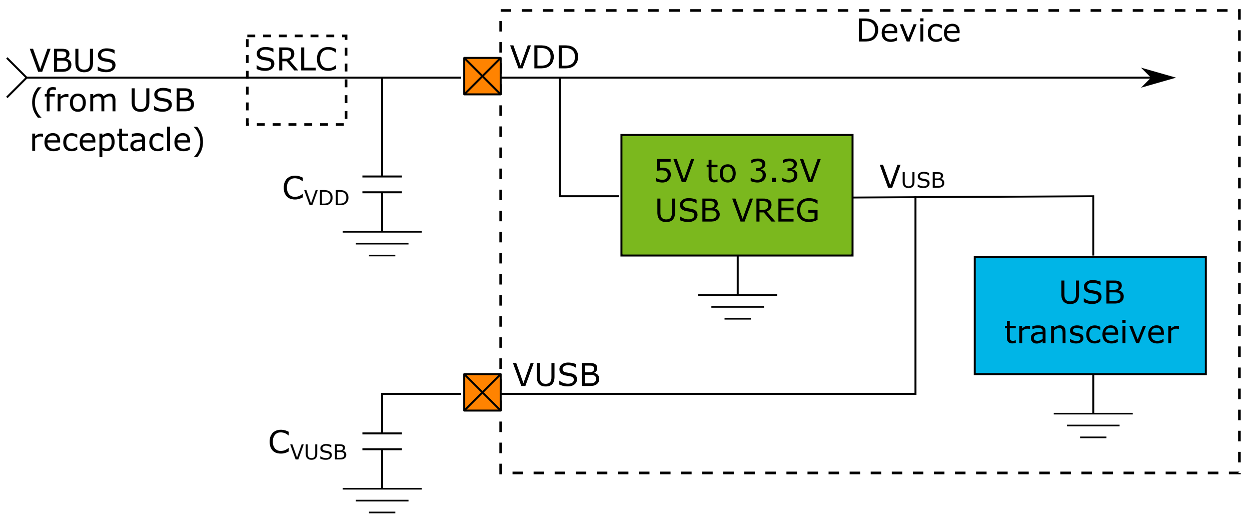
- 5V Bus-powered (**5b**)
- 5V Self-powered (**5s**)
- 3.3V Self-powered (**3s**)

- Dual-powered 5V Bus-dominance and 5V Self-powered (**5b5s**)
- Dual-powered 5V Bus-dominance and 3.3V Self-powered (**5b3s**)
- Dual-powered 5V Bus-dominance and Self-powered by variable 2V to 3V (**5b2s**)
- Dual-powered 5V Self-dominance and 5V Bus-powered (**5s5b**)

5b (5V Bus-powered)

In power configuration **5b**, the VBUS supply from the USB receptacle is connected, either directly or via SRLC, to the device's VDD pin(s). When the USB is not connected, the device is not powered. When USB is connected, the nominal 5V VBUS supply powers the device, and the internal voltage regulator must also be enabled, generating the required 3.3V on the VUSB pin.

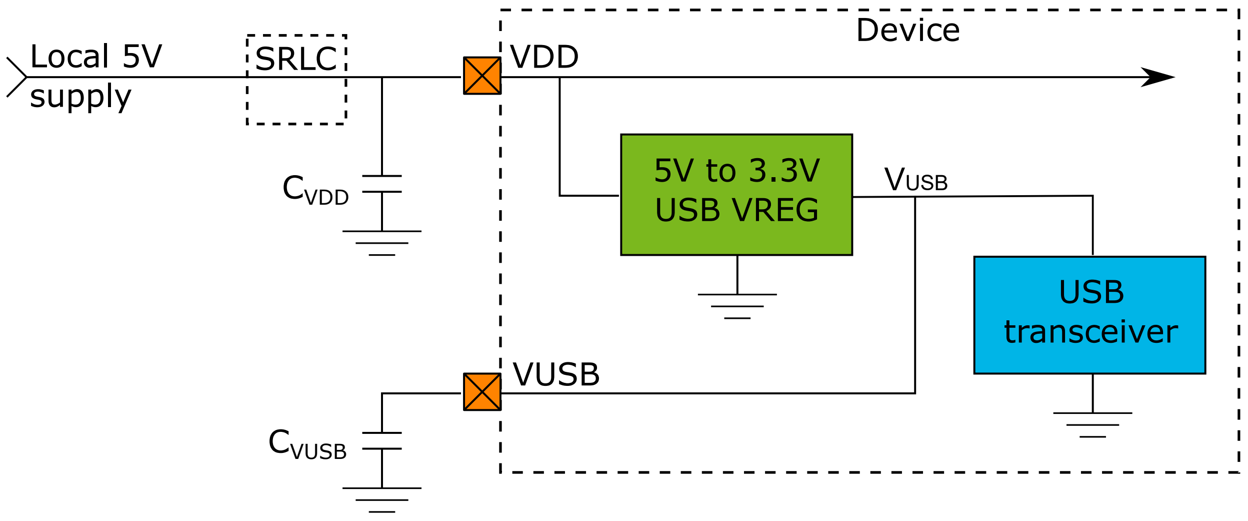
Figure 6-1. Power Configuration 5b



5s (5V Self-powered)

In power configuration **5s**, a local 5V supply is connected, either directly or via SRLC, to the device's VDD pin(s). Also, enable the internal voltage regulator while the USB is connected, generating the required 3.3V on the VUSB pin. The internal voltage regulator may be disabled when the USB is not connected.

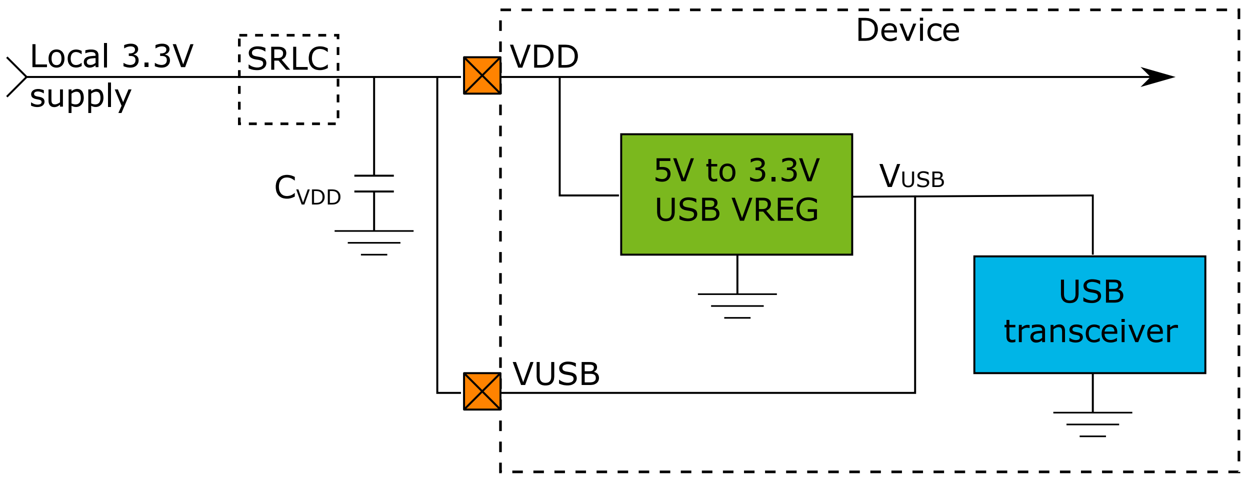
Figure 6-2. Power Configuration 5s



3s (3.3V Self-powered)

In power configuration **3s**, the VDD and VUSB pins are connected, and a local supply provides 3.3V power to them, either directly or via SRLC. The internal voltage regulator is unused, and both the USBVREG bit in the SYSCFG.VUSBCTRL register and the USBSYNC bit in the SYSCFG1 fuse can be written to `0'.

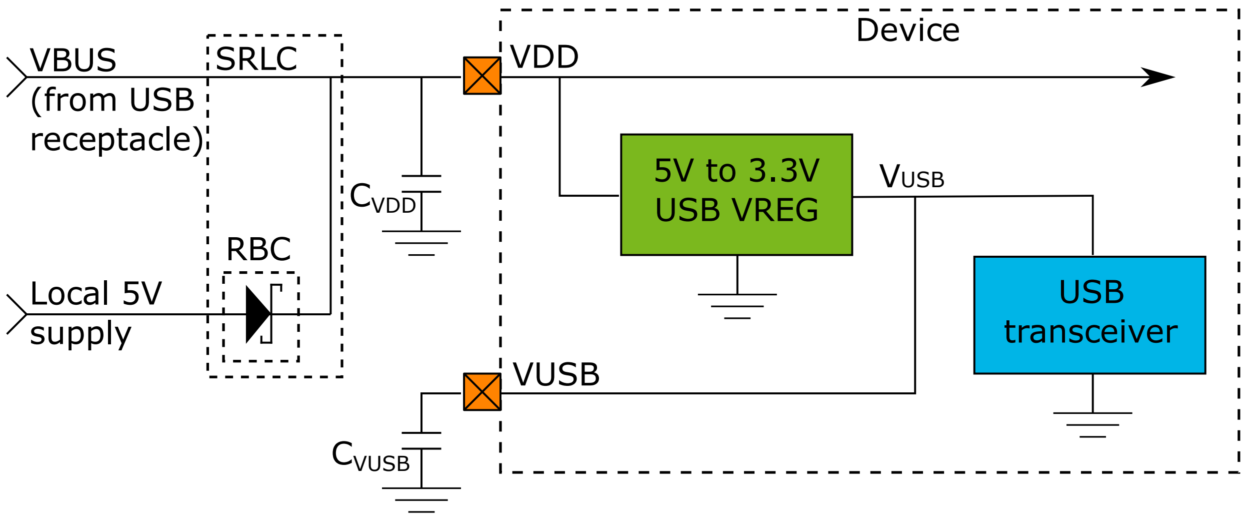
Figure 6-3. Power Configuration 3s



5b5s (Dual-powered 5V Bus-dominance and 5V Self-powered)

In power configuration **5b5s**, the USB VBUS supply is always used to power the device when the USB is connected. VBUS is connected, directly or via SRLC, to the device's VDD pin(s). The internal voltage regulator must also be enabled, thereby generating the required 3.3V on the VUSB pin. When the USB is not connected, typically through reverse-blocking circuitry (RBC) such as a Schottky diode or a transistor configuration, a local 5V supply provides power to the VDD pin(s). The internal voltage regulator may be disabled when the USB is not connected by writing the USBVREG bit in the SYSCFG.VUSBCTRL register to `0'.

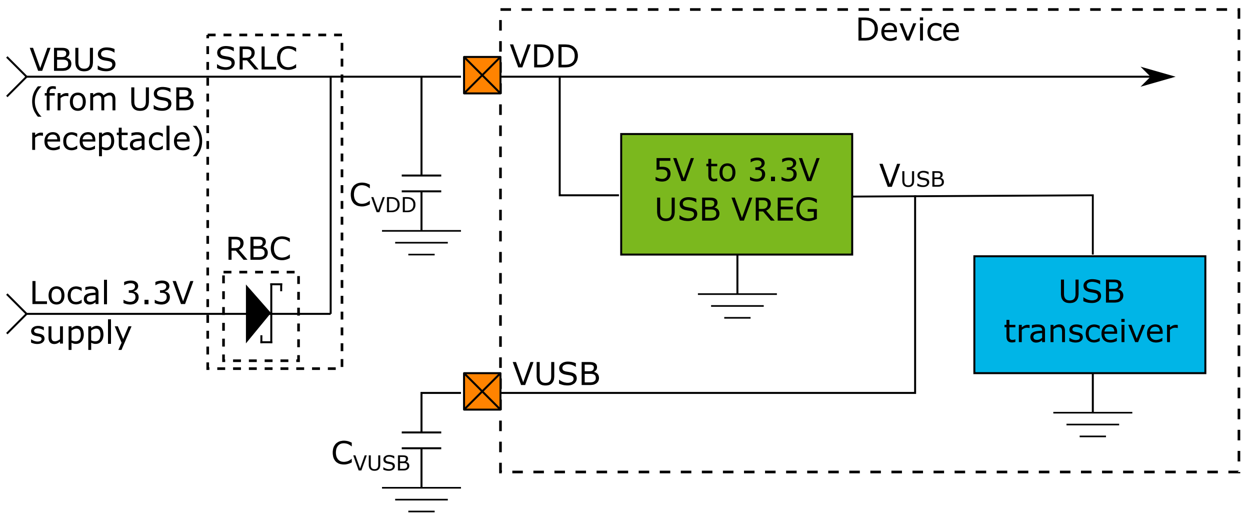
Figure 6-4. Power Configuration 5b5s



5b3s (Dual-powered 5V Bus-dominance and 3.3V Self-powered)

In power configuration **5b3s**, the USB VBUS supply is always used to power the device when the USB is connected. VBUS is connected, directly or via SRLC, to the device's VDD pin(s). The internal voltage regulator must also be enabled, thereby generating the required 3.3V on the VUSB pin. A local 3.3V supply provides power to the VDD pin(s) through RBC when the USB is disconnected. The internal voltage regulator will be inoperable when the USB is not connected because V_{DD} is below V_{DDUMIN}.

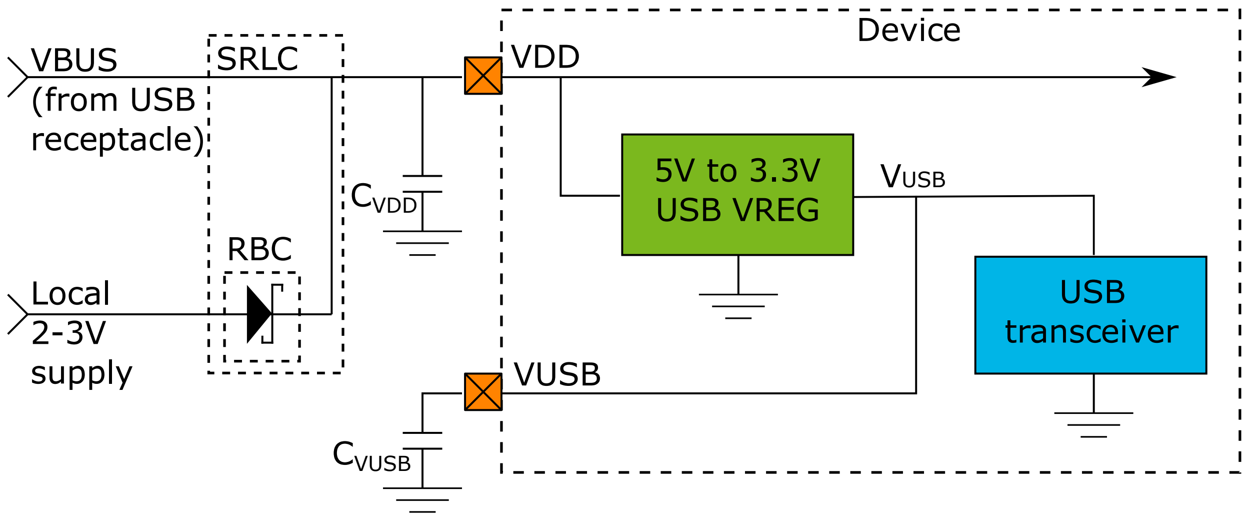
Figure 6-5. Power Configuration 5b3s



5b2s (Dual-powered 5V Bus-dominance and Self-powered by Variable 2V to 3V)

In power configuration **5b2s**, the USB VBUS supply is always used to power the device when the USB is connected. VBUS is connected, directly or via SRLC, to the device's VDD pin(s). The internal voltage regulator must also be enabled, generating the required 3.3V on the VUSB pin. When the USB is not connected, a local supply that can vary from 2V to 3V, such as a battery, provides power to the device’s VDD pin(s), typically through RBC. The internal voltage regulator will be inoperable when the USB is not connected because V_{DD} is below V_{DDUMIN}.

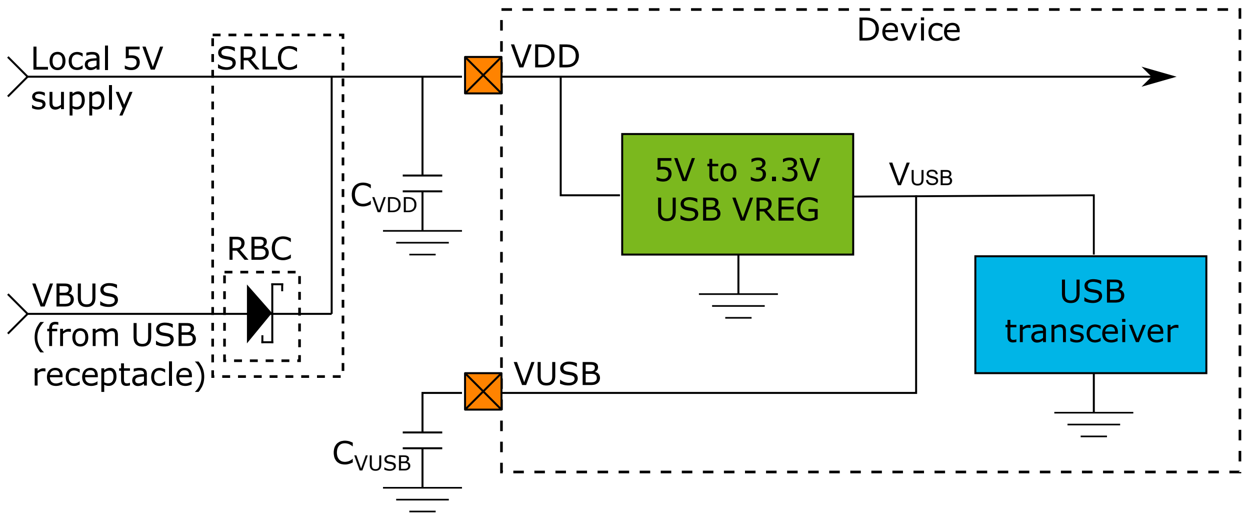
Figure 6-6. Power Configuration 5b2s



5s5b (Dual-powered 5V Self-dominance and 5V Bus-powered)

In power configuration **5s5b**, the device is powered by a local 5V supply whenever available. The local 5V supply is connected, directly or via SRLC, to the device's VDD pin(s). If the local supply fails or is unavailable, the device will be powered only when the USB is connected because VBUS provides power to the device's VDD pin(s), typically through RBC. The internal voltage regulator must be enabled when the USB is connected, generating the required 3.3V on the VUSB pin.

Figure 6-7. Power Configuration 5s5b



7. AVR® CPU

7.1 Features

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

7.2 Overview

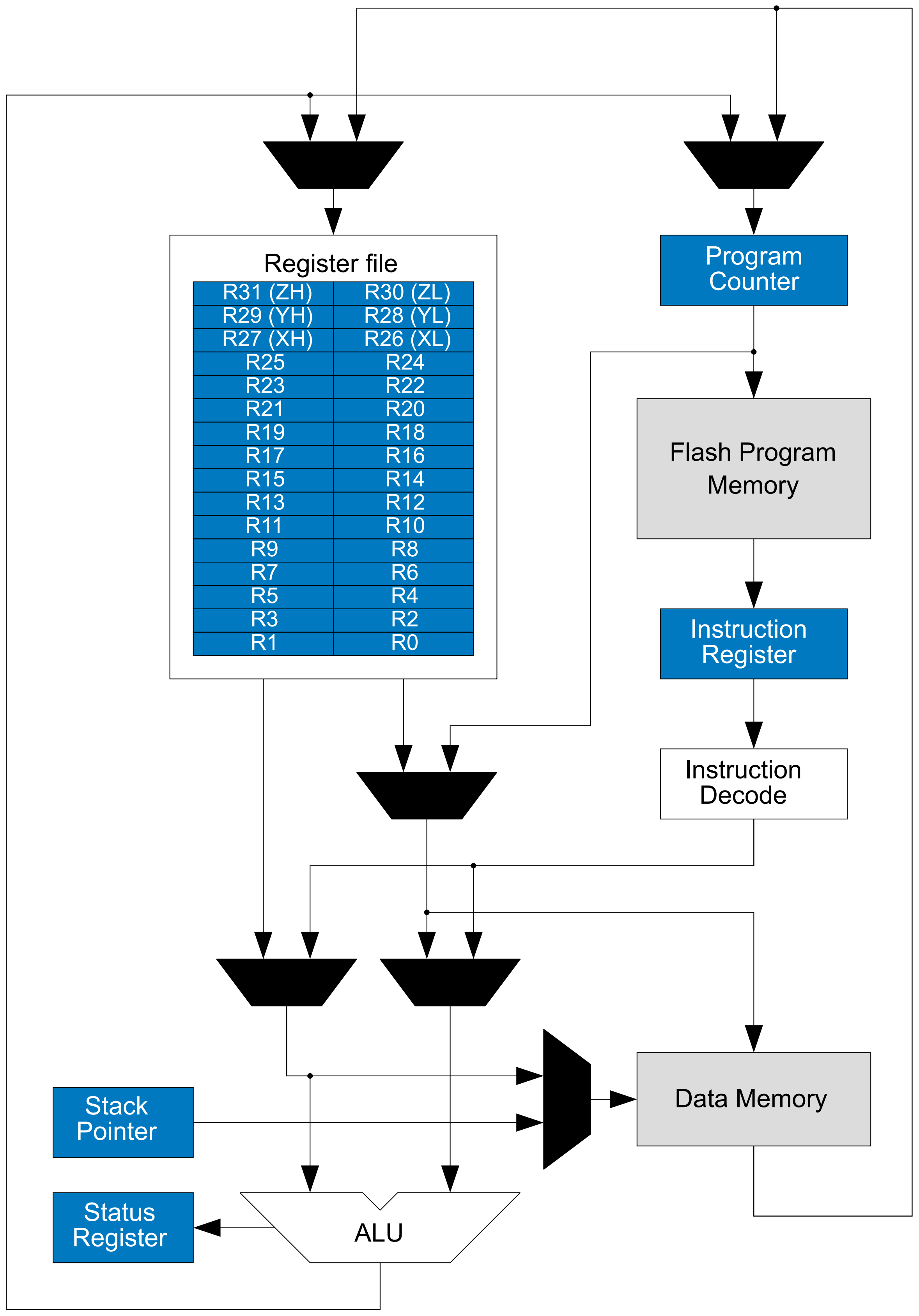
The AVR CPU can access memories, perform calculations, control peripherals, execute instructions from the program memory, and handle interrupts.

7.3 Architecture

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

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

Figure 7-1. AVR® CPU Architecture



7.3.1 Arithmetic Logic Unit (ALU)

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

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

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

7.3.1.1 Hardware Multiplier

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

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

A multiplication takes two CPU clock cycles.

7.4 Functional Description

7.4.1 Program Flow

After being reset, the CPU will execute instructions from the lowest address in the Flash program memory, 0x0000. The Program Counter (PC) addresses the next instruction to be fetched.

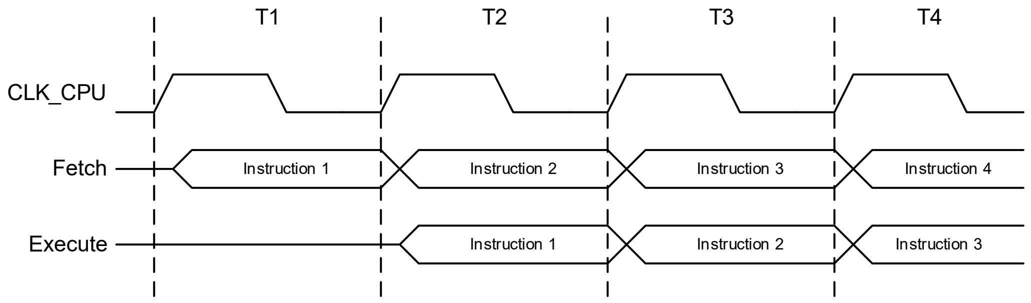
The CPU supports instructions that can change the program flow conditionally or unconditionally and are capable of addressing the whole address space directly. Most AVR instructions use a 16-bit word format, and a limited number use a 32-bit format.

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

7.4.2 Instruction Execution Timing

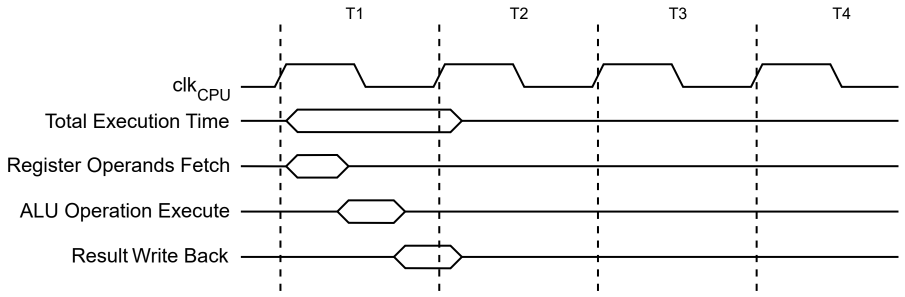
The AVR CPU is clocked by the CPU clock, CLK_CPU. No internal clock division is applied. The figure below shows the parallel instruction fetches and executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept enabling up to 1 MIPS/MHz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

Figure 7-2. The Parallel Instruction Fetches and Executions



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

Figure 7-3. Single Cycle ALU Operation



7.4.3 Status Register

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

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

7.4.4 Stack and Stack Pointer

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

Data are pushed and popped from the stack using the instructions given in Table 7-1, or by executing interrupts. The stack grows from higher to lower memory locations. This means that when pushing data onto the stack, the SP decreases, and when popping data off the stack, the SP increases. The SP is automatically set to the highest address of the internal SRAM after being reset. If the stack is changed, it must be set to point above the SRAM start address (see the *SRAM Data Memory* topic in the *Memories* section for the SRAM start address), and it must be defined before any subroutine calls are executed and before interrupts are enabled. See the table below for SP details.

Table 7-1. Stack Pointer Instructions

Instruction	Stack Pointer	Description
`PUSH`	Decremented by 1	Data are pushed onto the stack
`CALL` `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 subroutine or return from interrupt

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

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

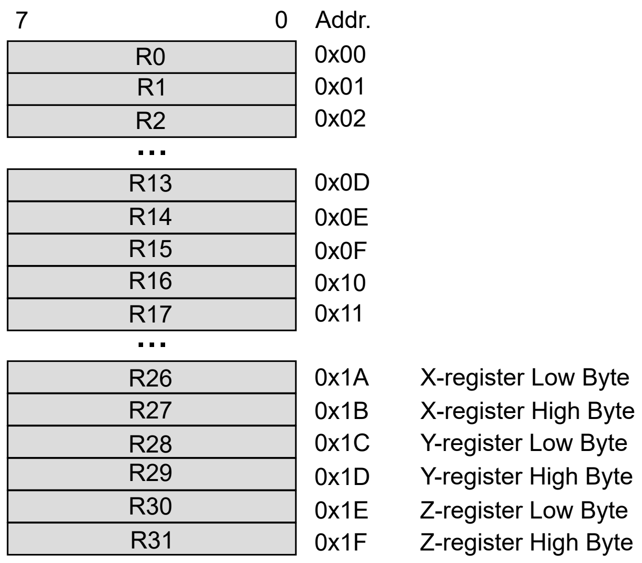
To prevent corruption when updating the SP from software, a write to SPL will automatically disable interrupts for up to four instructions or until the next I/O memory write, whichever comes first.

7.4.5 Register File

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

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

Figure 7-4. AVR® CPU General Purpose Working Registers

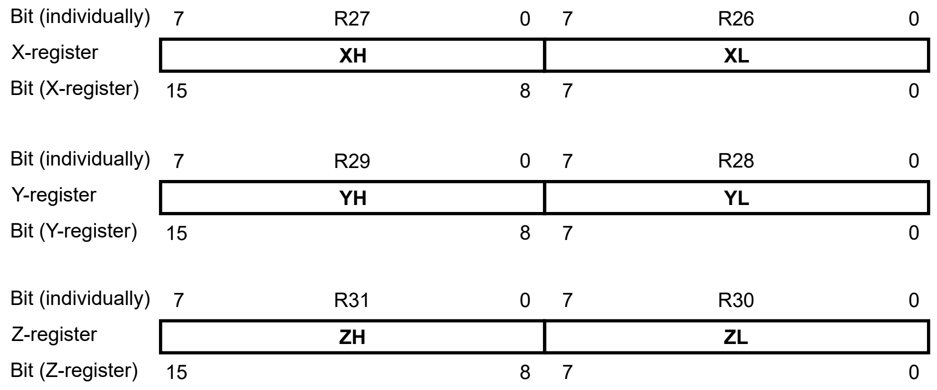


7.4.5.1 The X-, Y-, and Z-Registers

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

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

Figure 7-5. The X-, Y-, and Z-Registers


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

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

7.4.6 Configuration Change Protection (CCP)

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

Changes to the protected I/O registers or bits, or execution of protected instructions, are only possible after the CPU writes a signature to the CCP register. The different signatures are listed in the description of the CCP 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 normal, and the request is kept pending. After the CCP period is completed, any pending interrupts are executed according to their level and priority.

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

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

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

1. The software writes the signature that enables change of protected I/O registers to the CCP bit field in the CPU.CCP register.
2. Within four instructions, the software must write the appropriate data to the protected register. Most protected registers also contain a Write Enable/Change Enable/Lock bit. This bit must be written to '`1`' in the same operation as the data are written.

   The protected change is immediately disabled if the CPU performs write operations to the I/O register or data memory, if load or store accesses to Flash, NVMCTRL, or EEPROM are conducted, or if the `SLEEP` instruction is executed.

7.4.6.2 Sequence for Execution of Self-Programming

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

1. The software temporarily enables self-programming by writing the SPM signature to the CCP register (CPU.CCP).
2. Within four instructions, the software must execute the appropriate instruction. The protected change is immediately disabled if the CPU performs accesses to the Flash, NVMCTRL, or EEPROM, or if the `SLEEP` instruction is executed.

7.4.7 On-Chip Debug Capabilities

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

7.5 Register Summary

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

7.6 Register Description

7.6.1 Configuration Change Protection

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

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

Bits 7:0 – CCP[7:0] Configuration Change Protection

Writing the correct signature to this bit field allows changing protected I/O registers or executing protected instructions within the following four CPU instructions executed.

All interrupts are ignored during these cycles. After completing these cycles, the interrupts will be handled automatically by the CPU. Any pending interrupts will be executed according to their level and priority.

When the protected I/O register signature is written, CCP[0] will read '1' as long as the CCP feature is enabled.

When the protected self-programming signature is written, CCP[1] will read '1' as long as the CCP feature is enabled.

CCP[7:2] will always read '0'.

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

7.6.2 Stack Pointer

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

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

Only the number of bits required to address the available SRAM is implemented for each device. The remaining bits are set, so the Stack Pointer (SP) always points to the SRAM.

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

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

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

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

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

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

7.6.3 Status Register

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

The Status Register contains information about the result of the most recently executed arithmetic or logic instructions. 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 Bit
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 Transfer Bit
The bit copy instructions, Bit Load (`BLD`) and Bit Store (`BST`), use the T bit as source or destination for the operated bit.

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

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

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

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

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

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

8. Memories

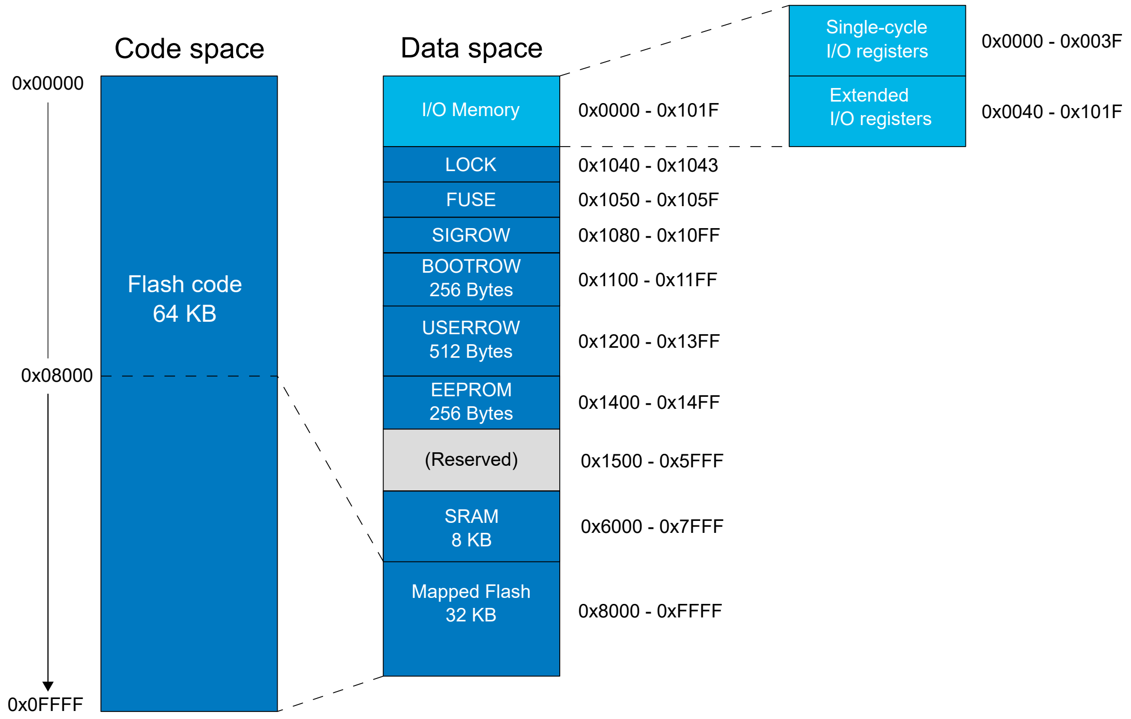
8.1 Overview

The main memories of the AVR64DU32/28 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.

8.2 Memory Map

The figure below shows the memory map for the highest memory derivative in the AVR DU Family. For further details, refer to the subsequent sections and the *Peripheral Address Map* table.

Figure 8-1. Memory Map



8.3 In-System Reprogrammable Flash Program Memory

The AVR64DU32/28 contains 64 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 the following three sections: The Boot Code section, the Application Code section, and the Application Data section. The code in one section may be restricted from writing to addresses in the other sections. The Program Counter can address the whole program memory.

For further details, refer to the Code Size (CODESIZE) and Boot Size (BOOTSIZE) descriptions and the *NVMCTRL - Nonvolatile Memory Controller* section.

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.

Each 32 KB section from 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 8-1. Physical Properties of Flash Memory

Property	AVR64DU28 AVR64DU32
Size	64 KB
Page size	512B
Number of pages	128
NRWW section size (bottom of overall Flash) / RWW section size	8 KB / 56 KB
Start address in data space	0x8000
Start address in code space	0x0

8.4 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 not possible. 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.

8.5 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 8-2. Physical Properties of SRAM Memory

Property	AVR64DU28 AVR64DU32
Size	8 KB
Start address	0x6000
End address	0x7FFF

8.6 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 8-3. Physical Properties of EEPROM Memory

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

8.7 SIGROW - Signature Row

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

All the AVR64DU32/28 devices have a device ID to identify family, Flash size, and pin count. The device ID consists of three DEVICEIDn bytes, as shown in the following table.

Table 8-4. Device ID

Device Name	Signature Byte Address and Value		
	0x00	0x01	0x02
AVR64DU32	`0x1E`	`0x96`	`0x21`
AVR64DU28	`0x1E`	`0x96`	`0x22`

8.7.1 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]							

8.7.2 Signature Row Description

8.7.2.1 Device ID

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

Each device has an ID identifying the device and its properties, such as memory sizes and pin count, and 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

8.7.2.2 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 on how to use the value stored in this bit field.

8.7.2.3 Serial Number Byte n

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

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

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

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

8.8 BOOTROW - Boot Row

The AVR64DU32/28 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.

8.9 USERROW - User Row

The AVR64DU32/28 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 device 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 an ordinary Flash. Refer to the *System Memory Address Map* for further details.

8.10 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 UPDI but can only be programmed or cleared by the UPDI. The configuration values stored in the fuses are written to their respective target registers at the end of the start-up sequence.

The fuses for peripheral configuration (FUSE) are preprogrammed, but the user can alter them. Altered values in the configuration fuse will be applicable only after a Reset.

Note: All reserved bits must be written to ‘`0`’ when writing the fuses.

8.10.1 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	OSCCFG	7:0					CLKSEL[3:0]			
0x03 … 0x04	Reserved									
0x05	SYSCFG0	7:0	CRCSRC[1:0]		CRCSEL	UPDIPINCFG	RSTPINCFG		BROWSAVE	EESAVE
0x06	SYSCFG1	7:0					USBSINK	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]							

8.10.2 Fuse Description

8.10.2.1 Watchdog Timer Configuration

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

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

8.10.2.2 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 Alternative 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
0x2	SAMPLE	BOD enabled in Sampled mode
0x3	-	Reserved

8.10.2.3 Oscillator Configuration

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

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

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

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

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

8.10.2.4 System Configuration 0

Name:	SYSCFG0
Offset:	0x05
Default:	0xD0
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
	CRCSRC[1:0]		CRCSEL	UPDIPINCFG	RSTPINCFG		BROWSAVE	EESAVE
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Default	1	1	0	1	0		0	0

Bits 7:6 – CRCSRC[1:0] CRC Source
This bit field controls which Flash section 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
0x0	FLASH	CRC of full Flash (boot, application code, and application data)
0x1	BOOT	CRC of the Boot section
0x2	BOOTAPP	CRC of the Application code and Boot sections
0x3	NOCRC	No CRC

Bit 5 – CRCSEL CRC Mode Selection
This bit controls the type of CRC performed by the CRCSCAN peripheral. Refer to the *CRCSCAN - 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 4 – UPDIPINCFG Configuration of UPDI Pin at Start-Up
This bit controls the UPDI pin configuration.

Value	Name	Description
0	GPIO	UPDI pin is configured as GPIO
1	UPDI	The UPDI pin is configured as a UPDI pin with pull-up enabled on PF7. This is the factory default value.

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

Value	Name	Description
0	INPUT	No external Reset. It can be used as a GPIO input pin. This is the factory default value.
1	$\overline{\text{RESET}}$	External Reset with pull-up enabled on PF6

Bit 1 – BROWSAVE Boot Row Save During Chip Erase
This bit controls if the Boot Row will be erased or not during a chip erase. If the device is locked, the Boot Row is erased by a chip erase regardless of this bit.

Value	Name	Description
0	DISABLE	The Boot Row is erased by a chip erase
1	ENABLE	The Boot Row is not erased by a chip erase

Bit 0 – EESAVE EEPROM Save During Chip Erase
This bit controls if the EEPROM will be erased or not during a chip erase.

Value	Name	Description
0	DISABLE	The EEPROM is erased by a chip erase
1	ENABLE	The EEPROM is not erased by chip erase, regardless of whether the device is locked or not

8.10.2.5 System Configuration 1

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

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

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

Bit 3 – USBSINK USB Voltage Regulator Current Sink Enable
This bit controls whether the USB Voltage Regulator can sink current.

Value	Name	Description
`0x0`	DISABLE	USBVREG can not sink current
`0x1`	ENABLE	USBVREG can sink current

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

8.10.2.6 Code Size

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

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

8.10.2.7 Boot Size

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

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

Note: If FUSE.BOOTSIZE and FUSE.CODESIZE are `0x00`, the entire Flash is the Boot Code section.

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 a Boot Code section.
For more details, refer to the *NVMCTRL - Nonvolatile Memory Controller* section.

8.10.2.8 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	NVM access through UPDI is disabled. Program and Debug Interface Disable (PDID): All erase and write commands to NVM must be executed from the Boot Code section (bootloader). Chip Erase and User Row write access is blocked. The CRC status is available.
`0x3`	BASIC	The UPDI peripheral and the UPDI pin are working as described in the UPDI section

Important: There is no way to recover from this mode, so once NVMACCDIS is enabled, it is not possible to re-enable UPDI access for device programming. Ensure that the consequences of enabling this feature are fully comprehended before enabling this mode.

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

8.11 LOCK - Memory Sections Access Protection

The device can be locked, making the memories unreadable 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, preventing the reading of application data or code using the debugger interface. Regular memory access from within the application is still enabled.

Writing a nonvalid key to the Lock Key (LOCK.KEY) register locks the device.

Table 8-5. Memory Access Unlocked (FUSE.LOCK 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[3]	Yes	Yes	Yes	Yes
FUSES	Yes	No	Yes	Yes
I/O Memory	Yes	Yes	Yes	Yes

Table 8-6. Memory Access Locked (FUSE.LOCK 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[3]	Yes	Yes	No	No
FUSES	Yes	No	No	No
I/O Memory	Yes	Yes	No	No

Notes:

1. Read operations marked No in the tables may appear successful, but the data is invalid. Hence, any attempt of code validation through the UPDI will fail on these memory sections.
2. Using the Fuse Write command in the Locked mode can write the USERROW, but it is impossible to read out the current USERROW values.
3. The BOOTROW can be read and written only from the BOOT section.

Important: The only way to unlock a device is a CHIPERASE, which won't erase the USERROW and not retain any application data.

8.11.1 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]							

8.11.2 Lock Description

8.11.2.1 Lock Key Fuse

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

Bit	31	30	29	28	27	26	25	24
	KEY[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	23	22	21	20	19	18	17	16
	KEY[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	15	14	13	12	11	10	9	8
	KEY[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

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

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

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

8.12 I/O Memory

All AVR64DU32/28 devices' I/O and peripheral registers are in the I/O memory space. Refer to the *Peripheral Address Map* table for further details.

For compatibility with future devices, if writing a register containing reserved bits, the reserved bits have to be written to '`0`'. Never write the reserved I/O memory addresses.

8.12.1 Single-Cycle I/O Registers

A single-cycle CPU instruction using the `IN` or `OUT` instruction can access the I/O memory ranging from `0x00` to `0x3F`.

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

- VPORTx
  - Refer to the *PORT* 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, the `SBIS` or `SBIC` instruction can check the single bits.

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

8.12.2 Extended I/O Registers

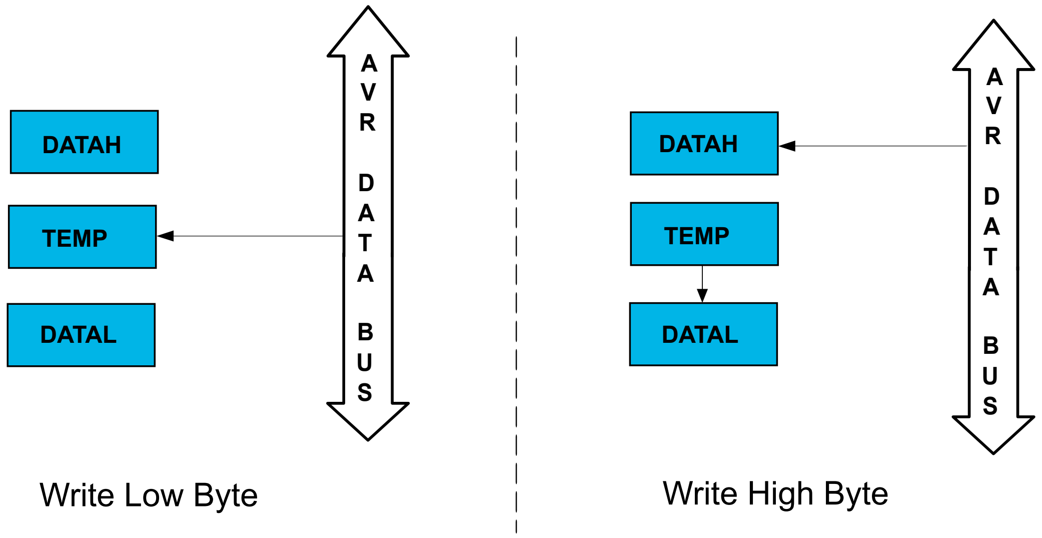
The I/O memory space ranging from `0x0040` to `0x103F` can only be accessed by the `LD/LDS/LDD` or `ST/STS/STD` instructions, transferring data between the 32 general purpose working registers (R0-R31) and the I/O memory space.

For further details, refer to the *Peripheral Address Map* table and the *Instruction Set Summary* section.

8.12.3 Accessing 16-Bit Registers

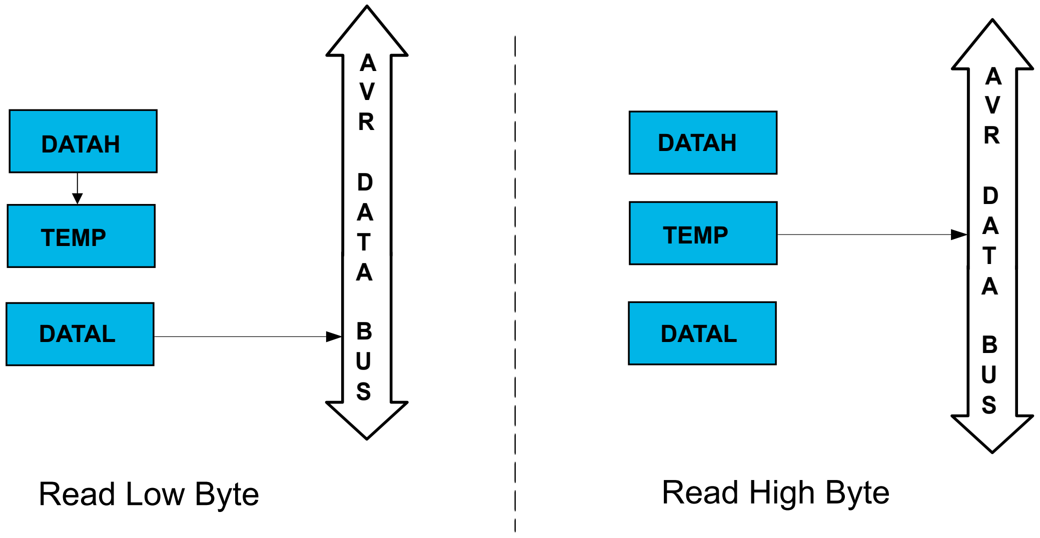
Most of the registers for the AVR64DU32/28 devices are 8-bit registers, but the devices also feature a few 16-bit registers. As the AVR data bus has a width of eight bits, accessing the 16-bit requires two read or write operations. All the 16-bit registers of the AVR64DU32/28 devices are connected to the 8-bit bus through a temporary (TEMP) register.

Figure 8-2. 16-Bit Register Write Operation



For a 16-bit write operation, the low byte register (e.g., DATAL) of the 16-bit register must be written before the high byte register (e.g., DATAH). Writing the low byte register will result in a write to the temporary register instead of the low byte register, as shown on the left side of Figure 8-2. 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 Figure 8-2.

Figure 8-3. 16-Bit Register Read Operation



For a 16-bit read operation, the low byte register (e.g., DATAL) of the 16-bit register must be read before the high byte register (e.g., DATAH). When the low byte register is read, the high byte register of the 16-bit register is copied into the temporary register in the same clock cycle, as shown on the left side of Figure 8-3. 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 Figure 8-3.

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.

8.12.4 Accessing 24- and 32-Bit Registers

For 24- and 32-bit registers, the read and write access is done the same way as described for 16-bit registers, except there are two temporary registers for 24-bit registers and three temporary registers for 32-bit registers. Write the Most Significant Byte (MSB) last when writing to the register. Read the Least Significant Byte (LSB) first when reading the register.

9. Peripherals and Architecture

9.1 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 9-1. Peripheral Address Map

Base Address	Name	Description	14-Pin	20-Pin	28-Pin	32-Pin
0x0000	VPORTA	Virtual Port A	X	X	X	X
0x0008	VPORTC	Virtual Port C	X	X	X	X
0x000C	VPORTD	Virtual Port D	X	X	X	X
0x0014	VPORTF	Virtual Port F	X	X	X	X
0x001C	GPR	General Purpose Registers	X	X	X	X
0x0030	CPU	CPU	X	X	X	X
0x0040	RSTCTRL	Reset Controller	X	X	X	X
0x0050	SLPCTRL	Sleep Controller	X	X	X	X
0x0060	CLKCTRL	Clock Controller	X	X	X	X
0x00A0	BOD	Brown-out Detector	X	X	X	X
0x00B0	VREF	Voltage Reference	X	X	X	X
0x0100	WDT	Watchdog Timer	X	X	X	X
0x0110	CPUINT	Interrupt Controller	X	X	X	X
0x0120	CRCSCAN	Cyclic Redundancy Check Memory Scan	X	X	X	X
0x0140	RTC	Real-Time Counter	X	X	X	X
0x01C0	CCL	Configurable Custom Logic	X	X	X	X
0x0200	EVSYS	Event System	X	X	X	X
0x0400	PORTA	Port A Configuration	X	X	X	X
0x0440	PORTC	Port C Configuration	X	X	X	X
0x0460	PORTD	Port D Configuration	X	X	X	X
0x04A0	PORTF	Port F Configuration	X	X	X	X
0x05E0	PORTMUX	Port Multiplexer	X	X	X	X
0x0600	ADC0	Analog-to-Digital Converter 0	X	X	X	X
0x0680	AC0	Analog Comparator 0	X	X	X	X
0x0800	USART0	Universal Synchronous Asynchronous Receiver Transmitter 0	X	X	X	X
0x0820	USART1	Universal Synchronous Asynchronous Receiver Transmitter 1	X	X	X	X
0x0900	TWI0	Two-Wire Interface 0	X	X	X	X
0x0940	SPI0	Serial Peripheral Interface 0	X	X	X	X
0x0A00	TCA0	Timer/Counter Type A instance 0	X	X	X	X
0x0B00	TCB0	Timer/Counter Type B instance 0	X	X	X	X
0x0B10	TCB1	Timer/Counter Type B instance 1	X	X	X	X
0x0C00	USB0	Universal Serial Bus instance 0	X	X	X	X
0x0F00	SYSCFG	System Configuration	X	X	X	X
0x1000	NVMCTRL	Nonvolatile Memory Controller	X	X	X	X

Table 9-2. System Memory Address Map

Base Address	Name	Description	14-Pin	20-Pin	28-Pin	32-Pin
0x1040	LOCK	Lock Bits	X	X	X	X
0x1050	FUSE	User Configuration	X	X	X	X
0x1080	SIGROW	Signature Row	X	X	X	X
0x1100	BOOTROW	Boot Row	X	X	X	X
0x1200	USERROW	User Row	X	X	X	X

9.2 Interrupt Vector Mapping

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

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

By writing to the corresponding Interrupt Enable bit in the peripheral's Interrupt Control register (peripheral.INTCTRL), an interrupt is enabled or disabled..

When the corresponding interrupt is enabled, an interrupt request is generated, and the interrupt flag is set. Interrupts must be enabled globally to generate interrupt requests. The interrupt request remains active until clearing the interrupt flag. See the peripheral's INTFLAGS register for details on clearing interrupt flags.

Table 9-3. Interrupt Vector Mapping

Vector Number	Program Address (word)	Peripheral Source	Description	28-Pin	32-Pin
0	0x00	RESET		X	X
1	0x02	NMI	Non-Maskable Interrupt available for: • CRCSCAN • CLKCTRL	X	X
2	0x04	BOD_VLM	Voltage Level Monitor Interrupt	X	X
3	0x06	CLKCTRL_CFD	External crystal oscillator or clock source failure	X	X
4	0x08	RTC_RTC	Real-Time Counter Overflow or Compare Match Interrupt	X	X
5	0x0A	RTC_PIT	Real-Time Counter Periodic Interrupt	X	X
6	0x0C	CCL_CCL	Configurable Custom Logic Interrupt	X	X
7	0X0E	USB_BUS	USB Bus Event Interrupt	X	X
8	0x10	USB_TAC	USB Transaction Complete Interrupt	X	X
9	0x12	PORTA_PORT	PORTA External Interrupt	X	X
10	0x14	TCA0 _OVF TCA0_LUNF	Normal: Timer/Counter Type A Overflow Interrupt Split: Timer/Counter Type A Low Underflow Interrupt	X	X
11	0x16	TCA0_HUNF	Normal: Unused Split: Timer/Counter Type A High Underflow Interrupt	X	X
12	0x18	TCA0_CMP0 TCA0_LCMP0	Normal: Timer/Counter Type A Compare 0 Interrupt Split: Timer/Counter Type A Low Compare 0 Interrupt	X	X
13	0x1A	TCA0_CMP1 TCA0_LCMP1	Normal: Timer/Counter Type A Compare 1 Interrupt Split: Timer/Counter Type A Low Compare 1 Interrupt	X	X
14	0x1C	TCA0_CMP2 TCA0_LCMP2	Normal: Timer/Counter Type A Compare 2 Interrupt Split: Timer/Counter Type A Low Compare 2 Interrupt	X	X

...........continued

Vector Number	Program Address (word)	Peripheral Source	Description	28-Pin	32-Pin
15	0x1E	TCB0_INT	Timer Counter Type B Capture/Overflow Interrupt	X	X
16	0x20	TWI0_TWIS	Two-Wire Interface Client Interrupt	X	X
17	0x22	TWI0_TWIM	Two-Wire Interface Host Interrupt	X	X
18	0x24	SPI0_INT	Serial Peripheral Interface Interrupt	X	X
19	0x26	USART0_RXC	Universal Synchronous Asynchronous Receiver and Transmitter Receive Complete Interrupt	X	X
20	0x28	USART0_DRE	Universal Synchronous Asynchronous Receiver and Transmitter Data Register Empty Interrupt	X	X
21	0x2A	USART0_TXC	Universal Synchronous Asynchronous Receiver and Transmitter Transmit Complete Interrupt	X	X
22	0x2C	PORTD_PORT	PORTD External Interrupt	X	X
23	0x2E	PORTC_PORT	PORTC External Interrupt	X	X
24	0x30	PORTF_PORT	PORTF External Interrupt	X	X
25	0x32	NVMCTRL_NVMREADY	Nonvolatile Memory Controller EEPROM/Flash Ready Interrupt	X	X
26	0x34	USART1_RXC	Universal Synchronous Asynchronous Receiver and Transmitter Receive Complete Interrupt	X	X
27	0x36	USART1_DRE	Universal Synchronous Asynchronous Receiver and Transmitter Data Register Empty Interrupt	X	X
28	0x38	USART1_TXC	Universal Synchronous Asynchronous Receiver and Transmitter Transmit Complete Interrupt	X	X
29	0x3A	TCB1_INT	Timer Counter Type B Capture/Overflow Interrupt	X	X
30	0x3C	AC0_AC	Analog Comparator Interrupt	X	X
31	0x3E	ADC0_ERROR	Analog-to-Digital Converter Error Interrupt	X	X
32	0x40	ADC0_RESRDY	Analog-to-Digital Converter Result Ready Interrupt	X	X
33	0x42	ADC0_SMPRDY	Analog-to-Digital Converter Sample Ready Interrupt	X	X

9.3 SYSCFG - System Configuration

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

9.3.1 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]			
0x02 ... 0x05	Reserved									
0x06	VUSBCTRL	7:0								USBVREG

9.3.2 Register Description

9.3.2.1 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.

9.3.2.2 USB Voltage System Control

Name:	VUSBCTRL
Offset:	0x06
Reset:	0x00
Property:	-

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

Bit 0 – USBVREG USB Voltage Regulator

This bit controls the status of the USB voltage regulator. See the description of the USB peripheral, *USB - Universal Serial Bus Device Controller*, for more details.

Value	Name	Description
0	DISABLE	The USB voltage regulator is disabled
1	ENABLE	The USB voltage regulator is enabled

10. GPR - General Purpose Registers

The AVR64DU32/28 devices provide four General Purpose registers that can be used for storing any information. They are applicable for storing global variables and interrupt flags. General Purpose registers, residing in the address range 0x1C-0x1F, are directly bit-accessible using the `SBI`, `CBI`, `SBIS`, and `SBIC` instructions.

10.1 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]							

10.2 Register Description

10.2.1 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

11. NVMCTRL - Nonvolatile Memory Controller

11.1 Features

- Unified Memory
- In-System Programmable
- True Read-While-Write Support
- Self-Programming and Boot Loader Support
- Configurable Sections for Write Protection:
  - Boot section for boot loader 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 from software
  - Can be written from UPDI on a locked device
  - Content is kept after chip erase
- Boot Row for Boot Data:
  - For storage of device specific data
  - Accessible from the BOOT section only

11.2 Overview

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

11.2.1 Block Diagram

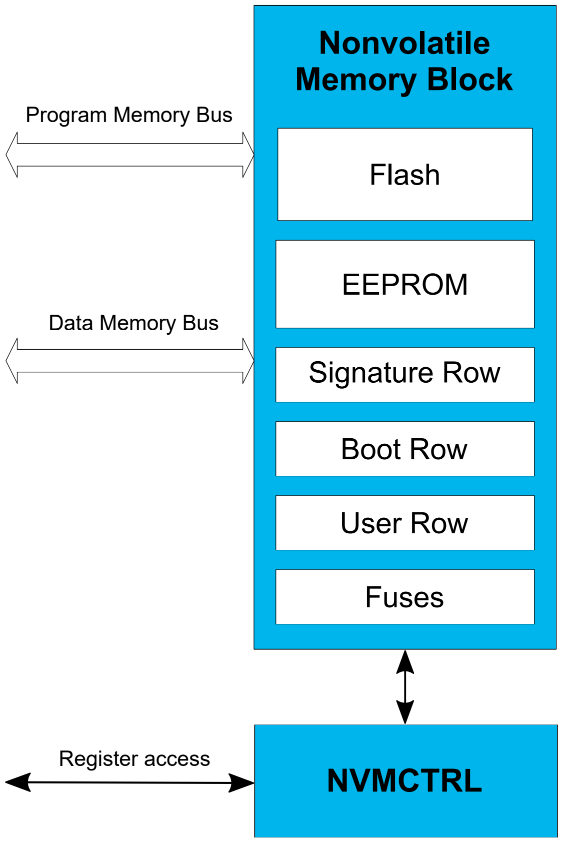


Figure 11-1. NVMCTRL Block Diagram

11.3 Functional Description

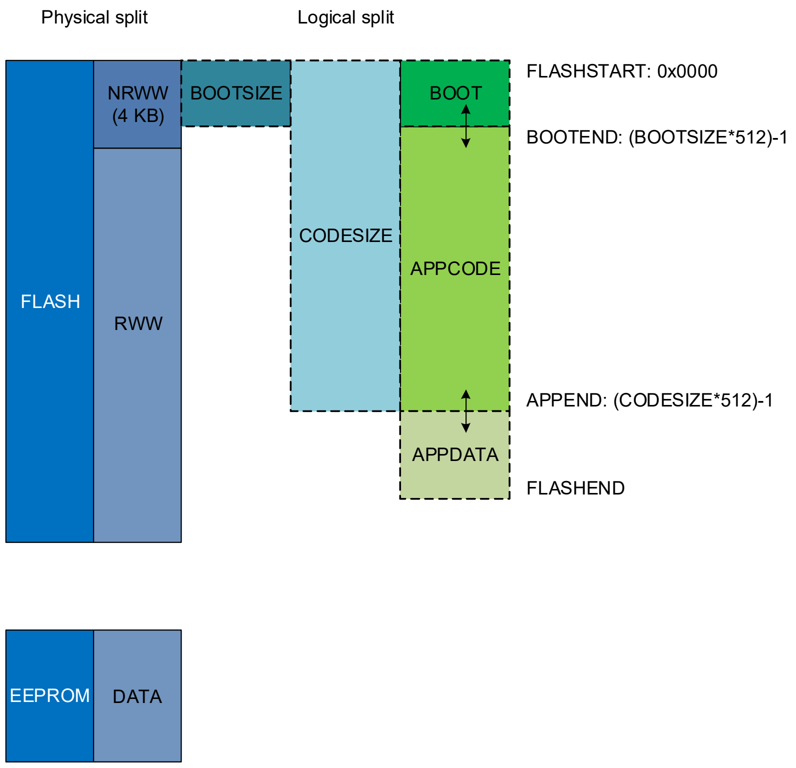
11.3.1 Memory Organization

11.3.1.1 Flash

The Flash is divided into a set of pages. A page is the smallest addressable unit when erasing the Flash. It is only possible to write or erase an entire page at a time. One page consists of 512 bytes.

Independent of page size, the Flash is organized in sections. The Flash is split into two physical sections to optimize the structure for boot loader applications: Read-While-Write (RWW) and No-Read-While-Write (NRWW). It is possible to divide these two sections into three logical sections: Boot Loader Code (BOOT), Application Code (APPCODE), and Application Data (APPDATA).

Figure 11-2. AVR DU Flash Memory Map



11.3.1.1.1 Physical Sections

The Flash is divided physically into two fixed sections: A **Read-While-Write (RWW)** section and a **No-Read-While-Write (NRWW)** section.

The main differences between the two sections are as follows:

- When erasing or writing a page located inside the RWW Flash, the NRWW Flash can be read during the operation
- When erasing or writing a page located inside the NRWW Flash, the CPU is halted during the entire operation

The syntax “Read-While-Write section” refers to which section to program (erase or write), not the one read. Only the code located inside the NRWW Flash is accessible by either executing a CPU instruction or reading data while the RWW Flash is being written or erased.

Note: The interrupt code in the RWW section may halt the CPU if the associated interrupt is triggered while the RWW section is erased or written. Disable or place the interrupt code in the Boot Code (BOOT) section to avoid this.

The figures and table below explain these two physical Flash sections in detail:

Figure 11-3. Read-While-Write Scenarios

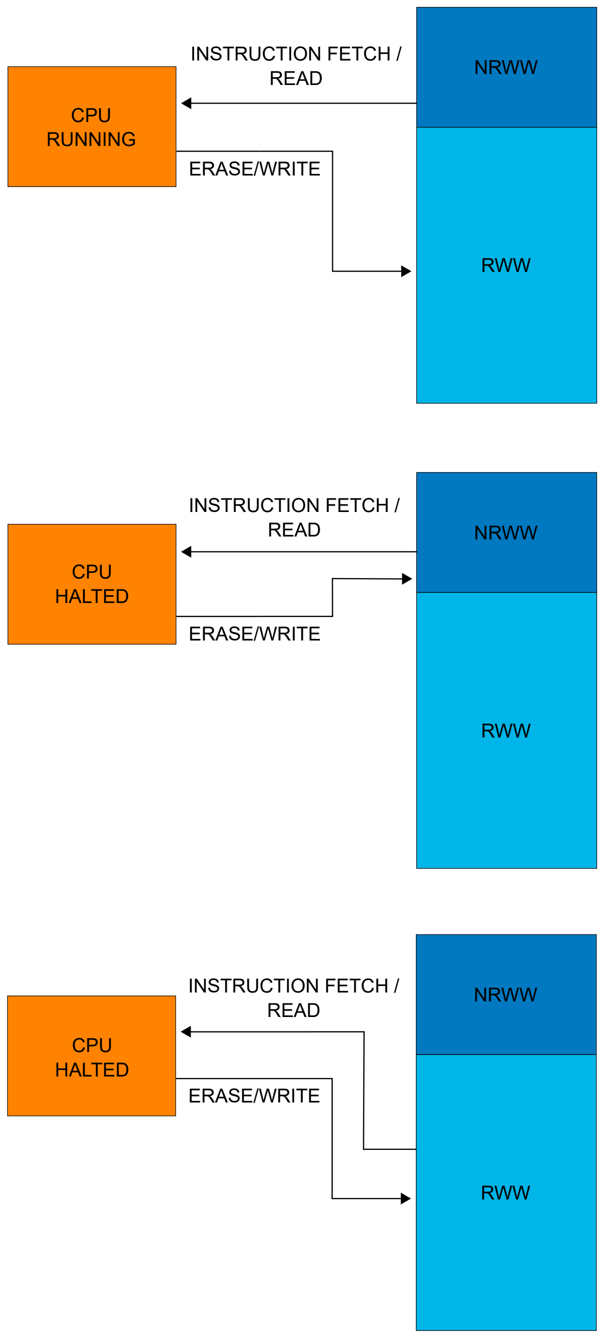


Table 11-1. Read-While-Write Scenarios

Flash Section Erased/Written	Flash Section Accessed	CPU
RWW section	NRWW section	Running
NRWW section	NRWW section	Halted
RWW section	RWW section	Halted

Notes:

- The User Row is located in the RWW Flash. When erasing or writing a page in the User Row, the NRWW Flash can be read during the operation.
- The Boot Row is located in the RWW Flash. When erasing or writing a page in the Boot Row, the NRWW Flash can be read during the operation.
- The physical section sizes are device-dependent. Refer to the *Memory Overview* section for further details.

11.3.1.1.2 Logical Sections

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

- **Boot Code (BOOT)** - The code executed from the BOOT area has write access to all other Flash sections. Boot loader software must be placed in this section if used.
- **Application Code (APPCODE)** - The code executed from the APPCODE area has limited write access to other Flash sections. Executable application code is typically placed in this section.
- **Application Data (APPDATA)** - Flash section without write access. Parameters are usually placed in this section.

Note: For detailed inter-section read/write access, refer to the *Flash Access Protection* section.

Section Sizes

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

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 until the end of the Flash.

Table 11-2. Setting Up Flash Sections

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

Using the BOOT section for the application code is recommended if there is no boot loader software.

Notes:

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

Example 11-1. Size of Flash Sections Example

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

11.3.1.1.3 Flash Access Protections

Inter-Section Write Protection

For security reasons, it is impossible to write to the section of Flash from which the code is currently executing. Code writing to the APPCODE section must execute from the BOOT section, and code writing to the APPDATA section must execute from either the BOOT section or the APPCODE section.

Table 11-3. Write Protection for Self-Programming

Program Execution Section	Section Being Addressed	Programming Allowed?
BOOT	BOOT	No
	APPCODE	Yes
	APPDATA	
	EEPROM	
	USERROW	
	BOOTROW	Yes
APPCODE	BOOT	No
	APPCODE	
	APPDATA	Yes
	EEPROM	
	USERROW	
	BOOTROW	No
APPDATA	BOOT	No
	APPCODE	
	APPDATA	
	EEPROM	
	USERROW	
	BOOTROW	No

Flash Read/Write 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 an effect only when leaving the BOOT section.

The three write protection bits (EEWP, APPDATAWP and APPCODEWP) in the Control B (NVMCTRL.CTRLB) register can be set to prevent writes respectively to the EEPROM or the APPDATA or APPCODE sections.

11.3.1.2 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.

11.3.1.3 Signature Row

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

11.3.1.4 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.

11.3.1.5 Boot Row

The Boot Row is 256 bytes of Flash. This section is only accessible from the BOOT section and can be used to store various data, such as keys. This page is optionally erased by a chip erase.

The Boot Row is erased/written as an ordinary Flash and located in the RWW section of the Flash. It cannot be written when the Boot Row Write Protection (BOOTROWWP) bit in the Control C (NVMCTRL.CTRLC) register is set to '`1`'.

11.3.1.6 Fuses

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

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

11.3.2 Memory Access

For read/write operations, the Flash memory can be accessed 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.

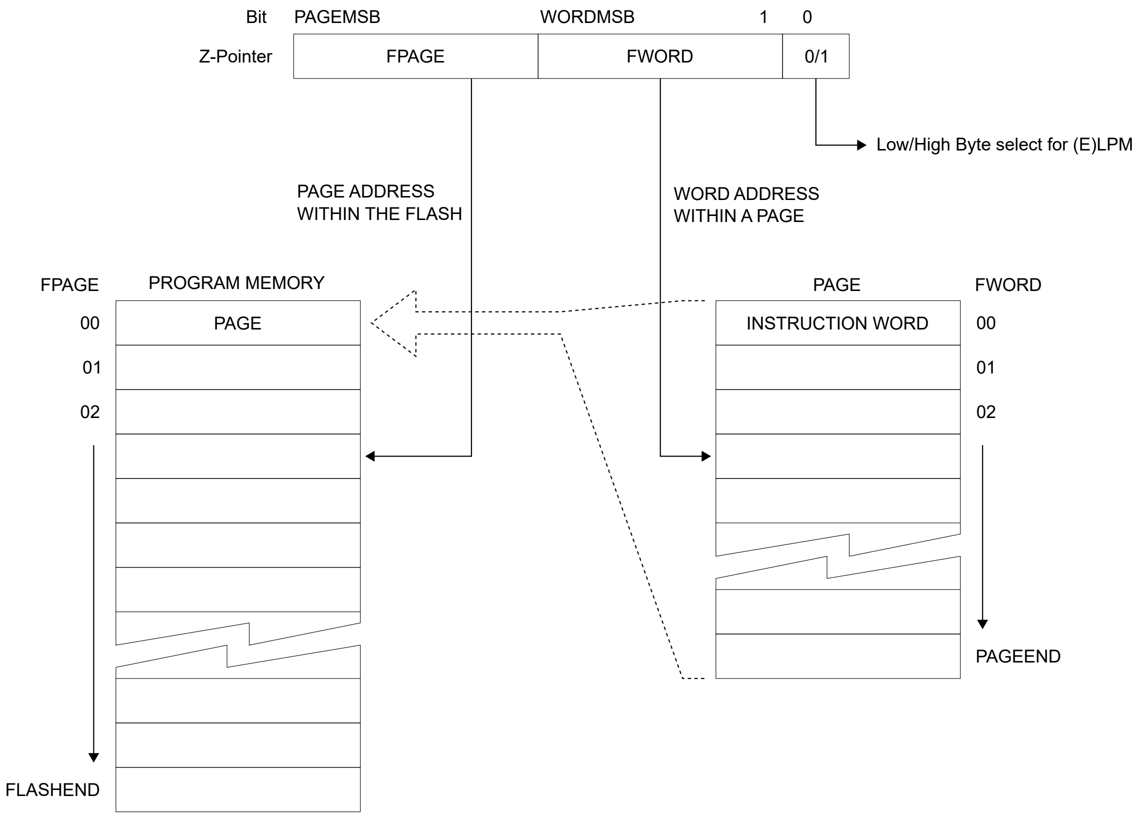
Additionally, the Flash memory is byte accessible when accessed through the CPU data space, meaning that it shares the same address space and instructions as SRAM, EEPROM, and I/O registers and it is accessible using `LD/ST` instructions in assembly.

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

Addressing Flash Memory in Code Space

For read and write access to the Flash memory in the code space, use the Z-pointer for `LPM`/`SPM` access.

Figure 11-4. 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 Figure 11-4. The word address in the page (FWORD) is held by the Least Significant bits in the Address Pointer, while the Most Significant bits in the Address Pointer hold the Flash page address (FPAGE). Together, FWORD and FPAGE hold an absolute address to a word in the Flash.

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

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

Once 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 Flash area in the data space has only 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 NVMCTRL.CTRLB register.

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

11.3.2.1 Programming

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 an ordinary Flash. When the erasing operation is used, the entire User Row is erased at once. The User Row writing has byte granularity.

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

Table 11-4. Programming Granularity

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

11.3.2.2 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.

11.3.2.3 Command Modes

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

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

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

To write a command in the NVMCTRL.CTRLA register, the following sequence needs to be executed:
1. Confirm that any previous operation is completed by reading the Busy (EEBUSY and FBUSY) flags in the NVMCTRL.STATUS register.
2. Write the appropriate key to the Configuration Change Protection (CPU.CCP) register to unlock the NVM Control A (NVMCTRL.CTRLA) register.
3. Write the desired command value to the CMD bit field in the Control A (NVMCTRL.CTRLA) register within the next four instructions.

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 NVMCTRL.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 NVMCTRL.CTRLB register.
3. Write the desired command value to the NVMCTRL.CTRLA register as described before.
4. Write to the correct address in the data space/code space using the `ST*`/`SPM` instructions.
5. Optional: If multiple write operations are required, go to step 4.
6. Write a `NOOP` or `NOCMD` command to the NVMCTRL.CTRLA register to clear the current command.

11.3.2.3.1 Page Write Command

The Page Write (FLPW/EEPW) commands write the content of the page buffer to the Flash or EEPROM.

If the write is to the Flash, the CPU will execute code as explained in 11.3.1.1.1. Physical Sections.

If the write is to the EEPROM, the CPU will continue executing code while the operation is ongoing. Erase the page/byte before performing a write.

The page buffer used will automatically be cleared after finishing the operation.

11.3.2.3.2 Page Erase Command

The Page Erase (FLPER/EEPER) commands erase the current page. Write one byte in the page buffer for the Page Erase command to take effect.

For erasing the Flash, a dummy write to one address in the desired page is required first, followed by command execution. The whole page in the Flash will then be erased. The CPU will stop or continue based on the same conditions as the Page Write command.

When executing the command for the EEPROM, only the bytes written in the page buffer will be erased. To erase a specific byte, write to its corresponding address before executing the command. Update all the bytes in the page buffer before writing the command to erase a whole page. The CPU will continue running code while the operation is ongoing.

The page buffer used will automatically be cleared after finishing the operation.

11.3.2.3.3 Flash Multi-Page Erase Mode

The Multi-Page Erase (FLMPERn) mode will allow each write to the memory array to erase multiple pages. When enabling FLMPERn, it is possible to select between erasing 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 11-5. Flash Multi-Page Erase

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

Note: FPAGE is the page number when doing a Flash erase. Refer to 11.3.2. Memory Access for details.

11.3.2.3.4 EEPROM Write Mode

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

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

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

11.3.2.3.5 EEPROM Erase/Write Mode

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

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

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

11.3.2.3.6 EEPROM Byte Erase Mode

The EEPROM Byte Erase (EEBER) mode 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 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.

11.3.2.3.7 EEPROM Multi-Byte Erase Mode

The EEPROM Multi-Byte Erase (EEMBERn) mode allows erasing several bytes in one operation. When enabling the EEMBERn mode, it is possible to select between erasing 2, 4, 8, 16, or 32 bytes in one operation.

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

Table 11-6. 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 be halted until finishing the current operation.

11.3.2.3.8 Chip Erase Command

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

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

This command can only be started from the UPDI.

11.3.2.3.9 EEPROM Erase Command

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

11.3.3 Preventing Flash/EEPROM Corruption

A Flash/EEPROM write or erase can cause memory corruption if the supply voltage is too low for the CPU and the Flash/EEPROM to operate 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.

11.3.4 Interrupts

Table 11-7. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
EEREADY	NVM	The EEPROM is ready for new write/erase operations
FLREADY		The Flash is ready for new write/erase operations

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

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

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. Refer to the NVMCTRL.INTFLAGS register for details on how to clear interrupt flags.

11.3.5 Sleep Mode Operation

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

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

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

11.3.6 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a 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 11-8. NVMCTRL - Registers under Configuration Change Protection

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

11.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		CMD[6:0]						
0x01	CTRLB	7:0	FLMAPLOCK		FLMAP[1:0]		EEWP	APPDATAWP	BOOTRP	APPCODEWP
0x02	CTRLC	7:0							BOOTROWWP	UROWWP
0x03	Reserved									
0x04	INTCTRL	7:0							FLREADY	EEREADY
0x05	INTFLAGS	7:0							FLREADY	EEREADY
0x06	STATUS	7:0		ERROR[2:0]					FBUSY	EEBUSY
0x07	Reserved									
0x08	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
0x0A ... 0x0B	Reserved									
0x0C	ADDR	7:0	ADDR[7:0]							
		15:8	ADDR[15:8]							
		23:16	ADDR[23:16]							
		31:24								

11.5 Register Description

11.5.1 Control A

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

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

Bits 6:0 – CMD[6:0] Command

Write this bit field to enable or issue a command. The Chip Erase and EEPROM Erase commands start when writing the command. The others enable an erase or write operation. The operation is started by doing a store instruction to an address location.

A change from one command to another must always go through a No command (`NOCMD`) or No operation (`NOOP`) command. If attempting written to 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 NVMCTRL.STATUS register.

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

11.5.2 Control B

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

Bit	7	6	5	4	3	2	1	0
	FLMAPLOCK		FLMAP[1:0]		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
Setting this bit to '1' prevents further updates of FLMAP[1:0]. Only a Reset can clear this bit.

Bits 5:4 – FLMAP[1:0] Flash Section Mapped into Data Space
Select which 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. For devices with Flash size of 32 KB or less, changing this bitfield value has no effect since the entire Flash is mapped to the CPU data space.

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 Protection
Writing a '1' to this bit protects the EEPROM from further writes.
This bit can only be written to '1'. It is cleared to '0' only by Reset.

Bit 2 – APPDATAWP Application Data Section Write Protection
Writing a '1' to this bit protects the application data section from further writes.
This bit can only be written to '1'. It is cleared to '0' only by Reset.

Bit 1 – BOOTRP Boot Section Read Protection
Writing a '1' to this bit protects the BOOT section from a read and instruction fetch. If issuing a read from the application section, it will return '0'. An instruction fetch from the BOOT section will return a `NOP` instruction.
This bit can only be written to '1' from the BOOT section. It is cleared to '0' only by Reset. The bit will only take effect when the BOOT section is left the first time after the bit is written.

Bit 0 – APPCODEWP Application Code Section Write Protection
Writing a '1' to this bit protects the application code section from further writes.
This bit can only be written to '1'. It is cleared to '0' only by Reset.

11.5.3 Control C

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

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

Bit 1 – BOOTROWWP Boot Row Write Protect

Writing this bit to `1` prevents further updates to the Boot Row. Only a reset can clear this bit.

Bit 0 – UROWWP User Row Write Protect

Writing this bit to `1` prevents further updates to the User Row. Only a reset can clear this bit.

11.5.4 Interrupt Control

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

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

Bit 1 – FLREADY Flash Ready Interrupt

Writing a '1' to this bit enables the interrupt indicating that the Flash is ready for new write/erase operations.

This is a level interrupt that will be triggered only when the FLREADY bit in the INTFLAGS register is set to '1'. The interrupt must be disabled in the interrupt handler before triggering a Flash write/erase operation, as the FLREADY bit will not be cleared before this command is issued.

Bit 0 – EEREADY EEPROM Ready Interrupt

Writing a '1' to this bit enables the interrupt indicating that the EEPROM is ready for new write/erase operations.

This is a level interrupt that will be triggered only when the EEREADY bit in the INTFLAGS register is set to '1'. The interrupt must be disabled in the interrupt handler before triggering an EEPROM write/erase operation, as the EEREADY bit will not be cleared before this command is issued.

11.5.5 Interrupt Flags

Name:	INTFLAGS
Offset:	0x05
Reset:	0x00
Property:	-

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

Bit 1 – FLREADY FLREADY Interrupt Flag

This flag is set continuously as long as the Flash is not busy. This flag is cleared by writing a '`1`' to it.

Bit 0 – EEREADY EEREADY Interrupt Flag

This flag is set continuously as long as the EEPROM is not busy. This flag is cleared by writing a '`1`' to it.

11.5.6 Status

Name:	STATUS
Offset:	0x06
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

Error code bit field reports the status of the last programming operation. `INVALIDCMD` and `WRITEPROTECT` are cleared only if/whenthe operation that caused the error is followed by a legal one. If `CMDCOLLISION` error occurs, then any new programming operation is ignored, until the error is cleared. Ensure no programming operation is ongoing (see FBUSY and EEBUSY flags) before clearing this error, otherwise the error will be reported once more.

The Error Code bit field can be cleared by writing '`0`' to it.

Notes: Rules for error/halting:

1. If changing command while programming is ongoing, then `CMDCOLLISION` error is set.
2. If ERROR = `CMDCOLLISION`, then the programming operation is ignored.
3. If accessing (read/write) while NVM section is busy, then the CPU is halted.

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

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.

11.5.7 Data

Name:	DATA
Offset:	0x08
Reset:	0x00
Property:	-

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

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

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

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

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

Bits 15:0 – DATA[15:0] Data Register
The Data register will contain the last read value from Flash, EEPROM, or NVMCTRL. For EEPROM access, only DATA[7:0] is used.

11.5.8 Address

Name: ADDR
Offset: 0x0C
Reset: 0x00
Property: -

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

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

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

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

Bit	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
The Address register contains the address of the last accessed memory location. Only the number of bits required to access the memory is used.

12. CLKCTRL - Clock Controller

12.1 Features

- All Clocks and Clock Sources Are Automatically Enabled When Requested by Peripherals
- Internal Oscillators:
  - Up to 24 MHz Internal High-Frequency Oscillator (OSCHF)
  - 32.768 kHz Ultra-Low Power Oscillator (OSC32K)
  - 48 MHz Phase-Locked Loop (PLL48M)
- Auto-Tuning for Improved Internal Oscillator Accuracy
- External Clock Options:
  - 32.768 kHz Crystal Oscillator (XOSC32K)
  - High-Frequency Crystal Oscillator (XOSCHF)
  - External clock
- Main Clock Features:
  - Safe run-time switching
  - Prescaler with a division factor ranging from 1 to 64
  - Clock Failure Detection with automatic clock switching to an internal source

12.2 Overview

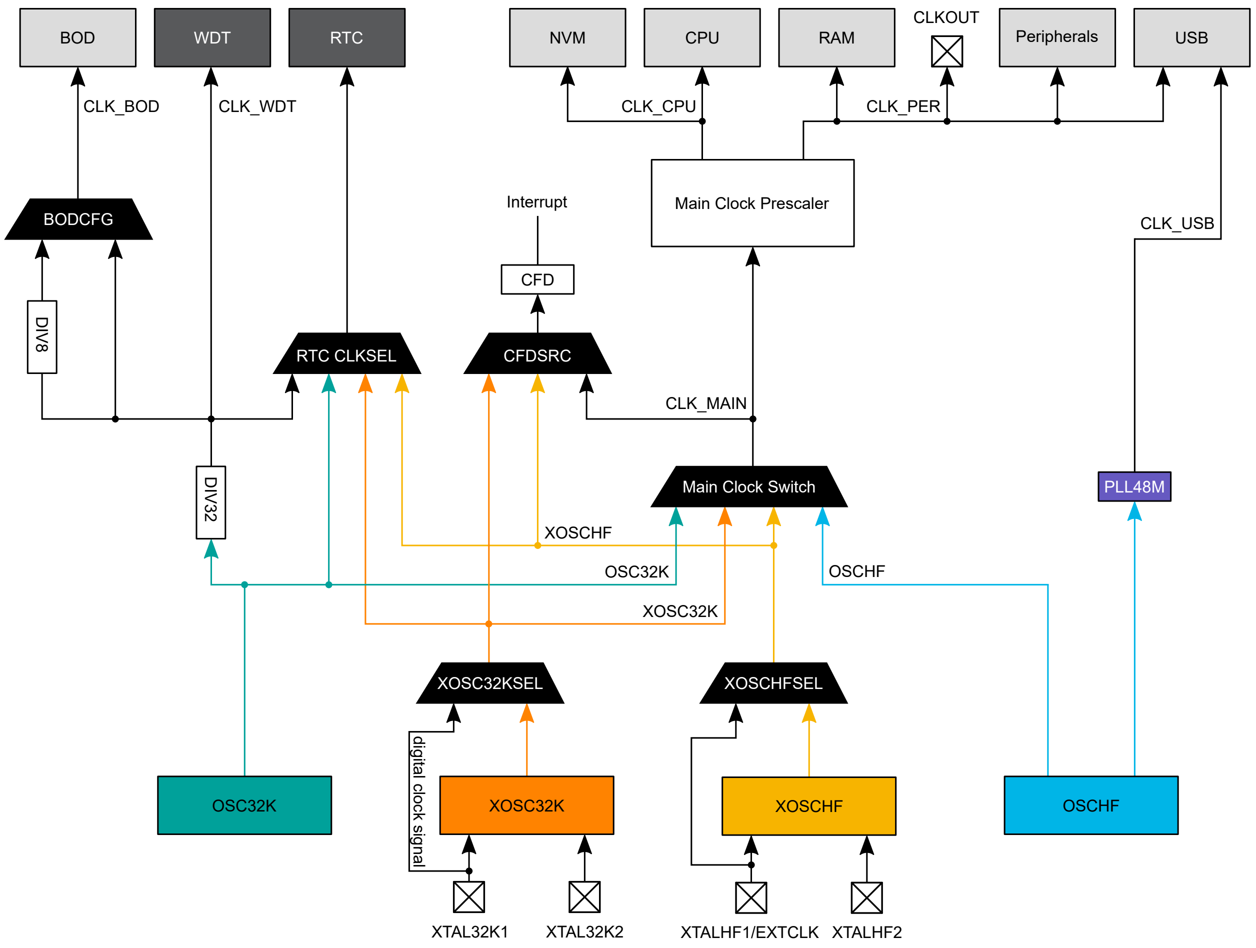
The Clock Controller (CLKCTRL) controls, distributes and prescales the clock signals from the available oscillators and supports internal and external clock sources.

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.

12.2.1 Block Diagram

Figure 12-1. CLKCTRL Block Diagram



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

- Main Clock (CLK_MAIN) is always running in Active and Idle sleep modes. If requested, it will also run in Standby sleep mode.
- CLK_MAIN is prescaled and distributed by the clock controller:
  - CLK_CPU is used by the CPU, SRAM and the Nonvolatile Memory Controller (NVMCTRL) peripheral
  - CLK_PER is used by all peripherals 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 CLK_RTC clock source may be changed only if the peripheral is disabled.
  - CLK_WDT is used by the Watchdog Timer (WDT). It will be requested when the WDT is enabled.
  - CLK_BOD is used by the Brown-out Detector (BOD). It will be requested when the BOD is enabled in Sampled mode. A fuse controls the alternative clock source.

- CLK_USB is used by the Universal Serial Bus (USB) peripheral. It will be requested when the USB is enabled.
- Clock Failure Detector (CFD) is an asynchronous mechanism to detect a failure on an external crystal or clock source

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

12.2.2 Signal Description

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

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

12.3 Functional Description

12.3.1 Initialization

To initialize a clock source as the main clock, these steps 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.

12.3.2 Main Clock Selection and Prescaler

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

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

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

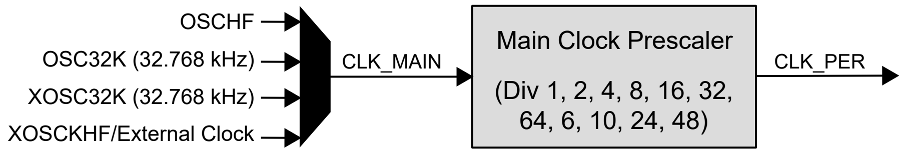
Upon the selection of an external clock source, a switch to the chosen clock source will occur only if edges are detected. Until a sufficient number of clock edges are detected, the switch will not occur, and it will not be possible to change to another clock source again without executing a Reset.

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

CAUTION If an external clock source fails while used as the CLK_MAIN source, the clock source will default to the start-up clock source only if the CFD is enabled. If the CFD is not enabled, only the Watchdog Timer (WDT) can provide a System Reset. For more details, refer to the *Clock Failure Detection (CFD)* section.

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

Figure 12-2. Main Clock and Prescaler



12.3.3 Main Clock After Reset

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

12.3.4 Clock Sources

All the internal clock sources are automatically enabled when 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 or 32.768 kHz oscillator.

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

12.3.4.1 Internal Oscillators

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

12.3.4.1.1 Internal High-Frequency Oscillator (OSCHF)

The OSCHF supports output frequencies of 1, 2, 3, and 4 MHz and multiples of four, up to 24 MHz, which can be used as the main clock or peripheral clock. The OSCHF also supports a separate fixed 4 MHz output to the Phase-Locked Loop (PLL48M).

12.3.4.1.2 32.768 kHz Oscillator (OSC32K)

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

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

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

12.3.4.2 External Clock Sources

These external clock sources are available:

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

12.3.4.2.1 High-Frequency Crystal Oscillator (XOSCHF)

This oscillator supports two input options:

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

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

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

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

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

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

12.3.4.2.2 32.768 kHz Crystal Oscillator (XOSC32K)

This oscillator supports two input options:

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

Configure the input option by writing the Source Select (SEL) bit in the XOSC32K Control A (CLKCTRL.XOSC32KCTRLA) register.

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

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

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

12.3.5 Tuning the Internal High-Frequency Oscillator

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

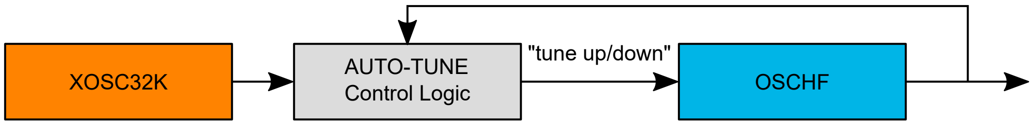
Manual Tuning

Disable the Automatic Oscillator Tune (AUTOTUNE) bit field in the Internal High-Frequency Oscillator Control A (CLKCTRL.OSCHFCTRLA) register by writing '0' to the bit field. Tune the output frequency of the OSCHF manually by writing the Oscillator Frequency Tuning (TUNE) bit field in the Internal High-Frequency Oscillator Frequency Tune (CLKCTRL.OSCHFTUNE) register.

Auto-Tune Against an External Crystal Oscillator

The OSCHF output frequency can be calibrated by automatic tuning against XOSC32K. Enable the auto-tune by writing the 32K setting to the AUTOTUNE bit field in the OSCHFCTRLA register, locking the OSCHFTUNE register, and no manual tuning is possible. The OSCHFTUNE register is updated with the latest auto-tuned value when AUTOTUNE is disabled.

Figure 12-3. OSCHF Auto-Tune Block Diagram



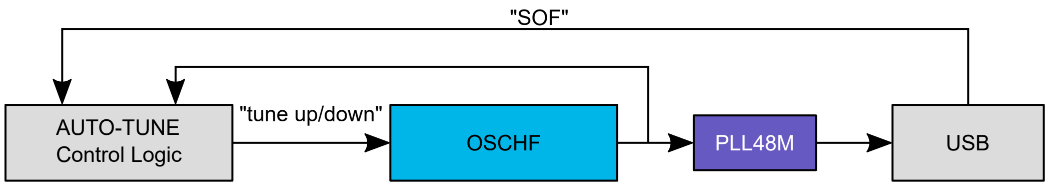
Refer to the *Electrical Characteristics* section for details.

Auto-Tune Against Start-of-Frame signal

The output frequency of the OSCHF must be tuned against the Start-of-Frame (SOF) signal from the USB when using the USB peripheral.

Enable the auto-tune by writing the SOF setting to the AUTOTUNE bit field in the OSCHFCTRLA register, locking the OSCHFTUNE register, and no manual tuning is possible. The SOF auto-tune has two options for algorithm selection, Binary Search and Incremental Search, respectively. These can be chosen by writing the ALGSEL bit in the OSCHFCTRLA register. Using the binary search algorithm may temporarily change the oscillator output frequency up to half the tune range, affecting running timing critical asynchronous peripherals, such as the USART or a timer generating PWM waveforms. The OSCHFTUNE register is updated with the latest auto-tuned value when AUTOTUNE is disabled.

Figure 12-4. OSCHF Start-Of-Frame Auto-Tune Block Diagram



Refer to the *USB* section for details.

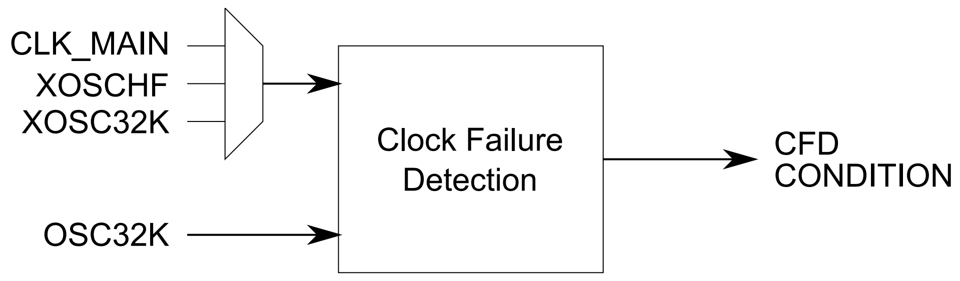
12.3.6 Clock Failure Detection (CFD)

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

12.3.6.1 Clock Failure Detection (CFD) Operation

The Clock Failure Detection (CFD) feature detects a failed oscillator or clock source by checking for edges on the selected oscillator/clock. If no edges are detected within a specific time, a CFD condition is issued and triggers an interrupt or forces the device to switch to a stable internal clock source.

Figure 12-5. 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 Main Clock Control C (CLKCTRL.MCLKCTRLC) register. In sleep, the CFD will only be enabled if the selected source is active.

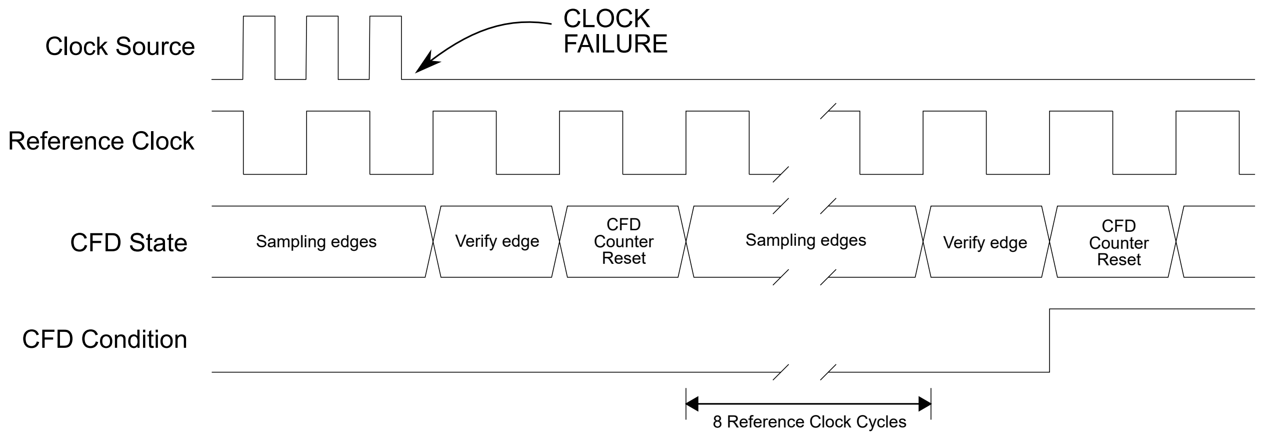
If a CFD condition occurs, the CFD interrupt flag in the Main Clock Interrupt Flags (CLKCTRL.MCLKINTFLAGS) register is set. If the interrupt is enabled, an interrupt request is issued. The Interrupt Type (INTTYP) bit in the Main Clock Interrupt Control (CLKCTRL.MCLKINTCTRL) register determines if a normal interrupt or a Non-Maskable Interrupt (NMI) will be issued. If the NMI is selected, and more than one interrupt source is set to NMI, it is necessary to check the vector to see which source generated an interrupt.

If the monitored clock source is the main clock and it fails, everything running on it will stop. If this happens, the CFD condition will overwrite the Clock Selection (CLKSEL) bit field in the Main Clock Control A (CLKCTRL.MCLKCTRLA) register to select the start-up clock source, which is changed back to its Reset frequency.

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

When the CLKSEL is overridden by a CFD event, the CLKOUT signal will be disabled.

Figure 12-6. Clock Failure Detection (CFD) Timing Diagram



12.3.6.2 Condition Clearing

The CFD condition is cleared after a Reset, the monitored source starts toggling again, or the CFD flag in the Main Clock Interrupt Flags (CLKCTRL.MCLKINTFLAGS) register is set. 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 (CFD) interrupt enable bit in the Main Clock Interrupt Control (CLKCTRL.MCLKCTRL) register. If it is the main clock that is being monitored, changing back to the default start-up clock will make the main clock start toggling again, clearing the condition.

12.3.6.3 CFD Test

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

12.3.6.3.1 Testing the CFD Without Influencing the Main Clock

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

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

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

12.3.6.3.2 Testing the CFD and Changing the Main Clock to the Start-Up Clock Source

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

12.3.7 Interrupts

Table 12-1. Available Interrupt Vectors and Sources

Interrupt Vector Name	Interrupt Source Name	Description	Condition
NMI	CFD	External crystal oscillator or clock source failure	The CFD flag in CLKCTRL.MCLKINTFLAGS is '`1`' and the INTTYPE bit in CLKCTRL.MCLKINTCTRL is '`1`'
CLKCTRL			The CFD flag in CLKCTRL.MCLKINTFLAGS is '`1`' and the INTTYPE bit in CLKCTRL.MCLKINTCTRL is '`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.

12.3.8 Sleep Mode Operation

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

The main clock will always run in Active 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.

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

12.3.9 Configuration Change Protection (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 12-2. CLKCTRL - Registers Under Configuration Change Protection

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

12.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	MCLKCTRLA	7:0	CLKOUT				CLKSEL[3:0]			
0x01	MCLKCTRLB	7:0				PDIV[3:0]				PEN
0x02	MCLKCTRLC	7:0					CFDSRC[1:0]		CFDTST	CFDEN
0x03	MCLKINTCTRL	7:0	INTTYPE							CFD
0x04	MCLKINTFLAGS	7:0								CFD
0x05	MCLKSTATUS	7:0				EXTS	XOSC32KS	OSC32KS	OSCHFS	SOSC
0x06	MCLKTIMEBASE	7:0				TIMEBASE[4:0]				
0x07	Reserved									
0x08	OSCHFCTRLA	7:0	RUNSTDBY	ALGSEL	FRQSEL[3:0]				AUTOTUNE[1:0]	
0x09	OSCHFTUNE	7:0		TUNE[6:0]						
0x0A	OSCHFSTATUS	7:0							ATLOCK	ATSYNC
0x0B ... 0x17	Reserved									
0x18	OSC32KCTRLA	7:0	RUNSTDBY							
0x19 ... 0x1B	Reserved									
0x1C	XOSC32KCTRLA	7:0	RUNSTDBY		CSUT[1:0]			SEL	LPMODE	ENABLE
0x1D ... 0x1F	Reserved									
0x20	XOSCHFCTRLA	7:0	RUNSTDBY		CSUTHF[1:0]		FRQRANGE[1:0]		SELHF	ENABLE
0x21 ... 0x24	Reserved									
0x25	USBPLLSTATUS	7:0								PLLS

12.5 Register Description

12.5.1 Main Clock Control A

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

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

Bit 7 – CLKOUT Main Clock Out

This bit controls whether the main clock is available on the Main Clock Out (CLKOUT) pin or not when the main clock is running.
This bit is cleared when a '`0`' is written to it or when a Clock Failure Detection (CFD) condition with the main clock as 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

Bits 3:0 – CLKSEL[3:0] Clock Select

This bit field controls the source for the Main Clock (CLK_MAIN).

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

12.5.2 Main Clock Control B

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

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

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

Value	Name	Description
0x0	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

12.5.3 Main Clock Control C

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

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

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

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

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

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

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

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

Value	Description
0	CFD is disabled
1	CFD is enabled

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

12.5.4 Main Clock Interrupt Control

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

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

Bit 7 – INTTYPE Interrupt Type

This bit controls the type of CFD interrupt.

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

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

Bit 0 – CFD Clock Failure Detection Interrupt Enable

This bit controls whether the CFD interrupt is enabled or not.

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

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

12.5.5 Main Clock Interrupt Flags

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

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

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

12.5.6 Main Clock Status

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

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

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 runs 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 runs when the SEL bit is '1'.

Bit 2 – OSC32KS OSC32K Status

Value	Description
0	OSC32K is not stable
1	OSC32K is stable

Bit 1 – OSCHFS Internal High-Frequency Oscillator Status

Value	Description
0	OSCHF is not stable
1	OSCHF is stable

Bit 0 – SOSC Main Clock Oscillator Changing

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

12.5.7 Main Clock Timebase

Name: MCLKTIMEBASE
Offset: 0x06
Reset: 0x00

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.

12.5.8 Internal High-Frequency Oscillator Control A

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

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

Bit 6 – ALGSEL Algorithm Selection

This bit determines which tuning algorithm is used after a USB bus reset. It must be written simultaneously with writing the SOF setting to the AUTOTUNE bit field to configure the tuning logic. This bit is considerable only when the SOF setting is written to the AUTOTUNE bit field.

Value	Description
0	Binary search(1)
1	Incremental search up to five tune steps

Note:
1. Using the binary search algorithm may temporarily change the oscillator output frequency to up half the tune range, affecting timing critical asynchronous peripherals running, such as the USART or a timer generating PWM waveforms.

Bits 5:2 – FRQSEL[3:0] Frequency Select

This bit field controls the internal high-frequency oscillator (OSCHF) output frequency.

Value	Name	Description
0x00	1M	1 MHz output
0x01	2M	2 MHz output
0x02	3M	3 MHz output
0x03	4M	4 MHz output (default)
0x04	-	Reserved
0x05	8M	8 MHz output
0x06	12M	12 MHz output
0x07	16M	16 MHz output
0x08	20M	20 MHz output
0x09	24M	24 MHz output
Other	-	Reserved

Bits 1:0 – AUTOTUNE[1:0] Automatic Oscillator Tune

This bit field controls whether the internal high-frequency oscillator (OSCHF) auto-tune functionality is enabled or not.

Value	Name	Description
`0x00`	OFF	Automatic oscillator frequency tune disabled
`0x01`	32K	Automatic oscillator frequency tune against XOSC32K enabled
`0x02`	SOF	Automatic oscillator frequency tune against USB SOF enabled

12.5.9 Internal High-Frequency Oscillator Frequency Tune

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

Bit	7	6	5	4	3	2	1	0
		TUNE[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 – TUNE[6:0] Oscillator Frequency Tune

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 up from the oscillator's target frequency. Thus, the register's acceptable input value range is -32 to +31.
Writing to bit 6 has no effect, as bit 5 will be mirrored to bit 6 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 field in the Internal High-Frequency Oscillator Control A (CLKCTRL.OSCHFCTRLA) register is written with a value other than '0', 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.

12.5.10 Internal High-Frequency Oscillator Status

Name:	OSCHFSTATUS
Offset:	0x0A
Reset:	0x00
Property:	-

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

Bit 1 – ATLOCK Automatic oscillator tune in lock

This bit is set when the tune algorithm has locked on the correct frequency.

Value	Description
0	Selected auto-tune algorithm is not in lock
1	Selected auto-tune algorithm is in lock

Bit 0 – ATSYNC Auto-tune Synchronized

This bit is set when the new value written to the AUTOTUNE bit field in the OSCHFCTRLA register has been synchronized to the 1 MHz clock domain.

Value	Description
0	Synchronization in progress
1	Synchronization done/No synchronization in progress

12.5.11 Internal 32.768 kHz Oscillator Control A

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

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

Bit 7 – RUNSTDBY Run Standby

This bit controls whether the 32.768 kHz Oscillator (OSC32K) is always running.

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.

12.5.12 External 32.768 kHz Crystal Oscillator Control A

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

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

Bit 7 – RUNSTDBY Run Standby

This bit controls whether the 32.768 kHz Crystal Oscillator (XOSC32K) is always running 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	Description
0	External crystal connected to the XTAL32K1 and XTAL32K2 pins
1	External clock on the XTAL32K1 pin

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

Bit 1 – LPMODE Low-Power Mode

This bit controls whether the 32.768 kHz Crystal Oscillator (XOSC32K) is in Low-Power mode.

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

12.5.13 External High-Frequency Oscillator Control A

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

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY		CSUTHF[1:0]		FRQRANGE[1:0]		SELHF	ENABLE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit 7 – RUNSTDBY Run Standby
This bit controls whether the External High-Frequency Oscillator (XOSCHF) is always running or not when the ENABLE bit is '1'.

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

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

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

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

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

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

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

Note: If a crystal with a frequency 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

12.5.14 USB PLL Status

Name: USBPLLSTATUS
Offset: 0x25
Reset: 0x00

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

Bit 0 – PLLS PLL Status

This bit shows the PLL status.

Value	Description
0	PLL is not running
1	PLL is running

13. SLPCTRL - Sleep Controller

13.1 Features

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

13.2 Overview

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

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

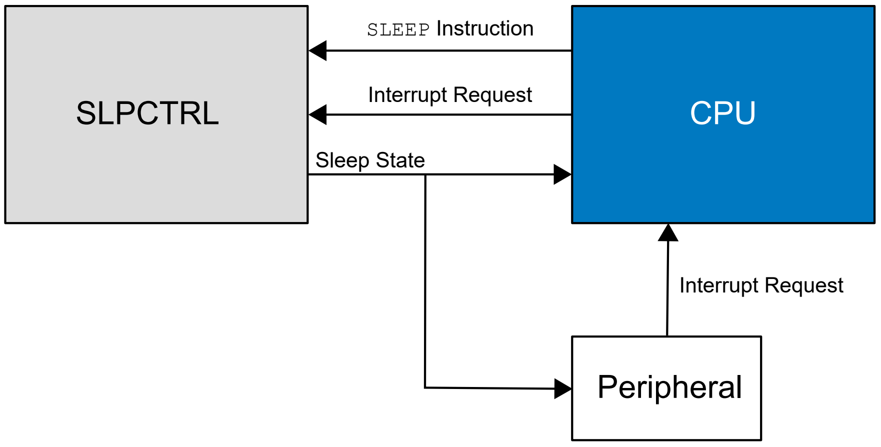
All sleep modes are available and can be entered from the Active mode. In Active mode, the CPU is executing application code. When the device enters sleep mode, the program execution is stopped. The application code decides which sleep mode to enter and when.

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

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

13.2.1 Block Diagram

Figure 13-1. SLPCTRL Block Diagram



13.3 Functional Description

13.3.1 Initialization

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

1. Configure and enable the interrupts that can wake the device from sleep.

Enable also the global interrupts.

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

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

13.3.2 Voltage Regulator Configuration

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

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

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

Table 13-1. Voltage Regulator Power Modes Description

Voltage Regulator Power Mode	Description	Condition	Active/Idle	Standby/Power-Down
Normal (AUTO)	Maximum performance in Active mode and Idle mode	External clock or fast oscillator	Maximum Performance	Low Power
		32.768 kHz oscillator	Low Power	Low Power
Performance (FULL)	Maximum performance in all modes (Active and Sleep) and fast start-up from all sleep modes		Maximum Performance	Maximum Performance

13.3.3 Operation

13.3.3.1 Sleep Modes

Three different sleep modes can be enabled to reduce power consumption.

Idle	The CPU stops executing code, resulting in reduced power consumption. All peripherals are running, and all interrupt sources can wake the device.
Standby	All high-frequency clocks are stopped unless running in Standby sleep mode is enabled for a peripheral or clock. This is enabled by writing the corresponding RUNSTDBY bit to '1'. The power consumption is dependent on the enabled functionality. A subset of interrupt sources can wake the device[(1)].
Power-Down	All high-frequency clocks are stopped, resulting in a power consumption lower than the Idle sleep mode. When operating at temperatures above 70°C, the power consumption can be reduced further by writing the High-Temperature Low Leakage Enable (HTLLEN) bit in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register to '1'. A subset of the peripherals are running, and a subset of interrupt sources can wake the device.[(1)]

Important: The TWI address match and CCL wake-up sources must be disabled when High-Temperature Low Leakage Enable is activated to avoid unpredictable behavior.

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

Refer to the *Wake-up Time* section for information on how the wake-up time is affected by the different sleep modes.

Table 13-2. Sleep Mode Activity Overview for Peripherals

Clock	Peripheral	Active in Sleep Mode			
		Idle	Standby	Power-Down VREGCTRL.HTLLEN =0	Power-Down VREGCTRL.HTLLEN =1
CLK_CPU	CPU				
CLK_RTC	RTC	X	X(1,2)	X(2)	X(2)
CLK_WDT	WDT	X	X	X	X
CLK_BOD(3)	BOD	X	X	X	X
(4)	CCL	X	X(1)		
CLK_PER	EVSYS	X	X	X	X
	ACn	X	X(1)		
	ADCn				
	TCAn				
	TCBn				
	All other peripherals	X			

Notes:
1. RUNSTDBY bit of the corresponding peripheral must be set to enter an active state.
2. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY to be set to enter an active state. In Power-Down sleep mode, only the PIT functionality is available.
3. CLK_BOD is required only when the BOD is running in Sampled mode.
4. The clock domain depends on the clock source selected for CCL.

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

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

Notes:
1. RUNSTDBY bit of the corresponding peripheral must be set to enter an active state.
2. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY to be set to enter an active state. In Power-Down sleep mode, only the PIT functionality is available.
3. CLK_BOD is required only when the BOD is running in Sampled mode.

Table 13-4. Sleep Mode Wake-Up Sources

Wake-Up Source	Active in Sleep Mode			
	Idle	Standby	Power-Down VREGCTRL.HTLLEN =0	Power-Down VREGCTRL.HTLLEN =1
PORT Pin interrupt	X	X[1]	X[1]	X[1]
BOD VLM interrupt	X	X	X	X
RTC interrupts	X	X[2,4]	X[4]	X[4]
Periodic Interrupt	X	X	X[4,5]	
TWI Address Match interrupt	X	X	X[5]	
CCL interrupts	X	X[2]	X[3]	X[3]
USART Start-of-Frame interrupt	X	X		
TCA0 interrupts	X	X		
TCB0 interrupts	X	X		
ADC0 window	X	X		
AC0 Compare interrupt	X	X		
USB Resume interrupt	X	X		
All other interrupts	X			

Notes:
1. The I/O pin has to be configured according to *Asynchronous Sensing Pin Properties* in the PORT section.
2. RUNSTDBY bit of the corresponding peripheral must be set to enter an active state.
3. CCL can wake up the device if the path through LUTn is asynchronous (FILTSEL=`0x0` and EDGEDET=`0x0` in the LUT n Control A (CCL.LUTnCTRLA) register).
4. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY to be set to enter an active state. In Power-Down sleep mode, only the PIT functionality is available.
5. Not available when High-Temperature Low Leakage is enabled (*i.e.*, HTLLEN = '`1`' in SLPCTRL.VREGCTRL).

13.3.3.2 Wake-Up Time

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

- In Idle 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 13-5. Sleep Modes and Start-Up Time

Sleep Mode	Start-Up Time
Idle	Six clock cycles

...........continued

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

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

In addition to the normal wake-up time, it is possible to make the device wait until the BOD is ready before executing the code. This is done by writing `0x3` to the BOD operation mode in the Active and Idle (ACTIVE) bit field in the BOD Configuration (FUSE.BODCFG) fuse. If the BOD is ready before the normal wake-up time, the total wake-up time will be the same. If the BOD takes longer than the normal wake-up time, the wake-up time will be extended until the BOD is ready. This ensures correct supply voltage whenever code is executed.

13.3.4 Debug Operation

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

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

13.3.5 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 13-6. SLPCTRL - Registers Under Configuration Change Protection

Register	Key
SLPCTRL.VREGCTRL	IOREG

13.4 Register Summary

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

13.5 Register Description

13.5.1 Control A

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

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

Bits 3:1 – SMODE[2:0] Sleep Mode

Writing these bits selects the desired sleep mode when the Sleep Enable (SEN) bit is written to '`1`' and the `SLEEP` instruction is executed.

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

Bit 0 – SEN Sleep Enable

This bit must be written to '`1`' before the `SLEEP` instruction is executed to make the microcontroller enter the selected sleep mode.

13.5.2 Voltage Regulator Control Register

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

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

Bit 4 – HTLLEN High-Temperature Low Leakage Enable

This bit controls whether the current leakage is reduced or not when operating at temperatures above 70°C.

Value	Name	Description
0	OFF	High-temperature low leakage disabled(1)
1	ON	High-temperature low leakage enabled(2,3)

WARNING

1. If entering the Standby sleep mode, this bit must be '0'.
2. This will only have an effect when PMODE is set to AUTO and must only be used for the Power-Down sleep mode.
3. The TWI address match and CCL wake-up sources must be disabled before writing this bit to '1'.

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	The regulator will run with maximum performance in active/idle mode unless the 32.768 kHz oscillator source is selected. Power saving in deep sleep modes.
0x1	FULL	Maximum performance voltage regulator drive strength in all modes. Faster start-up from sleep modes.
Other	-	Reserved

14. RSTCTRL - Reset Controller

14.1 Features

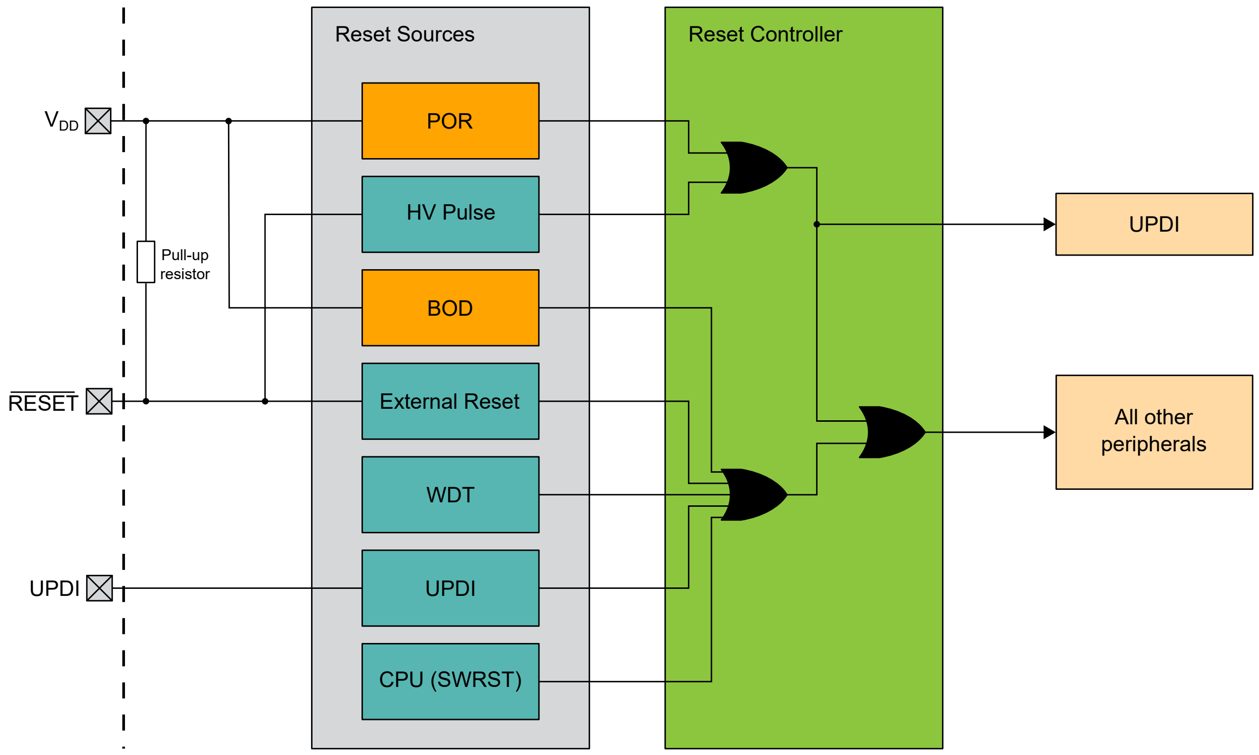
- Returns the Device to an Initial State After a Reset
- Identifies the Previous Reset Source
- Power Supply Reset Sources:
  - Power-on Reset (POR)
  - Brown-out Detector (BOD) Reset
- User Reset Sources:
  - External Reset ($\overline{\text{RESET}}$)
  - Watchdog Timer (WDT) Reset
  - Software Reset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

14.2 Overview

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

14.2.1 Block Diagram

Figure 14-1. Reset System Overview



14.2.2 Signal Description

Signal	Description	Type
$\overline{\text{RESET}}$	External Reset (active-low)	Digital input
UPDI	Unified Program and Debug Interface	Digital input

14.3 Functional Description

14.3.1 Initialization

The RSTCTRL is always enabled, but some of the Reset sources must be enabled individually (either by Fuses or 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.

14.3.2 Operation

14.3.2.1 Reset Sources

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

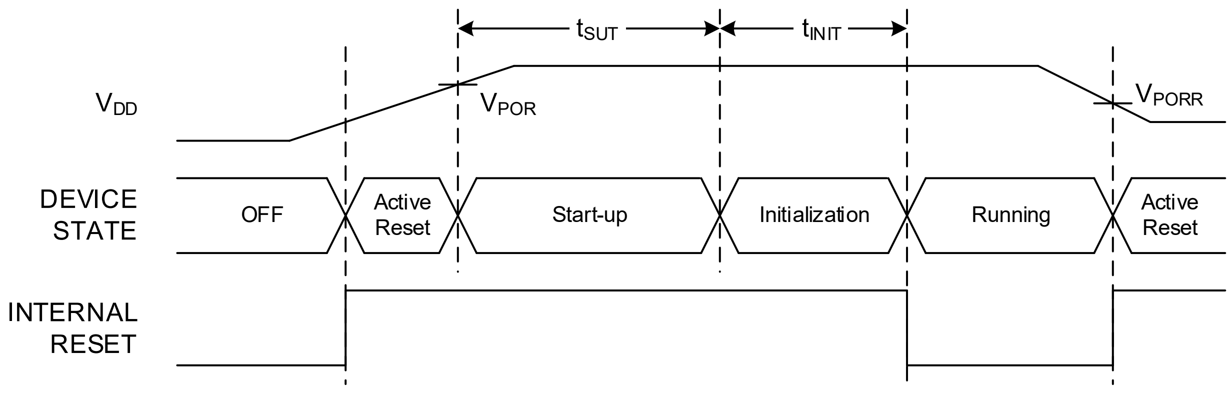
There are two types of Resets based on the source:

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

14.3.2.1.1 Power-on Reset (POR)

The Power-on Reset (POR) aim is to ensure a safe start-up of logic and memories. An on-chip detection circuit is always enabled and generates this. The POR is activated when the V_{DD} rises and gives active Reset as long as V_{DD} is below the POR threshold voltage (V_{POR}). The Reset will last until the Start-up and Reset initialization sequence is finished. Fuses determine the Start-Up Time (SUT). Reset is activated again, without delay, when V_{DD} falls below the detection level (V_{PORR}).

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

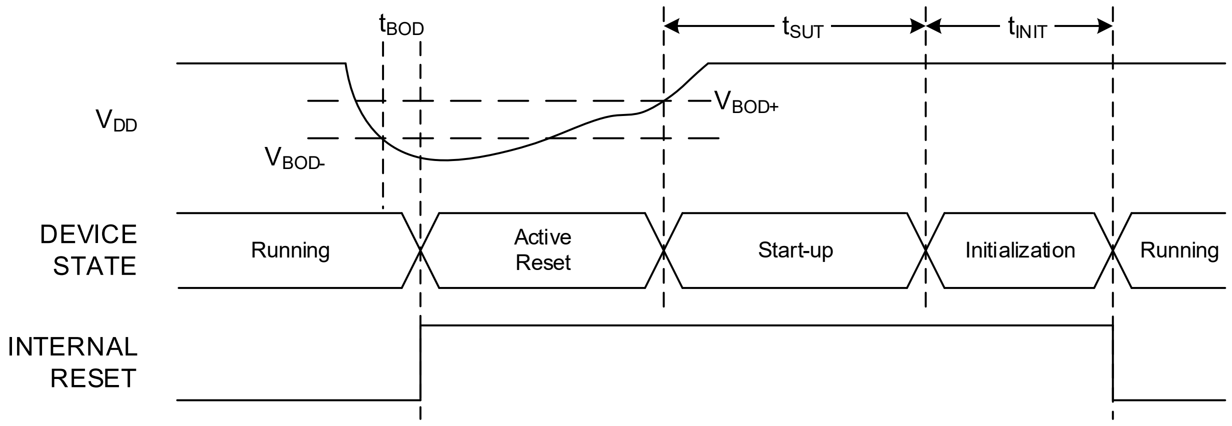


14.3.2.1.2 Brown-out Detector (BOD) Reset

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

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

Figure 14-3. Brown-out Detector Reset



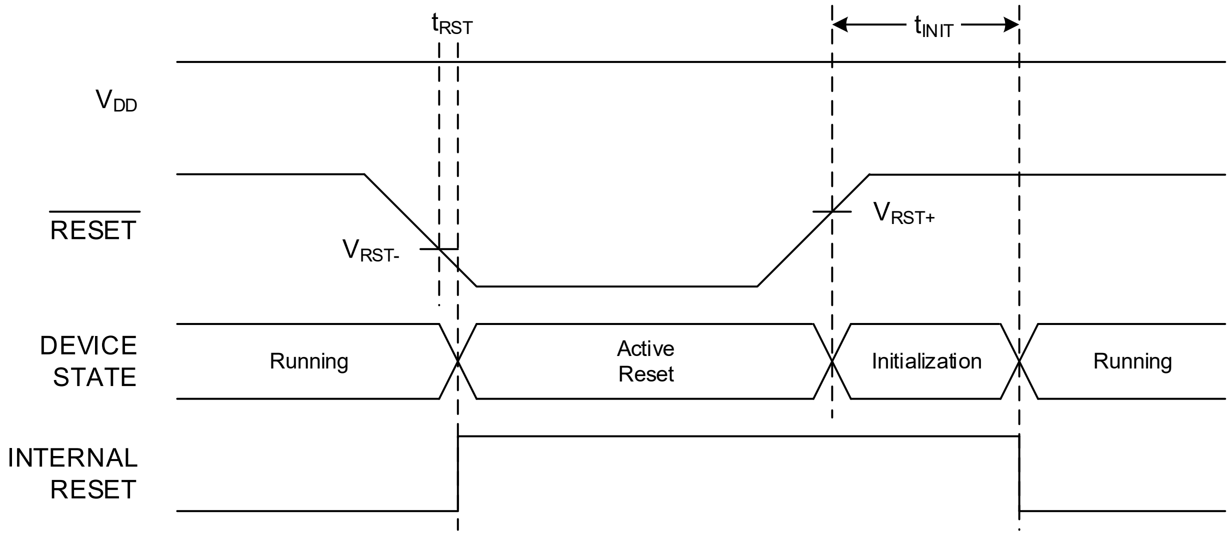
14.3.2.1.3 External Reset ($\overline{\text{RESET}}$)

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

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

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

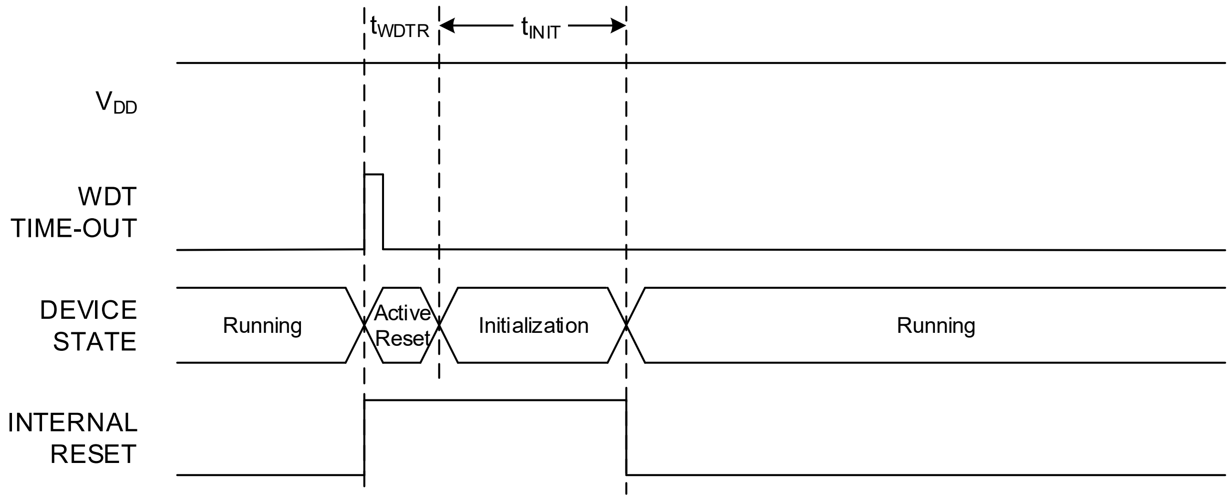
Figure 14-4. External Reset Characteristics



14.3.2.1.4 Watchdog Timer (WDT) Reset

The Watchdog Timer (WDT) is a system function that monitors the program's operation. A Watchdog Reset will be issued if the software doesn't handle the WDT according to the programmed time-out period. Find more details in the *WDT - Watchdog Timer* section.

Figure 14-5. Watchdog Reset



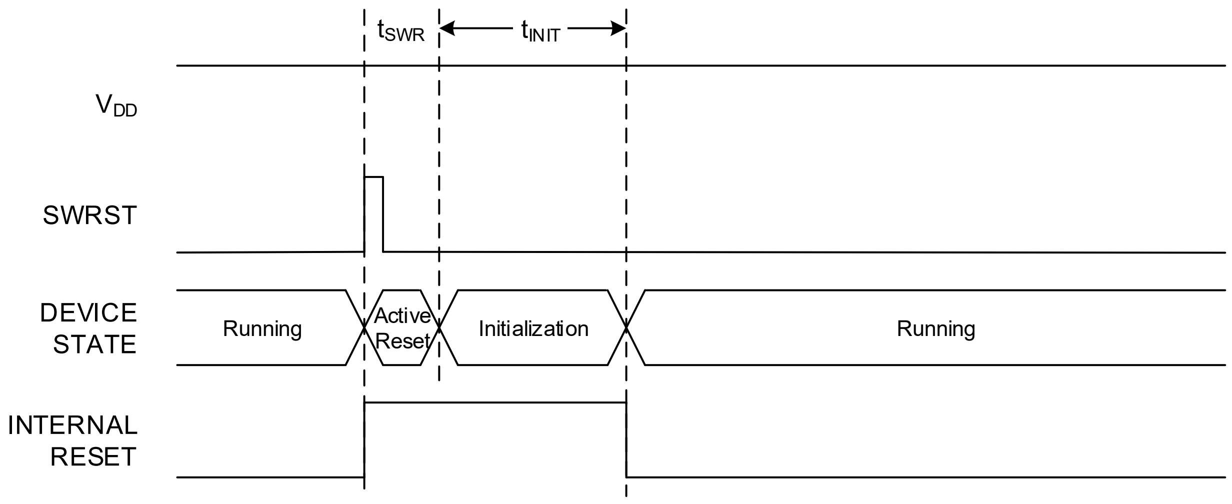
Note: The time t_{WDTR} is approximately 150 ns.

14.3.2.1.5 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 14-6. Software Reset



Note: The time t_{SWR} is approximately 150 ns.

14.3.2.1.6 Unified Program and Debug Interface (UPDI) Reset

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

14.3.2.1.7 High Voltage (HV) Pulse

A device Reset is issued if a high voltage is applied to or removed from the $\overline{\text{RESET}}$ pin. When using the HV pulse to enable the UPDI, it will cause a device Reset. Refer to the *UPDI - Unified Program and Debug Interface* section for more information on the HV pulse.

14.3.2.1.8 Domains Affected By Reset

The following logic domains are affected by the various Resets:

Table 14-1. Logic Domains Affected by Various Resets

Reset Type	Reset of BOD configuration	Fuses are Reloaded	Reset of UPDI	Reset of Other Volatile Logic
POR	X	X	X	X
BOD	X	X	X	X
External Reset		X		X
Watchdog Reset		X		X
Software Reset		X		X
UPDI Reset		X		X
High Voltage (HV) Pulse(1)		X	X	X

Note: 1. An HV pulse can cause a Reset but should not be used intentionally as a Reset source.

14.3.2.2 Reset Time

The Reset time can be split into two parts.

The first part is when any of the Reset sources are active. This part depends on the input to the Reset sources. The external Reset is active as long as the $\overline{\text{RESET}}$ pin is low. The Power-on Reset (POR) and the Brown-out Detector (BOD) are active 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 caused the Reset, this time will be increased with the start-up time given by the Start-Up Time (SUT) bit field in the System Configuration 1 (FUSE.SYSCFG1) fuse. The internal Reset initialization time will also increase if the Cyclic Redundancy Check Memory Scan (CRCSCAN) is configured to run at start-up. This configuration can be changed in the CRC Source (CRCSRC) bit field in the System Configuration 0 (FUSE.SYSCFG0) fuse.

14.3.3 Sleep Mode Operation

The RSTCTRL operates in Active mode and all sleep modes.

14.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a 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 14-2. RSTCTRL - Registers Under Configuration Change Protection

Register	Key
RSTCTRL.SWRR	IOREG

14.4 Register Summary

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

14.5 Register Description

14.5.1 Reset Flag Register

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

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

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

Bit 5 – UPDIRF UPDI Reset Flag
This bit is set to '1' if either a UPDI Reset has occurred or a reset caused by an HV pulse has occurred.

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

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

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

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

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

14.5.2 Software Reset Register

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

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

Bit 0 – SWRST Software Reset

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

15. CPUINT - CPU Interrupt Controller

15.1 Features

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

15.2 Overview

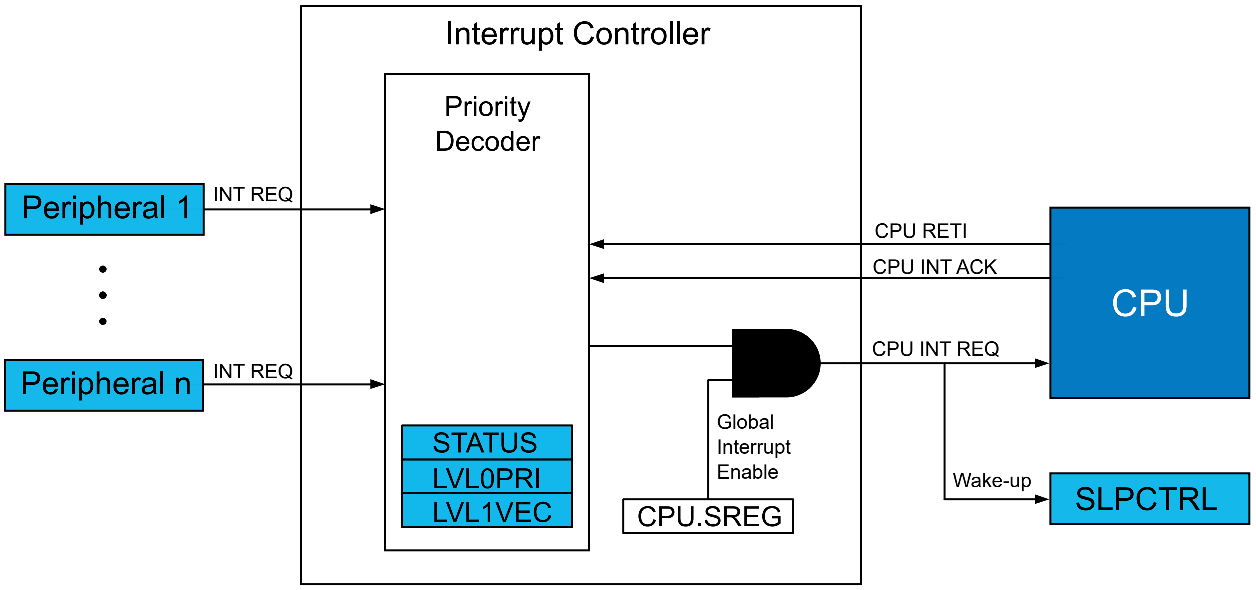
An interrupt request signals a state change inside a peripheral and can be used to alter the program execution. The peripherals can have one or more interrupts. All interrupts are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when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

15.2.1 Block Diagram

Figure 15-1. CPUINT Block Diagram



15.3 Functional Description

15.3.1 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.

15.3.2 Operation

15.3.2.1 Enabling, Disabling and Resetting

The global enabling of interrupts is done by writing a '`1`' to the Global Interrupt Enable (I) bit in the CPU Status (CPU.SREG) register. To disable interrupts globally, write a '`0`' to the I bit in CPU.SREG.

The desired interrupt lines must also be enabled in the respective peripheral by writing to the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

The interrupt flags are not automatically cleared after the interrupt is executed. The respective INTFLAGS register descriptions provide information on how to clear specific flags.

15.3.2.2 Interrupt Vector Locations

The 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.

15.3.2.3 Interrupt Response Time

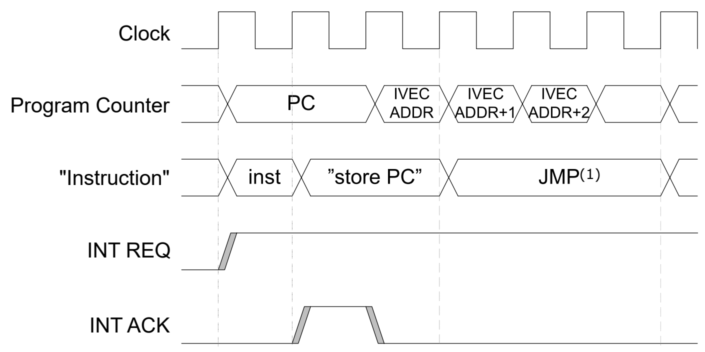
The minimum interrupt response time is represented in the following table.

Table 15-1. Minimum Interrupt Response Time

	Flash Size > 8 KB	Flash Size ≤ 8 KB
Finish ongoing instruction	One cycle	One cycle
Store PC to stack	Two cycles	Two cycles
Jump to interrupt handler	Three cycles (`jmp`)	Two cycles (`rjmp`)

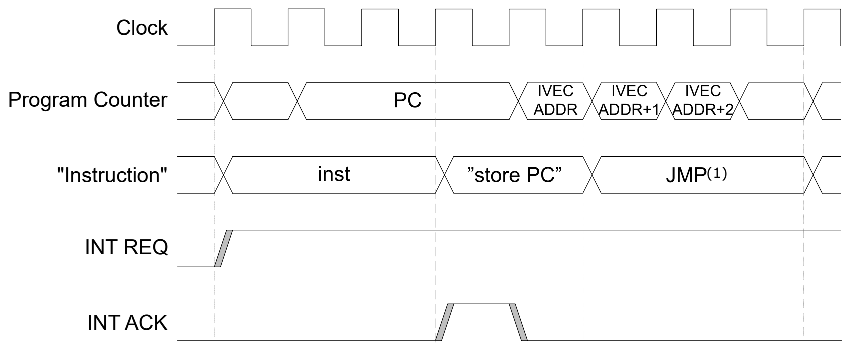
After the Program Counter is pushed on the stack, the program vector for the interrupt is executed. See the following figure.

Figure 15-2. Interrupt Execution of Single-Cycle Instruction



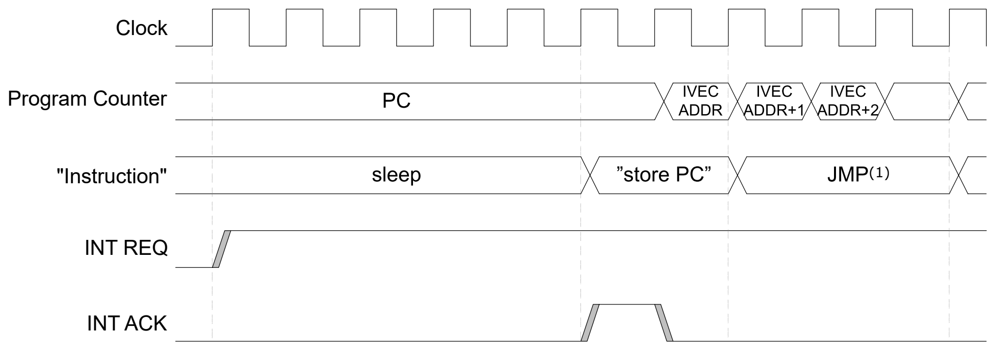
If an interrupt occurs during the execution of a multi-cycle instruction, the instruction is completed before the interrupt is served, as shown in the following figure.

Figure 15-3. Interrupt Execution of Multi-Cycle Instruction



If an interrupt occurs when the device is in a sleep mode, the interrupt execution response time is increased by five clock cycles, as shown in the figure below. Also, the response time is increased by the start-up time from the selected sleep mode.

Figure 15-4. Interrupt Execution From Sleep



A return from an interrupt handling routine takes four to five clock cycles, depending on the size of the Program Counter. During these clock cycles, the Program Counter is popped from the stack, and the Stack Pointer is incremented.

Note:

1. Devices with 8 KB of Flash or less use `RJMP` instead of `JMP`, which takes only two clock cycles.

15.3.2.4 Interrupt Priority

All interrupt vectors are assigned to one of three possible priority levels, as shown in the table below. An interrupt request from a high-priority source will interrupt any ongoing interrupt handler from a normal-priority source. When returning from the high-priority interrupt handler, the execution of the normal-priority interrupt handler will resume.

Table 15-2. Interrupt Priority Levels

Priority	Level	Source
Highest	Non-Maskable Interrupt	Device-dependent and statically assigned
...	Level 1 (high priority)	One vector is optionally user selectable as level 1
Lowest	Level 0 (normal priority)	The remaining interrupt vectors

15.3.2.4.1 Non-Maskable Interrupts

A Non-Maskable Interrupt (NMI) will be executed regardless of the 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.

15.3.2.4.2 High-Priority Interrupt

It is possible to assign one interrupt request to level 1 (high priority) by writing its interrupt vector number to the CPUINT.LVL1VEC register. This interrupt request will have a higher priority than the other (normal priority) interrupt requests. The priority level 1 interrupts will interrupt the level 0 interrupt handlers.

15.3.2.4.3 Normal-Priority Interrupts

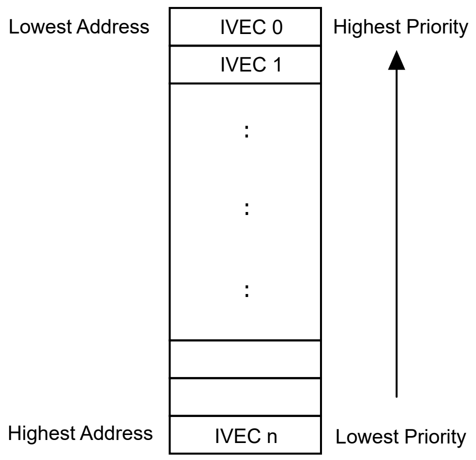
All interrupt vectors other than NMI are assigned to priority level 0 (normal) by default. The user may override this by assigning one of these vectors as a high-priority vector. The device will have many normal-priority vectors, and some of these may be pending at the same time. Two different scheduling schemes are available to choose which of the pending normal-priority interrupts to service first: Static or round robin.

IVEC is the interrupt vector mapping, as listed in the *Peripherals and Architecture* 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.

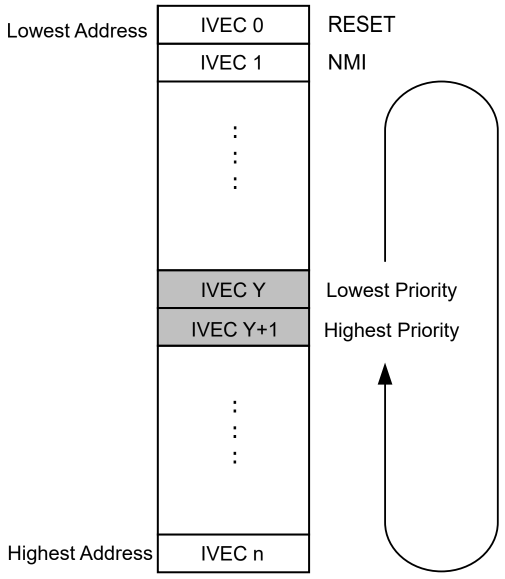
Figure 15-5. Default Static Scheduling



Modified Static Scheduling

The default priority can be changed by writing a vector number to the CPUINT.LVL0PRI register. This vector number will be assigned the lowest priority. The next interrupt vector in the IVEC will have the highest priority among the LVL0 interrupts, as shown in the following figure.

Figure 15-6. Static Scheduling When CPUINT.LVL0PRI Is Different from Zero



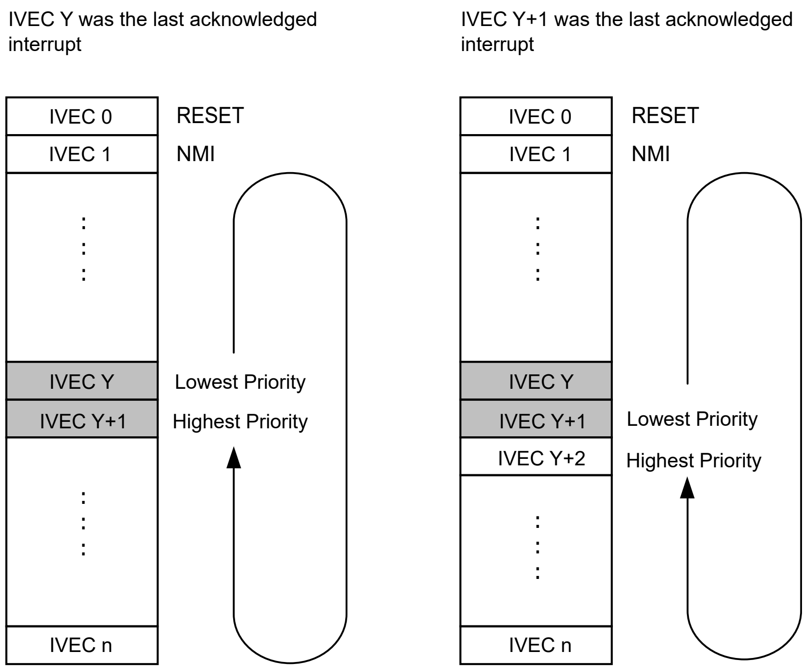
Here, value Y has been written to CPUINT.LVL0PRI so that the interrupt vector Y+1 has the highest priority. Note that, In this case, the priorities will wrap so that the lowest address no longer has the highest priority, not including RESET and NMI, which will always have the highest priority.

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 15-7. Round Robin Scheduling



The round robin scheduling for LVL0 interrupt requests is enabled by writing a '`1`' to the Round Robin Priority Enable (LVL0RR) bit in the Control A (CPUINT.CTRLA) register.

15.3.2.5 Compact Vector Table

The Compact Vector Table (CVT) is a feature to allow 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.

15.3.3 Debug Operation

When using a level 1 priority interrupt, it is important to make sure the Interrupt Service Routine is configured correctly as it may cause the application to be stuck in an interrupt loop with level 1 priority.

By reading the CPUINT STATUS (CPUINT.STATUS) register, it is possible to see if the application has executed the correct `RETI` (interrupt return) instruction. The CPUINT.STATUS register contains state information, which ensures that the CPUINT returns to the correct interrupt level when the `RETI` instruction is executed at the end of an interrupt handler. Returning from an interrupt will return the CPUINT to the state it had before entering the interrupt.

15.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a 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 15-3. CPUINT - Registers Under Configuration Change Protection

Register	Key
The IVSEL and CVT bit fields in CPUINT.CTRLA	IOREG

15.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		IVSEL	CVT					LVL0RR
0x01	STATUS	7:0	NMIEX						LVL1EX	LVL0EX
0x02	LVL0PRI	7:0	LVL0PRI[7:0]							
0x03	LVL1VEC	7:0	LVL1VEC[7:0]							

15.5 Register Description

15.5.1 Control A

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

Bit	7	6	5	4	3	2	1	0
		IVSEL	CVT					LVL0RR
Access		R/W	R/W					R/W
Reset		0	0					0

Bit 6 – IVSEL Interrupt Vector Select

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

15.5.2 Status

Name:	STATUS
Offset:	0x01
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
	NMIEX						LVL1EX	LVL0EX
Access	R						R	R
Reset	0						0	0

Bit 7 – NMIEX Non-Maskable Interrupt Executing
This flag is set if a non-maskable interrupt is executing. The flag is cleared when returning (RETI) from the interrupt handler.

Bit 1 – LVL1EX Level 1 Interrupt Executing
This flag is set when a priority level 1 interrupt is executing, or when the interrupt handler has been interrupted by an NMI. The flag is cleared when returning (RETI) from the interrupt handler.

Bit 0 – LVL0EX Level 0 Interrupt Executing
This flag is set when a priority level 0 interrupt is executing, or when the interrupt handler has been interrupted by a priority level 1 interrupt or an NMI. The flag is cleared when returning (RETI) from the interrupt handler.

15.5.3 Interrupt Priority Level 0

Name:	LVL0PRI
Offset:	0x02
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
	LVL0PRI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – LVL0PRI[7:0] Interrupt Priority Level 0
This register is used to modify the priority of the LVL0 interrupts. See the section Normal-Priority Interrupts for more information.

15.5.4 Interrupt Vector with Priority Level 1

Name:	LVL1VEC
Offset:	0x03
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
	LVL1VEC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – LVL1VEC[7:0] Interrupt Vector with Priority Level 1
This bit field contains the number of the single vector with increased priority level 1 (LVL1). If this bit field has the value `0x00`, no vector has LVL1. Consequently, the LVL1 interrupt is disabled.

16. EVSYS - Event System

16.1 Features

- System for Direct Peripheral-to-Peripheral Signaling
- Peripherals Can Directly Produce, Use, and React to Peripheral Events
- Short and Predictable Response Time
- Up to 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

16.2 Overview

The Event System (EVSYS) enables direct peripheral-to-peripheral signaling. It allows a change in one peripheral (the event generator) to trigger actions in other peripherals (the event users) through event channels, without using the CPU. It is designed to provide a short and predictable response time between peripherals, allowing for autonomous peripheral control and interaction, and for synchronized timing of actions in several peripheral modules. Thus, 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.

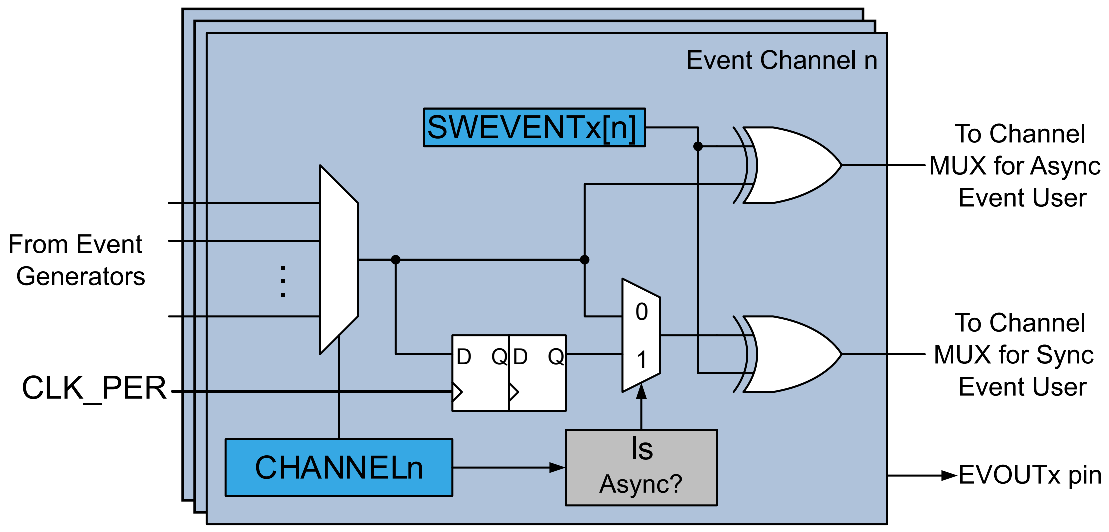
A change of the event generator's state is referred to as an event and usually corresponds to one of the peripheral's interrupt conditions. Events can be forwarded directly to other peripherals using the dedicated event routing network. The routing of each channel is configured in software, including 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.

16.2.1 Block Diagram

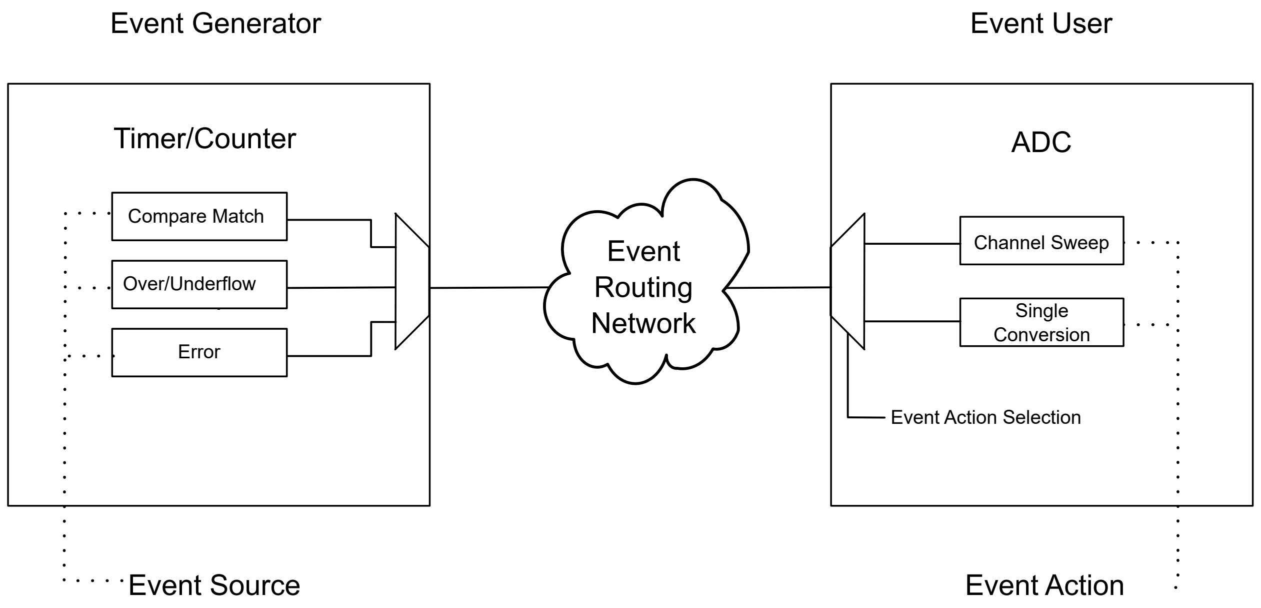
Figure 16-1. EVSYS Block Diagram



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 16-2. Example of Event Source, Generator, User, and Action



16.2.2 Signal Description

Signal	Type	Description
EVOUTx	Digital output	Event output, one output per I/O Port

16.3 Functional Description

16.3.1 Initialization

To utilize events, the Event System, the generating peripheral, and the peripheral(s) using the event must be set up accordingly:

1. Configure the generating peripheral appropriately. For example, if the generating peripheral is a timer, set the prescaling, the Compare register, etc., so that the desired event is generated.
2. Configure the event user peripheral(s) appropriately. For example, if the ADC is the event user, set the ADC prescaler, resolution, conversion time, etc., as desired, and configure the ADC conversion to start at the reception of an event.
3. Configure the Event System to route the desired source. In this case, the Timer/Compare match to the desired event channel. This may, for example, be Channel 0, which is accomplished by writing to the Channel 0 Generator Selection (EVSYS.CHANNEL0) register.
4. Configure the ADC to listen to this channel by writing to the corresponding User x Channel MUX (EVSYS.USERx) register.

16.3.2 Operation

16.3.2.1 Event User Multiplexer Setup

Each event user has one dedicated event user multiplexer selecting which event channel to listen to. The application configures these multiplexers by writing to the corresponding EVSYS.USERx register.

16.3.2.2 Event System Channel

An event channel can be connected to one of the event generators.

The source for each event channel is configured by writing to the respective Channel n Generator Selection (EVSYS.CHANNELn) register.

16.3.2.3 Event Generators

Each event channel has several possible event generators, but only one can be selected at a time. The event generator for a channel is selected by writing to the respective Channel n Generator Selection (EVSYS.CHANNELn) register. By default, the channels are not connected to any event generator. For details on event generation, refer to the documentation of the corresponding peripheral.

A generated event is either synchronous or asynchronous to the device peripheral clock (CLK_PER). Asynchronous events can be generated outside the normal edges of the peripheral clock, making the system respond faster than the selected clock frequency would suggest. Asynchronous events can also be generated while the device is in a sleep mode when the peripheral clock is not running.

Any generated event is classified as either a pulse event or a level event. In both cases, the event can be either synchronous or asynchronous, with properties according to the table below.

Table 16-1. Properties of Generated Events

Event Type	Sync/Async	Description
Pulse	Sync	An event generated from CLK_PER that lasts one clock cycle
	Async	An event generated from a clock other than CLK_PER lasting one clock cycle
Level	Sync	An event generated from CLK_PER that lasts multiple clock cycles
	Async	An event generated without a clock (for example, a pin or a comparator), or an event generated from a clock other than CLK_PER that lasts multiple clock cycles

The properties of both the generated event and the intended event user must be considered in order to ensure reliable and predictable operation.

The table below shows the available event generators for this device family.

Table 16-2. Event Generator Types

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
RTC	OVF	Overflow	Pulse	CLK_RTC	One CLK_RTC period
	CMP	Compare Match			
	EVGEN0	Selectable prescaled RTC event	Level		Prescaled CLK_RTC period
	EVGEN1				
CCL	LUTn	LUT output level	Level	Asynchronous	Depends on CCL configuration
AC0	OUT	Comparator output level	Level	Asynchronous	Given by AC level
ADC0	RESRDY	Result ready	Pulse	CLK_PER	One CLK_PER period
	SAMPRDY	Sample ready			
	WCMP	Window compare			

...........continued

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
PORTx	EVGENn	Pin level	Level	Asynchronous	Given by pin level
USARTn	XCK	USART Baud clock	Level	TXCLK	Minimum two CLK_PER periods
SPI0	SCK	SPI Server clock	Level	CLK_PER	Minimum two CLK_PER periods
TCA0	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			
TCB0	CAPT	CAPT flag set[(1)]	Pulse	CLK_PER	One CLK_PER period
	OVF	Overflow			
TCB1	CAPT	CAPT flag set[(1)]	Pulse	CLK_PER	One CLK_PER period
	OVF	Overflow			
USB	SETUP	SETUP received	Pulse	CLK_PER	One CLK_PER period
	SOF	SOF token received			
	CRC	CRC error			
	UNFOVF	USB under- or overflow			
	RX	Data byte received			
	TX	Data byte transmitted			

Note:
1. The operational mode of the timer decides when the CAPT flag is raised. Refer to the TCB section for details.

16.3.2.4 Event Users

The event channel to listen to is selected by configuring the event user. An event user may require the event signal to be either synchronous or asynchronous to the peripheral clock. An asynchronous event user can respond to events in sleep modes when clocks are not running. Such events can be responded to outside the normal edges of the peripheral clock, making the event user respond faster than the clock frequency would suggest. For details on the requirements of each peripheral, refer to the documentation of the corresponding peripheral.

Most event users implement edge or level detection to trigger actions in the corresponding peripheral based on the incoming event signal. In both cases, a user can either be synchronous, which requires that the incoming event is generated from the peripheral clock (CLK_PER), or asynchronous, if not. Some asynchronous event users do not apply event input detection but use the event signal directly. The different event user properties are described in general in the table below.

Table 16-3. Properties of Event Users

Input Detection	Async/Sync	Description
Edge	Sync	An event user is triggered by an event edge and requires that the incoming event is generated from CLK_PER
	Async	An event user is triggered by an event edge and has asynchronous detection or an internal synchronizer
Level	Sync	An event user is triggered by an event level and requires that the incoming event is generated from CLK_PER
	Async	An event user is triggered by an event level and has asynchronous detection or an internal synchronizer
No detection	Async	An event user will use the event signal directly

The table below shows the available event users for this device family.

Table 16-4. Event User Types

USER Name		Description	Input Detection	Async/Sync
Peripheral	Input			
CCL	LUTnx	LUTn input x or clock signal	No detection	Async
ADC0	START	ADC start on event	Edge	Async
EVSYS	EVOUTx	Forward event signal to pin	No detection	Async
USARTn	IRDA	IrDA mode input	Level	Sync
TCA0	CNTA	Count on positive event edge	Edge	Sync
		Count on any event edge	Edge	
		Count while event signal is high	Level	
		Event level controls count direction	Level	
	CNTB	Event level controls count direction	Level	Sync
		Restart counter on positive event edge	Edge	
		Restart counter on any event edge	Edge	
		Restart counter while event signal is high	Level	
TCBn	CAPT	Time-out check	Edge	Sync
		Input capture on event	Edge	
		Input capture frequency measurement	Edge	
		Input capture pulse-width measurement	Edge	
		Input capture frequency and pulse-width measurement	Edge	
		Single-shot	Edge	Both
	COUNT	Count on event	Edge	Sync

16.3.2.5 Synchronization

Events can be either synchronous or asynchronous to the peripheral clock. Each Event System channel has two subchannels: one asynchronous and one synchronous.

The asynchronous subchannel is identical to the event output from the generator. If the event generator generates a signal asynchronous to the peripheral clock, the signal on the asynchronous subchannel will be asynchronous. If the event generator generates a signal synchronous to the peripheral clock, the signal on the asynchronous subchannel will also be synchronous.

The synchronous subchannel is identical to the event output from the generator, if the event generator generates a signal synchronous to the peripheral clock. If the event generator generates a signal asynchronous to the peripheral clock, this signal is first synchronized before being routed onto the synchronous subchannel. Depending on when it occurs, synchronization will delay the event by two to three clock cycles. The Event System automatically performs this synchronization if an asynchronous generator is selected for an event channel.

16.3.2.6 Software Event

The application can generate a software event. Software events on Channel n are issued by writing a '`1`' to the Software Event Channel Select (CHANNEL[n]) bit in the Software Events (EVSYS.SWEVENTx) register. A software event appears as a pulse on the Event System channel, inverting the current event signal for one clock cycle.

Event users see software events as no different from those produced by event generating peripherals.

16.3.3 Sleep Mode Operation

When configured, the Event System will work in all sleep modes. Software events represent one exception since they require a peripheral clock.

Asynchronous event users are able to respond to an event without their clock running in Standby sleep mode. Synchronous event users require their clock to be running to be able to respond to events. Such users will only work in Idle sleep mode or in Standby sleep mode, if configured to run in Standby mode by setting the RUNSTDBY bit in the appropriate register.

Asynchronous event generators are able to generate an event without their clock running, that is, in Standby sleep mode. Synchronous event generators require their clock to be running to be able to generate events. Such generators will only work in Idle sleep mode or in Standby sleep mode, if configured to run in Standby mode by setting the RUNSTDBY bit in the appropriate register.

16.3.4 Debug Operation

This peripheral is unaffected by entering Debug mode.

16.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	SWEVENTA	7:0	SWEVENTA[7:0]							
0x01	SWEVENTB	7:0	SWEVENTB[7:0]							
0x02 ... 0x0F	Reserved									
0x10	CHANNEL0	7:0	CHANNEL0[7:0]							
0x11	CHANNEL1	7:0	CHANNEL1[7:0]							
0x12	CHANNEL2	7:0	CHANNEL2[7:0]							
0x13	CHANNEL3	7:0	CHANNEL3[7:0]							
0x14	CHANNEL4	7:0	CHANNEL4[7:0]							
0x15	CHANNEL5	7:0	CHANNEL5[7:0]							
0x16 ... 0x1F	Reserved									
0x20	USERCCLLUT0A	7:0	USER[7:0]							
...										
0x33	USERTCB1COUNT	7:0	USER[7:0]							

16.5 Register Description

16.5.1 Software Events

Name:	SWEVENTx
Offset:	0x00 + x*0x01 [x=0..1]
Reset:	0x00
Property:	-

Write bits in this register to create a software event on the corresponding event channels. Bits 0-7 in the EVSYS.SWEVENTA register correspond to event channels 0-7. If the number of available event channels is between eight and 15, these are available in the EVSYS.SWEVENTB register, where bit n corresponds to event channel 8+n.

Refer to the *Peripheral Overview* section for the available number of Event System channels.

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

Bits 7:0 – SWEVENTx[7:0] Software Event Channel Select

Writing a bit in this bit group to '1' will generate a single-pulse event on the corresponding event channel by inverting the signal on the event channel for one peripheral clock cycle.

16.5.2 Channel n Generator Selection

Name: CHANNELn
Offset: 0x10 + n*0x01 [n=0..5]
Reset: 0x00
Property: -

Each channel can be connected to one event generator. Not all generators can be connected to all channels. Refer to the table below to see which generator sources can be routed onto each channel and the generator value to be written to EVSYS.CHANNELn to achieve this routing. Writing the value `0x00` to EVSYS.CHANNELn turns the channel off.

Refer to the *Peripheral Overview* section for the available number of Event System channels.

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

Bits 7:0 – CHANNELn[7:0] Channel Generator Selection
The specific generator name corresponding to each bit group configuration is given by combining *Peripheral* and *Output* from the table below in the following way: PERIPHERAL_OUTPUT.

Table 16-5. Event Generators

GENERATOR			Async/Sync	Description	Channel Availability
Value	Name				
	Peripheral	Output			
0x01	UPDI	SYNCH	Sync	Rising edge of SYNCH character detection	All channels
0x06	RTC	OVF	Async	Counter overflow	All channels
0x07		CMP		Compare match	
0x08		EVGEN0		Selectable prescaled RTC event	
0x09		EVGEN1			
0x10	CCL	LUT0	Async	LUT output level	All channels
0x11		LUT1			
0x12		LUT2			
0x13		LUT3			
0x20	AC0	OUT	Async	Comparator output level	All channels
0x24	ADC0	RESRDY	Sync	Result ready	All channels
0x25		SAMPRDY		Sample ready	
0x26		WCMP		Window compare	
0x40	PORTA(1)	EVGEN0	Async	Pin level (2)	All channels
0x41		EVGEN1			
0x44	PORTC(1)	EVGEN0	Async	Pin level (2)	All channels
0x45		EVGEN1			
0x46	PORTD(1)	EVGEN0	Async	Pin level (2)	All channels
0x47		EVGEN1			
0x4A	PORTF(1)	EVGEN0	Async	Pin level (2)	All channels
0x4B		EVGEN1			
0x60	USART0	XCK	Sync	Clock signal in SPI Host mode and synchronous USART Host mode	All channels
0x61	USART1				
0x68	SPI0	SCK	Sync	SPI server clock signal	All channels

...........continued

GENERATOR			Async/Sync	Description	Channel Availability
Value	Name				
	Peripheral	Output			
0x80	TCA0	OVF_LUNF	Sync	Overflow/Low-byte timer underflow	All channels
0x81		HUNF		High-byte timer underflow	
0x84		CMP0_LCMP0		Compare channel 0 match/Low-byte timer compare channel 0 match	
0x85		CMP1_LCMP1		Compare channel 1 match/Low-byte timer compare channel 1 match	
0x86		CMP2_LCMP2		Compare channel 2 match/Low-byte timer compare channel 2 match	
0xA0	TCB0	CAPT	Sync	CAPT Interrupt flag set(3)	All channels
0xA1		OVF		Counter overflow	
0xA2	TCB1	CAPT	Sync	CAPT Interrupt flag set(3)	All channels
0xA3		OVF		Counter overflow	
0xC0	USB0	SETUP	Sync	SETUP received	All channels
0xC1		SOF		SOF token received	
0xC2		CRC		CRC error	
0xC3		UNFOVF		USB under- or overflow	
0xC4		RX		Data byte received	
0xC5		TX		Data byte transmitted	

Notes:

1. Not all peripheral instances are available for all pin counts. Refer to the *Peripherals and Architecture* section for details.
2. Event from PORT pin will be zero if 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.

16.5.3 User Channel MUX

Name: USER
Offset: 0x20 + n*0x01 [n=0..19]
Reset: 0x00
Property: -

Each event user can be connected to one channel and several users can be connected to the same channel. The following table lists all Event System users with their corresponding user ID number and name. The user name is given by combining USER with Peripheral and Input from the table below in the following way: USERPERIPHERALINPUT.

Table 16-6. Event Users

USER			Async/ Sync	Description
n	Name			
	Peripheral	Input		
0	CCL	LUT0A	Async	CCL LUT0 event input A
1		LUT0B		CCL LUT0 event input B
2		LUT1A		CCL LUT1 event input A
3		LUT1B		CCL LUT1 event input B
4		LUT2A		CCL LUT2 event input A
5		LUT2B		CCL LUT2 event input B
6		LUT3A		CCL LUT3 event input A
7		LUT3B		CCL LUT3 event input B
8	ADC0	START	Async	ADC start on event
9	EVSYS	EVOUTA	Async	EVSYS pin output A
10		EVOUTD		Event output D
11		EVOUTF (1)		Event output F
12	USART0	IRDA	Sync	USART0 IrDA event input
13	USART1	IRDA		USART1 IrDA event input
14	TCA0	CNTA	Sync	Count on event or control count direction
15		CNTB		Restart on event or control count direction
16	TCB0	CAPT	Both(2)	Start, stop, capture, restart or clear counter
17		COUNT	Sync	Count on event
18	TCB1	CAPT	Both(2)	Start, stop, capture, restart or clear counter
19		COUNT	Sync	Count on event

Notes:
1. Not all peripheral instances are available for all pin counts. Refer to the *Peripherals and Architecture* section for details.
2. Depends on Timer/Counter operational mode. Refer to the TCB section for 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
Configures which Event System channel the user is connected to.

Value	Description
0	OFF, no channel is connected to this Event System user

Value	Description
n	The event user is connected to CHANNEL(n-1)

17. PORTMUX - Port Multiplexer

17.1 Overview

The Port Multiplexer (PORTMUX) can either enable or disable the functionality of the pins or change between default and alternative pin positions. Available options are described in detail in the PORTMUX register map and depend on the actual pin and its properties.

Refer to the *I/O Multiplexing and Considerations* section for available pins and functions.

17.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	EVSYSROUTEA	7:0	EXTBRK		EVOUTF		EVOUTD			EVOUTA
0x01	CCLROUTEA	7:0					LUT3	LUT2	LUT1	LUT0
0x02	USARTROUTEA	7:0				USART1[1:0]		USART0[2:0]		
0x03 ... 0x04	Reserved									
0x05	SPIROUTEA	7:0						SPI0[2:0]		
0x06	TWIROUTEA	7:0							TWI0[1:0]	
0x07	TCAROUTEA	7:0						TCA0[2:0]		
0x08	TCBROUTEA	7:0							TCB1	TCB0

17.3 Register Description

17.3.1 EVSYS Pin Position

Name: EVSYSROUTEA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	EXTBRK		EVOUTF		EVOUTD			EVOUTA
Access	R/W		R/W		R/W			R/W
Reset	0		0		0			0

Bit 7 – EXTBRK External Break Pin

This bit selects the pin for the External Break signal.

Value	Name	Description
0x0	DEFAULT	External Break signal on PA6
0x1	ALT1	External Break signal on PF2

Bit 5 – EVOUTF Event Output F

This bit controls the pin position for event output F.

Value	Name	Description
0x0	DEFAULT	EVOUT on PF2
0x1	ALT1	EVOUT on PF7

Bit 3 – EVOUTD Event Output D

This bit controls the pin position for event output D.

Value	Name	Description
0x0	DEFAULT	EVOUT on PD2
0x1	ALT1	EVOUT on PD7

Bit 0 – EVOUTA Event Output A

This bit controls the pin position for event output A.

Value	Name	Description
0x0	DEFAULT	EVOUT on PA2
0x1	ALT1	EVOUT on PA7

17.3.2 CCL LUTn Pin Position

Name:	CCLROUTEA
Offset:	0x01
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
					LUT3	LUT2	LUT1	LUT0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – LUT3 CCL LUT 3 Signals
This bit field controls the pin positions for CCL LUT 3 signals.

Value	Name	Description			
		OUT	IN0	IN1	IN2
0	DEFAULT	PF3	PF0	PF1	PF2
1	-	Reserved	Reserved	Reserved	Reserved

Bit 2 – LUT2 CCL LUT 2 Signals
This bit field controls the pin positions for CCL LUT 2 signals.

Value	Name	Description			
		OUT	IN0	IN1	IN2
0	DEFAULT	PD3	PD0	PD1	PD2
1	ALT1	PD6	PD0	PD1	PD2

Bit 1 – LUT1 CCL LUT 1 Signals
This bit field controls the pin positions for CCL LUT 1 signals.

Value	Name	Description			
		OUT	IN0	IN1	IN2
0	DEFAULT	PC3	-	-	-
1	-	Reserved	Reserved	Reserved	Reserved

Bit 0 – LUT0 CCL LUT 0 Signals
This bit field controls the pin positions for CCL LUT 0 signals.

Value	Name	Description			
		OUT	IN0	IN1	IN2
0	DEFAULT	PA3	PA0	PA1	PA2
1	ALT1	PA6	PA0	PA1	PA2

17.3.3 USARTn Pin Position

Name: USARTROUTEA
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				USART1[1:0]		USART0[2:0]		
Access				R/W	R/W	R/W		
Reset				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	No pin connection			
`0x2`	ALT2	PD6	PD7	-	-
`0x3`	NONE	No pin connection			

Bits 4:2 – 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`	-	Reserved			
`0x5`	NONE	No pin connection			

17.3.4 SPIn Pin Position

Name:	SPIROUTEA
Offset:	0x05
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
						SPI0[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

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
0x4	ALT4	PD4	PD5	PD6	PD7
0x7	NONE	Not connected			Set to '1'
other	-	Reserved			

17.3.5 TWI Pin Positions

Name:	TWIROUTEA
Offset:	0x06
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
							TWI0[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – TWI0[1:0] TWI 0 Signals
This bit field controls the pin positions for TWI 0 signals in both Host and Client mode.

Value	Name	Description	
		SDA	SCL
`0x0`	DEFAULT	PA2	PA3
`0x1`	ALT1	PA2	PA3
`0x2`	Reserved	-	-
`0x3`	ALT3	PA0	PA1

17.3.6 TCAn Pin Position

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

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	Name	Description					
		WO0	WO1	WO2	WO3	WO4	WO5
`0x0`	PORTA	PA0	PA1	PA2	PA3	PA4	PA5
`0x2`	PORTC	-	-	-	PC3	-	-
`0x3`	PORTD	PD0	PD1	PD2	PD3	PD4	PD5
`0x5`	PORTF	PF0	PF1	PF2	PF3	PF4	PF5
`Other`	-	Reserved					

17.3.7 TCBn Pin Position

Name: TCBROUTEA
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							TCB1	TCB0
Access							R/W	R/W
Reset							0	0

Bit 1 – TCB1 TCB1 Output
This bit controls pin position for TCB1 output.

Value	Name	Description
0	DEFAULT	WO on PA3
1	ALT1	WO on PF5

Bit 0 – TCB0 TCB0 Output
This bit controls pin position for TCB0 output.

Value	Name	Description
0	DEFAULT	WO on PA2
1	ALT1	WO on PF4

18. PORT - I/O Pin Configuration

18.1 Features

- General Purpose Input and Output Pins with Individual Configuration:
  - Pull-up
  - Inverted I/O
  - Input voltage threshold
- Interrupts and Events:
  - Sense both edges
  - Sense rising edges
  - Sense falling edges
  - Sense low level
- Optional Slew Rate Control per I/O Port
- Asynchronous Pin Change Sensing That Can Wake the Device From All Sleep Modes
- Efficient and Safe Access to Port Pins
  - Hardware Read-Modify-Write (RMW) through dedicated toggle/clear/set registers
  - Mapping of often-used PORT registers into bit-accessible I/O memory space (virtual ports)

18.2 Overview

The 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.

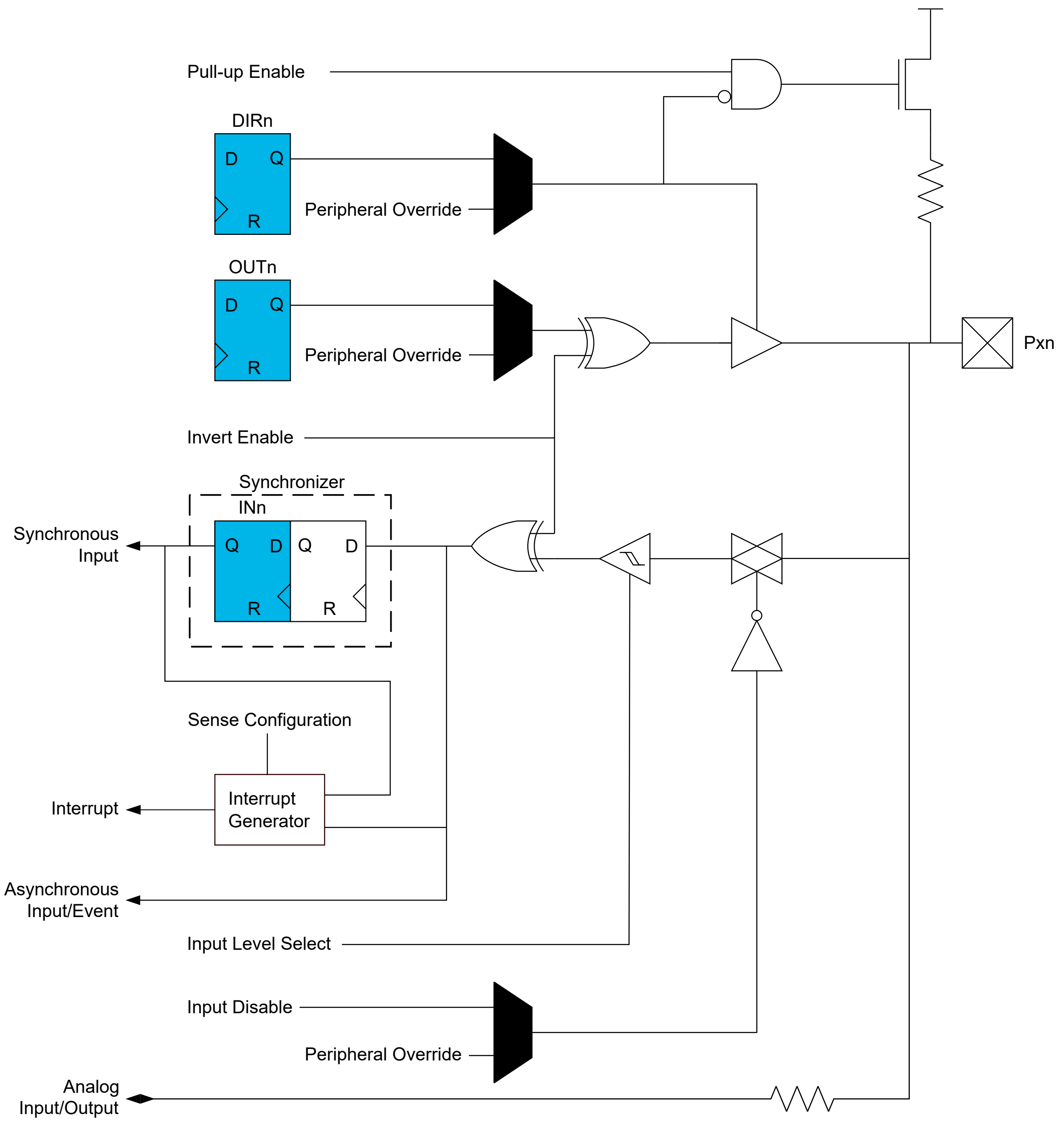
The PORT also supports asynchronous input sensing with interrupts and events for selectable pin change conditions. Asynchronous pin change sensing means that a pin change can trigger an interrupt and wake the device from sleep, including sleep modes where CLK_PER is stopped.

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.

18.2.1 Block Diagram

Figure 18-1. PORT Block Diagram



18.2.2 Signal Description

Signal	Type	Description
Pxn	I/O pin	I/O pin n on PORTx

18.3 Functional Description

18.3.1 Initialization

After Reset, all outputs are tri-stated, and digital input buffers enabled even if there is no clock running.

The following steps are all optional when initializing PORT operation:

- Enable or disable the output driver for pin Pxn by respectively writing '1' to bit n in the PORTx.DIRSET or PORTx.DIRCLR register
- Set the output driver for pin Pxn to high or low level respectively by writing '1' to bit n in the PORTx.OUTSET or PORTx.OUTCLR register
- Read the input of pin Pxn by reading bit n in the PORTx.IN register
- Configure the individual pin configurations and interrupt control for pin Pxn in PORTx.PINnCTRL

Important: For 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.

18.3.2 Operation

18.3.2.1 Basic Functions

Each pin group x has its own set of PORT registers. I/O pin Pxn can be controlled by the registers in PORTx.

To use pin number n as an output, write bit n of the PORTx.DIR register to '1'. This can be done by writing bit n in the PORTx.DIRSET register to '1', which will avoid disturbing the configuration of other pins in that group. The nth bit in the PORTx.OUT register must be written to the desired output value.

Similarly, writing a PORTx.OUTSET bit to '1' will set the corresponding bit in the PORTx.OUT register to '1'. Writing a bit in PORTx.OUTCLR to '1' will clear that bit in PORTx.OUT to '0'. Writing a bit in PORTx.OUTTGL or PORTx.IN to '1' will toggle that bit in PORTx.OUT.

To use pin n as an input, bit n in the PORTx.DIR register must be written to '0' to disable the output driver. This can be done by writing bit n in the PORTx.DIRCLR register to '1', which will avoid disturbing the configuration of other pins in that group. The input value can be read from bit n in the PORTx.IN register as long as the ISC bit is not set to INPUT_DISABLE.

Writing a bit to '1' in PORTx.DIRTGL will toggle that bit in PORTx.DIR and toggle the direction of the corresponding pin.

18.3.2.2 Port Configuration

The Port Control (PORTx.PORTCTRL) register 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.

18.3.2.3 Pin Configuration

The Pin n Control (PORTx.PINnCTRL) register is used to configure inverted I/O, pull-up, and input sensing of a pin. The control register for pin n is at the byte address $\mathrm{PORTx} + 0\mathrm{x}10 + n$.

All input and output on the respective pin n can be inverted by writing a ‘1’ to the Inverted I/O Enable (INVEN) bit in PORTx.PINnCTRL. When INVEN is ‘1’, the PORTx.IN/OUT/OUTSET/OUTTGL registers will have inverted operation for this pin.

Toggling the INVEN bit causes an edge on the pin, which can be detected by all peripherals using this pin and is seen by interrupts or events if enabled.

The Input Level Select (INLVL) bit controls the input voltage threshold for pin n in PORTx.PINnCTRL. A selection of Schmitt trigger thresholds derived from the supply voltage or TTL levels 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 18.3.3. Interrupts for further details.

The digital input buffer for pin n can be disabled by writing the INPUT_DISABLE setting to ISC. This can reduce power consumption and may reduce noise if the pin is used as analog input. While configured to INPUT_DISABLE, bit n in PORTx.IN will not change since the input synchronizer is disabled.

18.3.2.4 Multi-Pin Configuration

The multi-pin configuration function 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 must be applied to each port.

For the multi-pin configuration, port pins can be configured and modified by writing to the following registers.

Table 18-1. Multi-Pin Configuration Registers

Register	Description
PORTx.PINCONFIG	PINnCTRL (ISC, PULLUPEN, INLVL and INVEN) setting to 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;
```

18.3.2.5 Virtual Ports

The Virtual PORT registers map the most frequently used regular PORT registers into the I/O Register space with single-cycle bit access. Access to the Virtual PORT registers has the same outcome as access to the ordinary 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 18-2. 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.

18.3.2.6 Peripheral Override

Peripherals, such as USARTs, ADCs and timers, may be connected to I/O pins. Such peripherals will usually have a primary and, optionally, one or more alternate I/O pin connections, selectable by PORTMUX or a multiplexer inside the peripheral. By configuring and enabling such peripherals, the general purpose I/O pin behavior normally controlled by PORT will be overridden in a peripheral-dependent way. Some peripherals may not override all the PORT registers, leaving the PORT module to control some aspects of the I/O pin operation.

Refer to the description of each peripheral for information on the peripheral override. Any pin in a PORT that is not overridden by a peripheral will continue to operate as a general purpose I/O pin.

18.3.3 Interrupts

Table 18-3. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
PORTx	PORT interrupt	INTn in PORTx.INTFLAGS is raised as configured by the Input/Sense Configuration (ISC) bit in PORTx.PINnCTRL

Each PORT pin n can be configured as an interrupt source. Each interrupt can be individually enabled or disabled by writing to ISC in PORTx.PINnCTRL.

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

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

When setting or changing interrupt settings, 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

18.3.3.1 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.

18.3.4 Events

PORT can generate the following events:

Table 18-4. Event Generators in PORTx

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
PORTx	EVGEN0SEL	Pin input level	Level	Asynchronous	Given by pin level
PORTx	EVGEN1SEL	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.

18.3.5 Sleep Mode Operation

Except for interrupts and input synchronization, all pin configurations are independent of sleep modes. All pins can wake the device from sleep. See the PORT Interrupt section for further details.

Peripherals connected to the PORTs can be affected by sleep modes, described in the respective peripherals' data sheet section.

Important: The PORTs will always use the Peripheral Clock (CLK_PER). Input synchronization will halt when this clock stops.

18.3.6 Debug Operation

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 result during debugging.

18.4 Register Summary - PORTx

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	DIR	7:0	DIR[7:0]							
0x01	DIRSET	7:0	DIRSET[7:0]							
0x02	DIRCLR	7:0	DIRCLR[7:0]							
0x03	DIRTGL	7:0	DIRTGL[7:0]							
0x04	OUT	7:0	OUT[7:0]							
0x05	OUTSET	7:0	OUTSET[7:0]							
0x06	OUTCLR	7:0	OUTCLR[7:0]							
0x07	OUTTGL	7:0	OUTTGL[7:0]							
0x08	IN	7:0	IN[7:0]							
0x09	INTFLAGS	7:0	INT[7:0]							
0x0A	PORTCTRL	7:0								SRL
0x0B	PINCONFIG	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x0C	PINCTRLUPD	7:0	PINCTRLUPD[7:0]							
0x0D	PINCTRLSET	7:0	PINCTRLSET[7:0]							
0x0E	PINCTRLCLR	7:0	PINCTRLCLR[7:0]							
0x0F	Reserved									
0x10	PIN0CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x11	PIN1CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x12	PIN2CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x13	PIN3CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x14	PIN4CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x15	PIN5CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x16	PIN6CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x17	PIN7CTRL	7:0	INVEN	INLVL			PULLUPEN	ISC[2:0]		
0x18	EVGENCTRLA	7:0		EVGEN1SEL[2:0]				EVGEN0SEL[2:0]		

18.5 Register Description - PORTx

18.5.1 Data Direction

Name: DIR
Offset: 0x00
Reset: 0x00
Property: -

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

Bits 7:0 – DIR[7:0] Data Direction
This bit field controls the output driver for each PORTx pin.
This bit field does not control the digital input buffer. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The 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

18.5.2 Data Direction Set

Name:	DIRSET
Offset:	0x01
Reset:	0x00
Property:	-

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

Bits 7:0 – DIRSET[7:0] Data Direction Set

This bit field controls the output driver for each PORTx pin without using a read-modify-write operation.

Writing a '0' to bit n in this bit field has no effect.

Writing a '1' to bit n in this bit field will set the corresponding bit in PORTx.DIR, which will configure pin n (Pxn) as an output pin and enable the output driver.

Reading this bit field will return the value of PORTx.DIR.

18.5.3 Data Direction Clear

Name:	DIRCLR
Offset:	0x02
Reset:	0x00
Property:	-

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

Bits 7:0 – DIRCLR[7:0] Data Direction Clear

This bit field controls the output driver for each PORTx pin without using a read-modify-write operation.

Writing a '`0`' to bit n in this bit field has no effect.

Writing a '`1`' to bit n in this bit field will clear the corresponding bit in PORTx.DIR, which will configure pin n (Pxn) as an input-only pin and disable the output driver.

Reading this bit field will return the value of PORTx.DIR.

18.5.4 Data Direction Toggle

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

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

Bits 7:0 – DIRTGL[7:0] Data Direction Toggle

This bit field controls the output driver for each PORTx pin without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.DIR.
Reading this bit field will return the value of PORTx.DIR.

18.5.5 Output Value

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

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

Bits 7:0 – OUT[7:0] Output Value
This bit field controls the output driver level for each PORTx pin.
This configuration only 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

18.5.6 Output Value Set

Name:	OUTSET
Offset:	0x05
Reset:	0x00
Property:	-

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

Bits 7:0 – OUTSET[7:0] Output Value Set

This bit field controls the output driver level for each PORTx pin without using a read-modify-write operation.

Writing a '`0`' to bit n in this bit field has no effect.

Writing a '`1`' to bit n in this bit field will set the corresponding bit in PORTx.OUT, which will configure the output for pin n (Pxn) to be driven high.

Reading this bit field will return the value of PORTx.OUT.

18.5.7 Output Value Clear

Name: OUTCLR
Offset: 0x06
Reset: 0x00
Property: -

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

Bits 7:0 – OUTCLR[7:0] Output Value Clear

This bit field controls the output driver level for each PORTx pin without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear the corresponding bit in PORTx.OUT, which will configure the output for pin n (Pxn) to be driven low.
Reading this bit field will return the value of PORTx.OUT.

18.5.8 Output Value Toggle

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

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

Bits 7:0 – OUTTGL[7:0] Output Value Toggle

This bit field controls the output driver level for each PORTx pin without using a read-modify-write operation.
Writing a '`0`' to bit n in this bit field has no effect.
Writing a '`1`' to bit n in this bit field will toggle the corresponding bit in PORTx.OUT.
Reading this bit field will return the value of PORTx.OUT.

18.5.9 Input Value

Name:	IN
Offset:	0x08
Reset:	0x00
Property:	-

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

Bits 7:0 – IN[7:0] Input Value

This bit field shows the state of the PORTx pins when the digital input buffer is enabled.

Writing a '0' to bit n in this bit field has no effect.

Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.OUT.

If the digital input buffer is disabled, the input is not sampled, and the bit value will not change. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.

The 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

18.5.10 Interrupt Flags

Name:	INTFLAGS
Offset:	0x09
Reset:	0x00
Property:	-

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

Bits 7:0 – INT[7:0] Pin Interrupt Flag
Pin Interrupt Flag n is cleared by writing a '1' to it.
Pin Interrupt Flag n is set when the change or state of pin n (Pxn) matches the pin's Input/Sense Configuration (ISC) in PORTx.PINnCTRL.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear Pin Interrupt Flag n.

18.5.11 Port Control

Name: PORTCTRL
Offset: 0x0A
Reset: 0x00
Property: -

This register contains the slew rate limit enable bit for this port.

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

Bit 0 – SRL Slew Rate Limit Enable
This bit controls the slew rate limitation for all pins in PORTx.

Value	Description
0	Slew rate limitation is disabled for all pins in PORTx
1	Slew rate limitation is enabled for all pins in PORTx

18.5.12 Multi-Pin Configuration

Name:	PINCONFIG
Offset:	0x0B
Reset:	0x00
Property:	-

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, used for port input reads and interrupt conditions.

Value	Name	Description
0	ST	Schmitt Trigger derived from supply level
1	TTL	TTL Levels

Bit 3 – PULLUPEN Pull-Up Enable
This bit controls whether the internal pull-up of pin n is enabled or not when the pin is configured as input-only.

Value	Description
0	Pull-up disabled
1	Pull-up enabled

Bits 2:0 – ISC[2:0] Input/Sense Configuration
This bit field controls the input and sense configuration of pin n. The sense configuration determines 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.

18.5.13 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.

18.5.14 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.

18.5.15 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.

18.5.16 Pin n Control

Name:	PINnCTRL
Offset:	0x10 + n*0x01 [n=0..7]
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
	INVEN	INLVL			PULLUPEN	ISC[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 7 – INVEN Inverted I/O Enable

This bit controls whether the input and output for pin n are inverted or not.

Value	Description
0	Input and output values are not inverted
1	Input and output values are inverted

Bit 6 – INLVL Input Level Select

This bit controls the input voltage threshold for pin n, used for port input reads and interrupt conditions.

Value	Name	Description
0	ST	Schmitt Trigger derived from supply level
1	TTL	TTL Levels

Bit 3 – PULLUPEN Pull-Up Enable

This bit controls whether the internal pull-up of pin n is enabled or not when the pin is configured as input-only.

Value	Description
0	Pull-up disabled
1	Pull-up enabled

Bits 2:0 – ISC[2:0] Input/Sense Configuration

This bit field controls the input and sense configuration of pin n. The sense configuration determines 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.

18.5.17 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

18.6 Register Summary - VPORTx

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	DIR	7:0	DIR[7:0]							
0x01	OUT	7:0	OUT[7:0]							
0x02	IN	7:0	IN[7:0]							
0x03	INTFLAGS	7:0	INT[7:0]							

18.7 Register Description - VPORTx

18.7.1 Data Direction

Name:	DIR
Offset:	0x00
Reset:	0x00
Property:	-

Access to the Virtual PORT registers has the same outcome as access to the ordinary 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

18.7.2 Output Value

Name: OUT
Offset: 0x01
Reset: 0x00
Property: -

Access to the Virtual PORT registers has the same outcome as access to the ordinary 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

18.7.3 Input Value

Name:	IN
Offset:	0x02
Reset:	0x00
Property:	-

Access to the Virtual PORT registers has the same outcome as access to the ordinary 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

18.7.4 Interrupt Flags

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

Access to the Virtual PORT registers has the same outcome as access to the ordinary 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.

19. BOD - Brown-out Detector

19.1 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

19.2 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.

19.2.1 Block Diagram

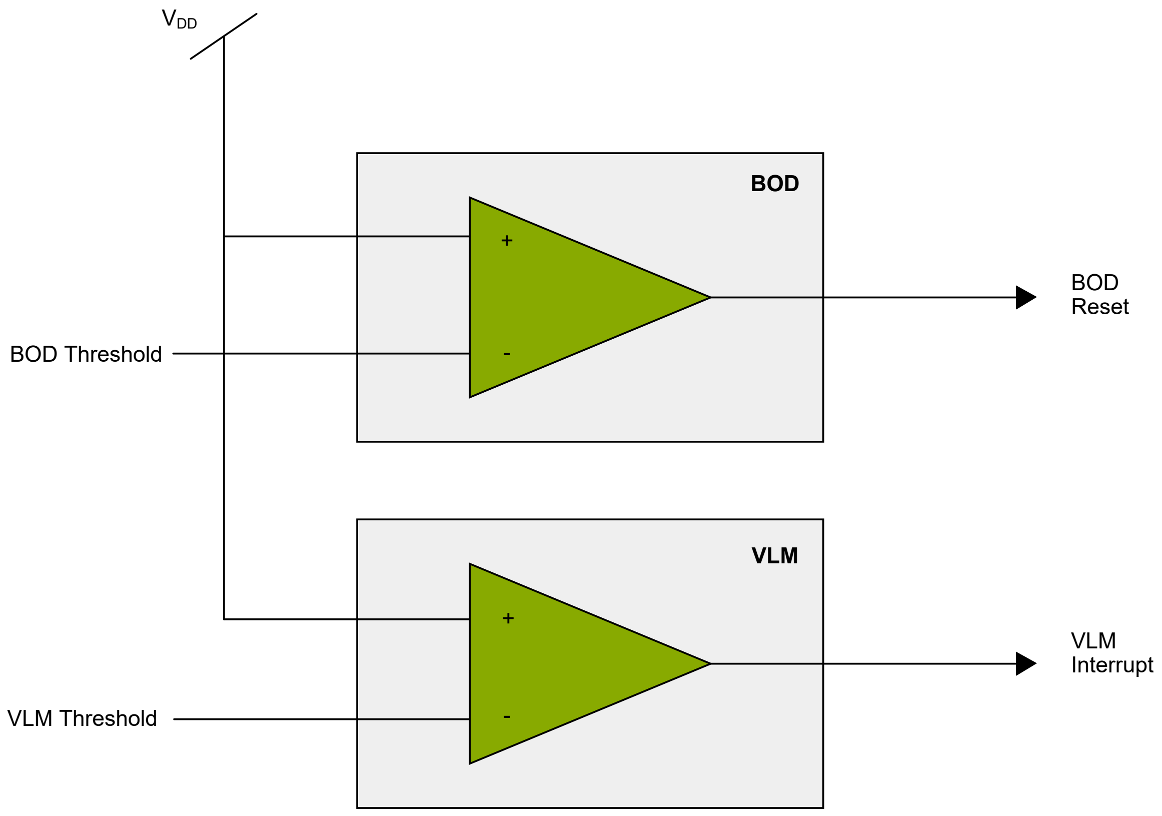


Figure 19-1. BOD Block Diagram

19.3 Functional Description

19.3.1 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.

19.3.2 Interrupts

Table 19-1. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
VLM	Voltage Level Monitor	Supply voltage crossing the VLM threshold as configured by the VLM Configuration (VLMCFG) bit field in the Interrupt Control (BOD.INTCTRL) register

The VLM interrupt will not be executed if the CPU is halted in Debug mode.

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

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

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

19.3.3 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.

19.3.4 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 19-2. Registers Under Configuration Change Protection

Register	Key
The SLEEP and SAMPFREQ bits in the BOD.CTRLA register	IOREG

19.4 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

19.5 Register Description

19.5.1 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	Description
0x0	Sample frequency is 128 Hz
0x1	Sample frequency is 32 Hz

Bits 3:2 – ACTIVE[1:0] Active
These bits select the BOD operation mode when the device is in Active mode or Idle sleep mode.
The Reset value is loaded from the ACTIVE bits in FUSE.BODCFG.
These bits are not under Configuration Change Protection (CCP).

Value	Name	Description
0x0	DIS	Disabled
0x1	ENABLED	Enabled in Continuous mode
0x2	SAMPLE	Enabled in Sampled mode
0x3	ENWAKE	Enabled in Continuous mode. Execution is halted at wake-up until BOD is running.

Bits 1:0 – SLEEP[1:0] Sleep
These bits select the BOD operation mode when the device is in Standby or Power-Down sleep mode. The Reset value is loaded from the SLEEP bits in FUSE.BODCFG.

Value	Name	Description
0x0	DIS	Disabled
0x1	ENABLED	Enabled in Continuous mode
0x2	SAMPLED	Enabled in Sampled mode
0x3	-	Reserved

19.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: Loaded from fuse
Property: -

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

Bits 2:0 – LVL[2:0] BOD Level
This bit field controls the BOD threshold level.
The Reset value is loaded from the BOD Level (LVL) bits in the BOD Configuration Fuse (FUSE.BODCFG).

Value	Name	Typical Values
`0x0`	BODLEVEL0	1.90V
`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.

19.5.3 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

19.5.4 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.

19.5.5 VLM Interrupt Flags

Name:	INTFLAGS
Offset:	0x0A
Reset:	0x00
Property:	-

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

Bit 0 – VLMIF VLM Interrupt Flag

This flag is set when a trigger from the VLM is given, as configured by the VLMCFG bit in the BOD.INTCTRL register. The flag is only updated when the BOD is enabled.

19.5.6 VLM Status

Name:	STATUS
Offset:	0x0B
Reset:	0x00
Property:	-

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

Bit 0 – VLMS VLM Status

This bit is only valid when the BOD is enabled.

Value	Name	Description
0	ABOVE	The voltage is above the VLM threshold level
1	BELOW	The voltage is below the VLM threshold level

20. VREF - Voltage Reference

20.1 Features

- One Programmable Voltage Reference Source Shared Between All Analog Comparators (ACs)
- The Reference Source Supports the Following Voltages:
  - 1.024V
  - 2.048V
  - 4.096V
  - 2.500V
  - VDD
  - VREFA

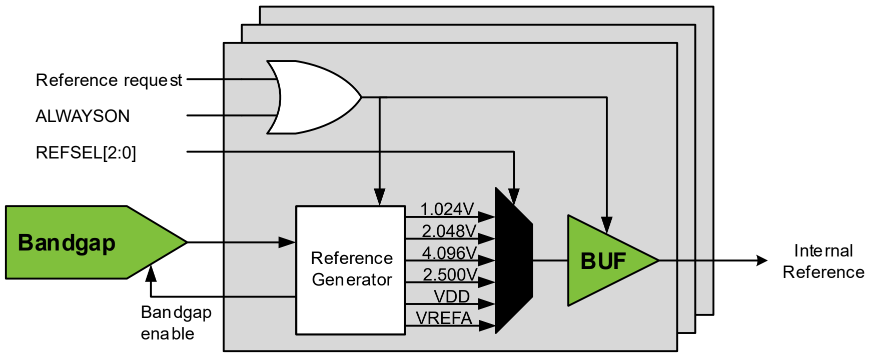
20.2 Overview

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 ACs by writing to the appropriate registers in the VREF peripheral.
Note: Registers within the ADC peripheral control the ADC voltage reference.

A voltage reference source is automatically enabled 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 ALWAYSON bit in the Analog Comparator Reference (ACREF) register. This will decrease the start-up time at the cost of increased power consumption.

20.2.1 Block Diagram

Figure 20-1. VREF Block Diagram



20.3 Functional Description

20.3.1 Initialization

The default configuration will enable the respective source when any AC requests 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 Analog Comparators (ACREF) register.

20.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	ACREF	7:0	ALWAYSON					REFSEL[2:0]		

20.5 Register Description

20.5.1 Analog Comparator Reference

Name:	ACREF
Offset:	0x00
Reset:	0x00
Property:	-

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

Bit 7 – ALWAYSON Reference Always On

This bit controls whether the ACs reference is always on or not.

Value	Description
0	The reference is automatically enabled when needed
1	The reference is always on

Bits 2:0 – REFSEL[2:0] Reference Select

This bit field controls the reference voltage level for ACs.

Value	Name	Description
0x0	1V024	Internal 1.024V reference[(1)]
0x1	2V048	Internal 2.048V reference[(1)]
0x2	4V096	Internal 4.096V reference[(1)]
0x3	2V500	Internal 2.500V reference[(1)]
0x4	-	Reserved
0x5	VDD	VDD as reference
0x6	VREFA	External reference from the VREFA pin
0x7	-	Reserved

Note:

1. The values given for internal references are only typical. Refer to the *Electrical Characteristics* section for further details.

21. WDT - Watchdog Timer

21.1 Features

- Issues a System Reset if the Watchdog Timer Is Not Cleared Before Its Time-Out Period
- Operates Asynchronously from the Peripheral Clock Using an Independent Oscillator
- Uses the 1.024 kHz Output of the 32.768 kHz Ultra-Low Power Oscillator (OSC32K)
- 11 Selectable Time-Out Periods, from 8 ms to 8s
- Two Operation Modes:
  - Normal mode
  - Window mode
- Configuration Lock to Prevent Unwanted Changes

21.2 Overview

The Watchdog Timer (WDT) is a system function for monitoring the correct program operation. When enabled, the WDT is a constantly running timer with a configurable time-out period. If the WDT is not reset within the time-out period, it will issue a system Reset, which allows the system to recover from situations such as runaway or deadlocked code. The WDT is reset by executing the `WDR` (Watchdog Timer Reset) instruction from software.

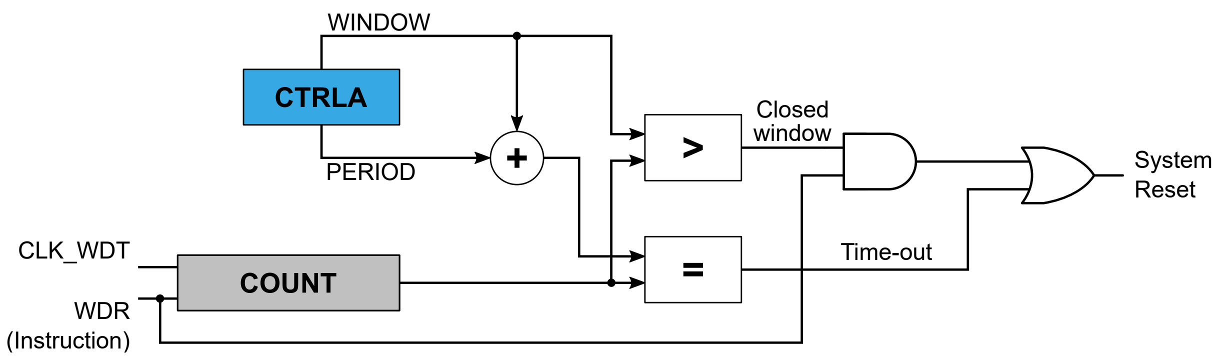
In addition to the Normal mode as described above, the WDT has a Window mode. The Window mode defines a time slot or “window” inside the time-out period during which the WDT must be reset. If the WDT is reset outside this window, either too early or too late, a system Reset will be issued. Compared to the Normal mode, the Window mode can catch situations where a code error causes constant `WDR` execution.

When enabled, the WDT will run in Active mode and all sleep modes. Since it is asynchronous (running from a CPU-independent clock source), it will continue to operate and be able to issue a system Reset, even if the main clock fails.

The WDT has a Configuration Change Protection (CCP) mechanism and a lock functionality, ensuring the WDT settings cannot be changed by accident.

21.2.1 Block Diagram

Figure 21-1. WDT Block Diagram



21.3 Functional Description

21.3.1 Initialization

1. The WDT is enabled when a non-zero value is written to the Period (PERIOD) bit field in the Control A (WDT.CTRLA) register.

2. Optional: Write a non-zero value to the Window (WINDOW) bit field in WDT.CTRLA to enable the Window mode operation.

All bits in the Control A register and the Lock (LOCK) bit in the Status (WDT.STATUS) register are write-protected by the Configuration Change Protection (CCP) mechanism.

A fuse (FUSE.WDTCFG) defines the Reset value of the WDT.CTRLA register. If the value of the PERIOD bit field in the FUSE.WDTCFG fuse is different than zero, the WDT is enabled, and the LOCK bit in the WDT.STATUS register is set at boot time.

21.3.2 Clocks

A 1.024 kHz clock (CLK_WDT) is sourced from the internal Ultra-Low Power Oscillator, OSC32K. Due to the ultra-low power design, the oscillator is less accurate than other oscillators featured in the device, and hence, the exact time-out period may vary from device to device. This variation must be considered when designing software that uses the WDT to ensure that the time-out periods used are valid for all devices. Refer to the *Electrical Characteristics* section for more specific information.

The WDT clock (CLK_WDT) is asynchronous to the peripheral clock. Due to this asynchronicity, writing to the WDT Control A (WDT.CTRLA) register will require synchronization between the clock domains. Refer to 21.3.6. Synchronization for further details.

21.3.3 Operation

21.3.3.1 Normal Mode

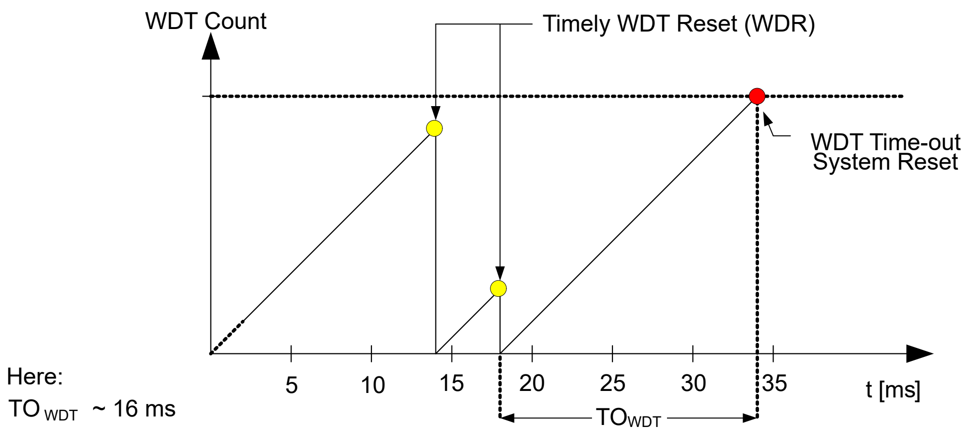
In the Normal mode operation, a single time-out period is set for the WDT. If the WDT is not reset from software using the `WDR` instruction during the defined time-out period, the WDT will issue a system Reset.

Each time the WDT is reset by software using the `WDR` instruction, a new WDT time-out period starts.

There are 11 possible WDT time-out periods (TO_{WDT}), selectable from 8 ms to 8s by writing to the Period (PERIOD) bit field in the Control A (WDT.CTRLA) register.

The figure below shows a typical timing scheme for the WDT operating in Normal mode.

Figure 21-2. Normal Mode Operation



The Normal mode is enabled as long as the Window (WINDOW) bit field in the WDT.CTRLA register is ‘`0x0`’.

21.3.3.2 Window Mode

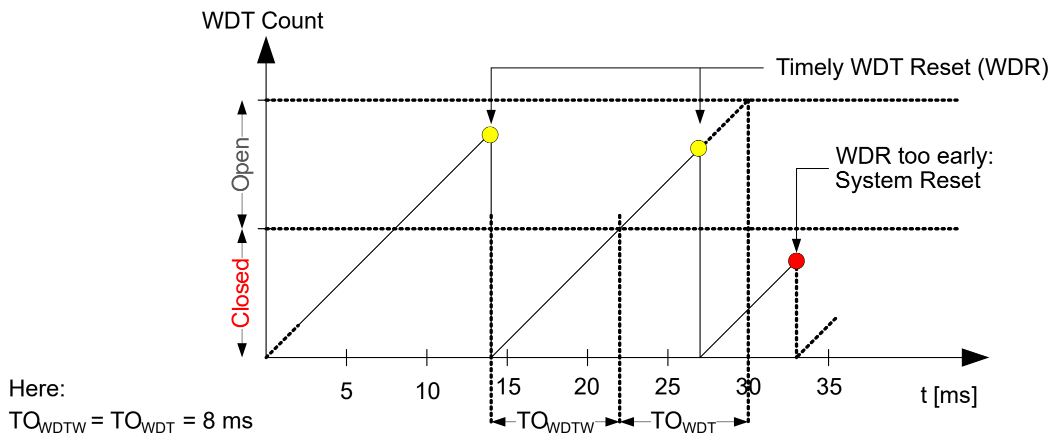
In Window mode operation, the WDT uses two different time-out periods: A closed window time-out period (TO_{WDTW}) and an open window time-out period (TO_{WDT}):

- TO_{WDTW} defines a duration from 8 ms to 8s, where the WDT should not be reset. If the WDT is reset during this period, the WDT will issue a system Reset.
- TO_{WDT}, which is also 8 ms to 8s, defines the duration of the open period during which the WDT can (and needs to) be reset. The open period will always follow the closed period, so the total duration of the time-out period is the sum of the closed window and the open window time-out periods.

When enabling the Window mode or going out of the Debug mode, the window is activated after the first `WDR` instruction.

The figure below shows a typical timing scheme for the WDT operating in Window mode.

Figure 21-3. Window Mode Operation



The Window mode is enabled by writing a non-zero value to the WINDOW bit field in the Control A (WDT.CTRLA) register and disabled by writing it to `0x0`.

21.3.3.3 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 21.3.7. Configuration Change Protection for further details.
- Locking the configuration by writing a ‘`1`’ to the Lock (LOCK) bit in the Status (WDT.STATUS) register. When this bit is ‘`1`’, the Control A (WDT.CTRLA) register cannot be changed. The LOCK bit can only be written to ‘`1`’ in software, while the device needs to be in Debug mode to be able to write it to ‘`0`’. Consequently, the WDT cannot be disabled from the software.

Note: The WDT configuration is loaded from fuses after Reset. If the PERIOD bit field is set to a non-zero value, the LOCK bit is automatically set in WDT.STATUS.

21.3.4 Sleep Mode Operation

The WDT will continue to operate in any sleep mode where the source clock is active.

21.3.5 Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in Debugging mode will halt the normal operation of the peripheral.

When halting the CPU in Debug mode, the WDT counter is reset.

When starting the CPU and when the WDT is operating in Window mode, the first closed window time-out period will be disabled, and a Normal mode time-out period is executed.

21.3.6 Synchronization

The Control A (WDT.CTRLA) register is synchronized when written, due to the asynchronicity between the WDT clock domain and the peripheral clock domain. The Synchronization Busy (SYNCBUSY) flag in the STATUS (WDT.STATUS) register indicates if there is an ongoing synchronization.

Writing to WDT.CTRLA while SYNCBUSY = `1` is not allowed.

The following bit fields are synchronized when written:

- The Period (PERIOD) bit field in Control A (WDT.CTRLA) register
- The Window (WINDOW) bit field in Control A (WDT.CTRLA) register

The `WDR` instruction will need two to three cycles of the WDT clock to be synchronized.

21.3.7 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 21-1. WDT - Registers Under Configuration Change Protection

Register	Key
WDT.CTRLA	IOREG
LOCK bit in WDT.STATUS	IOREG

21.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	WINDOW[3:0]				PERIOD[3:0]			
0x01	STATUS	7:0	LOCK							SYNCBUSY

21.5 Register Description

21.5.1 Control A

Name: CTRLA
Offset: 0x00
Reset: From FUSE.WDTCFG
Property: Configuration Change Protection

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

Bits 7:4 – WINDOW[3:0] Window
Writing a non-zero value to these bits enables the Window mode and selects the duration of the closed period accordingly.
The bits are optionally lock-protected:
- If the LOCK bit in WDT.STATUS is '1', all bits are change-protected (Access = R)
- If the LOCK bit in WDT.STATUS is '0', all bits can be changed (Access = R/W)

Value	Name	Description
0x0	OFF	-
0x1	8CLK	7.8125 ms
0x2	16CLK	15.625 ms
0x3	32CLK	31.25 ms
0x4	64CLK	62.5 ms
0x5	128CLK	0.125s
0x6	256CLK	0.250s
0x7	512CLK	0.500s
0x8	1KCLK	1.0s
0x9	2KCLK	2.0s
0xA	4KCLK	4.0s
0xB	8KCLK	8.0s
Other	-	Reserved

Note: Refer to the *Electrical Characteristics* section for specific information regarding the 32.768 kHz Ultra-Low Power Oscillator (OSC32K) accuracy.

Bits 3:0 – PERIOD[3:0] Period
Writing a non-zero value to this bit enables the WDT and selects the time-out period in the Normal mode accordingly. In the Window mode, these bits select the duration of the open window.
The bits are optionally lock-protected:
- If the LOCK bit in WDT.STATUS is '1', all bits are change-protected (Access = R)
- If the LOCK bit in WDT.STATUS is '0', all bits can be changed (Access = R/W)

Value	Name	Description
0x0	OFF	-
0x1	8CLK	7.8125 ms
0x2	16CLK	15.625 ms
0x3	32CLK	31.25 ms
0x4	64CLK	62.5 ms
0x5	128CLK	0.125s
0x6	256CLK	0.250s
0x7	512CLK	0.500s
0x8	1KCLK	1.0s

Value	Name	Description
`0x9`	2KCLK	2.0s
`0xA`	4KCLK	4.0s
`0xB`	8KCLK	8.0s
`Other`	-	Reserved

Note: Refer to the *Electrical Characteristics* section for specific information regarding the 32.768 kHz Ultra-Low Power Oscillator (OSC32K) accuracy.

21.5.2 Status

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

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

Bit 7 – LOCK Lock

Writing this bit to '1' write-protects the WDT.CTRLA register.
It is only possible to write this bit to '1'. This bit can be cleared in Debug mode only.
If the PERIOD value in the WDTCFG fuse is different from zero, the lock will be automatically set.
This bit is under CCP.

Bit 0 – SYNCBUSY Synchronization Busy

This bit is set after writing to the WDT.CTRLA register, while the data is being synchronized from the peripheral clock domain to the WDT clock domain.
This bit is cleared after finishing the synchronization.
This bit is not under CCP.

22. RTC - Real-Time Counter

22.1 Features

- 16-Bit Resolution
- Selectable Clock Sources
- Programmable 15-Bit Clock Prescaling
- One Compare Register
- One Period Register
- Clear Timer on Period Overflow
- Optional Interrupt/Event on Overflow and Compare Match
- Periodic Interrupt and Event
- Crystal Error Correction

22.2 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 32.768 kHz Internal Oscillator (OSC32K), or the OSC32K divided by 32.

The RTC peripheral includes a 15-bit programmable prescaler that can scale down the reference clock before it reaches the counter. A wide range of resolutions and time-out periods can be configured for the RTC. With a 32.768 kHz clock source, the maximum resolution is 30.5 μs, and time-out periods can be up to two seconds. With a resolution of 1s, the maximum time-out period is more than 18 hours (65536 seconds).

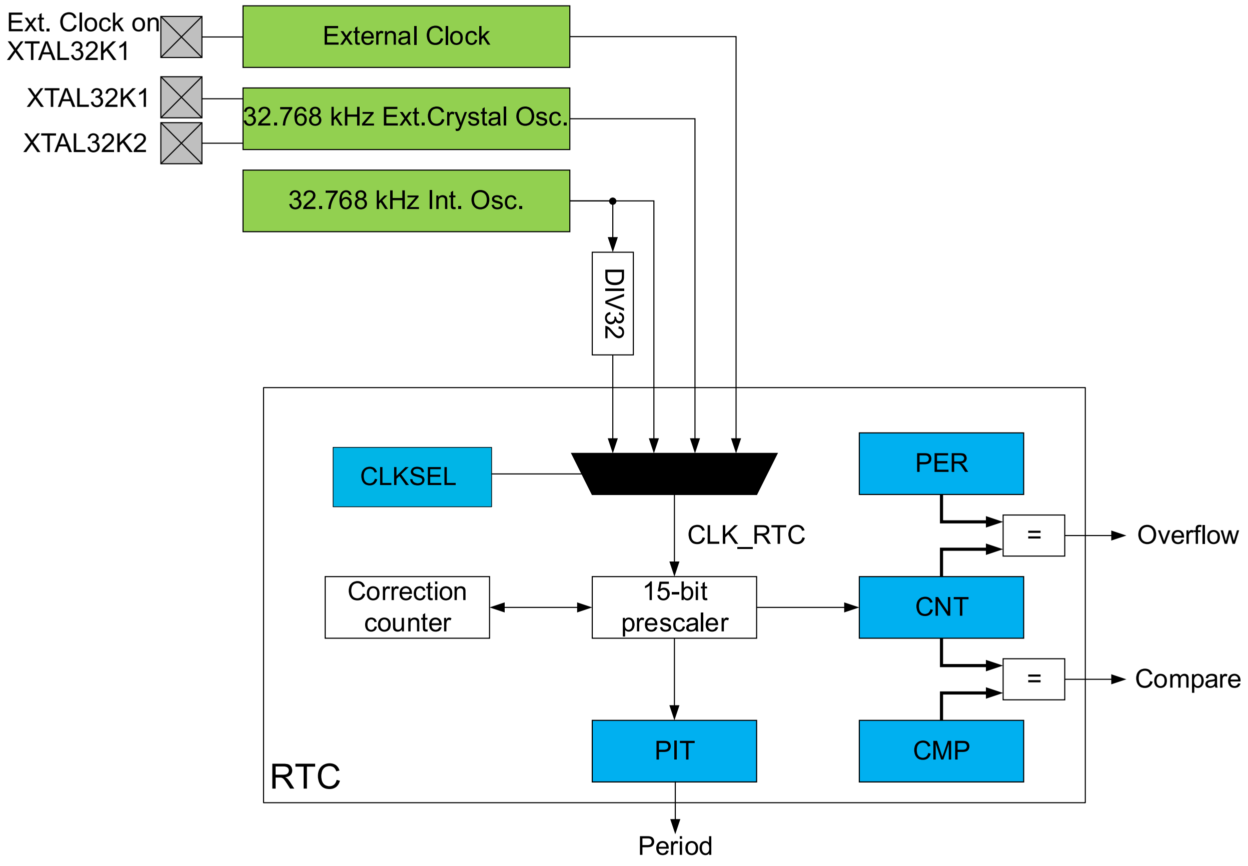
The RTC also supports crystal error correction when operated using external crystal selection. An externally calibrated value will be used for correction. The software can adjust the RTC with an accuracy of ±1 PPM, and the maximum adjustment is ±127 PPM. The RTC correction operation will either speed up (by skipping count) or slow down (by adding extra count) the prescaler to account for the crystal error.

PIT - Periodic Interrupt Timer

The PIT uses the same clock source (CLK_RTC) as the RTC function and can generate an interrupt request or a level event on every nth clock period. The n can be selected from {4, 8, 16,... 32768} for interrupts, and from {64, 128, 256,... 8192} for events.

22.2.1 RTC Block Diagram

Figure 22-1. Block Diagram



22.3 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.

22.4 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.

22.4.1 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.

22.4.1.1 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.

22.4.1.2 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.

22.4.2 Operation - RTC

22.4.2.1 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'.

22.5 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.

22.5.1 Initialization

To operate the PIT, follow these steps:

1. Configure the RTC clock CLK_RTC as described in section 22.4.1.1. Configure the Clock CLK_RTC.
2. Enable the interrupt by writing a '1' to the Periodic Interrupt (PI) bit in the PIT Interrupt Control (RTC.PITINTCTRL) register.
3. 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.

22.5.2 Operation - PIT

22.5.2.1 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'.

22.5.2.2 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

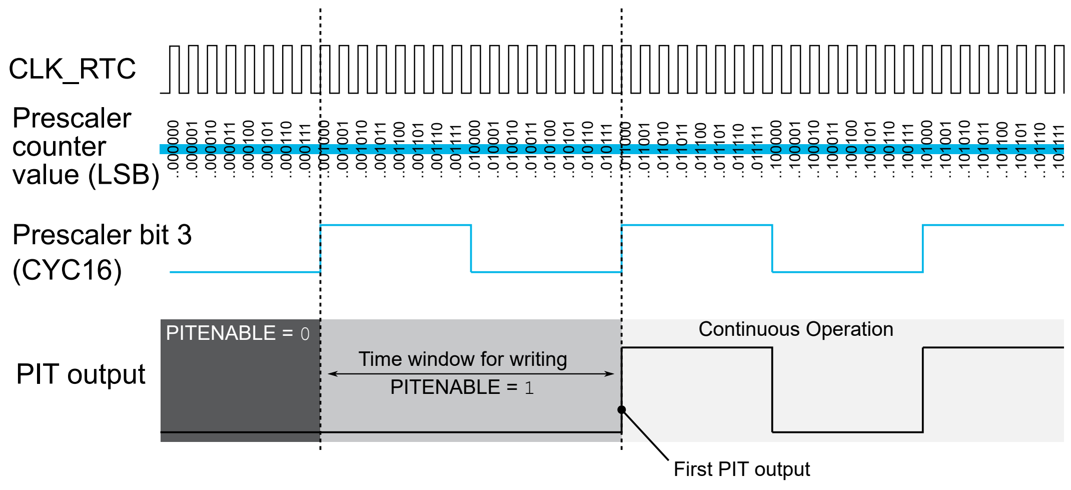
After the first interrupt, the PIT will continue toggling every ½ PIT period resulting in a full PIT period signal.

Example 22-1. 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 22-2. Timing Between PIT Enable and First Interrupt



22.6 Crystal Error Correction

The prescaler for the RTC and PIT can do internal frequency correction of the crystal clock by using the PPM error value from the Crystal Frequency Calibration (CALIB) register when the Frequency Correction Enable (CORREN) bit in the RTC.CTRLA register is '1'.

The CALIB register must be written by the user, based on the information about the frequency error. Perform the correction operation by adding or removing some cycles equal to the value given in the Error Correction Value (ERROR) bit field in the CALIB register spread throughout a million-cycle interval.

The RTC count value available through the Count (RTC.CNT) registers or in the PIT intervals will reflect the clock correction.

If disabling the correction feature, an ongoing correction cycle will be completed before the function is disabled.

Note: If using this feature with a negative correction, the minimum prescaler configuration is DIV2.

22.7 Events

The RTC can generate the events described in the following table:

Table 22-1. Event Generators in RTC

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
RTC	OVF	Overflow	Pulse	CLK_RTC	One CLK_RTC period
	COMP	Compare match			
	EVGEN0	Selactable prescaled CLK_RTC	Level		Defined by selected prescaler event
	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.

22.8 Interrupts

Table 22-2. 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.

22.9 Sleep Mode Operation

The RTC will continue to operate in Idle sleep mode. It will run in Standby sleep mode if the Run in Standby (RUNSTDBY) bit in RTC.CTRLA is set.

The PIT will continue to operate in any sleep mode.

22.10 Synchronization

Both the RTC and the PIT are asynchronous, operating from a different clock source (CLK_RTC) independently of the peripheral clock (CLK_PER). For Control and Count register updates, it will take some RTC and/or peripheral clock cycles before an updated register value is available in a register or until a configuration change affects the RTC or PIT, respectively. This synchronization time is described for each register in the *Register Description* section.

For some RTC registers, a Synchronization Busy (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.

22.11 Debug Operation

If the Debug Run (DBGRUN) bit in the Debug Control (RTC.DBGCTRL) register is '1', the RTC will continue normal operation. If DBGRUN is '0' and the CPU is halted, the RTC will halt the operation and ignore any incoming events.

If the Debug Run (DBGRUN) bit in the Periodic Interrupt Timer Debug Control (RTC.PITDBGCTRL) register is '1', the PIT will continue normal operation. If DBGRUN is '0' in the Debug mode and the CPU is halted, the PIT output will be low. When the PIT output is high at the time, a new positive edge occurs to set the interrupt flag when restarting from a break. The result is an additional PIT interrupt that does not happen during normal operation. If the PIT output is low at the break, the PIT will resume low without additional interrupt.

22.12 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY	PRESCALER[3:0]				CORREN		RTCEN
0x01	STATUS	7:0					CMPBUSY	PERBUSY	CNTBUSY	CTRLABUSY
0x02	INTCTRL	7:0							CMP	OVF
0x03	INTFLAGS	7:0							CMP	OVF
0x04	TEMP	7:0	TEMP[7:0]							
0x05	DBGCTRL	7:0								DBGRUN
0x06	CALIB	7:0	SIGN	ERROR[6:0]						
0x07	CLKSEL	7:0							CLKSEL[1:0]	
0x08	CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x0A	PER	7:0	PER[7:0]							
		15:8	PER[15:8]							
0x0C	CMP	7:0	CMP[7:0]							
		15:8	CMP[15:8]							
0x0E ... 0x0F	Reserved									
0x10	PITCTRLA	7:0		PERIOD[3:0]						PITEN
0x11	PITSTATUS	7:0								CTRLBUSY
0x12	PITINTCTRL	7:0								PI
0x13	PITINTFLAGS	7:0								PI
0x14	Reserved									
0x15	PITDBGCTRL	7:0								DBGRUN
0x16	PITEVGENCTRLA	7:0	EVGEN1SEL[3:0]				EVGEN0SEL[3:0]			

22.13 Register Description

22.13.1 Control A

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

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	PRESCALER[3:0]				CORREN		RTCEN
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit 7 – RUNSTDBY Run in Standby

Value	Description
0	RTC disabled in Standby sleep mode
1	RTC enabled in Standby sleep mode

Bits 6:3 – PRESCALER[3:0] Prescaler

These bits define the prescaling of the CLK_RTC clock signal.

Value	Name	Description
0x0	DIV1	RTC clock/1 (no prescaling)
0x1	DIV2	RTC clock/2
0x2	DIV4	RTC clock/4
0x3	DIV8	RTC clock/8
0x4	DIV16	RTC clock/16
0x5	DIV32	RTC clock/32
0x6	DIV64	RTC clock/64
0x7	DIV128	RTC clock/128
0x8	DIV256	RTC clock/256
0x9	DIV512	RTC clock/512
0xA	DIV1024	RTC clock/1024
0xB	DIV2048	RTC clock/2048
0xC	DIV4096	RTC clock/4096
0xD	DIV8192	RTC clock/8192
0xE	DIV16384	RTC clock/16384
0xF	DIV32768	RTC clock/32768

Bit 2 – CORREN Frequency Correction Enable

Value	Description
0	Frequency correction is disabled
1	Frequency correction is enabled

Bit 0 – RTCEN RTC Peripheral Enable

Value	Description
0	RTC peripheral is disabled
1	RTC peripheral is enabled

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. The application software must check that the CTRLABUSY flag in the RTC.STATUS register is cleared before writing to this register.

22.13.2 Status

Name:	STATUS
Offset:	0x01
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
					CMPBUSY	PERBUSY	CNTBUSY	CTRLABUSY
Access					R	R	R	R
Reset					0	0	0	0

Bit 3 – CMPBUSY Compare Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Compare (RTC.CMP) register in the RTC clock domain.

Bit 2 – PERBUSY Period Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Period (RTC.PER) register in the RTC clock domain.

Bit 1 – CNTBUSY Counter Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Count (RTC.CNT) register in the RTC clock domain.

Bit 0 – CTRLABUSY Control A Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Control A (RTC.CTRLA) register in the RTC clock domain.

22.13.3 Interrupt Control

Name:	INTCTRL
Offset:	0x02
Reset:	0x00
Property:	-

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

Bit 1 – CMP Compare Match Interrupt Enable

Enable interrupt-on-compare match (that is, when the value from the Count (RTC.CNT) register matches the value from the Compare (RTC.CMP) register).

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

Bit 0 – OVF Overflow Interrupt Enable

Enable interrupt-on-counter overflow (that is, when the value from the Count (RTC.CNT) register matched the value from the Period (RTC.PER) register and wraps around to zero).

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

22.13.4 Interrupt Flag

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

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

Bit 1 – CMP Compare Match Interrupt Flag

This flag is set when the value from the Count (RTC.CNT) register matches the value from the Compare (RTC.CMP) register.
Writing a '1' to this bit clears the flag.

Bit 0 – OVF Overflow Interrupt Flag

This flag is set when the value from the Count (RTC.CNT) register has reached the value from the Period (RTC.PER) register and wrapped to zero.
Writing a '1' to this bit clears the flag.

22.13.5 Temporary

Name:	TEMP
Offset:	0x4
Reset:	0x00
Property:	-

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section.

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

Bits 7:0 – TEMP[7:0] Temporary
Temporary register for read/write operations in 16-bit registers.

22.13.6 Debug Control

Name: DBGCTRL
Offset: 0x05
Reset: 0x00
Property: -

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

Bit 0 – DBGRUN Debug Run

Value	Description
0	The peripheral is halted in Break Debug mode and ignores events
1	The peripheral will continue to run in Break Debug mode when the CPU is halted

22.13.7 Crystal Frequency Calibration

Name:	CALIB
Offset:	0x06
Reset:	0x00
Property:	-

This register stores the error value and the type of correction to be done. The register is written by software with an error value based on external calibration and/or temperature correction/s.

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

Bit 7 – SIGN Error Correction Sign Bit
This bit shows the direction of the correction.

Value	Description
`0x0`	Positive correction causing the prescaler to count slower
`0x1`	Negative correction causing the prescaler to count faster. This requires that the minimum prescaler configuration is DIV2

Bits 6:0 – ERROR[6:0] Error Correction Value
The number of correction clocks for each million RTC clock cycles interval (PPM).

22.13.8 Clock Selection

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

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

Bits 1:0 – CLKSEL[1:0] Clock Select
Writing these bits select the source for the RTC clock (CLK_RTC).

Value	Name	Description
`0x0`	OSC32K	32.768 kHz from OSC32K
`0x1`	OSC1K	1.024 kHz from OSC32K
`0x2`	XTAL32K	32.768 kHz from XOSC32K
`0x3`	EXTCLK	External clock from XTAL32K1 pin

22.13.9 Count

Name:	CNT
Offset:	0x08
Reset:	0x0000
Property:	-

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

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

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

Bits 15:8 – CNT[15:8] Counter High Byte
These bits hold the MSB of the 16-bit Counter register.

Bits 7:0 – CNT[7:0] Counter Low Byte
These bits hold the LSB of the 16-bit Counter register.

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. The application software must check that the CNTBUSY flag in the RTC.STATUS register is cleared before writing to this register.

22.13.10 Period

Name:	PER
Offset:	0x0A
Reset:	0xFFFF
Property:	-

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

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

Bits 15:8 – PER[15:8] Period High Byte
These bits hold the MSB of the 16-bit Period register.

Bits 7:0 – PER[7:0] Period Low Byte
These bits hold the LSB of the 16-bit Period register.

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. The application software must check that the PERBUSY flag in the RTC.STATUS register is cleared before writing to this register.

22.13.11 Compare

Name:	CMP
Offset:	0x0C
Reset:	0x0000
Property:	-

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

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

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

Bits 15:8 – CMP[15:8] Compare High Byte
These bits hold the MSB of the 16-bit Compare register.

Bits 7:0 – CMP[7:0] Compare Low Byte
These bits hold the LSB of the 16-bit Compare register.

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. The application software must check that the CMPBUSY flag in the RTC.STATUS register is cleared before writing to this register.

22.13.12 Periodic Interrupt Timer Control A

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

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

Bits 6:3 – PERIOD[3:0] Period
Writing this bit field selects the number of RTC clock cycles between each interrupt.

Value	Name	Description
0x0	OFF	No interrupt
0x1	CYC4	4 cycles
0x2	CYC8	8 cycles
0x3	CYC16	16 cycles
0x4	CYC32	32 cycles
0x5	CYC64	64 cycles
0x6	CYC128	128 cycles
0x7	CYC256	256 cycles
0x8	CYC512	512 cycles
0x9	CYC1024	1024 cycles
0xA	CYC2048	2048 cycles
0xB	CYC4096	4096 cycles
0xC	CYC8192	8192 cycles
0xD	CYC16384	16384 cycles
0xE	CYC32768	32768 cycles
0xF	-	Reserved

Bit 0 – PITEN Periodic Interrupt Timer Enable
Writing this bit field enables the PIT

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. The application software must check that the CTRLBUSY flag in the RTC.PITSTATUS register is cleared before writing to this register.

22.13.13 Periodic Interrupt Timer Status

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

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

Bit 0 – CTRLBUSY PITCTRLA Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Periodic Interrupt Timer Control A (RTC.PITCTRLA) register in the RTC clock domain.

22.13.14 PIT Interrupt Control

Name:	PITINTCTRL
Offset:	0x12
Reset:	0x00
Property:	-

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

Bit 0 – PI Periodic Interrupt

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

22.13.15 PIT Interrupt Flag

Name:	PITINTFLAGS
Offset:	0x13
Reset:	0x00
Property:	-

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

Bit 0 – PI Periodic Interrupt Flag
This flag is set when a periodic interrupt is issued.
Writing a '1' clears the flag.

22.13.16 Periodic Interrupt Timer Debug Control

Name:	PITDBGCTRL
Offset:	0x15
Reset:	0x00
Property:	-

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

Bit 0 – DBGRUN Debug Run

Value	Description
0	The peripheral is halted in Break Debug mode and ignores events
1	The peripheral will continue to run in Break Debug mode when the CPU is halted

22.13.17 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

23. TCA - 16-bit Timer/Counter Type A

23.1 Features

- 16-Bit Timer/Counter
- Three Compare Channels
- Double-Buffered Timer Period Setting
- Double-Buffered Compare Channels
- Waveform Generation:
  - Frequency generation
  - Single-slope PWM (Pulse-Width Modulation)
  - Dual-slope PWM
- Count on Event
- Timer Overflow Interrupts/Events
- One Compare Match per Compare Channel
- Two 8-Bit Timer/Counters in Split Mode

23.2 Overview

The flexible 16-Bit PWM Timer/Counter type A (TCA) provides accurate program execution timing, frequency and waveform generation, and command execution.

A TCA consists of a base counter and a set of compare channels. The base counter can be used to count clock cycles or events or let events control how it counts clock cycles. It has direction control and 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.

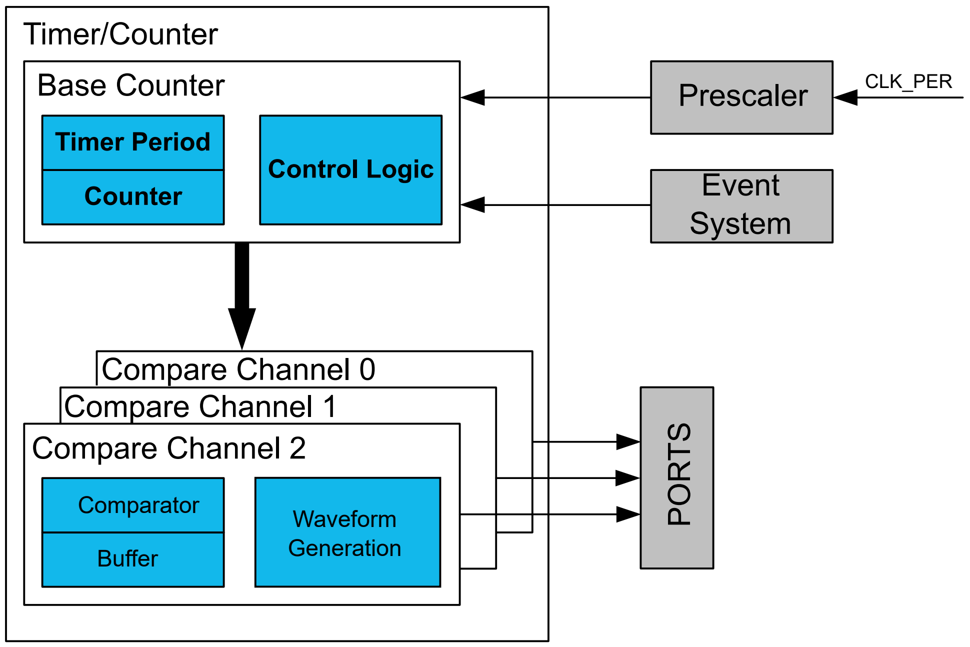
A timer/counter can be clocked and timed from the peripheral clock, with optional prescaling, or from the Event System. The Event System can also be used for direction control or synchronizing operations.

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).

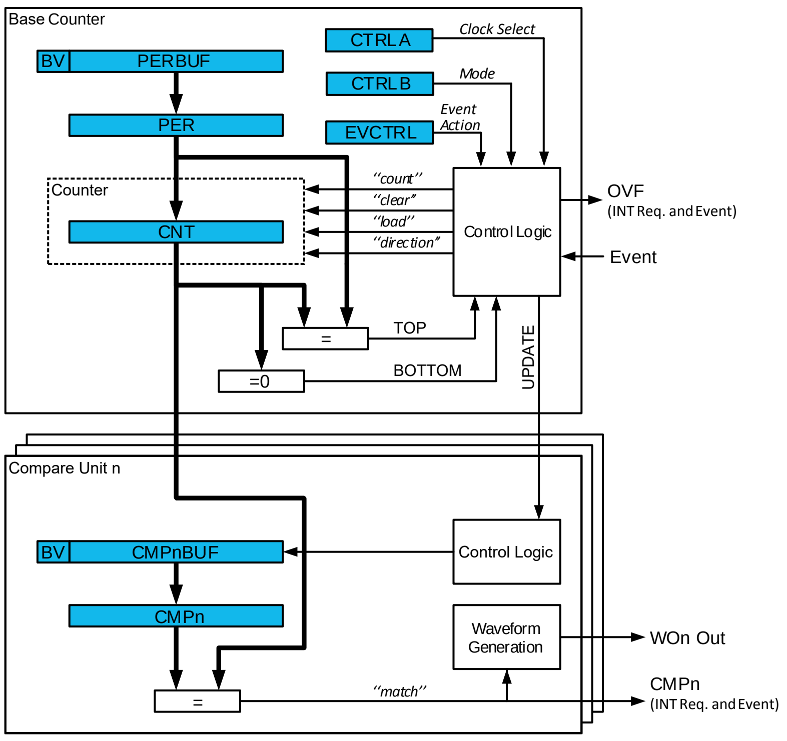
Figure 23-1. 16-Bit Timer/Counter and Closely Related Peripherals



23.2.1 Block Diagram

The figure below shows a detailed block diagram of the timer/counter.

Figure 23-2. Timer/Counter Block Diagram



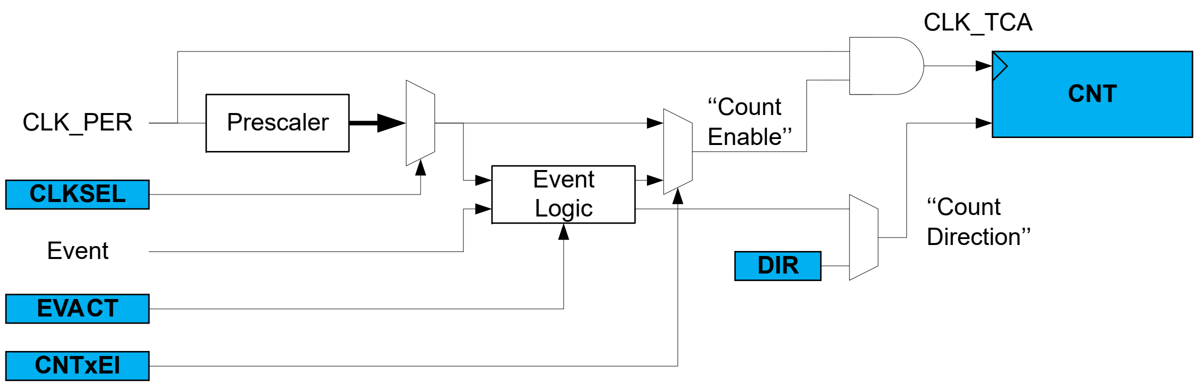
The Counter (TCAn.CNT) register, Period and Compare (TCAn.PER and TCAn.CMPn) registers, and their corresponding buffer registers (TCAn.PERBUF and TCAn.CMPnBUF) are 16-bit registers. All buffer registers have a Buffer Valid (BV) flag indicating when the buffer contains a new value.

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 23-3. Timer/Counter Clock Logic



23.2.2 Signal Description

Signal	Description	Type
WOn	Digital output	Waveform output

23.3 Functional Description

23.3.1 Definitions

The following definitions are used throughout the documentation:

Table 23-1. Timer/Counter Definitions

Name	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x0000
MAX	The counter reaches MAXimum when it becomes all ones
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence
UPDATE	The update condition is met when the timer/counter reaches BOTTOM or TOP, depending on the Waveform Generator mode. Buffered registers with valid buffer values will be updated unless the Lock Update (LUPD) bit in the TCAn.CTRLE register has been set.
CNT	Counter register value
CMP	Compare register value
PER	Period register value

In general, the term timer is used when the timer/counter is counting periodic clock ticks. The term counter is used when the input signal has sporadic or irregular ticks. The latter can be the case when counting events.

23.3.2 Initialization

To start using the timer/counter in a basic mode, follow these steps:

1. Write a TOP value to the Period (TCAn.PER) register.
2. Enable the peripheral by writing a '1' to the Enable (ENABLE) bit in the Control A (TCAn.CTRLA) register.
   The counter will start counting clock ticks according to the prescaler setting in the Clock Select (CLKSEL) bit field in TCAn.CTRLA.
3. Optional: By writing a '1' to the Enable Counter Event Input A (CNTAEI) bit in the Event Control (TCAn.EVCTRL) register, events are counted instead of clock ticks.
4. The counter value can be read from the Counter (CNT) bit field in the Counter (TCAn.CNT) register.

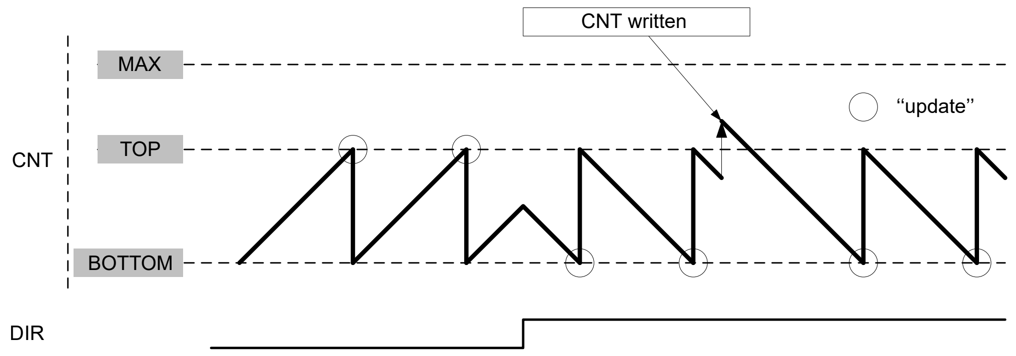
23.3.3 Operation

23.3.3.1 Normal Operation

In 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 23-4. Normal Operation



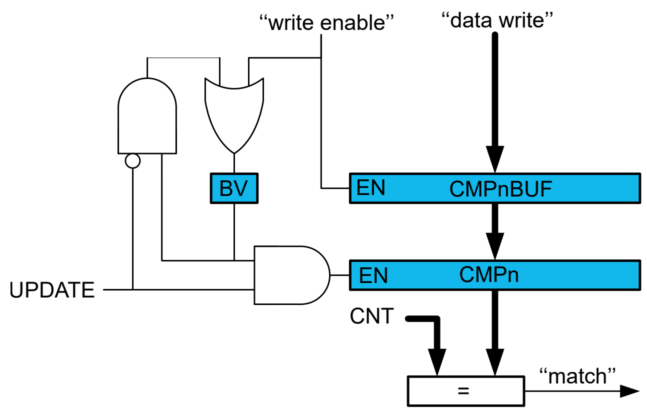
It is possible to change the counter value in the Counter (TCAn.CNT) register when the counter is running. The write access to TCAn.CNT register has higher priority than count, clear or reload, and will be immediate. The direction of the counter can also be changed during ordinary operation by writing to the Direction (DIR) bit in the Control E (TCAn.CTRLE) register.

23.3.3.2 Double Buffering

The Period (TCAn.PER) register value and the Compare n (TCAn.CMPn) register values are all double-buffered (TCAn.PERBUF and TCAn.CMPnBUF).

Each buffer register has a Buffer Valid (BV) flag (PERBV, CMPnBV) in the Control F (TCAn.CTRLF) register, which indicates that the buffer register contains a valid (new) value that can be copied into the corresponding Period or Compare register. When the Period register and Compare n registers are used for a compare operation, the BV flag is set when data are written to the buffer register and cleared on an UPDATE condition. The figure below shows this for a Compare (CMPn) register.

Figure 23-5. Period and Compare Double Buffering



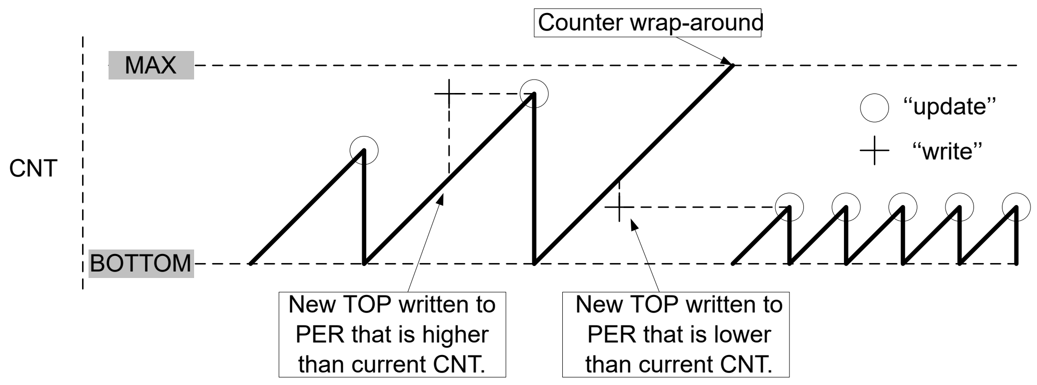
Both the TCAn.CMPn and TCAn.CMPnBUF registers are available as I/O registers, allowing the initialization and bypassing of the buffer register and the double-buffering function.

23.3.3.3 Changing the Period

The Counter period is changed by writing a new TOP value to the Period (TCAn.PER) register.

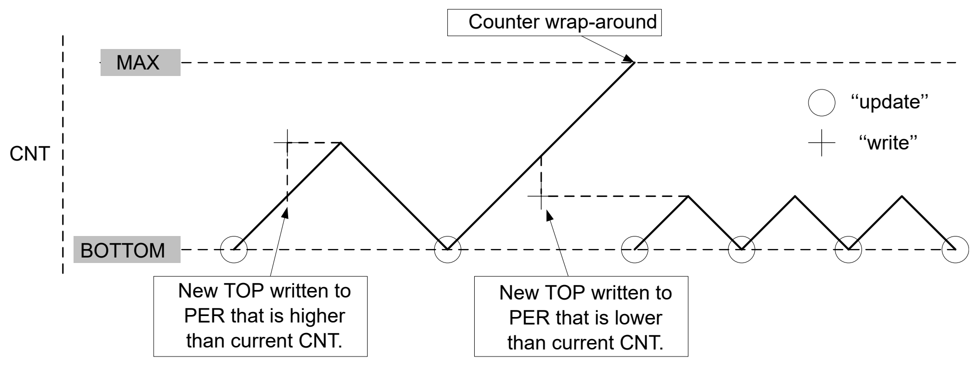
No Buffering: Any period update is immediate if not using double-buffering.

Figure 23-6. Changing the Period Without Buffering



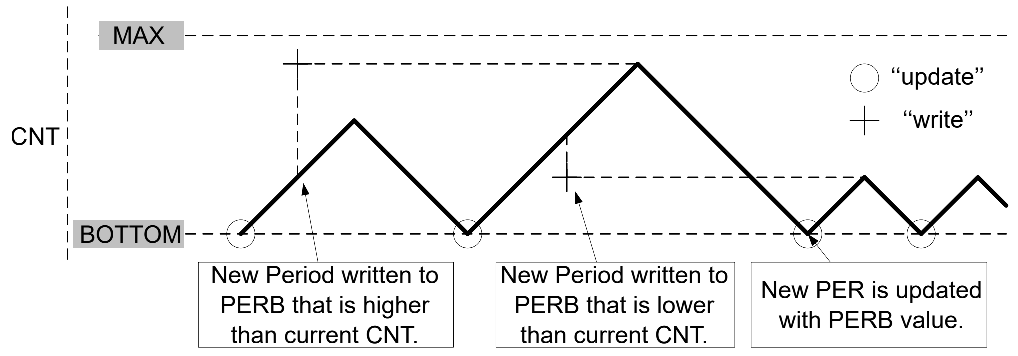
A counter wrap-around can occur in any mode of operation when counting up without buffering, as the TCAn.CNT and TCAn.PER registers are continuously compared. If writing a new TOP value to TCAn.PER lower than the current TCAn.CNT, the counter will wrap first before a compare match occurs.

Figure 23-7. 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 23-8. Changing the Period Using Buffering


MAX
CNT
BOTTOM
"update"
"write"
New Period written to PERB that is higher than current CNT.
New Period written to PERB that is lower than current CNT.
New PER is updated with PERB value.

Note: Buffering is used in figures illustrating TCA operation if not otherwise specified.

23.3.3.4 Compare Channel

Each Compare Channel n continuously compares the counter value (TCAn.CNT) with the Compare n (TCAn.CMPn) register. If TCAn.CNT equals TCAn.CMPn the Comparator n signals a match. The match will set the Compare Channel's interrupt flag at the next timer clock cycle - and the optional interrupt is generated.

The Compare n Buffer (TCAn.CMPnBUF) register provides 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.

23.3.3.4.1 Waveform Generation

The compare channels can be used for waveform generation on the corresponding port pins. The following requirements must be met to make the waveform visible on the connected port pin:

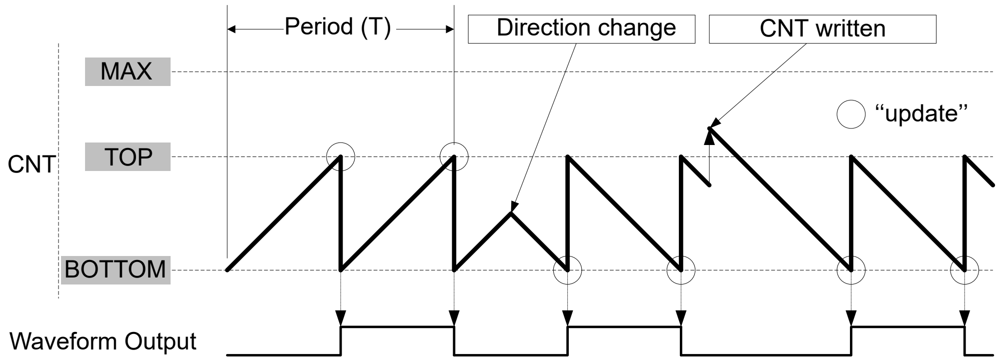
1. A Waveform Generation mode must be selected by writing the Waveform Generation Mode (WGMODE) bit field in the TCAn.CTRLB register.
2. The used compare channels must be enabled (CMPnEN = `1` in TCAn.CTRLB), which will override the output value for the corresponding pin. An alternative pin can be selected 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.

23.3.3.4.2 Frequency (FRQ) Waveform Generation

For frequency generation, the period time (T) is controlled by the TCAn.CMP0 register instead of the Period (TCAn.PER) register. The corresponding waveform generator output is toggled on each compare match between the TCAn.CNT and TCAn.CMPn registers.

Figure 23-9. Frequency Waveform Generation



The 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).

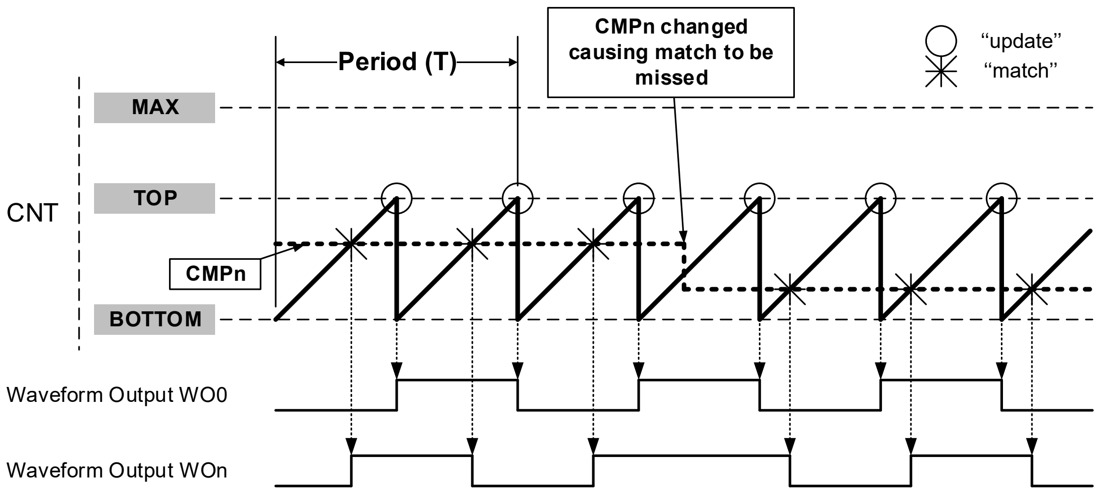
Use the TCAn.CMP1 and TCAn.CMP2 registers to get additional waveform outputs WOn. The waveforms WOn can either be identical or offset to WO0. The offset can be influenced by TCAn.CMPn, TCAn.CNT and the count direction. The offset in seconds t_{Offset} can be calculated using the equations in the table below. The equations are only valid when CMPn < CMP0.

Table 23-2. 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

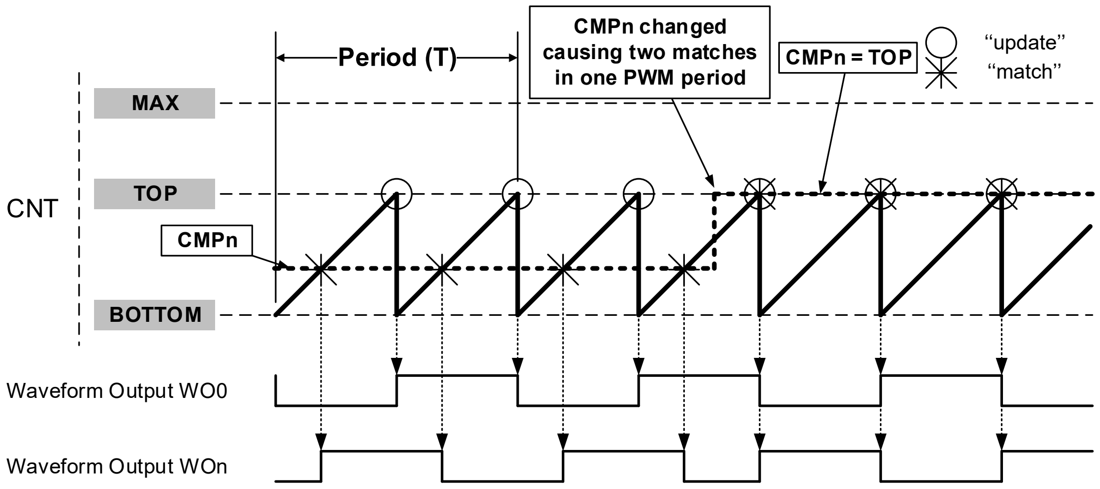
The figure below shows the leading and trailing offset for WOn, where both equations can be used. The correct equation is determined by count direction, and the state of CMPn vs. CNT when the timer is enabled or CMPn is changed.

Figure 23-10. Offset When Counting Up



The figure below shows how changing CMPn during run-time can invert the waveform.

Figure 23-11. Inverting Waveform Output

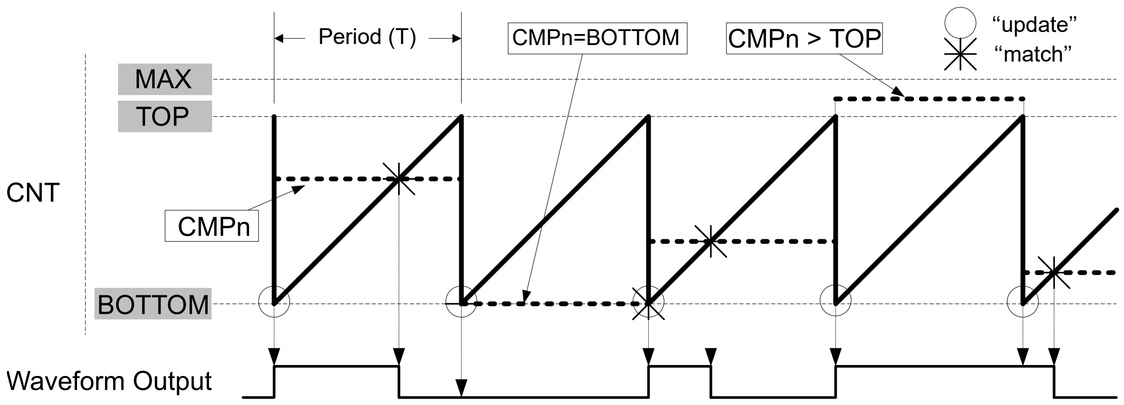


23.3.3.4.3 Single-Slope PWM Generation

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 23-12. 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}):

$$R_{\mathrm{PWM\_SS}} = \frac{\log(\mathrm{PER}+1)}{\log(2)}$$

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:

$$f_{\mathrm{PWM\_SS}} = \frac{f_{\mathrm{CLK\_PER}}}{N(\mathrm{PER}+1)}$$

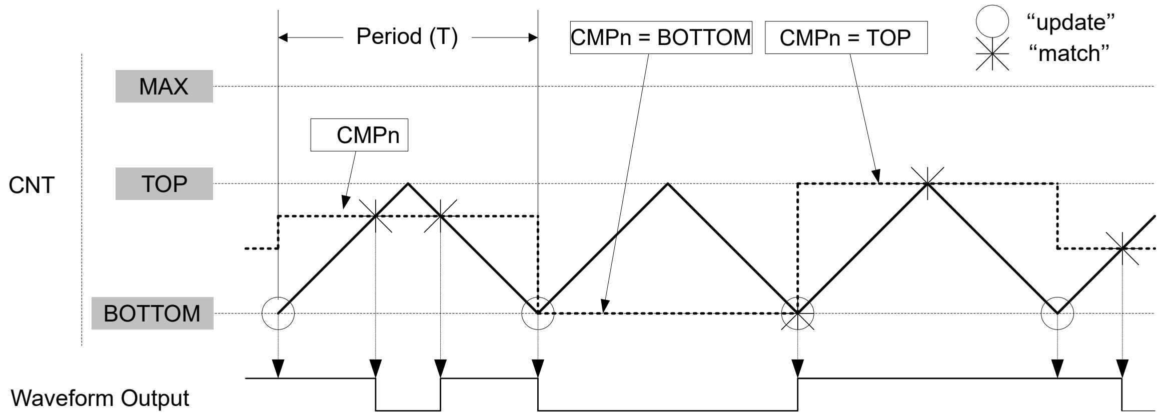
23.3.3.4.4 Dual-Slope PWM Generation

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 23-13. 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}):

$$R_{\mathrm{PWM\_DS}} = \frac{\log(\mathrm{PER}+1)}{\log(2)}$$

The PWM frequency depends on the period setting in the TCAn.PER register, the peripheral clock frequency (f_{CLK_PER}), and the prescaler divider selected in the CLKSEL bit field in the TCAn.CTRLA register. It is calculated by the following equation:

$$f_{\mathrm{PWM\_DS}} = \frac{f_{\mathrm{CLK\_PER}}}{2N \cdot \mathrm{PER}}$$

N represents the prescaler divider used.

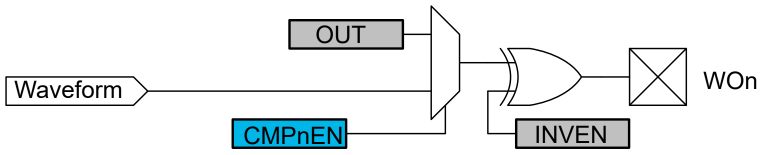
Using dual-slope PWM results in approximately half the maximum operation frequency compared to single-slope PWM operation due to twice the number of timer increments per period.

23.3.3.4.5 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 23-14. Port Override for Timer/Counter Type A



23.3.3.5 Timer/Counter Commands

A set of commands can be issued by software to immediately change the state of the peripheral. These commands give direct control of the UPDATE, RESTART and RESET signals. A command is issued by writing the respective value to the Command (CMD) bit field in the Control E (TCAn.CTRLESET) register.

An UPDATE command has the same effect as when an UPDATE condition occurs, except that the UPDATE command is not affected by the state of the Lock Update (LUPD) bit in the Control E (TCAn.CTRLE) register.

The software can force a restart of the current waveform period by issuing a RESTART command. In this case, the counter 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).

23.3.3.6 Split Mode - Two 8-Bit Timer/Counters

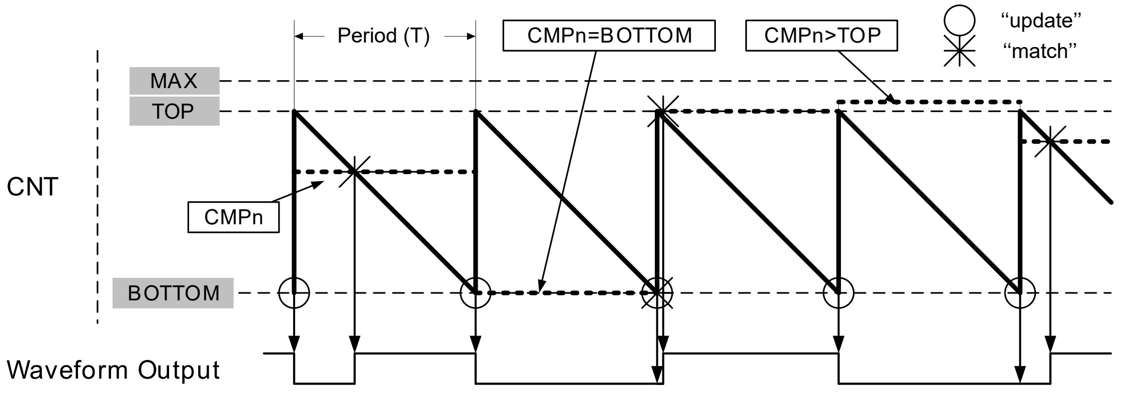
Split Mode Overview

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 23-15. 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 23.6. Register Summary - Split Mode).

Split Mode Differences Compared to Normal Mode

- Count:
  - Down-count only
  - Low Byte Timer Counter (TCAn.LCNT) register and High Byte Timer Counter (TCAn.HCNT) register are independent
- Waveform generation:
  - Single-slope PWM only (WGMODE = SINGLESLOPE in 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

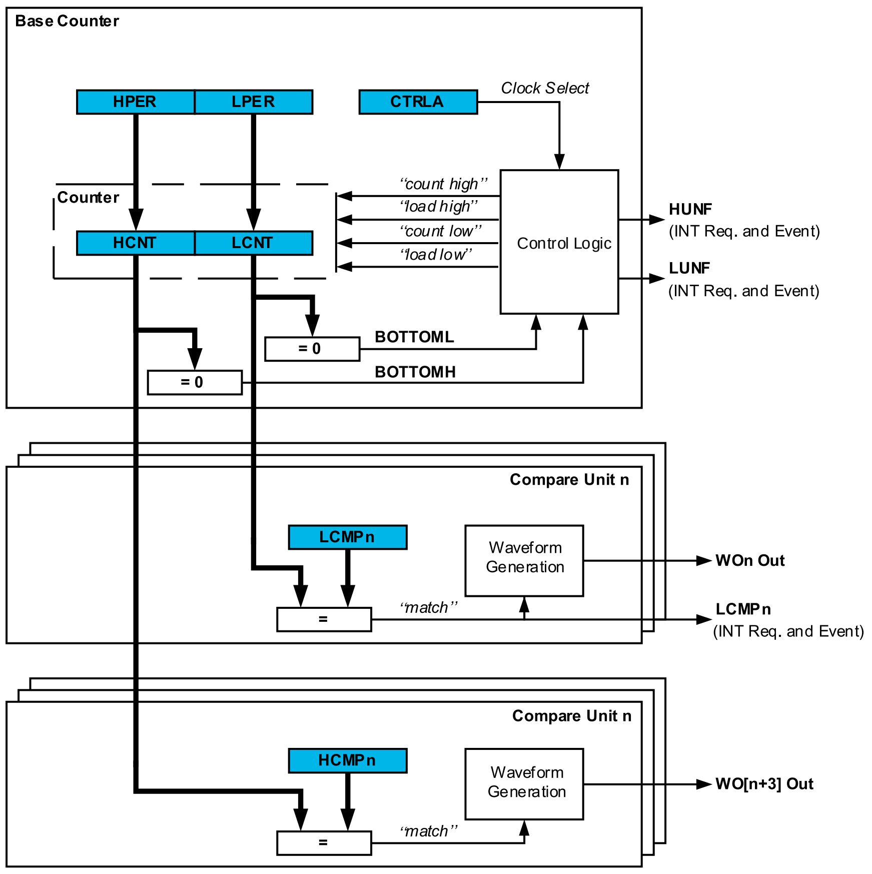


Figure 23-16. Timer/Counter Block Diagram Split Mode

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.

23.3.4 Events

The TCA can generate the events described in the table below. All event generators except TCAn_HUNF are shared between Normal mode and Split mode operation. The generator name indicates what specific signal the generator represents in each mode in the following way: OVF_LUNF corresponds to overflow in Normal mode and Low byte timer underflow in Split mode. The same applies to CMPn_LCMPn.

Table 23-3. 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 23-4. 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.

23.3.5 Interrupts

Table 23-5. 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 23-6. 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.

23.3.6 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.

23.4 Register Summary - Normal Mode

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY				CLKSEL[2:0]			ENABLE
0x01	CTRLB	7:0		CMP2EN	CMP1EN	CMP0EN	ALUPD	WGMODE[2:0]		
0x02	CTRLC	7:0						CMP2OV	CMP1OV	CMP0OV
0x03	CTRLD	7:0								SPLITM
0x04	CTRLECLR	7:0					CMD[1:0]		LUPD	DIR
0x05	CTRLESET	7:0					CMD[1:0]		LUPD	DIR
0x06	CTRLFCLR	7:0					CMP2BV	CMP1BV	CMP0BV	PERBV
0x07	CTRLFSET	7:0					CMP2BV	CMP1BV	CMP0BV	PERBV
0x08	Reserved									
0x09	EVCTRL	7:0	EVACTB[2:0]			CNTBEI	EVACTA[2:0]			CNTAEI
0x0A	INTCTRL	7:0		CMP2	CMP1	CMP0				OVF
0x0B	INTFLAGS	7:0		CMP2	CMP1	CMP0				OVF
0x0C ... 0x0D	Reserved									
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	TEMP	7:0	TEMP[7:0]							
0x10 ... 0x1F	Reserved									
0x20	CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x22 ... 0x25	Reserved									
0x26	PER	7:0	PER[7:0]							
		15:8	PER[15:8]							
0x28	CMP0	7:0	CMP[7:0]							
		15:8	CMP[15:8]							
0x2A	CMP1	7:0	CMP[7:0]							
		15:8	CMP[15:8]							
0x2C	CMP2	7:0	CMP[7:0]							
		15:8	CMP[15:8]							
0x2E ... 0x35	Reserved									
0x36	PERBUF	7:0	PERBUF[7:0]							
		15:8	PERBUF[15:8]							
0x38	CMP0BUF	7:0	CMPBUF[7:0]							
		15:8	CMPBUF[15:8]							
0x3A	CMP1BUF	7:0	CMPBUF[7:0]							
		15:8	CMPBUF[15:8]							
0x3C	CMP2BUF	7:0	CMPBUF[7:0]							
		15:8	CMPBUF[15:8]							

23.5 Register Description - Normal Mode

23.5.1 Control A - 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

23.5.2 Control B - Normal Mode

Name:	CTRLB
Offset:	0x01
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
		CMP2EN	CMP1EN	CMP0EN	ALUPD	WGMODE[2:0]		
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 4, 5, 6 – CMPEN Compare n Enable

In the FRQ and PWM Waveform Generation modes, the Compare n Enable (CMPnEN) bits will make the waveform output available on the pin corresponding to WOn, overriding the value in the corresponding PORT output register.

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 23-7. 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.

23.5.3 Control C - Normal Mode

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						CMP2OV	CMP1OV	CMP0OV
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – CMP2OV Compare Output Value 2
See CMP0OV.

Bit 1 – CMP1OV Compare Output Value 1
See CMP0OV.

Bit 0 – CMP0OV Compare Output Value 0
The CMPnOV bits allow direct access to the waveform generator's output 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.

23.5.4 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.

23.5.5 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)

23.5.6 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)

23.5.7 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.

23.5.8 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.

23.5.9 Event Control

Name:	EVCTRL
Offset:	0x09
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
	EVACTB[2:0]			CNTBEI	EVACTA[2:0]			CNTAEI
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:5 – EVACTB[2:0] Event Action B
These bits define what action the counter will take upon certain event conditions.

Value	Name	Description
0x0	NONE	No action
0x1	-	Reserved
0x2	-	Reserved
0x3	UPDOWN	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

23.5.10 Interrupt Control Register - Normal Mode

Name:	INTCTRL
Offset:	0x0A
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
		CMP2	CMP1	CMP0				OVF
Access		R/W	R/W	R/W				R/W
Reset		0	0	0				0

Bit 6 – CMP2 Compare Channel 2 Interrupt Enable
See CMP0.

Bit 5 – CMP1 Compare Channel 1 Interrupt Enable
See CMP0.

Bit 4 – CMP0 Compare Channel 0 Interrupt Enable
Writing the CMPn bit to '1' enables the interrupt from Compare Channel n.

Bit 0 – OVF Timer Overflow/Underflow Interrupt Enable
Writing the OVF bit to '1' enables the overflow/underflow interrupt.

23.5.11 Interrupt Flag Register - Normal Mode

Name:	INTFLAGS
Offset:	0x0B
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
		CMP2	CMP1	CMP0				OVF
Access		R/W	R/W	R/W				R/W
Reset		0	0	0				0

Bit 6 – CMP2 Compare Channel 2 Interrupt Flag
See the CMP0 flag description.

Bit 5 – CMP1 Compare Channel 1 Interrupt Flag
See the CMP0 flag description.

Bit 4 – CMP0 Compare Channel 0 Interrupt Flag
The Compare Interrupt (CMPn) flag is set on a compare match on the corresponding compare channel.
For all modes of operation, the CMPn flag will be set when a compare match occurs between the Count (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.

23.5.12 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

23.5.13 Temporary Bits for 16-Bit Access

Name:	TEMP
Offset:	0x0F
Reset:	0x00
Property:	-

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section.

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

Bits 7:0 – TEMP[7:0] Temporary Bits for 16-bit Access

23.5.14 Counter Register - Normal Mode

Name:	CNT
Offset:	0x20
Reset:	0x00
Property:	-

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

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.

23.5.15 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.

23.5.16 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.

23.5.17 Period Buffer Register

Name:	PERBUF
Offset:	0x36
Reset:	0xFFFF
Property:	-

This register serves as the buffer for the Period (TCAn.PER) register. Writing to this register from the CPU or UPDI will set the Period Buffer Valid (PERBV) bit in the TCAn.CTRLF register.

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

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

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

Bits 15:8 – PERBUF[15:8] Period Buffer High Byte
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.

23.5.18 Compare n Buffer Register

Name:	CMPnBUF
Offset:	0x38 + n*0x02 [n=0..2]
Reset:	0x00
Property:	-

This register serves as the buffer for the associated Compare n (TCAn.CMPn) register. Writing to this register from the CPU or UPDI will set the Compare Buffer valid (CMPnBV) bit in the TCAn.CTRLF register.

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

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

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

Bits 15:8 – CMPBUF[15:8] Compare High Byte
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.

23.6 Register Summary - Split Mode

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY				CLKSEL[2:0]			ENABLE
0x01	CTRLB	7:0		HCMP2EN	HCMP1EN	HCMP0EN		LCMP2EN	LCMP1EN	LCMP0EN
0x02	CTRLC	7:0		HCMP2OV	HCMP1OV	HCMP0OV		LCMP2OV	LCMP1OV	LCMP0OV
0x03	CTRLD	7:0								SPLITM
0x04	CTRLECLR	7:0					CMD[1:0]		CMDEN[1:0]	
0x05	CTRLESET	7:0					CMD[1:0]		CMDEN[1:0]	
0x06 ... 0x09	Reserved									
0x0A	INTCTRL	7:0		LCMP2	LCMP1	LCMP0			HUNF	LUNF
0x0B	INTFLAGS	7:0		LCMP2	LCMP1	LCMP0			HUNF	LUNF
0x0C ... 0x0D	Reserved									
0x0E	DBGCTRL	7:0								DBGRUN
0x0F ... 0x1F	Reserved									
0x20	LCNT	7:0	LCNT[7:0]							
0x21	HCNT	7:0	HCNT[7:0]							
0x22 ... 0x25	Reserved									
0x26	LPER	7:0	LPER[7:0]							
0x27	HPER	7:0	HPER[7:0]							
0x28	LCMP0	7:0	LCMP[7:0]							
0x29	HCMP0	7:0	HCMP[7:0]							
0x2A	LCMP1	7:0	LCMP[7:0]							
0x2B	HCMP1	7:0	HCMP[7:0]							
0x2C	LCMP2	7:0	LCMP[7:0]							
0x2D	HCMP2	7:0	HCMP[7:0]							

23.7 Register Description - Split Mode

23.7.1 Control A - 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

23.7.2 Control B - Split Mode

Name:	CTRLB
Offset:	0x01
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
		HCMP2EN	HCMP1EN	HCMP0EN		LCMP2EN	LCMP1EN	LCMP0EN
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – HCMP2EN High byte Compare 2 Enable
See HCMP0EN.

Bit 5 – HCMP1EN High byte Compare 1 Enable
See HCMP0EN.

Bit 4 – HCMP0EN High byte Compare 0 Enable
Setting the HCMPnEN bit in the FRQ or PWM Waveform Generation mode of operation will override the port output register for the corresponding WO[n+3] pin.

Bit 2 – LCMP2EN Low byte Compare 2 Enable
See LCMP0EN.

Bit 1 – LCMP1EN Low byte Compare 1 Enable
See LCMP0EN.

Bit 0 – LCMP0EN Low byte Compare 0 Enable
Setting the LCMPnEN bit in the FRQ or PWM Waveform Generation mode of operation will override the port output register for the corresponding WOn pin.

23.7.3 Control C - Split Mode

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		HCMP2OV	HCMP1OV	HCMP0OV		LCMP2OV	LCMP1OV	LCMP0OV
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – HCMP2OV High byte Compare 2 Output Value
See HCMP0OV.

Bit 5 – HCMP1OV High byte Compare 1 Output Value
See HCMP0OV.

Bit 4 – HCMP0OV High byte Compare 0 Output Value
The HCMPnOV bit allows direct access to the output 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.

23.7.4 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.

23.7.5 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

23.7.6 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

23.7.7 Interrupt Control Register - Split Mode

Name:	INTCTRL
Offset:	0x0A
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
		LCMP2	LCMP1	LCMP0			HUNF	LUNF
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit 6 – LCMP2 Low byte Compare Channel 2 Interrupt Enable
See LCMP0.

Bit 5 – LCMP1 Low byte Compare Channel 1 Interrupt Enable
See LCMP0.

Bit 4 – LCMP0 Low byte Compare Channel 0 Interrupt Enable
Writing the LCMPn bit to ‘1’ enables the low byte Compare Channel n interrupt.

Bit 1 – HUNF High byte Underflow Interrupt Enable
Writing the HUNF bit to ‘1’ enables the high byte underflow interrupt.

Bit 0 – LUNF Low byte Underflow Interrupt Enable
Writing the LUNF bit to ‘1’ enables the low byte underflow interrupt.

23.7.8 Interrupt Flag Register - Split Mode

Name:	INTFLAGS
Offset:	0x0B
Reset:	0x00
Property:	-

Bit	7	6	5	4	3	2	1	0
		LCMP2	LCMP1	LCMP0			HUNF	LUNF
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit 6 – LCMP2 Low byte Compare Channel 2 Interrupt Flag
See LCMP0 flag description.

Bit 5 – LCMP1 Low byte Compare Channel 1 Interrupt Flag
See LCMP0 flag description.

Bit 4 – LCMP0 Low byte Compare Channel 0 Interrupt Flag
The Low byte Compare Interrupt (LCMPn) flag is set on a compare match on the corresponding compare channel in the low byte timer.
For all modes of operation, the LCMPn flag will be set when a compare match occurs between the Low Byte Timer Counter (TCAn.LCNT) register and the corresponding Compare n (TCAn.LCMPn) register. 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.

23.7.9 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

23.7.10 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.

23.7.11 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.

23.7.12 Low Byte Timer Period Register - Split Mode

Name:	LPER
Offset:	0x26
Reset:	0xFF
Property:	-

The TCAn.LPER register contains the TOP value for the low byte timer.

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

Bits 7:0 – LPER[7:0] Period Value Low Byte Timer
This bit field holds the TOP value for the low byte timer.

23.7.13 High Byte Period Register - Split Mode

Name:	HPER
Offset:	0x27
Reset:	0xFF
Property:	-

The TCAn.HPER register contains the TOP value for the high byte timer.

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

Bits 7:0 – HPER[7:0] Period Value High Byte Timer
This bit field holds the TOP value for the high byte timer.

23.7.14 Compare Register n For Low Byte Timer - Split Mode

Name:	LCMPn
Offset:	0x28 + n*0x02 [n=0..2]
Reset:	0x00
Property:	-

The TCAn.LCMPn register represents the compare value of Compare Channel n for the low byte timer. This register is continuously compared to the counter value of the low byte timer, TCAn.LCNT. Normally, the outputs from the comparators are then used to generate waveforms.

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

Bits 7:0 – LCMP[7:0] Compare Value of Channel n
This bit field holds the compare value of channel n that is compared to TCAn.LCNT.

23.7.15 High Byte Compare Register n - Split Mode

Name:	HCMPn
Offset:	0x29 + n*0x02 [n=0..2]
Reset:	0x00
Property:	-

The TCAn.HCMPn register represents the compare value of Compare Channel n for the high byte timer. This register is continuously compared to the counter value of the high byte timer, TCAn.HCNT. Normally, the outputs from the comparators are then used to generate waveforms.

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

Bits 7:0 – HCMP[7:0] Compare Value of Channel n
This bit field holds the compare value of channel n that is compared to TCAn.HCNT.

24. TCB - 16-Bit Timer/Counter Type B

24.1 Features

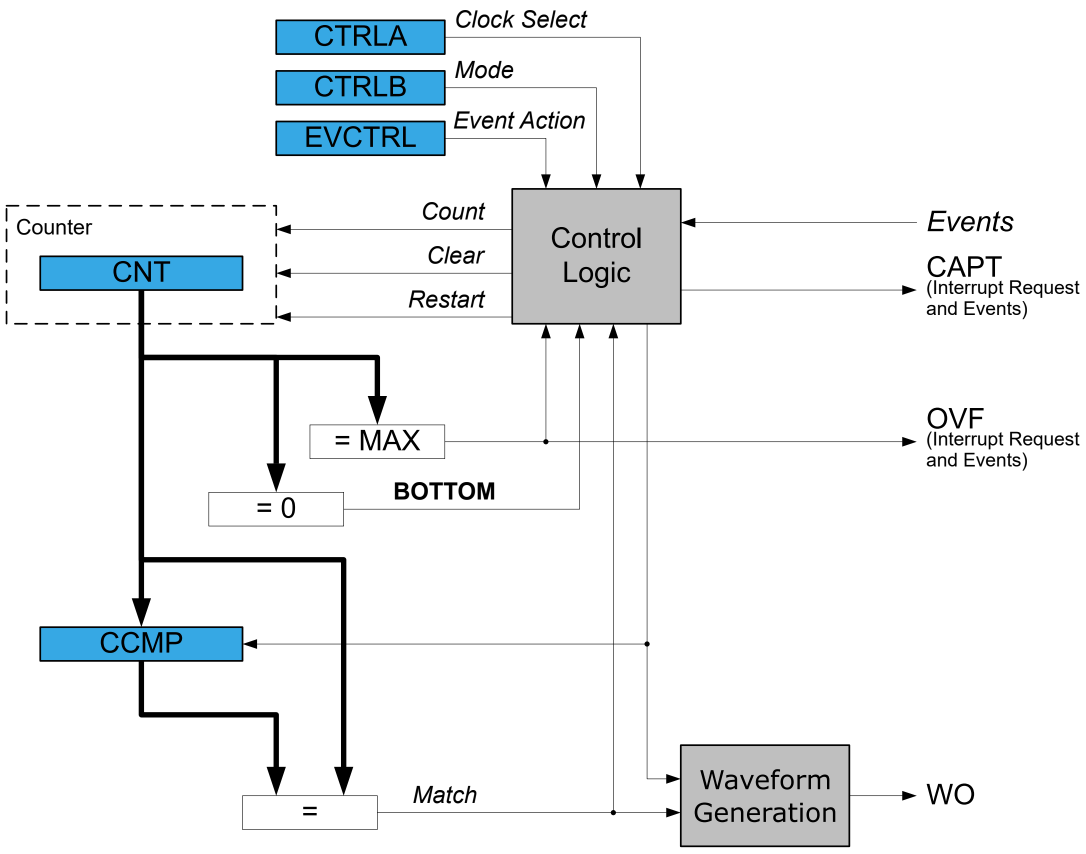
- 16-bit Counter Operation Modes:
  - Periodic interrupt
  - Time-out check
  - Input capture
    - On event
    - Frequency measurement
    - Pulse-width measurement
    - Frequency and pulse-width measurement
    - 32-bit capture
  - Single-shot
  - 8-bit Pulse-Width Modulation (PWM)
- Noise Canceler on Event Input
- Synchronize Operation with TCAn

24.2 Overview

The capabilities of the 16-bit Timer/Counter type B (TCB) include frequency and waveform generation and input capture on event with time and frequency measurement of digital signals. The TCB consists of a base counter and control logic that can be set in one of eight different modes, each mode providing unique functionality. The base counter is clocked by the peripheral clock with optional prescaling.

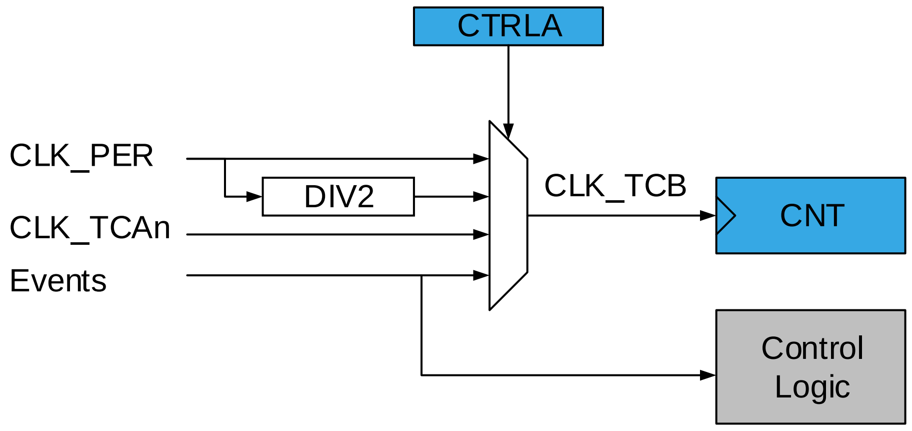
24.2.1 Block Diagram

Figure 24-1. 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 24-2. Timer/Counter Clock Logic



The Clock Select (CLKSEL) bit field in the Control A (TCBn.CTRLA) register selects one of the prescaler outputs directly, or an event channel as the clock (CLK_TCB) input.

Setting the timer/counter to use the clock from a TCAn allows the timer/counter to run in sync with that TCAn.

By using the EVSYS, any event source, such as an external clock signal on any I/O pin, may be used as the counter clock input or as a control logic input. When an event action controlled operation is used, the clock selection must be set to use an event channel as the counter input.

24.2.2 Signal Description

Signal	Description	Type
WO	Digital Asynchronous Output	Waveform Output

24.3 Functional Description

24.3.1 Definitions

The following definitions are used throughout the documentation:

Table 24-1. Timer/Counter Definitions

Name	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x0000
MAX	The counter reaches the maximum when it becomes 0xFFFF
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence
CNT	Count (TCBn.CNT) register value
CCMP	Capture/Compare (TCBn.CCMP) register value

Note: In general, the term ‘timer’ is used when the timer/counter is counting periodic clock ticks. The term ‘counter’ is used when the input signal has sporadic or irregular ticks.

24.3.2 Initialization

By default, the TCB is in Periodic Interrupt mode. Follow these steps to start using it:

1. Write a TOP value to the Compare/Capture (TCBn.CCMP) register.

2. Optional: Write the Compare/Capture Output Enable (CCMPEN) bit in the Control B (TCBn.CTRLB) register to ‘1’. This will make the waveform output available on the corresponding pin, overriding the value in the corresponding PORT output register.
3. Enable the counter by writing a ‘1’ to the ENABLE bit in the Control A (TCBn.CTRLA) register.
   The counter will start counting clock ticks according to the prescaler setting in the Clock Select (CLKSEL) bit field in the Control A (TCBn.CTRLA) register.
4. The counter value can be read from the Count (TCBn.CNT) register. The peripheral will generate a CAPT interrupt and event when the CNT value reaches TOP.
   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.

24.3.3 Operation

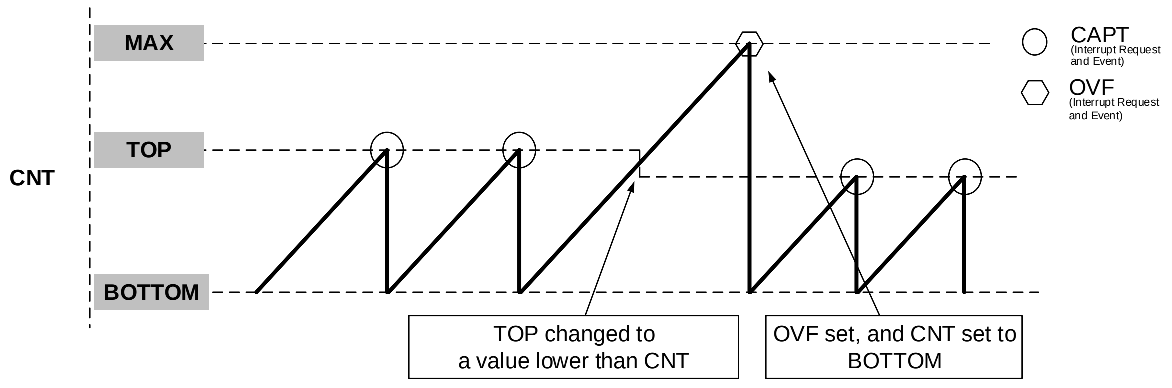
24.3.3.1 Modes

The timer can be configured to run in one of the eight different modes described in the sections below. The event pulse needs to be longer than one system clock cycle to ensure edge detection.

24.3.3.1.1 Periodic Interrupt Mode

In the Periodic Interrupt mode, the counter counts to the capture value and restarts from BOTTOM. A CAPT interrupt and event is generated when the CNT is equal to TOP. If TOP is updated to a value lower than CNT, upon reaching MAX, an OVF interrupt and event is generated, and the counter restarts from BOTTOM.

Figure 24-3. Periodic Interrupt Mode


MAX
TOP
CNT
BOTTOM
CAPT (Interrupt Request and Event)
OVF (Interrupt Request and Event)
TOP changed to a value lower than CNT
OVF set, and CNT set to BOTTOM

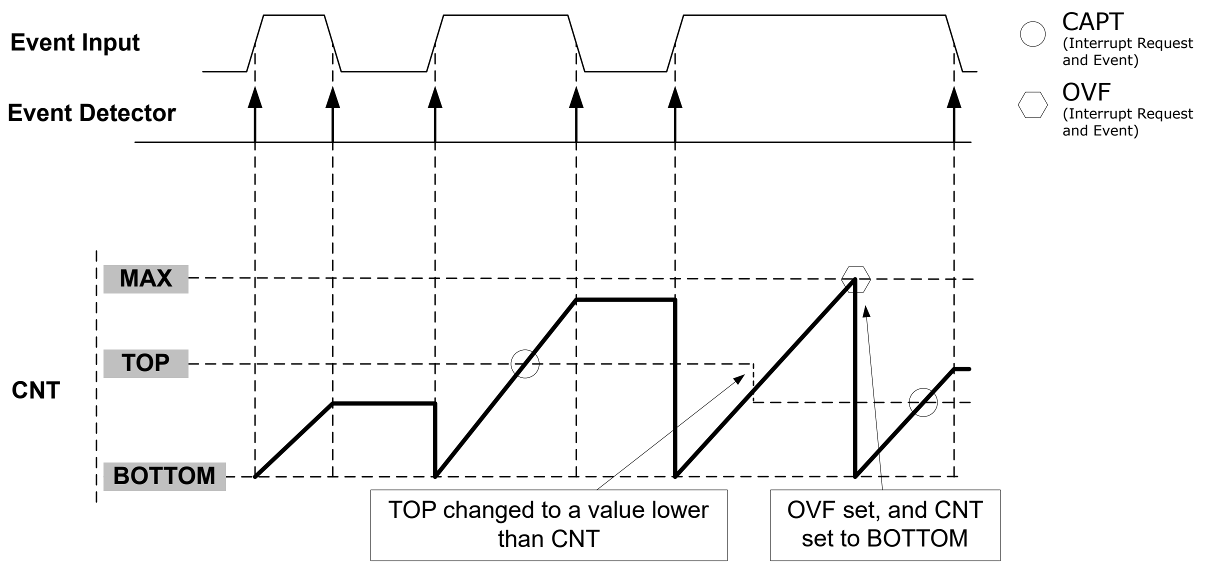
24.3.3.1.2 Time-Out Check Mode

In the Time-Out Check mode, the peripheral starts counting on the first signal edge and stops on the next signal edge detected on the event input channel. CNT remains stationary after the Stop edge (Freeze state). In Freeze state, the counter will restart on a new Start edge.

This mode requires TCB to be configured as an event user and is explained in the Events section.

Start or Stop edge is determined by the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register. If CNT reaches TOP before the second edge, a CAPT interrupt and event will be generated. If TOP is updated to a value lower than the CNT upon reaching MAX, an OVF interrupt and the simultaneous event is generated, and the counter restarts from BOTTOM. In Freeze state, reading the Count (TCBn.CNT) register or Compare/Capture (TCBn.CCMP) register, or writing the Run (RUN) bit in the Status (TCBn.STATUS) register has no effect.

Figure 24-4. Time-Out Check Mode


Event Input
Event Detector
CAPT
(Interrupt Request and Event)
OVF
(Interrupt Request and Event)
MAX
TOP
CNT
BOTTOM
TOP changed to a value lower than CNT
OVF set, and CNT set to BOTTOM

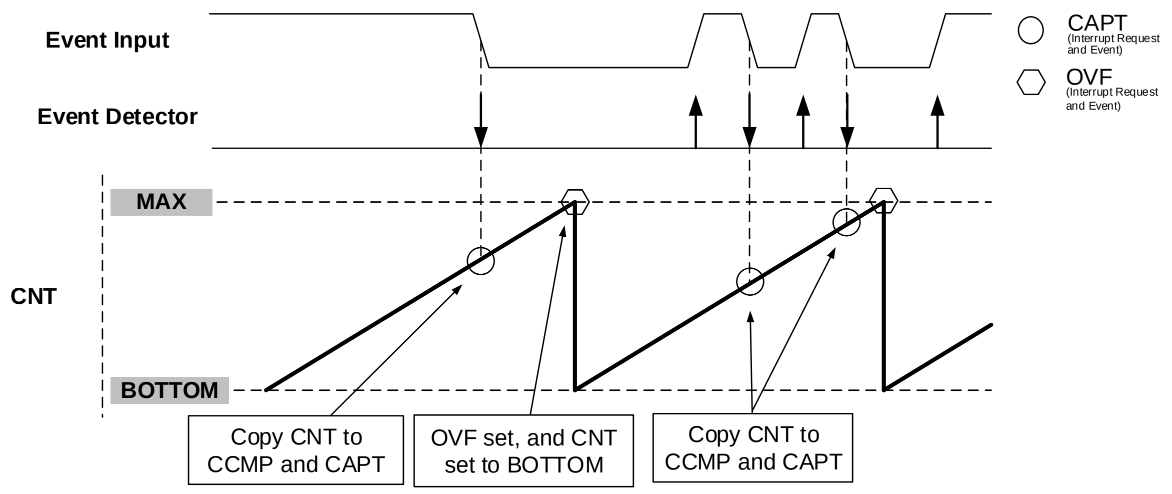
24.3.3.1.3 Input Capture on Event Mode

In the Input Capture on Event mode, the counter will count from BOTTOM to MAX. When an event is detected, the Count (TCBn.CNT) register value is transferred to the Compare/Capture (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 Compare/Capture (TCBn.CCMP) register. An OVF interrupt and event are generated when the CNT is MAX.

Figure 24-5. Input Capture on Event


Event Input
Event Detector
CNT
MAX
BOTTOM
CAPT
(Interrupt Request and Event)
OVF
(Interrupt Request and Event)
Copy CNT to CCMP and CAPT
OVF set, and CNT set to BOTTOM
Copy CNT to CCMP and CAPT

Important: It is recommended to write `0x0000` to the Count (TCBn.CNT) register when entering this mode from any other mode.

24.3.3.1.4 Input Capture Frequency Measurement Mode

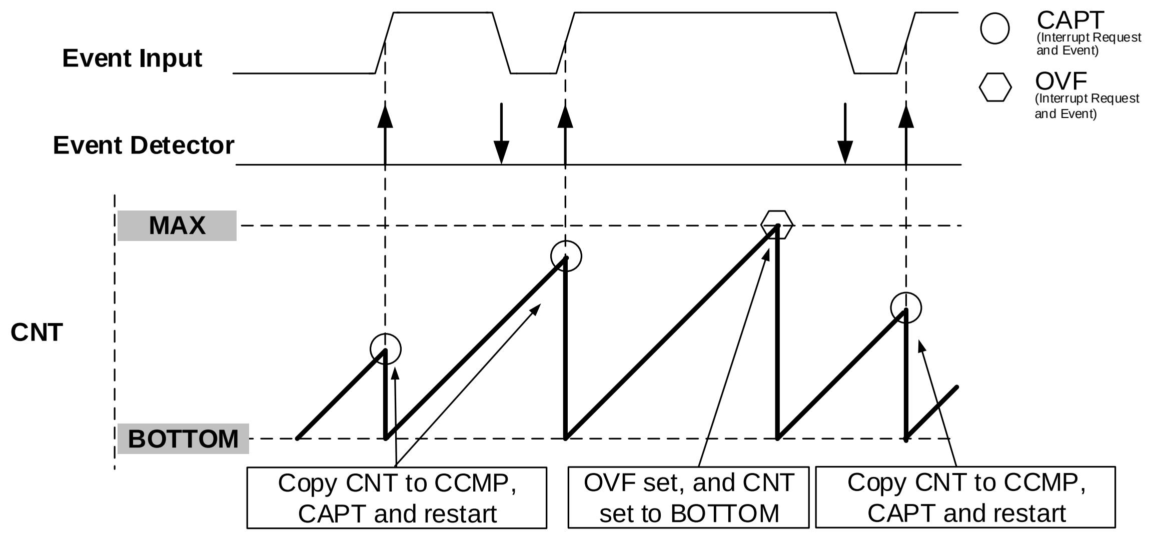
In the Input Capture Frequency Measurement mode, the TCB captures the counter value and restarts on either a positive or negative edge of the event input signal.

The CAPT interrupt flag is cleared automatically after reading the low byte of the Compare/Capture (TCBn.CCMP) register. An OVF interrupt and event is generated when the CNT value is MAX.

This mode requires TCB to be configured as an event user and is explained in the Events section.

The figure below illustrates this mode when configured to act on a rising edge.

Figure 24-6. Input Capture Frequency Measurement

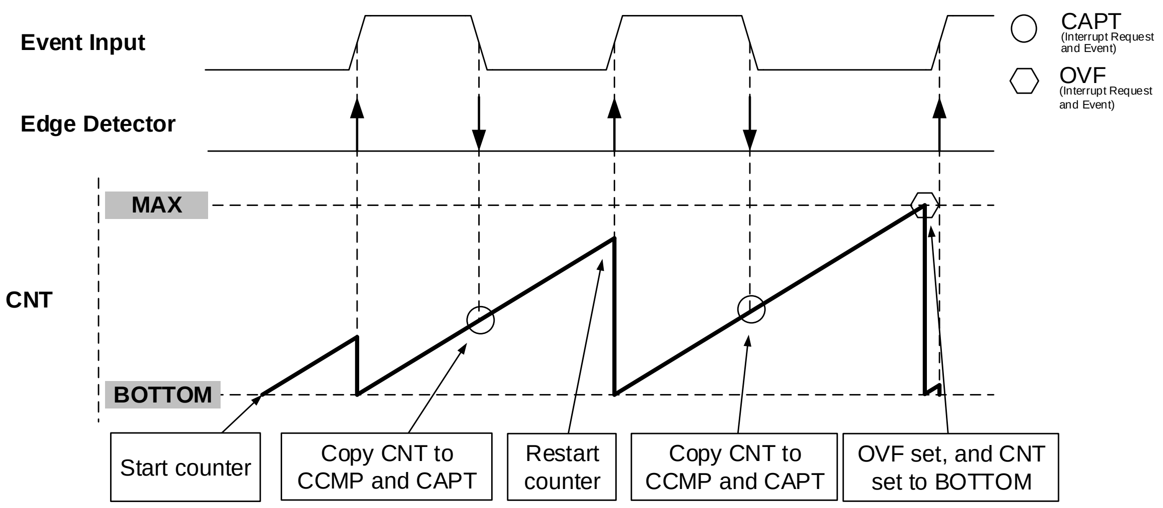

Event Input
Event Detector
CNT
MAX
BOTTOM
CAPT
(Interrupt Request and Event)
OVF
(Interrupt Request and Event)
Copy CNT to CCMP, CAPT and restart
OVF set, and CNT set to BOTTOM
Copy CNT to CCMP, CAPT and restart

24.3.3.1.5 Input Capture Pulse-Width Measurement Mode

In the Input Capture Pulse-Width Measurement mode, the input capture pulse-width measurement will restart the counter on a positive edge and capture 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 timer will automatically switch between rising and falling edge detection, but a minimum edge separation of two clock cycles is required for correct behavior.

This mode requires TCB to be configured as an event user and is explained in the Events section.

Figure 24-7. Input Capture Pulse-Width Measurement


Event Input
Edge Detector
CNT
MAX
BOTTOM
CAPT (Interrupt Request and Event)
OVF (Interrupt Request and Event)
Start counter
Copy CNT to CCMP and CAPT
Restart counter
Copy CNT to CCMP and CAPT
OVF set, and CNT set to BOTTOM

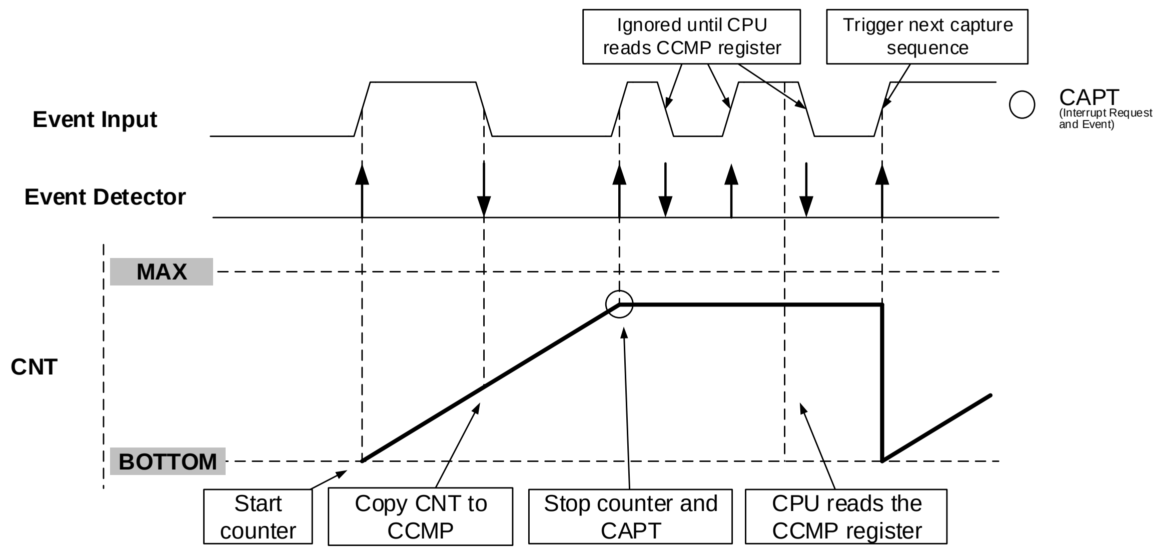
24.3.3.1.6 Input Capture Frequency and Pulse-Width Measurement Mode

In the Input Capture Frequency and Pulse-Width Measurement mode, the timer will start counting when a positive edge is detected on the event input signal. The count value is captured on the following falling edge. The counter stops when the second rising edge of the event input signal is detected 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 timer/counter is ready for a new capture sequence. Therefore, read the Count (TCBn.CNT) register before the Compare/Capture (TCBn.CCMP) register since it is reset to BOTTOM at the next positive edge of the event input signal. An OVF interrupt and event are generated when the CNT value is MAX.

Figure 24-8. Input Capture Frequency and Pulse-Width Measurement


Ignored until CPU reads CCMP register
Trigger next capture sequence
Event Input
CAPT
(Interrupt Request and Event)
Event Detector
MAX
CNT
BOTTOM
Start counter
Copy CNT to CCMP
Stop counter and CAPT
CPU reads the CCMP register

24.3.3.1.7 Single-Shot Mode

Use the Single-Shot mode to generate a pulse with a duration defined by the Compare (TCBn.CCMP) register every time a rising or falling edge is observed on a connected event channel.

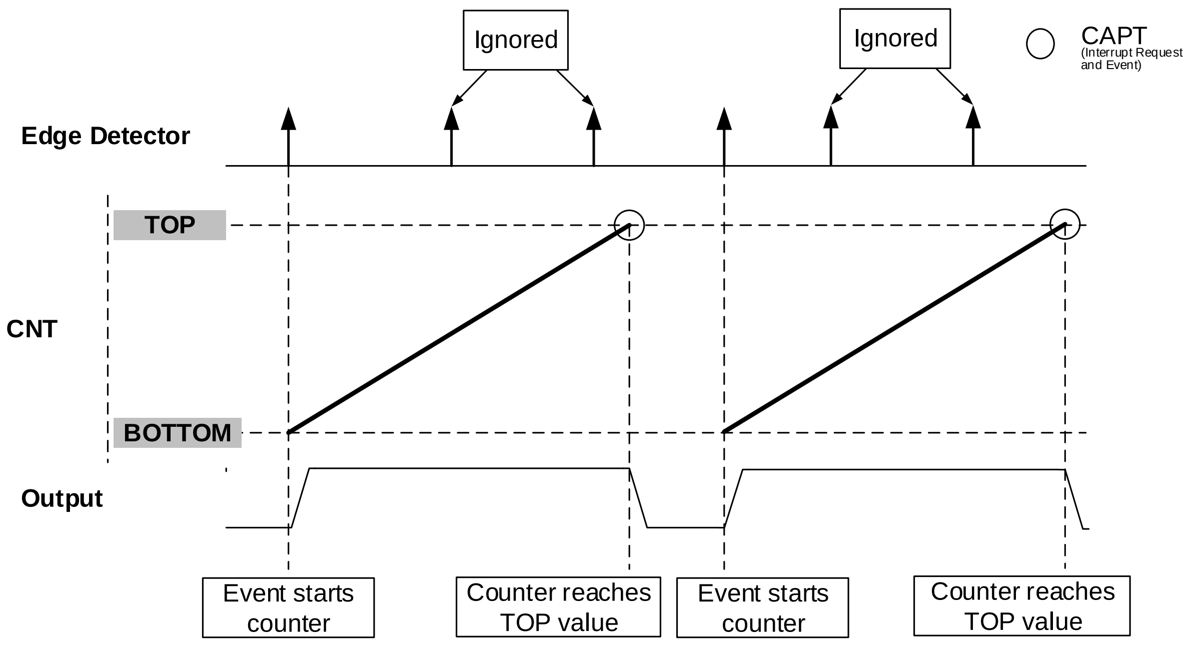
This mode requires TCB to be configured as an event user and is explained in the Events section.

When the counter stops, the output pin is set low. If an event is detected on the connected event channel, the timer will reset and start counting from BOTTOM to TOP while driving its output high. Read the Status (TCBn.STATUS) register RUN bit to see if the counter is counting. Once the value of CNT reaches the CCMP register, the counter will cease counting. Simultaneously, the output pin will transition to a low state for at least one counterclock cycle (TCB_CLK). During this period, any new event that occurs will be disregarded. Following this, there is a two peripheral clock cycles (PER_CLK) delay before the output is set high after receiving a new event. When the EDGE bit of the TCB.EVCTRL register is written to ‘1’, and any edge can trigger the start of the counter. Only positive edges trigger the start if the EDGE bit is ‘0’.

The counter will start counting as soon as the peripheral is enabled, even without triggering by an event or if the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register is modified while the peripheral is enabled, which is prevented by writing TOP to the Counter 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 register prevents this as well.

If the Event Asynchronous (ASYNC) bit in the Control B (TCBn.CTRLB) register is written to ‘1’, the timer will react asynchronously to an incoming event. An edge on the event will immediately cause the output signal to be set. The counter will still start counting two complete clock cycles after receiving the event, resulting in an observed delay of two to three clock cycles.

Figure 24-9. Single-Shot Mode

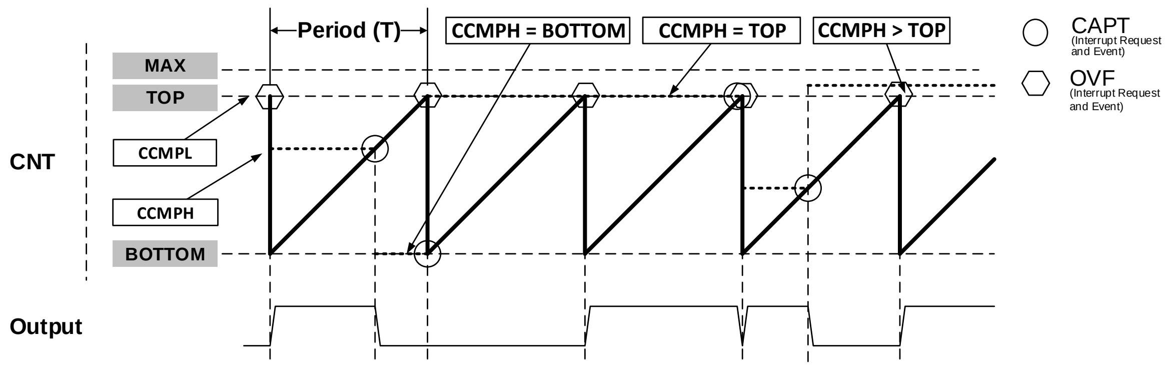


24.3.3.1.8 8-bit PWM Mode

The TCB can be configured to run in 8-bit PWM mode, where each register pair in the 16-bit Compare/Capture (TCBn.CCMPH and TCBn.CCMPL) register 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 24-10. 8-bit PWM Mode



24.3.3.2 Output

Timer synchronization and output logic level depend on the selected Timer Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) register. In Single-Shot mode, the timer/counter can be configured so that the signal generation happens asynchronously to an incoming event (ASYNC = 1 in TCBn.CTRLB). Then, the output signal is set immediately at the incoming event instead of being synchronized to the TCB clock. Due to synchronization delay for the counter, the waveform output will be set high three to four CLK_TCB cycles more than what is defined by the TOP value.

Writing the Compare/Capture Output Enable (CCMPEN) bit in TCBn.CTRLB to ‘1’ enables the waveform output, making 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 24-2. 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 Compare/Capture Pin Initial Value (CCMPINIT) bit in the Control B (TCBn.CTRLB) register selects the initial output level
0	Not applicable	Not applicable	No output

Changing modes while the peripheral is enabled is not recommended, as this can produce an unpredictable output. There is a possibility that an interrupt flag is set during the timer configuration. It is recommended to clear the Timer/Counter Interrupt Flags (TCBn.INTFLAGS) register after configuring the peripheral.

24.3.3.3 32-Bit Input Capture

Two 16-bit Timer/Counter Type B (TCBn) can be combined to work as a true 32-bit input capture.

One TCB is counting the two LSBs. Once this counter reaches MAX, an overflow (OVF) event is generated, and the counter wraps around. The second TCB is configured to count these OVF events and thus provides the two MSBs. The 32-bit counter value is concatenated from the two counter values.

To function as a 32-bit counter, the two TCBs and the system have to be set up as described in the following paragraphs.

System Configuration

- Configure a source (TCA, events, CLK_PER) for the count input for the LSB TCB, according to the application requirements
- Configure the Event System to route the OVF events from the LSB TCB (event generator) to the 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

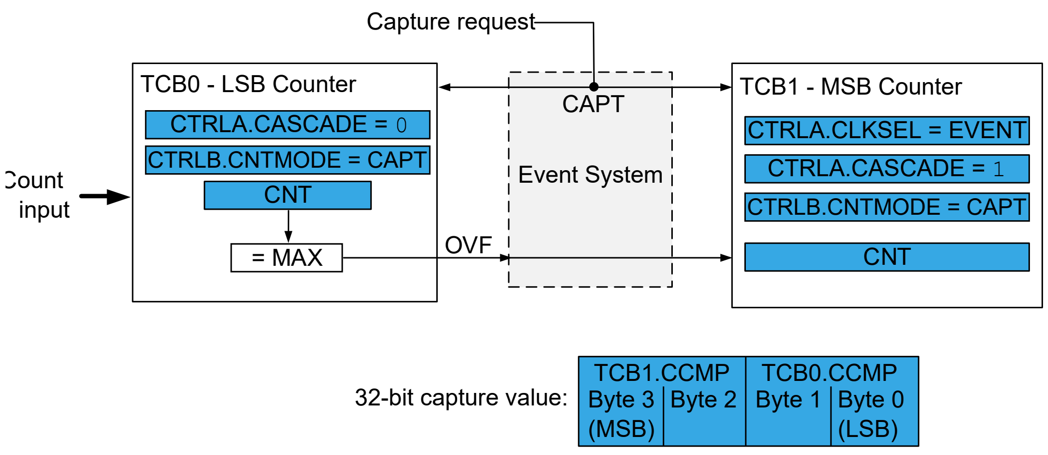
To acquire a 32-bit counter value, send a CAPT event to both TCBs. Both TCBs are running in the same Capture mode, so each will capture the current counter value (CNT) in the respective Capture/Compare (CCMP) register. The 32-bit capture value is formed by concatenating the two CCMP registers.

Example 24-1. Using TCB0 as LSB Counter and TCB1 as MSB Counter

TCB0 is counting the count input, and TCB1 is counting the OVF signals from TCB0. 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).



24.3.3.4 Noise Canceler

The Noise Canceler improves the noise immunity by using a simple digital filter scheme. When the Noise Filter (FILTER) bit in the Event Control (TCBn.EVCTRL) register is enabled, the peripheral monitors the event channel and keeps a record of the last four observed samples. If four consecutive samples are equal, the input is considered to be stable, and the signal is fed to the edge detector.

When enabled, the Noise Canceler introduces an additional delay of four peripheral clock cycles between a change applied to the input and the update of the Input Compare register.

The Noise Canceler uses the peripheral clock and is, therefore, not affected by the prescaler.

24.3.3.5 Synchronized with Timer/Counter Type A

The TCB can be configured to use the clock (CLK_TCA) of a Timer/Counter type A (TCAn) by writing to the Clock Select (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.

24.3.4 Events

The TCB can generate the events described in the following table:

Table 24-3. Event Generators in TCB

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
TCBn	CAPT	CAPT flag set	Pulse	CLK_PER	One CLK_PER period
	OVF	OVF flag set			

The conditions for generating the CAPT and OVF events are identical to those that will raise the corresponding interrupt flags in the Timer/Counter Interrupt Flags (TCBn.INTFLAGS) register. Refer to the *EVSYS - Event System* section for more details regarding event users and Event System configuration.

The TCB can receive the events described in the following table:

Table 24-4. Event Users and Available Event Actions in TCB

User Name		Description	Input Detection	Async/Sync
Peripheral	Input			
TCBn	CAPT	Time-Out Check Count mode	Edge	Sync
		Input Capture on Event Count mode		
		Input Capture Frequency Measurement Count mode		
		Input Capture Pulse-Width Measurement Count mode		
		Input Capture Frequency and Pulse-Width Measurement Count mode		
		Single-Shot Count mode		Both
	COUNT	Event as clock source in combination with a count mode		Sync

CAPT and COUNT are TCB event users that detect and act upon input events.

The COUNT event user is enabled on the peripheral by modifying the Clock Select (CLKSEL) bit field in the Control A (TCBn.CTRLA) register to EVENT and setting up the Event System accordingly.

If the Capture Event Input Enable (CAPTEI) bit in the Event Control (TCBn.EVCTRL) register is written to '1', incoming events will result in an event action as defined by the Event Edge (EDGE) bit in Event Control (TCBn.EVCTRL) register, and the Timer Mode (CNTMODE) bit field in Control B (TCBn.CTRLB) register. The event 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.

24.3.5 Interrupts

Table 24-5. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
CAPT	TCB interrupt	Depending on the operating mode. See the description of the CAPT bit in the TCBn.INTFLAG register
OVF		The timer/counter overflows from MAX to BOTTOM

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

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

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

24.3.6 Sleep Mode Operation

TCBn is, by default, disabled in Standby sleep mode. It will be halted as soon as the sleep mode is entered.

The module can stay fully operational in the Standby sleep mode if the Run Standby (RUNSTDBY) bit in the TCBn.CTRLA register is written to '`1`'.

All operations are halted in Power-Down sleep mode.

24.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		RUNSTDBY	CASCADE	SYNCUPD	CLKSEL[2:0]			ENABLE
0x01	CTRLB	7:0		ASYNC	CCMPINIT	CCMPEN		CNTMODE[2:0]		
0x02 ... 0x03	Reserved									
0x04	EVCTRL	7:0		FILTER		EDGE				CAPTEI
0x05	INTCTRL	7:0							OVF	CAPT
0x06	INTFLAGS	7:0							OVF	CAPT
0x07	STATUS	7:0								RUN
0x08	DBGCTRL	7:0								DBGRUN
0x09	TEMP	7:0	TEMP[7:0]							
0x0A	CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x0C	CCMP	7:0	CCMP[7:0]							
		15:8	CCMP[15:8]							

24.5 Register Description

24.5.1 Control A

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

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY	CASCADE	SYNCUPD	CLKSEL[2:0]			ENABLE
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 6 – RUNSTDBY Run Standby
Writing a '1' to this bit will enable the peripheral to run in Standby sleep mode.

Bit 5 – CASCADE Cascade Two Timer/Counters
Writing this bit to '1' enables cascading of two 16-bit Timer/Counters type B (TCBn) for 32-bit operation using the Event System. This bit must be '1' for the timer/counter used for the two Most Significant Bytes (MSBs). When this bit is '1', the selected event source for capture (CAPT) is delayed by one peripheral clock cycle. This compensates the carry propagation delay when cascading two counters via the Event System.

Bit 4 – SYNCUPD Synchronize Update
When this bit is written to '1', the TCB will restart whenever TCAn is restarted or overflows. This can be used to synchronize capture with the PWM period. If TCAn is selected as the clock source, the TCB will restart when that TCAn is restarted. For other clock selections, it will restart together with TCA0.

Bits 3:1 – CLKSEL[2:0] Clock Select
Writing these bits selects the clock source for this peripheral.

Value	Name	Description
0x0	DIV1	CLK_PER
0x1	DIV2	CLK_PER / 2
0x2	TCA0	CLK_TCA from TCA0
0x3-0x6	-	Reserved
0x07	EVENT	Positive edge on event input

Bit 0 – ENABLE Enable
Writing this bit to '1' enables the Timer/Counter type B peripheral.

24.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		ASYNC	CCMPINIT	CCMPEN		CNTMODE[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – ASYNC Asynchronous Enable

Writing this bit to ‘1’ will allow asynchronous updates of the TCB output signal in Single-Shot mode.

Value	Description
0	The output will go HIGH when the counter starts after synchronization
1	The output will go HIGH when an event arrives

Bit 5 – CCMPINIT Compare/Capture Pin Initial Value

This bit is used to set the initial output value of the pin when a pin output is used. This bit has no effect in 8-bit PWM mode and Single-Shot mode.

Value	Description
0	Initial pin state is LOW
1	Initial pin state is HIGH

Bit 4 – CCMPEN Compare/Capture Output Enable

Writing this bit to ‘1’ enables the waveform output. This will make the waveform output available on the corresponding pin 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

Writing to this bit field selects the Timer mode.

Value	Name	Description
0x0	INT	Periodic Interrupt mode
0x1	TIMEOUT	Time-Out Check mode
0x2	CAPT	Input Capture on Event mode
0x3	FRQ	Input Capture Frequency Measurement mode
0x4	PW	Input Capture Pulse-Width Measurement mode
0x5	FRQPW	Input Capture Frequency and Pulse-Width Measurement mode
0x6	SINGLE	Single-Shot mode
0x7	PWM8	8-Bit PWM mode

24.5.3 Event Control

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

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

Bit 6 – FILTER Input Capture Noise Cancellation Filter
Writing this bit to '1' enables the Input Capture Noise Cancellation unit.

Bit 4 – EDGE Event Edge
This bit is used to select the event edge. The effect of this bit is dependent on the selected Count Mode (CNTMODE) bit field in TCBn.CTRLB. "—" means that an event or edge has no effect in this mode.

Count Mode	EDGE	Positive Edge	Negative Edge
Periodic Interrupt mode	0	—	—
	1	—	—
Timeout Check mode	0	Start counter	Stop counter
	1	Stop counter	Start counter
Input Capture on Event mode	0	Input Capture, interrupt	—
	1	—	Input Capture, interrupt
Input Capture Frequency Measurement mode	0	Input Capture, clear and restart counter, interrupt	—
	1	—	Input Capture, clear and restart counter, interrupt
Input Capture Pulse-Width Measurement mode	0	Clear and restart counter	Input Capture, interrupt
	1	Input Capture, interrupt	Clear and restart counter
Input Capture Frequency and Pulse Width Measurement mode	0	• On the 1st Positive: Clear and restart counter • 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 this bit to '1' enables the input capture event.

24.5.4 Interrupt Control

Name:	INTCTRL
Offset:	0x05
Reset:	0x00
Property:	-

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

Bit 1 – OVF Overflow Interrupt Enable
Writing this bit to '1' enables interrupt on overflow.

Bit 0 – CAPT Capture Interrupt Enable
Writing this bit to '1' enables interrupt on capture.

24.5.5 Interrupt Flags

Name:	INTFLAGS
Offset:	0x06
Reset:	0x00
Property:	-

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

Bit 1 – OVF Overflow Interrupt Flag

This bit is set when an overflow interrupt occurs. The flag is set whenever the timer/counter wraps from MAX to BOTTOM.

The bit is cleared by writing a '1' to the bit position.

Bit 0 – CAPT Capture Interrupt Flag

This bit is set when a capture interrupt occurs. The interrupt conditions are dependent on the Counter Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) register.

This bit is cleared by writing a '1' to it or when the Capture register is read in Capture mode.

Table 24-6. Interrupt Sources Set Conditions by Counter Mode

Counter Mode	Interrupt Set Condition	TOP Value	CAPT
Periodic Interrupt mode	Set when the counter reaches TOP	CCMP	CNT == TOP
Timeout Check mode	Set when the counter reaches TOP		
Single-Shot mode	Set when the counter reaches TOP		
Input Capture Frequency Measurement mode	Set on edge when the Capture register is loaded and the counter restarts; the flag clears when the capture is read	--	On Event, copy CNT to CCMP, and restart counting (CNT == BOTTOM)
Input Capture on Event mode	Set when an event occurs and the Capture register is loaded; the flag clears when the capture is read		On Event, copy CNT to CCMP, and continue counting
Input Capture Pulse-Width Measurement mode	Set on edge when the Capture register is loaded; the previous edge initialized the count; the flag clears when the capture is read		
Input Capture Frequency and Pulse-Width Measurement mode	Set on the second edge (positive or negative) when the counter is stopped; the flag clears when the capture is read		
8-bit PWM mode	Set when the counter reaches CCMH	CCML	CNT == CCMH

24.5.6 Status

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

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

Bit 0 – RUN Run

When the counter is running, this bit is set to '1'. When the counter is stopped, this bit is cleared to '0'.

The bit is read-only and cannot be set by UPDI.

24.5.7 Debug Control

Name: DBGCTRL
Offset: 0x08
Reset: 0x00
Property: -

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

Bit 0 – DBGRUN Debug Run

Value	Description
0	The peripheral is halted in Break Debug mode and ignores events
1	The peripheral will continue to run in Break Debug mode when the CPU is halted

24.5.8 Temporary Value

Name:	TEMP
Offset:	0x09
Reset:	0x00
Property:	-

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section.

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

Bits 7:0 – TEMP[7:0] Temporary Value

24.5.9 Count

Name:	CNT
Offset:	0x0A
Reset:	0x00
Property:	-

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

CPU and UPDI write access has priority over internal updates of the register.

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

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

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

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

24.5.10 Capture/Compare

Name:	CCMP
Offset:	0x0C
Reset:	0x00
Property:	-

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

This register has different functions depending on the mode of operation:
- For Capture operation, these registers contain the captured value of the counter at the time the capture occurs
- In Periodic Interrupt, Time-Out 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
These bits hold the MSB of the 16-bit compare, capture and top value.

Bits 7:0 – CCMP[7:0] Capture/Compare Value Low Byte
These bits hold the LSB of the 16-bit compare, capture and top value.

25. USART - Universal Synchronous and Asynchronous Receiver and Transmitter

25.1 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

25.2 Overview

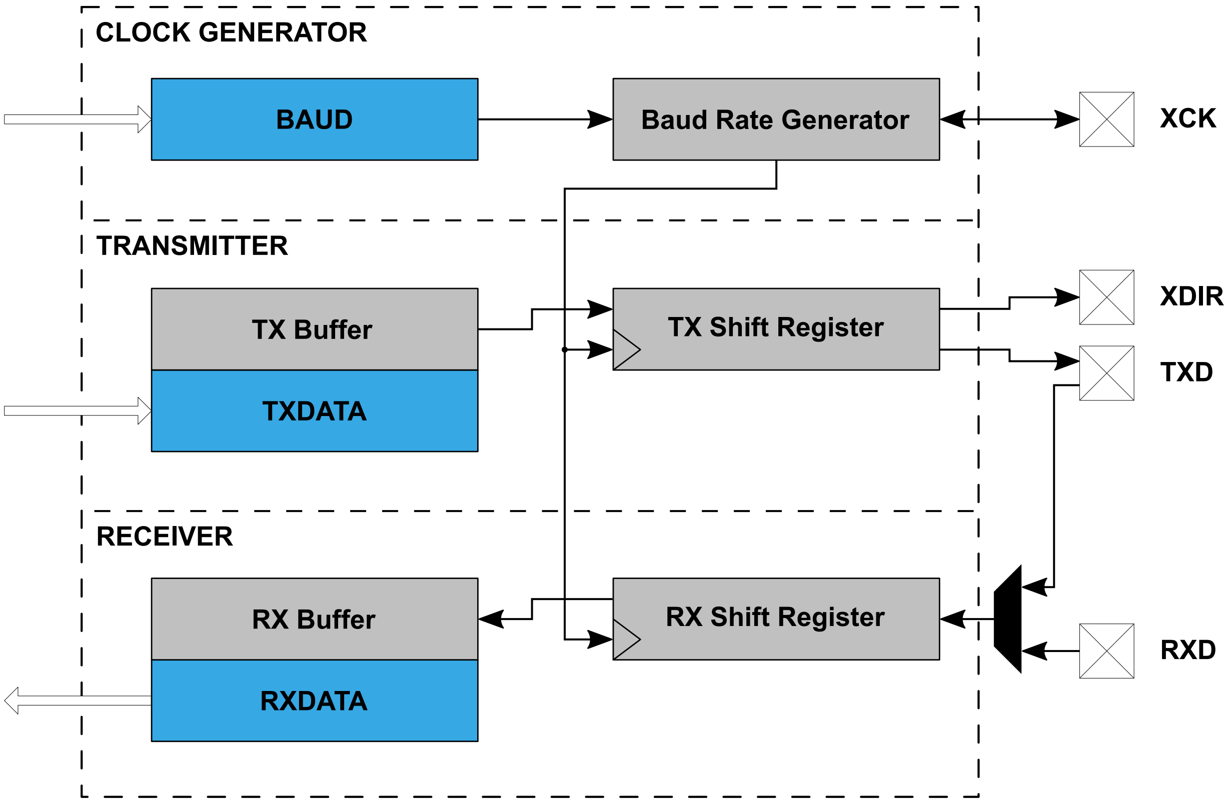
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a fast and flexible serial communication peripheral. The USART supports several different modes of operation that can accommodate multiple types of applications and communication devices. For example, the One-Wire Half-Duplex 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.

25.2.1 Block Diagram

Figure 25-1. USART Block Diagram



25.2.2 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

25.3 Functional Description

25.3.1 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

25.3.2 Operation

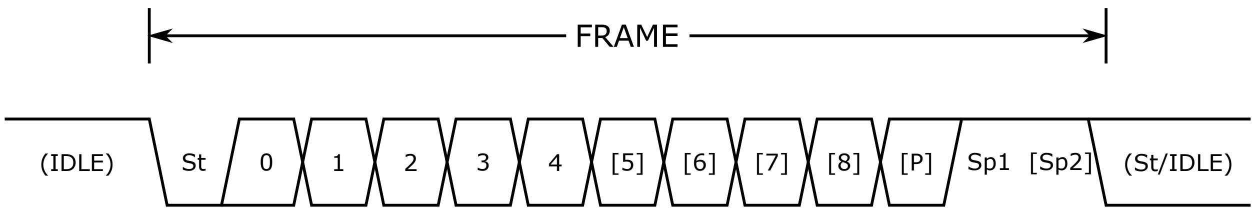
25.3.2.1 Frame Formats

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 25-2. Frame Formats

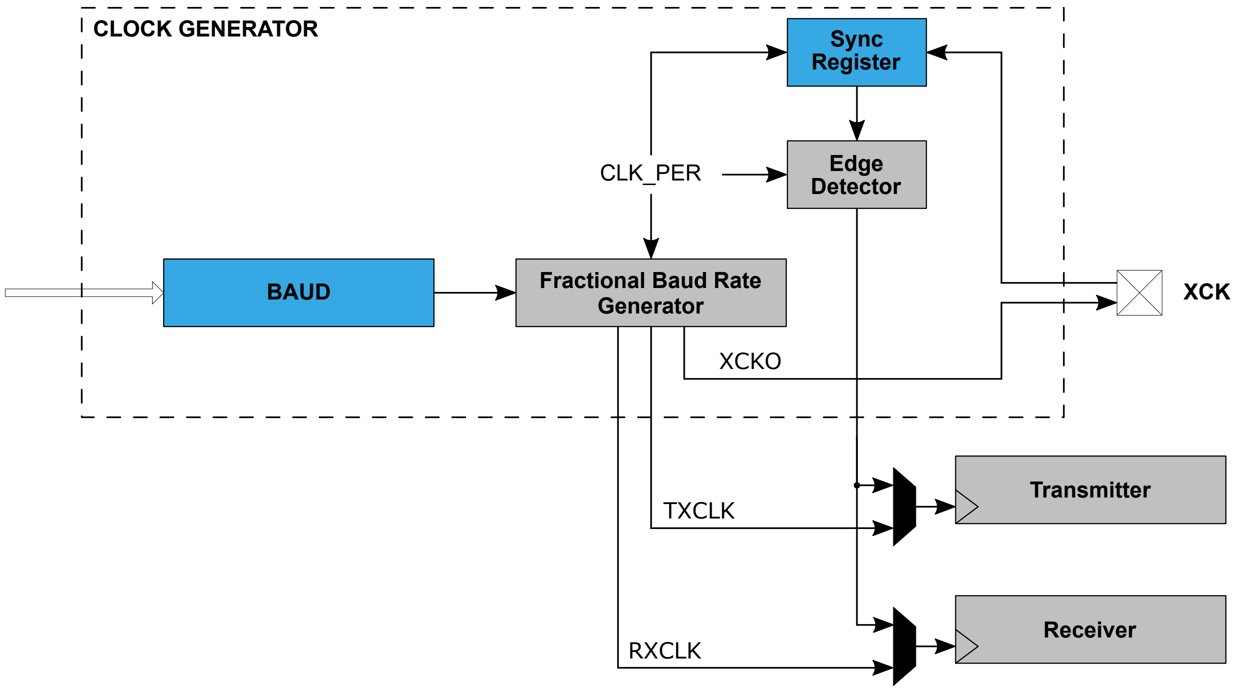


St	Start bit, always low
(n)	Data bits (0 to 8)
P	Parity bit, may be odd or even
Sp	Stop bit, always high
IDLE	No transfer on the communication line (RxD or TxD). The Idle state is always high.

25.3.2.2 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 25-3. Clock Generation Logic Block Diagram



25.3.2.2.1 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 25-1. 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 25-1. 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

25.3.2.3 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.

25.3.2.3.1 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.

25.3.2.4 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.

25.3.2.4.1 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.

25.3.2.4.2 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.

25.3.2.4.3 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.

25.3.3 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.

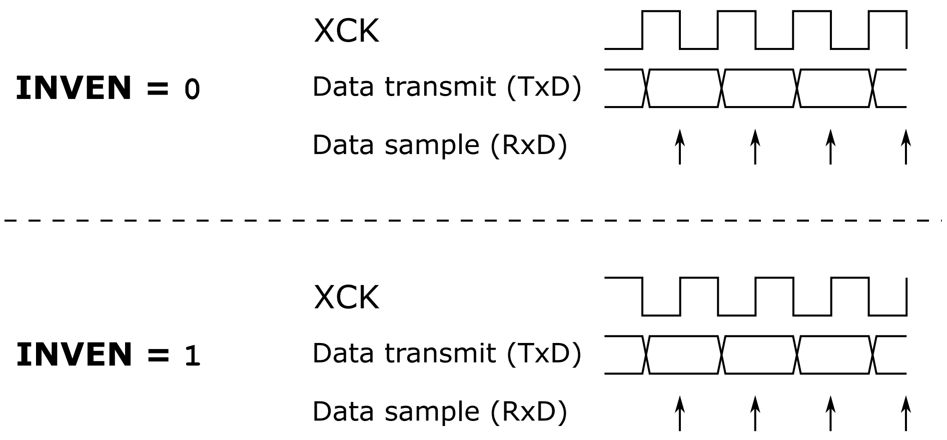
Four different modes of reception are available when communicating asynchronously. One of these modes can receive transmissions at twice the normal speed, sampling only eight times per bit instead of the normal 16. The other three operating modes use variations of synchronization logic, all receiving at normal speed.

25.3.3.1 Synchronous Operation

25.3.3.1.1 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 25-4. 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.

25.3.3.1.2 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.

25.3.3.1.3 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.

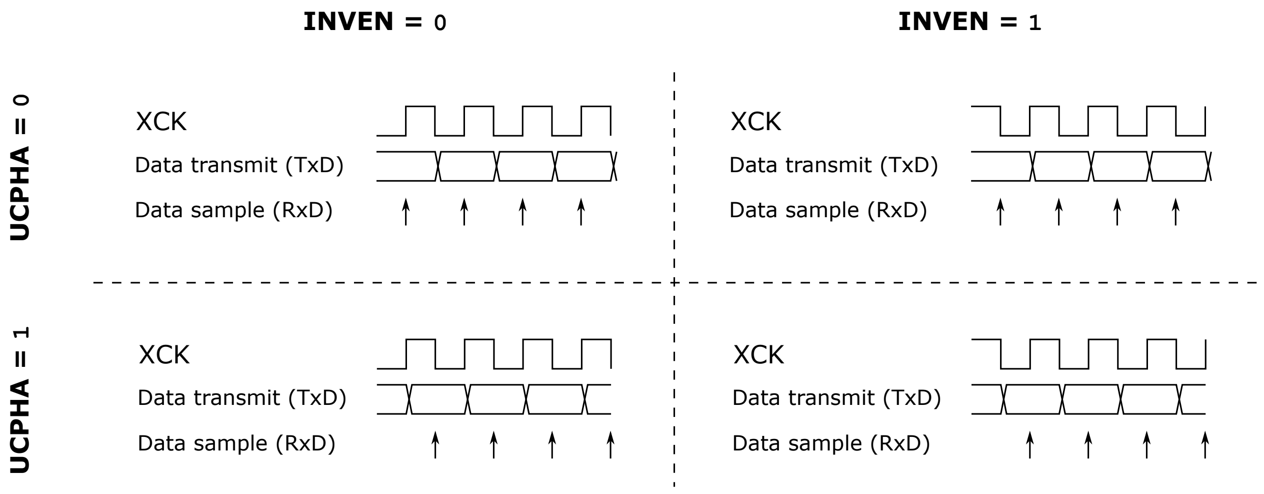
SPI does not use Start, Stop, or Parity bits, so the transmission frame can only consist of the Data bits.

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 25.3.2.2.1. The Fractional Baud Rate Generator.

Each Data bit is transmitted by pulling the data line high or low for one full clock period. The receiver will sample bits in the middle of the transmitter hold period, as shown in the figure below. It also shows how the timing scheme can be configured using the Inverted I/O Enable (INVEN) bit in the PORTx.PINnCTRL register and the USART Clock Phase (UCPHA) bit in the USARTn.CTRLC register.

Figure 25-5. Data Transfer Timing Diagrams



The table below further explains the figure above.

Table 25-2. 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 25.3.2.3. 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 25.3.2.4. 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 25-3. 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
-	$\overline{\text{SS}}$	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.

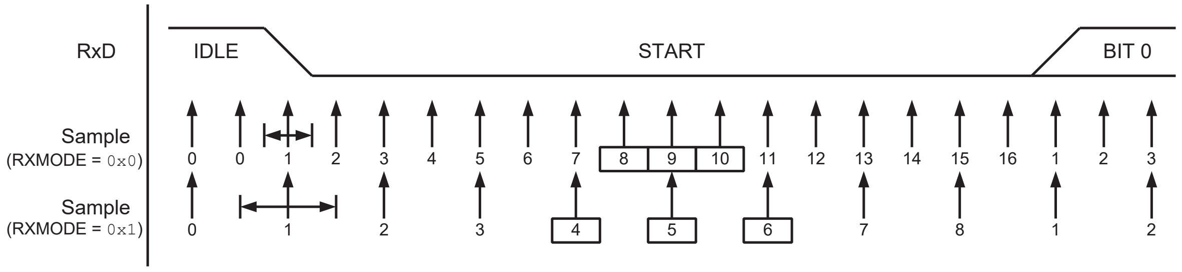
25.3.3.2 Asynchronous Operation

25.3.3.2.1 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 25.3.3.2.4. Double-Speed Operation for more details). The horizontal arrows show the maximum synchronization error. Note that the maximum synchronization error is larger in Double-Speed mode.

Figure 25-6. Start Bit Sampling

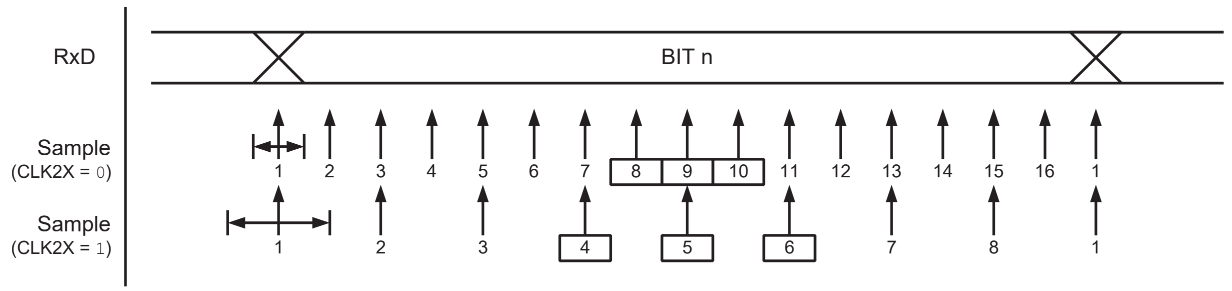


When the clock recovery logic detects a falling edge from the Idle (high) state to the Start bit (low), the Start bit detection sequence is initiated. In the figure above, sample 1 denotes the first sample reading '`0`'. The clock recovery logic then uses three subsequent samples (samples 8, 9, and 10 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.

25.3.3.2.2 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.

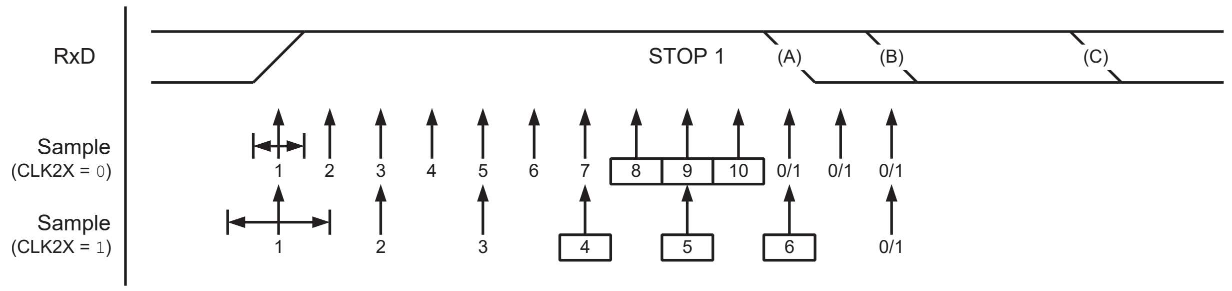
Figure 25-7. Sampling of Data and Parity Bits



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 25-8. 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.

25.3.3.2.3 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 25-4. 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 25-5. 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.

$$R_{SLOW} = \frac{S(D+1)}{S(D+1)+S_F-1} \qquad R_{FAST} = \frac{S(D+2)}{S(D+1)+S_M}$$

- 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

25.3.3.2.4 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`.

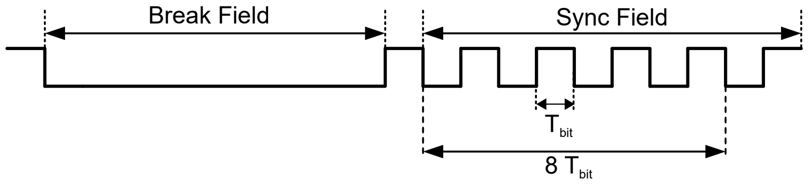
When enabled, the baud rate for a given asynchronous baud rate setting will be doubled, as shown in the equations in 25.3.2.2.1. The Fractional Baud Rate Generator. In this mode, the receiver will use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery. This requires a more accurate baud rate setting and peripheral clock. See 25.3.3.2.3. Error Tolerance for more details.

25.3.3.2.5 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 25-9. 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.

25.3.3.2.6 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.

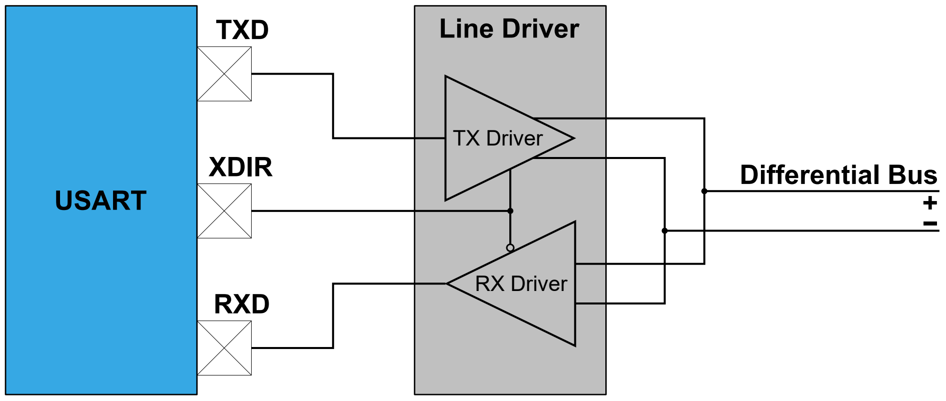
When the USART is transmitting to the TxD/RxD line, it will also receive its transmission. This can be used to detect overlapping transmissions by checking if the received data are the same as the transmitted data.

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'.

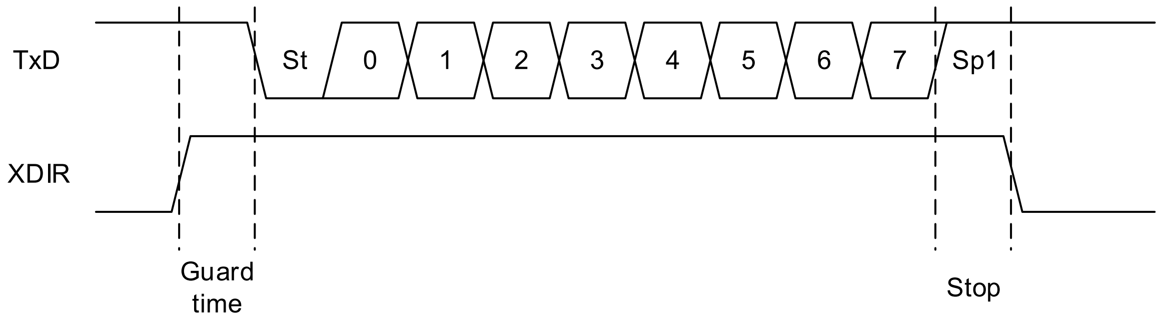
The RS-485 mode supports external line driver devices that convert a single USART transmission into corresponding differential pair signals. It implements automatic control of the XDIR pin that can be used to enable transmission or reception for the line driver device. The USART automatically drives 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 25-10. 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).

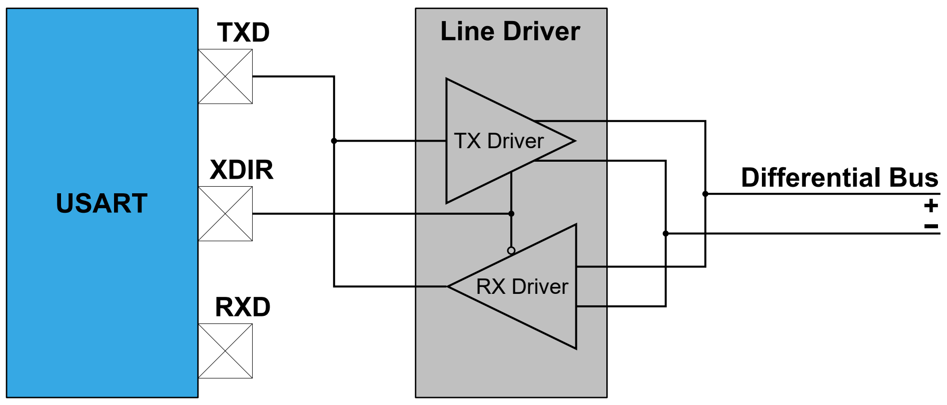
Figure 25-11. XDIR Drive Timing



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.

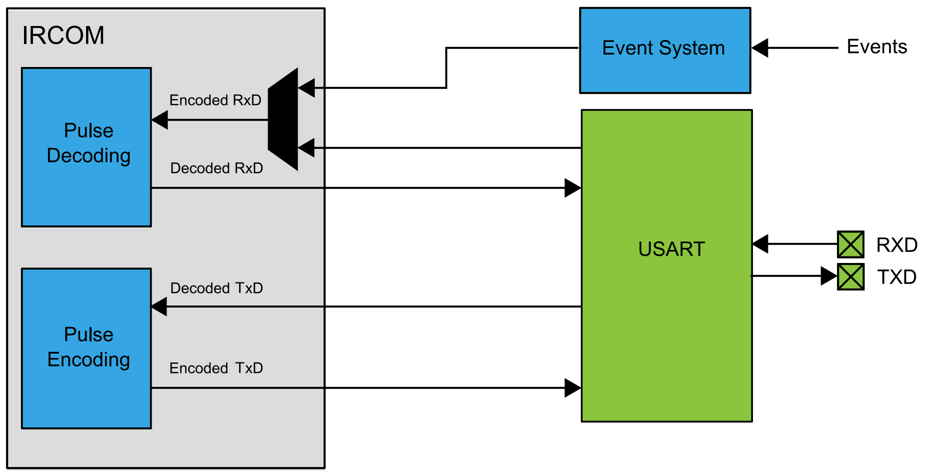
Figure 25-12. RS-485 with Loop-Back Mode Connection



25.3.3.2.7 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 25-13. 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.

25.3.4 Additional Features

25.3.4.1 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).

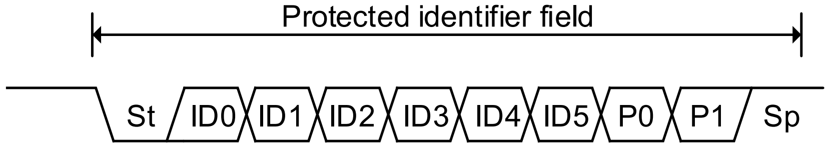
When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the Parity bit of the corresponding frame. If a parity error is detected, the Parity Error flag (the PERR bit in the USARTn.RXDATAH register) is set.

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 25-14. Protected Identifier Field and Mapping of Identifier and Parity Bits



25.3.4.2 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 25-6. USART Start Frame Detection Modes

SFDEN	RXSIF Interrupt	RXCIF Interrupt	Comment
0	x	x	Standard mode

...........continued

SFDEN	RXSIF Interrupt	RXCIF Interrupt	Comment
1	Disabled	Disabled	Only the oscillator is powered during the frame reception. If the interrupts are disabled and buffer overflow is ignored, all incoming frames will be lost
1	Disabled	Enabled	System/all clocks are awakened on Receive Complete interrupt
1	Enabled	x	System/all clocks are awakened when a Start bit is detected

Note: The `SLEEP` instruction will not shut down the oscillator if there is ongoing communication.

25.3.4.3 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.

25.3.4.3.1 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.

25.3.5 Events

The USART can generate the events described in the table below.

Table 25-7. 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 25-8. Event Users in USART

User Name		Description	Input Detection	Async/Sync
Peripheral	Input			
USARTn	IREI	USARTn IrDA event input	Pulse	Sync

25.3.6 Interrupts

Table 25-9. 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.

25.4 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]						

25.5 Register Description

25.5.1 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

25.5.2 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 25.3.4.1. 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.

25.5.3 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

25.5.4 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

25.5.5 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.

25.5.6 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

25.5.7 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 25.3.2.4.2. Disabling the Receiver for more information.

Value	Description
0	The USART receiver is disabled
1	The USART receiver is enabled

Bit 6 – TXEN Transmitter Enable

This bit controls whether the USART transmitter is enabled or not. Refer to 25.3.2.3.1. Disabling the Transmitter for more information.

Value	Description
0	The USART transmitter is disabled
1	The USART transmitter is enabled

Bit 4 – SFDEN Start-of-Frame Detection Enable

This bit controls whether the USART Start-of-Frame Detection mode is enabled or not. Refer to 25.3.4.2. Start-of-Frame Detection for more information.

Value	Description
0	The USART Start-of-Frame Detection mode is disabled
1	The USART Start-of-Frame Detection mode is enabled

Bit 3 – ODME Open Drain Mode Enable

This bit controls whether Open Drain mode is enabled or not. See 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 25.3.3.2.4. 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 25.3.4.3. Multiprocessor Communication for more information.

Value	Description
0	Multi-Processor Communication mode is disabled
1	Multi-Processor Communication mode is enabled

25.5.8 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 25.3.4.1. Parity for more information.

Value	Name	Description
0x0	DISABLED	Disabled
0x1	-	Reserved
0x2	EVEN	Enabled, even parity
0x3	ODD	Enabled, odd parity

Bit 3 – SBMODE Stop Bit Mode
This bit selects the number of Stop bits to be inserted by the transmitter.
The receiver ignores this setting.

Value	Description
0	1 Stop bit
1	2 Stop bits

Bits 2:0 – CHSIZE[2:0] Character Size
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)
0x07	9BITH	9-bit (High byte first)

25.5.9 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 25.3.3.1.3. 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

25.5.10 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 25-1.

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.

25.5.11 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)

25.5.12 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

25.5.13 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 25.3.3.2.7. IRCOM Mode of Operation for more information.

Value	Description
0	IrDA Event input is disabled
1	IrDA Event input is enabled

25.5.14 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.

25.5.15 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.

26. SPI - Serial Peripheral Interface

26.1 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

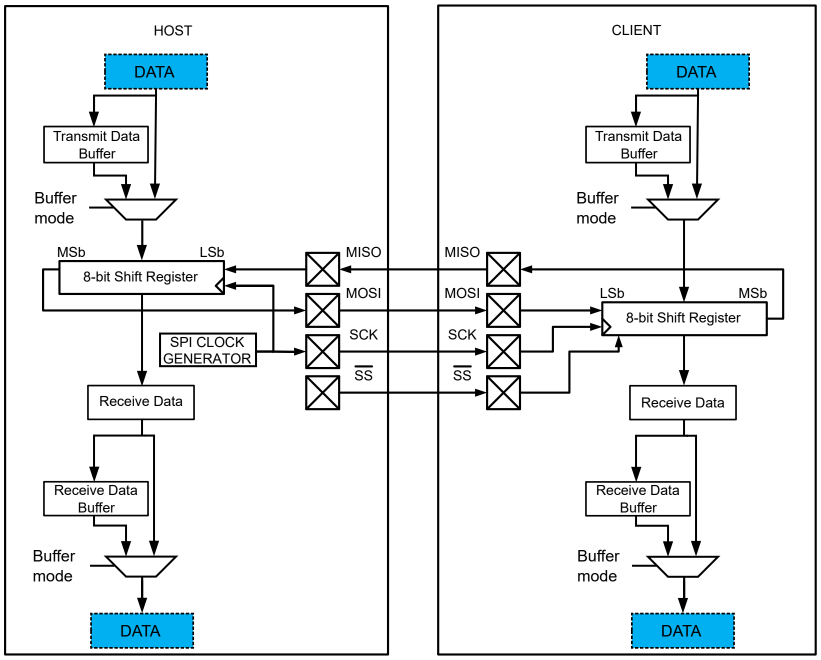
26.2 Overview

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.

26.2.1 Block Diagram

Figure 26-1. 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.

26.2.2 Signal Description

Table 26-1. 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 26-1 is overridden.

26.3 Functional Description

26.3.1 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.

26.3.2 Operation

26.3.2.1 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.

26.3.2.1.1 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.

26.3.2.1.2 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.

26.3.2.1.3 $\overline{SS}$ 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 26-2 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 26-2. Overview of the $\overline{\text{SS}}$ Pin Functionality When the SSD Bit in SPIn.CTRLB Is '`0`'

$\overline{\text{SS}}$ Configuration	$\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ line, it must be set by the application to re-enable the SPI Host mode.

26.3.2.2 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.

26.3.2.2.1 Normal Mode

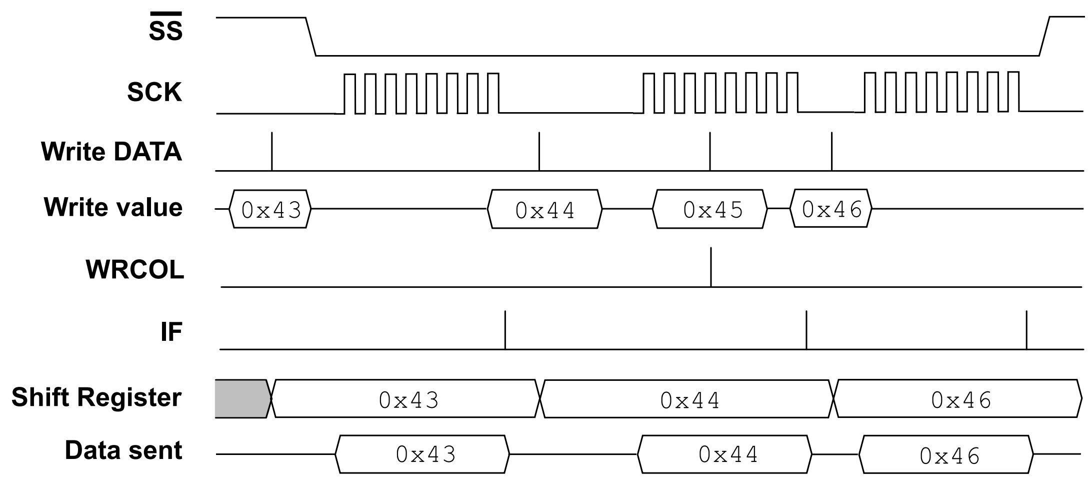
In Normal mode, the SPI peripheral will remain Idle as long as the $\overline{\text{SS}}$ pin is driven high. In this state, the software may update the contents of the DATA register, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{\text{SS}}$ pin is driven low. If the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ 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 26-2 shows a transmission sequence in Normal mode. Notice how the value `0x45` is written to the DATA register but never transmitted.

Figure 26-2. 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.

26.3.2.2.2 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 26-1 shows the extra buffers.

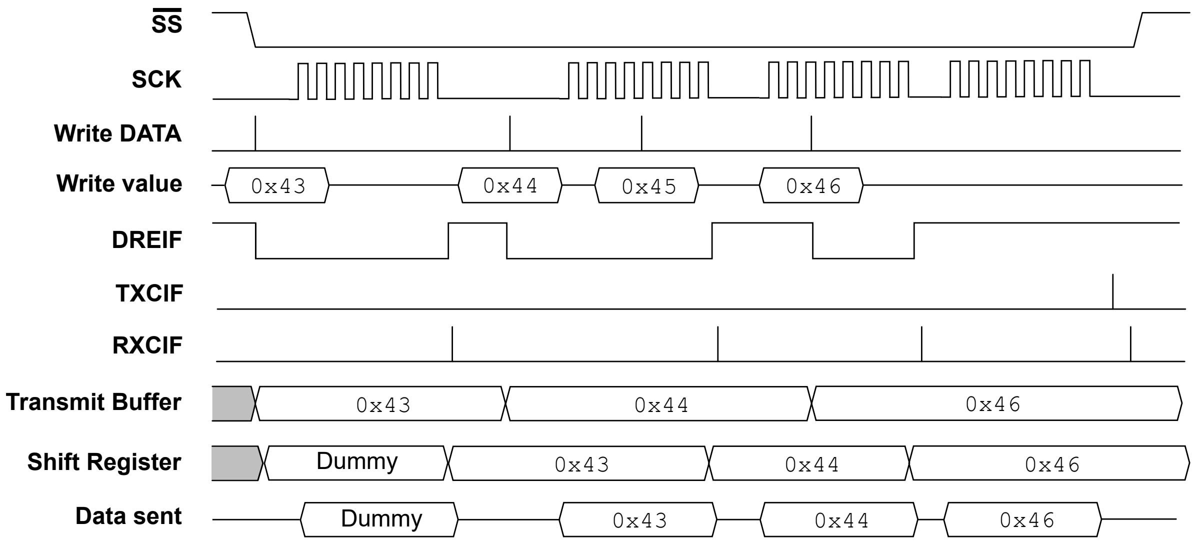
When Buffer mode is enabled it can work in two different ways. The Buffer Mode Wait for Receive (BUFWR) bit in the Control B (SPIn.CTRLB) register controls how the Buffer mode works. The details of how they work including timing diagrams are described below.

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 26-3 shows a transmission sequence with this configuration. Notice how the value `0x45` is written to the Data (SPIn.DATA) register but never transmitted.

Figure 26-3. SPI Timing Diagram in Buffer Mode with BUFWR in SPIn.CTRLB Written to '0'



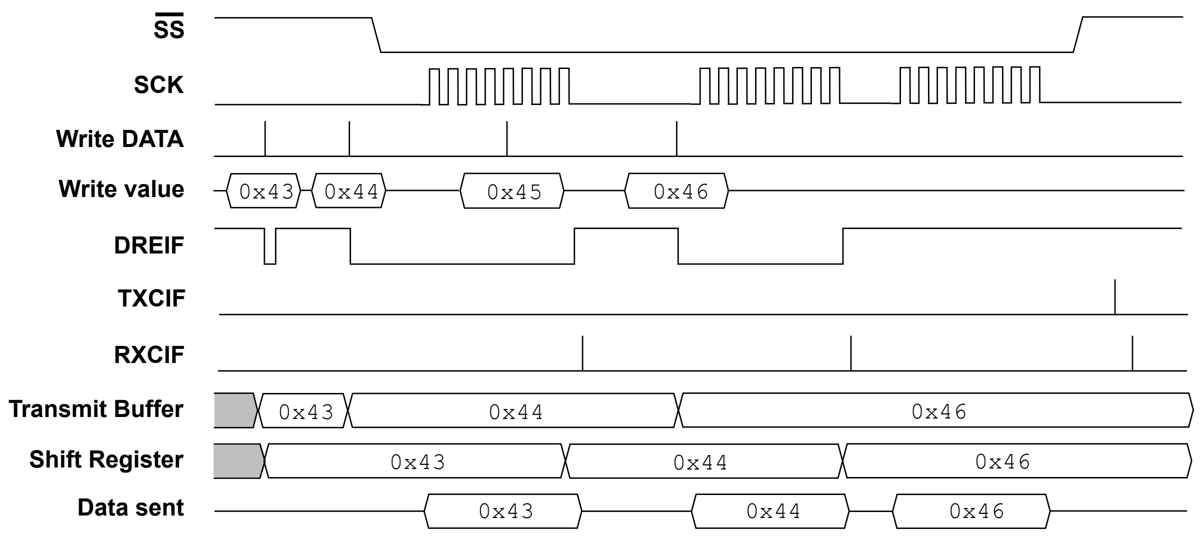
When the Wait for Receive (BUFWR) bit in SPIn.CTRLB is written to '0', all writes to the Data (SPIn.DATA) register go to the Transmit Data Buffer register. The figure above shows that the value 0x43 is written to the Data (SPIn.DATA) register but not immediately transferred to the shift register, so 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 $\overline{\text{SS}}$ pin is driven low. Figure 26-4 shows a transmission sequence with this configuration. Notice how the value 0x45 is written to the Data (SPIn.DATA) register but never transmitted.

Figure 26-4. 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`'.

26.3.2.2.3 $\overline{SS}$ 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 26-3 shows an overview of the $\overline{SS}$ pin functionality.

Table 26-3. 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.

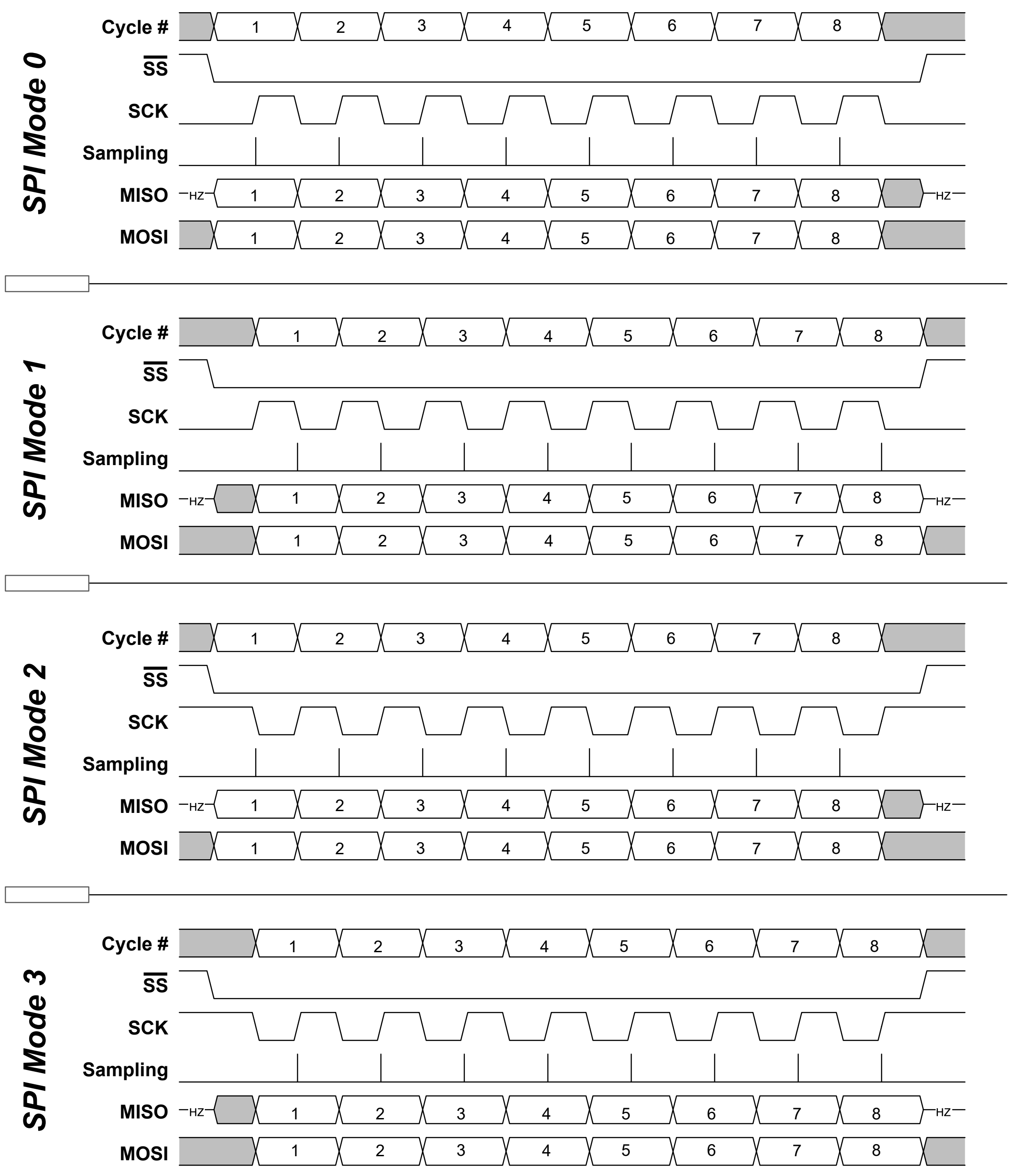
26.3.2.3 Data Modes

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 26-5. SPI Data Transfer Modes



26.3.2.4 Events

The SPI can generate the following events:

Table 26-4. 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.

26.3.2.5 Interrupts

Table 26-5. 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.

26.4 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]							

26.5 Register Description

26.5.1 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 $\overline{\text{SS}}$ 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

26.5.2 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 $\overline{\text{SS}}$ 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), $\overline{\text{SS}}$ 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

26.5.3 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.

26.5.4 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 $\overline{\text{SS}}$ 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.

26.5.5 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.

26.5.6 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.

27. TWI - Two-Wire Interface

27.1 Features

- Two-Wire Communication Interface
- Philips I^{2}C 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 Host and Client Operation
  - Combined (same 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

27.2 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).

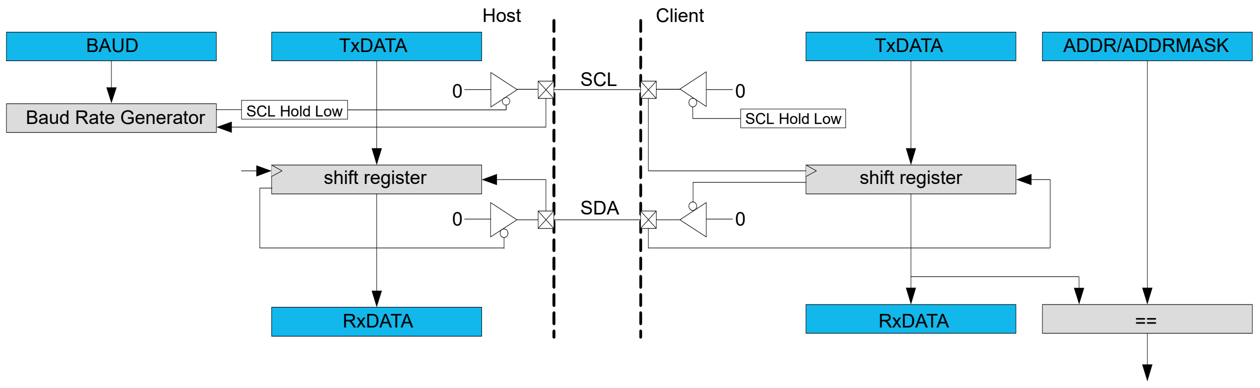
The TWI bus connects one or several client devices to one or several host devices. Any device connected to the bus can act as a host, a client, or both. The host generates the SCL using a Baud Rate Generator (BRG) and initiates data transactions by addressing one client and telling whether it wants to 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 Quick Command mode, where the host can address a client without exchanging data.

27.2.1 Block Diagram

Figure 27-1. TWI Block Diagram



27.2.2 Signal Description

Signal	Description	Type
SCL	Serial Clock Line	Digital I/O
SDA	Serial Data Line	Digital I/O

27.3 Functional Description

27.3.1 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.

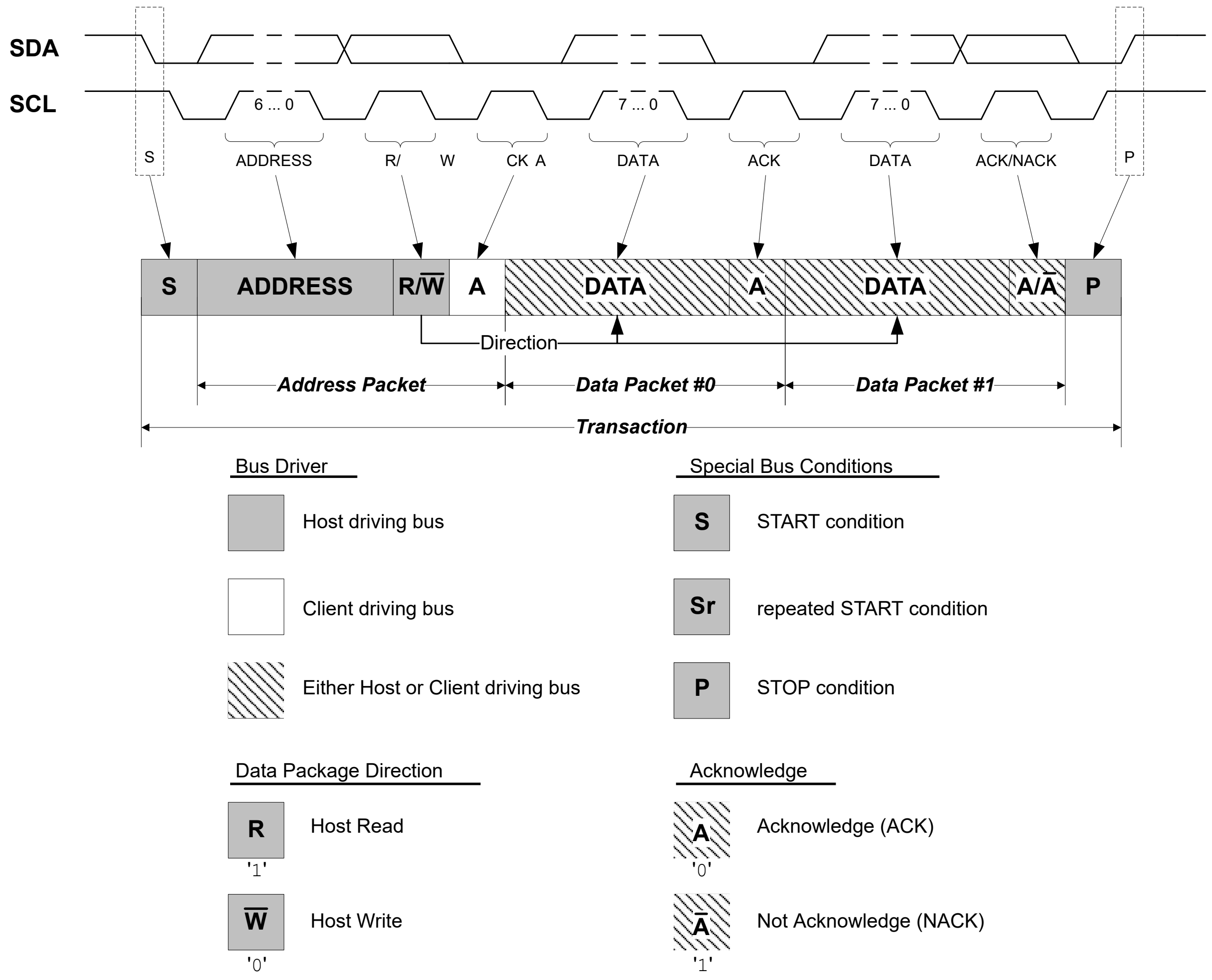
A unique address is assigned to each client device connected to the bus, and the host will use it to control the client and initiate a transaction. Several hosts can connect to the same bus, called a multi-host environment. An arbitration mechanism is provided for resolving bus ownership among hosts 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^{2}C 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 27-2. Basic TWI Transaction Diagram Topology for a 7-Bit Address Bus



27.3.2 TWI Basic Operation

27.3.2.1 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

27.3.2.1.1 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.

27.3.2.1.2 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.

27.3.2.2 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.

27.3.2.2.1 Clock Generation

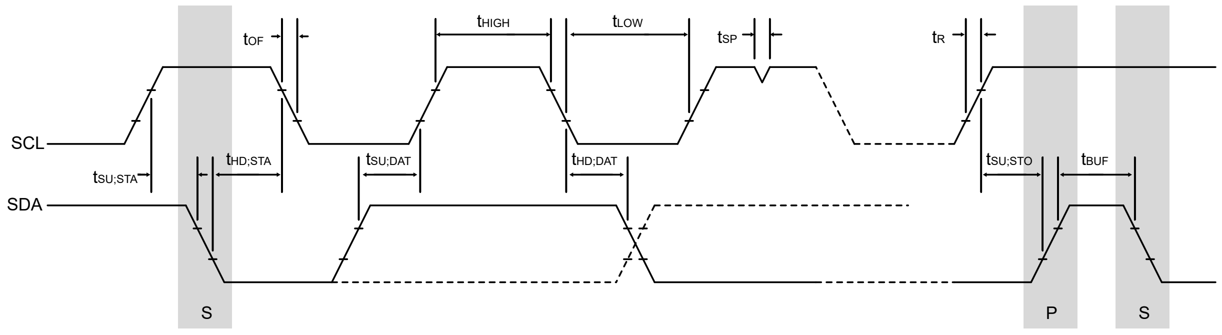
The TWI supports several transmission modes with different frequency limitations:

- 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 27-3. 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:

$$f_{\mathrm{SCL}} = \frac{1}{t_{\mathrm{LOW}} + t_{\mathrm{HIGH}} + t_{\mathrm{OF}} + t_{\mathrm{R}}}[\mathrm{Hz}]$$

The SCL clock is designed to have a 50/50 duty cycle, where the low period of the duty cycle comprises t_{OF} and t_{LOW}. t_{HIGH} will not start until a high state of SCL has been detected. The BAUD bit field in the TWIn.MBAUD register and the SCL frequency are related by the following formula:

$$f_{SCL} = \frac{f_{CLK\_PER}}{10 + 2 \times BAUD + f_{CLK\_PER} \times t_R} \qquad (1)$$

Equation 1 can be transformed to express BAUD:

$$BAUD = \frac{f_{CLK\_PER}}{2 \times f_{SCL}} - \left(5 + \frac{f_{CLK\_PER} \times t_R}{2}\right) \qquad (2)$$

Calculation of the BAUD Value

To ensure operation within the specifications of the desired speed mode (Sm, Fm, Fm+), follow these steps:

1. Calculate a value for the BAUD bit field using equation 2
2. Calculate t_{LOW} using the BAUD value from step 1:

$$t_{LOW} = \frac{BAUD + 5}{f_{CLK\_PER}} - t_{OF} \qquad (3)$$

3. Check if your t_{LOW} from equation 3 is above the specified minimum of the desired mode (t_{LOW_Sm}= 4700 ns, t_{LOW_Fm}= 1300 ns, t_{LOW_Fm+}= 500 ns)
   - If the calculated t_{LOW} is above the limit, use the BAUD value from equation 2
   - If the limit is not met, calculate a new BAUD value using equation 4, below, where t_{LOW_mode} is either t_{LOW_Sm}, t_{LOW_Fm}, or t_{LOW_Fm+} from the mode specifications:

$$BAUD = f_{CLK\_PER} \times (t_{LOW\_mode} + t_{OF}) - 3 \qquad (4)$$

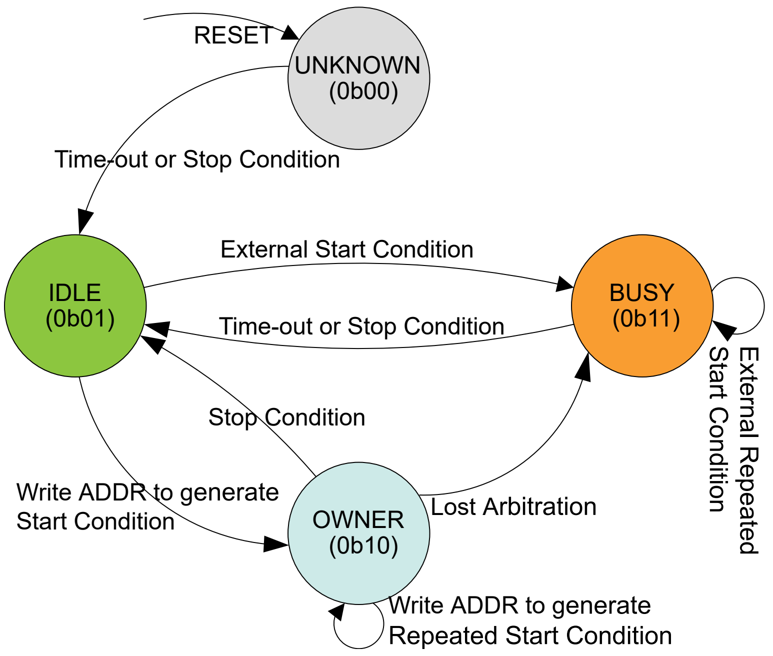
27.3.2.2.2 TWI Bus State Logic

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 27-4. Bus State Diagram


RESET
UNKNOWN (0b00)
Time-out or Stop Condition
IDLE (0b01)
External Start Condition
BUSY (0b11)
Time-out or Stop Condition
External Repeated Start Condition
Stop Condition
Write ADDR to generate Start Condition
OWNER (0b10)
Lost Arbitration
Write ADDR to generate Repeated Start Condition

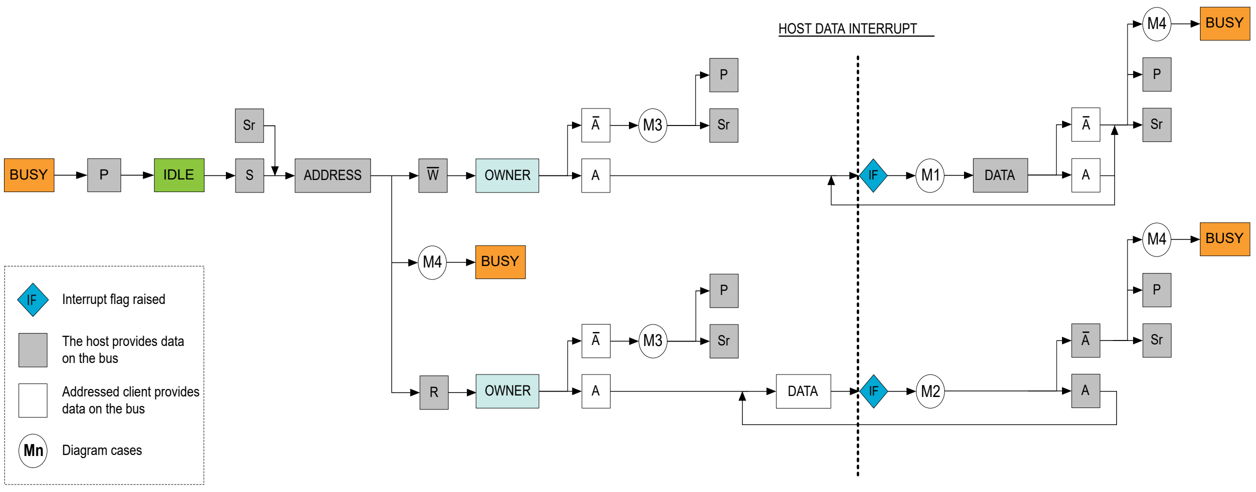
1. **Unknown**: The bus state machine is active when the TWI host is enabled. After enabling the TWI host, performing a system Reset, or disabling the TWI host, the bus state is Unknown.
2. **Idle**: The bus state machine can be forced to enter the Idle state by writing `0x1` to the Bus State (BUSSTATE) 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.

27.3.2.2.3 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 27-5. 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^{2}C 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

27.3.2.2.4 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.

27.3.2.2.5 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.

27.3.2.3 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/$\overline{\text{W}}$ 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.

27.3.2.3.1 Receiving Address Packets

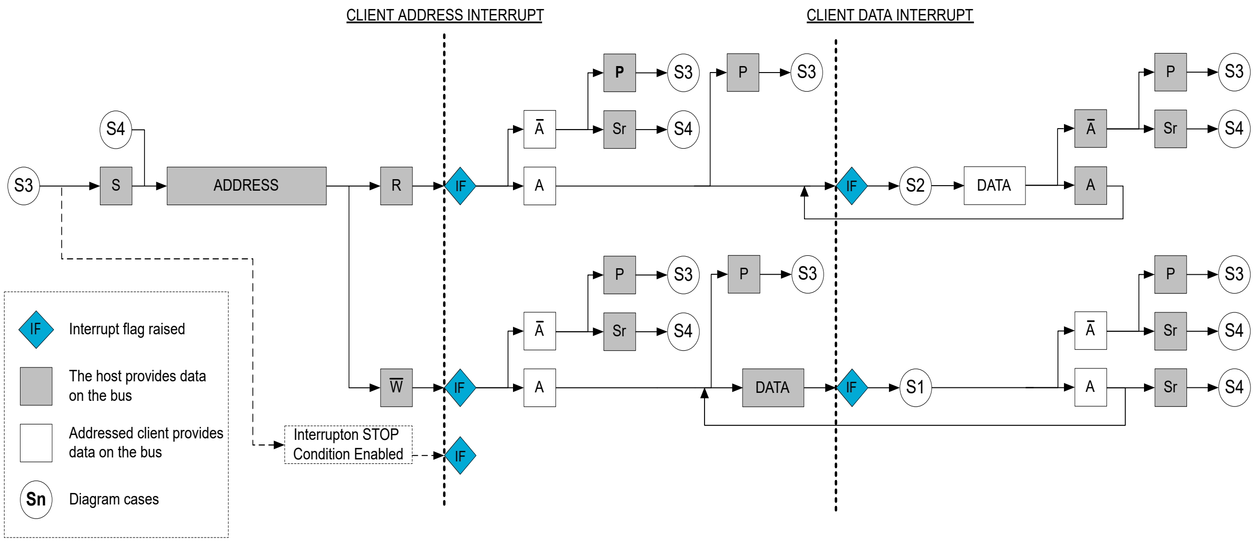
When the TWI is configured as a client, it will wait for a Start condition to be detected. When this happens, the successive address packet will be received and checked by the address match logic. The client will ACK a correct address and store the address in the Client Data (TWIn.SDATA) register. If 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'

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 27-6. 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.

27.3.2.3.2 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

27.3.2.3.3 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.

27.3.3 Additional Features

27.3.3.1 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 recommended to write to the Host Baud Rate (TWIn.MBAUD) register before setting the time-out because it is dependent on the baud rate setting.

A frequency of 100 kHz can be used for the SMBus environment. For the Standard mode (Sm) and Fast mode (Fm), the operating frequency has slew rate limited output, while for the Fast mode Plus (Fm+), it has x10 output drive strength.

The TWI also allows for an SMBus compatible SDA hold time configured in the SDA Hold Time (SDAHOLD) bit field from the Control A (TWIn.CTRLA) register.

27.3.3.1.1 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:

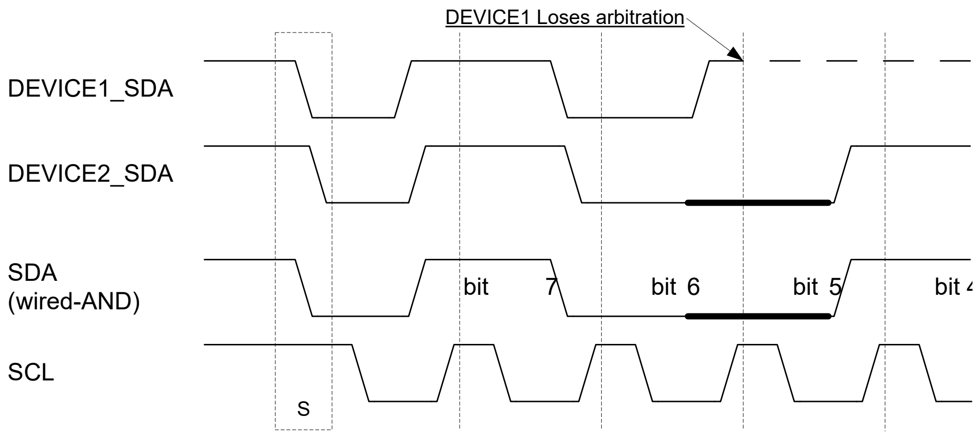
- Table 1 of the SMBus 2.0 specifications gives a maximum clock low timeout ($T_{timeout}$) of 25-35 ms, This can be implemented by connecting the SCL pin to the TCB peripheral using the Event System. Configure the TCB in Time-Out Check mode with a desired timeout value.
- Layer 3 (network layer) features such as packet error checking (PEC), address resolution protocol (ARP). These can be implemented in software if required.

27.3.3.2 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 27-7. TWI Arbitration



27.3.3.3 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^{2}C 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'.

27.3.3.4 Quick Command Mode

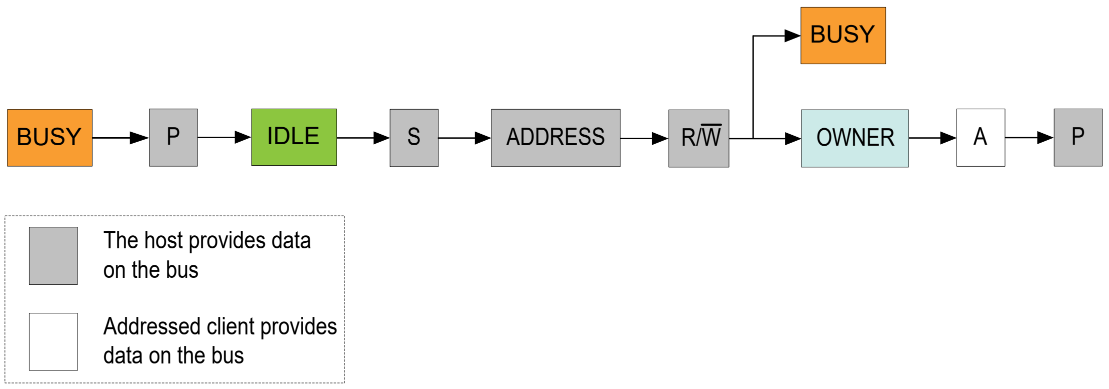
In Quick Command mode, the R/$\overline{W}$ bit from the address packet denotes the command. This mode is enabled by writing '1' to the Quick Command Enable (QCEN) bit in the Host Control A (TWIn.MCTRLA) register. There are no data sent or received.

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 27-8. Quick Command Frame Format


BUSY
P
IDLE
S
ADDRESS
R/$\overline{W}$
BUSY
OWNER
A
P
The host provides data on the bus
Addressed client provides data on the bus

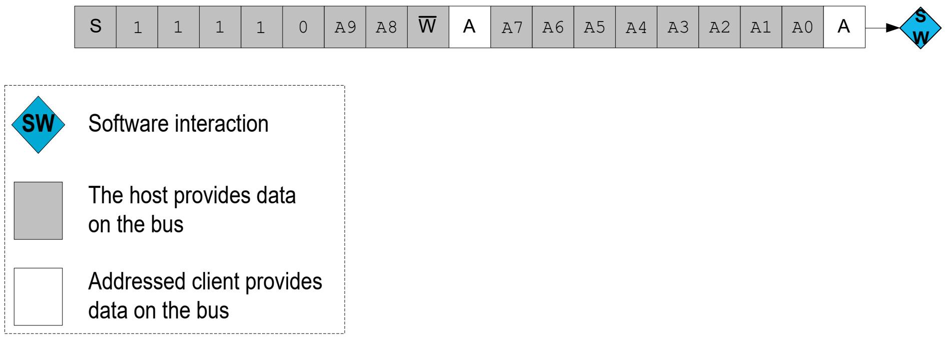
27.3.3.5 10-Bit Address

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 27-9. 10-Bit Address Transmission


S
1
1
1
1
0
A9
A8
$\overline{W}$
A
A7
A6
A5
A4
A3
A2
A1
A0
A
SW
SW
Software interaction
The host provides data on the bus
Addressed client provides data on the bus

27.3.4 Interrupts

Table 27-1. 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.

27.3.5 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.

27.3.6 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.

27.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0				SDASETUP	SDAHOLD[1:0]		FMPEN	
0x01	Reserved									
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

27.5 Register Description

27.5.1 Control A

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

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

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 .

Value	Name	Description
0	OFF	Operating in Standard mode or Fast mode
1	ON	Operating in Fast mode Plus

27.5.2 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

27.5.3 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^{2}C
0x1	50US	50 μs - SMBus (assume the baud rate is set to 100 kHz)
0x2	100US	100 μs (assume the baud rate is set to 100 kHz)
0x3	200US	200 μs (assume the baud rate is set to 100 kHz)

Bit 1 – SMEN Smart Mode Enable

Writing a '1' to this bit enables the 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.

27.5.4 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 27-2. 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.

27.5.5 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

27.5.6 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.

27.5.7 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.

27.5.8 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.

27.5.9 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.

27.5.10 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 27-3. Command Settings

Value	Name	DIR	Description	
0x0	NOACT	X	No action	
0x1	—	X	Reserved	
0x2	COMPTRANS	$\overline{\text{W}}$	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	$\overline{\text{W}}$	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.

27.5.11 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 host must be enabled, and the main clock frequency must be at least four times the SCL frequency. . 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

27.5.12 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^{2}C protocol. This feature is enabled when this bit is set to ‘1’.

27.5.13 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.

27.5.14 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.

28. USB - Universal Serial Bus Device Controller

28.1 Features

- USB 2.0 Full-Speed (12 Mbps) Device-Compliant Interface
- Integrated Internal USB Transceiver. No External Components are Needed.
- 16 Endpoint Addresses with Full Endpoint Flexibility for Up to 32 Endpoints
  - One input endpoint per endpoint address
  - One output endpoint per endpoint address
- Configurable Endpoint Transfer Types
  - Control transfers
  - Interrupt transfers
  - Bulk transfers
  - Isochronous transfers
- Configurable Data Payload Size per Endpoint, up to 1023 Bytes
- Endpoint Configuration Tables and Data Buffers are Located in the Device's Internal RAM
  - Configurable location for endpoint configuration data
  - Configurable location for each endpoint's data buffer
- Built-In Direct Memory Access (DMA) to the Device's Internal RAM for:
  - Access to endpoint configuration tables
  - Read and write endpoint data during the transfer
- Multipacket Transfer for Reduced Interrupt Load and Software Intervention
  - Data payload exceeding maximum packet size in one continuous transfer
  - No interrupts or software interaction on the packet transaction level
- Transaction Complete FIFO for Easy Work Flow when Using Multiple Endpoints
  - Tracks all completed transactions in first in, first out work-queue
- Remote Wake-Up of Host
- Connection to Event System
- On-Chip Debug Capability During USB Transactions

28.2 Overview

The USB peripheral is a USB 2.0 full-speed (12 Mbps) device-compliant interface.

16 endpoint addresses are supported, each with one input and one output endpoint, giving totally 32 configurable endpoints, including one control endpoint. Each endpoint address is supported by the four transfer types - Control, interrupt, bulk, or isochronous. The data payload size is also selectable. It supports data payloads up to 1023 bytes.

No dedicated memory is allocated for or included in the USB peripheral. The device's internal SRAM keeps the configuration and the data buffer for each endpoint address. The memory locations used for endpoint configurations and data buffers are fully configurable. The amount of allocated memory is dynamic according to the configured number of endpoint addresses and the configuration of the endpoints. The USB peripheral has built-in direct memory access (DMA) and will read/write data from/to theSRAM during a USB transaction.

The multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint transferred as multiple packets without software intervention, reducing the CPU intervention and the interrupts needed for USB transfers.

For low-power operation, the microcontroller can enter any sleep mode except power down when the USB bus is idle and a suspend condition is given. Upon a bus resume, the USB peripheral can wake up the microcontroller.

The figures below provide overviews of how the USB peripheral performs USB OUT and IN transfers.

Figure 28-1. USB OUT Transfer: Data Packet From Host To USB Device

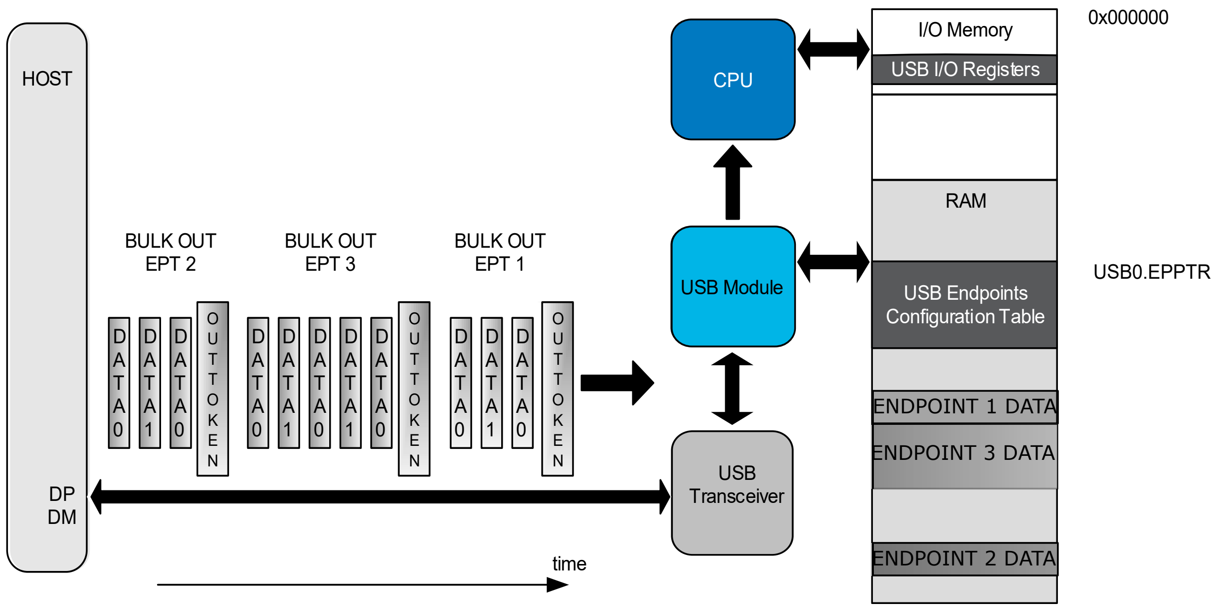
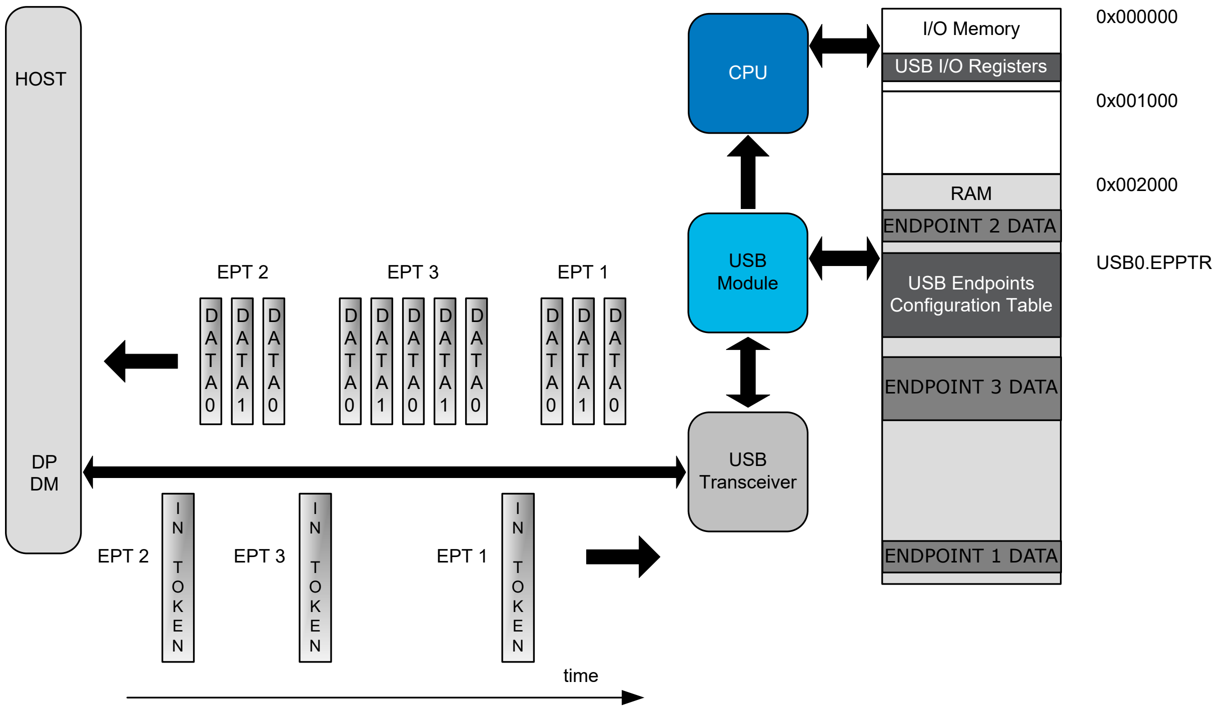
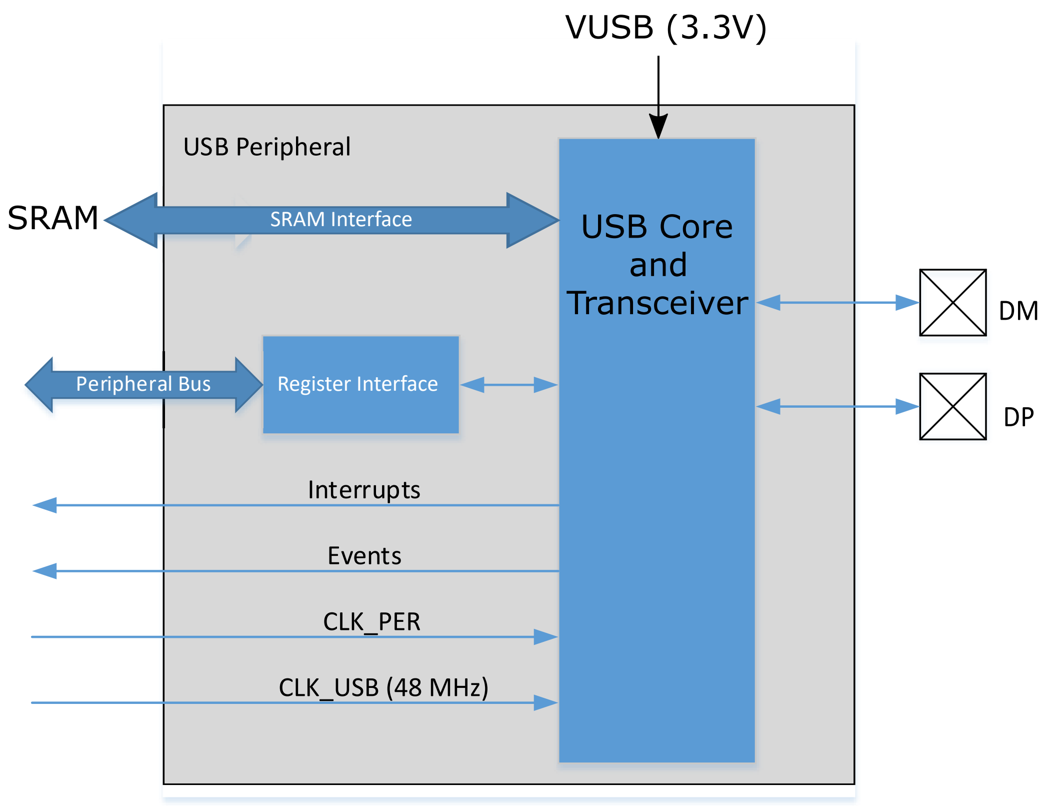


Figure 28-2. USB IN Transfer: Data Packet From USB Device To Host After Request From Host



28.2.1 Block Diagram

Figure 28-3. USB Block Diagram



28.2.2 Signal Description

Signal	Description	Type
DM	Data minus (D-)	Digital
DP	Data plus (D+)	Digital

28.3 Functional Description

28.3.1 Initialization

The USB peripheral requires a correct clock setup and available VUSB power to operate properly.

Refer to the *Electrical Characteristics* section for the operating voltage of the device.

28.3.1.1 Clock Generation

The USB peripheral requires two clock signals; a peripheral clock (CLK_PER) and a USB clock (CLK_USB). CLK_PER is used for the register and SRAM interfaces and must be at least 12 MHz. CLK_USB must be a nominal 48 MHz. This clock is run-time calibrated using the 1 ms interval between Start-of-Frame (SOF) packets as a reference. See the *CLKCTRL - Clock Controller* section for information on configuring these clocks.

28.3.1.2 VUSB Power

For the USB peripheral’s internal transceiver to operate correctly and meet USB specifications, the VUSB pin of the device requires a nominal 3.3V power supply, which can be provided by an external source or generated by an internal 5V (VDD) to 3.3V (VUSB) voltage regulator. See the *USB*

Power Supply Configurations section for more information. For detailed electrical requirements on the VUSB power supply and characteristics of the internal regulator, refer to the *Electrical Characteristics* section. The SYSCFG peripheral controls the internal USB voltage regulator and checks the USB regulator status (ready or not ready). Refer to the *SYSCFG - System Configuration* section for additional information on these registers.

28.3.2 Operation

This section gives an overview of the USB peripheral operation during regular transactions. For general details on USB and the USB protocol, refer to http://www.usb.org and the USB specification documents.

28.3.2.1 SOF - Start-of-Frame

When a Start-of-Frame (SOF) token is detected, the Start-of-Frame Interrupt Flag (SOFIF) in the Interrupt Flags A (USB.INTFLAGSA) register is set.

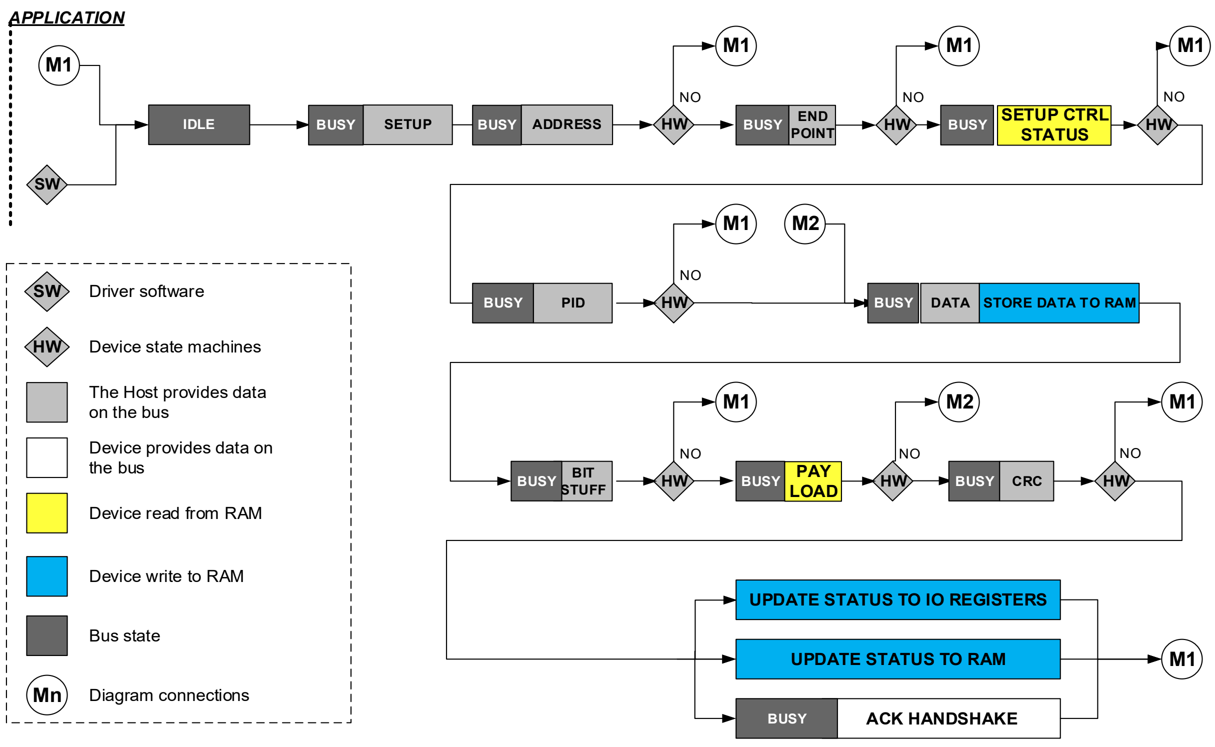
If the Store Frame Number Enable (STRFNUM) bit in the Control A (USB.CTRLA) register is '1', the frame number from the token is stored in the Frame Number (USB.FRAMENUM) register in the endpoint configuration table. If a Cyclic Redundancy Check (CRC) or bit-stuff error occurs, the Frame Error (FRAMEERR) bit in the FRAMENUM register is set.

28.3.2.2 SETUP

When a SETUP token is detected and if the device address n of the token packet does not match the endpoint, the packet is discarded, and the USB peripheral returns to idle and waits for the next token packet.

When the address matches, the USB peripheral then fetches the Endpoint Status (EP[n].[dir].STATUS) register and Endpoint Control (EP[n].[dir].CTRL) register from the addressed output endpoint in the endpoint configuration table. If the endpoint type is not a Control endpoint, the USB peripheral returns to idle and waits for the next token packet.

Figure 28-4. SETUP Transaction



If the endpoint type matches, the USB peripheral then fetches the Endpoint Data Pointer (EP[n].[dir].DATAPTR) register and waits for a DATA0 packet. If a PID error or any other PID than DATA0 is detected, the USB peripheral returns to idle and waits for the next token packet.

When the data PID matches, the incoming data are written to the data buffer pointed to by DATAPTR. If detecting a bit-stuff error in the incoming data, the USB peripheral returns to idle and waits for the next token packet. If the number of received data bytes exceeds the endpoint's maximum data payload size, as specified by the Data Size (BUFSIZE) bit field in CTRL, the remainder of the received data bytes are discarded. The packet will still be checked for bit-stuff and CRC errors. Eight bytes are written to RAM independent of the maximum payload specified by BUFSIZE, as SETUP transactions always have eight data bytes. If there was a bit-stuff or CRC error in the packet, the USB peripheral returns to idle and waits for the next token packet.

If data are successfully received, an ACK handshake is returned to the host, and the number of received data bytes, excluding the CRC, is written to the Endpoint Byte Counter (EP[n].[dir].CNT) register.

Finally, the Endpoint SETUP Complete Flag (EPSETUP), the Data Buffer Not Acknowledge Flag (BUSNAK), and the Data Toggle Flag (TOGGLE) are set when the remaining flags in STATUS are cleared for both the addressed input and output endpoints. If the Automatic Global NAK (GNAUTO) bit in the Control B (USB.CTRLB) register is '1', the Global NAK (GNAK) bit in CTRLB is set. The BUSNAK bit, and possibly the GNAK bit, is set to ensure that the USB peripheral will NAK all future IN and OUT transactions until the FW is ready to process them. The ACK response for setup tokens is not affected by GNAK or BUSNAK. The Setup Transaction Complete Interrupt Flag (SETUP) in the Interrupt Flags B (USB.INTFLAGSB) register is set.

Control transfers use the endpoint buffers differently from IN and OUT endpoints. The data packet in the Setup stage is stored in the endpoint's OUT data buffer, while data packets in the Data stage (i.e., IN or OUT) are stored in the IN data buffer. In other words, EP[n].OUT.DATAPTR points to the received Setup stage packet and EP[n].IN.DATAPTR points to data to be transmitted or received in the Data stage.

28.3.2.3 GNAK - Global Not Acknowledge

Writing a '1' to the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register will return a NAK response on all transactions for non-isochronous endpoints. Any transaction will be ignored for isochronous endpoints.

The application can write the GNAK bit, or it can be set by hardware when a SETUP packet is received, if the Automatic Global NAK (GNAUTO) bit in USB.CTRLB is '1'.

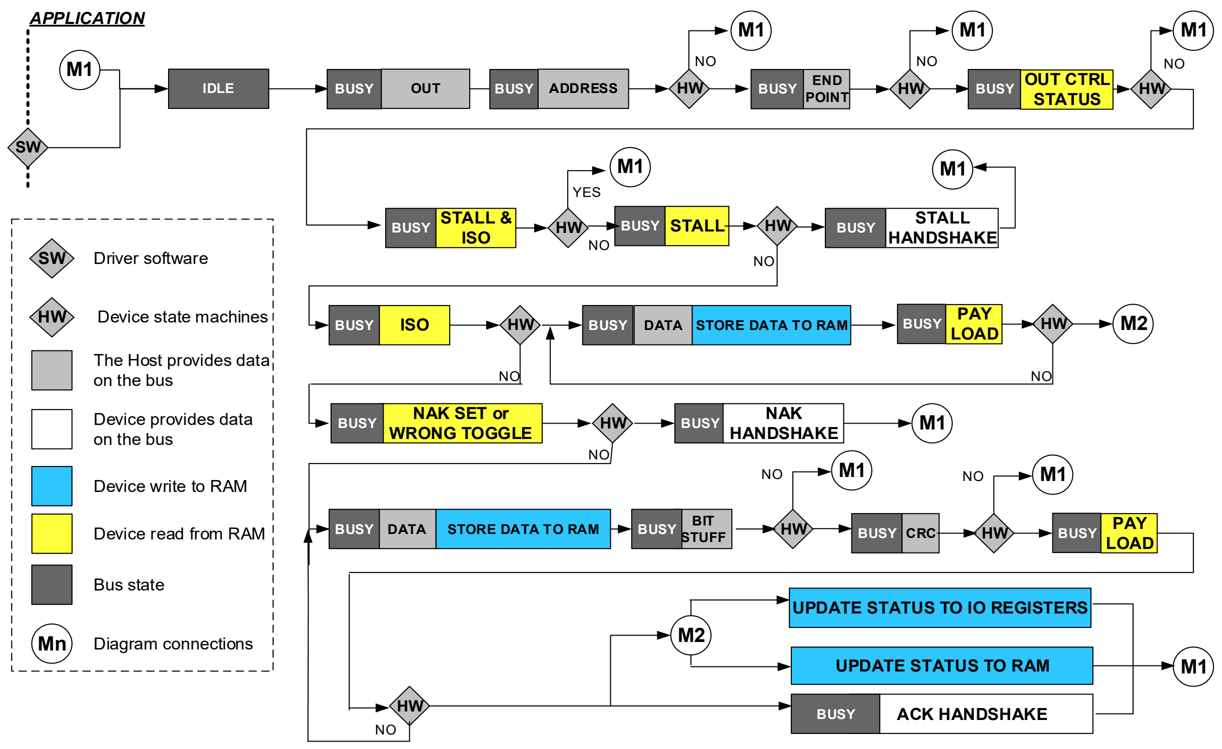
After GNAK is set to '1', the GNAK Operation Done Interrupt Flag (GNDONE) in the Interrupt Flags B (USB.INTFLAGSB) register is set while effectuating the global NAK response.

28.3.2.4 OUT Transaction

When an OUT token is detected and if the device address n of the token packet does not match that of the endpoint, the packet is discarded, and the USB peripheral returns to idle and waits for the next token packet.

When the address matches, the USB peripheral fetches the Endpoint Status (EP[n].OUT.STATUS) register and Endpoint Control (EP[n].OUT.CTRL) register from the addressed output endpoint in the endpoint configuration table. If the endpoint is disabled, the USB peripheral returns to idle and waits for the next token packet.

Figure 28-5. OUT Transaction



If the endpoint type matches, the USB peripheral fetches the Endpoint Data Pointer (EP[n].OUT.DATAPTR) register and waits for a DATA0 or DATA1 packet. If a PID error or any other PID than DATA0 or DATA1 is detected, the USB peripheral returns to idle and waits for the next token packet.

The following behavior depends on the endpoint type:

- Control Endpoint
- Isochronous Endpoint
- Bulk or Interrupt Endpoint

28.3.2.4.1 Control Endpoint

- If the Respond with STALL (DOSTALL) bit in the Endpoint CTRL is set, the incoming data are discarded. A STALL handshake is returned to the host. The Endpoint STALLED Flags (STALLED) in the Endpoint STATUS and in the Interrupt Flags A (INTFLAGSA) register are set.
- If the Endpoint SETUP Complete Flag (EPSETUP) in the Endpoint STATUS is set, the incoming data are discarded. The Underflow/Overflow Flag (UNFOVF) in the Endpoint STATUS and the Overflow Interrupt Flag (OVF) in the INTFLAGSA are set. A NAK handshake is returned to the host.
- The PID is checked against the Data Toggle (TOGGLE) bit in the Endpoint STATUS. Unless they match, the incoming data are discarded, and an ACK handshake is returned to the host. If the Data Buffer Not Acknowledge Flag (BUSNAK) in the Endpoint STATUS or the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register is set, the incoming data are discarded. UNFOVF and OVF are set. A NAK handshake is returned to the host.

The incoming data are written to the data buffer in RAM pointed to by the Endpoint Data Pointer (EP[n].OUT.DATAPTR) register. If detecting a bit-stuff error in the incoming data, the USB peripheral returns to idle and waits for the next token packet. The remainders of the received data bytes are discarded if the number of received data bytes exceeds the maximum data payload specified in the Data Size (BUFSIZE) bit field in the Endpoint CTRL. The packet is checked for bit-stuff and CRC errors. If it contains a bit-stuff or CRC error, the USB peripheral returns to idle and waits for the next token packet. The first CRC byte may be written to SRAM if receiving fewer bytes than the maximum payload specified by BUFSIZE. It will be written if the last proper data bit is written to an even address and CNT<MCNT when multipacket is enabled or CNT<EPSIZE when multipacket is not enabled.

If successfully receiving data, an ACK handshake is returned to the host, and the number of received data bytes, excluding CRC, is written to CNT.

Finally, BUSNACK and Transaction Complete Flag (TRNCOMPL) are set, and TOGGLE is toggled. The endpoint address is written to the FIFO if this is enabled.

28.3.2.4.2 Isochronous Endpoint

Data from DATA0 and DATA1 packets are accepted. If the Data Buffer Not Acknowledge Flag (BUSNAK) in the Endpoint STATUS or the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register is set, the incoming data is discarded. The Underflow/Overflow Flag (UNFOVF) in the Endpoint STATUS and Overflow Interrupt Flag (OVF) in INTFLAGSA are set.

The incoming data are written to the data buffer in RAM pointed to by the Endpoint Data Pointer (EP[n].OUT.DATAPTR) register. If detecting a bit-stuff error in the incoming data, the USB peripheral returns to idle and waits for the next token packet. The remainders of the received data bytes are discarded if the number of received data bytes exceeds the maximum data payload specified in the Data Size (BUFSIZE) bit field in Endpoint CTRL. The packet is checked for bit-stuff and CRC errors. If it contains a bit-stuff or CRC error, the USB peripheral returns to idle and waits for the next token packet. The first CRC byte may be written to RAM if receiving fewer bytes than the maximum payload specified by SIZE.

If data was successfully received, the number of received data bytes, excluding CRC, is written to CNT.

If there is a bit-stuff or CRC error, CRC in STATUS will indicate that the received data contain errors.

The Transaction Complete Interrupt Flag (TRNCOMPL) in INTFLAGSB is set, and the endpoint address is written to the FIFO if this is enabled.

28.3.2.4.3 BULK or INTERRUPT Endpoint

- If the Respond with STALL (DOSTALL) bit in the Endpoint CTRL is set, the incoming data are discarded. A STALL handshake is returned to the host. The Endpoint STALLED Flags (STALLED) in the Endpoint STATUS and the Interrupt Flags A (INTFLAGSA) registers are set.
- The PID is checked against the Data Toggle (TOGGLE) bit in the STATUS endpoint. The incoming data are discarded if they don’t match, and an ACK handshake is returned to the host. If the Data Buffer Not Acknowledge Flag (BUSNAK) in endpoint STATUS or the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register is set, the incoming data are discarded. The Underflow/Overflow Flag (UNFOVF) in endpoint STATUS and Overflow Interrupt Flag (OVF) in INTFLAGSA are set.

The incoming data are written to the data buffer in RAM pointed to by the Endpoint Data Pointer (EP[n].OUT.DATAPTR) register. If detecting a bit-stuff error in the incoming data, the USB peripheral returns to idle and waits for the next token packet. The remainders of the received data bytes are discarded if the number of received data bytes exceeds the maximum data payload specified in the Data Size (BUFSIZE) bit field in endpoint CTRL. The packet is checked for bit-stuff and CRC errors. If it contains a bit-stuff or CRC error, the USB peripheral returns to idle and waits for the next token packet. The first CRC byte may be written to RAM if receiving fewer bytes than the maximum payload specified in the Data Size (BUFSIZE) bit field.

If data was successfully received, an ACK handshake is returned to the host, and the number of received data bytes, excluding CRC, is written to CNT.

Finally, BUSNACK and Transaction Complete Flag (TRNCOMPL) are set, and TOGGLE is toggled. The Transaction Complete Interrupt Flag (TRNCOMPL) in INTFLAGSB is set, and the endpoint address is written to the FIFO if this if enabled.

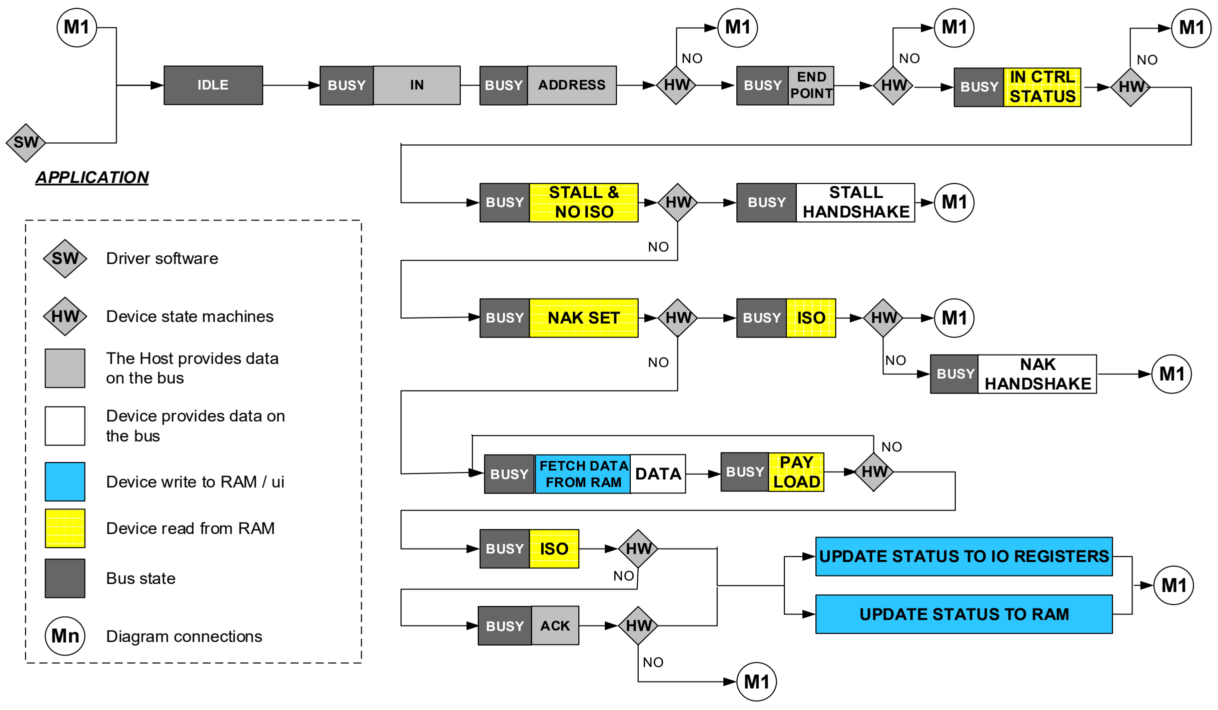
The USB peripheral does not distinguish between INTERRUPT and BULK endpoints and treats them identically.

28.3.2.5 IN Transaction

When an IN token is detected and if the device address n of the token packet does not match that of the endpoint, the packet is discarded and the USB peripheral returns to idle, waiting for the next token packet.

When the address matches, the USB peripheral then fetches the Endpoint Status (EP[n].IN.STATUS) register and Endpoint Control (EP[n].IN.CTRL) register from the addressed output endpoint in the endpoint configuration table. If the endpoint is disabled, the USB peripheral returns to idle and waits for the next token packet.

Figure 28-6. IN Transaction



The following behavior depends on the endpoint type:

- Control Endpoint
- Isochronous Endpoint
- Bulk or Interrupt Endpoint

28.3.2.5.1 Control Endpoint

- If the Respond with STALL (DOSTALL) bit in the Endpoint CTRL is set, the incoming data are discarded. A STALL handshake is returned to the host. The Endpoint STALLED Flags (STALLED) in Endpoint STATUS and in the Interrupt Flags A (INTFLAGSA) register are set.
- If the Endpoint SETUP Complete Flag (EPSETUP) in the Endpoint STATUS is set, the incoming data are discarded. The Underflow/Overflow Flag (UNFOVF) in the Endpoint STATUS and Underflow Interrupt Flag (UNF) in INTFLAGSA are set. A NAK handshake is returned to the host.
- If the Data Buffer Not Acknowledge Flag (BUSNAK) in the Endpoint STATUS or the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register is set, the incoming data are discarded. UNFOVF and UNF are set. A NAK handshake is returned to the host.

The data in the data buffer in RAM pointed to by the Endpoint Data Pointer (EP[n].IN.DATAPTR) register is sent to the host in a DATA0 or DATA1 packet according to the Data Toggle (TOGGLE) bit in endpoint STATUS. When the number of data bytes specified in the endpoint's CNT is sent, the CRC is appended and sent to the host.

The USB peripheral is waiting for an ACK handshake from the host. If an ACK handshake is not received within 16-bit times, the USB peripheral returns to idle and waits for the next token packet. If successfully receiving an ACK handshake, then BUSNACK and Transaction Complete Flag (TRNCOMPL) are set, and TOGGLE is toggled. The endpoint address is written to the FIFO if this is enabled.

28.3.2.5.2 Isochronous Endpoint

- The Underflow/Overflow Flag (UNFOVF) in Endpoint STATUS and Underflow Interrupt Flag (UNF) in INTFLAGSA are set if the Data Buffer Not Acknowledge Flag (BUSNAK) in Endpoint STATUS or the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register is set.

The data in the data buffer in SRAM pointed to by the Endpoint Data Pointer (EP[n].IN.DATAPTR) register are sent to the host in a DATA0 packet. When the number of data bytes specified in the endpoint's CNT is sent, the CRC is appended and sent to the host.

BUSNACK and Transaction Complete Flag (TRNCOMPL) in the Endpoint STATUS are set. TRNCOMPL in INTFLAGSB is set, and the endpoint address is written to the FIFO if this is enabled.

28.3.2.5.3 BULK or INTERRUPT Endpoint

- If the Respond with STALL (DOSTALL) bit in the Endpoint (USB.EP[n].OUT.CTRL) register is set, the incoming data is discarded. A STALL handshake is returned to the host. The Endpoint STALLED (STALLED) flags in the Endpoint Status (EP[n].OUT.STATUS) register and in the Interrupt Flags A (INTFLAGSA) register are set.
- If the Data Buffer Not Acknowledge (BUSNAK) flag in Endpoint STATUS or the Global NAK (GNAK) bit in the Control B (USB.CTRLB) register is set, the incoming data is discarded. The Underflow/Overflow (UNFOVF) flag in Endpoint STATUS and Underflow Interrupt (UNF) flag in the Interrupt Flags A (INTFLAGSA) register are set. A NAK handshake is returned to the host.

The data in the data buffer in RAM pointed to by the Endpoint Data Pointer (EP[n].IN.DATAPTR) register is sent to the host in a DATA0 or DATA1 packet according to the Data Toggle (TOGGLE) bit in Endpoint STATUS. When the number of data bytes specified in the endpoint's CNT is sent, the CRC is appended and sent to the host.

The USB peripheral is waiting for an ACK handshake from the host. If an ACK handshake is not received within 16-bit times, the USB peripheral returns to idle and waits for the next token packet. If successfully receiving an ACK handshake, then BUSNACK and Transaction Complete (TRNCOMPL) flag are set, and TOGGLE is toggled. The endpoint address is written to the FIFO if this is enabled.

28.3.2.6 Multipacket Transfer

A Multipacket transfer enables a data payload exceeding the endpoint maximum data payload size to be transferred as multiple packets without software intervention, reducing interrupts and software intervention to higher level USB transfer, not each packet. The Multipacket transfer is identical to the IN and OUT transactions described above unless noted in this section. The Multipacket does not require any special consideration on the host side. The Multipacket transfer is not recommended for isochronous endpoints except when used to write-protect buffers smaller than the endpoint size.

The Multipacket is enabled by writing a '`1`' to the Multipacket Transfer Enable (MULTIPKT) bit in the Endpoint Control (EP[n].[dir].CTRL) register.

The Multipacket Byte Count (EP[n].[dir].MCNT) and Endpoint Data Pointer (EP[n].[dir].DATAPTR) registers are used to setting the size and address of the RAM buffer to be processed by the USB peripheral for a specific endpoint. The USB peripheral will split the required USB data transfers buffer without software intervention.

Figure 28-7. Multipacket Overview

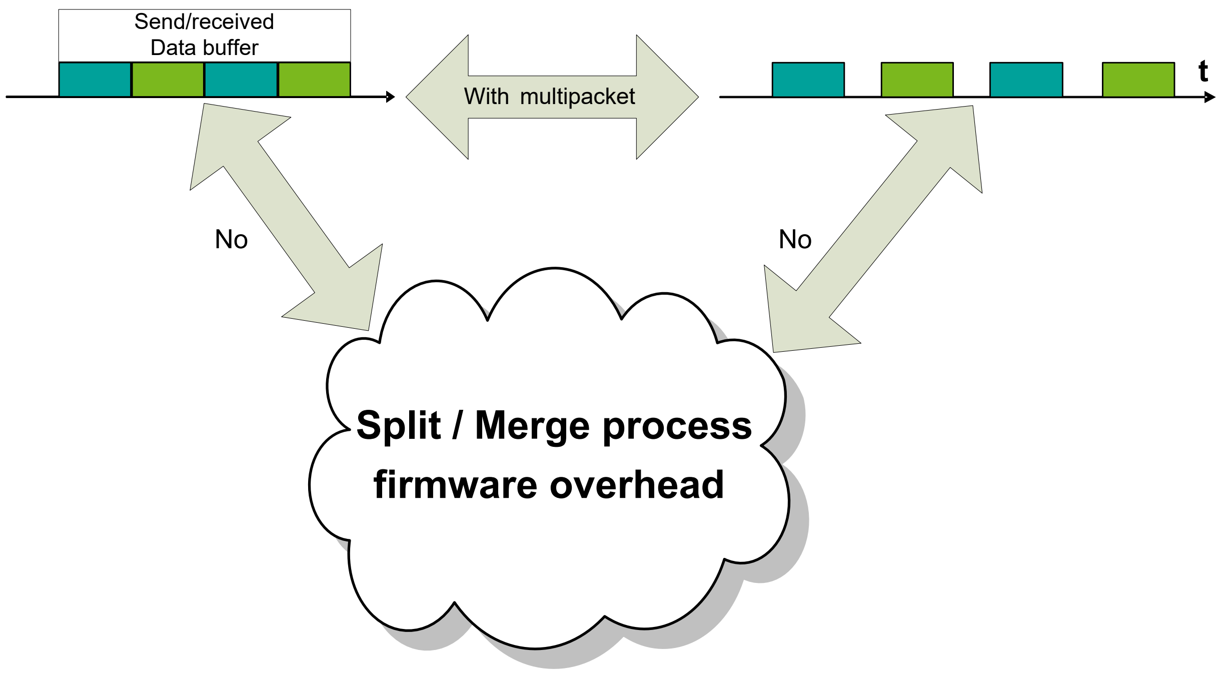
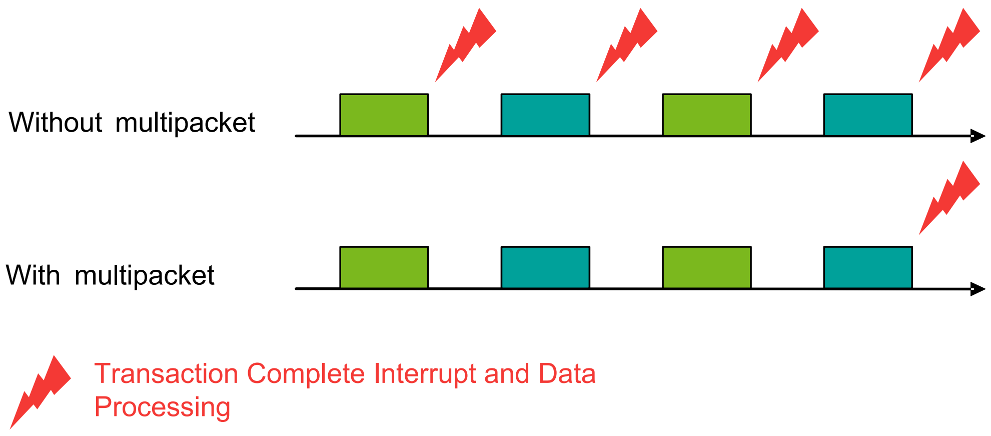


Figure 28-8. Multipacket Reduces CPU Load



28.3.2.6.1 Multipacket For Input Endpoints

The total number of data bytes to be sent is written to CNT, as for an ordinary operation. The Multipacket Byte Count (MCNT) register is used to store the number of sent bytes, and must be written to zero when setting up a new transfer. When an IN token is received, Endpoint CNT and Endpoint MCNT are fetched. If CNT minus MCNT is less than the Endpoint SIZE, Endpoint CNT minus Endpoint MCNT bytes are transmitted. Otherwise SIZE number of bytes are transmitted. The last packet sent will be zero-length if the AZLP bit is set and the Endpoint CNT is a multiple of SIZE.

If sending a maximum payload size packet (i.e., not the last transaction), MCNT will be incremented by the SIZE. STATUS.TOGGLE will be toggled if the endpoint is not isochronous after the transaction completion. If sending a short packet (i.e., the last transaction), MCNT is incremented by the data payload. STATUS.TOGGLE will be toggled if the endpoint is not isochronous, and STATUS.BUSNACK, STATUS.TRNCOMPL and INTFLAGSB.TRNCOMPL will be set.

28.3.2.6.2 Multipacket For Output Endpoints

The number of data bytes received is stored in Endpoint CNT, as for an ordinary operation. Since Endpoint CNT is updated after each transaction, it must be set to zero when setting up a new transfer. Write all the received bytes to MCNT.

TOGGLE management for non-isochronous packets and NACK management are as for ordinary operation.

If a maximum payload size packet is received, CNT will increment by SIZE after the completed transaction, and STATUS.TOGGLE will toggle if the endpoint is not isochronous. If the updated Endpoint CNT is equal to MCNT and AZLP is not set, STATUS.BUSNACK, STATUS.TRNCOMPL and INTFLAGSB.TRNCOMPL will be set.

If MULTIPKT and AZLP are set, and CNT is equal to MCNT and is a multiple of endpoint size, then STATUS.BUSNACK, STATUS.TRNCOMPL and INTFLASB.TRNCOMPL will not be set before an additional ZLP is received.

If a short or oversized packet is received, Endpoint CNT will increment by the data payload after completing the transaction. STATUS.TOGGLE will be toggled if the endpoint is not isochronous and STATUS.BUSNACK, STATUS.TRNCOMPL and INTFLAGSB.TRNCOMPL will be set.

28.3.2.6.3 Write Protect

Use the MULTIPKT bit combined with MCNT to set the maximum number of bytes the USB peripheral will write to RAM for OUT transactions. MCNT can be any number, including zero. The application code has to allocate a buffer at least as big as what MCNT is set to for the endpoint to ensure that no RAM corruption is possible. If MULIPKT and MCNT are unused, the allocated buffer has to be as large as set by BUFSIZE.

The write protect can be used to save RAM used by the application in cases where the used protocol is expected to not sending a multiple of the endpoint size.

28.3.2.6.4 Auto Zero Length Packet

Some device classes require a generation of a zero-length packet to signal the end of a transfer. If the transferred data are an exact multiple of the packet size for the specific endpoint, a ZLP is appended to signal the end of the transfer. Enable the auto zero length packet (AZLP) function to perform this generation automatically, thus removing the need for application software or CPU intervention to perform this task.

For IN Endpoints, the device will send a ZLP to the host after CNT bytes have been sent if CNT is a multiple of the maximum payload size. For OUT Endpoints, the device will respond with ACK to a ZLP if MCNT bytes are received while responding with NAK to any other data packet.

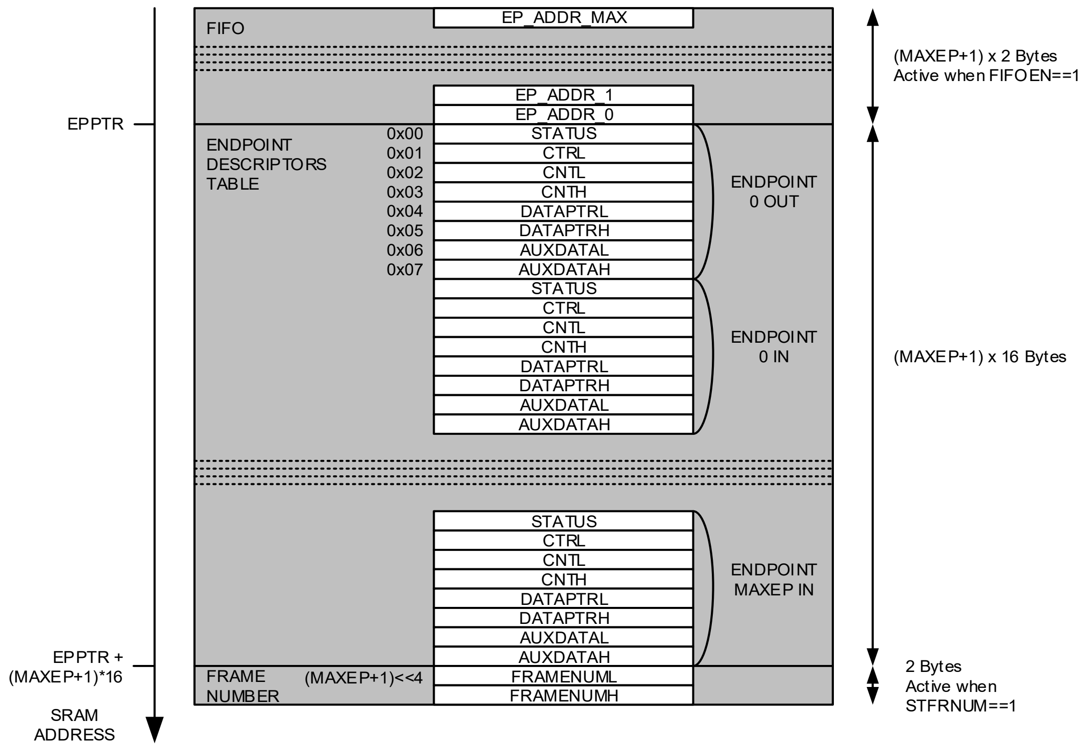
28.3.3 SRAM Memory Mapping

The USB peripheral uses the device’s internal SRAM to store:

- Transaction Complete FIFO
- Endpoint Configuration Table
- USB Frame Number

The Endpoint Configuration Table Pointer (USB.EPPTR) register is used to set the SRAM address for the endpoint configuration table. The USB frame number (FRAMENUM) register and the transaction complete (FIFO) table locations are derived from this. The location is selectable inside the device’s internal SRAM. The figure below gives the relative memory location of each area.

Figure 28-9. USB Memory Mapping (Descriptor table, FIFO, frame number, endpoint buffers) In RAM



28.3.3.1 Data Hazards On Endpoint Status

Both the USB hardware and the application will modify the Endpoint STATUS registers. To perform a safe write access to the Endpoint STATUS registers, the RMW interface provided by the STATUS{In/Out}{Clear/Set}n allows setting or clearing of bits in Endpoint n IN/OUT. By writing a bit mask to these registers, the USB will set or clear the appropriate bits in the RAM location at a safe time. When an RMW operation triggered by a write to STATUS{In/Out}{Clear/Set}n is ongoing, the RMWBUSY bit in INTFLAGSB is set.

The result of a write to STATUS{In/Out}{Clear/Set} while RMWBUSY is set is undefined.

28.3.4 Events

The USB peripheral can generate several events. These are available to the event system, allowing latency-free signaling to other peripherals or performance analysis of USB operation.

Figure 28-10. Interrupts and Events Scheme Summary

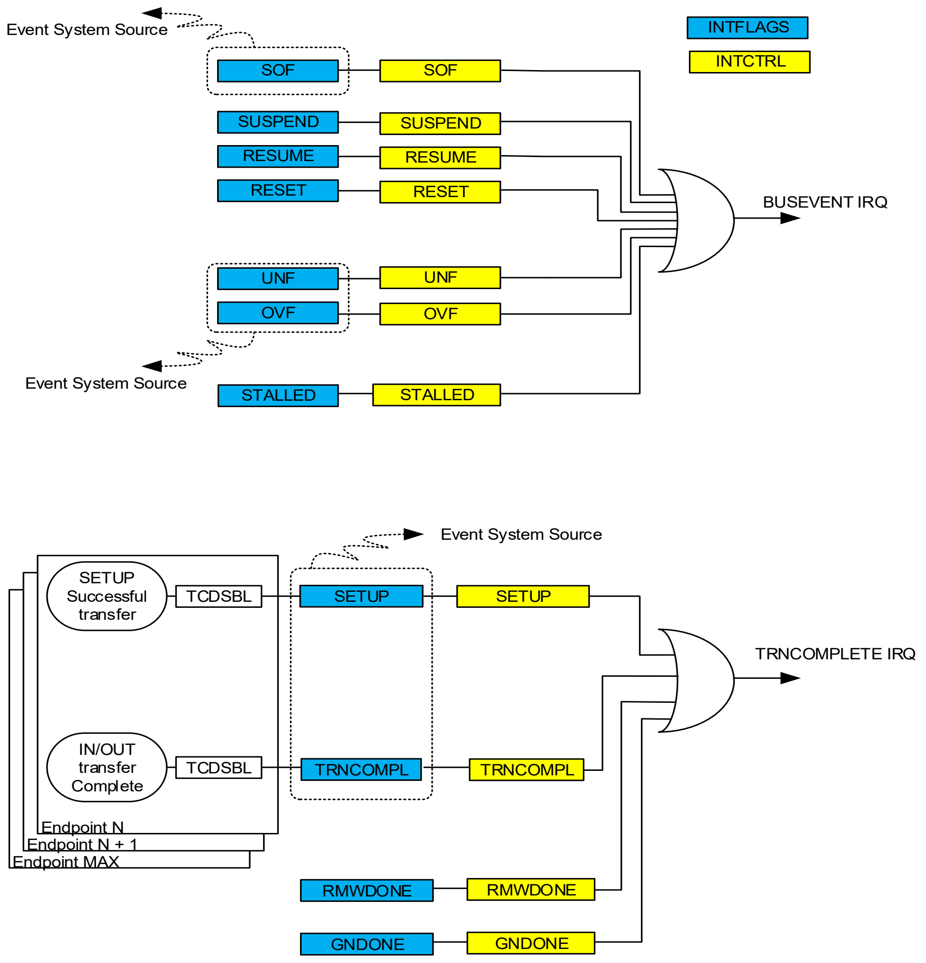


Table 28-1. Event Generators in USB

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
USB	SETUPIF				
	SOFIF				
	CRCIF				
	UNFIF/OVFIF				

Refer to the *Event System (EVSYS)* section for more details about event types and Event System configuration.

28.3.5 Interrupts

The peripheral has ten interrupt sources. These are split between two interrupt vectors, the transaction complete (TRNCOMPL) interrupt and the bus event (BUSEVENT) interrupt. An interrupt group is enabled by setting its interrupt level (INTLVL), while different interrupt sources are enabled individually or in groups.

The table below summarizes the interrupts and event sources for the USB peripheral and shows how they are enabled.

Table 28-2. Available Interrupt Vectors and Sources

Name	Vector Description	Interrupt Flag	Conditions
TRNCOMPL	Transaction Complete interrupt	SETUPIF	
		TRNIF	
BUSEVENT	Bus Event interrupt	SOFIF	
		SUSPENDIF	
		RESUMEIF	

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.

Transaction Complete Interrupt

The transaction complete interrupt is generated per endpoint. When an interrupt occurs, the associated endpoint number is registered and optionally added to the FIFO. The following two interrupt sources use the interrupt vector:

Table 28-3. Transaction complete interrupt sources.

Interrupt Source	Description
Transfer complete (TRNIF)	An IN or OUT transaction is completed
Setup complete (SETUPIF)	A SETUP transaction is completed

Bus Event Interrupt

The bus event (BUSEVENT) interrupt is used for all interrupts that signal various types of USB line events or error conditions. These interrupts are related to the USB lines and are generated for the USB peripheral and per endpoint. The following eight interrupts use the interrupt vector:

Table 28-4. Bus Event Interrupt Sources

Interrupt Source	Description
Start of frame (SOFIF)	A SOF token has been received
Suspend (SUSPENDIF)	The bus has been idle for 3 ms
Resume (RESUMEIF)	A non-idle state is detected when the bus is suspended. The interrupt is asynchronous and can wake the device from all sleep modes.
Reset (RSTIF)	A reset condition has been detected on the bus
Isochronous CRC error (CRCIF)	A CRC error or a bit-stuff error has been detected in an incoming packet to an isochronous endpoint
Underflow (UNFIF)	An endpoint is unable to return data to the host
Overflow (OVFIF)	An endpoint is unable to accept data from the host
STALL (STALLIF)	A STALL handshake has been returned to the host

28.3.6 Sleep Mode Operation

The USB peripheral is active in Standby sleep mode if it is enabled. In this case, the USB suspend/resume mechanism should be implemented, respecting the power consumption limits defined in the USB standard.

28.3.7 Debug Operation

The USB peripheral continues to operate normally as far as possible during the OCD break. However, higher-level protocols may time out, as transactions may not be processed on time.

28.4 Register Summary - USBn

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ENABLE		FIFOEN	STFRNUM	MAXEP[3:0]			
0x01	CTRLB	7:0					URESUME	GNAUTO	GNAK	ATTACH
0x02	BUSSTATE	7:0				WTRSM	URESUME	DRESUME	SUSPENDED	BUSRST
0x03	ADDR	7:0		ADDR[6:0]						
0x04	FIFOWP	7:0				FIFOWP[4:0]				
0x05	FIFORP	7:0				FIFORP[4:0]				
0x06	EPPTR	7:0	EPPTR[7:0]							
		15:8	EPPTR[15:8]							
0x07	INTCTRLA	7:0	SOF	SUSPEND	RESUME	RESET	STALLED	UNF	OVF	
0x08	INTCTRLB	7:0			TRNCOMPL				GNDONE	SETUP
0x09	INTFLAGSA	7:0	SOF	SUSPEND	RESUME	RESET	STALLED	UNF	OVF	
0x0A	INTFLAGSB	7:0			TRNCOMPL			RMWBUSY	GNDONE	SETUP
0x0B ... 0x3F	Reserved									
0x40	STATUS[0].OUTCLR	7:0	RMWSTATUS[7:0]							
0x41	STATUS[0].OUTSET	7:0	RMWSTATUS[7:0]							
0x42	STATUS[0].INCLR	7:0	RMWSTATUS[7:0]							
0x43	STATUS[0].INSET	7:0	RMWSTATUS[7:0]							
0x44	STATUS[1].OUTCLR	7:0	RMWSTATUS[7:0]							
0x45	STATUS[1].OUTSET	7:0	RMWSTATUS[7:0]							
0x46	STATUS[1].INCLR	7:0	RMWSTATUS[7:0]							
0x47	STATUS[1].INSET	7:0	RMWSTATUS[7:0]							
0x48	STATUS[2].OUTCLR	7:0	RMWSTATUS[7:0]							
0x49	STATUS[2].OUTSET	7:0	RMWSTATUS[7:0]							
0x4A	STATUS[2].INCLR	7:0	RMWSTATUS[7:0]							
0x4B	STATUS[2].INSET	7:0	RMWSTATUS[7:0]							
0x4C	STATUS[3].OUTCLR	7:0	RMWSTATUS[7:0]							
0x4D	STATUS[3].OUTSET	7:0	RMWSTATUS[7:0]							
0x4E	STATUS[3].INCLR	7:0	RMWSTATUS[7:0]							
0x4F	STATUS[3].INSET	7:0	RMWSTATUS[7:0]							
0x50	STATUS[4].OUTCLR	7:0	RMWSTATUS[7:0]							
0x51	STATUS[4].OUTSET	7:0	RMWSTATUS[7:0]							
0x52	STATUS[4].INCLR	7:0	RMWSTATUS[7:0]							
0x53	STATUS[4].INSET	7:0	RMWSTATUS[7:0]							
0x54	STATUS[5].OUTCLR	7:0	RMWSTATUS[7:0]							
0x55	STATUS[5].OUTSET	7:0	RMWSTATUS[7:0]							
0x56	STATUS[5].INCLR	7:0	RMWSTATUS[7:0]							
0x57	STATUS[5].INSET	7:0	RMWSTATUS[7:0]							
0x58	STATUS[6].OUTCLR	7:0	RMWSTATUS[7:0]							
0x59	STATUS[6].OUTSET	7:0	RMWSTATUS[7:0]							
0x5A	STATUS[6].INCLR	7:0	RMWSTATUS[7:0]							
0x5B	STATUS[6].INSET	7:0	RMWSTATUS[7:0]							
0x5C	STATUS[7].OUTCLR	7:0	RMWSTATUS[7:0]							
0x5D	STATUS[7].OUTSET	7:0	RMWSTATUS[7:0]							
0x5E	STATUS[7].INCLR	7:0	RMWSTATUS[7:0]							
0x5F	STATUS[7].INSET	7:0	RMWSTATUS[7:0]							
0x60	STATUS[8].OUTCLR	7:0	RMWSTATUS[7:0]							
0x61	STATUS[8].OUTSET	7:0	RMWSTATUS[7:0]							
0x62	STATUS[8].INCLR	7:0	RMWSTATUS[7:0]							
0x63	STATUS[8].INSET	7:0	RMWSTATUS[7:0]							
0x64	STATUS[9].OUTCLR	7:0	RMWSTATUS[7:0]							
0x65	STATUS[9].OUTSET	7:0	RMWSTATUS[7:0]							
0x66	STATUS[9].INCLR	7:0	RMWSTATUS[7:0]							
0x67	STATUS[9].INSET	7:0	RMWSTATUS[7:0]							
0x68	STATUS[10].OUTCLR	7:0	RMWSTATUS[7:0]							
0x69	STATUS[10].OUTSET	7:0	RMWSTATUS[7:0]							
0x6A	STATUS[10].INCLR	7:0	RMWSTATUS[7:0]							

...........continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x6B	STATUS[10].INSET	7:0	RMWSTATUS[7:0]							
0x6C	STATUS[11].OUTCLR	7:0	RMWSTATUS[7:0]							
0x6D	STATUS[11].OUTSET	7:0	RMWSTATUS[7:0]							
0x6E	STATUS[11].INCLR	7:0	RMWSTATUS[7:0]							
0x6F	STATUS[11].INSET	7:0	RMWSTATUS[7:0]							
0x70	STATUS[12].OUTCLR	7:0	RMWSTATUS[7:0]							
0x71	STATUS[12].OUTSET	7:0	RMWSTATUS[7:0]							
0x72	STATUS[12].INCLR	7:0	RMWSTATUS[7:0]							
0x73	STATUS[12].INSET	7:0	RMWSTATUS[7:0]							
0x74	STATUS[13].OUTCLR	7:0	RMWSTATUS[7:0]							
0x75	STATUS[13].OUTSET	7:0	RMWSTATUS[7:0]							
0x76	STATUS[13].INCLR	7:0	RMWSTATUS[7:0]							
0x77	STATUS[13].INSET	7:0	RMWSTATUS[7:0]							
0x78	STATUS[14].OUTCLR	7:0	RMWSTATUS[7:0]							
0x79	STATUS[14].OUTSET	7:0	RMWSTATUS[7:0]							
0x7A	STATUS[14].INCLR	7:0	RMWSTATUS[7:0]							
0x7B	STATUS[14].INSET	7:0	RMWSTATUS[7:0]							
0x7C	STATUS[15].OUTCLR	7:0	RMWSTATUS[7:0]							
0x7D	STATUS[15].OUTSET	7:0	RMWSTATUS[7:0]							
0x7E	STATUS[15].INCLR	7:0	RMWSTATUS[7:0]							
0x7F	STATUS[15].INSET	7:0	RMWSTATUS[7:0]							

28.5 Register Description - USBn

28.5.1 Control A

Name: CTRLA
Offset: 0x0
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	ENABLE		FIFOEN	STFRNUM	MAXEP[3:0]			
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit 7 – ENABLE USB Enable

Writing a '1' to this bit enables the USB peripheral.
Writing a '0' to this bit disables the USB peripheral and immediately aborts ongoing transactions.

Value	Description
1	The USB peripheral is enabled
0	The USB peripheral is disabled

Bit 5 – FIFOEN Transaction Complete FIFO Enable

Value	Description
1	The USB Transaction Complete FIFO (FIFO) is enabled. The FIFO stores the offset to the endpoint configuration table address of each endpoint that generates a transaction complete interrupt.
0	The FIFO is disabled and the RAM allocated for it is freed

Bit 4 – STFRNUM Store Frame Number Enable

This bit controls whether storing the last SOF token frame number in the Frame Number (FRAMENUM) register is enabled.

Value	Description
1	Storing the last SOF token frame number in FRAMENUM is enabled
0	Storing the last SOF token frame number in FRAMENUM is disabled

Bits 3:0 – MAXEP[3:0] Maximum Endpoint Address

This bit field selects the number of endpoint addresses used by the USB peripheral. Incoming packets with a higher endpoint number than this address will be discarded. Packets with an endpoint address lower than or equal to this address will cause the USB peripheral to look up the addressed endpoint in the endpoint configuration table. Example: If EP 0, 1 and 2 are used, set this field to 2.

28.5.2 Control B

Name: CTRLB
Offset: 0x1
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
					URESUME	GNAUTO	GNAK	ATTACH
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – URESUME Upstream Resume
This bit initiates an upstream resume on the USB bus.
The application must not initiate an upstream resume unless the USB bus has been in the suspend state for at least 5 ms. See the SUSPENDED bit in the Bus State (USB.BUSSTATE) register.
Writing a '0' to this bit has no effect.
Writing a '1' initiates an upstream resume on the USB bus
This bit is cleared when an upstream resume has been initiated.

Bit 2 – GNAUTO Automatic Global NAK
This bit selects if the peripheral automatically sends an Automatic Global NAK (GNAK) on receiving a SETUP transaction.

Value	Description
0	Automatic GNAK is disabled
1	Automatic GNAK is enabled

Bit 1 – GNAK Global NAK
Writing this bit to '1' causes all endpoints to respond with NAK on any transaction. When the Global NAK has been enabled throughout the USB peripheral, the Global NAK Done (GNDONE) flag in the Interrupt Flags B (INTFLAGSB) register is set.

Value	Description
0	Global NAK is disabled
1	Global NAK is enabled

Bit 0 – ATTACH Attach
This bit controls whether the internal pull-up on the D+ line is enabled.

Value	Description
0	The internal D+ pull-up is disabled and the device is detached from the USB bus
1	The internal D+ pull-up is enabled and the device is attached to the USB bus

28.5.3 Bus State

Name: BUSSTATE
Offset: 0x2
Reset: 0x00

This register reflects the real-time state of the USB bus and may change at any time.

Bit	7	6	5	4	3	2	1	0
				WTRSM	URESUME	DRESUME	SUSPENDED	BUSRST
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit 4 – WTRSM Wait Time Resume

Value	Description
0	The bus has not been IDLE for more than 5 ms (i.e., the T_WTRSM parameter in the USB 2.0 specification)
1	The bus has been IDLE for more than 5 ms, and transmission of upstream resume is allowed

Bit 3 – URESUME Upstream Resume

Value	Description
0	No upstream resume in progress
1	An upstream resume is in progress

Bit 2 – DRESUME Downstream Resume

Value	Description
0	No downstream resume in progress
1	A downstream resume is in progress

Bit 1 – SUSPENDED Bus Suspended

Value	Description
0	The USB bus is not in the suspended state
1	The USB bus is in the suspended state, i.e., the bus has been idle for at least 3 ms

Bit 0 – BUSRST Bus Reset

Value	Description
0	No reset condition has been detected
1	A reset condition has been detected (the bus has been driven to SE0 for at least 2.5 μs)

28.5.4 Device Address

Name: ADDR
Offset: 0x3
Reset: 0x00

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

Bits 6:0 – ADDR[6:0] Device Address
This bit field contains the USB address the device will respond to.

28.5.5 FIFO Write Pointer

Name:	FIFOWP
Offset:	0x4
Reset:	0x00

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

Bits 4:0 – FIFOWP[4:0] FIFO Write Pointer

This bit field contains the transaction complete FIFO write pointer, pointing to the last FIFO entry updated by hardware. This bit field value is updated by hardware when a USB transaction has been completed.

Writing any value to this bit field will reset both the FIFO Write Pointer and the FIFO Read Pointer (FIFOWP and FIFORP) registers.

28.5.6 FIFO Read Pointer

Name:	FIFORP
Offset:	0x5
Reset:	0x00

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

Bits 4:0 – FIFORP[4:0] FIFO Read Pointer

This bit field contains the transaction complete FIFO read pointer, pointing to the next FIFO entry to be handled by the application. This bit field value is updated by hardware after being read by the CPU.

Writing any value to this bit field will reset both the FIFO Write Pointer and the FIFO Read Pointer (FIFOWP and FIFORP) registers.

28.5.7 Endpoint Configuration Table Pointer

Name:	EPPTR
Offset:	0x6
Reset:	0x00

Bit	15	14	13	12	11	10	9	8
	EPPTR[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
	EPPTR[7:0]							
Access	R/W	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 – EPPTR[15:0] Endpoint Configuration Table Pointer
This bit field specifies the RAM address where the endpoint configuration table is located. The pointer to the endpoint configuration table must be aligned to a 16-bit word, i.e., EPPTR[0] must be zero. Only the number of bits required to address the device’s available internal RAM is implemented. Unused bits will always read as zero.

28.5.8 Interrupt Control A

Name:	INTCTRLA
Offset:	0x7
Reset:	0x00

Writing a '`1`' to any bit in this register enables the corresponding interrupt in the Interrupt Flags A (INTFLAGSA) register.

Writing a '`0`' to any bit in this register disables the corresponding interrupt in INTFLAGSA.

Bit	7	6	5	4	3	2	1	0
	SOF	SUSPEND	RESUME	RESET	STALLED	UNF	OVF	
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 – SOF Start-of-Frame Interrupt Enable

Bit 6 – SUSPEND Suspend Interrupt Enable

Bit 5 – RESUME Resume Interrupt Enable

Bit 4 – RESET Reset Interrupt Enable

Bit 3 – STALLED STALL Interrupt Enable

Bit 2 – UNF Underflow Interrupt Enable

Bit 1 – OVF Overflow Interrupt Enable

28.5.9 Interrupt Control B

Name:	INTCTRLB
Offset:	0x8
Reset:	0x00

Writing a '1' to any bit in this register enables the corresponding interrupt in the Interrupt Flags B (INTFLAGSB) register.

Writing a '0' to any bit in this register disables the corresponding interrupt in INTFLAGSB.

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

Bit 5 – TRNCOMPL Transaction Complete Interrupt Enable

Bit 1 – GNDONE GNACK Operation Done Interrupt Enable

Bit 0 – SETUP SETUP Transaction Complete Interrupt Enable

28.5.10 Interrupt Flags A

Name:	INTFLAGSA
Offset:	0x9
Reset:	0x00

Bit	7	6	5	4	3	2	1	0
	SOF	SUSPEND	RESUME	RESET	STALLED	UNF	OVF	
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 – SOF Start-of-Frame Interrupt Flag
This flag is set when a Start-of-Frame packet has been received.
Writing a '1' to this bit will clear this flag.

Bit 6 – SUSPEND Suspend Interrupt Flag
This flag is set when the bus has been idle for 3 ms.
Writing a '1' to this bit will clear this flag.

Bit 5 – RESUME Resume Interrupt Flag
This flag is set when a non-idle state has been detected on the bus while the USB peripheral is in the suspend state. This flag is set both in upstream and downstream resume. This interrupt is asynchronous and can wake the CPU from sleep modes where the system clock is stopped, such as Standby sleep mode.
Writing a '1' to this bit will clear this flag.

Bit 4 – RESET Reset Interrupt Flag
This flag is set when a reset condition has been detected on the bus.
Writing a '1' to this bit will clear this flag.

Bit 3 – STALLED STALL Interrupt Flag
This flag is set when the USB peripheral has responded with a STALL handshake to either an IN or an OUT transaction. The flag is cleared by writing this bit to '1'.

Bit 2 – UNF Underflow Interrupt Flag
This flag is set when the addressed endpoint in an IN transaction does not have data to send to the host. The flag is cleared by writing this bit to '1'.

Bit 1 – OVF Overflow Interrupt Flag
This flag is set when the addressed endpoint in an OUT transaction is not ready to accept data from the host. The flag is cleared by writing this bit to '1'.

28.5.11 Interrupt Flags B

Name:	INTFLAGSB
Offset:	0x0A
Reset:	0x00

Bit	7	6	5	4	3	2	1	0
			TRNCOMPL			RMWBUSY	GNDONE	SETUP
Access			R/W			R	R/W	R/W
Reset			0			0	0	0

Bit 5 – TRNCOMPL Transaction Complete Interrupt Flag

If FIFO is disabled This flag is set when the Transaction Complete (TRNCOMPL) flag in the Endpoint Status (EPn.STATUS) register is set while the Transaction Complete Interrupt Disable (TCDSBL) bit in the Endpoint Control (EPn.CTRL) register is '0'. Writing a '1' to this bit will clear this flag.

If FIFO is enabled This flag is set when there are unread elements in the FIFO. This flag is cleared when the last element has been read from the FIFO so that the FIFO is empty, i.e., the FIFO Read Pointer is equal to the FIFO Write Pointer.

Bit 2 – RMWBUSY RMW Busy Flag

This flag is set by an RMW operation initiated by writing to an RMW register. This flag is intended to be polled and does not have a corresponding Interrupt Enable bit in INTCTRLB. The RMW registers must not be written when this flag is set.

Value	Name	Description
0	NOTBUSY	Not Busy with RMW operation
1	BUSY	Busy with RMW operation

Bit 1 – GNDONE GNACK Operation Done Interrupt Flag

This flag is set when the Global NACK has been made effective, i.e., has propagated through the USB logic, and its effects can be seen on the USB bus. The flag is not set if Automatic GNACK (GNAUTO) is configured in the Control B (CTRLB) register.

Bit 0 – SETUP SETUP Transaction Complete Interrupt Flag

This flag is set when a SETUP transaction has successfully completed.
Writing a '1' to this bit will clear the flag.

28.5.12 Endpoint n OUT Status Clear

Name:	STATUS[n].OUTCLR
Offset:	0x40 + n*0x04 [n=0..15]
Reset:	0x00
Property:	W

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

Bits 7:0 – RMWSTATUS[7:0] Read-Modify-Write Endpoint n STATUS

Write bits in this bit field to `'1'` to perform the desired bitwise read-modify-write operation to the corresponding bit positions in the endpoint (EPn) STATUS register. Refer to the register description of the EPn.STATUS register for the individual bit field names and descriptions.

28.5.13 Endpoint n OUT Status Set

Name:	STATUS[n].OUTSET
Offset:	0x41 + n*0x04 [n=0..15]
Reset:	0x00
Property:	W

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

Bits 7:0 – RMWSTATUS[7:0] Read-Modify-Write Endpoint n STATUS
Write bits in this bit field to `'1'` to perform the desired bitwise read-modify-write operation to the corresponding bit positions in the endpoint (EPn) STATUS register. Refer to the register description of the EPn.STATUS register for the individual bit field names and descriptions.

28.5.14 Endpoint n IN Status Clear

Name:	STATUS[n].INCLR
Offset:	0x42 + n*0x04 [n=0..15]
Reset:	0x00
Property:	W

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

Bits 7:0 – RMWSTATUS[7:0] Read-Modify-Write Endpoint n STATUS

Write bits in this bit field to `'1'` to perform the desired bitwise read-modify-write operation to the corresponding bit positions in the endpoint (EPn) STATUS register. Refer to the register description of the EPn.STATUS register for the individual bit field names and descriptions.

28.5.15 Endpoint n IN Status Set

Name:	STATUS[n].INSET
Offset:	0x43 + n*0x04 [n=0..15]
Reset:	0x00
Property:	W

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

Bits 7:0 – RMWSTATUS[7:0] Read-Modify-Write Endpoint n STATUS
Write bits in this bit field to `'1'` to perform the desired bitwise read-modify-write operation to the corresponding bit positions in the endpoint (EPn) STATUS register. Refer to the register description of the EPn.STATUS register for the individual bit field names and descriptions.

28.6 Register Summary - USB_EP - Control, Bulk and Interrupt Endpoints

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
-0x20	FIFO31	7:0	EPNUM[3:0]				DIR			
...										
-0x01	FIFO0	7:0	EPNUM[3:0]				DIR			
0x00	EP[0].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x00	EP[0].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x00	Reserved									
0x01	EP[0].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x01	EP[0].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x02	EP[0].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x04	EP[0].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x06	EP[0].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x08	EP[0].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x08	EP[0].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x09	EP[0].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x09	EP[0].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x0A	EP[0].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x0C	EP[0].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x0E	EP[0].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x10	EP[1].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x10	EP[1].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x11	EP[1].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x11	EP[1].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x12	EP[1].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x14	EP[1].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x16	EP[1].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x18	EP[1].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x18	EP[1].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x19	EP[1].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x19	EP[1].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x1A	EP[1].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x1C	EP[1].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x1E	EP[1].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x20	EP[2].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x20	EP[2].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x21	EP[2].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x21	EP[2].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x22	EP[2].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x24	EP[2].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x26	EP[2].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x28	EP[2].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x28	EP[2].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x29	EP[2].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x29	EP[2].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		

...........continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2A	EP[2].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x2C	EP[2].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x2E	EP[2].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x30	EP[3].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x30	EP[3].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x31	EP[3].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x31	EP[3].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x32	EP[3].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x34	EP[3].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x36	EP[3].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x38	EP[3].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x38	EP[3].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x39	EP[3].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x39	EP[3].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x3A	EP[3].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x3C	EP[3].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x3E	EP[3].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x40	EP[4].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x40	EP[4].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x41	EP[4].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x41	EP[4].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x42	EP[4].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x44	EP[4].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x46	EP[4].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x48	EP[4].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x48	EP[4].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x49	EP[4].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x49	EP[4].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x4A	EP[4].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x4C	EP[4].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x4E	EP[4].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x50	EP[5].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x50	EP[5].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x51	EP[5].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x51	EP[5].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x52	EP[5].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x54	EP[5].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x56	EP[5].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x58	EP[5].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x58	EP[5].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x59	EP[5].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x59	EP[5].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		

...........continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x5A	EP[5].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x5C	EP[5].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x5E	EP[5].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x60	EP[6].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x60	EP[6].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x61	EP[6].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x61	EP[6].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x62	EP[6].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x64	EP[6].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x66	EP[6].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x68	EP[6].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x68	EP[6].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x69	EP[6].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x69	EP[6].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x6A	EP[6].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x6C	EP[6].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x6E	EP[6].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x70	EP[7].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x70	EP[7].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x71	EP[7].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x71	EP[7].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x72	EP[7].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x74	EP[7].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x76	EP[7].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x78	EP[7].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x78	EP[7].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x79	EP[7].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x79	EP[7].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x7A	EP[7].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x7C	EP[7].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x7E	EP[7].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x80	EP[8].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x80	EP[8].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x81	EP[8].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x81	EP[8].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x82	EP[8].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x84	EP[8].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x86	EP[8].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x88	EP[8].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x88	EP[8].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x89	EP[8].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x89	EP[8].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		

...........continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x8A	EP[8].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x8C	EP[8].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x8E	EP[8].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x90	EP[9].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x90	EP[9].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x91	EP[9].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x91	EP[9].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x92	EP[9].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x94	EP[9].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x96	EP[9].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x98	EP[9].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0x98	EP[9].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0x99	EP[9].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0x99	EP[9].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0x9A	EP[9].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x9C	EP[9].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0x9E	EP[9].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xA0	EP[10].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xA0	EP[10].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xA1	EP[10].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xA1	EP[10].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xA2	EP[10].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xA4	EP[10].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xA6	EP[10].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xA8	EP[10].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xA8	EP[10].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xA9	EP[10].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xA9	EP[10].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xAA	EP[10].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xAC	EP[10].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xAE	EP[10].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xB0	EP[11].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xB0	EP[11].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xB1	EP[11].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xB1	EP[11].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xB2	EP[11].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xB4	EP[11].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xB6	EP[11].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xB8	EP[11].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xB8	EP[11].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xB9	EP[11].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xB9	EP[11].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		

...........continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xBA	EP[11].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xBC	EP[11].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xBE	EP[11].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xC0	EP[12].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xC0	EP[12].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xC1	EP[12].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xC1	EP[12].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xC2	EP[12].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xC4	EP[12].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xC6	EP[12].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xC8	EP[12].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xC8	EP[12].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xC9	EP[12].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xC9	EP[12].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xCA	EP[12].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xCC	EP[12].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xCE	EP[12].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xD0	EP[13].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xD0	EP[13].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xD1	EP[13].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xD1	EP[13].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xD2	EP[13].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xD4	EP[13].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xD6	EP[13].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xD8	EP[13].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xD8	EP[13].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xD9	EP[13].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xD9	EP[13].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xDA	EP[13].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xDC	EP[13].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xDE	EP[13].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xE0	EP[14].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xE0	EP[14].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xE1	EP[14].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xE1	EP[14].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xE2	EP[14].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xE4	EP[14].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xE6	EP[14].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xE8	EP[14].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xE8	EP[14].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xE9	EP[14].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xE9	EP[14].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		

...........continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xEA	EP[14].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xEC	EP[14].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xEE	EP[14].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xF0	EP[15].OUT.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xF0	EP[15].OUT.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xF1	EP[15].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xF1	EP[15].OUT.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xF2	EP[15].OUT.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xF4	EP[15].OUT.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xF6	EP[15].OUT.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0xF8	EP[15].IN.STATUS	7:0		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
0xF8	EP[15].IN.STATUS	7:0	CRC	UNFOVF	TRNCOMPL				BUSNACK	
0xF9	EP[15].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
0xF9	EP[15].IN.CTRL	7:0	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
0xFA	EP[15].IN.CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0xFC	EP[15].IN.DATAPTR	7:0	DATAPTR[7:0]							
		15:8	DATAPTR[15:8]							
0xFE	EP[15].IN.MCNT	7:0	MCNT[7:0]							
		15:8	MCNT[15:8]							
0x0100	FRAMENUM	7:0	FRAMENUM[7:0]							
		15:8	FRAMEERR					FRAMENUM[10:8]		

28.7 Register Description - USB_EP - Control, Bulk and Interrupt Endpoints

28.7.1 FIFO Entry n

Name: FIFOn
Offset: -0x01 + n*-0x01 [n=0..31]
Reset: 0x00

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

Bits 7:4 – EPNUM[3:0] Endpoint Number
Endpoint number (address) in FIFO entry

Bit 3 – DIR Endpoint Direction
Direction of Endpoint in FIFO entry

Value	Name	Description
0	OUT	OUT Endpoint
1	IN	IN Endpoint

28.7.2 Endpoint Status - Default Type

Name:	EPn.STATUS
Offset:	0x00 + n*0x08 [n=0..31]
Reset:	0x00

This register description is valid for all modes except the Isochronous mode. When the Endpoint Type (TYPE) bit field in the Endpoint Control (EPn.CTRL) register is written to '`ISO`', see Endpoint Status - Isochronous Type for the correct description.

The Endpoint Status register is modified by the USB hardware. The application modifies this register using the RMW register interface to avoid data hazards. See the "*Endpoint Registers – Concurrency and Hazards*" section for more details. Flag/status registers in AVR are usually cleared by writing a '`1`' and can be set only by hardware. These status bits in RAM are an exception where you can write '`0`' to clear them or '`1`' to set them.

Bit	7	6	5	4	3	2	1	0
		UNFOVF	TRNCOMPL	EPSETUP	STALLED		BUSNACK	TOGGLE
Access		R/W	R/W	R/W	R/W		R/W	R/W
Reset		x	x	x	x		x	x

Bit 6 – UNFOVF Underflow / Overflow Endpoint Flag
The UNF flag is set when an input endpoint is not ready to send data to the host in response to an `IN` token for input endpoints.
The OVF flag is set when an output endpoint is not ready to accept data from the host following an `OUT` token for output endpoints.

Bit 5 – TRNCOMPL Transaction Complete Flag
This flag is set when an `IN` or `OUT` transaction on the endpoint has successfully completed. When setting this flag, the USB hardware will also set the Transaction Complete (TRNCOMPL) flag in the Interrupt Flags B (INTFLAGSB) register - unless the TRNCOMPL Interrupt Disable (TCDSBL) bit in the Endpoint Control (EPn.CTRL) register is '`1`'.
For multipacket transfers, TRNCOMPL is not set for each transaction but only when a multipacket transfer has been completed.

Bit 4 – EPSETUP Endpoint SETUP Complete Flag
This flag is set when a SETUP transaction has been completed successfully. When this flag is set, all `IN` and `OUT` transactions to control endpoints are responded to with a NAK.
This flag must be cleared by the application before decoding the request sent in the setup stage of a control transfer.

Bit 3 – STALLED Endpoint STALL Flag
This flag is set when a STALL handshake has been sent to the host in response to an `IN` or `OUT` transaction.

Bit 1 – BUSNACK Data Buffer Not Acknowledge
This flag is set when a packet has been received. When this bit is set the USB peripheral will discard incoming data to the data buffer in an `OUT` transaction and will not return any data from the data buffer in an `IN` transaction. For control, bulk and interrupt endpoints a NAK handshake is returned. After processing the packet, this flag must be cleared by the application. This bit can be written by the application to force returning a NAK to the host, e.g., during initialization. The setting of BUSNACK does not affect the ACKing of SETUP packets.

Bit 0 – TOGGLE Data Toggle

This indicates if a DATA0 or DATA1 PID is expected in the next data packet for an output endpoint, and if a DATA0 or DATA1 PID will be sent in the next transaction for an input endpoint. In most cases, TOGGLE is updated by hardware. The Programming Manual describes the exceptional cases where TOGGLE needs to be updated by the application.

Value	Name	Description
`0x0`	DATA0	Data packet DATA0
`0x1`	DATA1	Data packet DATA1

28.7.3 Endpoint Status - Isochronous Type

Name:	EPn.STATUS
Offset:	0x00 + n*0x08 [n=0..31]
Reset:	0x00

This register description is only valid when the Endpoint Type (TYPE) bit field in the Endpoint Control (EP.CTRL) register is written to 'ISO'. For the other endpoint types, see Endpoint Status - Default Type for the correct description.

The Endpoint Status register is modified by the USB hardware. The application modifies this register using the RMW register interface to avoid data hazards. See the "*Endpoint Registers – Concurrency and Hazards*" section for more details. Flag/status registers in AVR are usually cleared by writing a `'1'` and can be set only by hardware. These status bits in RAM are an exception where you can write `'0'` to clear them or `'1'` to set them.

Bit	7	6	5	4	3	2	1	0
	CRC	UNFOVF	TRNCOMPL				BUSNACK	
Access	R/W	R/W	R/W				R/W	
Reset	x	x	x				x	

Bit 7 – CRC CRC Error Flag

This flag is set for isochronous out endpoints when a CRC error has been detected in an incoming data packet.

Bit 6 – UNFOVF Underflow / Overflow Endpoint Flag

The UNF flag is set when an input endpoint is not ready to send data to the host in response to an `IN` token for input endpoints.
The OVF flag is set when an output endpoint is not ready to accept data from the host following an `OUT` token for output endpoints.

Bit 5 – TRNCOMPL Transaction Complete Flag

This flag is set when an `IN` or `OUT` transaction on the endpoint has successfully completed. When setting this flag, the USB hardware will also set the Transaction Complete (TRNCOMPL) flag in the Interrupt Flags B (INTFLAGSB) register - unless the TRNCOMPL Interrupt Disable (TCDSBL) bit in the Endpoint Control (EPn.CTRL) register is `'1'`.
For multipacket transfers, TRNCOMPL is not set for each transaction but only when a multipacket transfer has been completed.

Bit 1 – BUSNACK Data Buffer Not Acknowledge

This flag is set when a packet has been received. When this bit is set the USB peripheral will discard incoming data to the data buffer in an `OUT` transaction and will not return any data from the data buffer in an `IN` transaction. For control, bulk and interrupt endpoints a NAK handshake is returned. After processing the packet, this flag must be cleared by the application. This bit can be written by the application to force returning a NAK to the host, e.g., during initialization. The setting of BUSNACK does not affect the ACKing of SETUP packets.

28.7.4 Endpoint Control - Default Type

Name:	EPn.CTRL
Offset:	0x01 + n*0x08 [n=0..31]
Reset:	0x00

This register description is valid for all modes except the Isochronous mode. When the Endpoint Type (TYPE) bit field in this register is written to `'ISO'`, see Endpoint Control - Isochronous Type for the correct description.

Bit	7	6	5	4	3	2	1	0
	TYPE[1:0]		MULTIPKT	AZLP	TCDSBL	DOSTALL	BUFSIZE[1:0]	
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 7:6 – TYPE[1:0] Endpoint Type
These bits enable and select the endpoint type. If the endpoint is disabled, the remaining seven endpoint configuration bytes are never read or written by the USB peripheral, and their RAM locations are free to use for other application data.

Value	Name	Description
`0x0`	DISABLE	Endpoint Disabled
`0x1`	CONTROL	Control
`0x2`	BULKINT	Bulk or Interrupt
`0x3`	ISOCHRONOUS	Isochronous

Bit 5 – MULTIPKT Multipacket Transfer Enable
Writing this bit to `'1'` enables multipacket transfers. The multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without interrupts or firmware intervention. When MULTIPKT is `'1'`, MCNT defines the maximum size of a single OUT packet, thereby preventing overrun of the data reception buffer.

Bit 4 – AZLP Automatic Zero Length Packet
Writing this bit to `'1'` allows the USB peripheral to manage a transaction complete signaling with ZLP without firmware interaction. When Multipacket Transfer Enable (MULTIPKT) is disabled, this bit has no effect. The behavior depends on the transfer direction:
IN: An ZLP is automatically sent if the last data packet of a multipacket transfer is equal to ep_size.
OUT: If the last packet of a multipacket transfer was equal to ep_size, the Transaction Complete (TRNCOMPL) flag and interrupt will be set by the next ZLP.

Bit 3 – TCDSBL TRNCOMPL Interrupt Disable
When this bit is written to `'1'`, the Transaction Complete (TRNCOMPL) flag in the Interrupt Flags B (INTFLAGSB) register is not set together with the endpoint's TRNCOMPL flag in the Endpoint Status (EPn.STATUS) register. The FIFO will not store the endpoint configuration table address for this endpoint upon transaction complete if TCDSBL is `'1'`.

Bit 2 – DOSTALL Endpoint Will Respond with STALL
When this bit is written to `'1'`, the USB controller will respond to IN or OUT transactions with a STALL handshake. No STALL will be sent if the endpoint is a control endpoint and the Endpoint Setup (EPSETUP) bit in the EPn STATUS register is `'1'`. DOSTALL is direction-specific, i.e., IN transactions are stalled if IN.CTRL.DOSTALL is set and OUT transactions are stalled if OUT.CTRL.DOSTALL is set. STATUS.EPSETUP for control IN endpoints can override CTRL.DOSTALL, ensuring that the device doesn't STALL the data or status stage of a control transfer before the CPU has had time to process the request, in case CTRL.DOSTALL was set from the previous request.

Bits 1:0 – BUFSIZE[1:0] Data Size (Default)

This bit field configures the maximum data payload size for the endpoint. Incoming data bytes exceeding the maximum data payload size are discarded. For control endpoints, this field only affects IN and OUT transactions. For SETUP transaction, the maximum data payload is always eight bytes.

Value	Name	Description
`0x0`	DEFAULT_BUF8	8 bytes buffer size
`0x1`	DEFAULT_BUF16	16 bytes buffer size
`0x2`	DEFAULT_BUF32	32 bytes buffer size
`0x3`	DEFAULT_BUF64	64 bytes buffer size

28.7.5 Endpoint Control - Isochronous Type

Name: EPn.CTRL
Offset: 0x01 + n*0x08 [n=0..31]
Reset: 0x00

This register description is valid only when the Endpoint Type (TYPE) bit field in this register is written to 'ISO'. For the other endpoint types, see EPn.CTRL - Default Type for the correct description.

Bit	7	6	5	4	3	2	1	0
	TYPE[1:0]		MULTIPKT		TCDSBL	BUFSIZE[2:0]		
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	x	x	x		x	x	x	x

Bits 7:6 – TYPE[1:0] Endpoint Type
These bits enable and select the endpoint type. If the endpoint is disabled, the remaining seven endpoint configuration bytes are never read or written by the USB peripheral, and their RAM locations are free to use for other application data.

Value	Name	Description
`0x0`	DISABLE	Endpoint Disabled
`0x1`	CONTROL	Control
`0x2`	BULKINT	Bulk or Interrupt
`0x3`	ISOCHRONOUS	Isochronous

Bit 5 – MULTIPKT Multipacket Transfer Enable
Writing this bit to `'1'` enables multipacket transfers. The multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without interrupts or firmware intervention. When MULTIPKT is `'1'`, MCNT defines the maximum size of a single OUT packet, thereby preventing overrun of the data reception buffer.

Bit 3 – TCDSBL TRNCOMPL Interrupt Disable
When this bit is written to `'1'`, the Transaction Complete (TRNCOMPL) flag in the Interrupt Flags B (INTFLAGSB) register is not set together with the endpoint's TRNCOMPL flag in the Endpoint Status (EPn.STATUS) register. The FIFO will not store the endpoint configuration table address for this endpoint upon transaction complete if TCDSBL is `'1'`.

Bits 2:0 – BUFSIZE[2:0] Data Size (Isochronous)
This bit field configures the maximum data payload size for the endpoint. Incoming data bytes exceeding the maximum data payload size are discarded. For control endpoints, this field only affects IN and OUT transactions. For SETUP transaction, the maximum data payload is always eight bytes.

Value	Name	Description
`0x0`	ISO_BUF8	8 bytes buffer size
`0x1`	ISO_BUF16	16 bytes buffer size
`0x2`	ISO_BUF32	32 bytes buffer size
`0x3`	ISO_BUF64	64 bytes buffer size
`0x4`	ISO_BUF128	128 bytes buffer size
`0x5`	ISO_BUF256	256 bytes buffer size
`0x6`	ISO_BUF512	512 bytes buffer size
`0x7`	ISO_BUF1023	1023 bytes buffer size

28.7.6 Endpoint Byte Counter

Name:	EPn.CNT
Offset:	0x02 + n*0x08 [n=0..31]
Reset:	-

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	x	x	x	x	x	x	x	x

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	x	x	x	x	x	x	x	x

Bits 15:0 – CNT[15:0] Endpoint Data Byte Count

SETUP/OUT: The 16-bit value CNT holds the number of bytes received in the last OUT/SETUP transfer. For a multipacket transfer in progress it keeps the number of bytes received so far. It is updated by the hardware at the end of a packet.

IN: Holds the total number of bytes to send to the host in the next IN transfer. It is not changed by the hardware during MULTIPKT transfers.

CNT must only be written by software while the endpoint is deactivated, i.e., GNACK or NACK is set.

28.7.7 Endpoint Data Pointer

Name:	EPn.DATAPTR
Offset:	0x04 + n*0x08 [n=0..31]
Reset:	-

Bit	15	14	13	12	11	10	9	8
	DATAPTR[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
	DATAPTR[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 15:0 – DATAPTR[15:0] Endpoint Data Pointer

The 16-bit value DATAPTR contains the RAM address of the endpoint data buffer. For control endpoints, DATAPTR for the OUT endpoint is used for SETUP transactions, and DATAPTR for the IN endpoint is used for OUT and IN transactions.

28.7.8 Multipacket Byte Count

Name: EPn.MCNT
Offset: 0x06 + n*0x08 [n=0..31]
Reset: -

Bit	15	14	13	12	11	10	9	8
	MCNT[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
	MCNT[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 15:0 – MCNT[15:0] Multipacket Byte Count

For IN endpoints, MCNT holds the number of bytes that have been transferred by the USB so far. Write it to zero when setting up a new transfer. It is updated when a packet transfer is completed. MCNT holds the maximum number of received bytes for OUT endpoints. At most MCNT number of bytes is written to RAM. It will not be changed by hardware.

This bit field is only used if Multipacket Transfer Enable (MULTIPKT) in the Endpoint Control (EPn.CTRL) register is enabled. MCNT must only be changed by software while the endpoint is deactivated, i.e., GNACK or NACK is set.

When the multipacket transfer is not used this RAM location is free to use for other application data.

28.7.9 Frame Number

Name: FRAMENUM
Offset: 0x100
Reset: 0x00

Bit	15	14	13	12	11	10	9	8
	FRAMEERR					FRAMENUM[10:8]		
Access	R/W					R/W	R/W	R/W
Reset	x					x	x	x

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

Bit 15 – FRAMEERR Frame Error
This flag is set if a CRC or bit-stuffing error was detected in the most recently received Start-of-Frame packet.

Bits 10:0 – FRAMENUM[10:0] Frame Number
FRAMENUM holds the frame number from the most recently received Start-of-Frame packet.

29. CRCSCAN - Cyclic Redundancy Check Memory Scan

29.1 Features

- CRC-16-CCITT
- Check of the Entire Flash Section, Application Code, and/or Boot Section
- Selectable NMI Trigger on Failure
- User-Configurable Check During Internal Reset Initialization

29.2 Overview

The Cyclic Redundancy Check (CRC) is an important safety feature. It scans the Nonvolatile Memory (NVM) making sure the code is correct.

The device will not execute code if Flash fault has occurred. By ensuring no code corruption has occurred, a potentially unintended behavior in the application that can cause a dangerous situation can be avoided. The CRC scan can be set up to scan the entire Flash, only the boot section, or both the boot and application code sections.

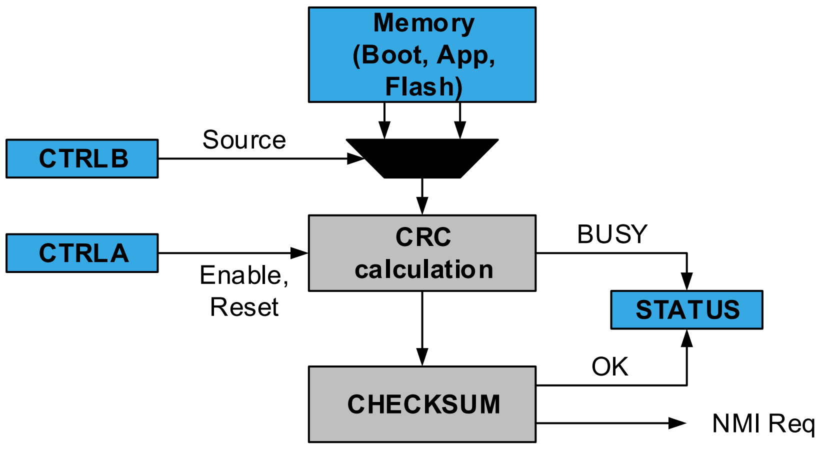
The CRC generates a checksum that is compared to a pre-calculated one. If the two checksums match, the Flash is OK, and the application code can start running.

The BUSY bit in the Status (CRCSCAN.STATUS) register indicates if a CRC scan is ongoing or not, while the OK bit in the Status (CRCSCAN.STATUS) register indicates if the checksum comparison matches or not.

The CRCSCAN can be set up to generate a Non-Maskable Interrupt (NMI) if the checksums do not match.

29.2.1 Block Diagram

Figure 29-1. Cyclic Redundancy Check Block Diagram



29.3 Functional Description

29.3.1 Initialization

To enable a CRC in software (or via the debugger):

1. Write the Source (SRC) bit field of the Control B (CRCSCAN.CTRLB) register to select the desired source settings.
2. Enable the CRCSCAN by writing a '1' to the ENABLE bit in the Control A (CRCSCAN.CTRLA) register.

3. The CRC will start after three cycles. The CPU will continue executing during these three cycles.

   The CRCSCAN can be configured to perform a code memory scan before the device leaves Reset. If this check fails, the CPU is not allowed to start normal code execution. This feature is enabled and controlled by the CRCSRC field in FUSE.SYSCFG0 (see the *Fuses* section for more information).

   If the CRCSCAN is enabled, a successful CRC check will have the following outcome:

   - Normal code execution starts
   - The ENABLE bit in CRCSCAN.CTRLA will be '1'
   - The SRC bit field in CRCSCAN.CTRLB will reflect the checked section(s)
   - The OK flag in CRCSCAN.STATUS will be '1'

   If the CRCSCAN is enabled, a non-successful CRC check will have the following outcome:

   - Normal code execution does not start. The CPU will hang executing no code.
   - The ENABLE bit in CRCSCAN.CTRLA will be '1'
   - The SRC bit field in CRCSCAN.CTRLB will reflect the checked section(s)
   - The OK flag in CRCSCAN.STATUS will be '0'
   - This condition may be observed using the debug interface

29.3.2 Operation

When operating, the CRCSCAN has priority access to the Flash and will stall the CPU until completed.

The CRC will use three clock cycles for each 16-bit fetch. The CRCSCAN can be configured to do a scan from start-up.

An *n*-bit CRC applied to a data block of arbitrary length will detect any single alteration (error burst) up to *n* bits in length. For longer error bursts a fraction $1\text{-}2^{-n}$ will be detected.

The CRC generator supports CRC-16-CCITT.

The polynomial options are:

- CRC-16-CCITT: $x^{16} + x^{12} + x^5 + 1$

The CRC reads byte-by-byte the content of the section(s) it is set up to check, starting with byte 0, and generates a new checksum per byte. The byte is sent through a shift register as depicted below, starting with the Most Significant bit. If the last bytes in the section contain the correct checksum, the CRC will pass. See 29.3.2.1. Checksum for how to place the checksum. The initial value of the Checksum register is 0xFFFF.

29.3.2.1 Checksum

The pre-calculated checksum must be present in the last location of the section to be checked. If the BOOT section is to be checked, the checksum must be saved in the last bytes of the BOOT section. The same is done for APPLICATION and the entire Flash. Table 29-1 shows explicitly how the checksum must be stored for the different sections. Refer to the CRCSCAN.CTRLB register description for how to configure the sections to be checked.

Table 29-1. Placement of the Pre-Calculated Checksum for CRC16 in Flash

Section to Check	CHECKSUM[15:8]	CHECKSUM[7:0]
BOOT	BOOTEND-1	BOOTEND
BOOT and APPLICATION	APPEND-1	APPEND
Full Flash	FLASHEND-1	FLASHEND

29.3.3 Interrupts

Table 29-2. Available Interrupt Vectors and Sources

Name	Vector Description	Conditions
NMI	Non-Maskable Interrupt	CRC failure

When the interrupt condition occurs the OK flag in the Status (CRCSCAN.STATUS) register is cleared to ‘0’.

A Non-Maskable Interrupt (NMI) is enabled by writing a ‘1’ to the respective Enable (NMIEN) bit in the Control A (CRCSCAN.CTRLA) register, but can only be disabled with a System Reset. An NMI is generated when the OK flag in the CRCSCAN.STATUS register is cleared, and the NMIEN bit is ‘1’. The NMI request remains active until a System Reset and cannot be disabled.

An NMI can be triggered even if interrupts are not globally enabled.

29.3.4 Sleep Mode Operation

In all CPU Sleep modes, the CRCSCAN is halted and will resume operation when the CPU wakes up.

The CRCSCAN starts operation three cycles after writing the Enable (ENABLE) bit in the Control A (CRCSCAN.CTRLA) register. During these three cycles, it is possible to enter Sleep mode. In this case:

1. The CRCSCAN will not start until the CPU is woken up.
2. Any interrupt handler will execute after CRCSCAN has finished.

29.3.5 Debug Operation

Whenever the debugger reads or writes a peripheral or memory location, the CRCSCAN will be disabled.

If the CRCSCAN is busy when the debugger accesses the device, the CRCSCAN will restart the ongoing operation when the debugger accesses an internal register or when the debugger disconnects.

The BUSY bit in the Status (CRCSCAN.STATUS) register will read ‘1’ if the CRCSCAN was busy when the debugger caused it to disable, but it will not actively check any section as long as the debugger keeps it disabled. There are synchronized CRC status bits in the debugger's internal register space, which can be read by the debugger without disabling the CRCSCAN. Reading the debugger's internal CRC status bits will make sure that the CRCSCAN is enabled.

It is possible to write the CRCSCAN.STATUS register directly from the debugger:

- BUSY bit in CRCSCAN.STATUS:
  - Writing the BUSY bit to ‘0’ will stop the ongoing CRC operation (so that the CRCSCAN does not restart its operation when the debugger allows it).
  - Writing the BUSY bit to ‘1’ will make the CRC start a single check with the settings in the Control B (CRCSCAN.CTRLB) register, but not until the debugger allows it.

  As long as the BUSY bit in CRCSCAN.STATUS is ‘1’, CRCSCAN.CTRLB and the Non-Maskable Interrupt Enable (NMIEN) bit in the Control A (CRCSCAN.CTRLA) register cannot be altered.
- OK bit in CRCSCAN.STATUS:
  - Writing the OK bit to ‘0’ can trigger a Non-Maskable Interrupt (NMI) if the NMIEN bit in CRCSCAN.CTRLA is ‘1’. If an NMI has been triggered, no writes to the CRCSCAN are allowed.
  - Writing the OK bit to ‘1’ will make the OK bit read as ‘1’ when the BUSY bit in CRCSCAN.STATUS is ‘0’.

Writes to CRCSCAN.CTRLA and CRCSCAN.CTRLB from the debugger are treated in the same way as writes from the CPU.

29.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RESET						NMIEN	ENABLE
0x01	CTRLB	7:0							SRC[1:0]	
0x02	STATUS	7:0							OK	BUSY

29.5 Register Description

29.5.1 Control A

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

If an NMI has been triggered this register is not writable.

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

Bit 7 – RESET Reset CRCSCAN

Writing this bit to '1' resets the CRCSCAN. The CRCSCAN Control and Status (CRCSCAN.CTRLA, CRCSCAN.CTRLB, CRCSCAN.STATUS) register will be cleared one clock cycle after the RESET bit is written to '1'.
If NMIEN is '0', this bit is writable both when the CRCSCAN is busy (the BUSY bit in CRCSCAN.STATUS is '1') and not busy (the BUSY bit is '0'), and will take effect immediately.
If NMIEN is '1', this bit is only writable when the CRCSCAN is not busy (the BUSY bit in CRCSCAN.STATUS is '0').
The RESET bit is a strobe bit.

Bit 1 – NMIEN Enable NMI Trigger

When this bit is written to '1', any CRC failure will trigger an NMI.
This bit can only be cleared by a System Reset. It is not cleared by a write to the RESET bit.
This bit can only be written to '1' when the CRCSCAN is not busy (the BUSY bit in CRCSCAN.STATUS is '0').

Bit 0 – ENABLE Enable CRCSCAN

Writing this bit to '1' enables the CRCSCAN with the current settings. It will stay '1' even after a CRC check has completed, but writing it to '1' again will start a new check.
Writing the bit to '0' has no effect.
The CRCSCAN can be configured to run a scan during the microcontroller (MCU) start-up sequence to verify the Flash sections before letting the CPU start normal code execution (see the 29.3.1. Initialization section). If this feature is enabled, the ENABLE bit will read as '1' when normal code execution starts.
To see whether the CRCSCAN is busy with an ongoing check, poll the BUSY bit in the Status (CRCSCAN.STATUS) register.

29.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

The Control B register contains the source settings for the CRC. It is not writable when the CRCSCAN is busy, or when an NMI has been triggered.

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

Bits 1:0 – SRC[1:0] CRC Source
The SRC bit field selects which section of the Flash will be checked by the CRCSCAN. To set up section sizes, refer to the *Fuses* section.
The CRCSCAN can be enabled during internal Reset initialization to verify Flash sections before letting the CPU start (see the *Fuses* section). If the CRCSCAN is enabled during internal Reset initialization, the SRC bit field will read out as FLASH, BOOTAPP, or BOOT when normal code execution starts (depending on the configuration).

Value	Name	Description
0x0	FLASH	The CRC is performed on the entire Flash (boot, application code, and application data sections).
0x1	BOOTAPP	The CRC is performed on the boot and application code sections of Flash.
0x2	BOOT	The CRC is performed on the boot section of Flash.
0x3	-	Reserved.

29.5.3 Status

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

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

Bit 1 – OK CRC OK

When this bit is read as '1', the previous CRC completed successfully. The bit is set to '1' by default before a CRC scan is run. The bit is not valid unless BUSY is '0'.

Bit 0 – BUSY CRC Busy

When this bit is read as '1', the CRCSCAN is busy. As long as the module is busy, the access to the control registers is limited.

30. CCL - Configurable Custom Logic

30.1 Features

- Glue Logic for General Purpose PCB Design
- Four 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

30.2 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.

30.2.1 Block Diagram

Figure 30-1. CCL Block Diagram

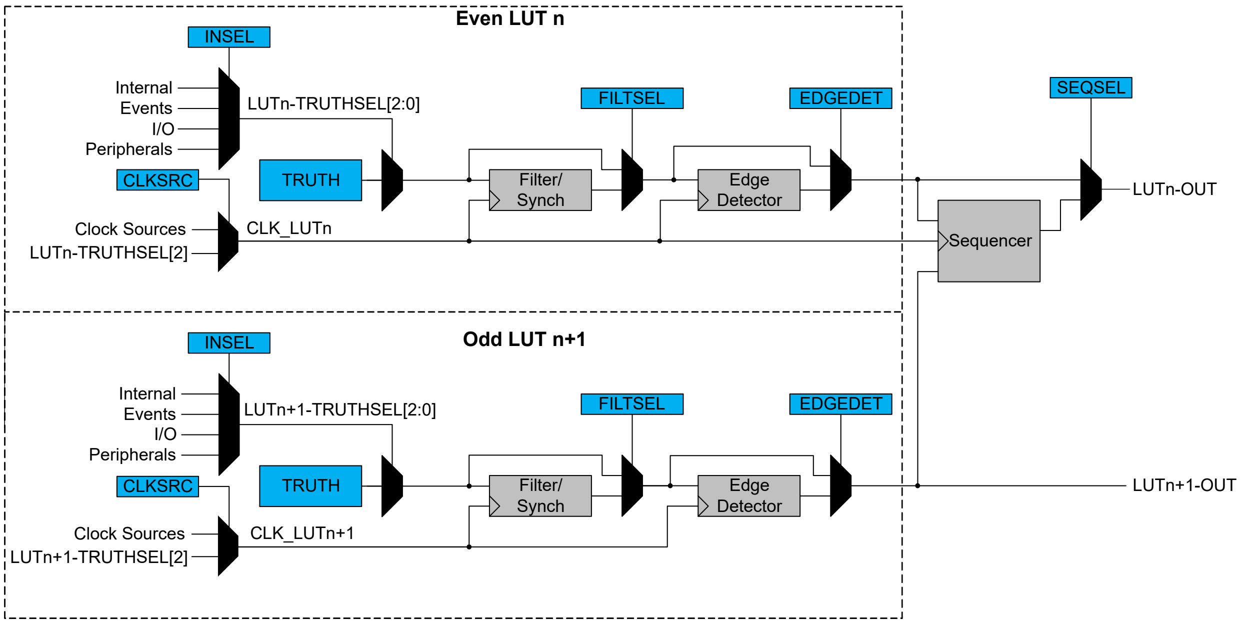


Table 30-2. Sequencer and LUT Connection

Sequencer	Even and Odd LUT
SEQ0	LUT0 and LUT1
SEQ1	LUT2 and LUT3

30.2.2 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.

30.2.2.1 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	AC0		AC0 OUT	
0x07	USARTn	USART0 TXD	USART1 TXD	USART1 TXD
0x08	SPI0	SPI0 MOSI	SPI0 MOSI	SPI0 SCK
0x09	TCA0	WO0	WO1	WO2
0x0A	TCBn	TCB0 WO	TCB1 WO	TCB2 WO

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.

30.3 Functional Description

30.3.1 Operation

30.3.1.1 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.

30.3.1.2 Enabling, Disabling, and Resetting

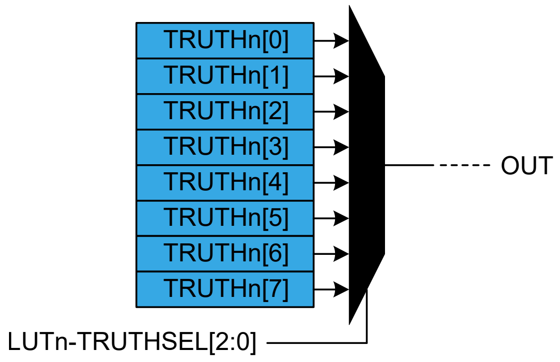
The CCL is enabled by writing a '1' to the ENABLE bit in the Control A (CCL.CTRLA) register. The CCL is disabled by writing a '0' to that ENABLE bit.

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.

30.3.1.3 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 30-2. 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 30-3. 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 30-1. LUT Output for CCL.TRUTHn = `0x42`

If CCL.TRUTHn is configured to `0x42`, the LUT output will be `1` when the inputs are `'b001` or `'b110` and `0` for any other combination of inputs.

30.3.1.4 Truth Table Inputs Selection

Input Overview

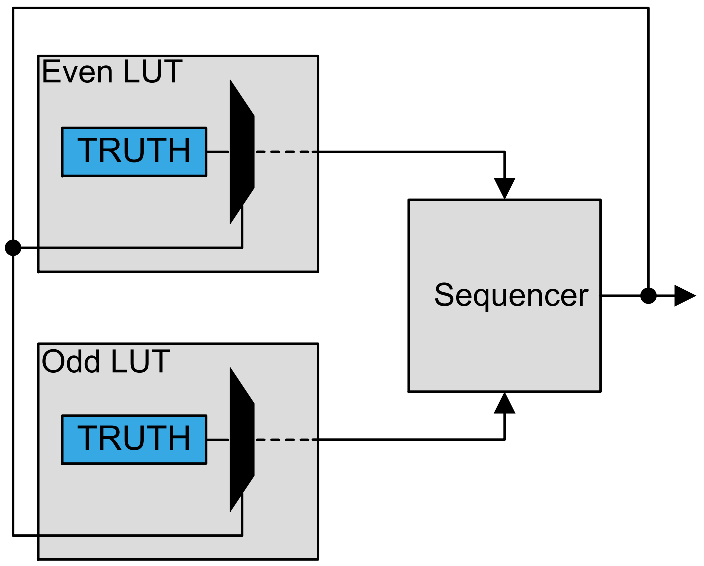
The inputs can be individually:

- 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 30-3. Feedback Input Selection



When selected (INSELy = FEEDBACK in LUTnCTRLx), the sequencer (SEQ) output is used as input for the corresponding LUTs.

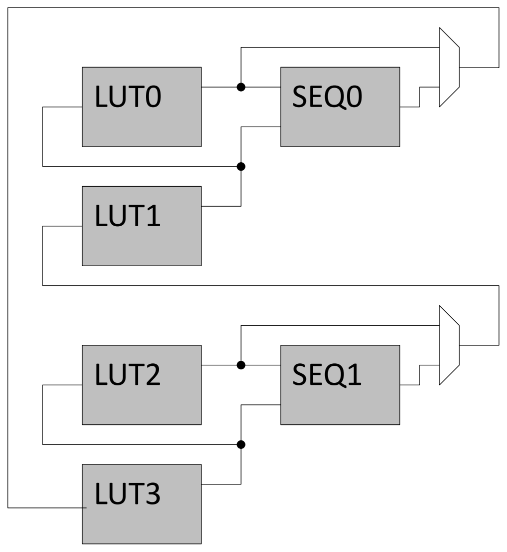
Linked LUT (LINK)

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 30-2. 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 30-4. 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.

30.3.1.5 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.

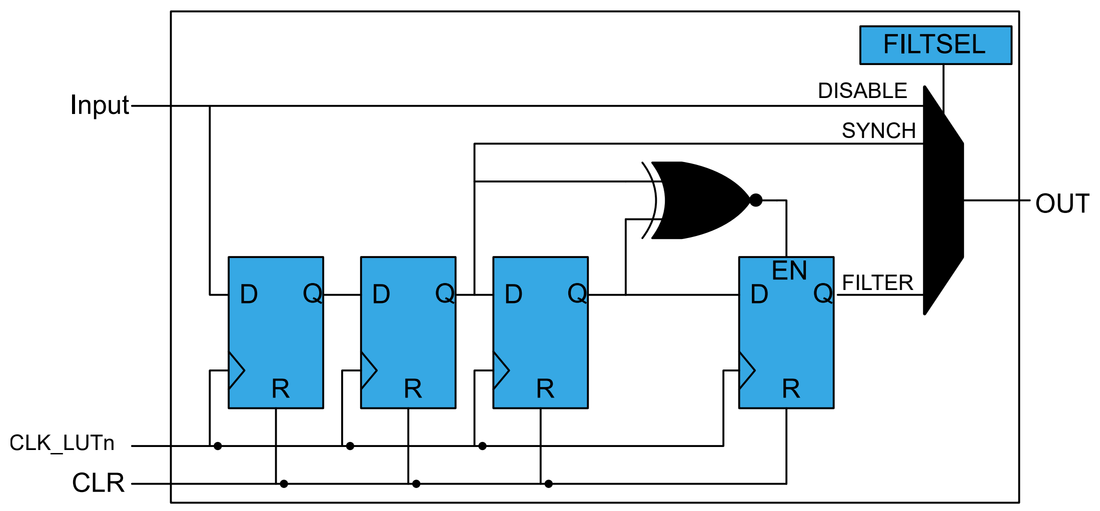
The Filter Selection (FILTSEL) bits in the LUT n Control A (CCL.LUTnCTRLA) registers define the digital filter options.

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 30-5. Filter



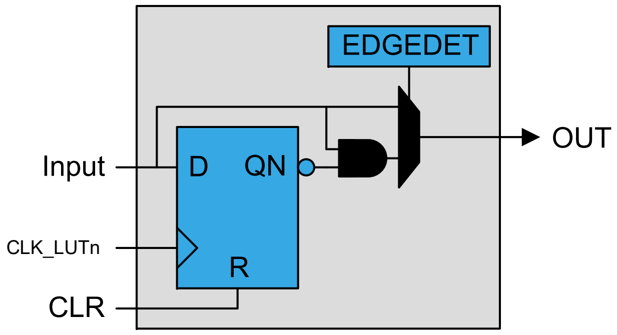
30.3.1.6 Edge Detector

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 30-6. Edge Detector



30.3.1.7 Sequencer Logic

Each LUT pair can be connected to a sequencer. The sequencer can function as either D flip-flop, JK flip-flop, gated D latch, or RS latch. The function is selected by writing the Sequencer Selection (SEQSEL) bit group in the Sequencer Control (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 (DFF)

The D input is driven by the even LUT output, and the G input is driven by the odd LUT output.

Figure 30-7. D Flip-Flop

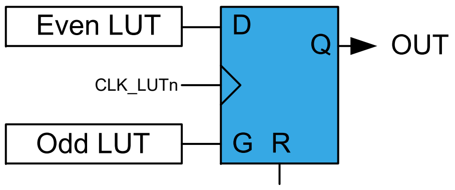


Table 30-4. DFF Characteristics

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
0	1	0	Clear
0	0	X	Hold state (no change)

JK Flip-Flop (JK)

The J input is driven by the even LUT output, and the K input is driven by the odd LUT output.

Figure 30-8. JK Flip-Flop

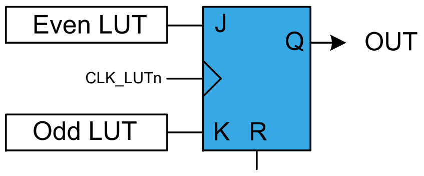


Table 30-5. JK Characteristics

R	J	K	OUT
1	X	X	Clear
0	0	0	Hold state (no change)
0	0	1	Clear
0	1	0	Set
0	1	1	Toggle

Gated D Latch (DLATCH)

The D input is driven by the even LUT output, and the G input is driven by the odd LUT output.

Figure 30-9. D Latch

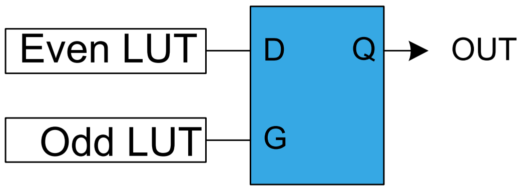


Table 30-6. D Latch Characteristics

G	D	OUT
0	X	Hold state (no change)
1	0	Clear
1	1	Set

RS Latch (RS)

The S input is driven by the even LUT output, and the R input is driven by the odd LUT output.

Figure 30-10. RS Latch

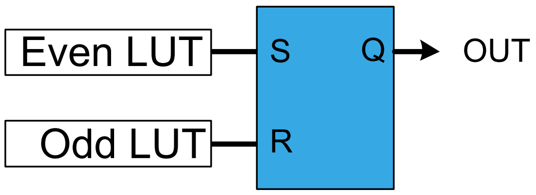


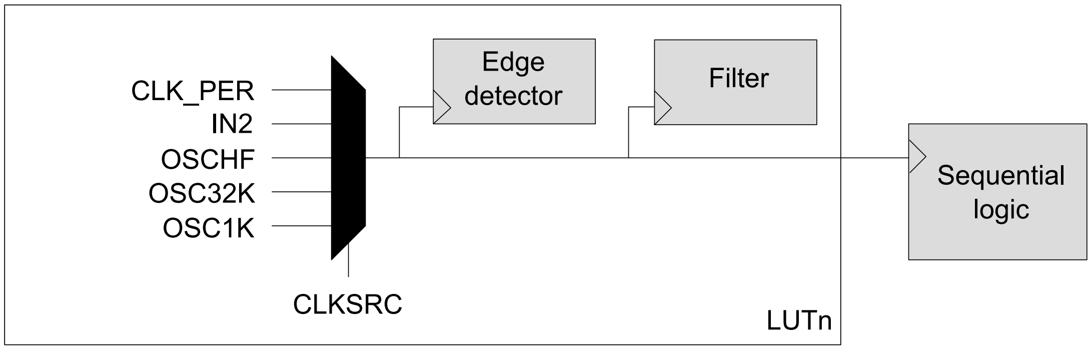
Table 30-7. RS Latch Characteristics

S	R	OUT
0	0	Hold state (no change)
0	1	Clear
1	0	Set
1	1	Forbidden state

30.3.1.8 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 30-11. 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.

30.3.2 Interrupts

Table 30-8. 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.

30.3.3 Events

The CCL can generate the events shown in the table below.

Table 30-9. 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 30-10. 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.

30.3.4 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.

30.4 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 ... 0x04	Reserved									
0x05	INTCTRL0	7:0	INTMODE3[1:0]		INTMODE2[1:0]		INTMODE1[1:0]		INTMODE0[1:0]	
0x06	Reserved									
0x07	INTFLAGS	7:0					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	TRUTH0[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	TRUTH1[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	TRUTH2[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	TRUTH3[7:0]							

30.5 Register Description

30.5.1 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

30.5.2 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

30.5.3 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

30.5.4 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

30.5.5 Interrupt Flag

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

Bit	7	6	5	4	3	2	1	0
					INT3	INT2	INT1	INT0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 0, 1, 2, 3 – 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.

30.5.6 LUT n Control A

Name:	LUTnCTRLA
Offset:	0x08 + n*0x04 [n=0..3]
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	Input Source	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

30.5.7 LUT n Control B

Name:	LUTnCTRLB
Offset:	0x09 + n*0x04 [n=0..3]
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`	AC0	AC0 OUT input source
`0x7`	USART1	USART1 TXD input source
`0x8`	SPI0	SPI0 MOSI input source
`0x9`	TCA0	TCA0 WO1 input source
`0xA`	TCB1	TCB1 WO 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`	USART0	USART0 TXD input source
`0x8`	SPI0	SPI0 MOSI input source
`0x9`	TCA0	TCA WO0 input source
`0xA`	TCB0	TCB0 WO input source
`Other`	-	Reserved

30.5.8 LUT n Control C

Name:	LUTnCTRLC
Offset:	0x0A + n*0x04 [n=0..3]
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`	AC0	AC0 OUT input source
`0x7`	USART1	USART1 TXD input source
`0x8`	SPI0	SPI0 SCK input source
`0x9`	TCA0	TCA0 WO2 input source
`0xA`	TCB1	TCB1 WO input source
`Other`	-	Reserved

30.5.9 TRUTHn

Name: TRUTHn
Offset: 0x0B + n*0x04 [n=0..3]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
	TRUTHn[7:0]							
Access	R/W	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 – TRUTHn[7:0] Truth Table
These bits determine the output of LUTn according to the LUTn-TRUTHSEL[2:0] inputs.

Bit Name	Value	Description
TRUTHn[0]	0	The output of LUTn is 0 when the inputs are 'b000
	1	The output of LUTn is 1 when the inputs are 'b000
TRUTHn[1]	0	The output of LUTn is 0 when the inputs are 'b001
	1	The output of LUTn is 1 when the inputs are 'b001
TRUTHn[2]	0	The output of LUTn is 0 when the inputs are 'b010
	1	The output of LUTn is 1 when the inputs are 'b010
TRUTHn[3]	0	The output of LUTn is 0 when the inputs are 'b011
	1	The output of LUTn is 1 when the inputs are 'b011
TRUTHn[4]	0	The output of LUTn is 0 when the inputs are 'b100
	1	The output of LUTn is 1 when the inputs are 'b100
TRUTHn[5]	0	The output of LUTn is 0 when the inputs are 'b101
	1	The output of LUTn is 1 when the inputs are 'b101
TRUTHn[6]	0	The output of LUTn is 0 when the inputs are 'b110
	1	The output of LUTn is 1 when the inputs are 'b110
TRUTHn[7]	0	The output of LUTn is 0 when the inputs are 'b111
	1	The output of LUTn is 1 when the inputs are 'b111

31. AC - Analog Comparator

31.1 Features

- Selectable Response Time
- Selectable Hysteresis
- Analog Comparator Output Available on Pin
- Comparator Output Inversion Available
- Flexible Input Selection:
  - Three positive pins
  - Three negative pins
  - Internal reference voltage generator (DACREF)
- Interrupt Generation on:
  - Rising edge
  - Falling edge
  - Both edges
- Event Generation:
  - Comparator output

31.2 Overview

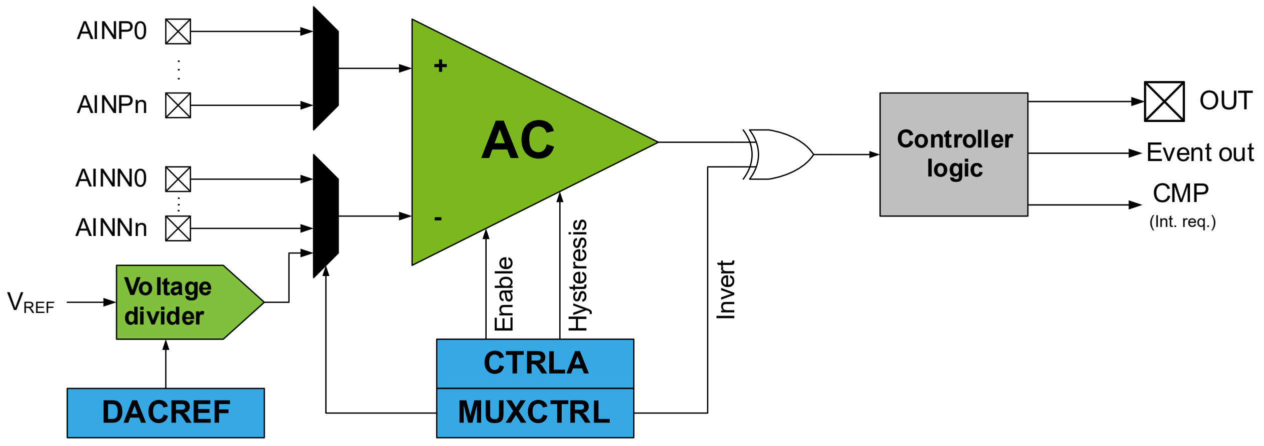
The analog comparator (AC) compares the voltage levels on two inputs and gives a digital output based on this comparison. The AC can be configured to generate interrupt requests and/or events based on several different combinations of input change.

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.

31.2.1 Block Diagram

Figure 31-1. AC Block Diagram



31.2.2 Signal Description

Signal	Description	Type
AINNn	Negative input n	Analog
AINPn	Positive input n	Analog
OUT	Comparator output of AC	Digital

31.3 Functional Description

31.3.1 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.

31.3.2 Operation

31.3.2.1 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.

31.3.2.2 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.

31.3.2.3 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.

31.3.2.4 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.

31.3.3 Events

The AC can generate the following events:

Table 31-1. 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.

31.3.4 Interrupts

Table 31-2. 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.

31.3.5 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.

31.4 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	Reserved									
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				CMPSTATE				CMPIF

31.5 Register Description

31.5.1 Control A

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

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	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. The shortest propagation delay t_{RESP} and the highest consumption.
`0x1`	PROFILE1	Power profile 1
`0x2`	PROFILE2	Power profile 2
`Other`	-	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.

31.5.2 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	-	Reserved
0x2	-	Reserved
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	AINN1	Negative pin 1
0x2	AINN2	Negative pin 2
0x3	-	Reserved
0x4	DACREF	DAC reference
Other	-	Reserved

31.5.3 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:

$$V_{\text{DACREF}} = \frac{\text{DACREF}[7:0]}{256} \times V_{\text{REF}}$$

31.5.4 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 triggers an interrupt request.

Table 31-3. 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.

31.5.5 Status

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

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

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. 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.

32. ADC - Analog-to-Digital Converter

32.1 Features

- 10-Bit, 170 ksps, Analog-to-Digital Converters (ADC) with Independent Voltage Reference Sources
  - Up to 13 bits with oversampling
- Conversion Rate Up to 170 ksps, at 10-bit Resolution
- Up to 21 I/O Pin Inputs, in Addition to Internal Inputs from Sensors and References
- Single-Ended Conversion
- Multiple Internal ADC Reference Voltages[1]
  - V_{DD}
  - 1.024V
  - 2.048V
  - 2.500V
  - 4.096V
- External Reference Input
- Single and Free-Running Conversions
- Series and Burst Accumulation Modes
- Accumulation of Up to 64 Conversions
- Left or Right Adjusted Result
- Interrupts on Conversion Complete
- Optional Event Triggered Conversion
- Configurable Window Comparator

Note: The internal voltage reference sources for ADC0 and ADC1 are independent.

32.2 Overview

The Analog-to-Digital Converter (ADC) peripheral features a 10-bit single-ended ADC with 10-bit resolution. The ADC is connected to an analog input multiplexer for selection between multiple inputs. The ADC measures the voltage between the selected input and 0V (GND). The ADC inputs can be internal (for example, a voltage reference) or external analog input pins.

An ADC conversion can be started by software or by using the Event System (EVSYS) to route an event from other peripherals. This makes it possible to sample input signals periodically, trigger an ADC conversion on a particular condition, and trigger ADC conversions in Standby sleep mode. A digital window compare feature is available for monitoring the input signal. It can also 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, with minimum software intervention required.

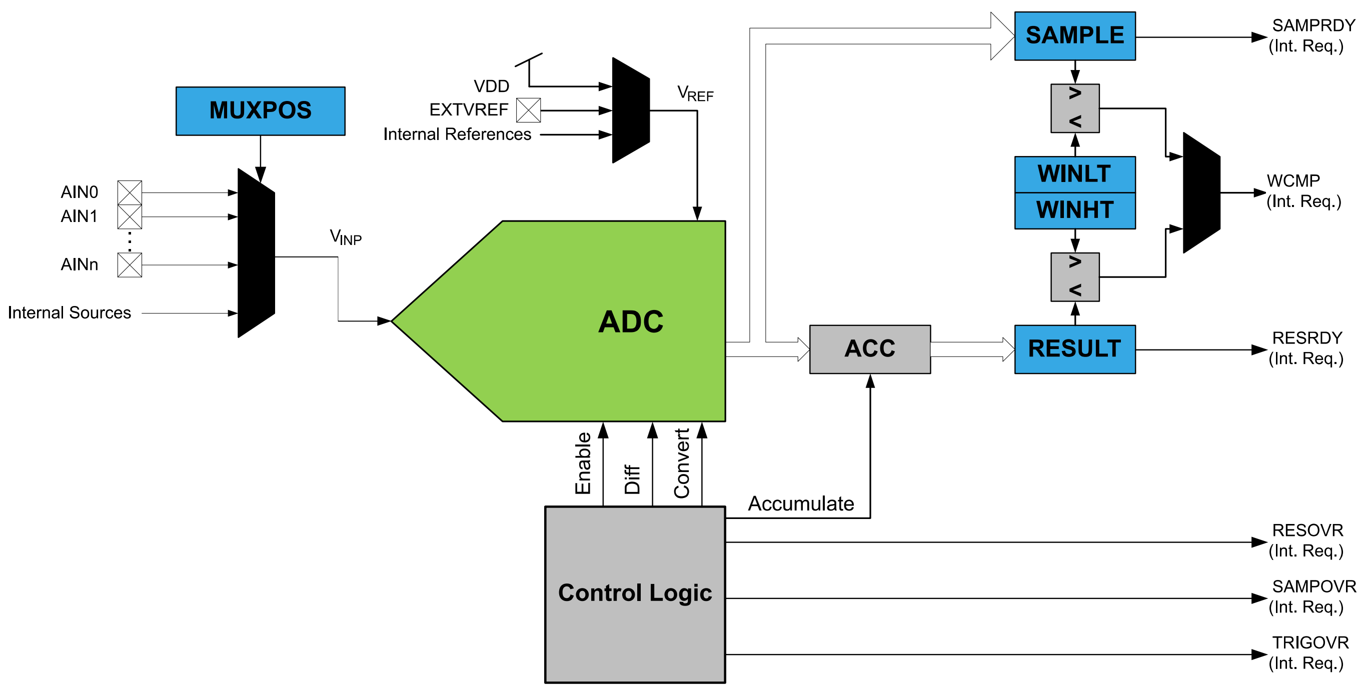
The ADC input signal is fed through a sample-and-hold circuit that ensures that the input voltage to the ADC is kept at a constant level during the conversion.

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).

32.2.1 Block Diagram

Figure 32-1. Block Diagram



32.2.2 Signal Description

Pin Name	Type	Description
AIN[n:0]	Analog input	Analog input pin
EXTVREF	Analog input	External voltage reference pin

32.3 Functional Description

32.3.1 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.

32.3.2 Basic Operation

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 TIMEBASE (CLKCTRL.TIMEBASE) register of the Clock Controller.
2. Enable the ADC by writing a ‘1’ to the ENABLE bit in the Control A (ADCn.CTRLA) register.
3. Configure the Prescaler (PRESC) bit field in the Control B (ADCn.CTRLB) register.
4. Configure the Reference Select (REFSEL) bit field in the Control C (ADCn.CTRLC) register.

5. Configure the Sample Duration (SAMPDUR) bit field in the Control E (ADCn.CTRLE) register.
6. 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.
7. Optional: Enable the Free-Running mode by writing a '1' to the Free-Running (FREERUN) bit in the Control F (ADCn.CTRLF) register.
8. Configure a positive input by writing to the MUXPOS bit field in the Positive Input Multiplexer (ADCn.MUXPOS) register.
9. Configure the mode of operation for the ADC by writing to the MODE bit field in the Command (ADCn.COMMAND) register.
10. Configure how an ADC conversion will start by writing to the START bit field in the Command (ADCn.COMMAND) register. If the IMMEDIATE command is written, a conversion will start immediately.
11. Wait until the Result Ready (RESRDY) bit in the Interrupt Flags (ADCn.INTFLAGS) register is '1' before reading the updated Result (ADCn.RESULT) register.

32.3.3 Operation

32.3.3.1 Operation Modes

The ADC supports four different operation modes, with single-ended conversions possible for each mode. This is 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 and can be configured with the accumulated result. The number of samples to accumulate is controlled by the SAMPNUM bit field in the Control F (ADCn.CTRLF) register. 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 32-1. Operation Modes

Operation Mode	COMMAND Mode	Conversions Per Trigger	Accumulation Type	RESULT Update
Single 8-bit	0	1	N/A	Every conversion
Single 10-bit	1			
Series Accumulation	2	1	Full	After SAMPNUM conversions
Burst Accumulation	3	SAMPNUM	Full	After SAMPNUM conversions

Table 32-2. Effect of FREERUN and LOWLAT

Operation Mode	START/Trigger	FREERUN	LOWLAT	Behavior
Series	IMMEDIATE/ MUX_WRITE/ EVENT_TRIGGER	X	0	Trigger one conversion in a series. ADC warm-up is required before each conversion, and the ADC is not kept warm after the conversion is complete.
		X	1	Trigger one conversion in a series. The ADC is kept warm between conversions and after the series accumulation has been completed.(1)

...........continued

Operation Mode	START/Trigger	FREERUN	LOWLAT	Behavior
Burst	IMMEDIATE/ MUX_WRITE/ EVENT_TRIGGER	0	0	Start a burst accumulation. ADC warm-up is required before the first conversion and the ADC is not kept warm after completing the burst accumulation.
		0	1	Start a burst accumulation. ADC warm-up is required before the first conversion and the ADC is kept warm after completing the burst accumulation.
		1	X	Start a burst accumulation. No ADC warm-up is required before the first conversion if LOWLAT is set to '1' and the ADC has had time to warm up. A new trigger is automatically issued when a burst accumulation is complete such that a new burst accumulation is started immediately, always keeping the ADC warm.(1)

Note:
1. There may be an ADC warm-up on the first conversion if the ADC was cold at that point. Enabling LOWLAT does not force the main clock to run when the ADC is not converting. The main clock source will stop if no other peripheral is requesting it. Therefore, a clock source startup time delay may be experienced when triggering a conversion even though LOWLAT is enabled. The main clock must be kept running even if the ADC is not requesting it to avoid this delay. This can be achieved by either enabling RUNSTDBY in the clock controller, using the IDLE sleep mode, or ensuring that another peripheral is requesting the clock.

32.3.3.1.1 Accumulator Test

The Accumulator Test (ACCTEST) mode aids in diagnostics of the accumulator and result formatting hardware and can, combined with targeted input stimulus, achieve high fault coverage quickly. It is configured in the MODE bitfield in the Command (ADCn.COMMAND) register.

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 will accumulate the written value in the internal 16-bit accumulator
- Reading the RESULT register will return the accumulated result

Note: 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 it is accumulated.

32.3.3.2 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 FREERUN bit in the Control F (ADCn.CTRLF) register before starting the first conversion. This bit does not affect Series accumulation.

An ongoing conversion can be aborted by writing the STOP value to the START bit field in the Command (ADCn.COMMAND) register and a new conversion can be started immediately. Triggering 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.

32.3.3.3 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 (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 and SAMPLE registers. A new conversion will start immediately, since the settling time associated with the updated parameter must pass before the ADC is ready.

32.3.3.4 Output Formats

The output from an ADC conversion is given by the following equations:

Equation 32-1. Single-Ended 10-bit Conversion

$$\mathrm{RES} = \frac{V_{INP}}{V_{REF}} * 1024 \qquad \{RES \in \mathbb{Z} : 0 \le RES \le 1023\}$$

Equation 32-2. Single-Ended 8-bit Conversion

$$\mathrm{RES} = \frac{V_{INP}}{V_{REF}} * 256 \qquad \{RES \in \mathbb{Z} : 0 \le RES \le 255\}$$

Where V_{INP} is the positive input to the ADC, *RES* is the conversion result, and V_{REF} is the selected voltage reference.

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). All accumulation modes will accumulate samples in an internal sample accumulator, configured by the Sample Accumulation Number Select (SAMPNUM) bit field in the Control F (ADCn.CTRLF) register. The sample accumulator is sufficiently wide to avoid overflow for all supported accumulation configurations. The accumulated result will automatically be transferred to the 16-bit Result register at the end of a Burst or Series mode accumulation. In single conversion modes, the Result register will be updated with the latest sample, identical to the Sample (ADCn.SAMPLE) register.

The Left Adjust (LEFTADJ) bit in the Control F (ADCn.CTRLF) register enables left shift of the output data in the modes where this is supported. If enabled, this will left shift the output from both the Result (ADCn.RESULT) and the Sample (ADCn.SAMPLE) registers.

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 32-3. RESULT Register

MODE	LEFTADJ	RES[15:12]	RES[11:8]	RES[7:0]
0	X(1)	0x00		Conversion[7:0]
1	0	0x00	Conversion[9:0]	
	1	Conversion[9:0] << 4		
2, 3	X(1)	Accumulation[15:0]		

Note:
1. Left adjust is not available in 8-bit mode or accumulation modes.

The following table shows the Sample register output formats by mode of operation and left adjustment.

Table 32-4. SAMPLE Register

MODE	LEFTADJ	SAMPLE[15:12]	SAMPLE[11:8]	SAMPLE[7:0]
0	X	0x00		Conversion[7:0]
		Sign extension		Signed conversion[7:0]
Other	0	0x00	Conversion[9:0]	
		Sign extension	Signed conversion[9:0]	
	1	Conversion[9:0] << 4		
		Signed conversion[9:0] << 4		

32.3.3.5 ADC Clock

The ADC clock (CLK_ADC) is scaled down from the peripheral clock (CLK_PER). This can be configured by the Prescaler (PRESC) bit field in the Control B (ADCn.CTRLB) register. The limitations of the *ADC Conversion Timing Specifications* in the ADC section of the *Electrical Characteristics* apply.

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 TIMEBASE register in the Clock Controller (CLKCTRL) peripheral. Refer to the *TIMEBASE register description in the CLKCTRL* section for details.

Round up the timebase to the closest integer. The following code snippet shows how to do this using the `ceil` function.

```
#include <math.h>
        #define CLK_PER 3333333ul // 20 MHz/6 = 3.333333 MHz
        #define TIMEBASE_VALUE ((uint8_t) ceil(CLK_PER*0.000001))
```

32.3.3.6 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. V_{DD} must be at least 0.5V higher than the selected internal reference voltage.

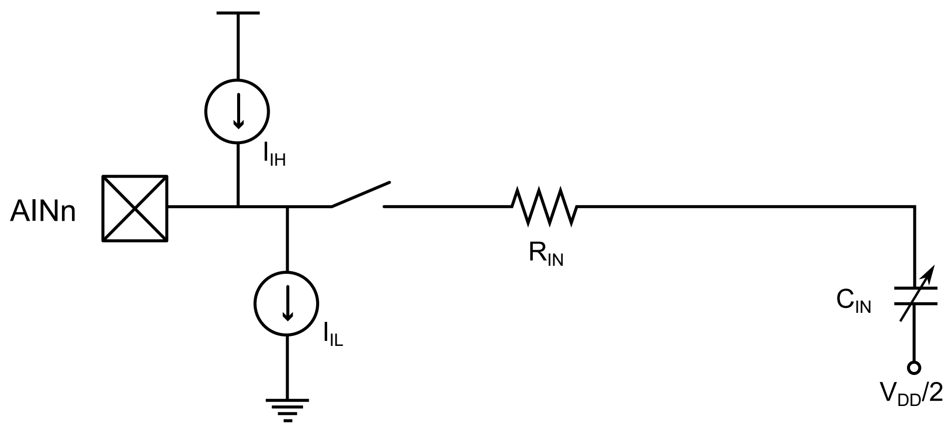
The input and reference selections are not buffered. Changing any of these while a conversion is ongoing will corrupt the output. To safely change input or reference when using Free-Running mode, disable Free-Running mode, and wait for the conversion to complete before making any changes. Enable Free-Running mode again before starting the next conversion.

After switching input or reference, the ADC requires time to settle. Refer to the *Electrical Characteristics* section for further details.

32.3.3.6.1 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 32-2. 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.

32.3.3.7 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. The following table shows the different initialization times needed by the analog modules.

Table 32-5. ADC Initialization Timing

Analog Module	LOWLAT	Initialization Time (µs)
ADC	0	6
	1[(2)]	0
Settling of internal references	0	40
	1[(2)]	2[(1)]
Settling internal Temperature Sensor input	0	40
	1[(2)]	0
Settling of internal Analog Comparator Reference DAC input	N/A	20

Notes:
1. The 2 µs timing is when changing from one internal reference voltage to another internal reference voltage. If changing from a non-internal reference to an internal reference, the full 40 µs delay will be used.
2. The LOWLAT timing values are valid between two conversions that both have the LOWLAT bit written to '1'.

Example: Selecting Tempsense as input and using V_{DD} as the reference will give a 40 µs initialization time. Using the Tempsense with the 1.024V internal reference will result in a 40 µs initialization time.

The ADC can be put in Low-Latency mode by writing a '1' to the LOWLAT bit in the Control A (ADCn.CTRLA) register. 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, as shown in the table above. The ADC Busy (ADCBUSY) bit in the Status (ADCn.STATUS) register can be used to check if initialization is ongoing.

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 required by the input signal characteristics, SAMPDUR can be increased.

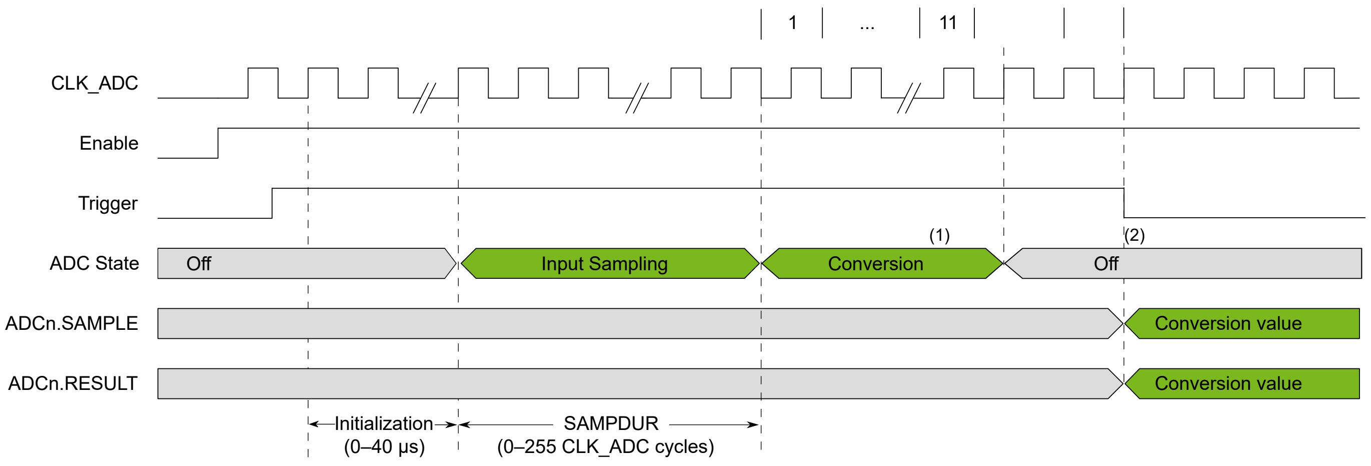
If an internal reference is used, the value of the SAMPDUR bit field in the Control E (ADCn.CTRLE) register must be ≥ 4 μs * f_{CLK_ADC}.

The internal reference has an offset cancellation mechanism with a limited hold time. Offset cancellation is performed at the start of every conversion and is valid for the threshold afterward. The customer should ensure that his conversion has been completed before the threshold has expired by appropriate selection of ADC clock frequency and SAMPDUR. Refer to the *Electrical Characteristics* for the value of the threshold.

32.3.3.7.1 Single Conversion

The figure below shows the timing diagram for the ADC when running in Single 8- or 10-bit mode.

Figure 32-3. Timing Diagram - Single Conversion



Notes:

1. In Single 8-bit mode, the length of the Conversion state is nine CLK_ADC cycles. In all other modes, it is thirteen cycles.
2. 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.
3. The time from the conversion has finished to the outputs are available in the registers is 0.5 CLK_ADC cycles followed by 1 CLK_MAIN cycle. With minimum prescaling, this sums up to 1 CLK_ADC cycle.

The total conversion time for a single result is calculated by:

$$t_{conv}\,(\text{10-bit}) = t_{initialization} + \frac{\text{SAMPDUR}+13}{f_{\text{CLK\_ADC}}}$$

$$t_{conv}\,(\text{8-bit}) = t_{initialization} + \frac{\text{SAMPDUR}+11}{f_{\text{CLK\_ADC}}}$$

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:

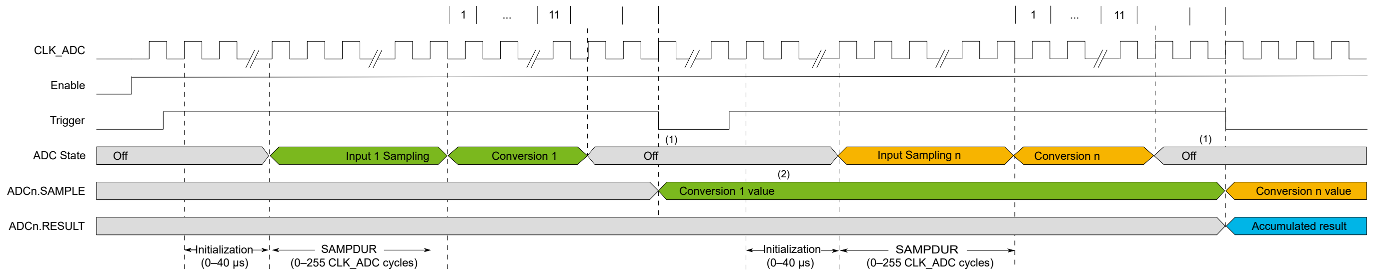
$$f_{\text{conv}}\,(10\text{-bit}) = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR}+13}$$

$$f_{\text{conv}}\,(8\text{-bit}) = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR}+11}$$

32.3.3.7.2 Series Accumulation

The figure below shows the timing diagram for the ADC when running in Series Accumulation mode.

Figure 32-4. Timing Diagram - Series Accumulation



Notes:

1. 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, eliminating the initialization time when triggering the following conversion.
2. The time from the conversion has finished to the outputs are available in the registers is 0.5 CLK_ADC cycles followed by 1 CLK_MAIN cycle. The last conversion and accumulation require an additional CLK_MAIN cycle. With minimum prescaling, this sums up to 1.5 CLK_ADC cycles before the final outputs are available.

The number of samples to accumulate is set by the Sample Number (SAMPNUM) bit field in the Control F (ADCn.CTRLF) register.

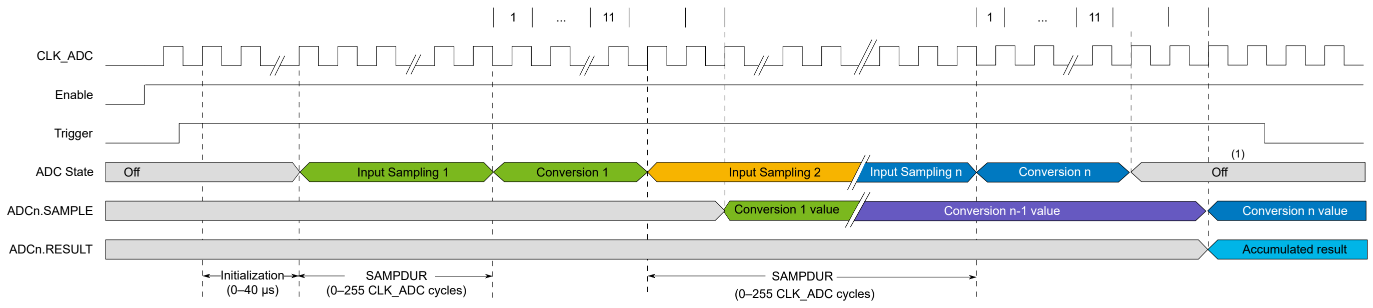
The total conversion time for each separate sample is calculated by:

$$t_{conv} = t_{initialization} + \frac{\text{SAMPDUR}+13}{f_{\text{CLK\_ADC}}}$$

32.3.3.7.3 Burst Accumulation

The figure below shows the timing diagram for the ADC when running in Burst Accumulation mode.

Figure 32-5. Timing Diagram - Burst Accumulation



Notes:

1. 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, eliminating the initialization time when triggering the following conversion.
2. The time from the conversion has finished to the outputs are available in the registers is 0.5 CLK_ADC cycles followed by 1 CLK_MAIN cycle. The last conversion and accumulation require an additional CLK_MAIN cycle. With minimum prescaling, this sums up to 1.5 CLK_ADC cycles before the final outputs are available.

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:

$$t_{conv} = t_{initialization} + \frac{(\mathrm{SAMPDUR} + 12) \times \mathrm{SAMPNUM} \; + \; 1.5}{f_{\mathrm{CLK\_ADC}}}$$

The Burst Accumulation conversion rate (f_{conv}) is calculated by:

$$f_{\mathrm{conv}} = \frac{f_{\mathrm{CLK\_ADC}}}{\mathrm{SAMPDUR+12}}$$

32.3.3.8 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 the 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 $\geq 35\ \mu\mathrm{s} \times f_{\mathrm{CLK\_ADC}}$ 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 an almost linear relationship with the temperature. Due to process variations, the temperature sensor output voltage varies between individual devices at the same temperature. The individual compensation factors determined during production test are stored in the Signature Row. These compensations factors are generated for the internal 2.048V reference.

- SIGROW.TEMPSENSE0 contains the slope of the temperature sensor characteristics
- SIGROW.TEMPSENSE1 contains the offset of the temperature sensor characteristics

In order to achieve more accurate results, the result of the temperature sensor measurement must be processed in the application software using compensation values from device production or user calibration.
The temperature (in kelvin) is calculated by the following equation:

$$T = \frac{(\mathrm{Offset} - \mathrm{ADC\ Result}) \times \mathrm{Slope}}{1024}$$

It is recommended to follow these steps in the application code when using the compensation values from the Signature Row:

```
#define SCALING_FACTOR 1024 // Used to get a whole number in the signature row

uint16_t sigrow_offset = SIGROW.TEMPSENSE1; // Read unsigned offset from signature row
uint16_t sigrow_slope = SIGROW.TEMPSENSE0;  // Read unsigned gain/slope from signature row
uint16_t adc_reading = ADC0.RESULT;         // ADC0 conversion result

uint32_t temp = sigrow_offset - adc_reading;
temp *= sigrow_slope;   // Result will overflow 16-bit variable
temp += SCALING_FACTOR / 2;          // Ensure correct rounding on division below
temp /= SCALING_FACTOR;              // Round off to nearest degree in kelvin
uint16_t temperature_in_K = temp;
int16_t temperature_in_C = temp - 273;
```

If another reference (V_{ADCREF}) than 2.048V is required, the offset and slope values need to be adjusted according to the following equations:

$$\text{Slope} = \text{TEMPSENSE0} \times \frac{V_{ADCREF}}{2.048V}$$

$$\text{Offset} = \text{TEMPSENSE1} \times \frac{2.048V}{V_{ADCREF}}$$

32.3.3.9 Window Comparator

The ADC can raise the Window Comparator Interrupt (WCMP) flag in the Interrupt Flags (ADCn.INTFLAGS) register and request an interrupt (WCMP) when the output of a conversion or accumulation is above and/or below certain thresholds. The available modes are:

- The 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.

32.3.4 Events

The ADC can generate the following events:

Table 32-6. Event Generators in ADC

Generator Name		Description	Event Type	Generating Clock Domain	Length of Event
Peripheral	Event				
ADCn	RESRDY	Result ready	Pulse	CLK_PER	One clock period
	SAMPRDY	Sample ready	Pulse	CLK_PER	One clock period
	WCMP	Window compare	Pulse	CLK_PER	One clock period

The conditions for generating an event are identical to those that will raise the corresponding flag in the Interrupt Flags (ADCn.INTFLAGS) register.

The ADC has one event user for detecting and acting upon input events. The table below describes the event user and the associated functionality.

Table 32-7. 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.

32.3.5 Interrupts

Table 32-8. Available Interrupt Vectors and Sources

Name	Vector Description	Interrupt Flag	Conditions
ERROR	Error interrupt	TRIGOVR	A new conversion is triggered while another is in progress
		SAMPOVR	A new conversion overwrites an unread sample in ADCn.SAMPLE
		RESOVR	A new conversion or accumulation overwrites an unread result in ADCn.RESULT
SAMPRDY	Sample Ready interrupt	SAMPRDY	The sample is available in ADCn.SAMPLE
		WCMP	As defined by WINSRC and WINCM in ADCn.CTRLD
RESRDY	Result Ready interrupt	RESRDY	The result is available in ADCn.RESULT
		WCMP	As defined by WINSRC and WINCM in ADCn.CTRLD

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 ADC will finish a conversion before going to Idle/Standby sleep mode. 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.

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.

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.

32.3.7 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.

32.4 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	INTCTRL	7:0			TRIGOVR	SAMPOVR	RESOVR	WCMP	SAMPRDY	RESRDY
0x05	INTFLAGS	7:0			TRIGOVR	SAMPOVR	RESOVR	WCMP	SAMPRDY	RESRDY
0x06	STATUS	7:0								ADCBUSY
0x07	DBGCTRL	7:0								DBGRUN
0x08	CTRLE	7:0	SAMPDUR[7:0]							
0x09	CTRLF	7:0			FREERUN	LEFTADJ		SAMPNUM[2:0]		
0x0A	COMMAND	7:0		MODE[2:0]				START[2:0]		
0x0B	Reserved									
0x0C	MUXPOS	7:0			MUXPOS[5:0]					
0x0D ... 0x0F	Reserved									
0x10	RESULT	7:0	RESULT[7:0]							
		15:8	RESULT[15:8]							
0x12 ... 0x13	Reserved									
0x14	SAMPLE	7:0	SAMPLE[7:0]							
		15:8	SAMPLE[15:8]							
0x16 ... 0x17	Reserved									
0x18	TEMP	7:0	TEMP[7:0]							
0x19 ... 0x1B	Reserved									
0x1C	WINLT	7:0	WINLT[7:0]							
		15:8	WINLT[15:8]							
0x1E	WINHT	7:0	WINHT[7:0]							
		15:8	WINHT[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		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. **Note:** LOWLAT does not keep the clock source enabled when the ADC is not converting, so a clock startup delay may be experienced even though LOWLAT is set. Be sure that the clock source is always enabled to avoid a delay.

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

32.5.2 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

32.5.3 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

Note: The internal references can be used only if lower than V_{DD} - 0.5V.

32.5.4 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[15:0] is used as the Window Comparator source
1	SAMPLE	ADCn.SAMPLE[15:0] 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 threshold value and the 16-bit high threshold value, 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

32.5.5 Interrupt Control

Name:	INTCTRL
Offset:	0x04
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

32.5.6 Interrupt Flags

Name:	INTFLAGS
Offset:	0x05
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
Clear this flag 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
Clear this flag 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
Clear this flag 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
Clear this flag 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
Clear this flag 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
Clear this flag 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.

32.5.7 Status

Name:	STATUS
Offset:	0x06
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.

32.5.8 Debug Control

Name: DBGCTRL
Offset: 0x07
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

32.5.9 Control E

Name:	CTRLE
Offset:	0x08
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.

32.5.10 Control F

Name: CTRLF
Offset: 0x09
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

32.5.11 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 the section *Operation Modes* and the section *Accumulator Test* within.

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.

32.5.12 Positive Input Multiplexer

Name:	MUXPOS
Offset:	0x0C
Reset:	0x00
Property:	-

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

Bits 5:0 – MUXPOS[5:0] 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
`0x00-0x07`	AIN0-AIN7	ADC input pin 0-7
`0x08-0x0F`	-	Reserved
`0x10-0x1B`	AIN16-AIN27	ADC input pin 16-27
`0x1C-0x1E`	-	Reserved
`0x1F`	AIN31	ADC input pin 31
`0x20-0x3F`	-	Reserved
`0x40`	GND	Ground
`0x41`	-	Reserved
`0x42`	TEMPSENSE	Temperature sensor
`0x43`	-	Reserved
`0x44`	VDDDIV10	VDD divided by 10
`0x45-0x48`	-	Reserved
`0x49`	DACREF0	AC0 DAC Voltage Reference
`Other`	-	Reserved

32.5.13 Result

Name:	RESULT
Offset:	0x10
Reset:	0x00
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 ADCn.RESULT register.

Bits 7:0 – RESULT[7:0] Result byte 0
This bit field constitutes the LSB of the ADCn.RESULT register.

32.5.14 Sample

Name:	SAMPLE
Offset:	0x14
Reset:	0x00
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.

32.5.15 Temporary

Name:	TEMP
Offset:	0x18
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 *Memories* section.

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

Bits 7:0 – TEMP[7:0] Temporary
Temporary bit field for read/write operations in 16-bit registers.

32.5.16 Window Comparator Low Threshold

Name:	WINLT
Offset:	0x1C
Reset:	0x00
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.

32.5.17 Window Comparator High Threshold

Name:	WINHT
Offset:	0x1E
Reset:	0x00
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.

33. UPDI - Unified Program and Debug Interface

33.1 Features

- UPDI One-Wire Interface for External Programming and On-Chip-Debugging (OCD)
  - Enable programming by high-voltage or fuse
  - 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

33.2 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.

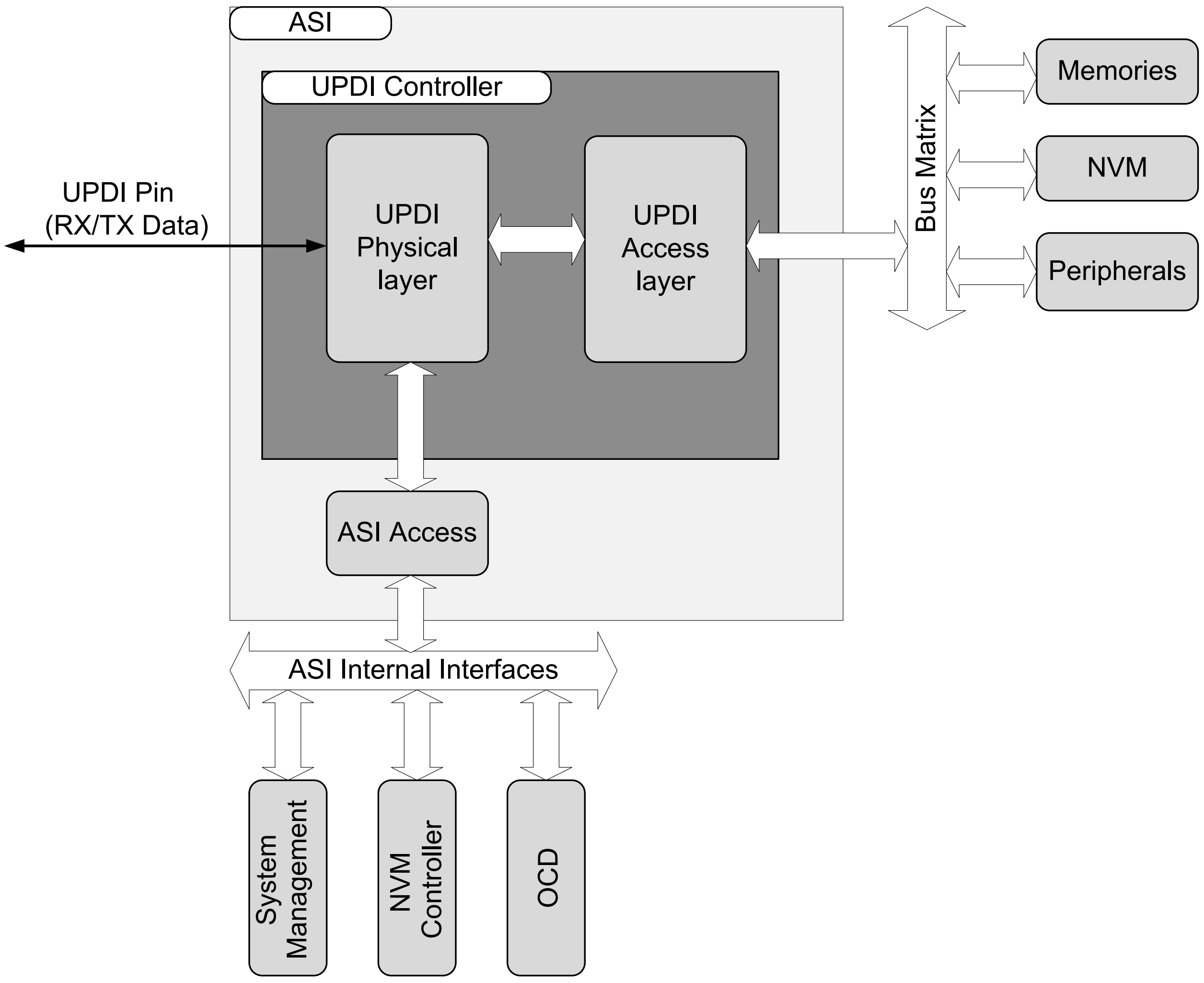
The ACC layer is the interface between the UPDI and the connected bus matrix. This layer grants access via the UPDI interface to the bus matrix with memory-mapped access to system blocks such as memories, NVM, and peripherals.

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.

33.2.1 Block Diagram

Figure 33-1. UPDI Block Diagram

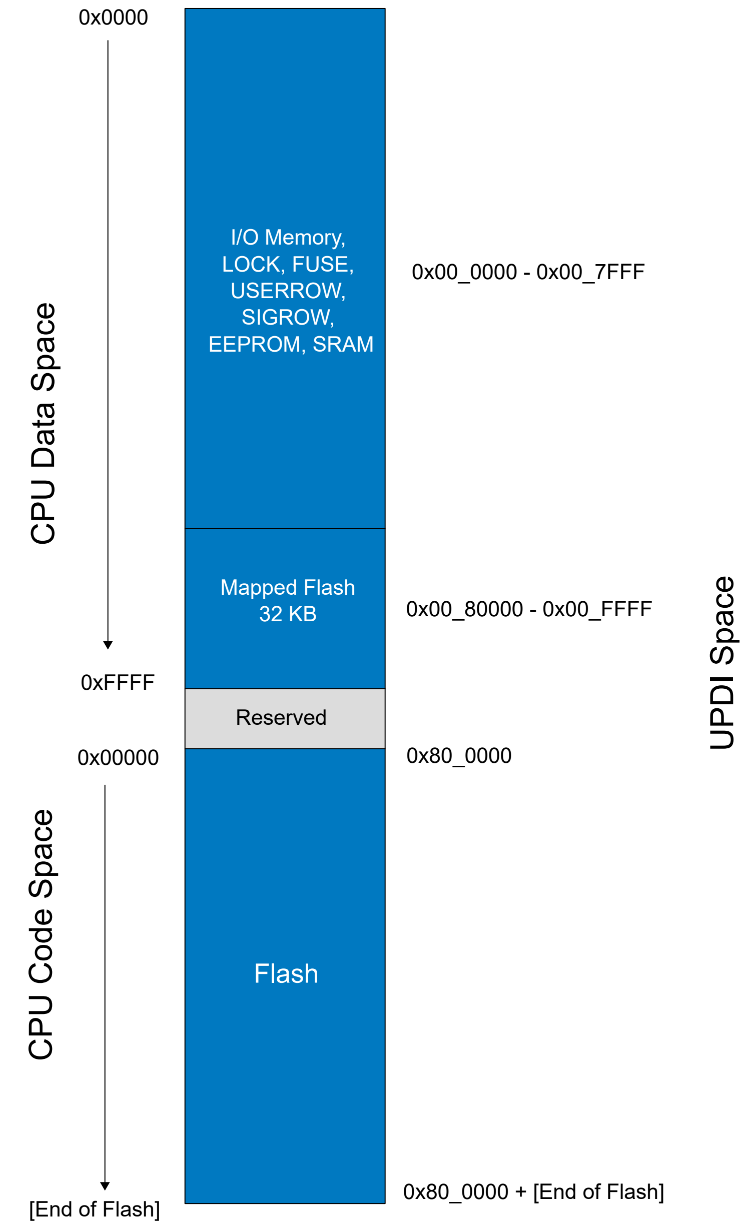


33.2.2 Addressing the Program Memory Space

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 33-2. Memory Map, As Seen From The UPDI



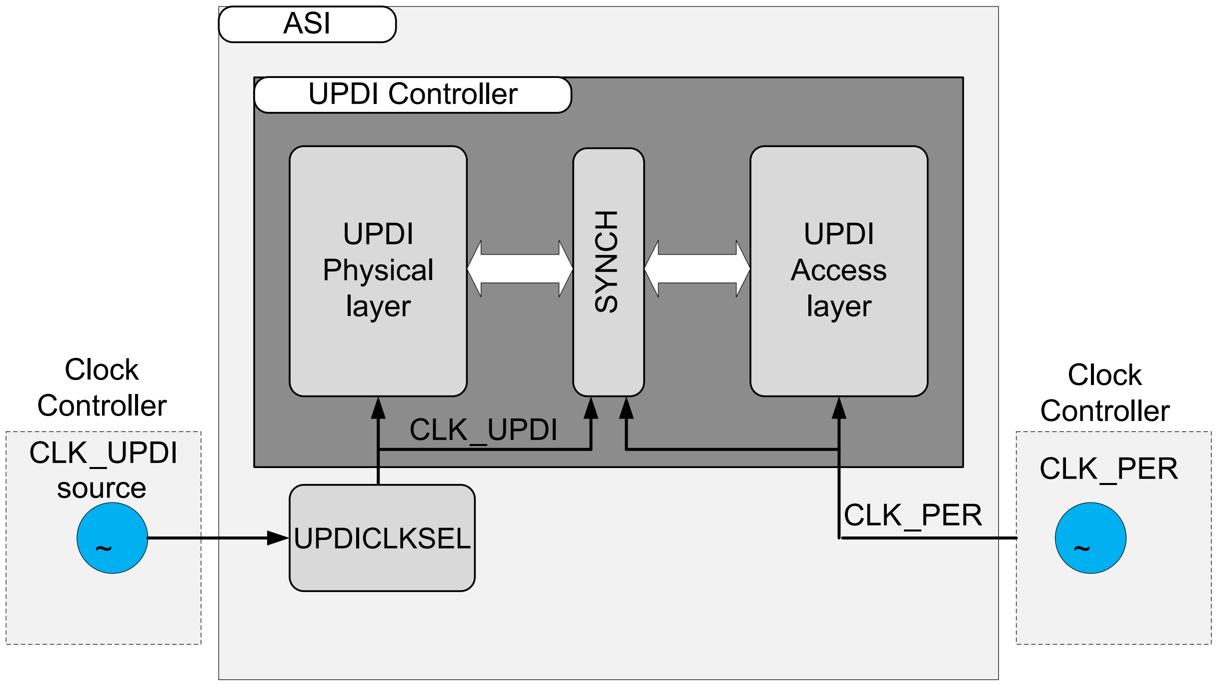
See the *Memories* sections for more details and exact addresses of the memory areas in a given device.

33.2.3 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 33-3. UPDI Clock Domains



33.2.4 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

33.2.5 I/O Lines and Connections

The UPDI uses the UPDI pin for half-duplex UART communication.

The $\overline{\text{RESET}}$ pin can be used to invoke and maintain UPDI operation.

The $\overline{\text{RESET}}$ pin can be used as $\overline{\text{RESET}}$ or input, depending on the configuration of the RSTPINCFG bit in the System Configuration (SYSCFG0) fuse. The UPDI pin can be used as GPIO or for UPDI function, depending on the configuration of the UPDIPINCFG bit in the SYSCFG0 fuse. The high voltage applied to the $\overline{\text{RESET}}$ pin can be used to invoke and maintain UPDI operation, thus overriding both pins configuration in the SYSCFG0 fuse.

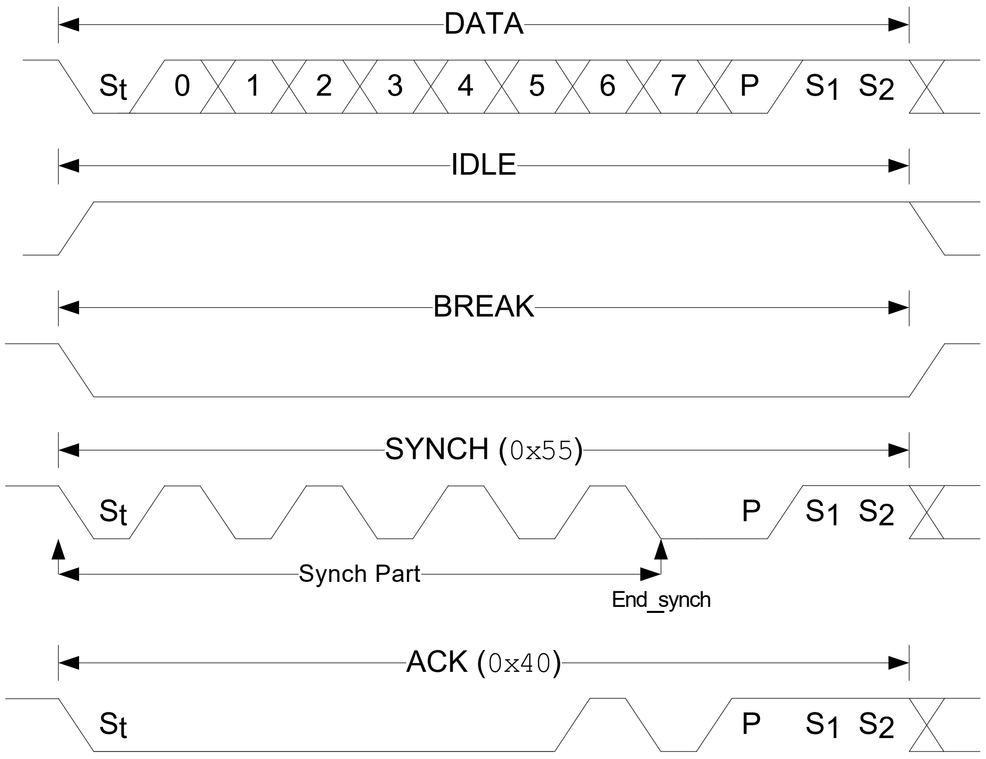
Refer to the 33.3.2.1. UPDI Enabling section for details about the required and optional pin configurations.

33.3 Functional Description

33.3.1 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 33-4. 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.

33.3.1.1 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 33-1. 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 33-2.

Table 33-2. 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

33.3.1.2 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.

33.3.1.2.1 BREAK in One-Wire Mode

In One-Wire mode, the programmer/debugger and UPDI can be totally out of synch, requiring a worst-case length for the BREAK character to be sure that the UPDI can detect it. Assuming the slowest UPDI clock speed of 4 MHz (250 ns), the maximum length of the 8-bit SYNCH pattern value that can be contained in 16 bits is
$65535 \times 250\ \mathrm{ns} = 16.4\ \mathrm{ms/byte} = 16.4\ \mathrm{ms}/8\ \mathrm{bits} = 2.05\ \mathrm{ms/bit}.$

This gives a worst-case BREAK frame duration of $2.05\ \mathrm{ms} \times 12\mathrm{bits}\ \approx 24.6\ \mathrm{ms}$ 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 33-3.

Table 33-3. Recommended BREAK Character Duration

UPDICLKSEL[1:0]	Recommended BREAK Character Duration
`0x1` (16 MHz)	6.15 ms
`0x2` (8 MHz)	12.30 ms
`0x3` (4 MHz)	24.60 ms

33.3.1.3 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.

33.3.1.3.1 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.

33.3.2 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.

33.3.2.1 UPDI Enabling

Depending on how the application has configured the UPDI pin, one of the following two methods can be used to enable the UPDI:

- **One-Wire Enable** - This method requires the UPDI pin to be configured in UPDI function mode by setting the UPDIPINCFG bit in the SYSCFG0 register to '`1`'.
- **HV Override of UPDI Pin** - This method is used if the UPDI pin is configured in GPIO mode (the UPDIPINCFG bit in SYSCFG0 is set to '`0`'). Applying an HV pulse on $\overline{\text{RESET}}$ will reconfigure the UPDI pin to UPDI function mode, thereby overriding the configuration in the SYSCFG0 fuse.

33.3.2.1.1 One-Wire Enable

The UPDI pin has a constant pull-up when enabled, and by driving the UPDI line low for more than 200 ns, a connected programmer/debugger will initiate the start-up sequence. As a prerequisite, the UPDI pin must be configured in UPDI function mode, either by setting the UPDIPINCFG bit in SYSCFG0 fuse to '`1`' or by applying an HV pulse on the $\overline{\text{RESET}}$ pin, thus overriding the configuration of UPDIPINCFG.

Follow this sequence to enable the UPDI:

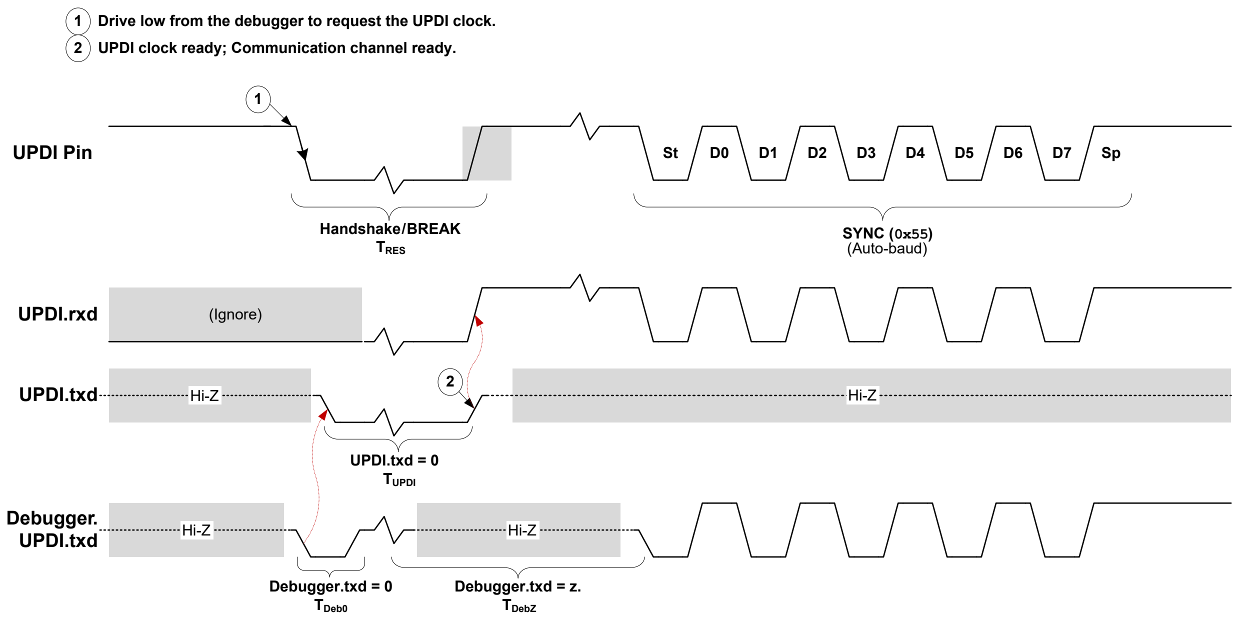
1. Drive the UPDI pin low for more than 200 ns, and release it.
   The UPDI pin has an internal pull-up resistor, and by driving the UPDI pin low for more than 200 ns, a connected programmer will initiate the start-up sequence:
   - An edge detector starts driving the UPDI pin low, so when the programmer 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 33-5, 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 33-5. 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 33.3.2.2.1. 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.

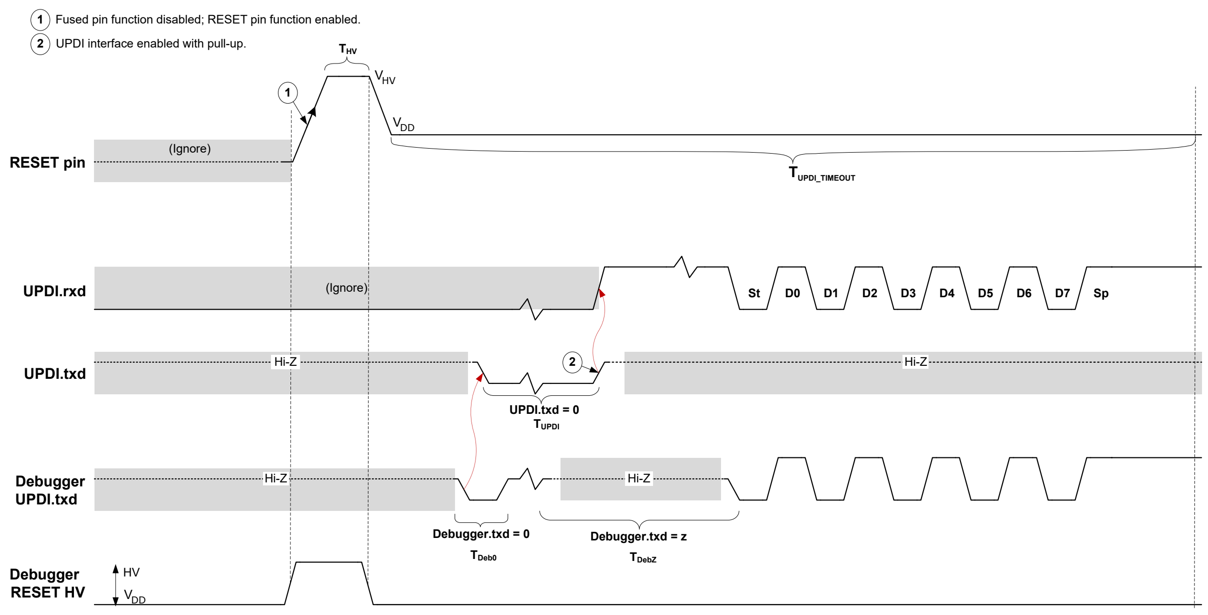
33.3.2.1.2 UPDI Enable with High-Voltage Override of UPDI Pin

An application can configure the UPDI pin as an I/O pin. In this case, the regular one-wire enable sequence cannot be used.

An HV pulse applied to the $\overline{\text{RESET}}$ pin will switch the pin functionality of the UPDI pin to the UPDI function.

1. Apply the HV signal, as described in Figure 33-6 and the *Electrical Characteristics* section. The HV must be applied after the POR has been released.
   This will override the pin configuration of the UPDI pin. The HV detection circuitry will trigger a device Reset. The CPU will remain halted until the reception of a valid UPDI key, or the expiration of $T_{UPDI\_TIMEOUT}$. If no such key is received, the device will be reset and the UPDI pin will have the configuration specified by the fuses.
2. Follow the regular one-wire enable sequence as described in 33.3.2.1.1. One-Wire Enable. A valid UPDI key must be sent before $T_{UPDI\_TIMEOUT}$.
3. When the UPDI is enabled by an HV pulse, only a POR will disable the override of the UPDI pin and restore the settings as configured by the fuses.

Figure 33-6. UPDI Enable Sequence by High-Voltage (HV) Programming



Notes:
1. If insufficient external protection is added to the UPDI pin, an ESD pulse can be interpreted by the device as a high-voltage override and enable the UPDI.
2. The actual threshold voltage for the UPDI HV activation depends on V_{DD}. See the *Electrical Characteristics* section for more details.
3. See the *Electrical Characteristics* section for the value of $T_{UPDI\_TIMEOUT}$.

33.3.2.2 UPDI Disabling

33.3.2.2.1 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.

33.3.2.2.2 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.

33.3.2.3 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.

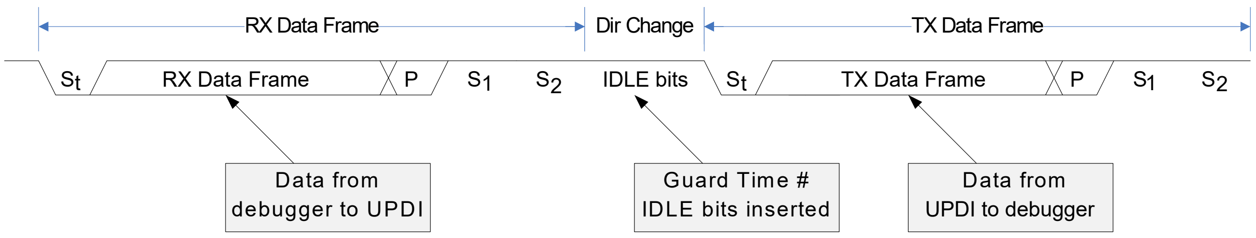
Whenever the UPDI detects an error, it will immediately enter an internal Error state to avoid unwanted system communication. In the Error state, the UPDI will ignore all incoming data requests, except when a BREAK character is received. The following procedure must always be applied when recovering 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.

33.3.2.4 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 33-7. UPDI Direction Change by Inserting Idle Bits



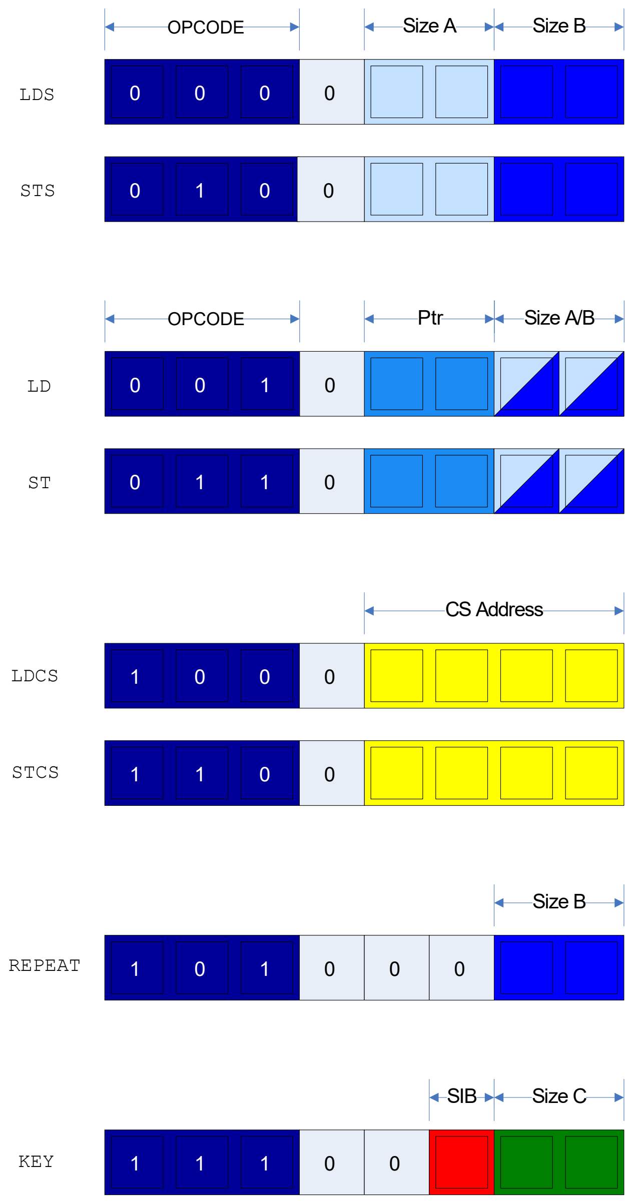
The UPDI guard time is the minimum Idle time that the connected debugger will experience when waiting for data from the UPDI. The maximum Idle time is the same as time-out. When the synchronization time plus the data bus accessing time is longer than the guard time, the Idle time before a transmission 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.

33.3.3 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 33-8 gives an overview of the UPDI instruction set.

Figure 33-8. 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

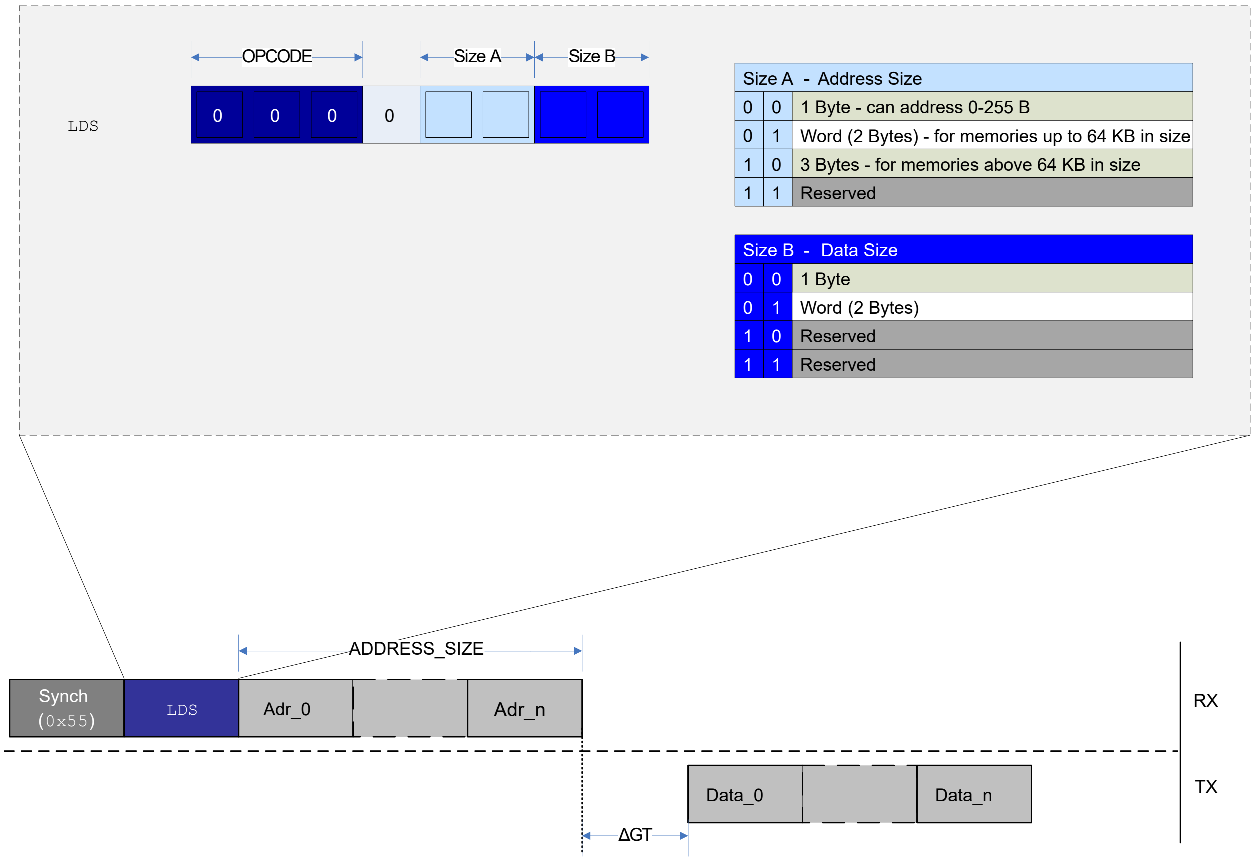
SIB - System Information Block Sel.	
0	Receive KEY
1	Send SIB

33.3.3.1 `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 33-9. `LDS` Instruction Operation



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

Size B		Data Size
0	0	1 Byte
0	1	Word (2 Bytes)
1	0	Reserved
1	1	Reserved

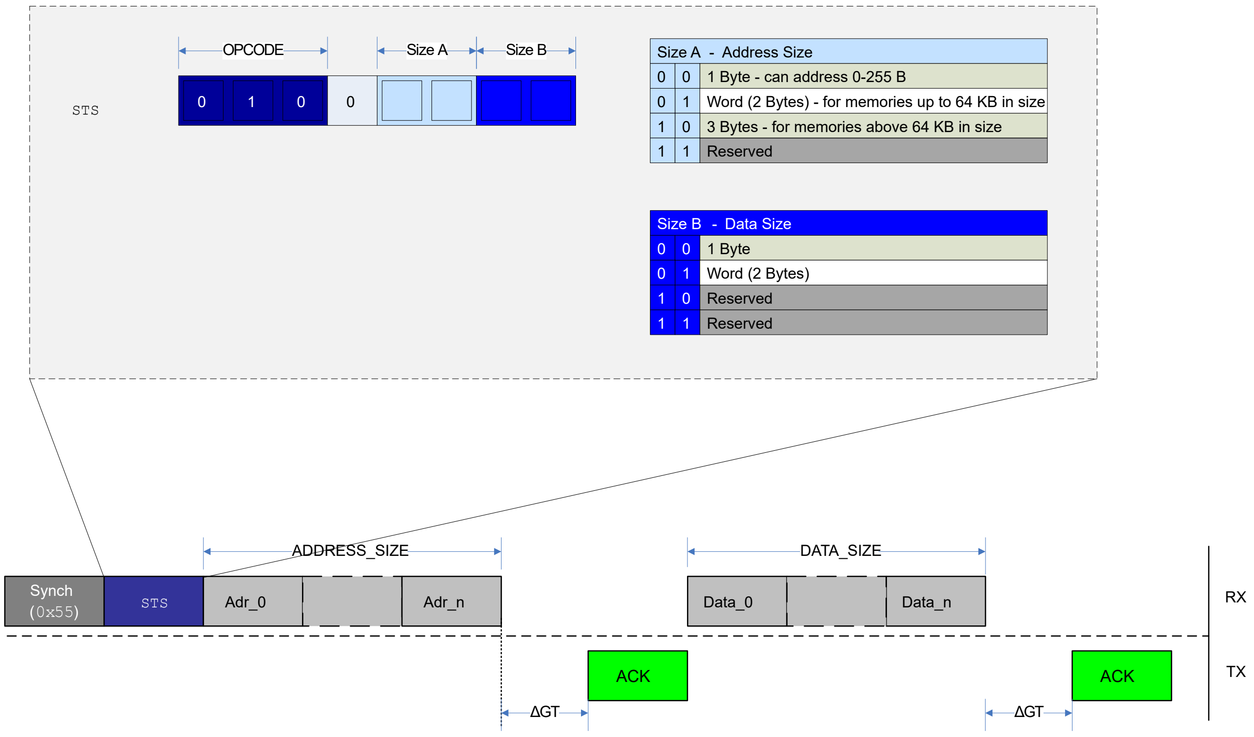
When the instruction is decoded and the address byte(s) are received as dictated by the decoded instruction, the DL layer will synchronize all required information to the ACC layer. This will handle the bus request and synchronize data buffered from the bus back to the DL layer, which will create a synchronization delay that must be taken into consideration upon receiving the data from the UPDI.

33.3.3.2 `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 33-10. The maximum size for both address and data is 32 bits.

The `STS` supports repeated memory access when combined with the `REPEAT` instruction.

Figure 33-10. `STS` Instruction Operation



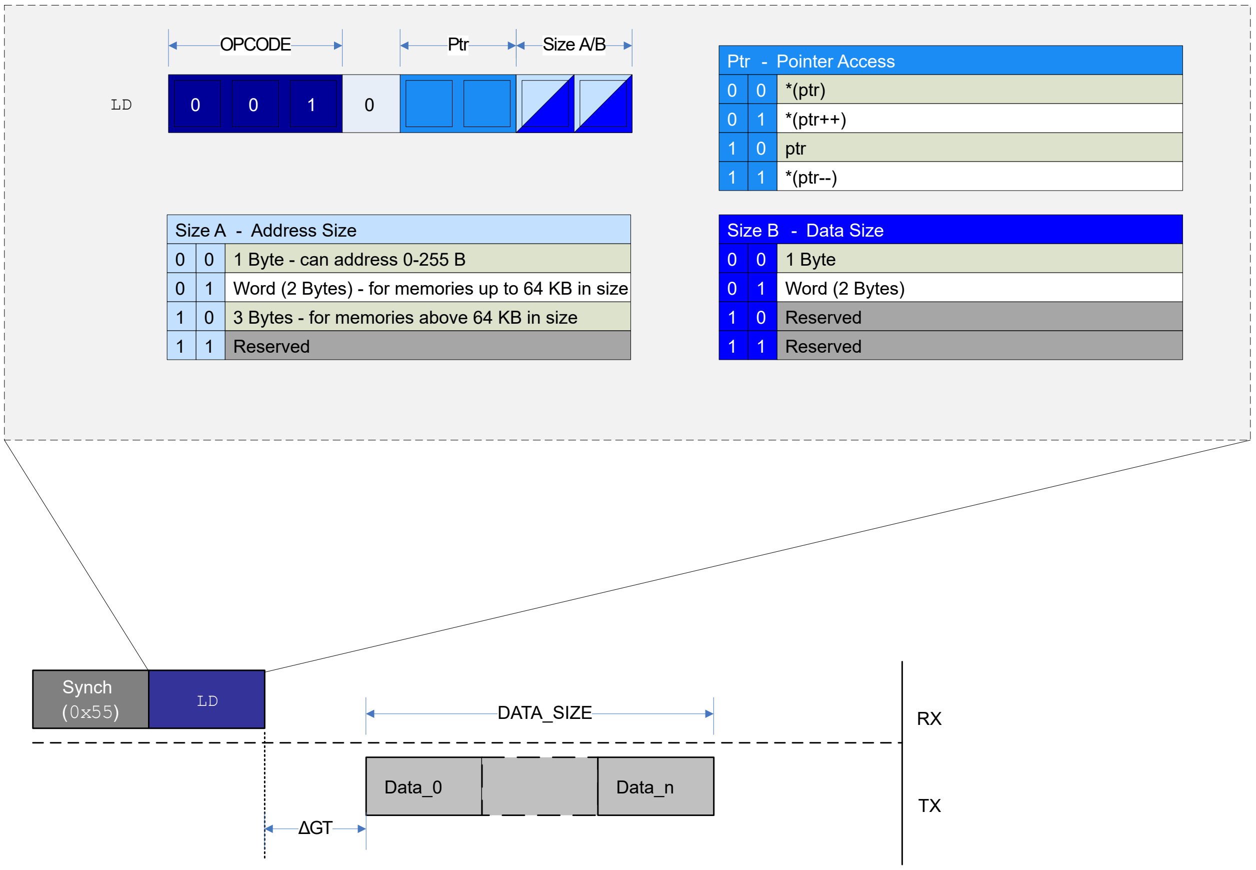
The transfer protocol for an `STS` instruction is depicted in Figure 33-10, following this sequence:

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.

33.3.3.3 `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 33-11. `LD` Instruction Operation



Ptr - Pointer Access		
0	0	*(ptr)
0	1	*(ptr++)
1	0	ptr
1	1	*(ptr--)

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

Size B - Data Size		
0	0	1 Byte
0	1	Word (2 Bytes)
1	0	Reserved
1	1	Reserved

Figure 33-11 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.

33.3.3.4 `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 33-12. `ST` Instruction Operation

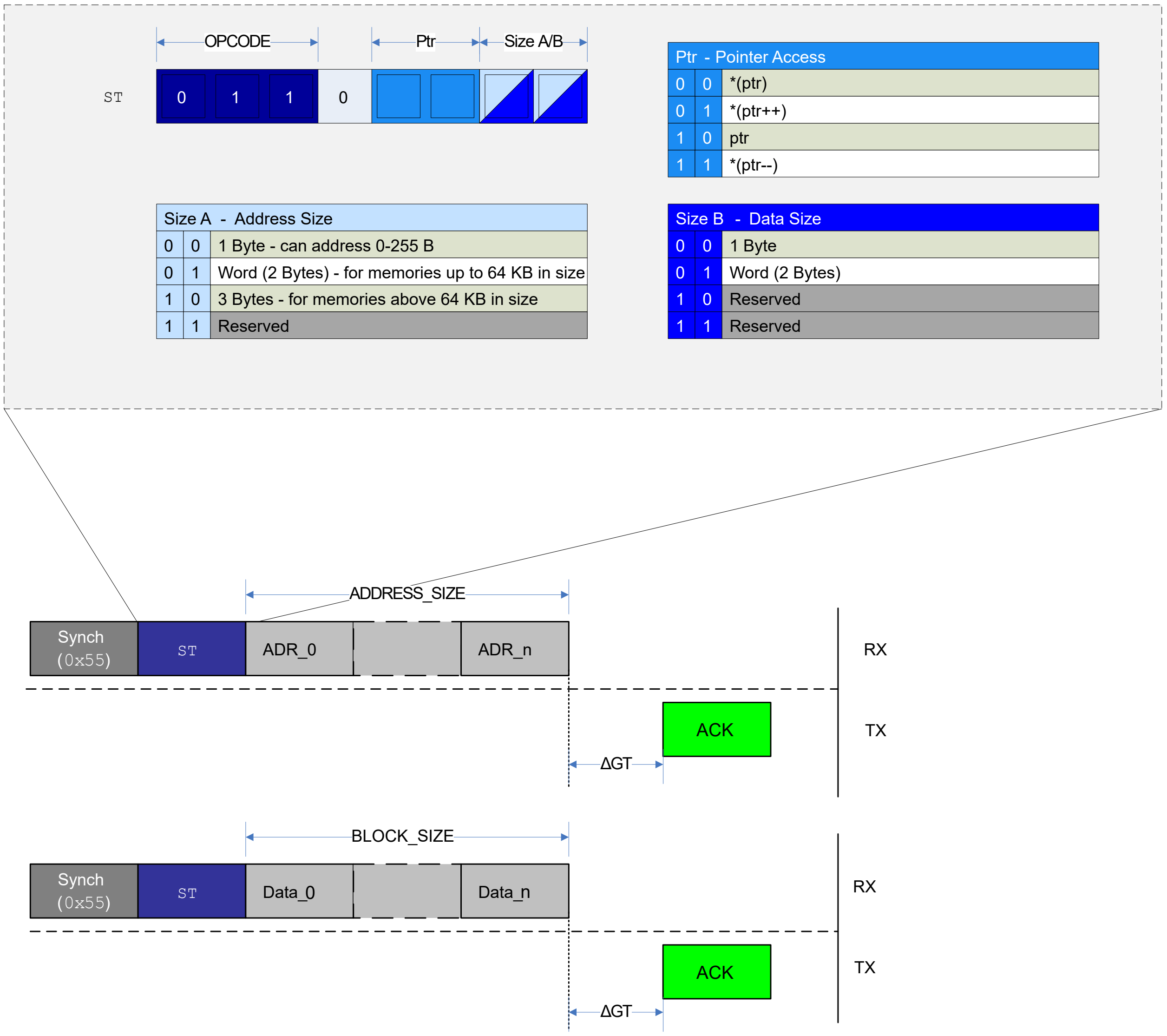


Figure 33-12 gives an example of an `ST` instruction to the UPDI Pointer register and the storage of regular data. A SYNCH character is sent before each instruction. In both cases, an Acknowledge (ACK) is sent back by the UPDI if the `ST` instruction was successful.

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.

33.3.3.5 `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 33-13. `LDCS` Instruction Operation

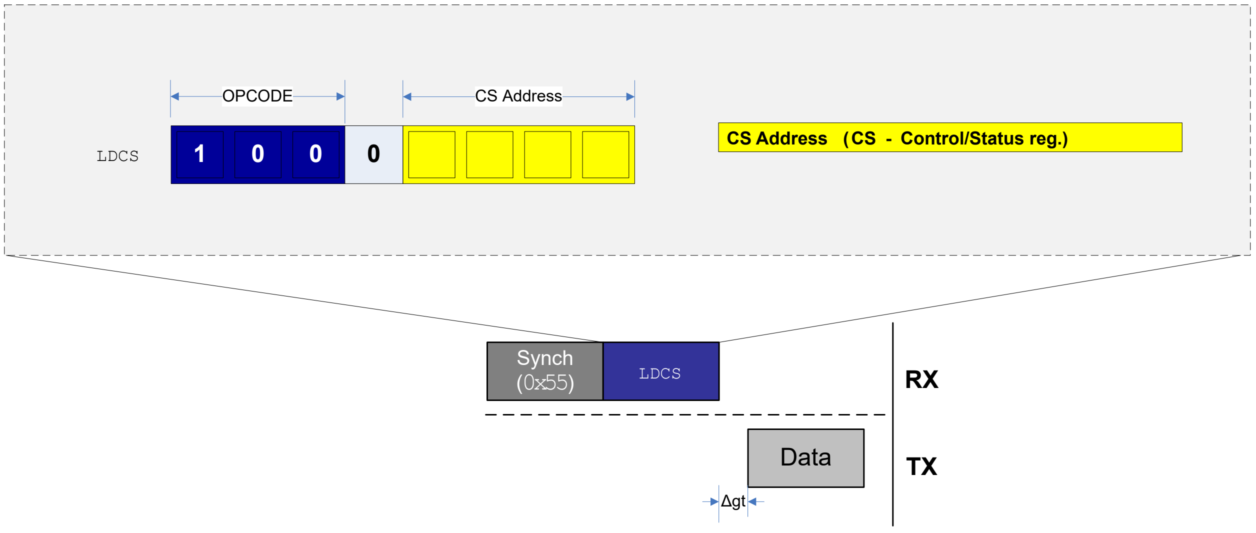


Figure 33-13 shows a typical example of `LDCS` data transmission. A data byte from the `LDCS` is transmitted from the UPDI after the guard time is completed.

33.3.3.6 `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 33-14. STCS Instruction Operation

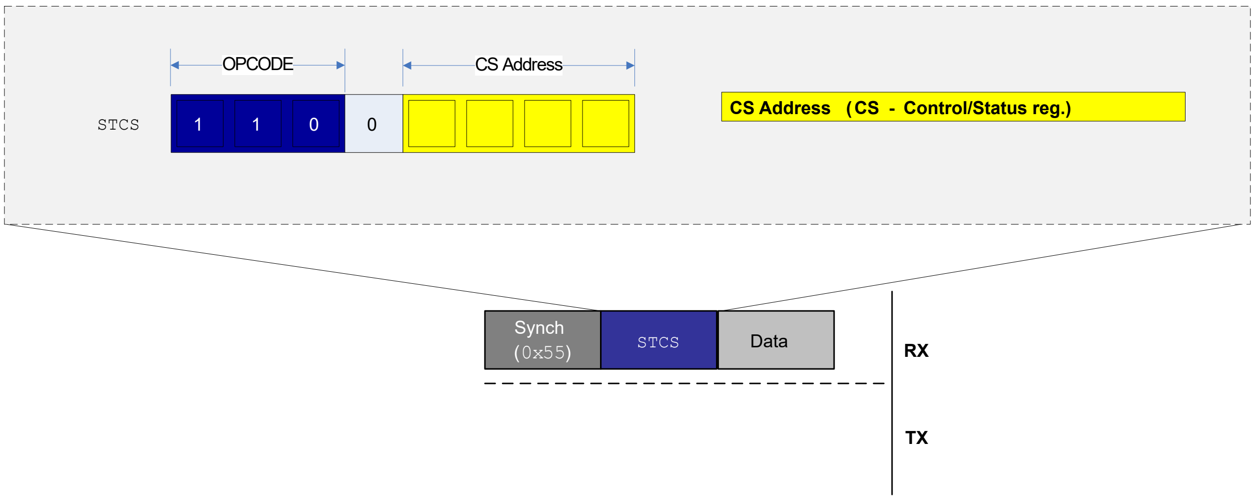


Figure 33-14 shows the data frame transmitted after the SYNCH character and the instruction frames. The STCS instruction byte can immediately be followed by the data byte. There is no response generated from the STCS instruction, as is the case for the ST and STS instructions.

33.3.3.7 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 33-15. `REPEAT` Instruction Operation Used with `ST` Instruction

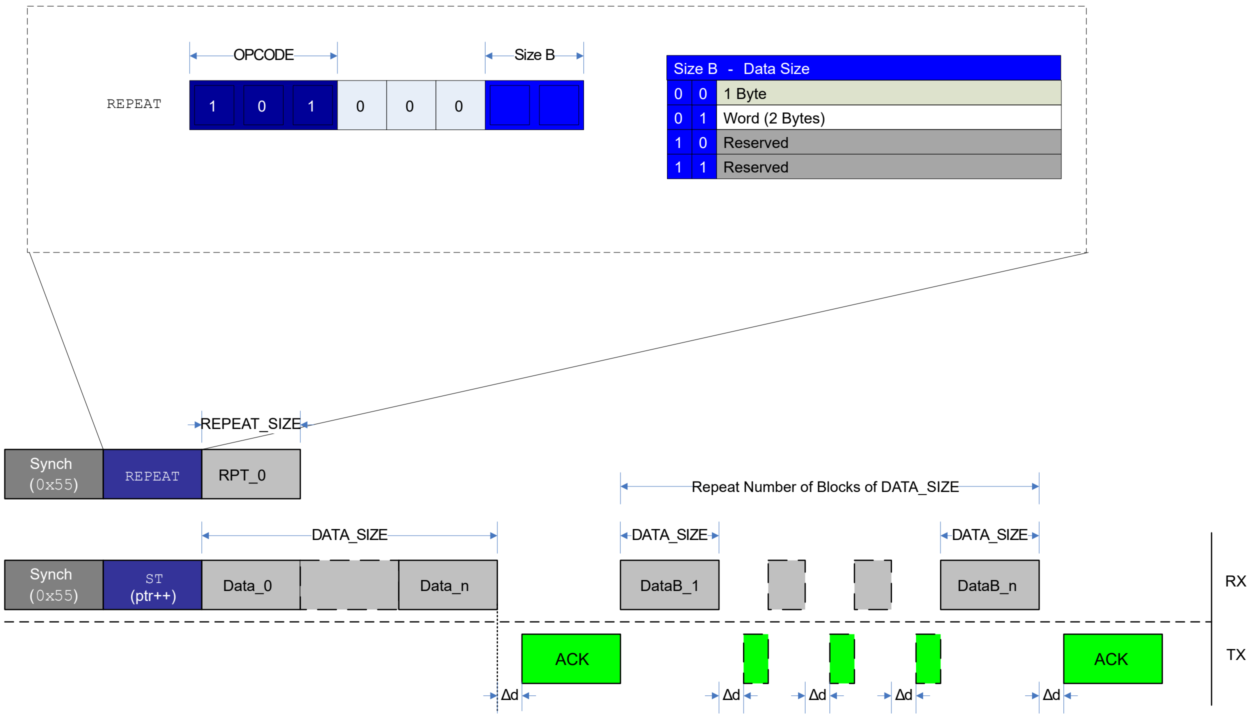
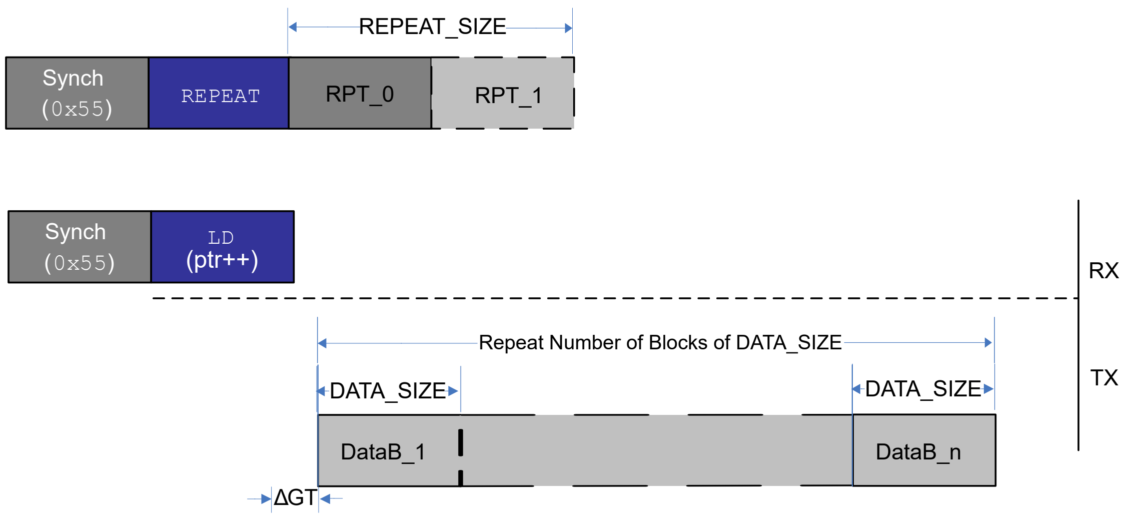


Figure 33-15 gives an example of a repeat operation with an `ST` instruction using pointer post-increment operation. After the `REPEAT` instruction is sent with `RPT_0` = *n*, the first `ST` instruction is issued with SYNCH and instruction frame. The next *n* `ST` instructions are executed by only sending data bytes according to the `ST` operand `DATA_SIZE` and maintaining the Acknowledge (ACK) handshake protocol.

Figure 33-16. `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`).

33.3.3.8 `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 33-17. `KEY` Instruction Operation

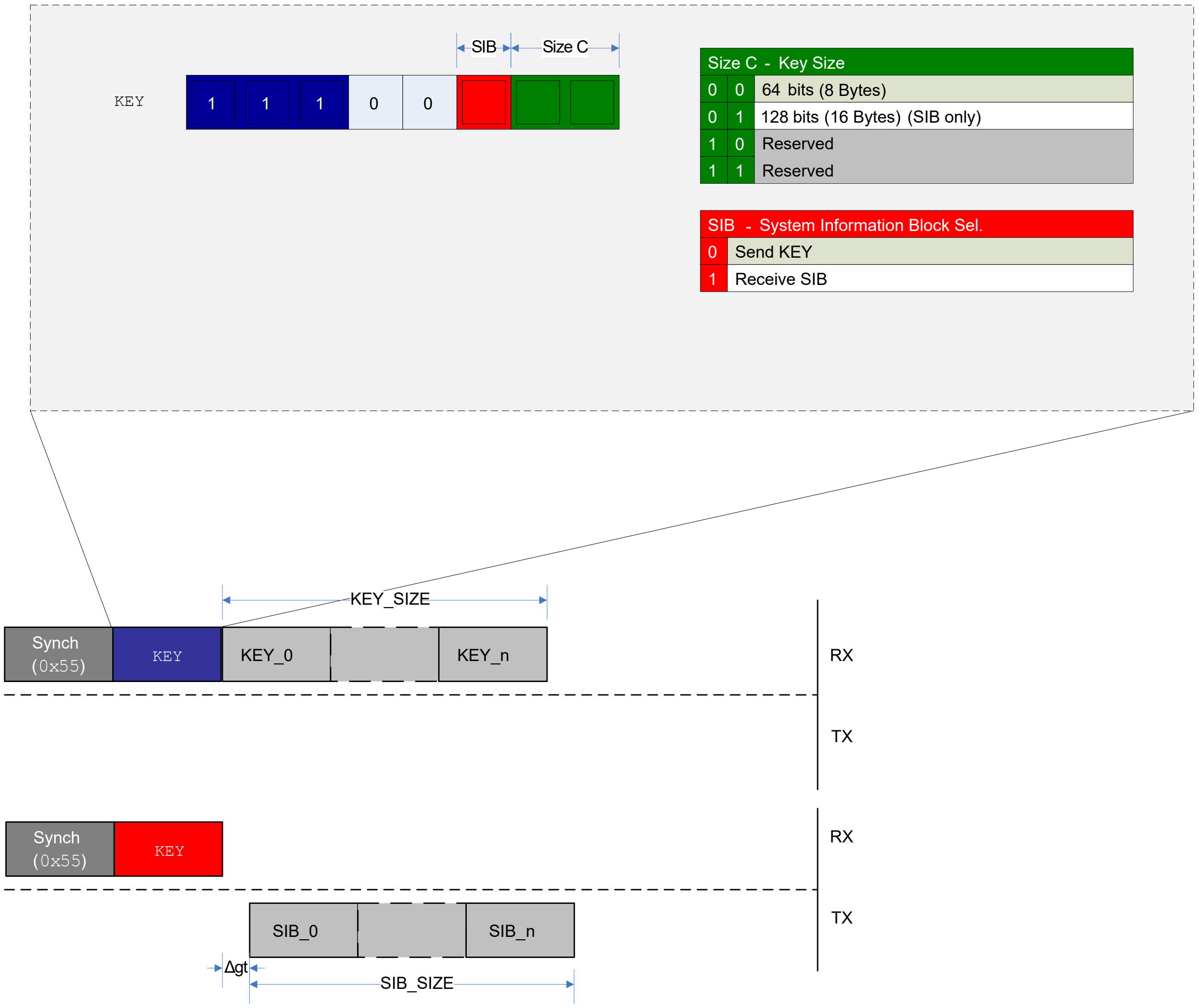


Figure 33-17 shows the transmission of a key and the reception of a SIB. In both cases, the Size C (`SIZE_C`) field in the operand determines the number of frames being sent or received. There is no response after sending a `KEY` to the UPDI. When requesting the SIB, data will be transmitted from the UPDI according to the current guard time setting.

33.3.4 CRC Checking of Flash During Boot

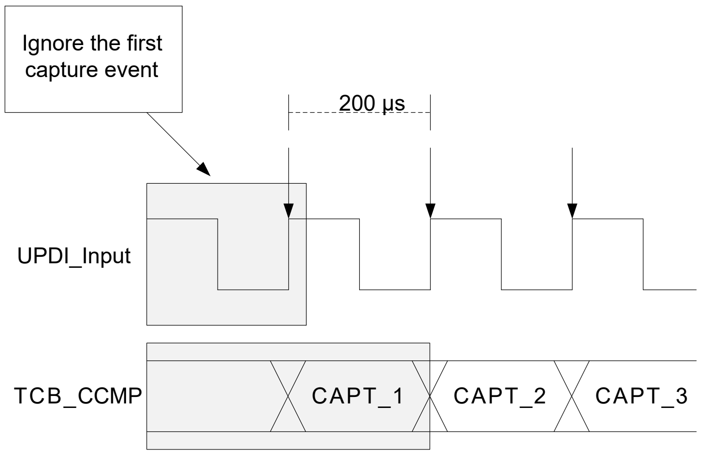
Some devices support running a CRC check of the Flash contents as part of the boot process. This check can be performed even when the device is locked. The result of this CRC check can be read from the ASI_CRC_STATUS register. Refer to the *CRCSCAN - Cyclic Redundancy Check Memory Scan* section in the device data sheet for more information on this feature.

33.3.5 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 33-18 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 33-18. UPDI System Clock Measurement Events



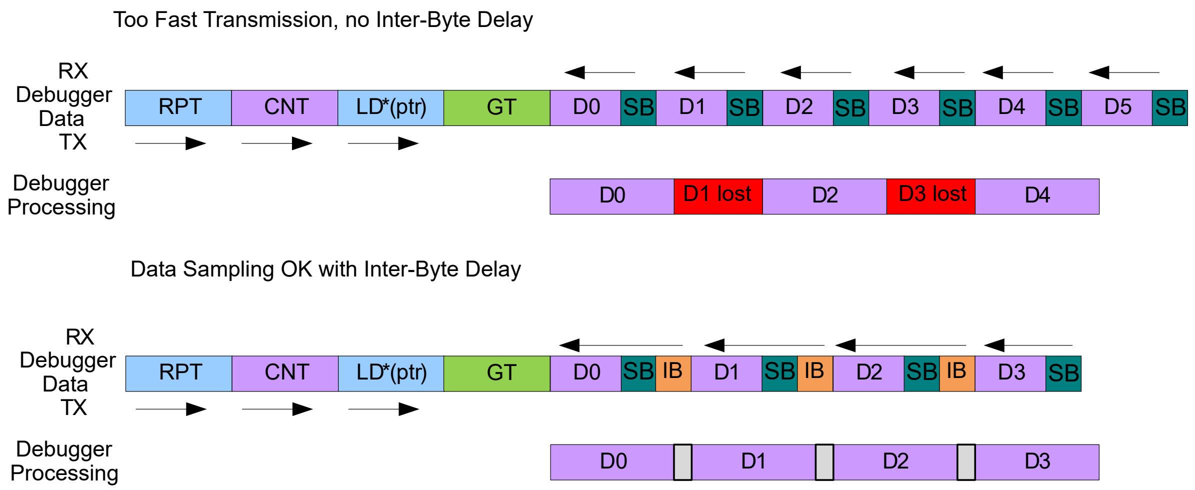
33.3.6 Inter-Byte Delay

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 33-19. Inter-Byte Delay Example with `LD` and `RPT`

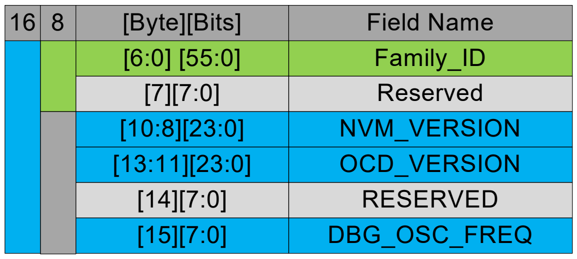


Notes:
1. GT denotes the guard time insertion.
2. SB is for Stop bit.
3. IB is the inserted inter-byte delay.
4. The rest of the frames are data and instructions.

33.3.7 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 33-20 for SIB format description and which data are available at different readout sizes.

Figure 33-20. 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

33.3.8 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 33.3.3.8. KEY - Set Activation Key or Send System Information Block. Table 33-4 describes the available keys and the condition required for starting the operation after the key has been loaded.

Table 33-4. 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 33-5 gives an overview of the available key signatures that must be shifted in to activate the interfaces.

Table 33-5. 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

33.3.8.1 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.

33.3.8.2 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 33.3.8.1. Chip Erase. If the part is already unlocked, this point can be skipped.
2. Enter the NVMPROG key by using the `KEY` instruction. See Table 33-5 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.

33.3.8.3 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 33-5 by using the `KEY` instruction. See Table 33-5 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.

33.3.9 Events

The UPDI can generate the following events:

Table 33-6. 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.

33.3.10 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.

33.4 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]		

33.5 Register Description

These registers are readable only through the UPDI with special instructions and are not readable through the CPU.

33.5.1 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.

33.5.2 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 33-7. 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

33.5.3 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

33.5.4 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.

33.5.5 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 33.3.8.1. Chip Erase section.

33.5.6 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

33.5.7 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)

33.5.8 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 in any mode (Fuse, HV).

33.5.9 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'.

33.5.10 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

34. 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.

35. Ordering Information

- Available ordering options can be found by:
  - Clicking on one of the following product page links:
    - AVR64DU32 Product Page
    - AVR64DU28 Product Page
  - Searching by product name at microchipdirect.com
  - Contacting the local sales representative

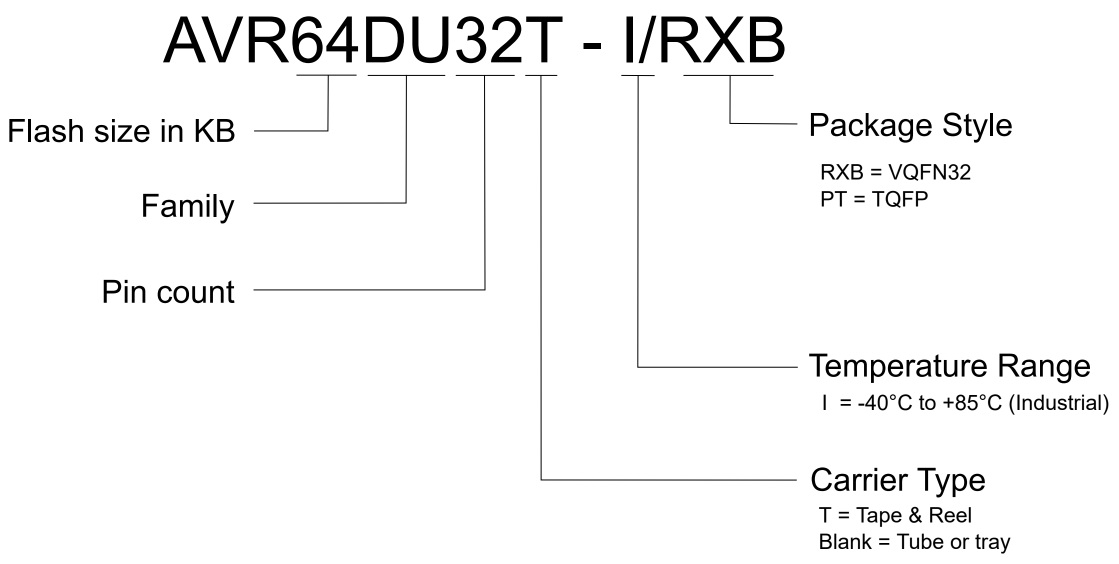
Note: Automotive-grade ordering codes (VAO suffix) are set up on request and not listed in the table below. Contact your local Microchip sales representative to request VAO ordering codes not present on the respective product page.

Table 35-1. Available Product Numbers

CPN	Flash/SRAM	Pin Count	Package Type	Temperature Range	Carrier Type
AVR64DU28-I/SP	64 KB/8 KB	28	SPDIP	-40°C to +85°C	Tube
AVR64DU28-I/SS	64 KB/8 KB	28	SSOP	-40°C to +85°C	Tube
AVR64DU28-I/STX	64 KB/8 KB	28	VQFN	-40°C to +85°C	Tube
AVR64DU28T-I/SS	64 KB/8 KB	28	SSOP	-40°C to +85°C	Tape & Reel
AVR64DU28T-I/STX	64 KB/8 KB	28	VQFN	-40°C to +85°C	Tape & Reel
AVR64DU32-I/PT	64 KB/8 KB	32	TQFP	-40°C to +85°C	Tray
AVR64DU32-I/RXB	64 KB/8 KB	32	VQFN	-40°C to +85°C	Tray
AVR64DU32T-I/PT	64 KB/8 KB	32	TQFP	-40°C to +85°C	Tape & Reel
AVR64DU32T-I/RXB	64 KB/8 KB	32	VQFN	-40°C to +85°C	Tape & Reel

Figure 35-1. 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 with the Tape & Reel option.

36. Package Drawings

36.1 Online Package Drawings

For the most recent package drawings:

1. Go to www.microchip.com/packaging.
2. Go to the package type-specific page, for example, VQFN.
3. Search for Drawing Number and Style to find the most recent package drawings.

Table 36-1. Drawing Numbers

Pin Count	Package Type	Drawing Number	Style
28	SPDIP	C04-00070	SP
28	SSOP	C04-00073	SS
28	VQFN	C04-00456	STX
32	TQFP	C04-00074	PT
32	VQFN	C04-21395	RXB

36.2 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.

36.2.1 28-Pin SPDIP

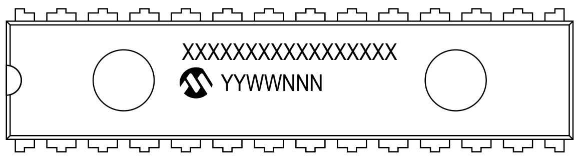


Figure 36-1. General

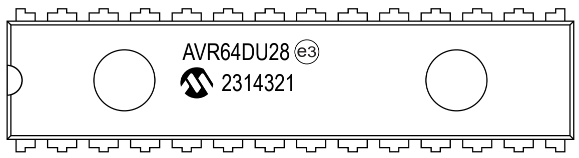


Figure 36-2. Example

36.2.2 28-Pin SSOP

Figure 36-3. General

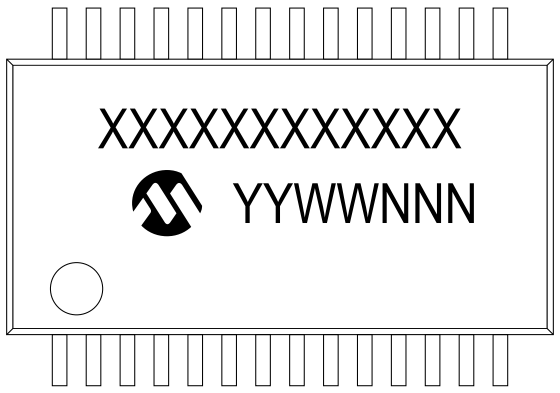
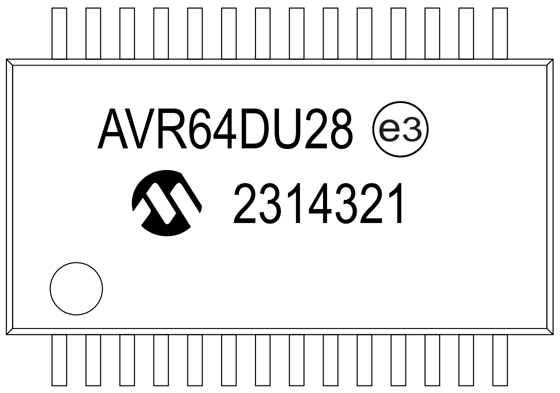


Figure 36-4. Example



36.2.3 28-Pin VQFN

Figure 36-5. General

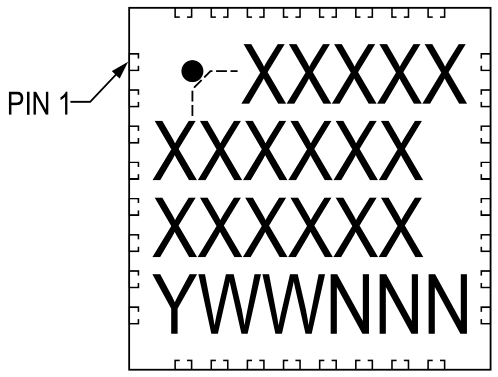
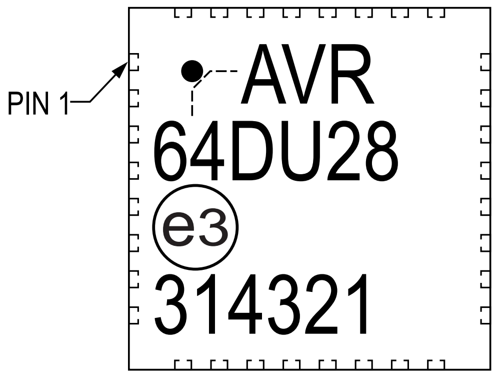


Figure 36-6. Example



36.2.4 32-Pin TQFP

Figure 36-7. General

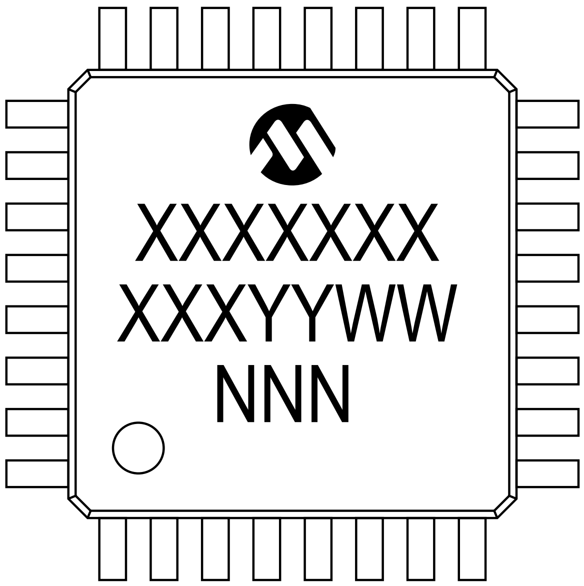
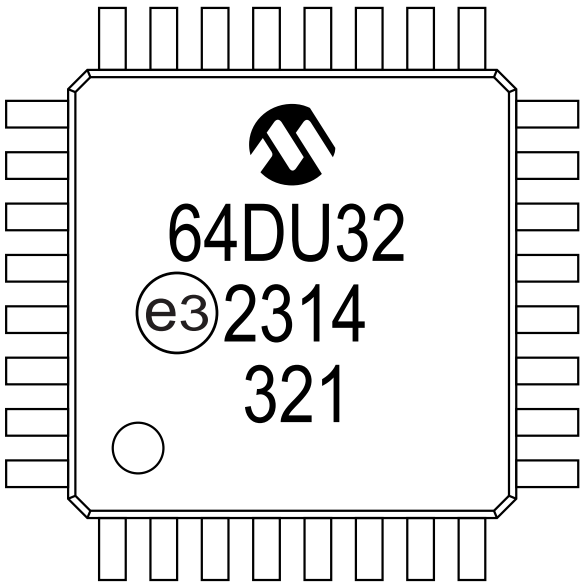


Figure 36-8. Example



36.2.5 32-Pin VQFN

Figure 36-9. General

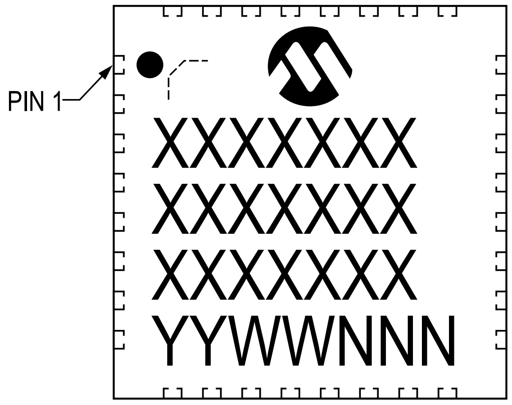
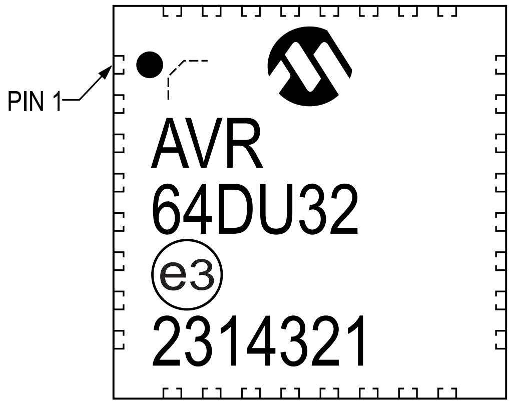


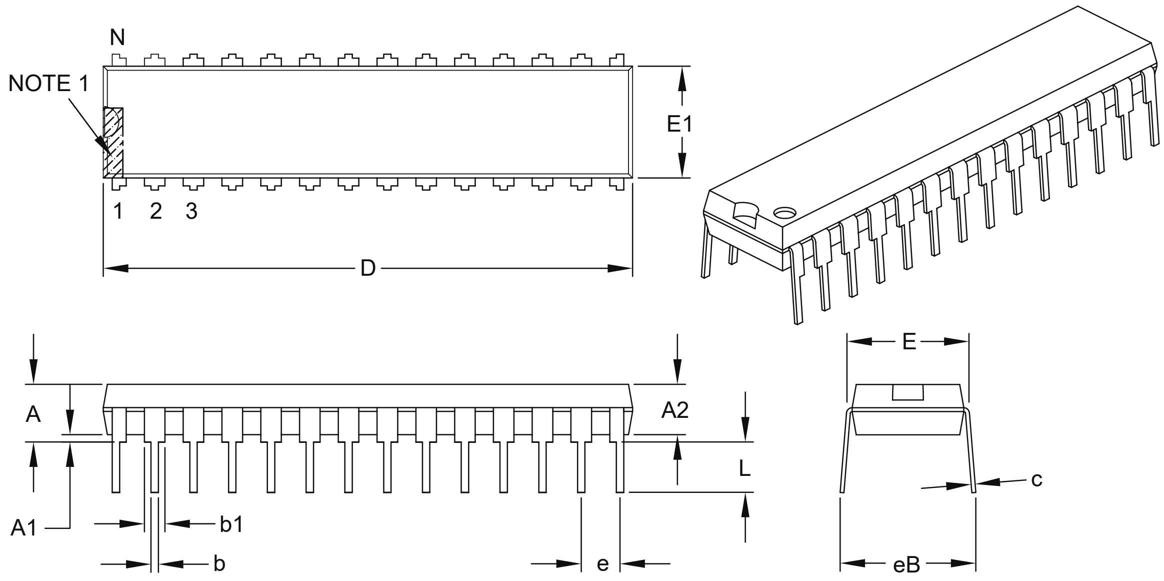
Figure 36-10. Example



36.3 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.

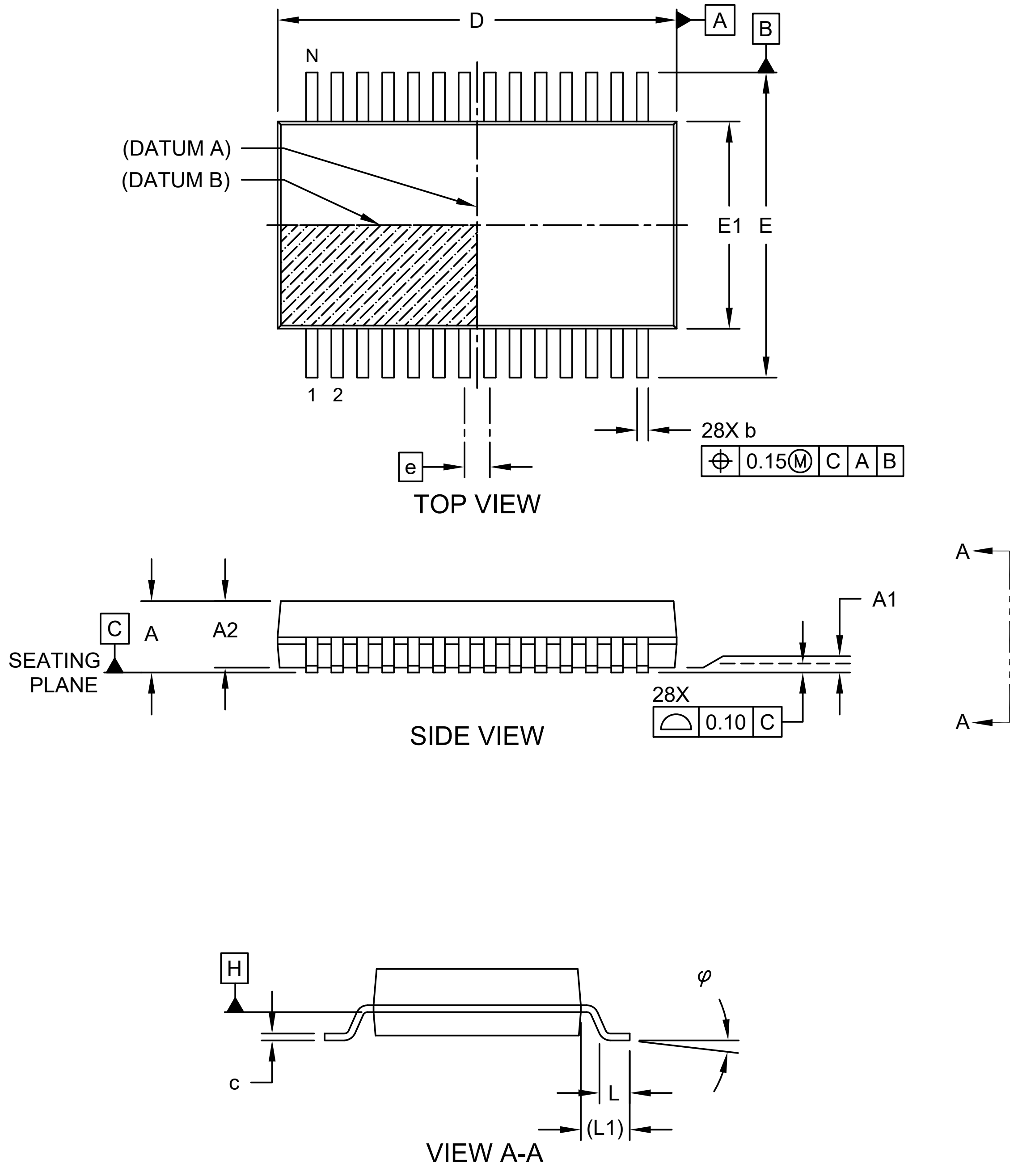
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

36.4 28-Pin SSOP

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

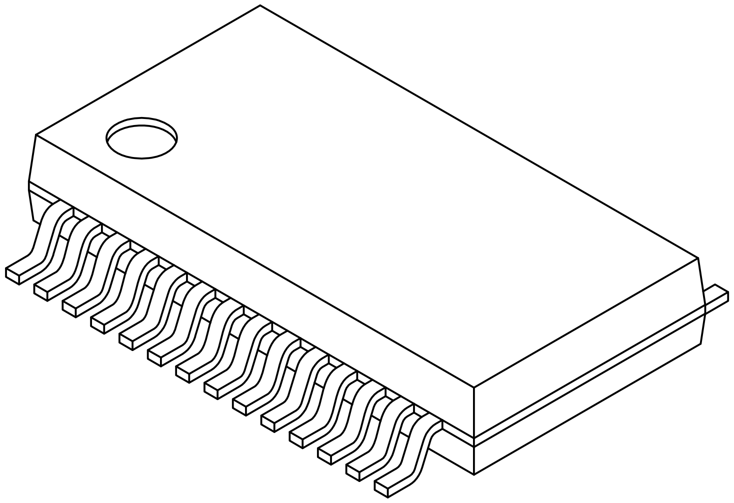
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



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

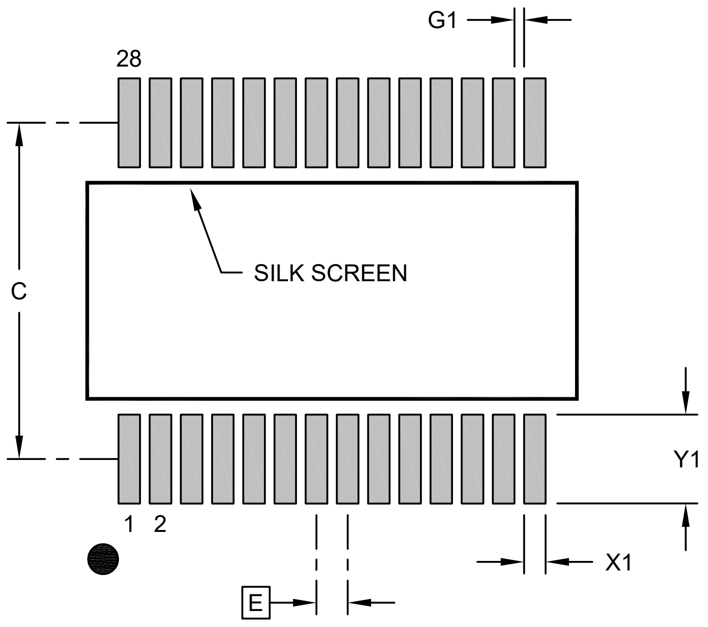
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   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:

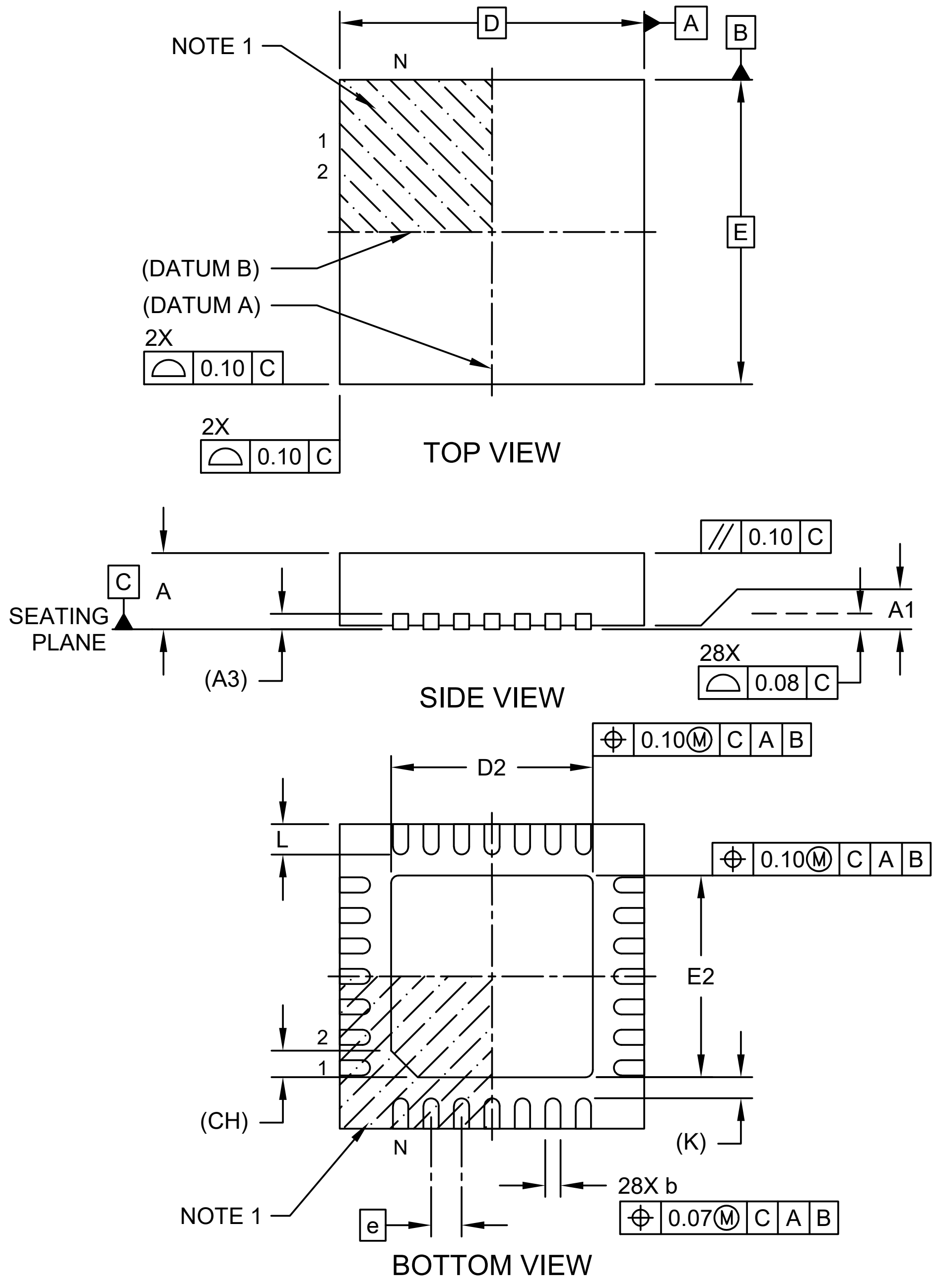
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2073 Rev B

36.5 28-Pin VQFN

**28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4x1.0 mm Body [VQFN]
With 2.65x2.65 mm Exposed Pad**

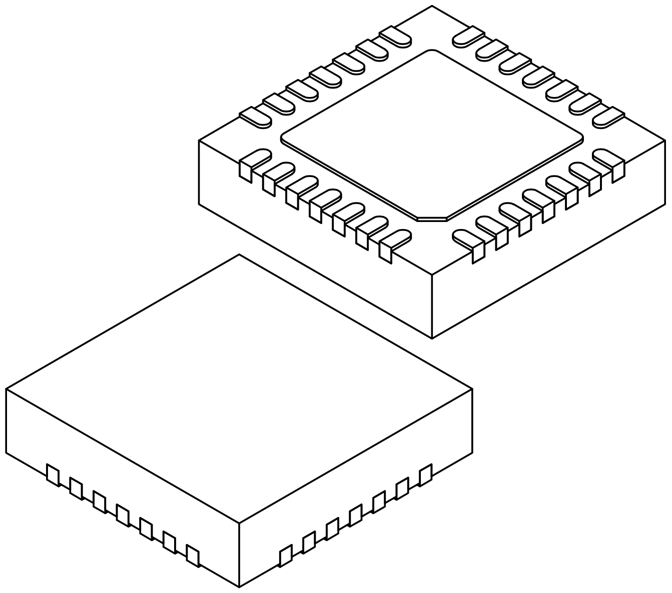
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-456 Rev C Sheet 1 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4x1.0 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



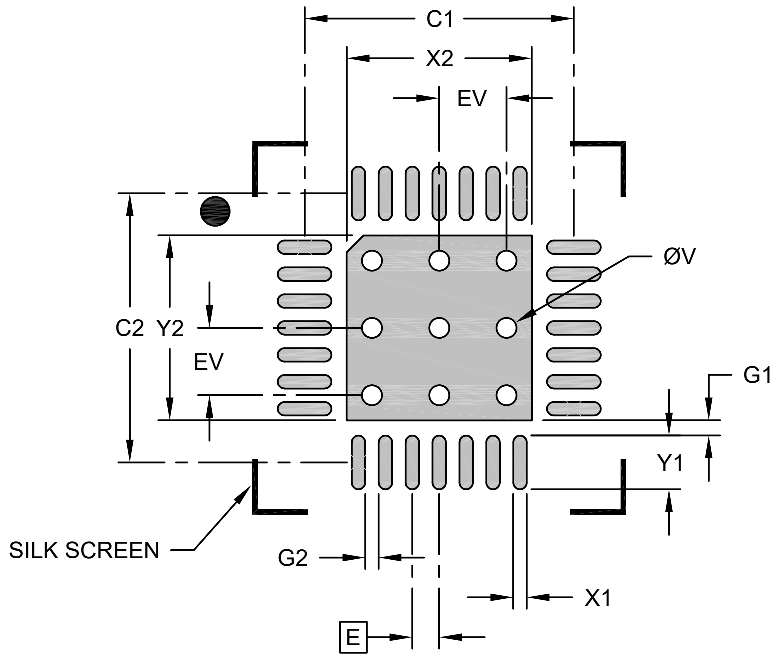
	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Number of Terminals	N		28	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		4.00 BSC	
Exposed Pad Length	D2	2.55	2.65	2.75
Overall Width	E		4.00 BSC	
Exposed Pad Width	E2	2.55	2.65	2.75
Exposed Pad Corner Chamfer	CH		0.35 REF	
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		0.275 REF	

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-456 Rev C Sheet 2 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4x1.0 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.40 BSC		
Optional Center Pad Width	X2			2.75
Optional 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:

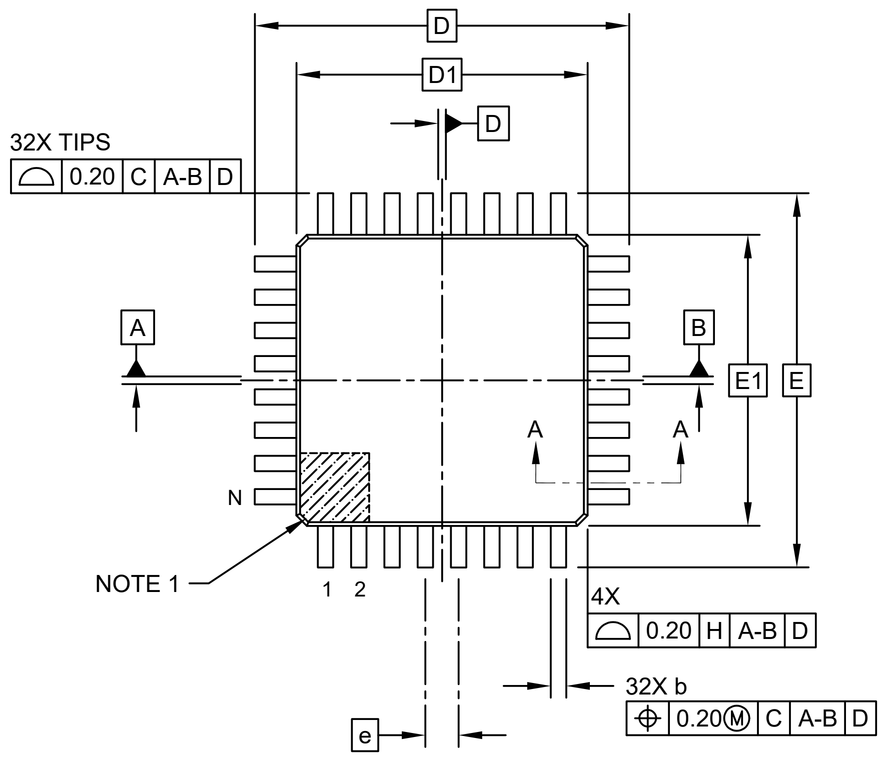
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2456 Rev C

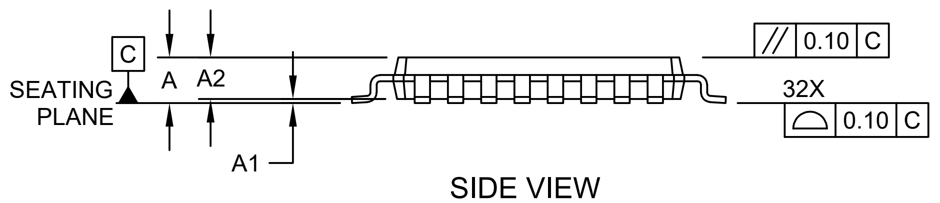
36.6 32-Pin TQFP

32-Lead Plastic Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP]
2.00 mm Footprint; Also Atmel Legacy Global Package Code AUT

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW

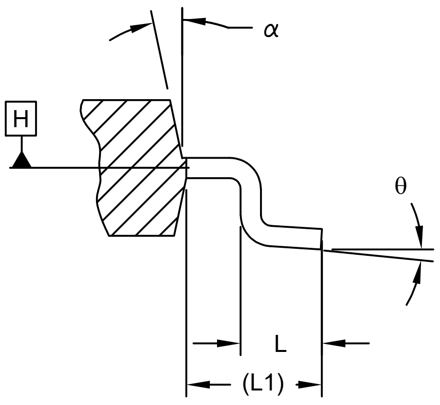


SIDE VIEW

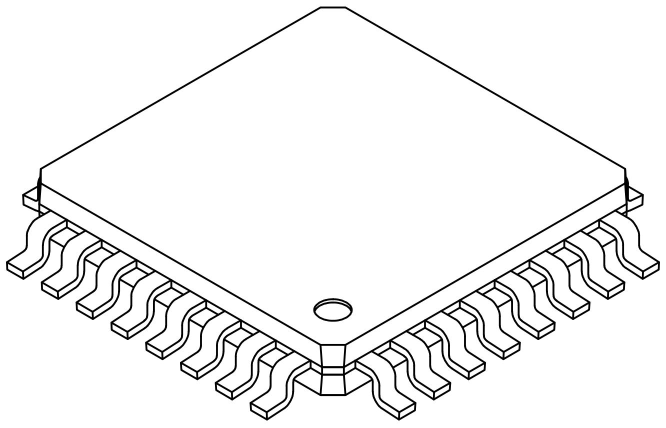
Microchip Technology Drawing C04-074-PT Rev D Sheet 1 of 2

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°

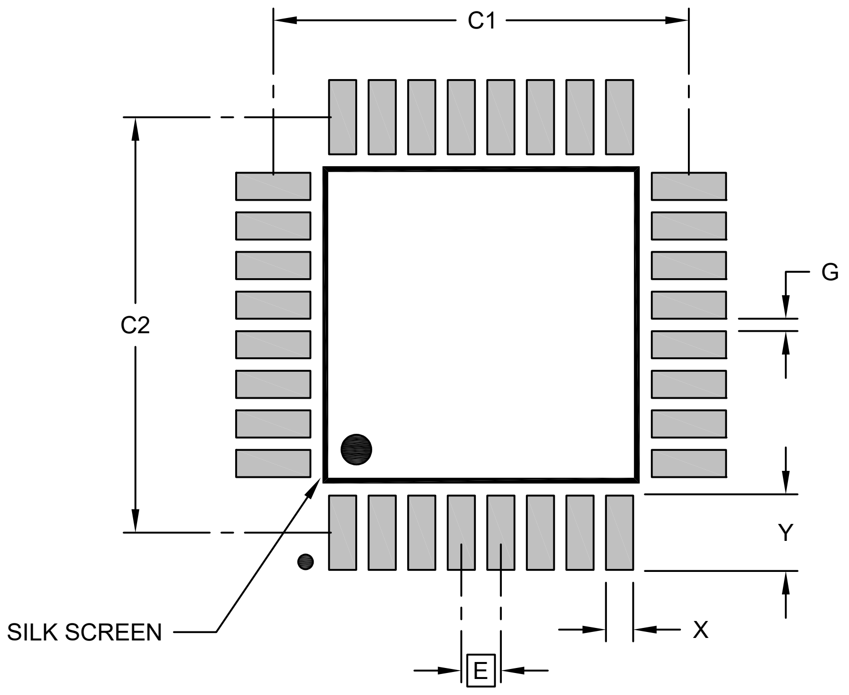
Notes:

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:

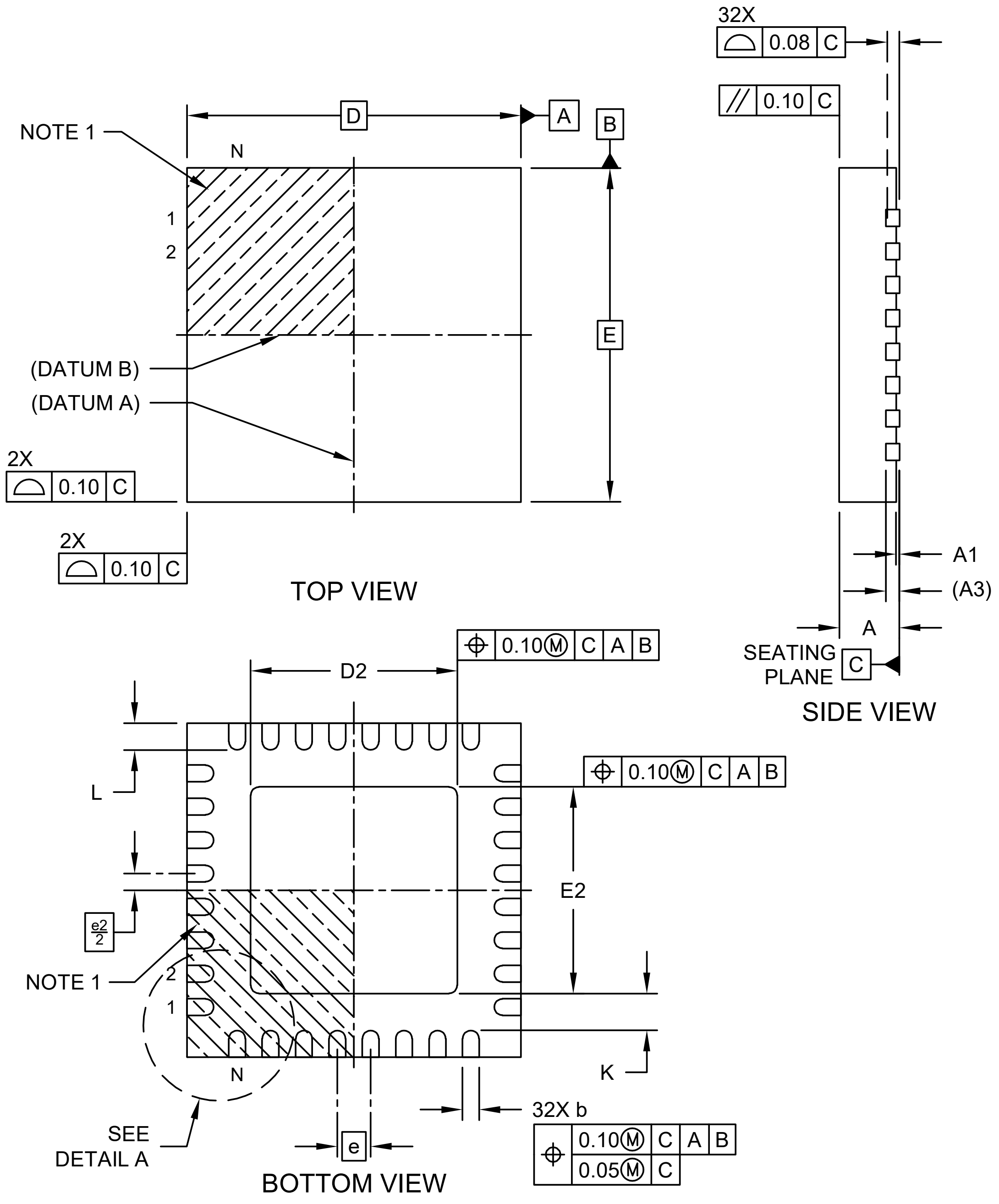
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2074-PT Rev D

36.7 32-Pin VQFN

32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

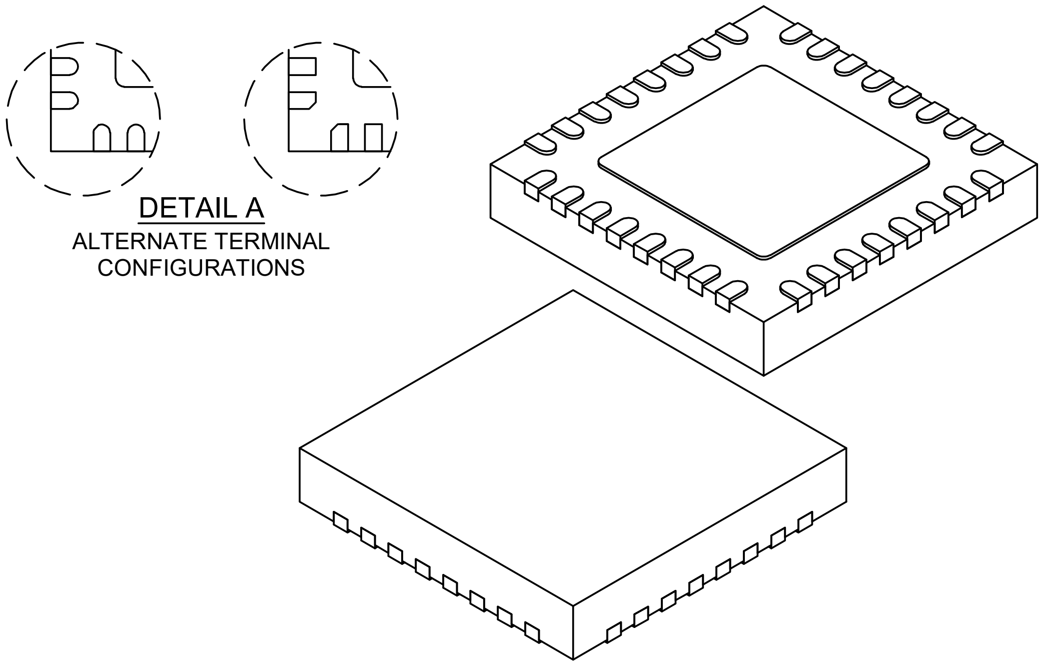
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-21395-RXB Rev C Sheet 1 of 2

32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Number of Terminals	N		32	
Pitch	e		0.50 BSC	
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		5.00 BSC	
Exposed Pad Length	D2	3.00	3.10	3.20
Overall Width	E		5.00 BSC	
Exposed Pad Width	E2	3.00	3.10	3.20
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K	0.20	-	-

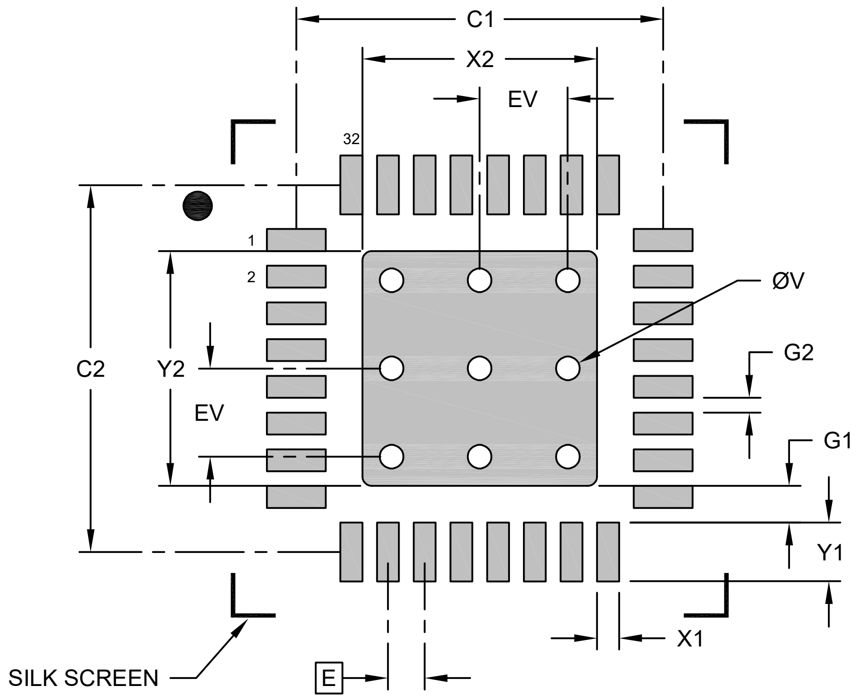
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21395-RXB Rev C Sheet 2 of 2

32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

	Units	MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Center Pad Width	X2			3.20
Center Pad Length	Y2			3.20
Contact Pad Spacing	C1		5.00	
Contact Pad Spacing	C2		5.00	
Contact Pad Width (X32)	X1			0.30
Contact Pad Length (X32)	Y1			0.80
Contact Pad to Center Pad (X32)	G1	0.20		
Contact Pad to Contact Pad (X28)	G2	0.20		
Thermal Via Diameter	V		0.33	
Thermal Via Pitch	EV		1.20	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23395-RXB Rev C

37. Electrical Characteristics

37.1 Disclaimer

All typical values are measured at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. All minimum and maximum values are valid across operating temperature and voltage unless otherwise specified.

The given typical values need to be considered for design guidance only, and the part variation around the values is expected.

37.2 Absolute Maximum Ratings

Stresses beyond those listed in this section can cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification, is unimplied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 37-1. Absolute Maximum Ratings

Parameter	Condition	Rating	Units
Ambient temperature under bias		-40 to +85	°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_{USB} pin		-0.3 to V_{DD}	V
On the $\overline{\text{RESET}}$ pin		-0.3 to 9	V
On all other pins		-0.3 to (V_{DD} + 0.3)	V
Maximum Current			
On the GND pin[(1)]		350	mA
On the V_{DD} pin[(1)]		350	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 device package's power dissipation characterizations may limit the maximum current rating. See 37.9. Thermal Specifications to calculate device specifications.
2. Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma\, I_{OH}\} + \Sigma\, \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma\, (V_{OI} \times I_{OL})$

37.3 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:** $V_{DDMIN} \leq V_{DD} \leq V_{DDMAX}$
- **Operating Voltage with USB Voltage Regulator enabled:** $V_{DDUMIN} \leq V_{DDU} \leq V_{DD}$
- **Operating Temperature:** $T_{A\_MIN} \leq T_A \leq T_{A\_MAX}$

The standard operating conditions for any device are defined as follows:

Table 37-2. Standard Operating Conditions

Parameter		Ratings	Units
V_{DD} — Operating Supply Voltage[1]			
Industrial temperature	V_{DDMIN}	+1.8	V
	V_{DDMAX}	+5.5	V
V_{DDU} — V_{DD} Range for Operating the USB Voltage Regulator[1]			
Industrial temperature	V_{DDUMIN}	+4.35	V
	V_{DDUMAX}	V_{DD}	V
V_{USB} — Supply Voltage for the USB Transceiver[1]			
Industrial temperature	V_{USB}	+3.3	V
f_{USB} — USB Input Clock Frequency			
Industrial temperature	f_{USBMIN}	47.88	MHz
	f_{USBMAX}	48.12	MHz
C_{VUSB} — External Capacitor Value on V_{USB} [2]			
Industrial temperature	C_{VUSB}	470	nF
T_A — Operating Ambient Temperature Range			
Industrial temperature	T_{A_MIN}	-40	°C
	T_{A_MAX}	+85	°C

Notes:
1. Refer to the Supply Voltage parameter in the section *Supply Voltage*.
2. Ceramic or film capacitor to ensure sufficiently low ESR between 0.5 and 4.0Ω.

37.4 Supply Voltage

Table 37-3. Supply Voltage

Symbol	Min.	Typ. †	Max.	Units	Conditions
Supply Voltage[1]					
V_{DD}	1.8	—	5.5	V	
Slew Rate	—	—	1.5	V/μs	$1.8V \leq V_{DD} \leq 5.5V$
V_{DD} Range for Operating the USB Voltage Regulator					
V_{DDU}	3.9	—	V_{DD_MAX}	V	$V_{DDU} = V_{DD}$
Supply Voltage for USB Transceiver					
V_{USB}	3.0	3.3	3.6	V	
Quiescent Input Current of the USB Regulator					
I_Q		180		μA	USB regulator without load
RAM Data Retention[2]					
V_{DR}	1.7	—	—	V	Device in Power-Down mode
Power-on Reset Release Voltage[4]					
V_{POR}	—	1.6	—	V	BOD disabled[3]
t_{POR}	—	1	—	μs	BOD disabled[3]
Power-on Reset Re-Arm Voltage[4]					
V_{PORR}	—	1.25	—	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]

...........continued

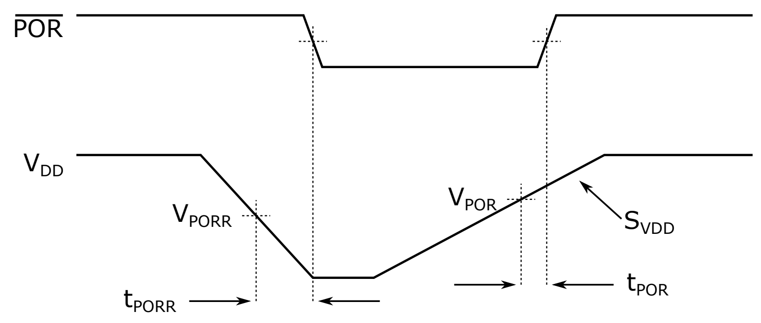
Symbol	Min.	Typ. †	Max.	Units	Conditions

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

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 section *RSTCTRL* for BOD trip point information.
4. Refer to the figure *$\overline{POR}$ and $\overline{PORR}$ with Slow Rising V_{DD}*.

Figure 37-1. $\overline{\text{POR}}$ and $\overline{\text{PORR}}$ with Slow Rising V_{DD}



Note:

- When $\overline{\text{POR}}$ is low, the device is held in Reset

37.5 Power Consumption

Table 37-4. Device Power Consumption

Operating Conditions:
V_{DD} = 3V
T_A = 25°C
System power consumption measured with peripherals disabled and I/O ports driven low with inputs disabled

Symbol	Description	Min.	Typ.	Max.	Units	Conditions
I_{DD}	Active power consumption	—	4.4	—	mA	OSCHF = 24 MHz
		—	1.1	—	mA	OSCHF = 4 MHz
		—	6	—	μA	OSC32K = 32.768 kHz
		—	4.2	—	mA	EXTCLK = 24 MHz
		—	—	—	μA	EXTCLK = 4 MHz
		—	9.0	—	μA	XOSC32K = 32.768 kHz (High Power)
		—	7.5	—	μA	XOSC32K = 32.768 kHz (Low Power)
I_{DD_IDLE}	Idle power consumption	—	2.1	—	mA	OSCHF = 24 MHz
		—	870	—	μA	OSCHF = 4 MHz
		—	3.2	—	μA	OSC32K = 32.768 kHz
		—	1.9	—	mA	EXTCLK = 24 MHz
		—	460	—	μA	EXTCLK = 4 MHz
		—	7.5	—	μA	XOSC32K = 32.768 kHz (High Power)
		—	6.0	—	μA	XOSC32K = 32.768 kHz (Low Power)

...........continued

Operating Conditions:
V_{DD} = 3V
T_A = 25°C
System power consumption measured with peripherals disabled and I/O ports driven low with inputs disabled

Symbol	Description	Min.	Typ.	Max.	Units	Conditions
I_{DD_BASE}[1]	Minimum power consumption in different sleep modes	—	0.65	—	µA	Power-Down or Standby sleep mode, all peripherals disabled[2]
		—	160	—	µA	Power-Down or Standby sleep mode, all peripherals disabled[3]
		—	0.9	—	µA	Power-Down sleep mode, all peripherals disabled[2,4]
I_{RST}	Reset power consumption	—	170	—	µA	$\overline{\text{RESET}}$ line pulled low

Notes:
1. Single supply mode.
2. PMODE configured to AUTO in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register.
3. PMODE configured to FULL in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register.
4. HTLLEN configured to ON in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register.

37.6 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 37-5. Peripherals Power Consumption[1]

Operating Conditions:
V_{DD} = 3.0V
T_A = 25°C
OSCHF at 4 MHz used as clock source
Device in Standby sleep mode

Symbol	Description	Min.	Typ. †	Max. 85°C	Unit	Conditions
I_{DD_WDT}	Watchdog Timer (WDT)	—	600	900	nA	32.768 kHz internal oscillator
I_{DD_VREF}	Voltage Reference (VREF)	—	175	300	µA	ADC0REF enabled, V_{REF} = 2.048V
		—	71	90	µA	ACREF enabled, V_{REF} = 2.048V
		—	40	60	µA	DACREF enabled, V_{REF} = 2.048V
I_{DD_BOD}	Brown-out Detector (BOD)	—	17	25	µA	Brown-out Detect (BOD) continuous
		—	1.6	10	µA	Brown-out Detect (BOD) sampling @128 Hz, including I_{DD_OSC32K}
		—	0.95	10	µA	Brown-out Detect (BOD) sampling @32 Hz, including I_{DD_OSC32K}
I_{DD_TCA}	16-bit Timer/Counter Type A (TCA)	—	6	—	µA	CLK_PER = OSCHF/4 = 1 MHz
I_{DD_TCB}	16-bit Timer/Counter Type B (TCB)	—	3.6	—	µA	
I_{DD_RTC}	Real-Time Counter (RTC)	—	0.7	18	µA	RTC running at 1.024 kHz from OSC32K
			3.9	20	µA	RTC running at 1.024 kHz from XOSC32K, XOSC32KCTRLA.LPMODE = 0
			2.1	18	µA	RTC running at 1.024 kHz from XOSC32K, XOSC32KCTRLA.LPMODE = 1
I_{DD_OSC32K}	32.768 kHz Internal Oscillator (OSC32K)	—	600	900	nA	

...........continued

Operating Conditions:

V_{DD} = 3.0V

T_A = 25°C

OSCHF at 4 MHz used as clock source

Device in Standby sleep mode

Symbol	Description	Min.	Typ. †	Max. 85°C	Unit	Conditions
$I_{DD\_XOSC32K}$	32.768 kHz Crystal Oscillator (XOSC32K)	—	2	—	μA	XOSC32KCTRLA.LPMODE = `0`
		—	1.2	—	μA	XOSC32KCTRLA.LPMODE = `1`
I_{DD_OSCHF}	Internal High Frequency Oscillator (OSCHF)	—	185	—	μA	OSCHF at 4 MHz
I_{DD_ADC}	Analog-to-Digital Converter (ADC)	—	300	600	nA	ADC - Nonconverting
		—	1.1	1.4	μA	ADC @60 ksps(2)
		—	1.1	1.5	μA	ADC @120 ksps(2)
I_{DD_AC}	Analog Comparator (AC)	—	70	105	μA	CTRLA.POWER = `0x0`
		—	17	30	μA	CTRLA.POWER = `0x1`
		—	12	20	μA	CTRLA.POWER = `0x2`
I_{DD_USB}	Universal Serial Bus (USB)	—	2.1	—	mA	V_{DD}=5.0V, VUSBCTRL.USBVREG = `0x1`
I_{DD_UART}	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	—	6.8	—	mA	
$I_{DD\_NVM\_WRITE}$	Flash Programming Write	—	9.2	—	mA	

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

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 section *Power Consumption*.
2. Average power consumption with ADC active in Free Running mode.

37.7 I/O Pins

General I/O Pins

Table 37-6. I/O Pin Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
Input Low Voltage						

...........continued

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
V_{IL}	I/O PORT:					
	• with Schmitt Trigger buffer	—	—	$0.2\times V_{DD}$	V	INLVL selecting `ST`(1)
	• with I^2C levels	—	—	$0.3\times V_{DD}$	V	
	• with SMBus 3.0 levels	—	—	0.8	V	
	• with LVBUF (TTL compatible)	—	—	0.9	V	INLVL selecting `TTL`(1)
	$\overline{\text{RESET}}$ pin	—	—	$0.2\times V_{DD}$	V	
Input High Voltage						
V_{IH}	I/O PORT:					
	• with Schmitt Trigger buffer	$0.8\times V_{DD}$	—	—	V	
	• with I^2C levels	$0.7\times V_{DD}$	—	—	V	
	• with SMBus 3.0 levels	1.35	—	—	V	$0°C \leq T_A \leq +85°C$, $2.5V \leq V_{DD} \leq 5.5V$
		1.45	—	—	V	$-40°C \leq T_A \leq +85°C$, $1.8V \leq V_{DD} \leq 5.5V$
	• with LVBUF (TTL compatible)	1.2	—	—	V	
	$\overline{\text{RESET}}$ Pin	$0.8\times V_{DD}$	—	—	V	
Input Leakage Current(2)						
I_{IL}	I/O PORTS	—	±5	±125	nA	$GND \leq V_{PIN} \leq V_{DD}$, pin at high-impedance, $T_A = 85°C$
	$\overline{\text{RESET}}$ Pin(3)	—	±50	±200	nA	$GND \leq V_{PIN} \leq V_{DD}$, pin at high-impedance, $T_A = 85°C$
Pull-up Current						
I_{PUR}		—	150	200	μA	$V_{DD} = 3.0V$, $V_{PIN} = GND$
Output Low Voltage						
V_{OL}	Standard I/O ports	—	—	0.6	V	$I_{OL} = 10$ mA, $V_{DD} = 3.0V$
Output High Voltage						
V_{OH}	Standard I/O ports	V_{DD}-0.7	—	—	V	$I_{OH} = 6$ mA, $V_{DD} = 3.0V$
I/O Rise Time						
t_{SR}	Rising	—	45	—	ns	PORTCTRL.SRL = `0x01`
	Rising	—	22	—	ns	PORTCTRL.SRL = `0x00`
	Falling	—	30	—	ns	PORTCTRL.SRL = `0x01`
	Falling	—	16	—	ns	PORTCTRL.SRL = `0x00`
Pin Capacitance						
C_{IO}	All I/O pins	—	5	—	pF	

...........continued

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

Notes:

1. The Input Level Select (INLVL) bit is configured in the PORT peripheral by writing either the PORTx.PINCONFIG register or the PORTx.PINnCTRL register.
2. The negative current is defined as the current sourced by the pin.
3. 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.

USB Pins

Table 37-7. USB Pin Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
Input Leakage Current[1]						
I_{ILUD}	Input leakage on USB data pins DM	—	±5	±125	nA	GND ≤ V_{PIN} ≤ V_{DD}, pin at high-impedance, T_A= 85°C
	Input leakage on USB data pins DP	—	±5	±125	nA	GND ≤ V_{PIN} ≤ V_{DD}, pin at high-impedance, T_A= 85°C
I/O Rise Time						
	DP rising	—	13	—	ns	
	DM rising	—	13	—	ns	
	DP falling	—	14	—	ns	
	DM falling	—	14	—	ns	
Input Capacitance						
C_{UD}	Input capacitance on USB data pins DM	—	15	100	pF	pin at high-impedance
	Input capacitance on USB data pins DP	—	15	100	pF	pin at high-impedance

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. Load condition is 50 pF. These parameters are for design guidance only and are not tested.

Notes:

1. The negative current is defined as the current sourced by the pin.
2. 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.

37.8 Memory Programming Specifications

Table 37-8. Memory Programming Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	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	

...........continued

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
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_{BOD}[1]	—	V_{DDMAX}	V	
t_{P_CE}	Chip Erase time	—	11	11.6	ms	
t_{P_PE}	Page Erase time	—	10	11.7	ms	
t_{P_WRD}	Byte/Word Write time	—	70	75	µs	

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. 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.

37.9 Thermal Specifications

Table 37-9. Thermal Specifications

Symbol	Description	Typ.	Unit	Conditions
θ_{JA}	Thermal Resistance Junction to Ambient	50	°C/W	28-pin SPDIP package (SP)
		47	°C/W	28-pin SSOP (SS)
		36	°C/W	28-pin VQFN (STX)
		57	°C/W	32-pin TQFP package (PT)
		33	°C/W	32-pin VQFN package (RXB)
T_{JMAX}	Maximum Junction Temperature			Refer to the *Absolute Maximum Ratings* section

Notes:

- Power dissipation is calculated as $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$
- Internal Power Dissipation is calculated as $P_{INTERNAL} = I_{DD} \times V_{DD}$, where I_{DD} is current to run the chip alone without driving any load on the output pins.
- Derated Power is calculated as $P_{DER} = PD_{MAX} (T_J - T_A)/\theta_{JA}$, where T_A = Ambient Temperature, T_J = Junction Temperature.

37.10 CLKCTRL

37.10.1 System Clock

Table 37-10. System Clock Timing Characteristics

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
f_{CLK_MAIN}	Main clock frequency[1,2]	—	—	24	MHz	

...........continued

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
f_{CY}	Instruction clock frequency	—	f_{CLK_MAIN}	—	MHz	
T_{CY}	Instruction period[3]	41.6	$1/f_{CY}$	—	ns	

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. 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*.
3. The Instruction Cycle Period (T_{CY}) is identical to the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type, under standard operating conditions with the device executing code. Exceeding these specified 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. The 'max' cycle time limit is 'DC' (no clock) for all devices when using an external clock input.

37.10.2 Internal Oscillators

Table 37-11. Internal Oscillator Specifications[1]

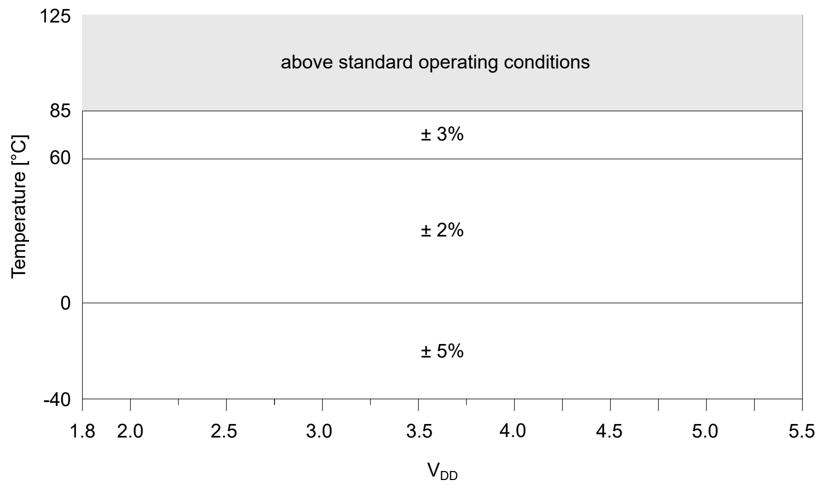
Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
f_{OSCHF}	OSCHF frequency	—	1[2] 2[2] 3[2]	—	MHz	
	Precision calibrated OSCHF frequency	—	4 8 12 16 20 24	—		
$\%_{CAL}$	OSCHF tune step size	—	0.4	—	%	
t_{OSCHF_ST}[3]	OSCHF wake-up from sleep start-up time	—	24	30	µs	Device in Idle or Standby sleep mode, VREGCTRL.PMODE = FULL
		—	220	600	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	1000	µs	Device in Power-Down sleep mode, VREGCTRL.PMODE = AUTO

† Data found in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

Notes:

1. To ensure these oscillator frequency tolerances, V_{DD} and GND must be capacitively decoupled as close to the device as possible. See the *Connection for Power Supply* section of the *Hardware Guidelines* section for details.
2. These parameters are not calibrated.
3. Wake-up times are measured from the wake-up event to code execution.

Figure 37-2. Precision Calibrated OSCHF ($f_{OSCHF} \geq$ 4 MHz) Frequency Accuracy Over Device V_{DD} and Temperature[1]



Note: The figure is applicable only for $f_{OSCHF} \geq$ 4 MHz.

37.10.3 XOSC32K

Table 37-12. 32.768 kHz Crystal Oscillator (XOSC32K) Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	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 *	Equivalent Series Resistance	—	100	—	kΩ	XOSC32KCTRLA.LPMODE = 0
		—	50	—	kΩ	XOSC32KCTRLA.LPMODE = 1
t_{XOSC32_ST} *	XOSC32 start-up time	—	300	—	ms	XOSC32KCTRLA.LPMODE = 0
		—	1000	—	ms	XOSC32KCTRLA.LPMODE = 1

† Data found in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are not tested and are for design guidance only.

***** These parameters are characterized but not tested in production.

37.10.4 XOSCHF

Table 37-13. High Frequency Crystal Oscillator (XOSCHF) Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
f_{XOSCHF} *	Frequency	4	—	24	MHz	
$C_{XTAL1/XTAL2}$ *	Parastatic pin capacitance	—	5	—	pF	
C_L *	Crystal load capacitance	—	12	—	pF	
ESR *	Equivalent Series Resistance	—	200	—	Ω	XTALHF = 4 MHz, XOSCHFCTRLA.FRQRANGE = 0x0
		—	60	—	Ω	XTALHF = 16 MHz, XOSCHFCTRLA.FRQRANGE = 0x1
		—	40	—	Ω	XTALHF = 24 MHz, XOSCHFCTRLA.FRQRANGE = 0x2
t_{XOSC32_HF} *	XOSCHF start-up time	—	700	—	ns	4 MHz crystal

...........continued

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
† Data found in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are not tested and are for design guidance only. * These parameters are characterized but not tested in production.						

37.10.5 External Clock

Figure 37-3. External Clock Waveform

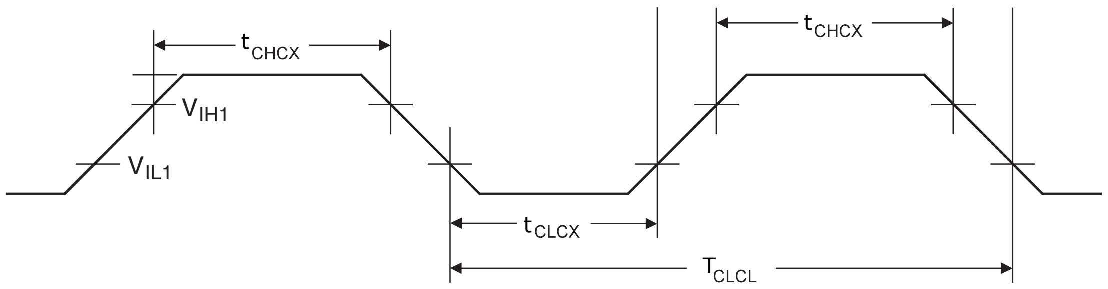


Table 37-14. External Clock Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
f_{CLCL}	Clock frequency	—	—	24	MHz	
T_{CLCL}	Clock period	41.6	—	—	ns	
t_{CHCX}	High time	—	40	—	%	
t_{CLCX}	Low time	—	40	—	%	
ΔT_{CLCL}	Change in period from cycle to cycle time	—	20	—	%	
† Data found in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are not tested and are for design guidance only.						

37.11 RSTCTRL

Table 37-15. Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-out Detector Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
t_{RST} *	$\overline{\text{RESET}}$ pin pulse-width low to ensure a Reset	2.5	—	—	µs	
R_{RST_UP} *	$\overline{\text{RESET}}$ pin pull-up resistor	—	35	—	kΩ	
T_{WDT} *	Watchdog Timer time-out period	—	500	—	ms	1:512 Prescaler
T_{SUT} *	Power-up timer period	—	64	—	ms	SUT = `0x07`
T_{OST} *	Oscillator start-up timer period(1)	—	1024	—	cycles	
V_{BOD}	Brown-out Detect Voltage(2)	1.80	1.90	2.10	V	BODLEVEL0
		2.30	2.45	2.65		BODLEVEL1
		2.55	2.70	2.90		BODLEVEL2
		2.70	2.85	3.05		BODLEVEL3
V_{BOD_HYS}	Brown-out Detect hysteresis	—	44	—	mV	
t_{BOD_ST}	Brown-out Detect start-up time	—	1.9	—	µs	
t_{BOD_128HZ}	BOD Response Time Sampling mode @128 Hz	—	7.81	—	ms	SAMPFREQ = `0`
t_{BOD_32HZ}	BOD Response Time Sampling mode @32 Hz	—	31.25	—	ms	SAMPFREQ = `1`
t_{BOD_RST}	Brown-out reset response time	—	3	—	µs	

...........continued

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.						
* These parameters are characterized but not tested in production.						

Notes:

1. By design, the Oscillator Start-up Timer (T_{OST}) counts the first 1024 cycles, independent of frequency.
2. To ensure these voltage tolerances, V_{DD} and AV_{DD} 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 37-16. Voltage Level Monitor Threshold Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
V_{DET} *	Voltage detection threshold	1	5	10	% of BOD threshold	VLMLVL = `0x01`
		9	15	22		VLMLVL = `0x02`
		19	25	32		VLMLVL = `0x03`

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

* These parameters are characterized but not tested in production.

37.12 VREF

Table 37-17. V_{REF} Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
V_{VREF_1V024}(1)	Internal Voltage Reference 1.024V	-4	—	+4	%	V_{DD} ≥ 2.5V, -40°C ≤ T_A ≤ +85°C
V_{VREF_2V048}(1)	Internal Voltage Reference 2.048V	-4	—	+4	%	V_{DD} ≥ 2.5V, -40°C ≤ T_A ≤ +85°C
V_{VREF_4V096}(1)	Internal Voltage Reference 4.096V	-4	—	+4	%	V_{DD} ≥ 4.55V, -40°C ≤ T_A ≤ +85°C
V_{VREF_2V500}(1)	Internal Voltage Reference 2.5V	-4	—	+4	%	V_{DD} ≥ 2.7V, -40°C ≤ T_A ≤ +85°C
V_{VREFA} *	VREFA input pin voltage	1.8	—	V_{DD}	V	V_{DD} < 2.7V
		1.024	—	V_{DD}		V_{DD} ≥ 2.7V
t_{INTREF} *	Delay for changing voltage reference	—	2	—	μs	
t_{VREF_ST} *	VREF start-up time	—	10	—	μs	CLKCTRL.MCLKCTRLA = `0x00` or `0x03`
		—	200	—	μs	CLKCTRL.MCLKCTRLA = `0x01` or `0x02`

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

* These parameters are characterized but not tested in production.

Note:

1. The V_{VREF_xVxxx} symbol refers to the respective values of the REFSEL bit fields in the VREF.ADC0REF, VREF.DAC0REF and VREF.ACREF registers.

37.13 USART

Figure 37-4. USART in SPI Mode - Timing Requirements in Host Mode

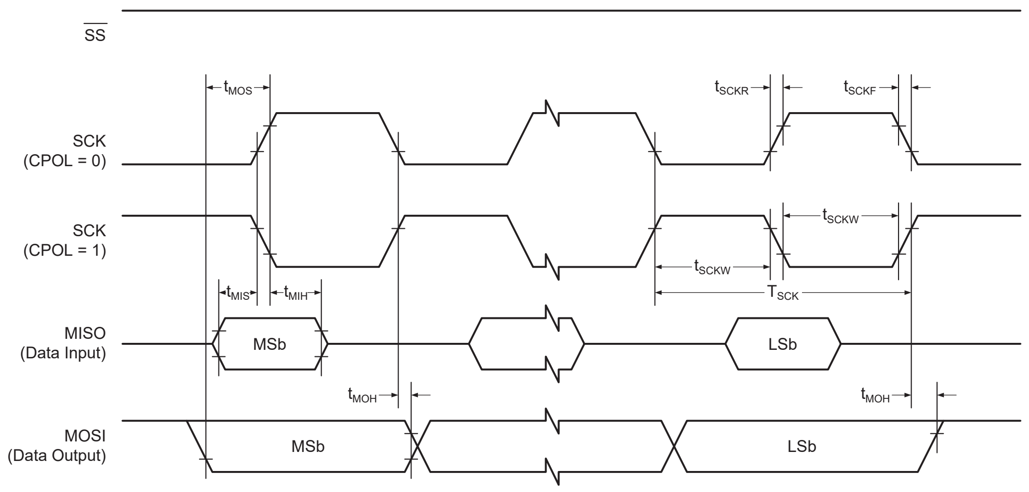


Table 37-18. USART in SPI Host Mode - Timing Characteristics

Symbol	Description	Min.	Typ.	Max.	Unit	Condition
f_{SCK}	SCK clock frequency	—	—	$f_{CLK\_PER}/2$	MHz	
T_{SCK}	SCK period	$2{\times}T_{CLK\_PER}$	—	—	ns	
t_{SCKW}	SCK high/low width	—	$0.5{\times}T_{SCK}$	—	ns	
t_{SCKR}	SCK rise time	—	2.7	—	ns	
t_{SCKF}	SCK fall time	—	2.7	—	ns	
t_{MIS}	MISO setup to SCK	—	10	—	ns	
t_{MIH}	MISO hold after SCK	—	10	—	ns	
t_{MOS}	MOSI setup to SCK	—	$0.5{\times}T_{SCK}$	—	ns	
t_{MOH}	MOSI hold after SCK	—	1.0	—	ns	

37.14 SPI

Figure 37-5. SPI - Timing Requirements in Host Mode

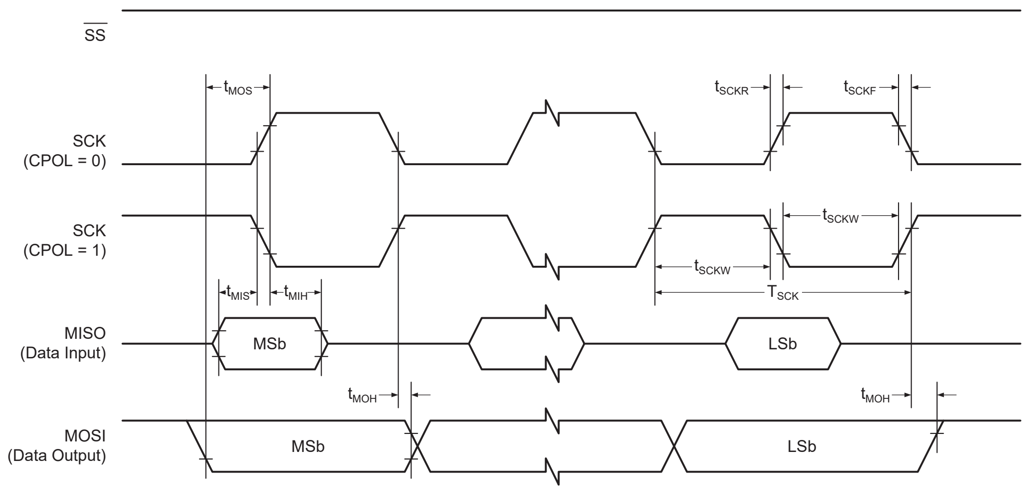


Table 37-19. SPI - Timing Characteristics in Host Mode

Symbol	Description	Min.	Typ.	Max.	Unit	Condition
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_{SCK}	—	ns	
t_{MIH}	MISO hold after SCK	—	0	—	ns	

1. These parameters are characterized but not tested in production.

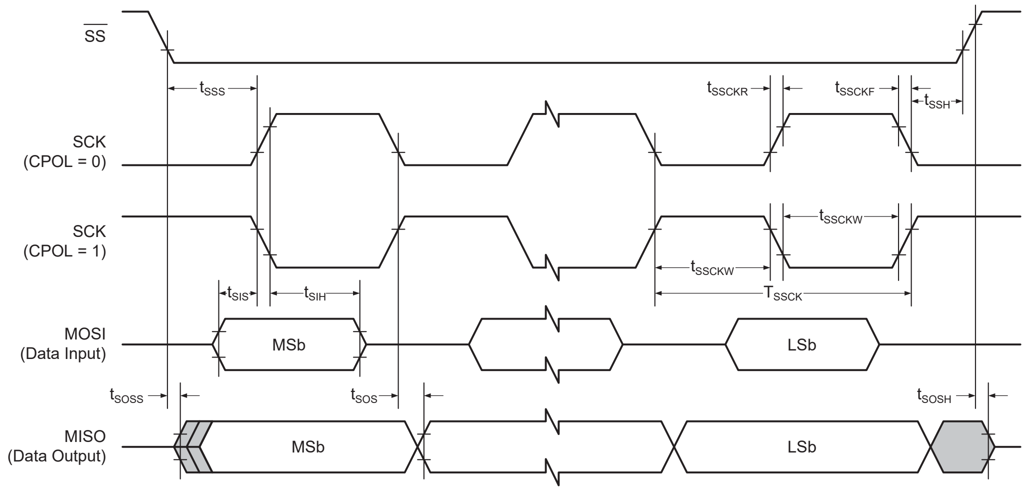


Figure 37-6. SPI - Timing Requirements in Client Mode

Table 37-20. SPI - Timing Characteristics in Client Mode

Symbol	Description	Min.	Typ.	Max.	Unit	Condition
f_{SSCK} (1)	Client SCK clock frequency	—	—	$f_{CLK\_PER}/6$	MHz	
T_{SSCK} (1)	Client SCK period	$6{\times}T_{CLK\_PER}$	—	—	ns	
t_{SSCKW} (1)	SCK high/low width	$3{\times}T_{CLK\_PER}$	—	—	ns	
t_{SIS} (1)	MOSI setup to SCK	0	—	—	ns	
t_{SIH} (1)	MOSI hold after SCK	$3{\times}T_{CLK\_PER}$	—	—	ns	
t_{SSS} (1)	SS low before SCK	T_{CLK_PER}	—	—	ns	
t_{SSH} (1)	SS high after SCK	T_{CLK_PER}	—	—	ns	
t_{SOS}	MISO valid after SCK	—	t_{SR} (2)	—	ns	$f_{SSCK}{\geq}f_{CLK\_PER}/6$
		—	—	—		$f_{SSCK}{<}f_{CLK\_PER}/6$
t_{SOSS}	MISO setup after SS low	—	t_{SR} (2)	—	ns	
t_{SOSH}	MISO hold after SS low	—	t_{SR} (2)	—	ns	

1. These parameters are characterized but not tested in production.
2. t_{SR} is the I/O pin rise time.

37.15 TWI

Figure 37-7. TWI - Timing Requirements

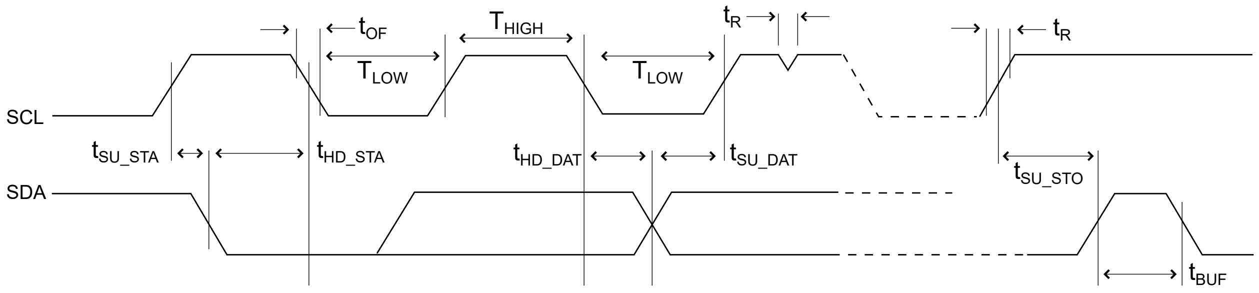


Table 37-21. TWI - Timing Characteristics

Symbol	Description	Min.	Typ.	Max.	Unit	Condition
f_{SCL}	SCL clock frequency	0	—	1000	kHz	Max. frequency requires system clock at 10 MHz
		0	—	400	kHz	For V_{DD} < 2.2V
V_{IH}	Input high voltage	0.7×V_{DD}	—	—	V	
V_{IL}	Input low voltage	—	—	0.3×V_{DD}	V	
V_{HYS}	Hysteresis of Schmitt Trigger inputs	0.1×V_{DD}		0.4×V_{DD}	V	
V_{OL}	Output low voltage	—	—	0.2xV_{DD}	V	I_{load} = 20 mA, Fast mode+
		—	—	0.4V		I_{load} = 3 mA, Normal mode, V_{DD} > 2V
		—	—	0.2×V_{DD}		I_{load} = 3 mA, Normal mode, V_{DD} ≤ 2V
I_{OL}	Low level output current	3	—	—	mA	f_{SCL} ≤ 400 kHz, V_{OL} = 0.4V
		20	—	—		f_{SCL} ≤ 1 MHz, V_{OL} = 0.4V
C_B	Capacitive load for each bus line	—	—	400	pF	f_{SCL} ≤ 100 kHz
		—	—	400		f_{SCL} ≤ 400 kHz
		—	—	550		f_{SCL} ≤1 MHz
t_R	Rise time for both SDA and SCL	—	—	1000	ns	f_{SCL} ≤ 100 kHz
		20	—	300		f_{SCL} ≤ 400 kHz
		—	—	120		f_{SCL} ≤ 1 MHz
t_{OF}	Output fall time from V_{IHmin} to V_{ILmax}	—	—	250	ns	f_{SCL} ≤ 100 kHz 10 pF < C_B < 400 pF
		20×(V_{DD}/5.5V)	—	250		f_{SCL} ≤ 400 kHz 10 pF < C_B < 400 pF
		20×(V_{DD}/5.5V)	—	120		f_{SCL} ≤ 1 MHz 10 pF < C_B < 400 pF
t_{SP}	Spikes suppressed by the input filter	0	—	50	ns	
I_L	Input current for each I/O pin	—	—	1	µA	0.1×V_{DD} < V_I < 0.9×V_{DD}
C_I	Capacitance for each I/O pin	—	—	10	pF	

...........continued

Symbol	Description	Min.	Typ.	Max.	Unit	Condition
R_P	Value of pull-up resistor	$(V_{DD}-V_{OL}(max)) /I_{OL}$	—	1000 ns/ $(0.8473 \times C_B)$	Ω	$f_{SCL} \le 100$ kHz
		—	—	300 ns/ $(0.8473 \times C_B)$		$f_{SCL} \le 400$ kHz
		—	—	120 ns/ $(0.8473 \times C_B)$		$f_{SCL} \le 1$ MHz
t_{HD_STA}	Hold time (repeated) Start condition	4.0	—	—	μs	$f_{SCL} \le 100$ kHz
		0.6	—	—		$f_{SCL} \le 400$ kHz
		0.26	—	—		$f_{SCL} \le 1$ MHz
		—	2.1	—	T_{SCL}	Start
		—	3.1	—		Repeated Start
T_{LOW}	Low period of SCL Clock	4.7	—	—	μs	$f_{SCL} \le 100$ kHz
		1.3	—	—		$f_{SCL} \le 400$ kHz
		0.5	—	—		$f_{SCL} \le 1$ MHz
T_{HIGH}	High period of SCL Clock	4.0	—	—	μs	$f_{SCL} \le 100$ kHz
		0.6	—	—		$f_{SCL} \le 400$ kHz
		0.26	—	—		$f_{SCL} \le 1$ MHz
t_{SU_STA}	Setup time for a repeated Start condition	4.7	—	—	μs	$f_{SCL} \le 100$ kHz
		0.6	—	—		$f_{SCL} \le 400$ kHz
		0.26	—	—		$f_{SCL} \le 1$ MHz
		—	3	—	T_{SCL}	
t_{HD_DAT}	Data hold time	—	0	—	ns	SDAHOLD[1:0] = `0x0`
		30	—	300		SDAHOLD[1:0] = `0x1`
		120	—	420		SDAHOLD[1:0] = `0x2`
		300	—	900		SDAHOLD[1:0] = `0x3`
t_{SU_DAT}	Data setup time	250	—	—	ns	$f_{SCL} \le 100$ kHz
		100	—	—		$f_{SCL} \le 400$ kHz
		50	—	—		$f_{SCL} \le 1$ MHz
t_{SU_STO}	Setup time for Stop condition	4	—	—	μs	$f_{SCL} \le 100$ kHz
		0.6	—	—		$f_{SCL} \le 400$ kHz
		0.26	—	—		$f_{SCL} \le 1$ MHz
		—	2	—	T_{SCL}	
t_{BUF}	Bus free time between a Stop and Start condition	4.7	—	—	μs	$f_{SCL} \le 100$ kHz
		1.3	—	—		$f_{SCL} \le 400$ kHz
		0.5	—	—		$f_{SCL} \le 1$ MHz
		—	2	—	T_{SCL}	

Note: I^2C Fm+ is supported only for above 2.7V.

37.16 ADC

Table 37-22. ADC Accuracy Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
Operating Conditions: V_{ADCREF} = 3.0V; ADC in 10-bit, single ended conversion mode; f_{CLK_ADC} = 1 MHz						
N_R	Resolution	—	—	10	bit	
E_{INL}	Integral nonlinearity error	-1.8	0.1	1.8	LSb	
E_{DNL}	Differential nonlinearity error(1)	-1	0.1	1	LSb	
E_{OFF}	Offset error	0	2.5	5	LSb	
E_{GAIN}	Gain error	-5	1.5	5	LSb	
V_{ADCREF} *	ADC reference voltage	1.024	—	V_{DD}	V	$f_{CLK\_ADC} \leq 500$ kHz
		1.8	—	V_{DD}	V	
V_{AIN}	Full-scale range	GND	—	V_{ADCREF}	V	
Z_{AIN}	Recommended impedance of analog voltage source	—	10	—	kΩ	
R_{VREFA}	ADC voltage reference ladder impedance(2)	—	50	—	kΩ	
	V_{DD}/10 divider accuracy (VDDDIV10)	—	±10	—	%	Measured with ADC using on-chip internal reference

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

***** 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 impedance seen by the VREFA pin when the external reference is selected.

Table 37-23. ADC Conversion Timing Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
T_{CLK_ADC} *	ADC clock period	0.5	—	8	µs	
t_{CNV}	Conversion time	—	$11.5T_{CLK\_ADC} + 2T_{CLK\_PER}$	—		
f_{ADC} *	Sample rate	8	—	170	ksps	CLK_PER=24 MHz
t_{SENSE} *	Delay for changing MUXPOS to TEMP	—	40	—	µs	
t_{ADC_INIT} *	Initialization time	—	6	—	µs	

† Data in the "Typ." column is at T_A = 25°C and V_{DD} = 3.0V unless otherwise specified. These parameters are for design guidance only and are not tested.

***** These parameters are characterized but not tested in production.

37.17 AC

Table 37-24. Analog Comparator Specifications

Operating Conditions: V_{DD} = 3.0V, T_A = 25°C						
Symbol	**Description**	**Min.**	**Typ.**	**Max.**	**Unit**	**Condition**
V_{IN}	Input voltage range	-0.2	—	V_{DD}	V	
V_{OFF}	Input offset voltage	—	±5	—	mV	0.7V < V_{IN} < (V_{DD}-0.7V)
		—	±20	—		-0.2V < V_{IN} < V_{DD}
CMRR	Common Mode input Rejection Ratio	—	70	—	dB	
V_{HYST}	Hysteresis	—	10	—	mV	HYSMODE = `0x1`
		—	25	—		HYSMODE = `0x2`
		—	50	—		HYSMODE = `0x3`
t_{RESP}	Response time, rising edge	—	85	—	ns	Power Profile 0
	Response time, falling edge	—	85	—	ns	
	Response time, rising edge	—	250	—	ns	Power Profile 1
	Response time, falling edge	—	220	—	ns	
	Response time, rising edge	—	460	—	ns	Power Profile 2
	Response time, falling edge	—	430	—	ns	

37.18 UPDI

Figure 37-8. UPDI Enable Sequence with Dedicated UPDI Pin

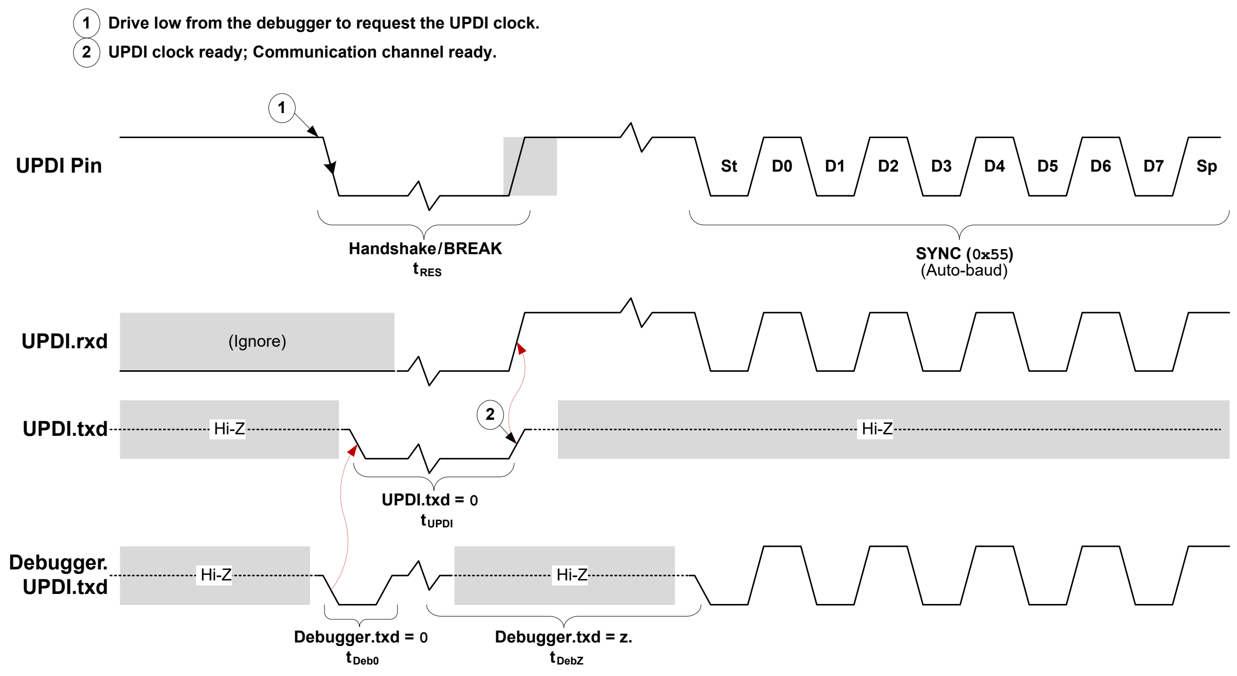


Table 37-25. UPDI Timing Specifications

Symbol	Description	Min.	Max.	Unit	
t_{RES} *	Duration of Handshake/Break on $\overline{\text{RESET}}$	10	200	µs	

...........continued

Symbol	Description	Min.	Max.	Unit	
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	$0°C \leq T_A \leq +50°C$
		—	0.9	Mbps	$T_A < 0°C$ or $T_A > +50°C$
* These parameters are characterized but not tested in production.					

Figure 37-9. UPDI Enable Sequence by High-Voltage (HV) Programming

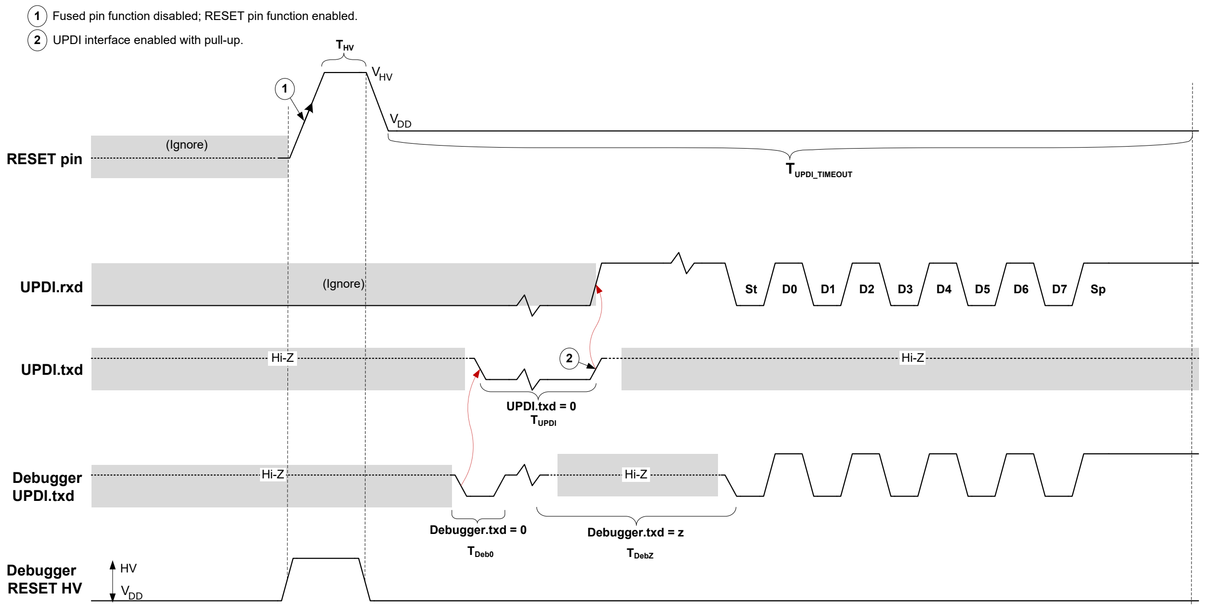


Table 37-26. UPDI HV Pulse Specifications

Symbol	Description	Min.	Typ.	Max.	Unit	Conditions
V_{HV} *	Debugger RESET HV signal level	$V_{DD}+2$	7.5	8.5	V	Never exceed the abs. max. ratings of the $\overline{\text{RESET}}$ pin
T_{HV} **	Debugger RESET HV signal duration	10			µs	
$T_{UPDI\_TIMEOUT}$ *	Time to receive valid key after HV pulse		65		ms	
* These parameters are characterized but not tested in production.						
** These parameters are for design guidance only and are not tested.						

38. Characteristics Graphs

Characteristics Graphs are not available at this time.

39. 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).

39.1 Revision History

Doc. Rev.	Date	Comments
A	02/2024	Preliminary data sheet

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

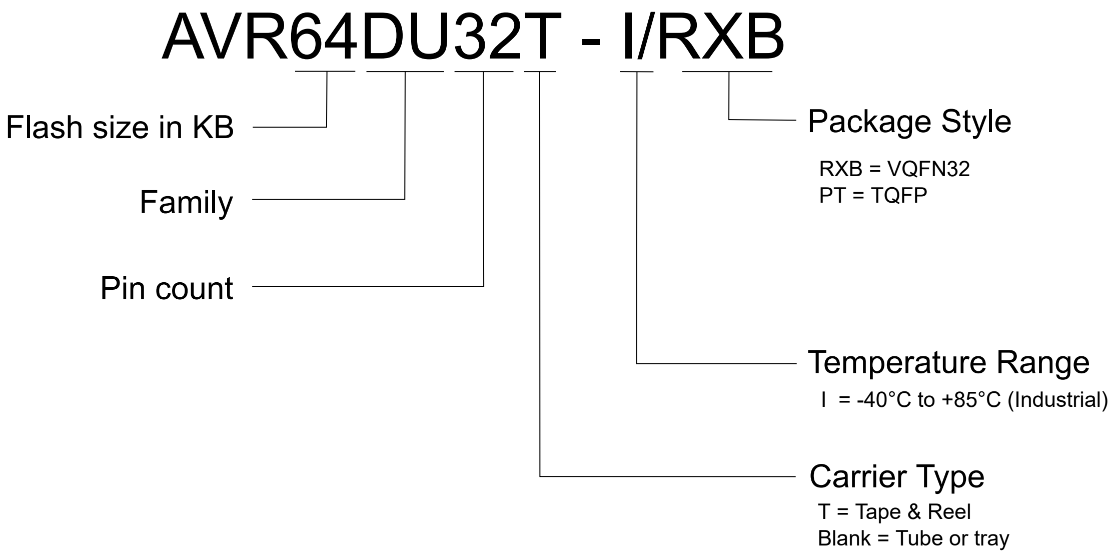
- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Note: 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 with the Tape & Reel option.

Trademarks

ISBN: 978-1-6683-3898-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra’anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820