8/16-bit Atmel AVR XMEGA Microcontrollers

XMEGA E MANUAL

PRELIMINARY

This document contains complete and detailed description of all modules included in the Atmel®AVR®XMEGA®E microcontroller family. The XMEGA E is a family of low-power, high-performance, and peripheral-rich CMOS 8/16-bit microcontrollers based on the AVR enhanced RISC architecture. The available XMEGA E modules described in this manual are:

- Atmel AVR CPU
- Memories
- EDMA - Enhanced direct memory access
- Event system
- System clock and clock options
- Power management and sleep modes
- Reset system
- WDT - Watchdog timer
- Interrupts and programmable multilevel interrupt controller
- PORT - I/O ports
- TC4/5 - 16-bit timer/counters
- WeX - Waveform extension
- Hi-Res - High resolution extension
- Fault - Fault extension
- RTC - Real-time counter
- TWI - Two-wire serial interface
- SPI - Serial peripheral interface
- USART - Universal synchronous and asynchronous serial receiver and transmitter
- IRCOM - Infrared communication module
- XCL - XMEGA custom logic
- CRC - Cyclic redundancy check
- ADC - Analog-to-digital converter
- DAC - Digital- to- analog converter
- AC - Analog comparator
- PDI - Program and debug interface
- Memory Programming
- Peripheral module address map
- Instruction set summary
- Manual revision history
- Table of contents

1. About the manual

This document contains in-depth documentation of all peripherals and modules available for the Atmel AVR XMEGA E microcontroller family. All features are documented on a functional level and described in a general sense. All peripherals and modules described in this manual may not be present in all XMEGA E devices.

For all device-specific information such as characterization data, memory sizes, modules, peripherals available and their absolute memory addresses, refer to the device datasheets. When several instances of a peripheral exists in one device, each instance will have a unique name. For example each port module (PORT) have unique name, such as PORTA, PORTB, etc. Register and bit names are unique within one module instance.

For more details on applied use and code examples for peripherals and modules, refer to the Atmel AVR XMEGA specific application notes available from http://www.atmel.com/avr.

1.1 Reading the manual

The main sections describe the various modules and peripherals. Each section contains a short feature list and overview describing the module. The remaining section describes the features and functions in more detail.

The register description sections list all registers and describe each register, bit and flag with their function. This includes details on how to set up and enable various features in the module. When multiple bits are needed for a configuration setting, these are grouped together in a bit group. The possible bit group configurations are listed for all bit groups together with their associated Group Configuration and a short description. The Group Configuration refers to the defined configuration name used in the Atmel AVR XMEGA assembler header files and application note source code.

The register summary sections list the internal register map for each module type.

The interrupt vector summary sections list the interrupt vectors and offset address for each module type.

1.2 Resources

A comprehensive set of development tools, application notes, and datasheets are available for download from http://www.atmel.com/avr.

1.3 Recommended reading

- XMEGA E device datasheets
- XMEGA application notes

This manual contains general modules and peripheral descriptions. The AVR XMEGA E device datasheets contains the device-specific information. The XMEGA application notes and Atmel Software Framework contain example code and show applied use of the modules and peripherals.

For new users, it is recommended to read the AVR1000 - Getting Started Writing C Code for Atmel XMEGA.

2. Overview

The AVR XMEGA E microcontrollers is a family of low-power, high-performance, and peripheral-rich CMOS 8/16-bit microcontrollers based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the XMEGA E devices achieve throughputs approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

The AVR CPU combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the arithmetic logic unit (ALU), allowing two independent registers to be accessed in a single instruction, executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs many times faster than conventional single-accumulator or CISC based microcontrollers.

The XMEGA E devices provide the following features: in-system programmable flash; internal EEPROM and SRAM; four-channel enhanced DMA controller (EDMA); eight-channel event system with asynchronous event support; programmable multilevel interrupt controller; up to 26 general purpose I/O lines; 16-bit real-time counter (RTC) with digital correction; up to three flexible, 16-bit timer/counters with capture, compare and PWM modes; up to two USARTs; one I^2C and SMBUS compatible two-wire serial interfaces (TWI); one serial peripheral interfaces (SPI); one XMEGA custom logic (XCL) with timer/counter and logic functions; CRC module; one 16-channel, 12-bit ADC with programmable gain, offset and gain correction, averaging, over-sampling and decimation; one 2-channel, 12-bit DAC; two analog comparators with window mode; programmable watchdog timer with separate internal oscillator; accurate internal oscillators with PLL and prescaler; and programmable brown-out detection.

The program and debug interface (PDI), a fast, two-pin interface for programming and debugging, is available. Selected devices also have an IEEE std. 1149.1 compliant JTAG interface, and this can also be used for on-chip debug and programming.

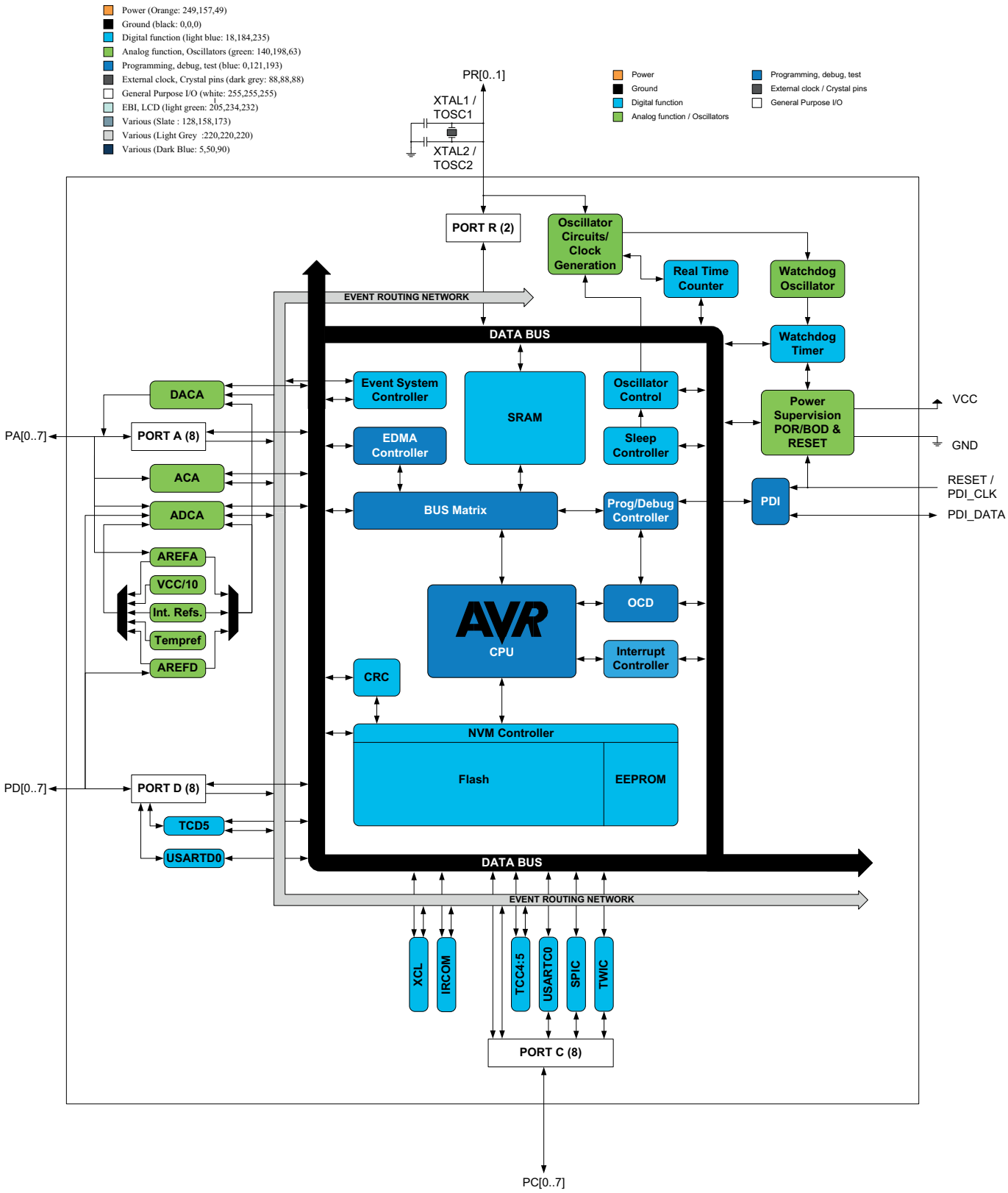
The Atmel AVR XMEGA devices have five software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, EDMA controller, event system, interrupt controller, and all peripherals to continue functioning. The power-down mode saves the SRAM and register contents, but stops the oscillators, disabling all other functions until the next TWI, or pin-change interrupt, or reset. In power-save mode, the asynchronous real-time counter continues to run, allowing the application to maintain a timer base while the rest of the device is sleeping. In standby mode, the external crystal oscillator keeps running while the rest of the device is sleeping. This allows very fast startup from the external crystal, combined with low power consumption. In extended standby mode, both the main oscillator and the asynchronous timer continue to run. To further reduce power consumption, the peripheral clock to each individual peripheral can optionally be stopped in active mode and idle sleep mode. The low power internal 8MHz oscillator allows very fast start-up time, combined with low power modes.

The devices are manufactured using Atmel high-density, nonvolatile memory technology. The program flash memory can be reprogrammed in-system through the PDI interface. A boot loader running in the device can use any interface to download the application program to the flash memory. By combining an 8/16-bit RISC CPU with In-system, self-programmable flash, the Atmel AVR XMEGA is a powerful microcontroller family that provides a highly flexible and cost effective solution for many embedded applications.

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

2.1 Block diagram

Figure 2-1. XMEGA E block diagram.



In Table 2-1 on page 5 a feature summary for the XMEGA E family is shown, split into one feature summary column for each sub-family. Each sub-family has identical feature set, but different memory options, refer to their device datasheet for ordering codes and memory options.

Table 2-1. XMEGA E feature summary overview.

Feature	Details / sub-family	E5
Pins, I/O	Total	32
	Programmable I/O pins	26
Memory	Program memory (KB)	8 - 32
	Boot memory (KB)	2 - 4
	SRAM (KB)	1 - 4
	EEPROM (Bytes)	512
	General purpose registers	4
Package	TQFP	32A
	QFN /VQFN	32Z
QTouch	Sense channels	56
EDMA Controller	Channels	4
Event System	Channels	8
	QDEC	1
	Rotary	1
Crystal Oscillator	0.4 - 16MHz XOSC	Yes
	32.768 kHz TOSC	Yes
Internal Oscillator	8MHz calibrated	Yes
	32MHz calibrated	Yes
	128MHz PLL	Yes
	32.768kHz calibrated	Yes
	32kHz ULP	Yes
Timer / Counter	TC4 - 16-bit, 4 CC	1
	TC5 - 16-bit, 2 CC	2
	Hi-Res	1
	WeX	1
	FAULT	2
	RTC	Yes
XMEGA Custom Logic	BTC0 - 8-bit, 1 CC	1
	BTC1 - 8-bit, 1 CC	1
	LUT, 2-input, one output	2

Feature	Details / sub-family	E5
Serial Communication	USART	2
	SPI	1
	TWI	1
CRC	CRC-16	Yes
	CRC-32	Yes
Analog to Digital Converter (ADC)		1
	Resolution (bits)	12
	Oversampling extra resolution (bits)	4
	Sampling speed (kbps)	300
	External inputs per ADC	16
	Conversion channels	1
	Offset/gain error correction	Yes
	Averaging (samples)	1 - 1024
Digital to Analog Converter (DAC)		1
	Resolution (bits)	12
	Sampling speed (kbps)	1000
	Output channels per DAC	2
Analog Comparator (AC)		2
Program and Debug Interface (PDI)	PDI	Yes

3. Atmel AVR CPU

3.1 Features

- 8/16-bit, high-performance Atmel AVR RISC CPU
  - 141 instructions
  - Hardware multiplier
- 32x8-bit registers directly connected to the ALU
- Stack in RAM
- Stack pointer accessible in I/O memory space
- Direct addressing of up to 16MB of program memory and 16MB of data memory
- True 16/24-bit access to 16/24-bit I/O registers
- Efficient support for 8-, 16-, and 32-bit arithmetic
- Configuration change protection of system-critical features

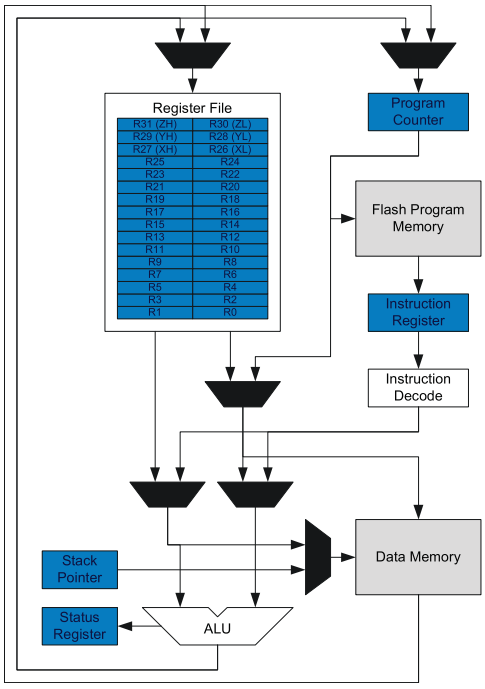
3.2 Overview

All Atmel AVR XMEGA devices use the 8/16-bit AVR CPU. The main function of the CPU is to execute the code and perform all calculations. The CPU is able to access memories, perform calculations, control peripherals, and execute the program in the flash memory. Interrupt handling is described in a separate section, “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.

3.3 Architectural overview

In order to maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate memories and buses for program and data. Instructions in the program memory are executed with single-level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This enables instructions to be executed on every clock cycle. For a summary of all AVR instructions, refer to “Instruction Set Summary” on page 424. For details of all AVR instructions, refer to http://www.atmel.com/avr.

Figure 3-1. Block diagram of the AVR CPU architecture.



The arithmetic logic unit (ALU) supports arithmetic and logic operations between registers or between a constant and a register. Single-register operations can also be executed in the ALU. After an arithmetic operation, the status register is updated to reflect information about the result of the operation.

The ALU is directly connected to the fast-access register file. The 32 x 8-bit general purpose working registers all have single clock cycle access time allowing single-cycle arithmetic logic unit operation between registers or between a register and an immediate. Six of the 32 registers can be used as three 16-bit address pointers for program and data space addressing, enabling efficient address calculations.

The memory spaces are linear. The data memory space and the program memory space are two different memory spaces.

The data memory space is divided into I/O registers, SRAM, and external RAM. In addition, the EEPROM is memory mapped in the data memory.

All I/O status and control registers reside in the lowest 4KB addresses of the data memory. This is referred to as the I/O memory space. The lowest 64 addresses can be accessed directly, or as the data space locations from 0x00 to 0x3F. The rest is the extended I/O memory space, ranging from 0x0040 to 0x0FFF. I/O registers here must be accessed as data space locations using load (LD/LDS/LDD) and store (ST/STS/STD) instructions.

The SRAM holds data. Code execution from SRAM is not supported. It can easily be accessed through the five different addressing modes supported in the AVR architecture. The first SRAM address is 0x2000.

Data addresses 0x1000 to 0x1FFF are reserved for EEPROM.

The program memory is divided in two sections, the application program section and the boot program section. Both sections have dedicated lock bits for write and read/write protection. The SPM instruction that is used for self-programming of the application flash memory must reside in the boot program section. The application section contains an application table section with separate lock bits for write and read/write protection. The application table section can be used for save storing of nonvolatile data in the program memory.

3.4 ALU - Arithmetic logic unit

The arithmetic logic unit supports arithmetic and logic operations between registers or between a constant and a register. Single-register operations can also be executed. The ALU operates in direct connection with all 32 general purpose registers. In a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed and the result is stored in the register file. After an arithmetic or logic operation, the status register 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 is supported, and the instruction set allows for efficient implementation of 32-bit arithmetic. The hardware multiplier supports signed and unsigned multiplication and fractional format.

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

A multiplication takes two CPU clock cycles.

3.5 Program flow

After reset, the CPU starts to execute instructions from the lowest address in the flash program memory '0.' The program counter (PC) addresses the next instruction to be fetched.

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

During interrupts and subroutine calls, the return address PC is stored on the stack. 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 reset, the stack pointer (SP) points to the highest address in the internal SRAM. The SP is read/write accessible in the I/O memory space, enabling easy implementation of multiple stacks or stack areas. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR CPU.

3.6 Instruction execution timing

The AVR CPU is clocked by the CPU clock, clk_{CPU}. No internal clock division is used. Figure 3-2 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept used to obtain up to 1MIPS/MHz performance with high power efficiency.

Figure 3-2. The parallel instruction fetches and instruction executions.

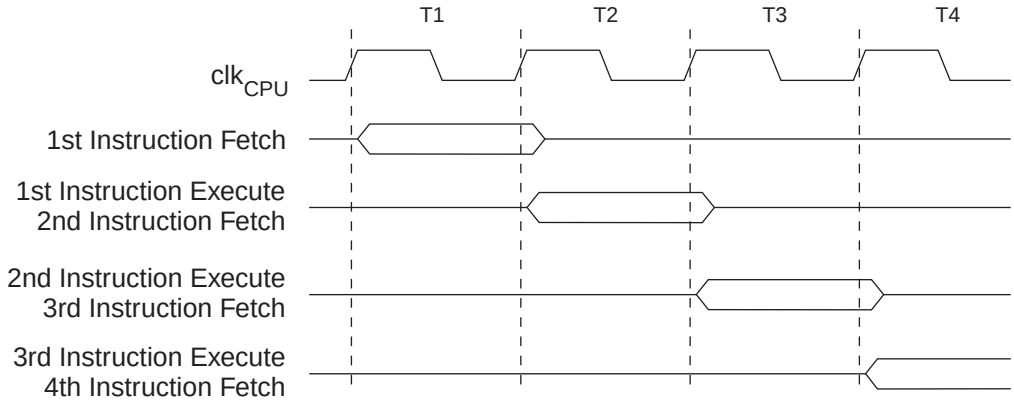
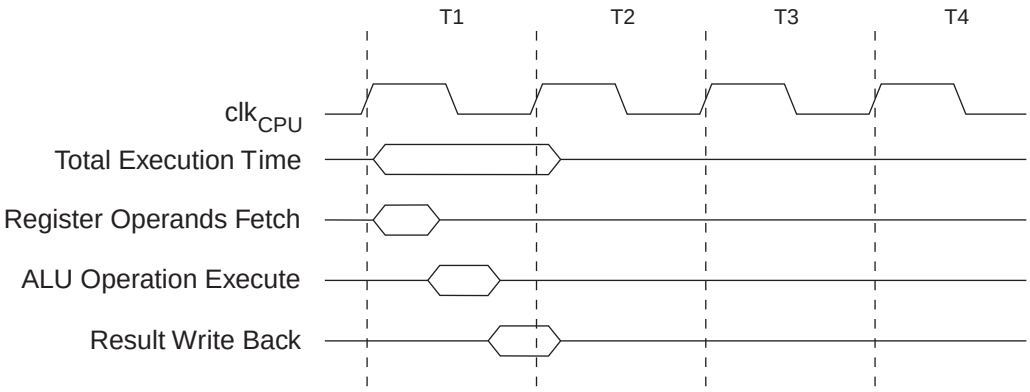


Figure 3-3 shows the internal timing concept for the register file. In a single clock cycle, an ALU operation using two register operands is executed and the result is stored back to the destination register.

Figure 3-3. Single cycle ALU operation.



3.7 Status register

The status register (SREG) contains information about the result of the most recently executed arithmetic or logic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the status register is updated after all ALU operations, as specified in the “Instruction Set Summary” on page 424. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The status register is not automatically stored when entering an interrupt routine nor restored when returning from an interrupt. This must be handled by software.

The status register is accessible in the I/O memory space.

3.8 Stack and stack pointer

The stack is used for storing return addresses after interrupts and subroutine calls. It can also be used for storing temporary data. The stack pointer (SP) register always points to the top of the stack. It is implemented as two 8-bit registers that are accessible in the I/O memory space. Data are pushed and popped from the stack using the PUSH and POP instructions. The stack grows from a higher memory location to a lower memory location. This implies that pushing data onto the stack decreases the SP, and popping data off the stack increases the SP. The SP is automatically loaded after reset, and the initial value is the highest address of the internal SRAM. If the SP is changed, it must be set to point above address 0x2000, and it must be defined before any subroutine calls are executed or before interrupts are enabled.

During interrupts or subroutine calls, the return address is automatically pushed on the stack. The return address can be two or three bytes, depending on program memory size of the device. For devices with 128KB or less of program memory, the return address is two bytes, and hence the stack pointer is decremented/incremented by two. For devices with more than 128KB of program memory, the return address is three bytes, and hence the SP is decremented/incremented by three. The return address is popped off the stack when returning from interrupts using the RETI instruction, and from subroutine calls using the RET instruction.

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

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

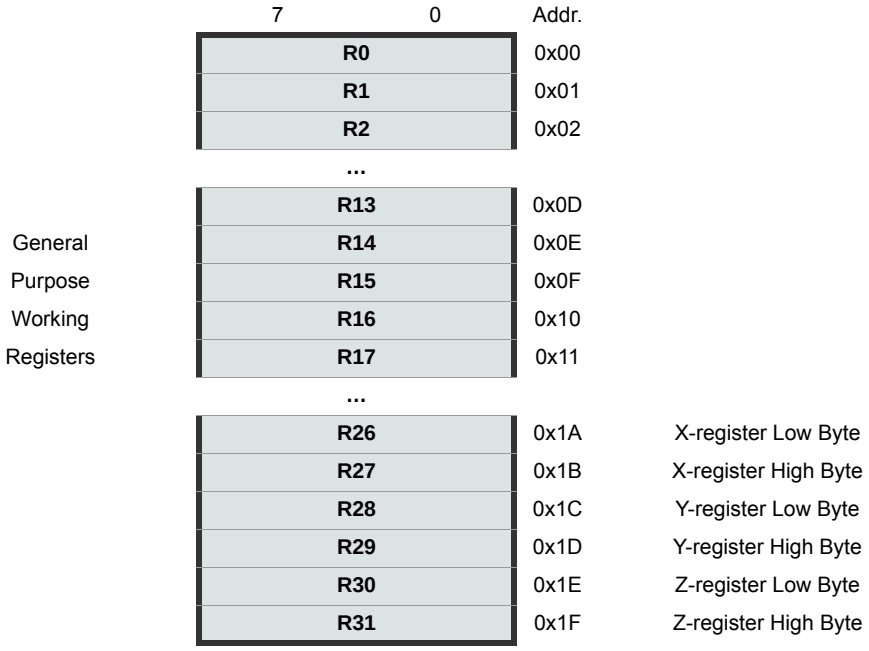
3.9 Register file

The register file consists of 32 x 8-bit general purpose working registers with single clock cycle access time. The register file supports the following input/output schemes:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Six of the 32 registers can be used as three 16-bit address register pointers for data space addressing, enabling efficient address calculations. One of these address pointers can also be used as an address pointer for lookup tables in flash program memory.

Figure 3-4. AVR CPU general purpose working registers.



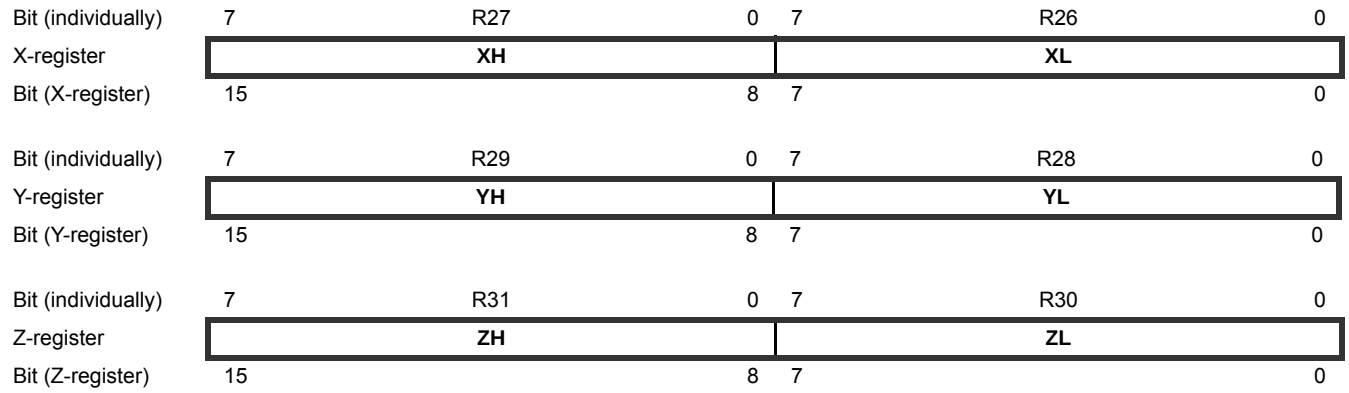
The register file is located in a separate address space, and so the registers are not accessible as data memory.

3.9.1 The X-, Y-, and Z- registers

Registers R26..R31 have added functions besides their general-purpose usage.

These registers can form 16-bit address pointers for addressing data memory. These three address registers are called the X-register, Y-register, and Z-register. The Z-register can also be used as an address pointer to read from and/or write to the flash program memory, signature rows, fuses, and lock bits.

Figure 3-5. The X-, Y- and Z-registers.



The lowest register address holds the least-significant byte (LSB), and the highest register address holds the most-significant byte (MSB). In the different addressing modes, these address registers function as fixed displacement, automatic increment, and automatic decrement (see the “Instruction Set Summary” on page 424 for details).

3.10 RAMP and extended indirect registers

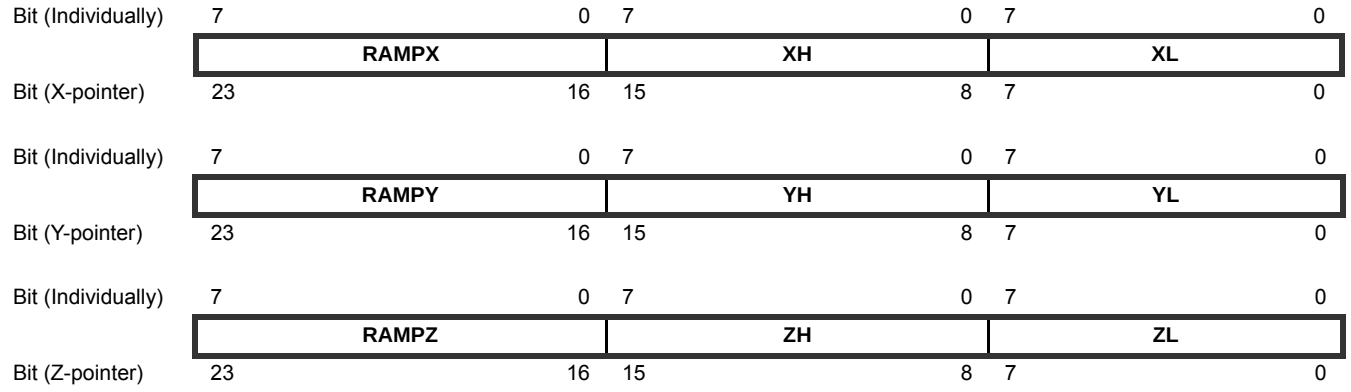
In order to access program memory or data memory above 64KB, the address pointer must be larger than 16 bits. This is done by concatenating one register to one of the X-, Y-, or Z-registers. This register then holds the most-significant byte (MSB) in a 24-bit address or address pointer.

These registers are available only on devices with external bus interface and/or more than 64KB of program or data memory space. For these devices, only the number of bits required to address the whole program and data memory space in the device is implemented in the registers.

3.10.1 RAMPX, RAMPY and RAMPZ registers

The RAMPX, RAMPY and RAMPZ registers are concatenated with the X-, Y-, and Z-registers, respectively, to enable indirect addressing of the whole data memory space above 64KB and up to 16MB.

Figure 3-6. The combined RAMPX + X, RAMPY + Y and RAMPZ + Z registers.



When reading (ELPM) and writing (SPM) program memory locations above the first 128KB of the program memory, RAMPZ is concatenated with the Z-register to form the 24-bit address. LPM is not affected by the RAMPZ setting.

3.10.2 RAMPD register

This register is concatenated with the operand to enable direct addressing of the whole data memory space above 64KB. Together, RAMPD and the operand will form a 24-bit address.

Figure 3-7. The combined RAMPD + K register.

3.10.3 EIND - Extended Indirect register

EIND is concatenated with the Z-register to enable indirect jump and call to locations above the first 128KB (64K words) of the program memory.

Figure 3-8. The combined EIND + Z register.

3.11 Accessing 16-bit registers

The AVR data bus is 8 bits wide, and so accessing 16-bit registers requires atomic operations. These registers must be byte-accessed using two read or write operations. 16-bit registers are connected to the 8-bit bus and a temporary register using a 16-bit bus.

For a write operation, the low byte of the 16-bit register must be written before the high byte. The low byte is then written into the temporary register. When the high byte of the 16-bit register is written, the temporary register is copied into the low byte of the 16-bit register in the same clock cycle.

For a read operation, the low byte of the 16-bit register must be read before the high byte. When the low byte register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read. When the high byte is read, it is then read from the temporary register.

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

Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

The temporary registers can also be read and written directly from user software.

3.11.1 Accessing 24- and 32-bit registers

For 24- and 32-bit registers, the read and write access is done in the same way as described for 16-bit registers, except there are two temporary registers for 24-bit registers and three for 32-bit registers. The least-significant byte must be written first when doing a write, and read first when doing a read.

3.12 Configuration change protection

System critical I/O register settings are protected from accidental modification. The SPM instruction is protected from accidental execution, and the LPM instruction is protected when reading the fuses and signature row. 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 described in the register description.

There are two modes of operation: one for protected I/O registers, and one for the protected instructions, SPM/LPM.

3.12.1 Sequence for write operation to protected I/O registers

1. The application code writes the signature that enable change of protected I/O registers to the CCP register.
2. Within four instruction cycles, the application code must write the appropriate data to the protected register. Most protected registers also contain a write enable/change enable bit. This bit must be written to one 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 or if the SPM, LPM, or SLEEP instruction is executed.

3.12.2 Sequence for execution of protected SPM/LPM

1. The application code writes the signature for the execution of protected SPM/LPM to the CCP register.
2. Within four instruction cycles, the application code must execute the appropriate instruction. The protected change is immediately disabled if the CPU performs write operations to the data memory or if the SLEEP instruction is executed.

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. EDMA requests are still handled, but do not influence the protected configuration change enable period. A signature written by EDMA is ignored.

3.13 Fuse lock

For some system-critical features, it is possible to program a fuse to disable all changes to the associated I/O control registers. If this is done, it will not be possible to change the registers from the user software, and the fuse can only be reprogrammed using an external programmer. Details on this are described in the datasheet module where this feature is available.

3.14 Register descriptions

3.14.1 CCP – Configuration Change Protection register

Bit	7	6	5	4	3	2	1	0
+0x04	CCP[7:0]							
Read/Write	W	W	W	W	W	W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCP[7:0]: Configuration Change Protection Bits**
  The CCP register must be written with the correct signature to enable change of the protected I/O register or execution of the protected instruction for a maximum period of four CPU instruction cycles. All interrupts are ignored during these cycles. After these cycles, interrupts will automatically be handled again by the CPU, and any pending interrupts will be executed according to their level and priority. When the protected I/O register signature is written, CCP[0] will read as one as long as the protected feature is enabled. Similarly when the protected SPM/LPM signature is written, CCP[1] will read as one as long as the protected feature is enabled. CCP[7:2] will always read as zero. Table 3-1 on page 15 shows the signature for the various modes.

Table 3-1. Modes of CPU change protection.

Signature	Group configuration	Description
0x9D	SPM	Protected SPM/LPM
0xD8	IOREG	Protected IO register

3.14.2 RAMPD – Extended Direct Addressing register

This register is concatenated with the operand for direct addressing (LDS/STS) of the whole data memory space on devices with more than 64KB of data memory. This register is not available if the data memory, including external memory, is less than 64KB.

Bit	7	6	5	4	3	2	1	0
+0x08	RAMPD[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RAMPD[7:0]: Extended Direct Addressing Bits**
  These bits hold the MSB of the 24-bit address created by RAMPD and the 16-bit operand. Only the number of bits required to address the available data memory is implemented for each device. Unused bits will always read as zero.

3.14.3 RAMPX – Extended X-Pointer register

This register is concatenated with the X-register for indirect addressing (LD/LDD/ST/STD) of the whole data memory space on devices with more than 64KB of data memory. This register is not available if the data memory, including external memory, is less than 64KB.

Bit	7	6	5	4	3	2	1	0
+0x09	RAMPX[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RAMPX[7:0]: Extended X-pointer Address Bits**
  These bits hold the MSB of the 24-bit address created by RAMPX and the 16-bit X-register. Only the number of bits required to address the available data memory is implemented for each device. Unused bits will always read as zero.

3.14.4 RAMPY – Extended Y-Pointer register

This register is concatenated with the Y-register for indirect addressing (LD/LDD/ST/STD) of the whole data memory space on devices with more than 64KB of data memory. This register is not available if the data memory, including external memory, is less than 64KB.

Bit	7	6	5	4	3	2	1	0
+0x0A	RAMPY[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RAMPY[7:0]: Extended Y-pointer Address Bits**
  These bits hold the MSB of the 24-bit address created by RAMPY and the 16-bit Y-register. Only the number of bits required to address the available data memory is implemented for each device. Unused bits will always read as zero.

3.14.5 RAMPZ – Extended Z-Pointer register

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

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

Bit	7	6	5	4	3	2	1	0
+0x0B	RAMPZ[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

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

3.14.6 EIND – Extended Indirect register

This register is concatenated with the Z-register for enabling extended indirect jump (EIJMP) and call (EICALL) to the whole program memory space on devices with more than 128KB of program memory. The register should be used for jumps to addresses below 128KB if ECALL/EIJMP are used, and it will not be used if CALL and IJMP commands are used. For jump or call to addresses below 128KB, this register is not used. This register is not available if the program memory in the device is less than 128KB.

Bit	7	6	5	4	3	2	1	0
+0x0C	EIND[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – EIND[7:0]: Extended Indirect Address Bits**
  These bits hold the MSB of the 24-bit address created by EIND and the 16-bit Z-register. Only the number of bits required to access the available program memory is implemented for each device. Unused bits will always read as zero.

3.14.7 SPL – Stack Pointer register Low

The SPH and SPL stack pointer pair represent the 16-bit SP value. The SP holds the stack pointer that points to the top of the stack. After reset, the stack pointer points to the highest internal SRAM address. To prevent corruption when updating the stack pointer from software, a write to SPL will automatically disable interrupts for the next four instructions or until the next I/O memory write.

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

Bit	7	6	5	4	3	2	1	0
+0x0D	SP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value(1)	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Note: 1. Refer to specific device datasheets for exact size.

- **Bit 7:0 – SP[7:0]: Stack Pointer Low Byte**
  These bits hold the LSB of the 16-bit stack pointer (SP).

3.14.8 SPH – Stack Pointer register High

Bit	7	6	5	4	3	2	1	0
+0x0E	SP[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value(1)	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Note: 1. Refer to specific device datasheets for the exact size.

- **Bit 7:0 – SP[15:8]: Stack Pointer High Byte**
  These bits hold the MSB of the 16-bit stack pointer (SP).

3.14.9 SREG – Status register

The status register (SREG) contains information about the result of the most recently executed arithmetic or logic instruction. For details information about the bits in this register and how they are affected by the different instructions see “Instruction Set Summary” on page 424.

Bit	7	6	5	4	3	2	1	0
+0x0F	I	T	H	S	V	N	Z	C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – I: Global Interrupt Enable**
  The global interrupt enable bit must be set for interrupts to be enabled. If the global interrupt enable register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. This bit is not cleared by hardware after an interrupt has occurred. This bit can be set and cleared by the application with the SEI and CLI instructions, as described in “Instruction Set Summary” on page 424. Changing the I flag through the I/O-register result in a one-cycle wait state on the access.

- **Bit 6 – T: Bit Copy Storage**
  The bit copy instructions bit load (BLD) and bit store (BST) use the T bit as source or destination for the operated bit. A bit from a register in the register file can be copied into this bit by the BST instruction, and this bit can be copied into a bit in a register in the register file by the BLD instruction.
- **Bit 5 – H: Half Carry Flag**
  The half carry flag (H) indicates a half carry in some arithmetic operations. Half carry is useful in BCD arithmetic.
- **Bit 4 – S: Sign Bit, S = N ⊕ V**
  The sign bit is always an exclusive or between the negative flag, N, and the two's complement overflow flag, V.
- **Bit 3 – V: Two's Complement Overflow Flag**
  The two's complement overflow flag (V) supports two's complement arithmetic.
- **Bit 2 – N: Negative Flag**
  The negative flag (N) indicates a negative result in an arithmetic or logic operation.
- **Bit 1 – Z: Zero Flag**
  The zero flag (Z) indicates a zero result in an arithmetic or logic operation.
- **Bit 0 – C: Carry Flag**
  The carry flag (C) indicates a carry in an arithmetic or logic operation.

3.15 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	Reserved	–	–	–	–	–	–	–	–	
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	Reserved	–	–	–	–	–	–	–	–	
+0x04	CCP	CCP[7:0]								15
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	RAMPD	RAMPD[7:0]								15
+0x09	RAMPX	RAMPX[7:0]								15
+0x0A	RAMPY	RAMPY[7:0]								16
+0x0B	RAMPZ	RAMPZ[7:0]								16
+0x0C	EIND	EIND[7:0]								16
+0x0D	SPL	SPL[7:0]								17
+0x0E	SPH	SPH[7:0]								17
+0x0F	SREG	I	T	H	S	V	N	Z	C	17

4. Memories

4.1 Features

- Flash program memory
  - One linear address space
  - In-system programmable
  - Self-programming and boot loader support
  - Application section for application code
  - Application table section for application code or data storage
  - Boot section for application code or bootloader code
  - Separate read/write protection lock bits for all sections
  - Built in fast CRC check of a selectable flash program memory section
- Data memory
  - One linear address space
  - Single-cycle access from CPU
  - SRAM
  - EEPROM
    - Byte and page accessible
    - Memory mapped for direct load and store
  - I/O memory
    - Configuration and status registers for all peripherals and modules
    - 4 bit-accessible general purpose registers for global variables or flags
  - Bus arbitration
    - Deterministic handling of priority between CPU, EDMA controller, and other bus masters
  - Separate buses for SRAM, EEPROM, I/O memory, and external memory access
    - Simultaneous bus access for CPU and EDMA controller
- Production signature row memory for factory programmed data
  - ID for each microcontroller device type
  - Serial number for each device
  - Calibration bytes for factory calibrated peripherals
- User signature row
  - One flash page in size
  - Can be read and written from software
  - Content is kept after chip erase

4.2 Overview

This section describes the different memory sections. The AVR architecture has two main memory spaces, the program memory and the data memory. Executable code can reside only in the program memory, while data can be stored in the program memory and the data memory. The data memory includes the internal SRAM, and EEPROM for nonvolatile data storage. All memory spaces are linear and require no memory bank switching. Nonvolatile memory (NVM) spaces can be locked for further write and read/write operations. This prevents unrestricted access to the application software.

A separate memory section contains the fuse bytes. These are used for configuring important system functions, and can only be written by an external programmer.

4.3 Flash program memory

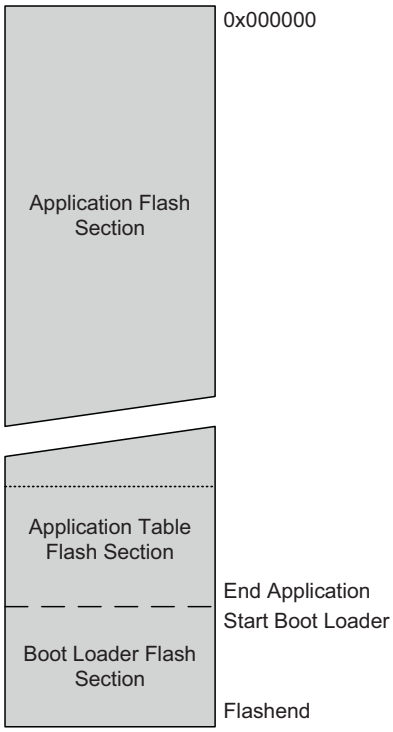
All XMEGA devices contain on-chip, in-system reprogrammable flash memory for program storage. The flash memory can be accessed for read and write from an external programmer through the PDI or from application software running in the device.

All AVR CPU instructions are 16 or 32 bits wide, and each flash location is 16 bits wide. The flash memory is organized in two main sections, the application section and the boot loader section, as shown in Figure 4-1 on page 21. The sizes

of the different sections are fixed, but device-dependent. These two sections have separate lock bits, and can have different levels of protection. The store program memory (SPM) instruction, used to write to the flash from the application software, will only operate when executed from the boot loader section.

The application section contains an application table section with separate lock settings. This enables safe storage of nonvolatile data in the program memory.

Figure 4-1. Flash memory sections.



4.3.1 Application section

The Application section is the section of the flash that is used for storing the executable application code. The protection level for the application section can be selected by the boot lock bits for this section. The application section can not store any boot loader code since the SPM instruction cannot be executed from the application section.

4.3.2 Application table section

The application table section is a part of the application section of the flash memory that can be used for storing data. The size is identical to the boot loader section. The protection level for the application table section can be selected by the boot lock bits for this section. The possibilities for different protection levels on the application section and the application table section enable safe parameter storage in the program memory. If this section is not used for data, application code can reside here.

4.3.3 Boot loader section

While the application section is used for storing the application code, the boot loader software must be located in the boot loader section because the SPM instruction can initiate programming when executing from this section. When programming, the CPU is halted, waiting for the flash operation to complete. The SPM instruction can access the entire flash, including the boot loader section itself. The protection level for the boot loader section can be selected by the boot loader lock bits. If this section is not used for boot loader software, application code can be stored here.

4.3.4 Production signature row

The production signature row is a separate memory section for factory programmed data. It contains calibration data for functions such as oscillators and analog modules. Some of the calibration values will be automatically loaded to the corresponding module or peripheral unit during reset. Other values must be loaded from the signature row and written to the corresponding peripheral registers from software. For details on calibration conditions such as temperature, voltage references, etc., refer to the device datasheet.

The production signature row also contains an ID that identifies each microcontroller device type and a serial number for each manufactured device. The serial number consists of the production lot number, wafer number, and wafer coordinates for the device.

The production signature row cannot be written or erased, but it can be read from application software and external programmers.

4.3.5 User signature row

The user signature row is a separate memory section that is fully accessible (read and write) from application software and external programmers. It is one flash page in size, and is meant for static user parameter storage, such as calibration data, custom serial number, identification numbers, random number seeds, etc. This section is not erased by chip erase commands that erase the flash, and requires a dedicated erase command. This ensures parameter storage during multiple program/erase operations and on-chip debug sessions.

4.4 Fuses and lockbits

The fuses are used to configure important system functions, and can only be written from an external programmer. The application software can read the fuses. The fuses are used to configure the startup configuration and reset sources such as brownout detector and watchdog.

The lock bits are used to set protection levels for the different flash sections (i.e., if read and/or write access should be blocked). Lock bits can be written by external programmers and application software, but only to stricter protection levels. Chip erase is the only way to erase the lock bits. To ensure that flash contents are protected even during chip erase, the lock bits are erased after the rest of the flash memory has been erased.

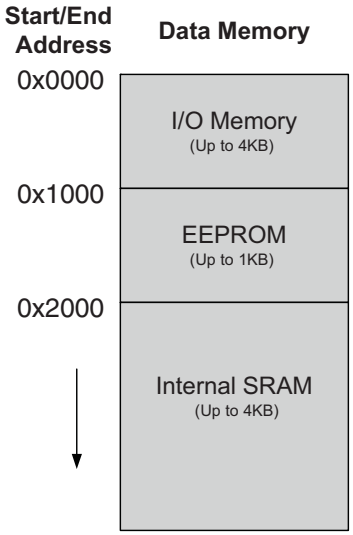
An unprogrammed fuse or lock bit will have the value one, while a programmed fuse or lock bit will have the value zero.

Both fuses and lock bits are reprogrammable like the flash program memory.

4.5 Data memory

The data memory contains the I/O memory, internal SRAM, EEPROM, and external memory, if available. The data memory is organized as one continuous memory section, as shown in Figure 4-2 on page 23.

Figure 4-2. Data memory map.



I/O memory, EEPROM, and SRAM will always have the same start addresses for all XMEGA devices.

4.6 Internal SRAM

The internal SRAM always starts at hexadecimal address 0x2000. SRAM is accessed by the CPU using the load (LD/LDS/LDD) and store (ST/STS/STD) instructions.

4.7 EEPROM

All XMEGA devices have EEPROM for nonvolatile data storage. It is addressable in a separate memory mapped space and accessed in normal data space. The EEPROM supports both byte and page access. EEPROM is accessible using load and store instructions, allowing highly efficient EEPROM reading and EEPROM buffer loading. EEPROM always starts at the hexadecimal address 0x1000.

4.8 I/O memory

The status and configuration registers for peripherals and modules, including the CPU, are addressable through I/O memory locations. All I/O locations can be accessed by the load (LD/LDS/LDD) and store (ST/STS/STD) instructions, which are used to transfer data between the 32 registers in the register file and the I/O memory. The IN and OUT instructions can address I/O memory locations in the range of 0x00 to 0x3F directly. In the address range 0x00 - 0x1F, single-cycle instructions for manipulation and checking of individual bits are available.

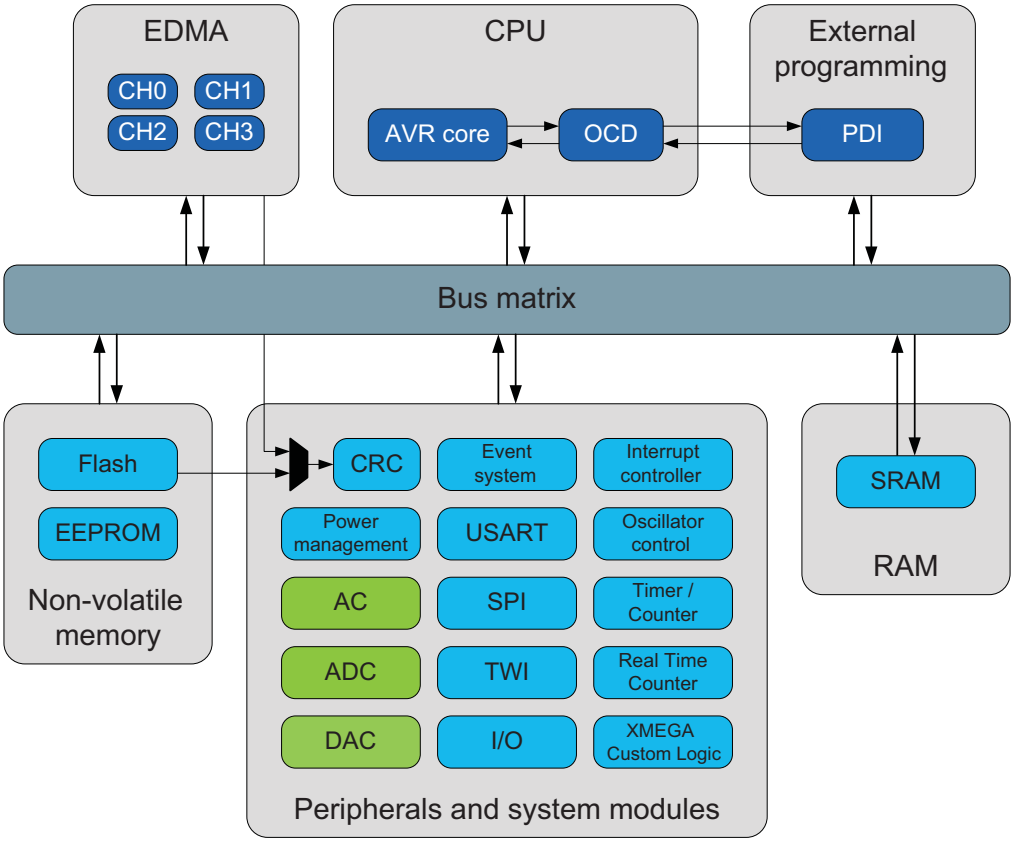
4.8.1 General purpose I/O registers

The lowest 4 I/O memory addresses are reserved as general purpose I/O registers. These registers can be used for storing global variables and flags, as they are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

4.9 Data memory and bus arbitration

Since the data memory is organized as three separate sets of memories, the different bus masters (CPU, EDMA controller read and EDMA controller write, etc.) can access different memory sections at the same time. See Figure 4-3 on page 24.

Figure 4-3. Bus access.



4.9.1 Bus priority

When several masters request access to the same bus, the bus priority is in the following order (from higher to lower priority):

1. Bus Master with ongoing access.
2. Bus Master with ongoing burst.
    1. Alternating EDMA controller read and EDMA controller write when they access the same data memory section.
3. Bus Master requesting burst access.
    1. CPU has priority.
4. Bus Master requesting bus access.
    1. CPU has priority.

4.10 Memory timing

Read and write access to the I/O memory takes one CPU clock cycle. A write to SRAM takes one cycle, and a read from SRAM takes two cycles. For burst read (EDMA), new data are available every cycle. EEPROM page load (write) takes one cycle, and three cycles are required for read. For burst read, new data are available every second cycle. Refer to the instruction summary for more details on instructions and instruction timing.

4.11 Device ID and revision

Each device has a three-byte device ID. This ID identifies Atmel as the manufacturer of the device and the device type. A separate register contains the revision number of the device.

4.12 I/O memory protection

Some features in the device are regarded as critical for safety in some applications. Due to this, it is possible to lock the I/O register related to the clock system, the event system, and the advanced waveform extensions. As long as the lock is enabled, all related I/O registers are locked and they can not be written from the application software. The lock registers themselves are protected by the configuration change protection mechanism. For details, refer to “Configuration change protection” on page 13.

4.13 Register description – NVM controller

4.13.1 ADDR0 – Address register 0

The ADDR0, ADDR1, and ADDR2 registers represent the 24-bit value, ADDR. This is used for addressing all NVM sections for read, write, and CRC operations.

Bit	7	6	5	4	3	2	1	0
+0x00	ADDR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

- **Bit 7:0 – ADDR[7:0]: Address Byte 0**
  This register gives the address low byte when accessing NVM locations.

4.13.2 ADDR1 – Address register 1

Bit	7	6	5	4	3	2	1	0
+0x01	ADDR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – ADDR[15:8]: Address Byte 1**
  This register gives the address high byte when accessing NVM locations.

4.13.3 ADDR2 – Address register 2

Bit	7	6	5	4	3	2	1	0
+0x02	ADDR[23:16]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – ADDR[23:16]: Address Byte 2**
  This register gives the address extended byte when accessing NVM locations.

4.13.4 DATA0 – Data register 0

The DATA0, DATA1, and DATA registers represent the 24-bit value, DATA. This holds data during NVM read, write, and CRC access.

Bit	7	6	5	4	3	2	1	0
+0x04	DATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DATA[7:0]: Data Byte 0**
  This register gives the data value byte 0 when accessing NVM locations.

4.13.5 DATA1 – Data register 1

Bit	7	6	5	4	3	2	1	0
+0x05	DATA[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DATA[15:8]: Data Byte 1**
  This register gives the data value byte 1 when accessing NVM locations.

4.13.6 DATA2 – Data register 2

Bit	7	6	5	4	3	2	1	0
+0x06	DATA[23:16]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DATA[23:16]: Data Byte 2**
  This register gives the data value byte 2 when accessing NVM locations.

4.13.7 CMD – Command register

Bit	7	6	5	4	3	2	1	0
+0x0A	–	CMD[6:0]						
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6:0 – CMD[6:0]: Command**
  These bits define the programming commands for the flash. Bit 6 is only set for external programming commands. See “Memory Programming” on page 403 for programming commands.

4.13.8 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x0B	–	–	–	–	–	–	–	CMDEX
Read/Write	R	R	R	R	R	R	R	S
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – CMDEX: Command Execute**
  Setting this bit will execute the command in the CMD register. This bit is protected by the configuration change protection (CCP) mechanism. Refer to “Configuration change protection” on page 13 for details on the CCP.

4.13.9 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x0C	–	–	–	–	–	–	EPRM	SPMLOCK
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – EPRM: EEPROM Power Reduction Mode**
  Setting this bit enables power saving for the EEPROM. The EEPROM will then be turned off in a manner equivalent to entering sleep mode. If access is required, the bus master will be halted for a time equal to the start-up time from idle sleep mode.
- **Bit 0 – SPMLOCK: SPM Locked**
  This bit can be written to prevent all further self-programming. The bit is cleared at reset, and cannot be cleared from software. This bit is protected by the configuration change protection (CCP) mechanism. Refer to "Configuration change protection" on page 13 for details on the CCP.

4.13.10 INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x0D	–	–	–	–	SPMLVL[1:0]		EELVL[1:0]	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:2 – SPMLVL[1:0]: SPM Ready Interrupt Level**
  These bits enable the interrupt and select the interrupt level, as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132. This is a level interrupt that will be triggered only when the NVMBUSY flag in the STATUS register is set to zero. Thus, the interrupt should not be enabled before triggering an NVM command, as the NVMBUSY flag will not be set before the NVM command is triggered. The interrupt should be disabled in the interrupt handler.
- **Bit 1:0 – EELVL[1:0]: EEPROM Ready Interrupt Level**
  These bits enable the EEPROM ready interrupt and select the interrupt level, as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132. This is a level interrupt that will be triggered only when the NVMBUSY flag in the STATUS register is set to zero. Thus, the interrupt should not be enabled before triggering an NVM command, as the NVMNVMBUSY flag will not be set before the NVM command is triggered. The interrupt should be disabled in the interrupt handler.

4.13.11 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x0F	NVMBUSY	FBUSY	–	–	–	–	EELOAD	FLOAD
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – NVMBUSY: Nonvolatile Memory Busy**
  The NVMBUSY flag indicates if the NVM (Flash, EEPROM, lock bit) is being programmed. Once an operation is started, this flag is set and remains set until the operation is completed. The NVMBUSY flag is automatically cleared when the operation is finished.
- **Bit 6 – FBUSY: Flash Busy**
  The FBUSY flag indicates if a flash programming operation is initiated. Once an operation is started, the FBUSY flag is set and the application section cannot be accessed. The FBUSY flag is automatically cleared when the operation is finished.
- **Bit 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – EELOAD: EEPROM Page Buffer Active Loading**
  The EELOAD flag indicates that the temporary EEPROM page buffer has been loaded with one or more data bytes. It remains set until an EEPROM page write or a page buffer flush operation is executed. For more details, see “Flash and EEPROM programming sequences” on page 405.
- **Bit 0 – FLOAD: Flash Page Buffer Active Loading**
  The FLOAD flag indicates that the temporary flash page buffer has been loaded with one or more data bytes. It remains set until an application page write, boot page write, or page buffer flush operation is executed. For more details, see “Flash and EEPROM programming sequences” on page 405.

4.13.12 LOCKBITS – Lock Bit register

Bit	7	6	5	4	3	2	1	0
+0x10	BLBB[1:0]		BLBA[1:0]		BLBAT[1:0]		LB[1:0]	
Read/Write	R	R	R	R	R	R	R	R
Initial Value	1	1	1	1	1	1	1	1

This register is a mapping of the NVM lock bits into the I/O memory space which enables direct read access from the application software. Refer to “LOCKBITS – Lock Bit register” on page 33 for a description.

4.14 Register descriptions – Fuses and lock bits

4.14.1 FUSEBYTE1 – Fuse Byte 1

Bit	7	6	5	4	3	2	1	0
+0x01	WDWPER[3:0]				WDPER[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – WDWPER[3:0]: Watchdog Window Timeout Period**
  These fuse bits are used to set initial value of the closed window for the Watchdog Timer in Window Mode. During reset these fuse bits are automatically written to the WPER bits Watchdog Window Mode Control Register. Refer to “WINCTRL – Window Mode Control register” on page 130 for details.
- **Bit 3:0 – WDPER[3:0]: Watchdog Timeout Period**
  These fuse bits are used to set the initial value of the watchdog timeout period. During reset, these fuse bits are automatically written to the PER bits in the watchdog control register. Refer to “CTRL – Control register” on page 129 for details.

4.14.2 FUSEBYTE2 – Fuse Byte 2

Bit	7	6	5	4	3	2	1	0
+0x02	–	BOOTRST	–	–	–	–	BODPD[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to one when this register is written.
- **Bit 6 – BOOTRST: Boot Loader Section Reset Vector**
  This fuse can be programmed so the reset vector is pointing to the first address in the boot loader flash section. The device will then start executing from the boot loader flash section after reset.

Table 4-1. Boot reset fuse.

BOOTRST	Reset address
0	Reset vector = Boot loader reset
1	Reset vector = Application reset (address 0x0000)

- **Bit 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to one when this register is written.
- **Bit 1:0 – BODPD[1:0]: BOD Operation in Power-down Mode**
  These fuse bits set the BOD operation mode in all sleep modes except idle mode.
  For details on the BOD and BOD operation modes, refer to “Brownout detection” on page 122.

Table 4-2. BOD operation modes in sleep modes.

BODPD[1:0]	Description
00	Reserved
01	BOD enabled in sampled mode
10	BOD enabled continuously
11	BOD disabled

4.14.3 FUSEBYTE4 – Fuse Byte 4

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	RSTDISBL	STARTUPTIME[1:0]		WDLOCK	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to one when this register is written.
- **Bit: 4 – RSTDISBL: External Reset Disable**
  This fuse can be programmed to disable the external reset pin functionality. When this is done, pulling the reset pin low will not cause an external reset. A reset is required before this bit will be read correctly after it is changed.
- **Bit 3:2 – STARTUPTIME[1:0]: Start-up time**
  These fuse bits can be used to set at a programmable timeout period from when all reset sources are released until the internal reset is released from the delay counter. A reset is required before these bits will be read correctly after they are changed.
  The delay is timed from the 1kHz output of the ULP oscillator. Refer to “Reset sequence” on page 121 for details.

Table 4-3. Start-up time.

STARTUPTIME[1:0]	1kHz ULP oscillator cycles
00	64
01	4
10	Reserved
11	0

- **Bit 1 – WDLOCK: Watchdog Timer Lock**
  The WDLOCK fuse can be programmed to lock the watchdog timer configuration. When this fuse is programmed, the watchdog timer configuration cannot be changed, and the ENABLE bit in the watchdog CTRL register is automatically set at reset and cannot be cleared from the application software. The WEN bit in the watchdog WINCTRL register is not set automatically, and needs to be set from software. A reset is required before this bit will be read correctly after it is changed.

Table 4-4. Watchdog timer lock.

WDLOCK	Description
0	Watchdog timer locked for modifications
1	Watchdog timer not locked

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to one when this register is written.

4.14.4 FUSEBYTE5 – Fuse Byte 5

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	BODACT[1:0]		EESAVE	BODLEVEL[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	–	–	–	–	–	–

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to one when this register is written.
- **Bit 5:4 – BODACT[1:0]: BOD Operation in Active Mode**
  These fuse bits set the BOD operation mode when the device is in active and idle modes. For details on the BOD and BOD operation modes, refer to “Brownout detection” on page 122.

Table 4-5. BOD operation modes in active and idle modes.

BODACT[1:0]	Description
00	Reserved
01	BOD enabled in sampled mode
10	BOD enabled continuously
11	BOD disabled

- **Bit 3 – EESAVE: EEPROM Preserved Through The Chip Erase**
  A chip erase command will normally erase the flash, EEPROM, and internal SRAM. If this fuse is programmed, the EEPROM is not erased during chip erase. This is useful if EEPROM is used to store data independently of the software revision.

Table 4-6. EEPROM preserved through chip erase.

EESAVE	Description
0	EEPROM is preserved during chip erase
1	EEPROM is erased during chip erase

Changes to the EESAVE fuse bit take effect immediately after the write timeout elapses. Hence, it is possible to update EESAVE and perform a chip erase according to the new setting of EESAVE without leaving and reentering programming mode.

- **Bit 2:0 – BODLEVEL[2:0]: Brownout Detection Voltage Level**
  These fuse bits sets the BOD voltage level. Refer to “Reset sequence” on page 121 for details. For BOD level nominal values, see Table 9-2 on page 123.

4.14.5 FUSEBYTE6 – Fuse Byte 6

Bit	7	6	5	4	3	2	1	0
+0x06	FDACT5	FDACT4	VALUE[5:0]					
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

- **Bit 7 – FDACT5: Fault Detection Action Timer/Counter 5**
  This fuse sets the fault detection action on Px4 and Px5 port pins, which are the default output pins for timer/counter 5 output compare channels. Table 4-7 on page 33 shows the possible settings.
- **Bit 6 – FDACT4: Fault Detection Action Timer/Counter 4**
  This fuse sets the fault detection action on Px0 to Px3 port pins, which are the default output pins for timer/counter 4 output compare channels. Table 4-7 on page 33 shows the possible settings.

Table 4-7. Fault detection action.

FDACT	Description
0	In reset state and until a timer/counter compare channel is enabled, the port pins are forced to the value set in the corresponding VALUEn fuse.
1	Default I/O pin configuration.

- **Bit 5:0 – VALUE[5:0]: Port Pin n Value**
  These fuses select the value that will be output on the corresponding port pin when an emergency fault occurs and if the corresponding FDACT fuse is set.

Table 4-8. Port pin value.

VALUEn	Description
0	The corresponding port pin output value is set to 0 (low level).
1	The corresponding port pin output value is set to 1 (high level).

4.14.6 LOCKBITS – Lock Bit register

Bit	7	6	5	4	3	2	1	0
+0x07	BLBB[1:0]		BLBA[1:0]		BLBAT[1:0]		LB[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

- **Bit 7:6 – BLBB[1:0]: Boot Lock Bit Boot Loader Section**
  These lock bits control the software security level for accessing the boot loader section. The BLBB bits can only be written to a more strict locking. Resetting the BLBB bits is possible only by executing a chip erase command.

Table 4-9. Boot lock bit for the boot loader section.

BLBB[1:0]	Group configuration	Description
11	NOLOCK	No lock – no restrictions for SPM and (E)LPM accessing the boot loader section.
10	WLOCK	Write lock – SPM is not allowed to write the boot loader section.
01	RLOCK	Read lock – (E)LPM executing from the application section is not allowed to read from the boot loader section. If the interrupt vectors are placed in the application section, interrupts are disabled while executing from the boot loader section.
00	RWLOCK	Read and write lock – SPM is not allowed to write to the boot loader section, and (E)LPM executing from the application section is not allowed to read from the boot loader section. If the interrupt vectors are placed in the application section, interrupts are disabled while executing from the boot loader section.

- **Bit 5:4 – BLBA[1:0]: Boot Lock Bit Application Section**
  These lock bits control the software security level for accessing the application section. The BLBA bits can only be written to a more strict locking. Resetting the BLBA bits is possible only by executing a chip erase command.

Table 4-10. Boot lock bit for the application section.

BLBA[1:0]	Group configuration	Description
11	NOLOCK	No Lock - no restrictions for SPM and (E)LPM accessing the application section.
10	WLOCK	Write lock – SPM is not allowed to write the application section.
01	RLOCK	Read lock – (E)LPM executing from the boot loader section is not allowed to read from the application section. If the interrupt vectors are placed in the boot loader section, interrupts are disabled while executing from the application section.
00	RWLOCK	Read and write lock – SPM is not allowed to write to the application section, and (E)LPM executing from the boot loader section is not allowed to read from the application section. If the interrupt vectors are placed in the boot loader section, interrupts are disabled while executing from the application section.

- **Bit 3:2 – BLBAT[1:0]: Boot Lock Bit Application Table Section**
  These lock bits control the software security level for accessing the application table section for software access. The BLBAT bits can only be written to a more strict locking. Resetting the BLBAT bits is possible only by executing a chip erase command

Table 4-11. Boot lock bit for the application table section.

BLBAT[1:0]	Group configuration	Description
11	NOLOCK	No lock – no restrictions for SPM and (E)LPM accessing the application table section.
10	WLOCK	Write lock – SPM is not allowed to write the application table
01	RLOCK	Read lock – (E)LPM executing from the boot loader section is not allowed to read from the application table section. If the interrupt vectors are placed in the boot loader section, interrupts are disabled while executing from the application section.
00	RWLOCK	Read and write lock – SPM is not allowed to write to the application table section, and (E)LPM executing from the boot loader section is not allowed to read from the application table section. If the interrupt vectors are placed in the boot loader section, interrupts are disabled while executing from the application section.

- **Bit 1:0 – LB[1:0]: Lock Bits**[1]
  These lock bits control the security level for the flash and EEPROM during external programming. These bits are writable only through an external programming interface. Resetting the lock bits is possible only by executing a chip erase command. All other access; using the TIF and OCD, is blocked if any of the Lock Bits are written to 0. These bits do not block any software access to the memory

Table 4-12. Lock bit protection mode.

LB[1:0]	Group configuration	Description
11	NOLOCK3	No lock – no memory locks enabled.
10	WLOCK	Write lock – programming of the flash and EEPROM is disabled for the programming interface. Fuse bits are locked for write from the programming interface.
00	RWLOCK	Read and write lock – programming and read/verification of the flash and EEPROM are disabled for the programming interface. The lock bits and fuses are locked for read and write from the programming interface.

Note: 1. Program the Fuse bits and Boot Lock bits before programming the Lock Bits.

4.15 Register description – Production signature row

4.15.1 RCOSC8M – Internal 8MHz Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
+0x00	RCOSC8M[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – RCOSC8M[7:0]: Internal 8MHz Oscillator Calibration Value**
  This byte contains the oscillator calibration value for the internal 8MHz oscillator. Calibration of the oscillator is performed during production testing of the device. During reset, this value is automatically loaded into calibration register for the 8MHz oscillator. Refer to "RC8MCAL – 8MHz Internal Oscillator Calibration register" on page 108 for more details.

4.15.2 RCOSC32K – Internal 32.768kHz Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
+0x02	RCOSC32K[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – RCOSC32K[7:0]: Internal 32.768kHz Oscillator Calibration Value**
  This byte contains the oscillator calibration value for the internal 32.768kHz oscillator. Calibration of the oscillator is performed during production testing of the device. During reset, this value is automatically loaded into the calibration register for the 32.768kHz oscillator. Refer to "RC32KCAL – 32kHz Oscillator Calibration register" on page 107 for more details.

4.15.3 RCOSC32M – Internal 32MHz Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
+0x03	RCOSC32M[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – RCOSC32M[7:0]: Internal 32MHz Oscillator Calibration Value**
  This byte contains the oscillator calibration value for the internal 32MHz oscillator. Calibration of the oscillator is performed during production testing of the device. During reset, this value is automatically loaded into calibration register B for the 32MHz DFLL. Refer to "CALB – DFLL Calibration register B" on page 109 for more details.

4.15.4 RCOSC32MA – Internal 32MHz RC Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
+0x04	RCOSC32MA[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – RCOSC32MA[7:0]: Internal 32MHz Oscillator Calibration Value**
  This byte contains the oscillator calibration value for the internal 32MHz oscillator. Calibration of the oscillator is performed during production testing of the device. During reset, this value is automatically loaded into calibration register A for the 32MHz DFLL. Refer to "CALA – DFLL Calibration register A" on page 109 for more details.

4.15.5 LOTNUM0 – Lot Number register 0

LOTNUM0, LOTNUM1, LOTNUM2, LOTNUM3, LOTNUM4, and LOTNUM5 contain the lot number for each device. Together with the wafer number and wafer coordinates, this gives a serial number for the device.

Bit	7	6	5	4	3	2	1	0
+0x08	LOTNUM0[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – LOTNUM0[7:0]: Lot Number Byte 0**
  This byte contains byte 0 of the lot number for the device.

4.15.6 LOTNUM1 – Lot Number register 1

Bit	7	6	5	4	3	2	1	0
+0x09	LOTNUM1[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – LOTNUM1[7:0]: Lot Number Byte 1**
  This byte contains byte 1 of the lot number for the device.

4.15.7 LOTNUM2 – Lot Number register 2

Bit	7	6	5	4	3	2	1	0
+0x0A	LOTNUM2[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – LOTNUM2[7:0]: Lot Number Byte 2**
  This byte contains byte 2 of the lot number for the device.

4.15.8 LOTNUM3- Lot Number register 3

Bit	7	6	5	4	3	2	1	0
+0x0B	LOTNUM3[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – LOTNUM3[7:0]: Lot Number Byte 3**
  This byte contains byte 3 of the lot number for the device.

4.15.9 LOTNUM4 – Lot Number register 4

Bit	7	6	5	4	3	2	1	0
+0x0C	LOTNUM4[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – LOTNUM4[7:0]: Lot Number Byte 4**
  This byte contains byte 4 of the lot number for the device.

4.15.10 LOTNUM5 – Lot Number register 5

Bit	7	6	5	4	3	2	1	0
+0x0D	LOTNUM5[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – LOTNUM5[7:0]: Lot Number Byte 5**
  This byte contains byte 5 of the lot number for the device.

4.15.11 WAFNUM – Wafer Number register

Bit	7	6	5	4	3	2	1	0
+0x10	WAFNUM[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	x	x	x	x	x

- **Bit 7:0 – WAFNUM[7:0]: Wafer Number**
  This byte contains the wafer number for each device. Together with the lot number and wafer coordinates, this gives a serial number for the device.

4.15.12 COORDX0 – Wafer Coordinate X register 0

COORDX0, COORDX1, COORDY0, and COORDY1 contain the wafer X and Y coordinates for each device. Together with the lot number and wafer number, this gives a serial number for each device.

Bit	7	6	5	4	3	2	1	0
+0x12	COORDX0[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – COORDX0[7:0]: Wafer Coordinate X Byte 0**
  This byte contains byte 0 of wafer coordinate X for the device.

4.15.13 COORDX1 – Wafer Coordinate X register 1

Bit	7	6	5	4	3	2	1	0
+0x13	COORDX1[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – COORDX0[7:0]: Wafer Coordinate X Byte 1**
  This byte contains byte 1 of wafer coordinate X for the device.

4.15.14 COORDY0 – Wafer Coordinate Y register 0

Bit	7	6	5	4	3	2	1	0
+0x14	COORDY0[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – COORDY0[7:0]: Wafer Coordinate Y Byte 0**
  This byte contains byte 0 of wafer coordinate Y for the device.

4.15.15 COORDY1 – Wafer Coordinate Y register 1

Bit	7	6	5	4	3	2	1	0
+0x15	COORDY1[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – COORDY1[7:0]: Wafer Coordinate Y Byte 1**
  This byte contains byte 1 of wafer coordinate Y for the device

4.15.16 ROOMTEMP – Room Temperature register

Bit	7	6	5	4	3	2	1	0
+0x1E	ROOMTEMP[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – ROOMTEMP[7:0]: Room Temperature Value**
  This byte contains the room temperature value.

4.15.17 HOTTEMP – Hot Temperature register

Bit	7	6	5	4	3	2	1	0
+0x1F	HOTTEMP[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – HOTTEMP[7:0]: Hot Temperature Value**
  This byte contains the hot temperature value.

4.15.18 ADCACAL0 – ADCA Calibration register 0

ADCACAL0 and ADCACAL1 contain the calibration value for the analog- to -digital converter A (ADCA). Calibration is done during production testing of the device. The calibration bytes are not loaded automatically into the ADC calibration registers, and so this must be done from software.

Bit	7	6	5	4	3	2	1	0
+0x20	ADCACAL0[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – ADCACAL0[7:0]: ADCA Calibration Byte 0**
  This byte contains byte 0 of the ADCA calibration data, and must be loaded into the ADCA CALL register.

4.15.19 ADCACAL1 – ADCA Calibration register 1

Bit	7	6	5	4	3	2	1	0
+0x21	ADCACAL1[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – ADCACAL1[7:0]: ADCA Calibration Byte 1**
  This byte contains byte 1 of the ADCA calibration data, and must be loaded into the ADCA CALH register.

4.15.20 ACACURRCAL – ACA Current Calibration register

Bit	7	6	5	4	3	2	1	0
+0x28	ACACURRCAL[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – ACACURRCAL[7:0]: ACA Current Calibration Byte**
  This byte contains the ACA current source calibration value, and must be loaded into the ACA CURRCALIB register.

4.15.21 TEMPSENSE2 – Temperature Sensor Calibration register 2

TEMPSENSE2 and TEMPSENSE3 contain the 12-bit ADCA value from a temperature measurement done with the internal temperature sensor. The measurement is done in production testing at room temperature, and can be used for single- or multi-point temperature sensor calibration.

Bit	7	6	5	4	3	2	1	0
+0x2C	TEMPSENSE2[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – TEMPSENSE2[7:0]: Temperature Sensor Calibration Byte 2**
  This byte contains the byte 2 of the temperature measurement.

4.15.22 TEMPSENSE3 – Temperature Sensor Calibration register 3

Bit	7	6	5	4	3	2	1	0
+0x2D	TEMPSENSE3[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – TEMPSENSE3[7:0]: Temperature Sensor Calibration Byte 3**
  This byte contains byte 3 of the temperature measurement.

4.15.23 TEMPSENSE0 – Temperature Sensor Calibration register 0

TEMPSENSE0 and TEMPSENSE1 contain the 12-bit ADCA value from a temperature measurement done with the internal temperature sensor. The measurement is done in production testing at 85°C, and can be used for single- or multi-point temperature sensor calibration.

Bit	7	6	5	4	3	2	1	0
+0x2E	TEMPSENSE0[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	x	x	x	x	x	x	x	x

- **Bit 7:0 – TEMPSENSE0[7:0]: Temperature Sensor Calibration Byte 0**
  This byte contains the byte 0 of the temperature measurement.

4.15.24 TEMPSENSE1 – Temperature Sensor Calibration register 1

Bit	7	6	5	4	3	2	1	0
+0x2F	TEMPSENSE1[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – TEMPSENSE1[7:0]: Temperature Sensor Calibration Byte 1**
  This byte contains byte 1 of the temperature measurement.

4.15.25 DACA0OFFCAL – DACA Offset Calibration register

Bit	7	6	5	4	3	2	1	0
+0x30	DACA0OFFCAL[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – DACA0OFFCAL[7:0]: DACA0 Offset Calibration Byte**
  This byte contains the offset calibration value for channel 0 in the digital -to -analog converter A (DACA). Calibration is done during production testing of the device. The calibration byte is not loaded automatically into the DAC channel 0 offset calibration register, so this must be done from software.

4.15.26 DACA0GAINCAL – DACA Gain Calibration register

Bit	7	6	5	4	3	2	1	0
+0x31	DACA0GAINCAL[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – DACA0GAINCAL[7:0]: DACA0 Gain Calibration Byte**
  This byte contains the gain calibration value for channel 0 in the digital -to -analog converter A (DACA). Calibration is done during production testing of the device. The calibration byte is not loaded automatically into the DAC gain calibration register, so this must be done from software.

4.15.27 DACA1OFFCAL – DACA Offset Calibration register

Bit	7	6	5	4	3	2	1	0
+0x34	DACA1OFFCAL[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – DACA1OFFCAL[7:0]: DACA1 Offset Calibration Byte**
  This byte contains the offset calibration value for channel 1 in the digital- to -analog converter A (DACA). Calibration is done during production testing of the device. The calibration byte is not loaded automatically into the DAC channel 1 offset calibration register, so this must be done from software.

4.15.28 DACA1GAINCAL – DACA Gain Calibration register

Bit	7	6	5	4	3	2	1	0
+0x35	DACA1GAINCAL[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

- **Bit 7:0 – DACA1GAINCAL[7:0]: DACA1 Gain Calibration Byte**
  This byte contains the gain calibration value for channel 1 in the digital -to- analog converter A (DACA). Calibration is done during production testing of the device. The calibration byte is not loaded automatically into the DAC channel 1 gain calibration register, so this must be done from software.

4.16 Register description – General purpose I/O memory

4.16.1 GPIORn – General Purpose I/O register n

Bit	7	6	5	4	3	2	1	0
+n	GPIORn[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

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.

4.17 Register descriptions – MCU control

4.17.1 DEVID0 – Device ID register 0

DEVID0, DEVID1, and DEVID2 contain the byte identification that identifies each microcontroller device type.

For details on the actual ID, refer to the device datasheet.

Bit	7	6	5	4	3	2	1	0
+0x00	DEVID0[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	1	1	1	1	0

- **Bit 7:0 – DEVID0[7:0]: Device ID Byte 0**
  Byte 0 of the device ID. This byte will always be read as 0x1E. This indicates that the device is manufactured by Atmel.

4.17.2 DEVID1 – Device ID register 1

Bit	7	6	5	4	3	2	1	0
+0x01	DEVID1[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0

- **Bit 7:0 – DEVID[7:0]: Device ID Byte 1**
  Byte 1 of the device ID indicates the flash size of the device.

4.17.3 DEVID2 – Device ID register 2

Bit	7	6	5	4	3	2	1	0
+0x02	DEVID2[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0

- **Bit 7:0 – DEVID2[7:0]: Device ID Byte 2**
  Byte 2 of the device ID indicates the device number.

4.17.4 REVID – Revision ID

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	REVID[3:0]			
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	1/0	1/0	1/0	1/0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use.
- **Bit 3:0 – REVID[3:0]: Revision ID**
  These bits contains the device revision. 0 = A, 1 = B, and so on.

4.17.5 ANAINIT – Analog Initialization register

Bit	7	6	5	4	3	2	1	0
+0x07	–	–	–	–	STARTUPDLYD[1:0]		STARTUPDLYA[1:0]	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:2 / 1:0 – STARTUPDLYx[1:0]: Analog Start-up Delay**
  Setting these bits enables sequential start of the internal components used for the ADC, DAC, and analog comparator with the main input/output connected to that port. When this is done, the internal components, such as voltage reference and bias currents, are started sequentially when the module is enabled. This reduces the peak current consumption during startup of the module. For maximum effect, the start-up delay should be set so that it is larger than 0.5µs.

Table 4-13. Analog start-up delay.

STARTUPDLYx	Group configuration	Description
00	NONE	Direct startup
11	2CLK	2 * Clk_{PER}
10	8CLK	8 * Clk_{PER}
11	32CLK	32 * Clk_{PER}

4.17.6 EVSYSLOCK – Event System Lock register

Bit	7	6	5	4	3	2	1	0
+0x08	–	–	–	EVSYS1LOCK	–	–	–	EVSYS0LOCK
Read/Write	R	R	R	R/W	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – EVSYS1LOCK: Event System Channel 4-7 Lock**
  Setting this bit will lock all registers in the event system related to event channels 4 to 7against for further modification. The following registers in the event system are locked: CH4MUX, CH4CTRL, CH5MUX, CH5CTRL, CH6MUX, CH6CTRL, CH7MUX, and CH7CTRL. This bit is protected by the configuration change protection mechanism. For details, refer to “Configuration change protection” on page 13.
- **Bit 3:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – EVSYS0LOCK: Event System Channel 0-3 Lock**
  Setting this bit will lock all registers in the event system related to event channels 0 to 3 for against further modification. The following registers in the event system are locked: CH0MUX, CH0CTRL, CH1MUX, CH1CTRL, CH2MUX, CH2CTRL, CH3MUX, and CH3CTRL. This bit is protected by the configuration change protection mechanism. F.For details, refer to “Configuration change protection” on page 13.

4.17.7 WEXLOCK – Waveform Extension Lock register

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	–	–	–	WEXCLOCK
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – WEXCLOCK: Waveform Extension Port C Lock**
  Setting this bit will lock all protected registers in the WEX module extension on port C, against further modification. This bit is protected by the configuration change protection mechanism.
  For details, refer to “Configuration change protection” on page 13.

4.17.8 FAULTLOCK – Fault Extension Lock register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	–	–	FAULTC5LOCK	FAULTC4LOCK
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – FAULTC5LOCK: Fault Lock for Timer/Counter 5 on Port C**
  Setting this bit will lock all protected registers in the FAULT module extension on port C of the timer/counter 5, against further modification. This bit is protected by the configuration change protection mechanism.
  For details refer to “Configuration change protection” on page 13.
- **Bit 0 – FAULTC4LOCK: Fault Lock for Timer/Counter 4 on Port C**
  Setting this bit will lock all protected registers in the FAULT module extension on port C of the timer/counter 4, against further modification. This bit is protected by the configuration change protection mechanism.
  For details refer to “Configuration change protection” on page 13.

4.18 Register summary – NVM controller

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	ADDR0	Address Byte 0								26
+0x01	ADDR1	Address Byte 1								26
+0x02	ADDR2	Address byte 2								26
+0x03	Reserved	-	-	-	-	-	-	-	-	
+0x04	DATA0	Data byte 0								26
+0x05	DATA1	Data byte 1								27
+0x06	DATA2	Data byte 2								27
+0x07	Reserved	-	-	-	-	-	-	-	-	
+0x08	Reserved	-	-	-	-	-	-	-	-	
+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CMD	-	CMD[6:0]							27
+0x0B	CTRLA	-	-	-	-	-	-	-	CMDEX	27
+0x0C	CTRLB	-	-	-	-	-	-	EPRM	SPMLOCK	28
+0x0D	INTCTRL	-	-	-	-	SPMLVL[1:0]		EELVL[1:0]		28
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	STATUS	NVMBUSY	FBUSY	-	-	-	-	EELOAD	FLOAD	29
+0x10	LOCKBITS	BLBB1:0]		BLBA[1:0]		BLBAT[1:0]		LB[1:0]		29

4.19 Register summary – Fuses and lockbits

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	Reserved	-	-	-	-	-	-	-	-	
+0x01	FUSEBYTE1	WDWPER[3:0]				WDPER[3:0]				30
+0x02	FUSEBYTE2	-	BOOTRST	-	-	-	-	BODPD[1:0]		30
+0x03	Reserved	-	-	-	-	-	-	-	-	
+0x04	FUSEBYTE4	-	-	-	RSTDISBL	STARTUPTIME[1:0]		WDLOCK	-	31
+0x05	FUSEBYTE5	-	-	BODACT[1:0]		EESAVE	BODLEVEL[2:0]			32
+0x06	FUSEBYTE6	FDACT5	FDACT4	VALUE[5:0]						33
+0x07	LOCKBITS	BLBB[1:0]		BLBA[1:0]		BLBAT[1:0]		LB[1:0]		33

4.20 Register summary – Production signature row

Address	Auto	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 3	Bit 1	Bit 0	Page
+0x00	YES	RCOSC8M	RCOSC8M[7:0]								36
+0x01		Reserved	-	-	-	-	-	-	-	-	
+0x02	YES	RCOSC32K	RCOSC32K[7:0]								36
+0x03	YES	RCOSC32M	RCOSC32M[7:0]								36
+0x04	YES	RCOSC32MA	RCOSC32MA[7:0]								36
+0x05		Reserved	-	-	-	-	-	-	-	-	
+0x06		Reserved	-	-	-	-	-	-	-	-	
+0x07		Reserved	-	-	-	-	-	-	-	-	
+0x08	NO	LOTNUM0	LOTNUM0 [7:0]								37
+0x09	NO	LOTNUM1	LOTNUM1 [7:0]								37

Address	Auto	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 3	Bit 1	Bit 0	Page
+0x0A	NO	LOTNUM2	LOTNUM2 [7:0]								37
+0x0B	NO	LOTNUM3	LOTNUM3 [7:0]								37
+0x0C	NO	LOTNUM4	LOTNUM4 [7:0]								37
+0x0D	NO	LOTNUM5	LOTNUM5 [7:0]								38
+0x0E		Reserved	-	-	-	-	-	-	-	-	
+0x0F		Reserved	-	-	-	-	-	-	-	-	
+0x10	NO	WAFNUM	WAFNUM [7:0]								38
+0x11		Reserved	-	-	-	-	-	-	-	-	
+0x12	NO	COORDX0	COORDX0 [7:0]								38
+0x13	NO	COORDX1	COORDX1 [7:0]								38
+0x14	NO	COORDY0	COORDY0 [7:0]								38
+0x15	NO	COORDY1	COORDY1 [7:0]								39
+0x16		Reserved	-	-	-	-	-	-	-	-	
+0x17		Reserved	-	-	-	-	-	-	-	-	
+0x18		Reserved	-	-	-	-	-	-	-	-	
+0x19		Reserved	-	-	-	-	-	-	-	-	
+0x1A		Reserved	-	-	-	-	-	-	-	-	
+0x1B		Reserved	-	-	-	-	-	-	-	-	
+0x1C		Reserved	-	-	-	-	-	-	-	-	
+0x1D		Reserved	-	-	-	-	-	-	-	-	
+0x1E	NO	ROOMTEMP	ROOMTEMP[7:0]								39
+0x1F	NO	HOTTEMP	HOTTEMP[7:0]								39
+0x20	NO	ADCACAL0	ADCACAL0[7:0]								39
+0x21	NO	ADCACAL1	ADCACAL1[7:0]								39
+0x22		Reserved	-	-	-	-	-	-	-	-	
+0x23		Reserved	-	-	-	-	-	-	-	-	
+0x24		Reserved	-	-	-	-	-	-	-	-	
+0x25		Reserved	-	-	-	-	-	-	-	-	
+0x26		Reserved	-	-	-	-	-	-	-	-	
+0x27		Reserved	-	-	-	-	-	-	-	-	
+0x28	NO	ACACURRCAL	ACACURRCAL[7:0]								40
+0x29		Reserved	-	-	-	-	-	-	-	-	
+0x2A		Reserved	-	-	-	-	-	-	-	-	
+0x2B		Reserved	-	-	-	-	-	-	-	-	
+0x2C	NO	TEMPSENSE2	TEMPSENSE2[7:0]								40
+0x2D	NO	TEMPSENSE3	-	-	-	-	TEMPSENSE3[11:8]				40
+0x2E	NO	TEMPSENSE0	TEMPSENSE0[7:0]								40
+0x2F	NO	TEMPSENSE1	-	-	-	-	TEMPSENSE1[11:8]				41
+0x30	NO	DACA0OFFCAL	DACA0OFFCAL[7:0]								41
+0x31	NO	DACA0GAINCAL	DACA0GAINCAL[1:0]								41
+0x32		Reserved	-	-	-	-	-	-	-	-	
+0x33		Reserved	-	-	-	-	-	-	-	-	

Address	Auto	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 3	Bit 1	Bit 0	Page
+0x34	NO	DACA1OFFCAL	DACA1OFFCAL[7:0]								41
+0x35	NO	DACA1GAINCAL	DACA1GAINCAL[7:0]								42
+0x36		Reserved	-	-	-	-	-	-	-	-	
+0x37		Reserved	-	-	-	-	-	-	-	-	
+0x38		Reserved	-	-	-	-	-	-	-	-	
+0x39		Reserved	-	-	-	-	-	-	-	-	
+0x3A		Reserved	-	-	-	-	-	-	-	-	
+0x3B		Reserved	-	-	-	-	-	-	-	-	
+0x3C		Reserved	-	-	-	-	-	-	-	-	
+0x3D		Reserved	-	-	-	-	-	-	-	-	
+0x3E		Reserved	-	-	-	-	-	-	-	-	
+0x3F		Reserved	-	-	-	-	-	-	-	-	

4.21 Register summary – General purpose I/O registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	GPIOR0	GPIOR0[7:0]								42
+0x01	GPIOR1	GPIOR1[7:0]								42
+0x02	GPIOR2	GPIOR2[7:0]								42
+0x03	GPIOR3	GPIOR3[7:0]								42
+0x04	Reserved	-	-	-	-	-	-	-	-	
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	Reserved	-	-	-	-	-	-	-	-	
+0x07	Reserved	-	-	-	-	-	-	-	-	
+0x08	Reserved	-	-	-	-	-	-	-	-	
+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

4.22 Register summary – MCU control

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DEVID0	DEVID0[7:0]								42
+0x01	DEVID1	DEVID1[7:0]								42
+0x02	DEVID2	DEVID2[7:0]								43
+0x03	REVID	-	-	-	-	REVID[7:0]				43
+0x04	Reserved	-	-	-	-	-	-	-	-	
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	Reserved	-	-	-	-	-	-	-	-	
+0x07	ANAINIT	-	-	-	-	STARTUPDLYD[1:0]		STARTUPDLYA[1:0]		43
+0x08	EVSYSLOC	-	-	-	EVSYS1LOCK	-	-	-	EVSYS0LOCK	44
+0x09	WEXLOCK	-	-	-	-	-	-	-	WEXCLOCK	44
+0x0A	FAULTLOCK	-	-	-	-	-	-	FAULTC5LOC	FAULTC4LOC	45
+0x0B	Reserved	-	-	-	-	-	-	-	-	

4.23 Interrupt vector summary

Table 4-14. NVM Interrupt vectors and their word offset address from NVM controller interrupt base.

Offset	Source	Interrupt description
0x00	EE_vect	Nonvolatile memory EEPROM interrupt vector
0x02	SPM_vect	Nonvolatile memory SPM interrupt vector

5. EDMA – Enhanced Direct Memory Access

5.1 Features

- The EDMA Controller allows data transfers with minimal CPU intervention
  - From data memory to data memory
  - From data memory to peripheral
  - From peripheral to data memory
  - From peripheral to peripheral
- Four peripheral EDMA channels with separate:
  - Transfer triggers
  - Interrupt vectors
  - Addressing modes
  - Data match
- Up to two standard EDMA with separate:
  - Transfer triggers
  - Interrupt vectors
  - Addressing modes
  - Data search
- Programmable channel priority
- From 1byte to 128KB of data in a single transaction
  - Up to 64K block transfer with repeat
  - 1 or 2 bytes burst transfers
- Multiple addressing modes
  - Static
  - Increment
- Optional reload of source and destination address at the end of each
  - Burst
  - Block
  - Transaction
- Optional Interrupt on end of transaction
- Optional connection to CRC Generator module for CRC on EDMA data

5.2 Overview

The enhanced direct memory access (EDMA) controller can transfer data between memories and peripherals, and thus offload these tasks from the CPU. It enables high data transfer rates with minimum CPU intervention, and frees up CPU time. The four EDMA channels enable up to four independent and parallel transfers.

The EDMA controller can move data between SRAM and peripherals, between SRAM locations and directly between peripheral registers. With access to all peripherals, the EDMA controller can handle automatic transfer of data to/from communication modules. The EDMA controller can also read from memory mapped EEPROM.

Data transfers are done in continuous bursts of 1 or 2 bytes. They build block transfers of configurable size from 1 byte to 64KB. Repeat option can be used to repeat once each block transfer for single transactions up to 128KB. Source and destination addressing can be static or incremental. Automatic reload of source and/or destination addresses can be done after each burst or block transfer, or when a transaction is complete. Application software, peripherals, and events can trigger EDMA transfers.

The EDMA channels have individual configuration and control settings. This includes source or destination pointers, transfer triggers, and transaction sizes. They have individual interrupt settings. Interrupt requests can be generated when a transaction is complete or when the EDMA controller detects an error on an EDMA channel.

To have flexibility in transfers, channels can be interlinked so that the second takes over the transfer when the first is finished.

The EDMA controller supports extended features such as double buffering, data match for peripherals or data search for SRAM or EEPROM.

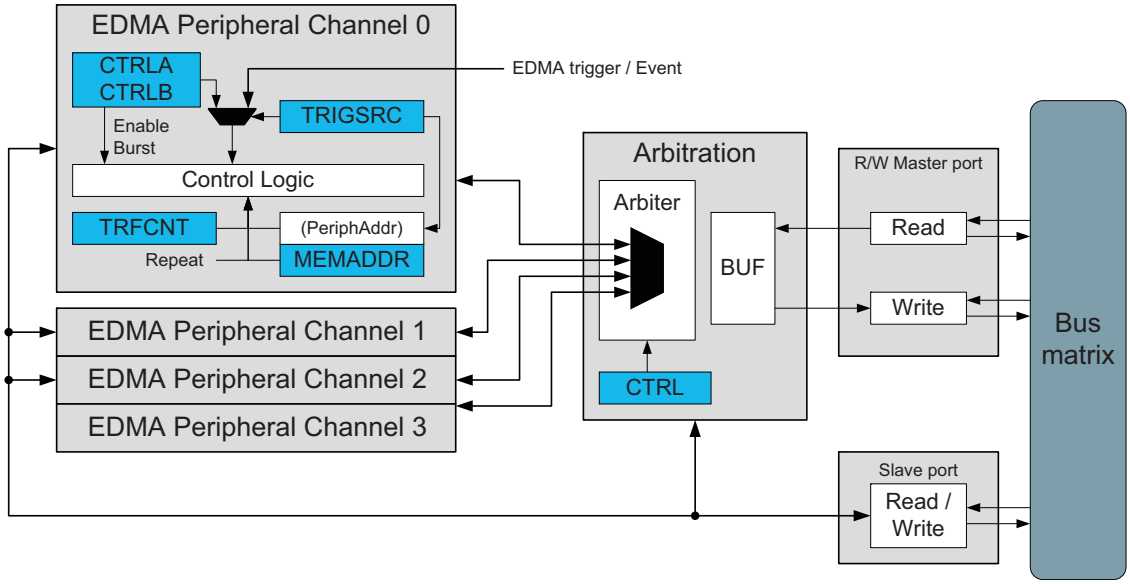
The EDMA controller supports two types of channel. Each channel type can be selected individually.

5.2.1 Peripheral channel

In peripheral channel configuration, a channel enables transfer from specific peripheral address to memory locations or from memory locations to specific peripheral address. The specific peripheral address is provided by the selected trigger source. In this configuration, up to four independent and parallel transfers are supported. The size of a block transfer is limited to 256 bytes for each peripheral channel. The repeat feature enables transfers up to 512 bytes. Two channels can be interlinked so that the second takes over the transfer when the first is finished.

In data match configuration, the EDMA compares the input data from the programmable source with a pattern contained in an EDMA register. As example, this mode can be used with serial peripherals to enable the transfer only if specific character/frame is received (ex. serial address field). Optionally, the transfer counter can be enabled to allow recognition within a window.

Figure 5-1. EDMA – full peripheral channel mode overview.



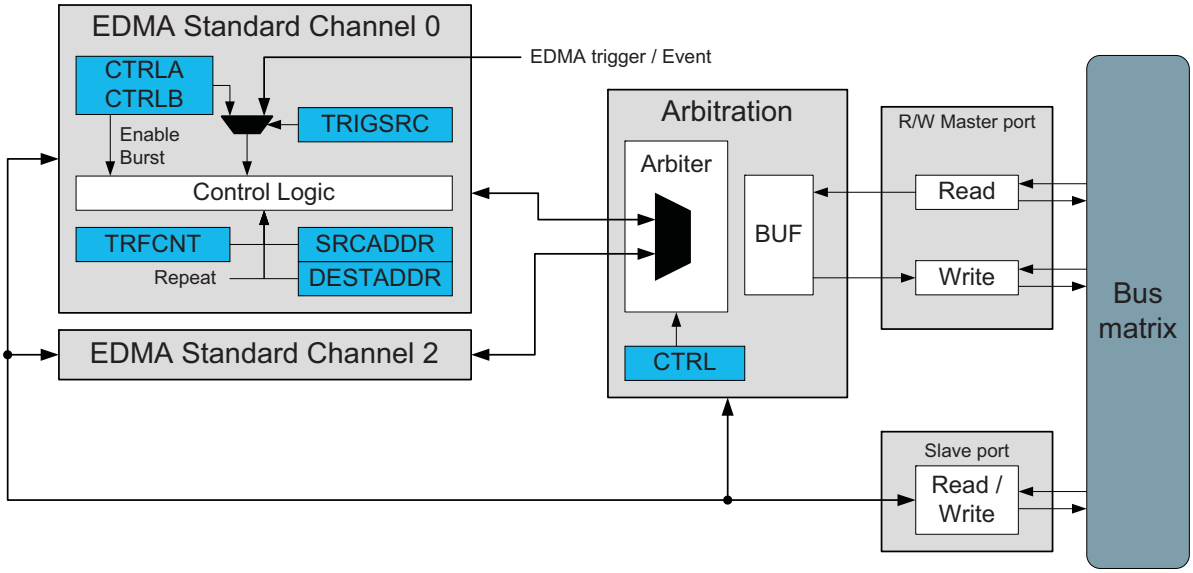
5.2.2 Standard channel

To create a standard channel, the EDMA controller uses resources of two peripheral channels. Register addresses are re-arranged and the standard channel 0 is configured with resources found in the peripheral channels 0 and 1, the standard channel 2 is configured with resources found in the peripheral channels 2 and 3.

In standard channel configuration, any transfer type can be enabled. The trigger source, source address and destination address are independent and separately programmable. The size of a block transfer can be set up to 65536 bytes (64K) for each standard channel. The repeat option enables transfers up to 131072 bytes (128K). Two channels can be interlinked so that the second takes over the transfer when the first is finished, and vice versa.

In data search configuration, the EDMA searches for the data (8-bit or 16-bit) contained in an EDMA register within a memory buffer. On a match, the source address register will provide to the user the intended data pointer.

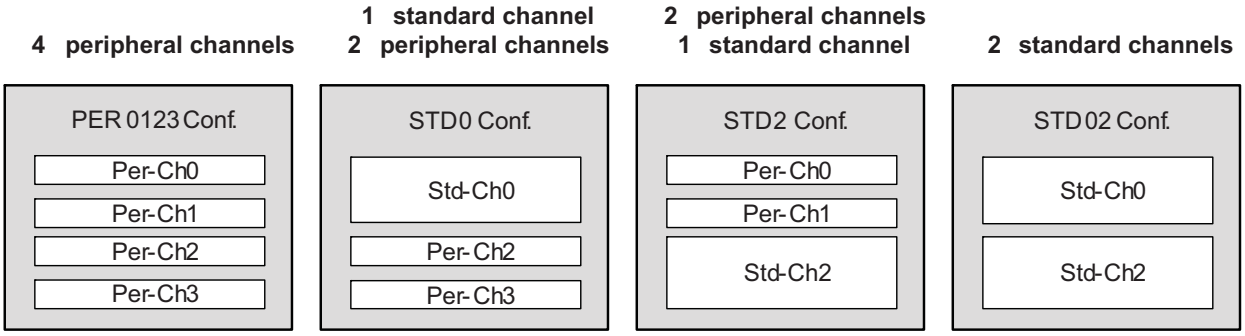
Figure 5-2. EDMA – full standard channel mode overview.



5.2.3 Channel combinations

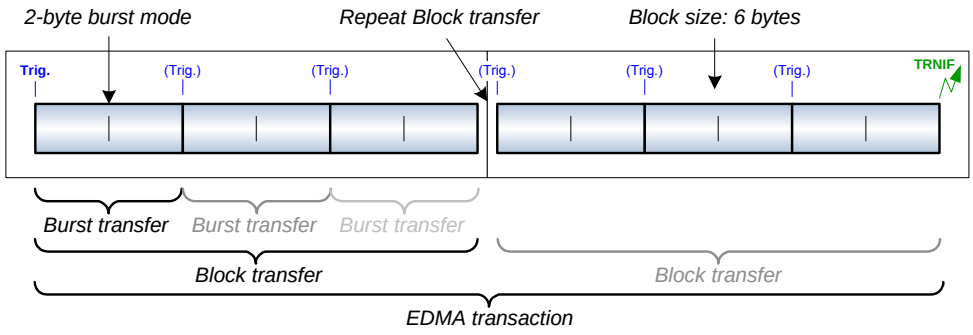
The EDMA can be configured in four different modes (CHMODE bits in “CTRL – Control register ” on page 59) mixing peripheral and standard channels.

Figure 5-3. EDMA – channel modes.



5.3 EDMA transaction

Figure 5-4. EDMA transaction.



A complete EDMA read and write operation between memories and/or peripherals is called an EDMA transaction. A transaction is done in data blocks, and the size of the transaction (number of bytes to transfer) is selectable from software and controlled by the block size and repeat bit settings. Each block transfer is divided into bursts.

5.3.1 Block transfer and repeat block transfer

The size of the block transfer is set by the block transfer count register, and can be programmed from 1 byte to 64KB. If the double buffering is not used, a repeat option can be enabled to repeat once a block transfer before a transaction is complete.

5.3.2 Burst transfer

Since the AVR CPU and EDMA controller use the same data buses, a block transfer is divided into smaller burst transfers. The burst transfer is selectable to 1 or 2 bytes. This means that if a transfer request is pending and the EDMA acquires the data bus, it will occupy the bus until all bytes in the burst are transferred.

A bus arbiter controls when the EDMA controller and the AVR CPU can use the bus. The CPU always has priority, and so as long as the CPU requests access to the bus, any pending burst transfer must wait. The CPU requests bus access when it executes an instruction that writes or reads data to SRAM, I/O memory or to the EEPROM. For more details on memory access bus arbitration refer to.

5.4 Transfer triggers

EDMA transfers can be started only when an EDMA transfer trigger is detected. A transfer trigger can be set-up by software, from an external trigger source (peripheral), or from an event. There are dedicated source trigger selections for each EDMA channel. The available trigger sources may vary from device to device, depending on the modules or peripherals that exist in the device. Using a transfer trigger for a module or peripherals that does not exist will have no effect. For a list of all transfer triggers of peripheral channels, refer to Table 5-8 on page 64 and for standard channels, refer to Table 5-18 on page 71.

By default, a trigger starts a block transfer operation. When the block transfer is complete, the channel is automatically disabled. When enabled again, the channel will wait for the next block transfer trigger.

It is possible to select the trigger to start a burst transfer instead of a block transfer. This is called a single-shot transfer, and for each trigger only one burst is transferred. In this configuration, when block repeat transfer mode is enabled (and if no double buffering mode), the next block transfer does not require a transfer trigger. It will start as soon as the previous block is done.

If a source generates a transfer trigger during an ongoing transfer, this will be kept pending, and the transfer can start when the ongoing one is done. Only one pending transfer can be kept, and so if the trigger source generates more transfer triggers when one is already pending, these will be lost.

In peripheral channel configuration, setting the trigger source automatically determines the peripheral register address and the data transfer direction.

5.5 Addressing and transfer count

5.5.1 Addressing in peripheral channel configuration

In peripheral channel configuration, the memory address for an EDMA transfer can either be static or automatically incremented and the 16-bit peripheral address is automatically incremented if 2-byte burst is set. When memory address increment is used, the default behavior is to update the memory address after each access. The original memory address is stored by the EDMA controller, and can be individually configured to be reloaded at the following points:

- End of each burst transfer
- End of each block transfer
- End of transaction
- Never reloaded

When 2-byte burst option is used to address 16-bit peripheral, the first byte access of the burst will be for the low byte of the 16-bit register (ex: ACDA.CH0RESL) the second, for the high byte (ex: ACDA.CH0RESH). The 1-byte burst option is reserved for 8-bit peripherals.

5.5.2 Addressing in standard channel configuration

In standard channel configuration, the source and destination address for an EDMA transfer can either be static or automatically incremented, with individual selections for source and destination. When address increment is used, the default behavior is to update the address after each access. The original source and destination addresses are stored by the EDMA controller, and so the source and destination addresses can be individually configured to be reloaded at the following points:

- End of each burst transfer
- End of each block transfer
- End of transaction
- Never reloaded

5.5.3 Transfer count reload

When the channel transaction complete interrupt flag is set, the transfer counter is reloaded. The transfer counter is not reloaded when the channel error interrupt flag is set.

5.6 Priority between channels

If several channels request a data transfer at the same time, a priority scheme is available to determine which channel is allowed to transfer data. Application software can decide whether one or more channels should have a fixed priority or if a round robin scheme should be used. A round robin scheme means that the channel that last transferred data will have the lowest priority.

5.7 Double buffering

Two channels can be interlinked so that two different EDMA transactions can be serialized, the second takes over the transfer when the first is finished.

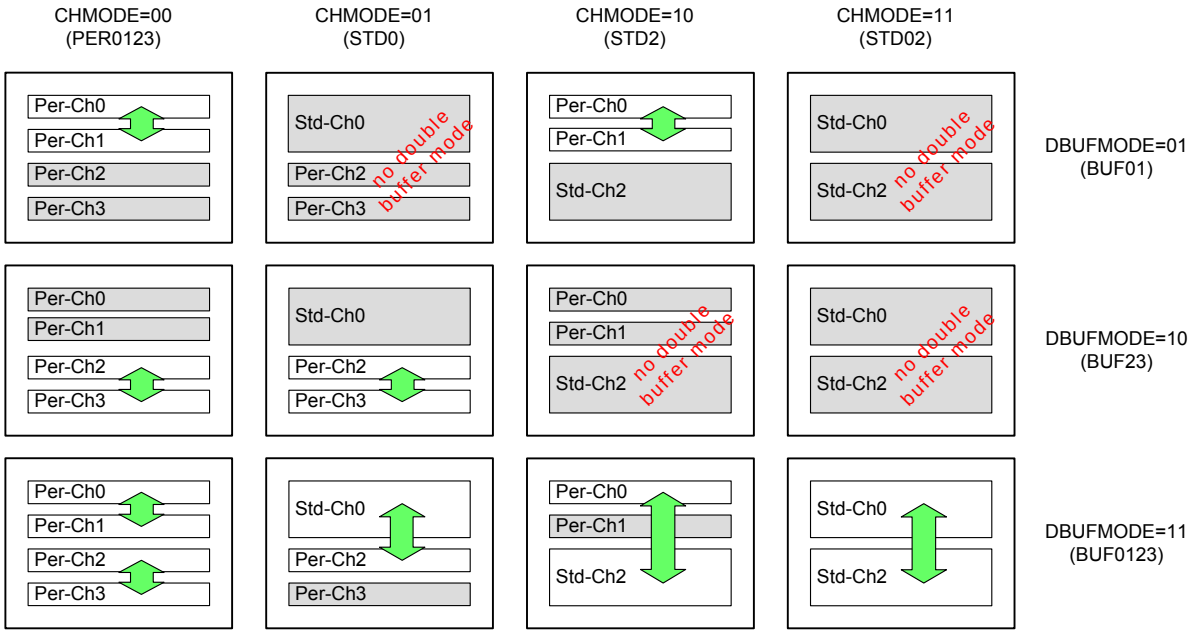
- This can leave time for the application to process the data transferred by the first channel, prepare fresh data buffers, and set up the channel registers again while the second channel is working.
- This can link two different processes as data match on serial peripheral and once matching (ex: address recognition) a transfer of valid data is enabled.

This is referred to as double buffering or chained transfers. The first channel is referred as the first software enabled channel within the pair of linked channels.

DBUFMODE bits in CRTL register (CTRL.DBUFMODE) configure the double buffer modes. At channel level, the REPEAT bit (CTRLA.REPEAT) of the second channel enables the link. The end of transfer (without error) on the first channel enables the second channel (CTRLA.REPEAT).

Note that double buffering is incompatible with repeat block transfer.

Figure 5-5. EDMA - Double buffer modes versus channel modes.



5.8 Data processing

5.8.1 Data match

This feature is available only for peripheral channels doing transfer from peripheral to memory locations.

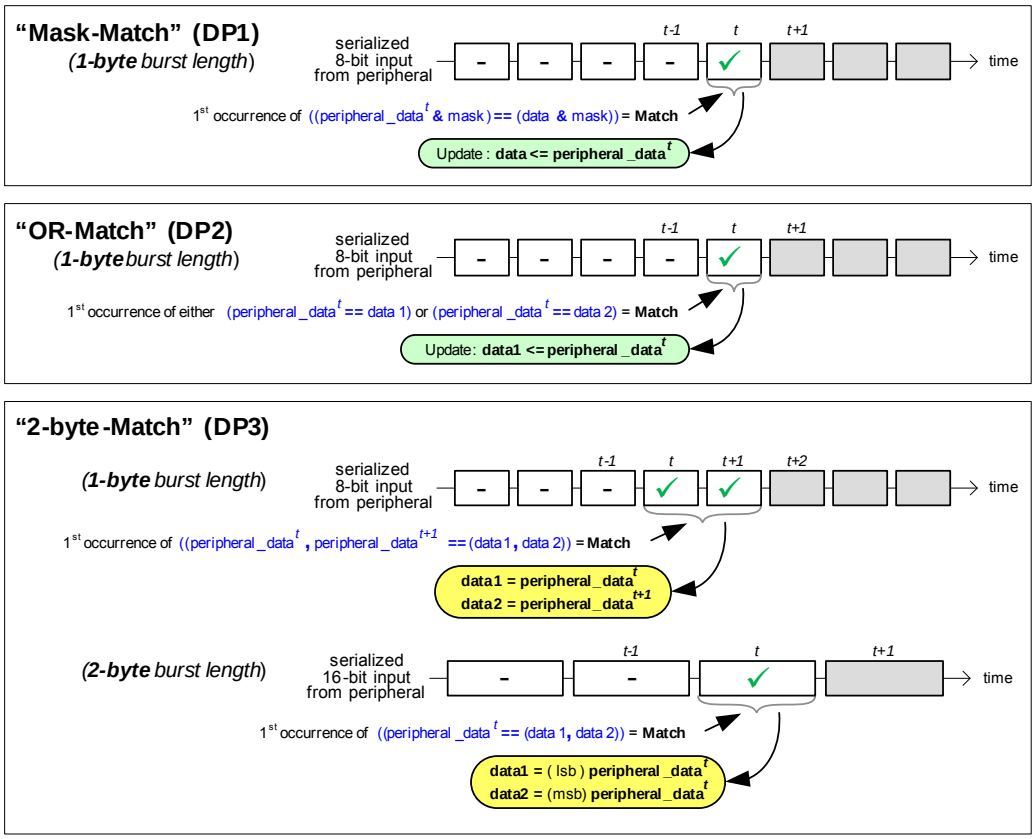
To avoid unnecessary data transfers between peripherals and data memory, the EDMA controller has a built-in data match feature. In this mode, the memory address register is set to store the data used during match operation. The operation stops on data match or if the transfer count reaches zero. If the block transfer counter is programmed with zero, then the data match is in free running mode and stopped when a match occurs. In case of no match an abort could be necessary.

If a data match occurs, the corresponding peripheral channel is disabled and optionally a transaction complete interrupt is generated. To know the true matched data in Mask-match or OR-match setting, the matched data is updated in the corresponding EDMA register. Note that the un-matched data are lost.

If the block transfer counter is used and no data match is detected, then the channel is disabled, the transfer counter is reloaded and optionally an error interrupt is generated.

Repeat block transfer mode is unavailable in data match operation.

Figure 5-6. EDMA – data match.



5.8.2 Data search

This feature is only available for standard channels.

To offload the CPU, the EDMA controller has a built-in data search feature. In this mode, the destination address register is set to store the bytes used for searching while the source address register is set to store the first address of the memory buffer to scan. The data search operation stops on match or if the transfer count reaches zero. If the block transfer counter is programmed with zero, then the data search is in free running mode and stopped when a match occurs. In case on no match, an abort could be necessary.

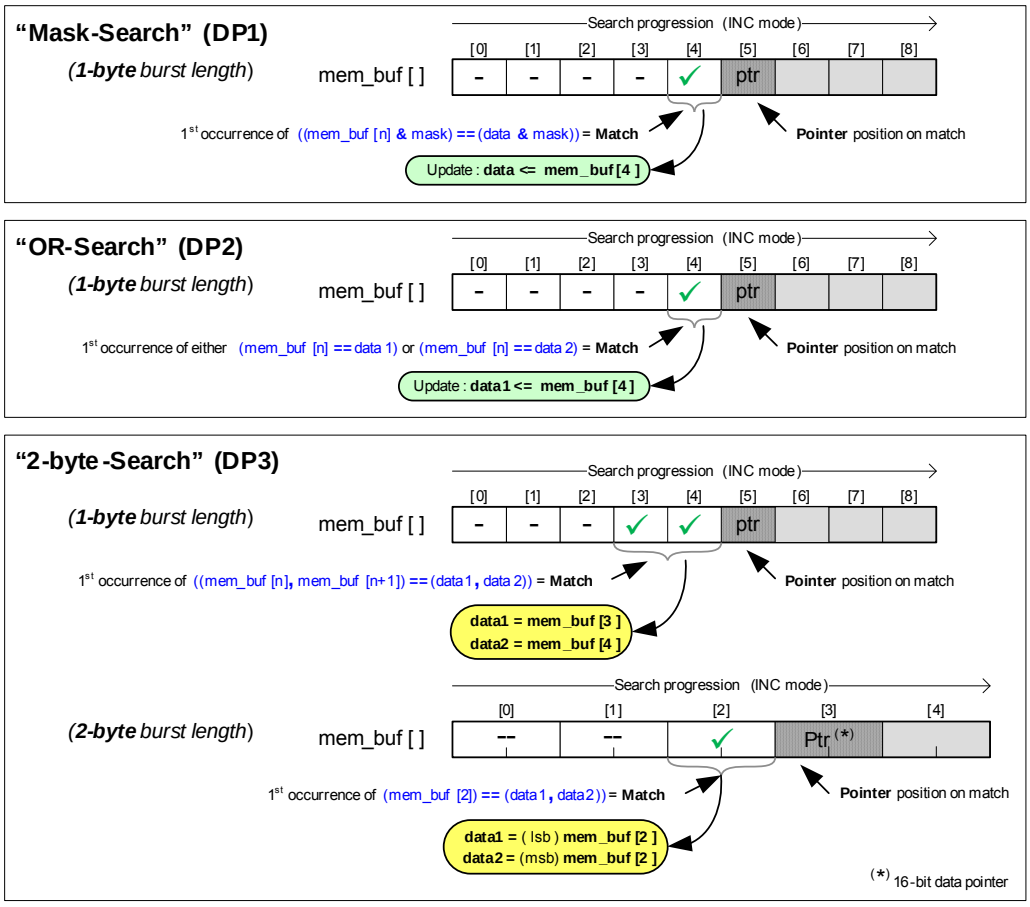
If a data match occurs, the corresponding standard channel is disabled and optionally a transaction complete interrupt is generated. To know the true matched data in Mask-match or OR-match setting, the matched data is updated in the corresponding EDMA register. The source address register can be used to compute the data pointer.

If the block transfer counter is used and no data match is detected, then the channel is disabled, the transfer counter is reloaded and optionally an error interrupt is generated.

Repeat block transfer mode is unavailable in data search operation.

In this mode, it is recommended to configure the increment mode and no reload mode for source address.

Figure 5-7. EDMA – data search



5.9 Error detection

The EDMA controller can detect erroneous operation. Error conditions are detected individually for each EDMA channel, and the error conditions are:

- Write to EEPROM locations
- Reading EEPROM when the EEPROM is off (sleep entered)
- EDMA controller or a busy channel is disabled in software during a transfer

5.10 Software reset

Both the EDMA controller and an EDMA channel can be reset from the user software. When the EDMA controller is reset, all registers associated with the EDMA controller, including channels, are cleared. A software reset can be done only when the EDMA controller is disabled.

When an EDMA channel is reset, all registers associated with the EDMA channel are cleared. A software reset can be done only when the EDMA channel is disabled.

5.11 Protection

In order to enable a safe operation:

- The channel mode bits (CTRL.CHMODE) are protected against user modification when the EDMA controller is enabled (ENABLE=1).
- Some channel bits and registers are protected against user modification during a transaction (CTRL.ENABLE=1):
  - REPEAT and SINGLE bits in CTRLA register
  - ADDCTRL (SRCADDCRTL) and DESTADDCTRL registers
  - ADDR (SRCADDR) and DESTADDR 16-bit registers
  - TRFCNTL (TRFCNT) and TRFCNTH registers.

Note that TRFREQ bit in CTRLA register and TRIGSRC register are not protected.

5.12 Interrupts

The EDMA controller can generate interrupts when an error is detected on an EDMA channel or when a transaction is complete for an EDMA channel. Each EDMA channel has a separate interrupt vector, and there are different interrupt flags for error and transaction complete.

5.13 Register description – EDMA controller

5.13.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	RESET	CHMODE[1:0]		DBUFMODE[1:0]		PRIMODE[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – ENABLE: Enable**
  Setting this bit enables the EDMA controller. If the EDMA controller is enabled and this bit is written to zero, the ENABLE bit is not cleared before the internal transfer buffer is empty, and the EDMA data transfer is aborted.
- **Bit 6 – RESET: Software Reset**
  Writing a one to RESET will be ignored as long as EDMA is enabled (ENABLE = 1). The software reset re-initializes the controller and the channel registers. This bit can be set only when the EDMA controller is disabled (ENABLE = 0).
- **Bit 5:4 – CHMODE[1:0]: Channel Mode**
  These bits set the channel in standard or peripheral mode, according to Table 5-1 on page 59.

Table 5-1. Channel configuration settings.

CMODE[1:0]	Group configuration	Description	Channel number
00	PER0123	4 peripheral channels	0,1,2,3
01	STD0	1 standard channel	0
		2 peripheral channels	2,3
10	STD2	2 peripheral channels	0,1
		1 standard channel	2
11	STD02	2 standard channels	0,2

This field can be set only when the EDMA controller is disabled (ENABLE = 0).

- **Bit 3:2 – DBUFMODE[1:0]: Double Buffer Mode**
  These bits enable the double buffer on the different channels according to Table 5-2 on page 59.

Table 5-2. EDMA double buffer settings.

DBUFMODE[1:0]	Group configuration	Description
00	DISABLED	No double buffer enabled
01	BUF01	Double buffer enabled on peripheral channels 0 and 1 (if exist)
10	BUF23	Double buffer enabled on peripheral channels 2 and 3 (if exist)
11	BUF0123	- If CHMOD = 00: Double buffer enabled on peripheral channels 0 and 1 and also on peripheral channels 2 and 3 - If CHMOD ! = 00: Double buffer enabled on channels 0 and 2 (irrespective of the channel configuration)

In buffer modes, REPEAT bit of each channel controls the link (ex: to set-up a link from CHx to CHy, REPEAT bit of CHy must be set).

There are no predefined channels order in the double buffer mode. The first channel that is enabled by software starts first and, at the end of its transaction, it enables the second channel for a new transaction if the corresponding REPEAT bit is set (hardware setting of CTRLA.ENABLE bit).

- **Bit 1:0 – PRIMODE[1:0]: Priority Mode**
  These bits determine the internal channel priority according to Table 5-3 on page 60.

Table 5-3. EDMA channel priority settings.

PRIMODE[1:0]	Group Configuration	Description
00	RR0123	Round robin
01	RR123	Channel0 > Round robin (channel 1, 2 and 3)
10	RR23	Channel0 > Channel1 > Round robin (channel 2 and 3)
11	CH0123	Channel0 > Channel1 > Channel2 > Channel3

5.13.2 INTFLAGS – Interrupt Status Flags register

Bit	7	6	5	4	3	2	1	0
+0x03	CH3ERRIF	CH2ERRIF	CH1ERRIF	CH0ERRIF	CH3TRNIF	CH2TRNIF	CH1TRNIF	CH0TRNIF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – CHnERRIF: Channel n Error Interrupt Flag**
  If an error condition is detected on EDMA channel n, the CHnERRIF flag will be set. Writing a one to this bit location will clear the flag.
  These flags are duplicated in each CTRLB channel register.
- **Bit 3:0 – CHnTRNIF: Channel n Transaction Complete Interrupt**
  When a transaction on channel n has been completed, the CHnTRFIF flag will be set. Writing a one to this bit location will clear the flag.
  These flags are duplicated in each CTRLB channel register.

5.13.3 STATUS –Status register

Bit	7	6	5	4	3	2	1	0
+0x04	CH3BUSY	CH2BUSY	CH1BUSY	CH0BUSY	CH3PEND	CH2PEND	CH1PEND	CH0PEND
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – CHnBUSY: Channel n Busy**
  When channel n starts an EDMA transaction, the CHnBUSY flag will be read as one. This flag is automatically cleared when the EDMA channel is disabled, when the channel n transaction complete interrupt flag is set, or if the EDMA channel n error interrupt flag is set.
- **Bit 3:0 – CHnPEND: Channel n Pending**
  If a block transfer is pending on EDMA peripheral channel n high, the CHnPEND flag will be read as one. This flag is automatically cleared when the block transfer starts or if the transfer is aborted.

5.13.4 TEMP – Temporary register

Bit	7	6	5	4	3	2	1	0
+0x06	TEMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – TEMP[7:0]: Temporary bits**
  This register is used when reading 16-bit registers in the EDMA controller. The high byte of the 16-bit register is stored here when the low byte is read by the CPU. This register can also be read and written from the user software. Reading and writing 16- bit registers requires special attention.
  For details, refer to “Accessing 16-bit registers” on page 13.

5.14 Register description – Peripheral channel

5.14.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	RESET	REPEAT	TRFREQ	–	SINGLE	–	BURSTLEN
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – ENABLE: Channel Enable**
  Setting this bit enables the peripheral channel. This bit is automatically cleared when the transaction is completed. If the peripheral channel is enabled and this bit is written to zero, the channel is disabled between bursts and the transfer is aborted.
- **Bit 6 – RESET: Software Reset**
  Setting this bit will reset the peripheral channel. It can only be set when the peripheral channel is disabled (CTRLA.ENABLE = 0). Writing a one to this bit will be ignored as long as the peripheral channel is enabled (CHEN=1). This bit is automatically cleared when reset is completed.
- **Bit 5 – REPEAT: Repeat Mode**
  Setting this bit enables the repeat mode. The repeat mode enables a “Repeat Block Transfer” if there is no double buffering mode. Else this bit enables the link for the buffer mode and it is cleared by hardware at the end of the first block transfer. A write to this bit will be ignored while the channel is enabled.
- **Bit 4 – TRFREQ: Transfer Request**
  Setting this bit requests a data transfer on the peripheral channel and acts as a software trigger. This bit is automatically cleared at the beginning of the data transfer. Writing this bit does not have any effect unless the peripheral channel is enabled.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2 – SINGLE: Single-Shot Data Transfer**
  Setting this bit enables the single-shot mode. The peripheral channel will then do a burst transfer of BURSTLEN bytes on the transfer trigger. A write to this bit will be ignored while the channel is enabled.
- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 0 – BURSTLEN: Burst Length**
  This bit defines the peripheral channel burst length according to Table 5-4 on page 62.
  This bit cannot be changed if the channel is busy.

Table 5-4. Peripheral channel burst length.

BURSTLEN	Group configuration	Description
00	1BYTE	1 byte burst
01	2BYTE	2 bytes burst

5.14.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]	
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – CHBUSY - Busy**
  When the peripheral channel starts an EDMA transaction, the BUSY flag will be read as one. This flag is automatically cleared when the EDMA channel is disabled, when the channel transaction complete interrupt flag is set or when the channel error interrupt flag is set.
- **Bit 6 – CHPEND - Pending**
  If a block transfer is pending on the peripheral channel, the PEND flag will be read as one. This flag is automatically cleared when the transfer starts or if the transfer is aborted.
- **Bit 5 – ERRIF - Error Interrupt Flag**
  If an error condition is detected on the peripheral channel, the ERRIF flag will be set and the optional interrupt is generated.
  Since the peripheral channel error interrupt shares the interrupt address with the peripheral channel n transaction complete interrupt, ERRIF will not be cleared when the interrupt vector is executed. This flag is cleared by writing a one to this location.
- **Bit 4 – TRNIF - Transaction Complete Interrupt Flag**
  When a transaction on the peripheral channel has been completed, the TRNIF flag will be set and the optional interrupt is generated. When repeat block transfer is not enabled, the transaction is completed and TRNIFR is set after the block transfer. Else, TRNIF is also set after the last block transfer.
  Since the peripheral channel transaction n complete interrupt shares the interrupt address with the peripheral channel error interrupt, TRNIF will not be cleared when the interrupt vector is executed. This flag is cleared by writing a one to this location.
- **Bit 3:2 – ERRINTLVL[1:0]: Channel Error Interrupt Level**
  These bits enable the interrupt for EDMA channel transfer errors and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger for the conditions when ERRIF is set.
- **Bit 1:0 – TRNINTLVL[1:0]: Channel Transaction Complete Interrupt Level**
  These bits enable the interrupt for EDMA channel transaction completes and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger for the conditions when TRNIF is set.

5.14.3 ADDCTRL – Address Control register

Bit	7	6	5	4	3	2	1	0
+0x02	-	-	RELOAD[1:0]		-	DIR[2:0]		
Read/Write	R	R	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:4 – RELOAD[1:0]: Memory Address Reload**
  These bits decide the memory address reload according to Table 5-5 on page 63.
  A write to these bits is ignored while the channel is busy.

Table 5-5. Memory address reload settings.

RELOAD[1:0]	Group configuration	Description
00	NONE	No reload performed.
01	BLOCK	Memory address register is reloaded with initial value at end of each block transfer.
10	BURST	Memory address register is reloaded with initial value at end of each burst transfer.
11	TRANSACTION	Memory address register is reloaded with initial value at end of each transaction.

- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – DIR[2:0]: Memory Address Mode**
  These bits decide the memory address mode, according to Table 5-6 on page 63 and Table 5-7 on page 63.
  These bits cannot be changed if the channel is busy.

Table 5-6. Memory address mode settings – Transfer memory to peripheral.

DIR[2:0]	Group configuration	Description
000	FIXED	Fixed memory address
001	INC	Increment
010	–	Reserved
011	–	Reserved
1xx	–	Reserved

Table 5-7. Memory address mode settings – Transfer peripheral to memory.

DIR[2:0]	Group configuration	Description
000	FIXED	Fixed memory address
001	INC	Increment
010	–	Reserved

DIR[2:0]	Group configuration	Description
011	–	Reserved
100	DP1	**“Mask-Match”** (1 byte) - data: ADDRL register - mask: ADDRH register (active bit-mask=1) Note: Only available in 1-byte burst length mode
101	DP2	**“OR-Match”** (1 byte) - data1: ADDRL register OR - data2: ADDRH register Note: Only available in 1-byte burst length mode
110	DP3	**“2-byte-Match”** (2 consecutive bytes) - data1 (1st data or lsb) in DESTADDRL register followed by - data2 (2nd data or msb) in DESTADDRH register.
111	–	Reserved

5.14.4 TRIGSRC – Trigger Source register

Bit	7	6	5	4	3	2	1	0
+0x04	TRIGSRC[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – TRIGSRC[7:0]: Peripheral Channel Trigger Source Select**
  These bits select which trigger source is used for triggering a transfer on the EDMA channel. Some modules or peripherals are not available as trigger source for EDMA peripheral channels. Other codes than those of Table 5-8 on page 64 will have no effect.
  If the interrupt flag related to the trigger source is cleared or the interrupt level enabled so that an interrupt is triggered, the EDMA request will be lost. Since an EDMA request can clear the interrupt flag, interrupts can be lost.

Note: For most trigger sources, the request is cleared by accessing a register belonging to the peripheral with the request. Refer to the different peripheral chapters for description on how requests are generated and cleared.

Table 5-8. Trigger codes for EDMA peripheral channels.

TRIGSRC[7:0]	Group configuration	Description
0x10	ADCA(1)	ADCA EDMA triggers base value
0x15	DACA(1)	DACA EDMA triggers base value
0x4A	SPIC	SPI C EDMA triggers base value
0x4C	USARTC0	USART C0 EDMA triggers base value
0x6C	USARTD0	USART D0 EDMA triggers base value

Note: 1. It is recommended to set BURST2 configuration when reading or writing 16-bits registers

Table 5-9. EDMA trigger source offset values for ADC triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	CH0	ADC channel 0 - transfer direction: peripheral to memory - EDMA reads CH0RES register

Table 5-10. EDMA trigger source offset values for DAC triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	CH0	DAC channel 0 - transfer direction: memory to peripheral - EDMA writes CH0DATA register
+0x01	CH1	DAC channel 1 - transfer direction: memory to peripheral - EDMA writes CH1DATA register

Table 5-11. EDMA trigger source offset values for USART triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	RXC	Receive complete - transfer direction: peripheral to memory - EDMA reads DATA register
+0x01	DRE	Data register empty - transfer direction: memory to peripheral - EDMA writes DATA register

Table 5-12. EDMA trigger source offset values for SPI triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	IFRXC	Transfer complete in standard mode or receive complete in double buffer mode - transfer direction: peripheral to memory - EDMA reads DATA register
+0x01	IFDRE	Transfer complete in standard mode or data register empty in double buffer mode - transfer direction: memory to peripheral - EDMA writes DATA register

5.14.5 TRFCNT – Block Transfer Count register

TRFCNT defines the number of bytes in a block transfer. The value of TRFCNT is decremented after each byte read by the EDMA channel.

When TRFCNT reaches zero, the register is reloaded with the last value written to it.

Bit	7	6	5	4	3	2	1	0
+0x06	TRFCNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

- **Bit 7:0 – TRFCNT[7 :0]: Block Transfer Count**
  The default value of this register is 0x01. If in transfer mode the user writes 0x00 to this register and fires an EDMA trigger, EDMA will perform 256 transfers. If this register is set to 0x00 in data match mode, the operation will have no count limit and will run up to a match occurs.

5.14.6 ADDRL – Memory Address register Low

ADDRL and ADDRH represent the 16-bit value ADDR, which is the memory address in a transaction executed by a peripheral channel. ADDRH is the most significant byte in the register. ADDR may be automatically incremented based on settings in the DIR bits in “ADDCTRL – Address Control register” on page 63.

In data match mode, ADDR is used for data to recognize, according to the Table 5-7 on page 63.

Bit	7	6	5	4	3	2	1	0
+0x08	ADDR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – ADDR[7 :0]: Memory Address Low Byte**
  These bits hold the low-byte of the 16-bit memory address.

5.14.7 ADDRH – Memory Address register High

Bit	7	6	5	4	3	2	1	0
+0x09	ADDR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – ADDR[15:8]: Memory Address High Byte**
  These bits hold the high-byte of the 16-bit memory address.

5.15 Register description – Standard channel

5.15.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	RESET	REPEAT	TRFREQ	-	SINGLE	-	BURSTLEN
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – ENABLE: Channel Enable**
  Setting this bit enables the standard channel. This bit is automatically cleared when the transaction is completed. If the standard channel is enabled and this bit is written to zero, the channel is disabled between bursts and the transfer is aborted.
- **Bit 6 – RESET: Software Reset**
  Setting this bit will reset the standard channel. It can only be set when the standard channel is disabled (CTRLA.ENABLE=0). Writing a one to this bit will be ignored as long as the standard channel is enabled (CTRLA.ENABLE=1). This bit is automatically cleared when reset is completed.
- **Bit 5 – REPEAT: Repeat Mode**
  Setting this bit enables the repeat mode. The repeat mode enables a “Repeat Block Transfer” if there is no double buffering mode. Else this bit enables the link for the buffer mode and it is cleared by hardware at the end of the first block transfer. A write to this bit will be ignored while the channel is enabled.
- **Bit 4 – TRFREQ: Transfer Request**
  Setting this bit requests a data transfer on the standard channel and acts as a software trigger. This bit is automatically cleared at the beginning of the data transfer. Writing this bit does not have any effect unless the standard channel is enabled.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2 – SINGLE: Single-Shot Data transfer**
  Setting this bit enables the single-shot mode. The standard channel will then do a burst transfer of BURSTLEN bytes on the transfer trigger. A write to this bit will be ignored while the channel is enabled.
- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 0 – BURSTLEN: Burst Length**
  This bit defines the standard channel burst length according to Table 5-13 on page 67.
  This bit cannot be changed if the channel is busy.

Table 5-13. Standard channel burst length.

BURSTLEN	Group configuration	Description
00	1BYTE	1 byte burst
01	2BYTE	2 bytes burst

5.15.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]	
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – CHBUSY - Channel Busy**
  When the standard channel starts an EDMA transaction, the CHBUSY flag will be read as one. This flag is automatically cleared when the EDMA channel is disabled, when the channel transaction complete interrupt flag is set or when the channel error interrupt flag is set.
- **Bit 6 – CHPEND - Channel Pending**
  If a block transfer is pending on the standard channel, the CHPEND flag will be read as one. This flag is automatically cleared when the transfer starts or if the transfer is aborted.
- **Bit 5 – ERRIF - Error Interrupt Flag**
  If an error condition is detected on the standard channel, the ERRIF flag will be set and the optional interrupt is generated.
  Since the standard channel error interrupt shares the interrupt address with the peripheral channel n transaction complete interrupt, ERRIF will not be cleared when the interrupt vector is executed. This flag is cleared by writing a one to this location.
- **Bit 4 – TRNIF - Transaction Complete Interrupt Flag**
  When a transaction on the standard channel has been completed, the TRNIF flag will be set and the optional interrupt is generated. When repeat block transfer is not enabled, the transaction is completed and TRNIFR is set after the block transfer. Else, TRNIF is also set after the last block transfer.
  Since the standard channel transaction n complete interrupt shares the interrupt address with the peripheral channel error interrupt, TRNIF will not be cleared when the interrupt vector is executed. This flag is cleared by writing a one to this location.
- **Bit 3:2 – ERRINTLVL[1:0]: Channel Error Interrupt Level**
  These bits enable the interrupt for EDMA channel transfer errors and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger for the conditions when ERRIF is set.
- **Bit 1:0 – TRNINTLVL[1:0]: Channel Transaction Complete Interrupt Level**
  These bits enable the interrupt for EDMA channel transaction completes and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger for the conditions when TRNIF is set.

5.15.3 SRCADDCTRL – Source Address Control register

Bit	7	6	5	4	3	2	1	0
+0x02	-	-	SRCRELOAD[1:0]		-	SRCDIR[2:0]		
Read/Write	R	R	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:4 – SRCRELOAD[1:0]: Source Address Reload**
  These bits decide the source address reload according to Table 5-14 on page 69. A write to these bits is ignored while the channel is busy.

Table 5-14. Source address reload settings.

SRCRELOAD[1:0]	Group configuration	Description
00	NONE	No reload performed.
01	BLOCK	Source address register is reloaded with initial value at end of each block transfer.
10	BURST	Source address register is reloaded with initial value at end of each burst transfer.
11	TRANSACTION	Source address register is reloaded with initial value at end of each transaction.

- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – SRCDIR[2:0]: Source Address Mode**
  These bits decide the source address mode, according to Table 5-15 on page 69.
  These bits cannot be changed if the channel is busy.

Table 5-15. Source address mode settings.

SRCDIRD[2:0]	Group configuration	Description
000	FIXED	Fixed address
001	INC	Increment
010	-	Reserved
011	-	Reserved
1xx	-	Reserved

5.15.4 DESTADDCTRL – Destination Address Control register

Bit	7	6	5	4	3	2	1	0
+0x03	-	-	DESTRELOAD[1:0]		-	DESTDIR[2:0		
Read/Write	R	R	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:4 – DESTRELOAD[1:0]: Destination Address Reload**
  These bits decide the destination address reload according to Table 5-16 on page 70.
  A write to these bits is ignored while the channel is busy.

Table 5-16. EDMA channel source address reload settings.

DESTRELOAD[1:0]	Group configuration	Description
00	NONE	No reload performed.
01	BLOCK	Destination address register is reloaded with initial value at end of each block transfer.
10	BURST	Destination address register is reloaded with initial value at end of each burst transfer.
11	TRANSACTION	Destination address register is reloaded with initial value at end of each transaction.

- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – DESTDIR[2:0]: Destination Address Mode**
  These bits decide the destination address mode, according to Table 5-17 on page 70.
  These bits cannot be changed if the channel is busy.

Table 5-17. Destination address mode settings.

DESTDIR[1:0]	Group configuration	Description
000	FIXED	Fixed address
001	INC	Increment
010	-	Reserved
011	-	Reserved
100	DP1	**"Mask-Search"** (1byte) - data: DESTADDRL register - mask: DESTADDRH register (active bit-mask=1) Note: Only available in 1-byte burst length mode
101	DP2	**"OR-Search"** (1 byte) - data1: DESTADDRL register OR - data2: DESTADDRH register Note: Only available in 1-byte burst length mode
110	DP3	**"2-byte-Search"** (2 consecutive bytes) - data1 (1st data or lsb) in DESTADDRL register followed by - data2 (2nd data or msb) in DESTADDRH register.
111	-	Reserved

5.15.5 TRIGSRC – Trigger Source register

Bit	7	6	5	4	3	2	1	0
+0x04	TRIGSRC[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – TRIGSRC[7:0]: Trigger Source Select**
  These bits select which trigger source is used for triggering a transfer on the EDMA standard channel. A zero value means that the trigger source is disabled. Table 5-18 on page 71 shows the peripherals and triggers which are supported by an EDMA standard channel. For modules or peripherals which do not exist for a device, the transfer trigger does not exist. Refer to the device datasheet for the list of peripherals available.
  If the interrupt flag related to the trigger source is cleared or the interrupt level enabled so that an interrupt is triggered, the EDMA request will be lost. Since an EDMA request can clear the interrupt flag, interrupts can be lost.

Table 5-18. EDMA trigger source base values for all modules and peripherals.

TRIGSRC base value	Group configuration	Description
0x00	OFF	Software triggers only (see TRFREQ bit in "CTRLA – Control register A" on page 61)
0x01	SYS	Event system EDMA triggers base value
0x10	ADCA	ADCA EDMA triggers base value
0x15	DACA	DACA EDMA triggers base value
0x40	TCC4	Timer/counter C4 EDMA triggers base value
0x46	TCC5	Timer/counter C5 EDMA triggers base value
0x4A	SPIC	SPI C EDMA triggers base value
0x4C	USARTC0	USART C0 EDMA triggers base value
0x66	TCD5	Timer/counter D5 EDMA triggers base value
0x6C	USARTD0	USART D0 EDMA triggers base value

Table 5-19. EDMA trigger source offset values for ADC triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	CH0	ADC channel 0

Table 5-20. EDMA trigger source offset values for DAC triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	CH0	DAC channel 0
+0x01	CH1	DAC channel 1

Table 5-21. EDMA trigger source offset values for event system triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	CH0	Event channel 0
+0x01	CH1	Event channel 1
+0x02	CH2	Event channel 2

Table 5-22. EDMA trigger source offset values for event system triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	OVF	Overflow/underflow
+0x01	ERR	Error
+0x02	CCA	Compare or capture channel A
+0x03	CCB	Compare or capture channel B
+0x04	CCC(1)	Compare or capture channel C
+0x05	CCD(1)	Compare or capture channel D

Note: 1. CC channel C and D triggers are available only for timer/counters 4.

Table 5-23. EDMA trigger source offset values for USART triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	RXC	Receive complete
+0x01	DRE	Data register empty

Table 5-24. EDMA trigger source offset values for SPI triggers.

TRIGSRC offset value	Group configuration	Description
+0x00	IFRXC	- Transfer complete in standard mode - Receive complete in double buffer mode
+0x01	IFDRE	- Transfer complete in standard mode - Data register empty in double buffer mode

The group configuration is the “base_offset;” for example, TCC5_CCA for the timer/counter C5 CC channel A the transfer trigger.

5.15.6 TRFCNTL – Block Transfer Count register Low

The TRFCNTH and TRFCNTL register pair represents the 16-bit value TRFCNT. TRFCNT defines the number of bytes in a block transfer. The value of TRFCNT is decremented after each byte read by the EDMA standard channel.

The default value of this 16-bit register is 0x0101 (not 0x0001). If the user writes 0x0000 to this 16-bit register and fires an EDMA trigger, EDMA will perform 65536 transfers or searches.

When TRFCNT reaches zero, the register is reloaded with the last value written to it.

Bit	7	6	5	4	3	2	1	0
+0x06	TRFCNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

- **Bit 7:0 – TRFCNT[7:0]: Block Transfer Count Low Byte**
  These bits hold the low-byte of the 16-bit block transfer count.

5.15.7 TRFCNTH – Block Transfer Count register High

Bit	7	6	5	4	3	2	1	0
+0x07	TRFCNT [15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

- **Bit 7:0 – TRFCNT[15:8]: Block Transfer Count High Byte**
  These bits hold the high-byte of the 16-bit block transfer count.

5.15.8 SRCADDRL – Source Address register Low

SRCADDRL and SRCADDRH represent the16-bit value SRCADDR, which is the source address in a transaction executed by a standard channel. SRCADDRH is the most significant byte in the register. SRCADDR may be automatically incremented based on settings in the SRCDIR bits in “SRCADDCTRL – Source Address Control register” .

Bit	7	6	5	4	3	2	1	0
+0x08	SRCADDR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – SRCADDR[7 :0]: Source Address Low Byte**
  These bits hold the low-byte of the 16-bit source address.

5.15.9 SRCADDRH – Source Address register High

Bit	7	6	5	4	3	2	1	0
+0x09	SRCADDR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – SRCADDR[15:8]: Source Address High Byte**
  These bits hold the high-byte of the 16-bit source address.

5.15.10 DESTADDRL – Destination Address register Low

DESTADDRL and DESTADDRH represent the16-bit value DESTADDR, which is the destination address in a transaction executed by a standard channel. DESTADDRH is the most significant byte in the register. DESTADDR may be automatically incremented based on settings in the DESTDIR bits in “DESTADDCTRL – Destination Address Control register” .

In data search mode, DESTADDR is used for data to recognize according to Table 5-17 on page 70.

Bit	7	6	5	4	3	2	1	0
+0x08	DESTADDR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DESTADDR[7 :0]: Destination Address Low Byte**
  These bits hold the low-byte of the 16-bit destination address.

5.15.11 DESTADDRH – Destination Address register High

Bit	7	6	5	4	3	2	1	0
+0x09	DESTADDR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DESTADDR[15:8]: Destination Address High Byte**
  These bits hold the high-byte of the 16-bit destination address.

5.16 Register summary – EDMA controller in PER0123 configuration.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	ENABLE	RESET	CHMODE[1:0]		DBUFMODE[1:0]		PRIMODE[1:0]		59
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	INTFLAGS	CH3ERRIF	CH2ERRIF	CH1ERRIF	CH0ERRIF	CH3TRNFIF	CH2TRNFIF	CH1TRNFIF	CH0TRNFIF	60
+0x04	STATUS	CH3BUSY	CH2BUSY	CH1BUSY	CH0BUSY	CH3PEND	CH2PEND	CH1PEND	CH0PEND	60
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TEMP	TEMP[7:0]								61
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x10/0x1F	CH0	Register addresses for EDMA peripheral channel 0								
+0x20/0x2F	CH1	Register addresses for EDMA peripheral channel 1								
+0x30/0x3F	CH2	Register addresses for EDMA peripheral channel 2								
+0x40/0x4F	CH3	Register addresses for EDMA peripheral channel 3								

5.17 Register summary – EDMA controller in STD0 configuration.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	ENABLE	RESET	CHMODE[1:0]		DBUFMODE[1:0]		PRIMODE[1:0]		59
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	INTFLAGS	CH3ERRIF	CH2ERRIF	–	CH0ERRIF	CH3TRNFIF	CH2TRNFIF	–	CH0TRNFIF	60
+0x04	STATUS	CH3BUSY	CH2BUSY	–	CH0BUSY	CH3PEND	CH2PEND	–	CH0PEND	60
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TEMP	TEMP[7:0]								61
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x10/0x1F	CH0	Register addresses for EDMA standard channel 0								
+0x20/0x2F	Reserved	–	–	–	–	–	–	–	–	
+0x30/0x3F	CH2	Register addresses for EDMA peripheral channel 2								
+0x40/0x4F	CH3	Register addresses for EDMA peripheral channel 3								

5.18 Register summary – EDMA controller in STD2 configuration.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	bit 0	Page
+0x00	CTRL	ENABLE	RESET	CHMODE[1:0]		DBUFMODE[1:0]		PRIMODE[1:0]		59
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	INTFLAGS	–	CH2ERRIF	CH1ERRIF	CH0ERRIF	–	CH2TRNFIF	CH1TRNFIF	CH0TRNFIF	60
+0x04	STATUS	–	CH2BUSY	CH1BUSY	CH0BUSY	–	CH2PEND	CH1PEND	CH0PEND	60
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TEMP	TEMP[7:0]								61
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x10/0x1F	CH0	Register addresses for EDMA peripheral channel 0								
+0x20/0x2F	CH1	Register addresses for EDMA peripheral channel 1								
+0x30/0x3F	CH2	Register addresses for EDMA standard channel 2								
+0x40/0x4F	Reserved	–	–	–	–	–	–	–	–	

5.19 Register summary – EDMA controller in STD02 configuration.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	ENABLE	RESET	CHMODE[1:0]		DBUFMODE[1:0]		PRIMODE[1:0]		59
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	INTFLAGS	–	CH2ERRIF	–	CH0ERRIF	–	CH2TRNFIF	–	CH0TRNFIF	60
+0x04	STATUS	–	CH2BUSY	–	CH0BUSY	–	CH2PEND	–	CH0PEND	60
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TEMP	TEMP[7:0]								61
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x10/0x1F	CH0	Register addresses for EDMA peripheral channel 0								
+0x20/0x2F	Reserved	–	–	–	–	–	–	–	–	
+0x30/0x3F	CH2	Register addresses for EDMA standard channel 2								
+0x40/0x4F	Reserved	–	–	–	–	–	–	–	–	

5.20 Register summary – EDMA peripheral channel.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	ENABLE	RESET	REPEAT	TRFREQ	–	SINGLE	–	BURSTLEN	61
+0x01	CTRLB	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]		62
+0x02	ADDCTRL	–	–	RELOAD[1:0]		–	DIR[2:0]			63
+0x03	Reserved	–	–	–	–	–	–	–	–	
+0x04	TRIGSRC	TRIGSRC[7:0]								64
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TRFCNTL	TRFCNT[7:0]								66
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	ADDRL	ADDR[7:0]								66
+0x09	ADDRH	ADDR[15:8]								66
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	Reserved	–	–	–	–	–	–	–	–	
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	

5.21 Register Summary – EDMA standard channel.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	ENABLE	RESET	REPEAT	TRFREQ	–	SINGLE	–	BURSTLEN	67
+0x01	CTRLB	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]		68
+0x02	SRCADDCTRL	–	–	SRCRELOAD[1:0]		–	SRCDIR[2:0]			68
+0x03	DESTADDCTRL	–	–	DESTRELOAD[1:0]		–	DESTDIR[2:0]			69
+0x04	TRIGSRC	TRIGSRC[7:0]								71
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TRFCNTL	TRFCNT[7:0]								73
+0x07	TRFCNTH	TRFCNT[15:8]								73
+0x08	SRCADDRL	SRCADDR[7:0]								73
+0x09	SRCADDRH	SRCADDR[15:8]								73
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	DESTADDRL	DESTADDR[7:0]								74
+0x0D	DESTADDRH	DESTADDR[15:8]								74
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	

5.22 Interrupt vector summary

Table 5-25. EDMA interrupt vectors and their word offset addresses from the EDMA controller interrupt base.

Offset	Source	Interrupt description
0x00	CH0_vect	EDMA controller channel 0 interrupt vector
0x02	CH1_vect	EDMA controller channel 1 interrupt vector
0x04	CH2_vect	EDMA controller channel 2 interrupt vector
0x06	CH3_vect	EDMA controller channel 3 interrupt vector

6. Event System

6.1 Features

- System for direct peripheral-to-peripheral communication and signaling
- Peripherals can directly send, receive, and react to peripheral events
  - CPU and EDMA controller independent operation
  - 100% predictable signal timing
  - Short and guaranteed response time
- Eight event channels for up to eight different and parallel signal routings and configurations
- Events can be sent and/or used by most peripherals, clock system, and software
- Additional functions include
  - Quadrature decoders with rotary filtering
  - Digital filtering of I/O pin state with flexible prescaler clock options
  - Simultaneous synchronous and asynchronous events provided to peripheral
- Works in all sleep modes

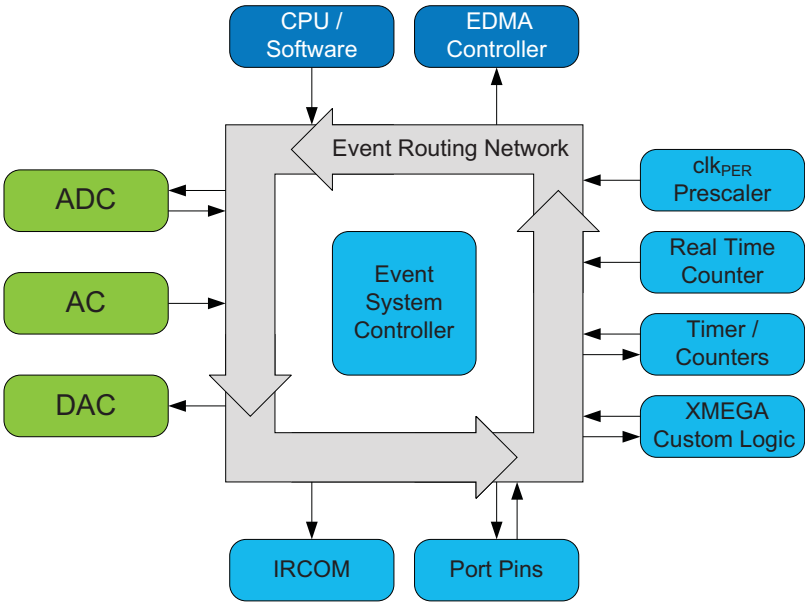
6.2 Overview

The event system enables direct peripheral-to-peripheral communication and signaling. It allows a change in one peripheral's state to automatically trigger actions in other peripherals. It is designed to provide a predictable system for short and predictable response times between peripherals. It allows for autonomous peripheral control and interaction without the use of interrupts, CPU, or EDMA controller resources, and is thus a powerful tool for reducing the complexity, size and execution time of application code. It allows for synchronized timing of actions in several peripheral modules. The event system enables also asynchronous event routing for instant actions in peripherals.

A change in a peripheral's state is referred to as an event, and usually corresponds to the peripheral's interrupt conditions. Events can be directly passed to other peripherals using a dedicated routing network called the event routing network. How events are routed and used by the peripherals is configured in software.

Figure 6-1 on page 80 shows a basic diagram of all connected peripherals. The event system can directly connect together analog to digital converters, analog comparators, I/O port pins, the real-time counter, timer/counters, IR communication module (IRCOM) and XMEGA Custom Logic (XCL). It can also be used to trigger EDMA transactions (EDMA controller). Events can also be generated from software and peripheral clock.

Figure 6-1. Event system overview and connected peripherals.



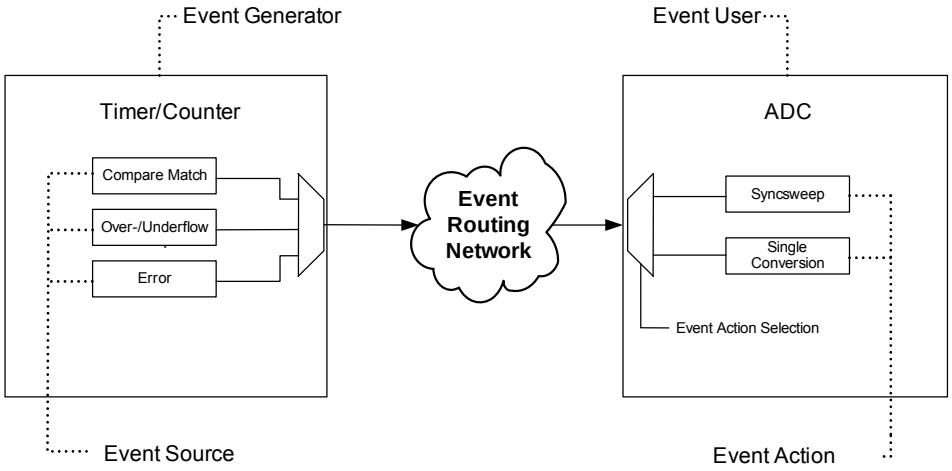
The event routing network consists of eight software-configurable multiplexers that control how events are routed and used. These are called event channels, and allow up to eight parallel event configurations and routings. The maximum routing latency of an external synchronous event is two peripheral clock cycles due to re-synchronization, but several peripherals can directly use the asynchronous event without any clock delay. The event system works in all sleep modes, but only asynchronous events can be routed in sleep modes where the system clock is not available.

6.3 Events

In the context of the event system, an indication that a change of state within a peripheral has occurred is called an event. There are three main types of events: signaling events, synchronous data events and asynchronous data events. Signaling events only indicate a change of state while data events contain additional information about the event.

The peripheral from which the event originates is called the event generator. Within each peripheral (for example, a timer/counter), there can be several event sources, such as a timer compare match or timer overflow. The peripheral using the event is called the event user, and the action that is triggered is called the event action.

Figure 6-2. Example of event source, generator, user and action.



Events can also be generated manually in software.

6.4 Signaling events

Signaling events are the most basic type of event. A signaling event does not contain any information apart from the indication of a change in a peripheral. Most peripherals can only generate and use signaling events. Unless otherwise stated, all occurrences of the word ”event” are to be understood as meaning signaling events, which is a strobe.

6.5 Data events

Data events differ from signaling events in that they contain information that event users can decode to decide event actions based on the receiver information. Data events can be synchronous or asynchronous.

Although the event routing network can route all events to all event users, those that are only meant to use signaling events do not have decoding capabilities needed to utilize data events.

How event users decode data events is shown in Table 6-1 on page 81.

Event users that can utilize data events can also use signaling events. This is configurable, and is described in the datasheet module for each peripheral.

6.6 Peripheral clock events

Each event channel includes a peripheral clock prescaler with a range from 1 (no prescaling) to 32768. This enables configurable periodic event generation based on the peripheral clock. It is possible to periodically trigger events in a peripheral or to periodically trigger synchronized events in several peripherals. Since each event channel include a prescaler, different peripherals can receive triggers with different intervals.

6.7 Software events

Events can be generated from software by writing the DATA and STROBE registers. The DATA register must be written first, since writing the STROBE register triggers the operation. The DATA and STROBE registers contain one bit for each event channel. Bit n corresponds to event channel n. It is possible to generate events on several channels at the same time by writing to several bit locations at once.

Software-generated events last for one clock cycle and will overwrite events from other event generators on that event channel during that clock cycle.

Table 6-1 on page 81 shows the different events, how they can be manually generated, and how they are decoded.

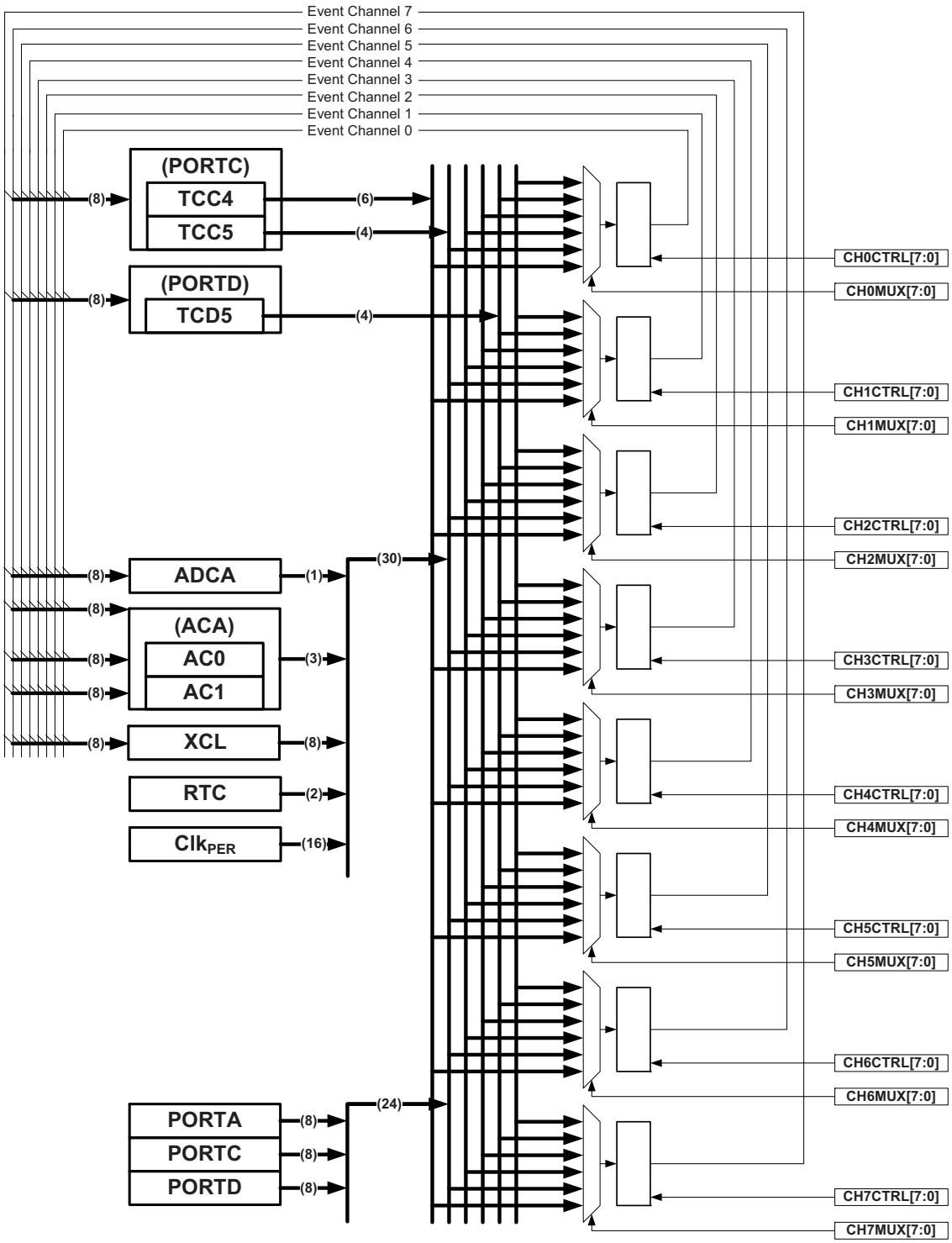
Table 6-1. Manually generated events and decoding of events.

STROBE	DATA	Data event user	Signaling event user
0	0	No Event	No Event
0	1	Data Event 01	No Event
1	0	Data Event 02	Signaling Event
1	1	Data Event 03	Signaling Event

6.8 Event routing network

The event routing network routes the events between peripherals. It consists of eight multiplexers (CHnMUX), which can each be configured to route any event source to any event users. The output from a multiplexer is referred to as an event channel. For each peripheral, it is selectable if and how incoming events should trigger event actions. Details on configurations can be found in the datasheet for each peripheral. The event routing network is shown in Figure 6-3 on page 82.

Figure 6-3. Event routing network.



Eight multiplexers means that it is possible to route up to eight events at the same time. It is also possible to route one event through several multiplexers.

Not all XMEGA devices contain all peripherals. This only means that a peripheral is not available for generating or using events. The network configuration itself is compatible between all devices.

Event selection for each channel and event type is shown in Table 6-2 on page 83:

Table 6-2. Event selection and event type.

		Event type		
Peripheral	**Event source**	**Strobe event**	**Synchronous data**	**Asynchronous data**
RTC	RTC_OVF	x		x
	RTC_CMP	x		x
AC	AC_CH0	x		x
	AC_CH1	x		x
	AC_WIN	x		
ADC	ADC_CH	x		
PRESCALER	PRESC_M	x		
PORTn	PORTn_PIN0	x	x	x
	PORTn_PIN1	x	x	x
	PORTn_PIN2	x	x	x
	PORTn_PIN3	x	x	x
	PORTn_PIN4	x	x	x
	PORTn_PIN5	x	x	x
	PORTn_PIN6	x	x	x
	PORTn_PIN7	x	x	x
TC4	TC4_OVF	x		
	TC4_ERR	x		
	TC4_CCA	x		
	TC4_CCB	x		
	TC4_CCC	x		
	TC4_CCD	x		
TC5	TC5_OVF	x		
	TC5_ERR	x		
	TC5_CCA	x		
	TC5_CCB	x		

		Event type		
Peripheral	**Event source**	**Strobe event**	**Synchronous data**	**Asynchronous data**
XCL	XCL_UNF0	x		
	XCL_UNF1	x		
	XCL_CC0	x		
	XCL_CC1	x		
	XCL_PEC0	x		
	XCL_PEC1	x		
	XCL_LUT0	x	x	x
	XCL_LUT1	x	x	x

6.9 Event timing

An event normally lasts for one peripheral clock cycle, but some event sources, such as a low level on an I/O pin, will generate events continuously. Details on this are described in the datasheet for each peripheral, but unless otherwise stated, an event lasts for one peripheral clock cycle.

It takes a maximum of two peripheral clock cycles from when an event is generated until the event actions in other peripherals are triggered. This ensures short and 100% predictable response times, independent of CPU or EDMA controller load or software revisions.

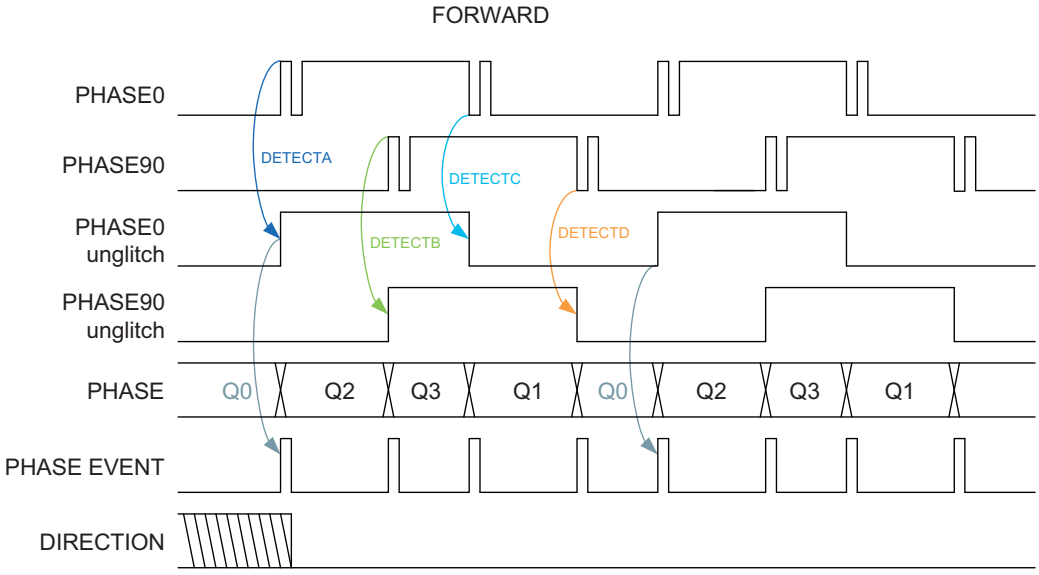
An asynchronous event is routed without any peripheral clock delay and it is present as long as the source is generating this event.

6.10 Filtering

Each event channel includes a digital filter. When this is enabled, an event must be sampled with the same value for a configurable number of system clock or prescaler clock cycles before it is accepted. This is primarily intended for pin change events. The default clock for a digital filter is the system clock. Optionally, the clock can be divided by using the prescaler with individual settings for each channel 0 to channel 3 or channel 4 to channel 7.

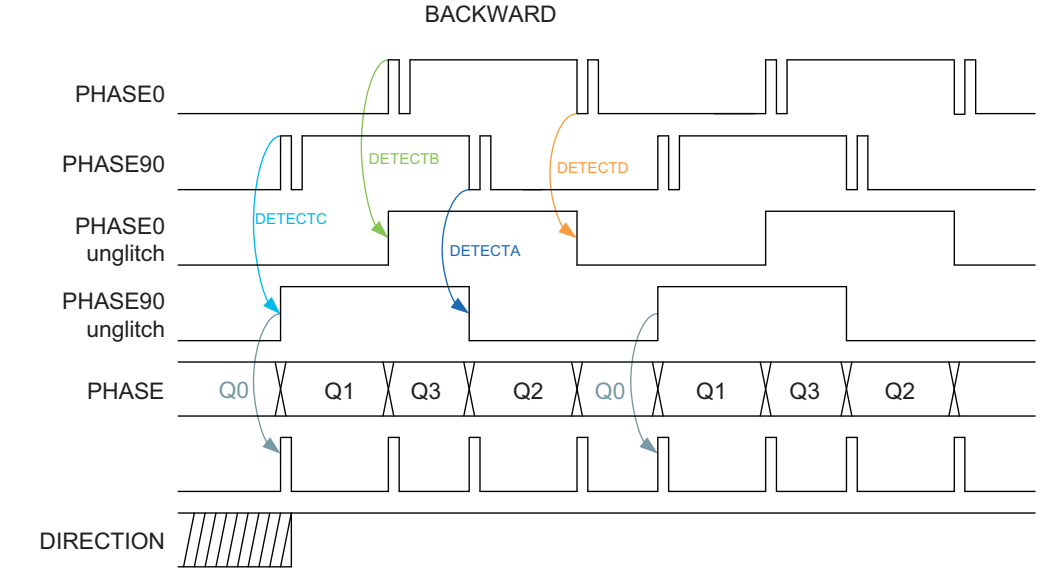
Event channels with quadrature decoder extension support rotary filter. Figure 6-4 on page 85 shows the output signals of the rotary filter. The rotary filter output controls the QDEC up, down and index operation. The digital filter can be enabled when using the rotary encoder.

Figure 6-4. Rotary encoder output signals.



DETECTA = Set PHASE0 unglitch when PHASE0 = 1 and PHASE90 = 0
DETECTB = Set PHASE90 unglitch when PHASE0 = 1 and PHASE90 = 1
DETECTC = Clear PHASE0 unglitch when PHASE0 = 0 and PHASE90 = 1
DETECTD = Clear PHASE90 unglitch when PHASE =0 and PHASE90 = 0

On phase event, update DIRECTION : 0=FORWARD/1=BACKWARD
PH0 rise : PH90
PH0 fall : not(PH90)
PH90 rise : not(PH0)
PH90 fall : PH0



6.11 Quadrature decoder

The event system includes three quadrature decoders (QDECs), which enable the device to decode quadrature input on I/O pins and send data events that a timer/counter can decode to count up, count down, or index/reset. Table 6-3 on page 86 summarizes which quadrature decoder data events are available, how they are decoded by timers, and how they can be generated. The QDECs and related features and control and status registers are available for event channels 0, 2, and 4.

Table 6-3. Quadrature decoder data events.

STROBE	DATA	Data event user	Signaling event user
0	0	No Event	No Event
0	1	Index/reset	No Event
1	0	Count down	Signaling Event
1	1	Count up	Signaling Event

6.11.1 Quadrature operation

A quadrature signal is characterized by having two square waves that are phase shifted 90 degrees relative to each other. Rotational movement can be measured by counting the edges of the two waveforms. The phase relationship between the two square waves determines the direction of rotation.

Figure 6-5. Quadrature signals from a rotary encoder.

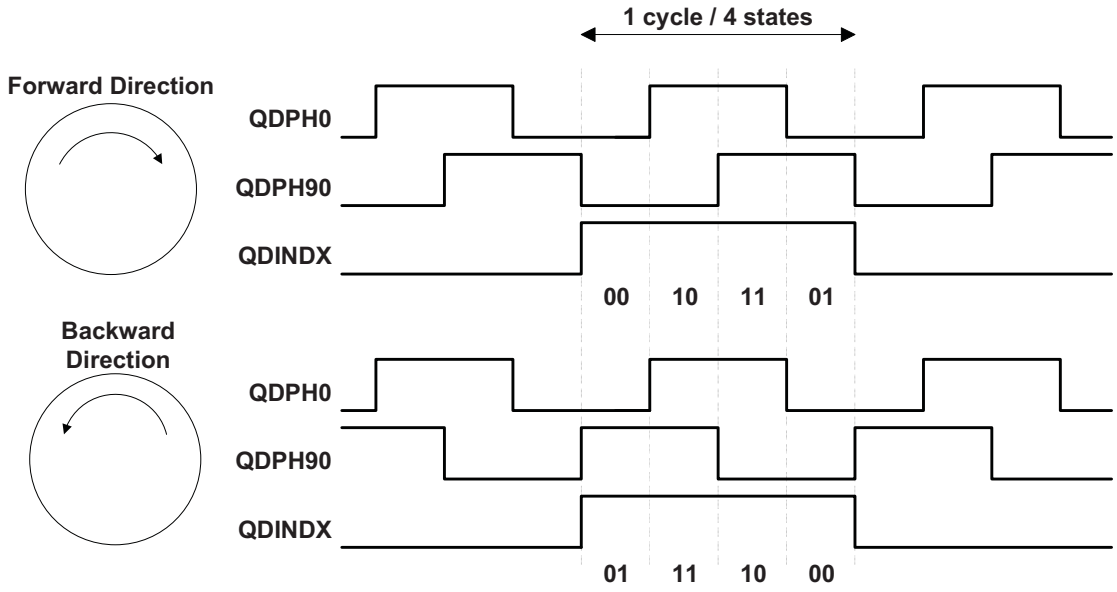


Figure 6-5 on page 86 shows typical quadrature signals from a rotary encoder. The signals QDPH0 and QDPH90 are the two quadrature signals. When QDPH90 leads QDPH0, the rotation is defined as positive or forward. When QDPH0 leads QDPH90, the rotation is defined as negative or reverse. The concatenation of the two phase signals is called the quadrature state or the phase state.

In order to know the absolute rotary displacement, a third index signal (QDINDX) can be used. This gives an indication once per revolution.

6.11.2 QDEC setup

For a full QDEC setup, the following is required:

- Two or three I/O port pins for quadrature signal input
- Two event system channels for quadrature decoding
- One timer/counter for up, down, and optional index count

The following procedure should be used for QDEC setup:

1. Choose two successive pins on a port as QDEC phase inputs.
2. Set the pin direction for QDPH0 and QDPH90 as input.

3. Set the pin configuration for QDPH0 and QDPH90 to low level sense.
4. Select the QDPH0 pin as a multiplexer input for an event channel, n.
5. Enable quadrature decoding and digital filtering in the event channel.
6. Optional:
    1. Set the digital filter control register (DFCTRL) options.
    2. Set up a QDEC index (QINDX).
    3. Select a third pin for QINDX input.
    4. Set the pin direction for QINDX as input.
    5. Set the pin configuration for QINDX to sense both edges.
    6. Select QINDX as a multiplexer input for event channel n+1.
    7. Set the quadrature index enable bit in event channel n+1.
    8. Select the index recognition mode for event channel n+1.
    9. Set quadrature decoding as the event action for a timer/counter.
    10. Select event channel n as the event source for the timer/counter.

- Set the period register of the timer/counter to ('line count' * 4 - 1), the line count of the quadrature encoder.
- Enable the timer/counter without clock prescaling.

The angle of a quadrature encoder attached to QDPH0, QDPH90 (and QINDX) can now be read directly from the timer/counter count register. If the count register is different from BOTTOM when the index is recognized, the timer/counter error flag is set. Similarly, the error flag is set if the position counter passes BOTTOM without the recognition of the index.

6.12 Register description

6.12.1 CHnMUX – Event Channel n Multiplexer register

Bit	7	6	5	4	3	2	1	0
+n	CHnMUX[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CHnMUX[7:0]: Channel Multiplexer**
  These bits select the event source according to Table 6-4 on page 88. This table is valid for all XMEGA devices regardless of whether the peripheral is present or not. Selecting event sources from peripherals that are not present will give the same result as when this register is zero. When this register is zero, no events are routed through. Manually generated events will override CHnMUX and be routed to the event channel even if this register is zero.

Table 6-4. CHnMUX bit settings.

CHnMUX[7:4]	CHnMUX[3:0]				Group configuration	Event source
0000	0	0	0	0		None (manually generated events only)
0000	0	0	0	1		(Reserved)
0000	0	0	1	x		(Reserved)
0000	0	1	x	x		(Reserved)
0000	1	0	0	0	RTC_OVF	RTC overflow
0000	1	0	0	1	RTC_CMP	RTC compare match
0000	1	0	1	x		(Reserved)
0000	1	1	x	x		(Reserved)
0001	0	0	0	0	ACA_CH0	ACA channel 0
0001	0	0	0	1	ACA_CH1	ACA channel 1
0001	0	0	1	0	ACA_WIN	ACA window
0001	0	x	x	x		(Reserved)
0010	0	0	0	0	ADCA_CH	ADCA channel
0010	x	x	x	x		(Reserved)
0011	x	x	x	X		(Reserved)
0100	x	x	x	x		(Reserved)
0101	0	n			PORTA_PINn(1)	PORTA pin n (n = 0,1,2 … or 7)
0101	1	x	x	x		(Reserved)
0110	0	n			PORTC_PINn(1)	PORTC pin n (n = 0,1,2 … or 7)
0110	1	n			PORTD_PINn(1)	PORTD pin n (n = 0,1,2 … or 7)
0111	x	x	x	x		(Reserved)
1000	M				PRESCALER_M	Clk_{PER} divide by 2^M (M = 0 to 15)

CHnMUX[7:4]	CHnMUX[3:0]				Group configuration	Event source
1001	x	x	x	x		(Reserved)
1010	x	x	x	x		(Reserved)
1011	0	E			See Table 6-5	XCL event type E
1011	1	x	x	x		(Reserved)
1100	0	E			See Table 6-6	Timer/counter C4 event type E
1100	1	E			See Table 6-6	Timer/counter C5 event type E
1101	0	x	X	x		(Reserved)
1101	1	E			See Table 6-6	Timer/counter D5 event type E
1110	x	x	x	x		(Reserved)
1111	x	x	x	x		(Reserved)

Note: 1. The description of how the ports generate events is described in "Port event" on page 146.

Table 6-5. XCL events.

T/C event E			Group configuration	Event type
0	0	0	XCL_UNF0	BTC0 underflow
0	0	1	XCL_UNF1	BTC1 underflow
0	1	0	XCL_CC0	BTC0 capture or compare
0	1	1	XCL_CC1	BTC1 capture or compare
1	0	0	XCL_PEC0	PEC0 restart
1	0	1	XCL_PEC1	PEC1 restart
1	1	0	XCL_LUT0	LUT0 output
1	1	1	XCL_LUT1	LUT1 output

Table 6-6. Timer/counter events.

T/C event E			Group configuration	Event type
0	0	0	TCxn_OVF	Over/Underflow (x = C,D)(n = 4 or 5)
0	0	1	TCxn_ERR	Error (x = C,D)(n = 4 or 5)
0	1	x		(Reserved)
1	0	0	TCxn_CCA	Capture or compare A (x = C,D)(n = 4 or 5)
1	0	1	TCxn_CCB	Capture or compare B (x = C,D)(n = 4 or 5)
1	1	0	TCxn_CCC	Capture or compare C (x = C)(n = 4)
1	1	1	TCxn_CCD	Capture or compare D (x = C)(n = 4)

6.12.2 CHnCTRL – Event Channel n Control register

Bit	7	6	5	4	3	2	1	0
+8x08 +n	ROTARY	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – ROTARY: Rotary**
  Setting this bit enables rotary filter. This bit is available only for CH0CTRL.
- **Bit 6:5 – QDIRM[1:0]: Quadrature Decode Index Recognition Mode**
  These bits determine the quadrature state for the QDPH0 and QDPH90 signals, where a valid index signal is recognized and the counter index data event is given according to Table 6-7 on page 90. These bits should only be set when a quadrature encoder with a connected index signal is used. These bits are available only for CH0CTRL.

Table 6-7. QDIRM bit settings.

QDIRM[1:0]		Index recognition state
0	0	{QDPH0, QDPH90} = 0b00
0	1	{QDPH0, QDPH90} = 0b01
1	0	{QDPH0, QDPH90} = 0b10
1	1	{QDPH0, QDPH90} = 0b11

- **Bit 4 – QDIEN: Quadrature Decode Index Enable**
  When this bit is set, the event channel will be used as a QDEC index source, and the index data event will be enabled.
  This bit is available only for CH0CTRL.
- **Bit 3 – QDEN: Quadrature Decode Enable**
  Setting this bit enables QDEC operation. This bit is ignored if the rotary encoder is enabled.
  This bit is available only for CH0CTRL.
- **Bit 2:0 – DIGFILT[2:0]: Digital Filter Coefficient**
  These bits define the length of digital filtering used. Events will be passed through to the event channel only when the event source has been active and sampled with the same level for the number of prescaler peripheral clock cycles defined by DIGFILT.

Table 6-8. Digital filter coefficient values.

DIGFILT[2:0]	Group configuration	Description
000	1SAMPLE	One sample
001	2SAMPLES	Two samples
010	3SAMPLES	Three samples
011	4SAMPLES	Four samples
100	5SAMPLES	Five samples
101	6SAMPLES	Six samples
110	7SAMPLES	Seven samples
111	8SAMPLES	Eight samples

6.12.3 STROBE – Event Strobe register

If the STROBE register location is written, each event channel will be set according to the STROBE[n] and corresponding DATA[n] bit settings, if any are unequal to zero.

A single event lasting for one peripheral clock cycle will be generated.

Bit	7	6	5	4	3	2	1	0
+0x10	STROBE[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

6.12.4 DATA – Event DATA register

This register contains the data value when manually generating a data event. This register must be written before the STROBE register. For details, see “STROBE – Event Strobe register” on page 91.

Bit	7	6	5	4	3	2	1	0
+0x11	DATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

6.12.5 DFCTRL – Digital Filter Control register

Bit	7	6	5	4	3	2	1	0
+0x12	PRESCFILT[3:0]				FILTSEL	PRESC[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – PRESCFILT[3:0]: Prescaler Filter**
  These bits define the prescaler filter settings, according to Table 6-9 on page 91.

Table 6-9. Prescaler filter settings.

PRESCFILT[3:0]	Group configuration	Description
xxx1	CH04	Enable prescaler filter for either channel 0 or 4
xx1x	CH15	Enable prescaler filter for either channel 1 or 5
x1xx	CH26	Enable prescaler filter for either channel 2 or 6
1xxx	CH37	Enable prescaler filter for either channel 3 or 7

- **Bit 3 – FILTSEL: Prescaler Filter Select**
  Setting this bit enables the prescaler clock option on event channels 4 to 7. Clearing this bit enables the prescaler clock option on event channels 0 to 3. This bit is used with settings defined by PRESCFILT bits.
- **Bit 2:0 – PRESC[2:0]: Prescaler**
  These bits select the digital filter clock prescaler settings, according to Table 6-10 on page 92.

Table 6-10. Prescaler options.

PRESC[2:0]	Group configuration	Description
000	CLKPER_8	Clk_{PER} divide by 2^3
001	CLKPER_64	Clk_{PER} divide by 2^6
010	CLKPER_512	Clk_{PER} divide by 2^9
011	CLKPER_4096	Clk_{PER} divide by 2^{12}
100	CLKPER_32768	Clk_{PER} divide by 2^{15}
101	–	(Reserved)
110	–	(Reserved)
1111	–	(Reserved)

6.13 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CH0MUX	CH0MUX[7:0]								88
+0x01	CH1MUX	CH1MUX[7:0]								88
+0x02	CH2MUX	CH2MUX[7:0]								88
+0x03	CH3MUX	CH3MUX[7:0]								88
+0x04	CH4MUX	CH4MUX[7:0]								88
+0x05	CH5MUX	CH5MUX[7:0]								88
+0x06	CH6MUX	CH6MUX[7:0]								88
+0x07	CH7MUX	CH7MUX[7:0]								88
+0x08	CH0CTRL	ROTARY	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			90
+0x09	CH1CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x0A	CH2CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x0B	CH3CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x0C	CH4CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x0D	CH5CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x0E	CH6CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x0F	CH7CTRL	-	-	-	-	-	DIGFILT[2:0]			90
+0x10	STROBE	STROBE[7:0]								91
+0x11	DATA	DATA[7:0]								91
+0x12	DFCTRL	PRESCFILT[3:0]				FILTSEL	PRESC[2:0]			91

7. System Clock and Clock Options

7.1 Features

- Fast start-up time
- Safe run-time clock switching
- Internal oscillators:
  - 32MHz run-time calibrated oscillator
  - 8MHz calibrated oscillator with 2MHz output and fast start-up
  - 32.768kHz calibrated oscillator
  - 32kHz ultra low power (ULP) oscillator with 1kHz output
- External clock options
  - 0.4MHz - 16MHz crystal oscillator
  - 32.768kHz crystal oscillator
  - External clock
- PLL with 20MHz - 128MHz output frequency
  - Internal and external clock options and 1× to 31× multiplication
  - Lock detector
- Clock prescalers with 1× to 2048× division
- Fast peripheral clocks running at 2 and 4 times the CPU clock
- Automatic run-time calibration of internal 32MHz oscillator
- External oscillator and PLL lock failure detection with optional non-maskable interrupt

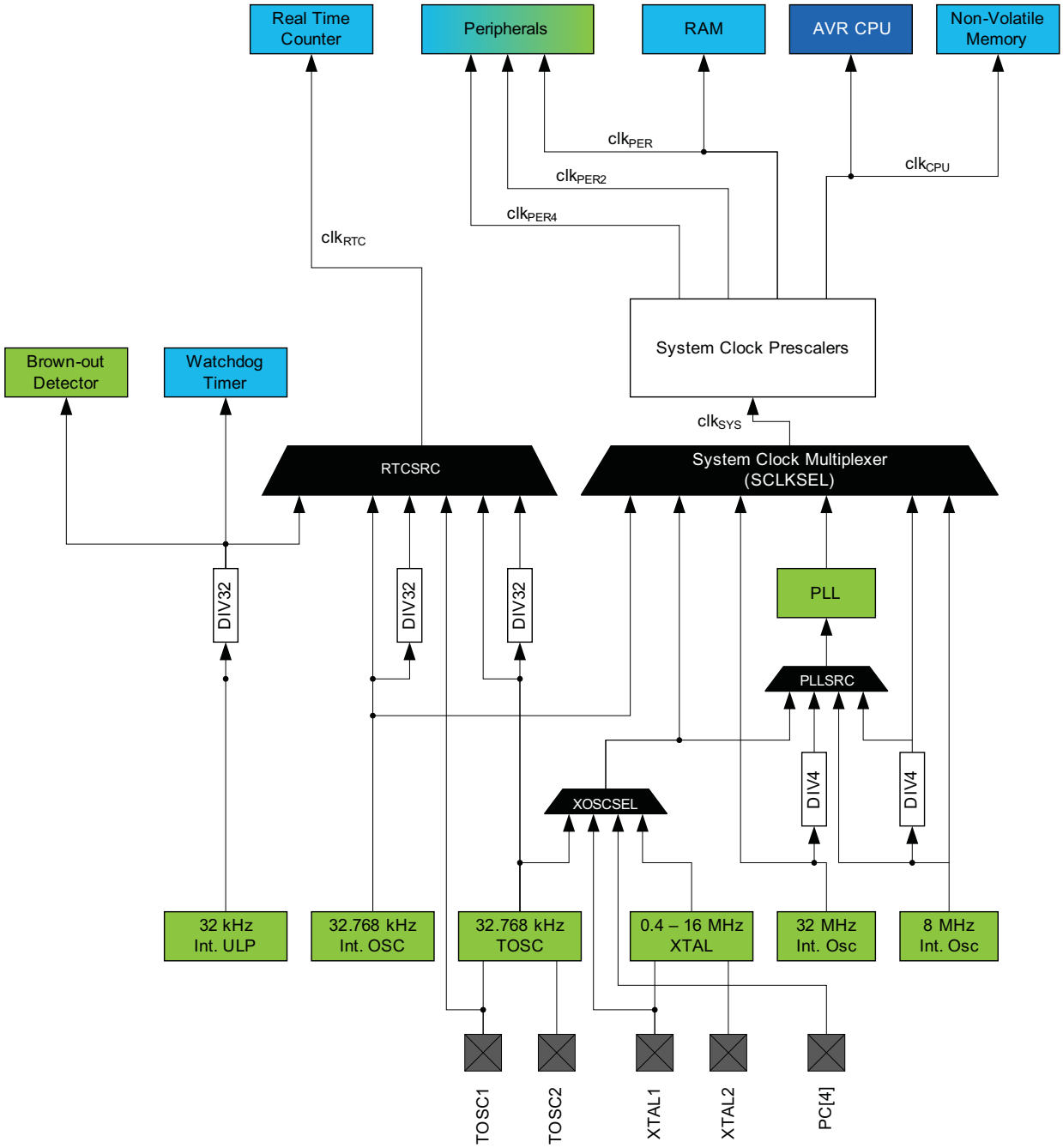
7.2 Overview

XMEGA devices have a flexible clock system supporting a large number of clock sources. It incorporates both accurate internal oscillators and external crystal oscillator and resonator support. A high-frequency phase locked loop (PLL) and clock prescalers can be used to generate a wide range of clock frequencies. A calibration feature (DFLL) is available, and can be used for automatic run-time calibration of the internal oscillators to remove frequency drift over voltage and temperature. An oscillator failure monitor can be enabled to issue a non-maskable interrupt and switch to the internal oscillator if the external oscillator or PLL fails.

When a reset occurs, all clock sources except the 32kHz ultra low power oscillator are disabled. After reset, the device will always start up running from the 2MHz output of 8MHz internal oscillator. During normal operation, the system clock source and prescalers can be changed from software at any time.

Figure 7-1 on page 95 presents the principal clock system in the XMEGA family of devices. Not all of the clocks need to be active at a given time. The clocks for the CPU and peripherals can be stopped using sleep modes and power reduction registers, as described in “Power Management and Sleep Modes” on page 112.

Figure 7-1. The clock system, clock sources and clock distribution.



7.3 Clock distribution

Figure 7-1 presents the principal clock distribution system used in XMEGA devices.

7.3.1 System clock - Clk_{SYS}

The system clock is the output from the main system clock selection. This is fed into the prescalers that are used to generate all internal clocks except the asynchronous clocks.

7.3.2 CPU clock – Clk_{CPU}

The CPU clock is routed to the CPU and nonvolatile memory. Halting the CPU clock inhibits the CPU from executing instructions.

7.3.3 Peripheral clock – Clk_{PER}

The majority of peripherals and system modules use the peripheral clock. This includes the DMA controller, event system, interrupt controller, external bus interface and RAM. This clock is always synchronous to the CPU clock, but may run even when the CPU clock is turned off.

7.3.4 Peripheral 2x/4x clocks – Clk_{PER2}/Clk_{PER4}

Modules that can run at two or four times the CPU clock frequency can use the peripheral 2× and peripheral 4× clocks.

7.3.5 Asynchronous clock – Clk_{RTC}

The asynchronous clock allows the real-time counter (RTC) to be clocked directly from an external 32.768kHz crystal oscillator or the 32 times prescaled output from the internal 32.768kHz oscillator or ULP oscillator. The dedicated clock domain allows operation of this peripheral even when the device is in sleep mode and the rest of the clocks are stopped.

7.4 Clock sources

The clock sources are divided in two main groups: internal oscillators and external clock sources. Most of the clock sources can be directly enabled and disabled from software, while others are automatically enabled or disabled, depending on peripheral settings. After reset, the device starts up running from the 2MHz output of 8MHz internal oscillator. The other clock sources, DFLL and PLL, are turned off by default.

7.4.1 Internal oscillators

The internal oscillators do not require any external components to run. For details on characteristics and accuracy of the internal oscillators, refer to the device datasheet.

7.4.1.1 32kHz ultra low power oscillator

This oscillator provides an approximate 32kHz clock. The 32kHz ultra low power (ULP) internal oscillator is a very low power clock source, and it is not designed for high accuracy. The oscillator employs a built-in prescaler that provides a 1kHz output. The oscillator is automatically enabled/disabled when it is used as clock source for any part of the device. This oscillator can be selected as the clock source for the RTC.

7.4.1.2 32.768kHz calibrated oscillator

This oscillator provides an approximate 32.768kHz clock. It is calibrated during production to provide a default frequency close to its nominal frequency. The calibration register can also be written from software for run-time calibration of the oscillator frequency. The oscillator employs a built-in prescaler, which provides both a 32.768kHz output and a 1.024kHz output.

7.4.1.3 32MHz run-time calibrated oscillator

The 32MHz run-time calibrated internal oscillator is a high-frequency oscillator. It is calibrated during production to provide a default frequency close to its nominal frequency. A digital frequency looked loop (DFLL) can be enabled for automatic run-time calibration of the oscillator to compensate for temperature and voltage drift and optimize the oscillator accuracy. This oscillator can also be adjusted and calibrated to any frequency between 30MHz and 55MHz.

7.4.1.4 8MHz calibrated oscillator

The 8MHz calibrated internal oscillator is the default system clock source after reset. It is calibrated during production to provide a default frequency close to its nominal frequency.

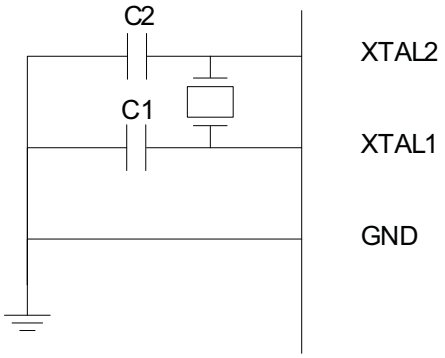
7.4.2 External clock sources

The XTAL1 and XTAL2 pins can be used to drive an external oscillator, either a quartz crystal or a ceramic resonator. XTAL1 or pin 4 from port C (PC4) can be used as input for an external clock signal. The TOSC1 and TOSC2 pins are dedicated to driving a 32.768kHz crystal oscillator.

7.4.2.1 0.4MHz - 16MHz crystal oscillator

This oscillator can operate in four different modes optimized for different frequency ranges, all within 0.4MHz - 16MHz. Figure 7-2 shows a typical connection of a crystal oscillator or resonator.

Figure 7-2. Crystal oscillator connection.

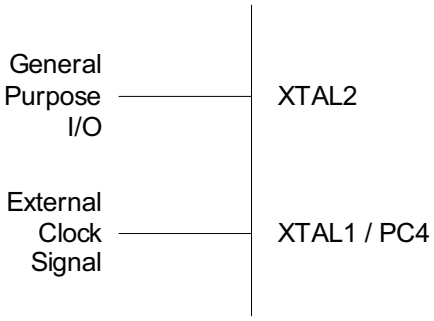


Two capacitors, C1 and C2, may be added to match the required load capacitance for the connected crystal.

7.4.2.2 External clock input

To drive the device from an external clock source, XTAL1 pin must be driven as shown in Figure 7-3. In this mode, XTAL2 can be used as a general I/O pin. Pin 4 from port C can be used as alternative position for external clock input.

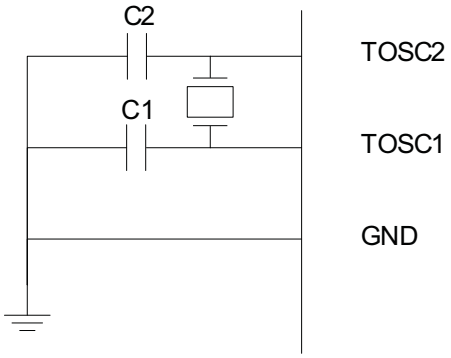
Figure 7-3. External clock drive configuration.



7.4.2.3 32.768kHz crystal oscillator

A 32.768kHz crystal oscillator can be connected between the TOSC1 and TOSC2 pins and enables a dedicated low frequency oscillator input circuit. A typical connection is shown in Figure 7-4 on page 98. A low power mode with reduced voltage swing on TOSC2 is available. This oscillator can be used as a clock source for the system clock and RTC, and as the DFLL reference clock.

Figure 7-4. 32.768kHz crystal oscillator connection.



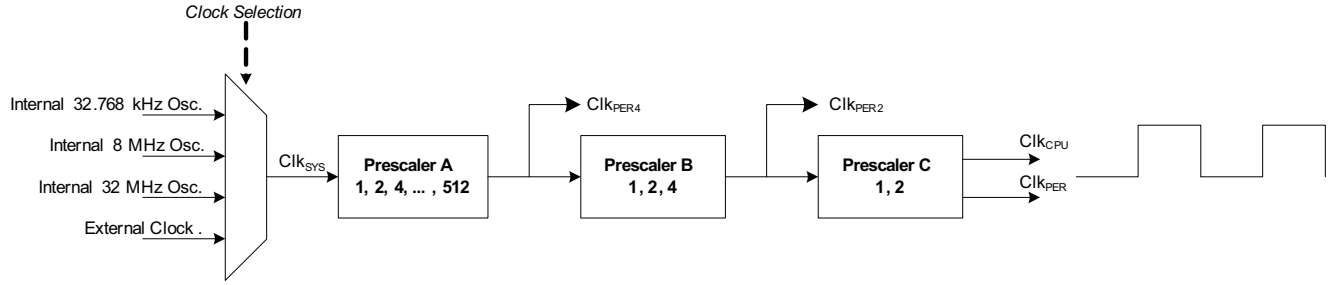
Two capacitors, C1 and C2, may be added to match the required load capacitance for the connected crystal. For details on recommended TOSC characteristics and capacitor load, refer to device datasheet.

7.5 System clock selection and prescalers

All the calibrated internal oscillators, the external clock sources (XOSC), and the PLL output can be used as the system clock source. The system clock source is selectable from software, and can be changed during normal operation. Built-in hardware protection prevents unsafe clock switching. It is not possible to select a non-stable or disabled oscillator as the clock source, or to disable the oscillator currently used as the system clock source. Each oscillator option has a status flag that can be read from software to check that the oscillator is ready.

The system clock is fed into a prescaler block that can divide the clock signal by a factor from 1 to 2048 before it is routed to the CPU and peripherals. The prescaler settings can be changed from software during normal operation. The first stage, prescaler A, can divide by a factor of from 1 to 512. Then, prescalers B and C can be individually configured to either pass the clock through or combine divide it by a factor from 1 to 4. The prescaler guarantees that derived clocks are always in phase, and that no glitches or intermediate frequencies occur when changing the prescaler setting. The prescaler settings are updated in accordance with the rising edge of the slowest clock.

Figure 7-5. System clock selection and prescalers.



Prescaler A divides the system clock, and the resulting clock is clk_{PER4}. Prescalers B and C can be enabled to divide the clock speed further to enable peripheral modules to run at twice or four times the CPU clock frequency. If Prescalers B and C are not used, all the clocks will run at the same frequency as the output from Prescaler A.

The system clock selection and prescaler registers are protected by the configuration change protection mechanism, employing a timed write procedure for changing the system clock and prescaler settings. For details, refer to “Configuration change protection” on page 13.

7.6 PLL with 1x-31x multiplication factor

The built-in phase locked loop (PLL) can be used to generate a high-frequency system clock. The PLL has a user-selectable multiplication factor of from 1 to 31. The output frequency, f_{OUT}, is given by the input frequency, f_{IN}, multiplied by the multiplication factor, PLL_FAC.

$$f_{OUT} = f_{IN} * PLL\_FAC$$

Four different clock sources can be chosen as input to the PLL:

- 2MHz output from 8MHz internal oscillator
- 8MHz internal oscillator
- 32MHz internal oscillator divided by 4
- 0.4MHz - 16MHz crystal oscillator
- External clock

To enable the PLL, the following procedure must be followed:

1. Enable reference clock source.
2. Set the multiplication factor and select the clock reference for the PLL.
3. Wait until the clock reference source is stable.
4. Enable the PLL.

Hardware ensures that the PLL configuration cannot be changed when the PLL is in use. The PLL must be disabled before a new configuration can be written.

It is not possible to use the PLL before the selected clock source is stable and the PLL has locked.

The reference clock source cannot be disabled while the PLL is running.

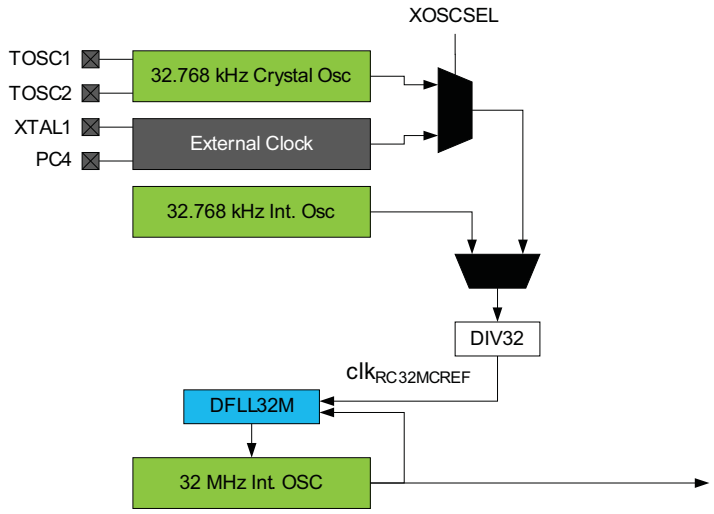
7.7 DFLL 32MHz

Built-in digital frequency locked loop (DFLL) can be used to improve the accuracy of the 32MHz internal oscillators. The DFLL compares the oscillator frequency with a more accurate reference clock to do automatic run-time calibration of the oscillator and compensate for temperature and voltage drift. The choices for the reference clock sources are:

- 32.768kHz calibrated internal oscillator
- 32.768kHz crystal oscillator connected to the TOSC pins
- External clock

The DFLL divides the oscillator reference clock by 32 to use a 1.024kHz reference. The reference clock is individually selected for each DFLL, as shown in Figure 7-6 on page 99.

Figure 7-6. DFLL reference clock selection.

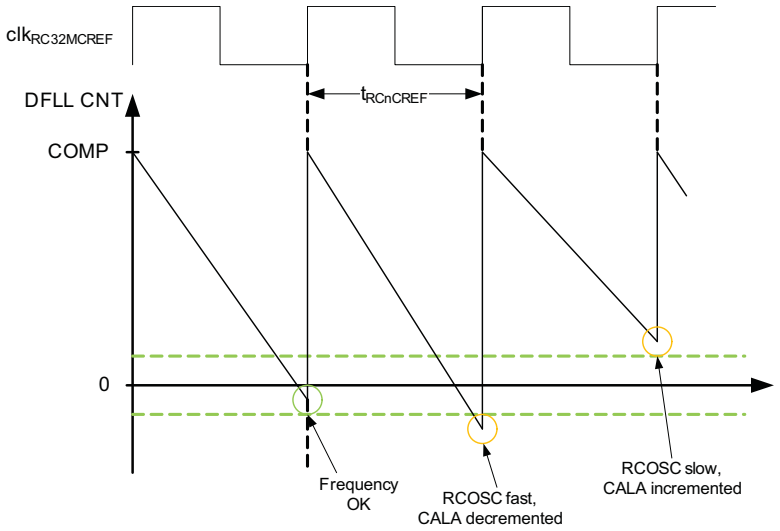


The value that should be written to the COMP register is given by the following formula:

$$COMP = hex(\frac{f_{OSC}}{f_{RC32MCREF}})$$

When the DFLL is enabled, it controls the ratio between the reference clock frequency and the oscillator frequency. If the internal oscillator runs too fast or too slow, the DFLL will decrement or increment its calibration register value by one to adjust the oscillator frequency. The oscillator is considered running too fast or too slow when the error is more than a half calibration step size.

Figure 7-7. Automatic run-time calibration.



The DFLL will stop when entering a sleep mode where the oscillators are stopped. After wake up, the DFLL will continue with the calibration value found before entering sleep. The reset value of the DFLL calibration register can be read from the production signature row.

When the DFLL is disabled, the DFLL calibration register can be written from software for manual run-time calibration of the oscillator.

7.8 PLL and external clock source failure monitor

A built-in failure monitor is available for the PLL and external clock source. If the failure monitor is enabled for the PLL and/or the external clock source, and this clock source fails (the PLL looses lock or the external clock source stops) while being used as the system clock, the device will:

- Switch to run the system clock from the 2MHz output from 8MHz internal oscillator
- Reset the oscillator control register and system clock selection register to their default values
- Set the failure detection interrupt flag for the failing clock source (PLL or external clock)
- Issue a non-maskable interrupt (NMI)

If the PLL or external clock source fails when not being used for the system clock, it is automatically disabled, and the system clock will continue to operate normally. No NMI is issued. The failure monitor is meant for external clock sources above 32kHz. It cannot be used for slower external clocks.

When the failure monitor is enabled, it will not be disabled until the next reset.

The failure monitor is stopped in all sleep modes where the PLL or external clock source, are stopped. During wake up from sleep, it is automatically restarted.

The PLL and external clock source failure monitor settings are protected by the configuration change protection mechanism, employing a timed write procedure for changing the settings. For details, refer to “Configuration change protection” on page 13.

7.9 Register description – Clock

7.9.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	SCLKSEL[2:0]		
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2:0 – SCLKSEL[2:0]: System Clock Selection**
  These bits are used to select the source for the system clock. See Table 7-1 for the different selections. Changing the system clock source will take two clock cycles on the old clock source and two more clock cycles on the new clock source. These bits are protected by the configuration change protection mechanism. For details, refer to “Configuration change protection” on page 13.
  SCLKSEL cannot be changed if the new clock source is not stable. The old clock can not be disabled until the clock switching is completed.

Table 7-1. System clock selection.

SCLKSEL[2:0]	Group configuration	Description
000	RC2MHZ	2MHz from 8MHz internal oscillator
001	RC32MHZ	32MHz internal oscillator
010	RC32KHZ	32.768kHz internal oscillator
011	XOSC	External oscillator or clock
100	PLL	Phase locked loop
101	RC8MHZ	8MHz internal oscillator
110	–	Reserved
111	–	Reserved

7.9.2 PSCTRL – Prescaler register

Bit	7	6	5	4	3	2	1	0
+0x01	–	PSADIV[4:0]					PSBCDIV[1:0]	
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 - Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6:2 – PSADIV[4:0]: Prescaler A Division Factor**
  These bits define the division ratio of the clock prescaler A according to Table 7-2. These bits can be written at run-time to change the frequency of the Clk_{PER4} clock relative to the system clock, Clk_{SYS}.

Table 7-2. Prescaler A division factor.

PSADIV[4:0]	Group configuration	Description
00000	1	No division
00001	2	Divide by 2
00011	4	Divide by 4
00101	8	Divide by 8
00111	16	Divide by 16
01001	32	Divide by 32
01011	64	Divide by 64
01101	128	Divide by 128
01111	256	Divide by 256
10001	512	Divide by 512
10011	6	Divide by 6
10101	10	Divide by 10
10111	12	Divide by 12
11001	24	Divide by 24
11011	48	Divide by 48
11101	–	Reserved
11111	–	Reserved

- **Bit 1:0 – PSBCDIV[1:0]: Prescaler B and C Division Factors**
  These bits define the division ratio of the clock prescalers B and C according to Table 7-3 on page 102. Prescaler B will set the clock frequency for the Clk_{PER2} clock relative to the Clk_{PER4} clock. Prescaler C will set the clock frequency for the Clk_{PER} and Clk_{CPU} clocks relative to the Clk_{PER2} clock. Refer to Figure 7-5 on page 98 for more details.

Table 7-3. Prescaler B and C division factors.

PSBCDIV[1:0]	Group configuration	Prescaler B division	Prescaler C division
00	1_1	No division	No division
01	1_2	No division	Divide by 2
10	4_1	Divide by 4	No division
11	2_2	Divide by 2	Divide by 2

7.9.3 LOCK – Lock register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	–	–	LOCK
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – LOCK: Clock System Lock**
  When this bit is written to one, the CTRL and PSCTRL registers cannot be changed, and the system clock selection and prescaler settings are protected against all further updates until after the next reset. This bit is protected by the configuration change protection mechanism. For details, refer to “Configuration change protection” on page 13.
  The LOCK bit can be cleared only by a reset.

7.9.4 RTCCTRL – RTC Control register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	RTCSRC[2:0]			RTCEN
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:1 – RTCSRC[2:0]: RTC Clock Source**
  These bits select the clock source for the real-time counter according to Table 7-4 on page 103.

Table 7-4. RTC clock source selection.

RTCSRC[2:0]	Group configuration	Description
000	ULP	1kHz from 32kHz internal ULP oscillator
001	TOSC	1.024kHz from 32.768kHz crystal oscillator on TOSC
010	RCOSC	1.024kHz from 32.768kHz internal oscillator
011	-	Reserved
100	-	Reserved
101	TOSC32	32.768kHz from 32.768kHz crystal oscillator on TOSC
110	RCOSC32	32.768kHz from 32.768kHz internal oscillator
111	EXTCLK	External clock from TOSC1

- **Bit 0 – RTCEN: RTC Clock Source Enable**
  Setting the RTCEN bit enables the selected RTC clock source for the real-time counter.

7.10 Register description – Oscillator

7.10.1 CTRL – Oscillator Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	RC8MLPM	RC8MEN	PLLEN	XOSCEN	RC32KEN	RC32MEN	RC2MEN
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6 – RC8MLPM: 8MHz Internal Oscillator Low Power Mode**
  Setting this bit enables the low power mode for the internal 8MHz oscillator. For details on characteristics and accuracy of the internal oscillator in this mode, refer to the device datasheet.
- **Bit 5 – RC8MEN: 8MHz Internal Oscillator Enable**
  Setting this bit will enable the 8MHz output of the internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See “STATUS – Oscillator Status register” on page 104.
- **Bit 4 – PLLEN: PLL Enable**
  Setting this bit enables the PLL. Before the PLL is enabled, it must be configured with the desired multiplication factor and clock source. See “STATUS – Oscillator Status register” on page 104.
- **Bit 3 – XOSCEN: External Oscillator Enable**
  Setting this bit enables the selected external clock source. Refer to “XOSCCTRL – XOSC Control register” on page 105 for details on how to select the external clock source. The external clock source should be allowed time to stabilize before it is selected as the source for the system clock. See “STATUS – Oscillator Status register” on page 104.
- **Bit 2 – RC32KEN: 32.768kHz Internal Oscillator Enable**
  Setting this bit enables the 32.768kHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See “STATUS – Oscillator Status register” on page 104.
- **Bit 1 – RC32MEN: 32MHz Internal Oscillator Enable**
  Setting this bit will enable the 32MHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See “STATUS – Oscillator Status register” on page 104.
- **Bit 0 – RC2MEN: 2MHz Internal Oscillator Enable**
  Setting this bit will enable the 2MHz output of 8MHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See “STATUS – Oscillator Status register” on page 104.
  By default, the 2MHz output from RC8MHz internal oscillator is enabled and this bit is set.

7.10.2 STATUS – Oscillator Status register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	RC8MRDY	PLLRDY	XOSCRDY	RC32KRDY	RC32MRDY	RC2MRDY
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5 – RC8MRDY: 8MHz Internal Oscillator Ready**
  This flag is set when the 8MHz output from RC8MHz internal oscillator is stable and is ready to be used as the system clock source.

- **Bit 4 – PLLRDY: PLL Ready**
  This flag is set when the PLL has locked on the selected frequency and is ready to be used as the system clock source.
- **Bit 3 – XOSCRDY: External Clock Source Ready**
  This flag is set when the external clock source is stable and is ready to be used as the system clock source.
- **Bit 2 – RC32KRDY: 32.768kHz Internal Oscillator Ready**
  This flag is set when the 32.768kHz internal oscillator is stable and is ready to be used as the system clock source.
- **Bit 1 – RC32MRDY: 32MHz Internal Oscillator Ready**
  This flag is set when the 32MHz internal oscillator is stable and is ready to be used as the system clock source.
- **Bit 0 – RC2MRDY: 2MHz Internal Oscillator Ready**
  This flag is set when the 2MHz output from RC8MHz internal oscillator is stable and is ready to be used as the system clock source.

7.10.3 XOSCCTRL – XOSC Control register

Bit	7	6	5	4	3	2	1	0
+0x02	FRQRANGE[1:0]		X32KLPM	XOSCPWR / XOSCSEL[4]	XOSCSEL[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – FRQRANGE[1:0]: 0.4 - 16MHz Crystal Oscillator Frequency Range Select**
  These bits select the frequency range for the connected crystal oscillator according to Table 7-5 on page 105.

Table 7-5. 16MHz crystal oscillator frequency range selection(1).

FRQRANGE[1:0]	Group configuration	Typical frequency range	Recommended range for capacitors C1 and C2 [pF]
00	04TO2	0.4MHz - 2MHz	100-300
01	2TO9	2MHz - 9MHz	10-40
10	9TO12	9MHz - 12MHz	10-40
11	12TO16	12MHz - 16MHz	10-30

Note: 1. Refer to Electrical Characteristics in device datasheet for finding the best setting for a given frequency.

- **Bit 5 – X32KLPM: Crystal Oscillator 32.768kHz Low Power Mode**
  Setting this bit enables the low power mode for the 32.768kHz crystal oscillator. This will reduce the swing on the TOSC2 pin.
- **Bit 4 – XOSCPWR: Crystal Oscillator Drive**
  Setting this bit will increase the current in the 0.4MHz - 16MHz crystal oscillator and increase the swing on the XTAL2 pin. This allows for driving crystals with higher load or higher frequency than specified by the FRQRANGE bits.
- **Bit 4 – XOSCSEL[4]: Crystal Oscillator Selection**
  This bit selects the pin position from which the external clock is used. When cleared, the external clock pin is XTAL1 pin. When set, the external clock pin is port C, pin 4. The selection is ignored if XOSCSEL[3:0] settings do not select the external clock option. For more details, refer to Table 7-6.
- **Bit 3:0 – XOSCSEL[3:0]: Crystal Oscillator Selection**
  These bits select the type and start-up time for the crystal or resonator that is connected to the XTAL or TOSC pins. See Table 7-6 for crystal selections. If an external clock or external oscillator is selected as the source for the system clock, see “CTRL – Oscillator Control register” on page 104. This configuration cannot be changed.

Table 7-6. 16MHz crystal oscillator frequency range selection.

XOSCSEL[3:0]	Group configuration	Selected clock source	Start-up time
0000	EXTCLK (3)	External Clock from XTAL1 pin	6 CLK
0010	32KHZ (3)	32.768kHz TOSC	16K CLK
0011	XTAL_256CLK (1)	0.4MHz - 16MHz XTAL	256 CLK
0111	XTAL_1KCLK (2)	0.4MHz - 16MHz XTAL	1K CLK
1011	XTAL_16KCLK	0.4MHz - 16MHz XTAL	16K CLK

Notes:
1. This option should be used only when frequency stability at startup is not important for the application. The option is not suitable for crystals.
2. This option is intended for use with ceramic resonators. It can also be used when the frequency stability at startup is not important for the application.
3. When the external oscillator is used as the reference for a DFLL, only EXTCLK and 32KHZ can be selected.

7.10.4 XOSCFAIL – XOSC Failure Detection register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	PLLFDIF	PLLFDEN	XOSCFDIF	XOSCFDEN
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3 – PLLFDIF: PLL Fault Detection Flag**
  If PLL failure detection is enabled, PLLFDIF is set when the PLL looses lock. Writing logic one to this location will clear PLLFDIF.
- **Bit 2 – PLLFDEN: PLL Fault Detection Enable**
  Setting this bit will enable PLL failure detection. A non-maskable interrupt will be issued when PLLFDIF is set. This bit is protected by the configuration change protection mechanism. Refer to “Configuration change protection” on page 13 for details.
- **Bit 1 – XOSCFDIF: Failure Detection Interrupt Flag**
  If the external clock source oscillator failure monitor is enabled, XOSCFDIF is set when a failure is detected. Writing logic one to this location will clear XOSCFDIF.
- **Bit 0 – XOSCFDEN: Failure Detection Enable**
  Setting this bit will enable the failure detection monitor, and a non-maskable interrupt will be issued when XOSCFDIF is set.
  This bit is protected by the configuration change protection mechanism. Refer to “Configuration change protection” on page 13 for details. Once enabled, failure detection can only be disabled by a reset.

7.10.5 RC32KCAL – 32kHz Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
+0x04	RC32KCAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RC32KCAL[7:0]: 32.768kHz Internal Oscillator Calibration Bits**
  This register is used to calibrate the 32.768kHz internal oscillator. A factory-calibrated value is loaded from the signature row of the device and written to this register during reset, giving an oscillator frequency close to 32.768kHz. The register can also be written from software to calibrate the oscillator frequency during normal operation.

7.10.6 PLLCTRL – PLL Control register

Bit	7	6	5	4	3	2	1	0
+0x05	PLLSRC[1:0]		PLLDIV	PLLFAC[4:0]				
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – PLLSRC[1:0]: Clock Source**
  The PLLSRC bits select the input source for the PLL according to Table 7-7.

Table 7-7. PLL Clock Source.

PLLSRC[1:0]	Group configuration	Description
00	RC2M	2MHz output from 8MHz internal oscillator.
01	RC8M	8MHz output from 8MHz internal oscillator.
10	RC32M	32MHz internal oscillator.
11	XOSC	External clock source (1)

Note: 1. The 32.768kHz TOSC cannot be selected as the source for the PLL. An external clock must be a minimum 0.4MHz to be used as the source clock.

- **Bit 5 – PLLDIV: PLL Divided Output Enable**
  Setting this bit will divide the output from the PLL by 2.
- **Bit 4:0 – PLLFAC[4:0]: Multiplication Factor**
  These bits select the multiplication factor for the PLL. The multiplication factor can be in the range of from 1x to 31x.

7.10.7 DFLLCTRL – DFLL Control register

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	–	–	RC32MCREF[1:0]		–
Read/Write	R	R	R	R	R	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 2:1 – RC32MCREF[1:0]: 32MHz Oscillator Calibration Reference**
  These bits are used to select the calibration source for the 32MHz DFLL according to the Table 7-8 on page 108. These bits will select only which calibration source to use for the DFLL. In addition, the actual clock source that is selected must enabled and configured for the calibration to function.

Table 7-8. 32MHz oscillator reference selection.

RC32MCREF[1:0]	Group configuration	Description
00	RC32K	32.768kHz internal oscillator.
01	XOSC32	32.768kHz crystal oscillator on TOSC.
1x	–	Reserved

- **Bit 0 - Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

7.10.8 RC8MCAL – 8MHz Internal Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
+0x07	RC8MCAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RC8MCAL[7:0]: 8MHz Internal Oscillator Calibration Bits**
  This register is used to calibrate the 8MHz internal oscillator. A factory-calibrated value is loaded from the signature row of the device and written to this register during reset, giving an oscillator frequency close to 8MHz. The register can also be written from software to calibrate the oscillator frequency during normal operation.

7.11 Register description – DFLL32M

7.11.1 CTRL – DFLL Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	–	–	ENABLE
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – ENABLE: DFLL Enable**
  Setting this bit enables the DFLL and auto-calibration of the internal oscillator. The reference clock must be enabled and stable before the DFLL is enabled.
  After disabling the DFLL, the reference clock can not be disabled before the ENABLE bit is read as zero.

7.11.2 CALA – DFLL Calibration register A

The CALA and CALB registers hold the 13-bit DFLL calibration value that is used for automatic run-time calibration of the internal oscillator. When the DFLL is disabled, the calibration registers can be written by software for manual run-time calibration of the oscillator. The oscillators will also be calibrated according to the calibration value in these registers when the DFLL is disabled.

Bit	7	6	5	4	3	2	1	0
+0x02	–	CALA[6:0]						
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	x	x	x	x	x	x	x

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6:0 – CALA[6:0]: DFLL Calibration Bits**
  These bits hold the part of the oscillator calibration value that is used for automatic runtime calibration. A factory-calibrated value is loaded from the signature row of the device and written to this register during reset, giving an oscillator frequency approximate to the nominal frequency for the oscillator. The bits cannot be written when the DFLL is enabled.

7.11.3 CALB – DFLL Calibration register B

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	CALB[5:0]					
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	x	x	x	x	x	x

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:0 – CALB[5:0]: DFLL Calibration Bits**
  These bits hold the part of the oscillator calibration value that is used to select the oscillator frequency. A factory-calibrated value is loaded from the signature row of the device and written to this register during reset, giving an oscillator frequency approximate to the nominal frequency for the oscillator. These bits are not changed during automatic run-time calibration of the oscillator. The bits cannot be written when the DFLL is enabled. When calibrating to a frequency different from the default, the CALA bits should be set to a middle value to maximize the range for the DFLL.

7.11.4 COMP1 – DFLL Compare register 1

The COMP1 and COMP2 register pair represent the frequency ratio between the oscillator and the reference clock. The initial value for these registers is the ratio between the internal oscillator frequency and a 1.024kHz reference.

Bit	7	6	5	4	3	2	1	0
+0x05	COMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – COMP1[7:0]: Compare Value Byte 1**
  These bits hold byte 1 of the 16-bit compare register.

7.11.5 COMP2 – DFLL Compare register 2

Bit	7	6	5	4	3	2	1	0
+0x06	COMP[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – COMP2[15:8]: Compare Value Byte 2**
  These bits hold byte 2 of the 16-bit compare register.

Table 7-9. Nominal DFLL32M COMP values for different output frequencies.

Oscillator frequency (MHz)	COMP value ($Clk_{RC32MCREF}$ = 1.024kHz)
30.0	0x7270
32.0	0x7A12
34.0	0x81B3
36.0	0x8954
38.0	0x90F5
40.0	0x9896
42.0	0xA037
44.0	0xA7D8
46.0	0xAF79
48.0	0xB71B
50.0	0xBEBC
52.0	0xC65D
54.0	0xCDFE

7.12 Register summary - Clock

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	–	SCLKSEL[2:0]			101
+0x01	PSCTRL	–	PSADIV[4:0]					PSBCDIV[1:0]		101
+0x02	LOCK	–	–	–	–	–	–	–	LOCK	103
+0x03	RTCCTRL	–	–	–	–	RTCSRC[2:0]			RTCEN	103
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	

7.13 Register summary - Oscillator

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	RC8MLPM	RC8MEN	PLLEN	XOSCEN	RC32KEN	RC32MEN	RC2MEN	104
+0x01	STATUS	–	–	RC8MRDY	PLLRDY	XOSCRDY	RC32KRDY	RC32MRDY	RC2MRDY	104
+0x02	XOSCCTRL	FRQRANGE[1:0]		X32KLPM	XOSCPWR XOSCSEL[4]	XOSCSEL[3:0]				105
+0x03	XOSCFAIL	–	–	–	–	PLLFDIF	PLLFDEN	XOSCFDIF	XOSCFDEN	106
+0x04	RC32KCAL	RC32KCAL[7:0]								107
+0x05	PLLCTRL	PLLSRC[1:0]		PLLDIV	PLLFAC[4:0]					107
+0x06	DFLLCTRL	–	–	–	–	–	RC32MMCREF[1:0]		–	107
+0x07	RC8MCAL	RC8MCAL[7:0]								108

7.14 Register summary – DFLL32M

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	–	–	–	ENABLE	108
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	CALA	–	CALA[6:0]							109
+0x03	CALB	–	–	CALB[5:0]						109
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	COMP1	COMP[7:0]								109
+0x06	COMP2	COMP[15:8]								110
+0x07	Reserved	–	–	–	–	–	–	–	–	

7.15 Interrupt vector summary

Table 7-10. Oscillator failure interrupt vector and its word offset address PLL and external oscillator failure interrupt base.

Offset	Source	Interrupt description
0x00	OSCF_vect	PLL and external oscillator failure interrupt vector (NMI)

8. Power Management and Sleep Modes

8.1 Features

- Power management for adjusting power consumption and functions
- Five sleep modes
  - Idle
  - Power down
  - Power save
  - Standby
  - Extended standby
- Power reduction register to disable clock and turn off unused peripherals in active and idle modes

8.2 Overview

Various sleep modes and clock gating are provided in order to tailor power consumption to application requirements. This enables the XMEGA microcontroller to stop unused modules to save power.

All sleep modes are available and can be entered from active mode. In active mode, the CPU is executing application code. When the device enters sleep mode, program execution is stopped and interrupts or a reset is used to wake the device again. The application code decides which sleep mode to enter and when. Interrupts from enabled peripherals and all enabled reset sources can restore the microcontroller from sleep to active mode.

In addition, power reduction registers provide a method to stop the clock to individual peripherals from software. When this is done, the current state of the peripheral is frozen, and there is no power consumption from that peripheral. This reduces the power consumption in active mode and idle sleep modes and enables much more fine-tuned power management than sleep modes alone.

8.3 Sleep modes

Sleep modes are used to shut down modules and clock domains in the microcontroller in order to save power. XMEGA microcontrollers have five different sleep modes tuned to match the typical functional stages during application execution. A dedicated sleep instruction (SLEEP) is available to enter sleep mode. Interrupts are used to wake the device from sleep, and the available interrupt wake-up sources are dependent on the configured sleep mode. When an enabled 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. If other, higher priority interrupts are pending when the wake-up occurs, their interrupt service routines will be executed according to their priority before the interrupt service routine for the wake-up interrupt is executed. After wake-up, the CPU is halted for four cycles before execution starts.

Table 8-1 on page 113 shows the different sleep modes and the active clock domains, oscillators, and wake-up sources.

Table 8-1. Active clock domains and wake-up sources in the different sleep modes.

	Active clock domain			Oscillators		Wake-up sources				
Sleep modes	**CPU clock**	**Peripheral clock**	**RTC clock**	**System clock source**	**RTC clock source**	**UART start of frame**	**Asynchronous port interrupts**	**TWI address match interrupts**	**Real time clock interrupts**	**All interrupts**
Idle		X	X	X	X		X	X	X	X
Power down							X	X		
Power save			X		X	X(1)	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

Note: 1. Only from internal 8MHz oscillator in low power mode

The wake-up time for the device is dependent on the sleep mode and the main clock source. The startup time for the system clock source must be added to the wake-up time for sleep modes where the system clock source is not kept running. For details on the startup time for the different oscillator options, refer to “System Clock and Clock Options” on page 94.

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

8.3.1 Idle mode

In idle mode the CPU and nonvolatile memory are stopped (note that any ongoing programming will be completed), but all peripherals, including the interrupt controller, event system and DMA controller are kept running. Any enabled interrupt will wake the device.

8.3.2 Power-down mode

In power-down mode, all clocks, including the real-time counter clock source, are stopped. This allows operation only of asynchronous modules that do not require a running clock. The only interrupts that can wake up the MCU are the two-wire interface address match interrupt, and asynchronous port interrupts.

8.3.3 Power-save mode

Power-save mode is identical to power down, with two exceptions. If the real-time counter (RTC) is enabled, it will keep running during sleep, and the device can also wake up from either an RTC overflow or compare match interrupt.

If the UART start frame detector is enabled, the device can also wake-up from any UART interrupt, including start frame interrupt. The internal 8MHz in low power mode must be used to wake-up the device from UART interrupts.

8.3.4 Standby mode

Standby mode is identical to power down, with two exceptions.

To reduce the wake-up time, the enabled system clock sources are kept running while the CPU, peripheral, and RTC clocks are stopped. If the UART start frame detector is enabled, the device can also wake-up from any UART interrupt, including start frame interrupt.

8.3.5 Extended standby mode

Extended standby mode is identical to power-save mode, with the exception that the enabled system clock sources are kept running while the CPU and peripheral clocks are stopped. This reduces the wake-up time.

If the UART start frame detector is enabled, the device can also wake-up from any UART interrupt, including start frame interrupt.

8.4 Power reduction registers

The power reduction (PR) registers provide a method to stop the clock to individual peripherals. When this is done, the current state of the peripheral is frozen and the associated I/O registers cannot be read or written. Resources used by the peripheral will remain occupied; hence, the peripheral should be disabled before stopping the clock. Enabling the clock to a peripheral again puts the peripheral in the same state as before it was stopped. This can be used in idle mode and active modes to reduce the overall power consumption. In all other sleep modes, the peripheral clock is already stopped.

Not all devices have all the peripherals associated with a bit in the power reduction registers. Setting a power reduction bit for a peripheral that is not available will have no effect.

8.5 Minimizing power consumption

There are several possibilities to consider when trying to minimize the power consumption in an AVR MCU controlled system. In general, correct sleep modes should be selected and used to ensure that only the modules required for the application are operating.

All unneeded functions should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

8.5.1 Analog-to-Digital Converter - ADC

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to “ADC – Analog to Digital Converter” on page 341 for details on ADC operation.

8.5.2 Analog Comparator - AC

When entering idle mode, the analog comparator should be disabled if not used. In other sleep modes, the analog comparator is automatically disabled. However, if the analog comparator is set up to use the internal voltage reference as input, the analog comparator should be disabled in all sleep modes. Otherwise, the internal voltage reference will be enabled, irrespective of sleep mode. Refer to “AC – Analog Comparator” on page 383 for details on how to configure the analog comparator.

8.5.3 Brownout detector

If the brownout detector is not needed by the application, this module should be turned off. If the brownout detector is enabled by the BODLEVEL fuses, it will be enabled in all sleep modes, and always consume power. In the deeper sleep modes, it can be turned off and set in sampled mode to reduce current consumption. Refer to “Brownout detection” on page 122 for details on how to configure the brownout detector.

8.5.4 Watchdog timer

If the watchdog timer is not needed in the application, the module should be turned off. If the watchdog timer is enabled, it will be enabled in all sleep modes and, hence, always consume power. Refer to “WDT – Watchdog Timer” on page 127 for details on how to configure the watchdog timer.

8.5.5 Internal 8MHz oscillator

If the low power mode is not needed by the application, this feature should be turned off. If the lower mode is enabled, it will be enabled in all sleep modes, and always consume power. Refer to “8MHz calibrated oscillator” on page 96 for details on how to enable the low power mode.

8.5.6 UART start frame detector

When entering the standby, extended standby or power save mode, the UART start frame detector should be disabled if not used. When entering the power down sleep mode, the UART start frame detector must be disabled. In all other sleep modes, the UART start frame detector is ignored. Refer to “USART” on page 271 for details on how to enable the start frame detector.

8.5.7 Port pins

When entering a sleep mode, all port pins should be configured to use minimum power. Most important is to ensure that no pins drive resistive loads. If the pin input sense is forced enabled, the corresponding pin input buffer will be enabled in all sleep modes, and always consume power. If the input sense is not forced enabled, in sleep modes where the Peripheral Clock (Clk_{PER}) is stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed.

When the UART start frame detector is enabled, the input buffers of the corresponding UART pins are forced enabled when entering sleep modes, and always consume power.

8.5.8 On-chip debug system

If the On-chip debug system is enabled and the chip enters sleep mode, the main clock source is enabled and hence always consumes power. In the deeper sleep modes, this will contribute significantly to the total current consumption.

8.6 Register description – Sleep

8.6.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	SMODE[2:0]			SEN
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:1 – SMODE[2:0]: Sleep Mode Selection**
  These bits select sleep modes according to Table 8-2 on page 116.

Table 8-2. Sleep mode.

SMODE[2:0]	Group configuration	Description
000	IDLE	Idle mode
001	–	Reserved
010	PDOWN	Power-down mode
011	PSAVE	Power-save mode
100	–	Reserved
101	–	Reserved
110	STDBY	Standby mode
111	ESTDBY	Extended standby mode

- **Bit 0 – SEN: Sleep Enable**
  This bit must be set to make the MCU enter the selected sleep mode when the SLEEP instruction is executed. To avoid unintentional entering of sleep modes, it is recommended to write SEN just before executing the SLEEP instruction and clear it immediately after waking up.

8.7 Register description – Power reduction

8.7.1 PRGEN – General Power Reduction register

Bit	7	6	5	4	3	2	1	0
+0x00	XCL	–	–	–	–	RTC	EVSYS	EDMA
Read/Write	R/W	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – XCL: XMEGA Custom Logic**
  Setting this bit stops the clock to the XMEGA Custom Logic. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.
- **Bit 6:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – RTC: Real-Time Counter**
  Setting this bit stops the clock to the real-time counter. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.
- **Bit 1 – EVSYS: Event System**
  Setting this stops the clock to the event system. When this bit is cleared, the module will continue as before it was stopped.
- **Bit 0 –EDMA: EDMA Controller**
  Setting this bit stops the clock to the EDMA controller. This bit can be set only if the EDMA controller is disabled.

8.7.2 PRPA – Power Reduction Port A register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	–	DAC	ADC	AC
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Note: Disabling of analog modules stops the clock to the analog blocks themselves and not only the interfaces.

- **Bit 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – DAC: Power Reduction DAC**
  Setting this bit stops the clock to the DAC. The DAC should be disabled before stopped.
- **Bit 1 – ADC: Power Reduction ADC**
  Setting this bit stops the clock to the ADC. The ADC should be disabled before stopped.
- **Bit 0 – AC: Power Reduction Analog Comparator**
  Setting this bit stops the clock to the analog comparator. The AC should be disabled before shutdown.

8.7.3 PRPC/D – Power Reduction Port C/D register

Bit	7	6	5	4	3	2	1	0
+0x03/+0x04	–	TWI	–	USART0	SPI	HIRES	TC5	TC4
Read/Write	R	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6 – TWI: Two-Wire Interface**
  Setting this bit stops the clock to the two-wire interface. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.
- **Bit 5 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 4 – USART0**
  Setting this bit stops the clock to USART0. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.
- **Bit 3 – SPI: Serial Peripheral Interface**
  Setting this bit stops the clock to the SPI. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.
- **Bit 2 – HIRES: High-Resolution Extension**
  Setting this bit stops the clock to the high-resolution extension for the timer/counters. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.
- **Bit 1 – TC5: Timer/Counter 5**
  Setting this bit stops the clock to timer/counter 5. When this bit is cleared, the peripheral will continue like before the shut down.
- **Bit 0 – TC4: Timer/Counter 4**
  Setting this bit stops the clock to timer/counter 4. When this bit is cleared, the peripheral will continue like before the shut down.

8.8 Register summary – Sleep

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	SMODE[2:0]			SEN	116

8.9 Register summary – Power reduction

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	PRGEN	XCL	–	–	–	–	RTC	EVSYS	EDMA	117
+0x01	PRPA	–	–	–	–	–	DAC	ADC	AC	117
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	PRPC	–	TWI	–	USART0	SPI	HIRES	TC5	TC4	118
+0x04	PRPD	–	–	–	USART0	–	–	TC5	–	118
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	

9. Reset System

9.1 Features

- Reset the microcontroller and set it to initial state when a reset source goes active
- Multiple reset sources that cover different situations
  - Power-on reset
  - External reset
  - Watchdog reset
  - Brownout reset
  - PDI reset
  - Software reset
- Asynchronous operation
  - No running system clock in the device is required for reset
- Reset status register for reading the reset source from the application code

9.2 Overview

The reset system issues a microcontroller reset and sets the device to its initial state. This is for situations where operation should not start or continue, such as when the microcontrollers operates below its power supply rating. If a reset source goes active, the device enters and is kept in reset until all reset sources have released their reset. The I/O pins are immediately tri-stated. The program counter is set to the reset vector location, and all I/O registers are set to their initial values. The SRAM content is kept. However, if the device accesses the SRAM when a reset occurs, the content of the accessed location can not be guaranteed.

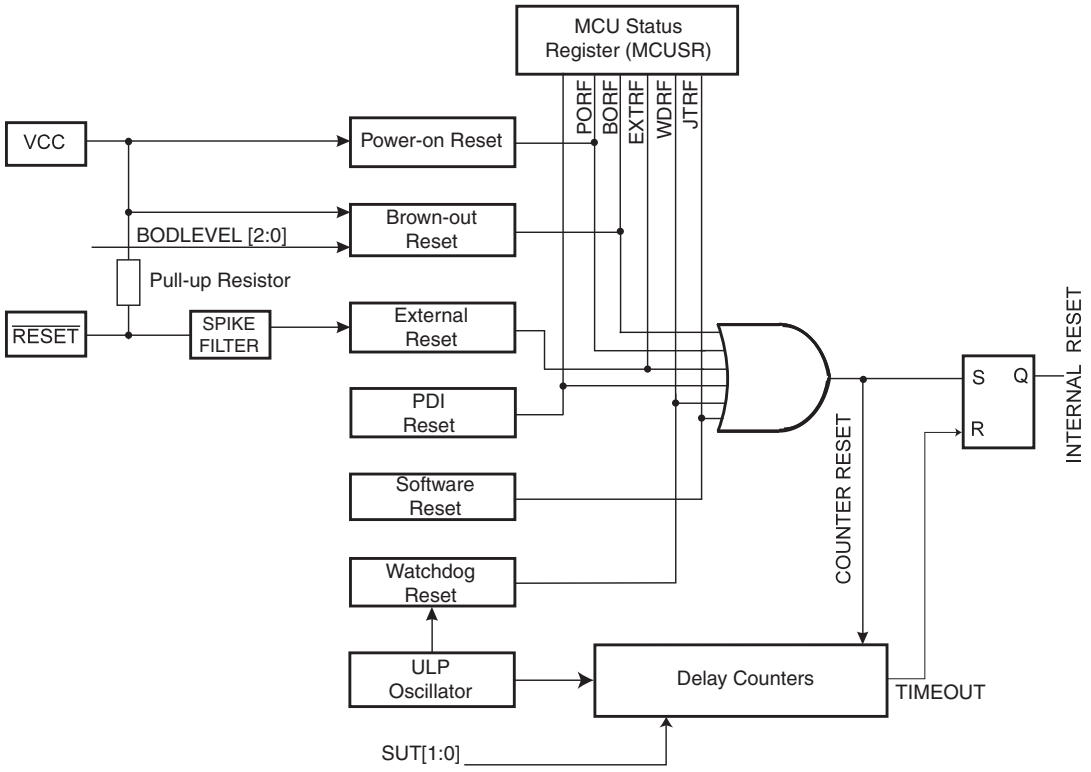
After reset is released from all reset sources, the default oscillator is started and calibrated before the device starts running from the reset vector address. By default, this is the lowest program memory address, 0, but it is possible to move the reset vector to the lowest address in the boot section.

The reset functionality is asynchronous, and so no running system clock is required to reset the device. The software reset feature makes it possible to issue a controlled system reset from the user software.

The reset status register has individual status flags for each reset source. It is cleared at power-on reset, and shows which sources have issued a reset since the last power-on.

An overview of the reset system is shown in Figure 9-1 on page 121.

Figure 9-1. Reset system overview.



9.3 Reset sequence

A reset request from any reset source will immediately reset the device and keep it in reset as long as the request is active. When all reset requests are released, the device will go through three stages before the device starts running again:

- Reset counter delay
- Oscillator startup
- Oscillator calibration

If another reset requests occurs during this process, the reset sequence will start over again.

9.3.1 Reset counter

The reset counter can delay reset release with a programmable period from when all reset requests are released. The reset delay is timed from the 1kHz output of the ultra low power (ULP) internal oscillator, and in addition 24 system clock cycles (clk_{SYS}) are counted before reset is released. The reset delay is set by the STARTUPTIME fuse bits. The selectable delays are shown in Table 9-1 on page 121.

Table 9-1. Reset delay.

SUT[1:0]	Number of 1kHz ULP oscillator clock cycles	Recommended usage
00	64K Clk_{ULP}+ 24 Clk_{SYS}	Stable frequency at startup
01	4K Clk_{ULP} + 24 Clk_{SYS}	Slowly rising power
10	Reserved	–
11	24 Clk_{SYS}	Fast rising power or BOD enabled

Whenever a reset occurs, the clock system is reset and the 2MHz output from the internal 8MHz oscillator is chosen as the source for Clk_{SYS}.

9.3.2 Oscillator startup

After the reset delay, the 8MHz internal oscillator clock is started, and its calibration values are automatically loaded from the production signature row to the calibration registers.

9.4 Reset sources

9.4.1 Power-on reset

A power-on reset (POR) is generated by an on-chip detection circuit. The POR is activated when the V_{CC} rises and reaches the POR threshold voltage (V_{POT}), and this will start the reset sequence.

The POR is also activated to power down the device properly when the V_{CC} falls and drops below the V_{POT} level.

The V_{POT} level is higher for falling V_{CC} than for rising V_{CC}. Consult the datasheet for POR characteristics data.

Figure 9-2. MCU startup, $\overline{\text{RESET}}$ tied to V_{CC}.

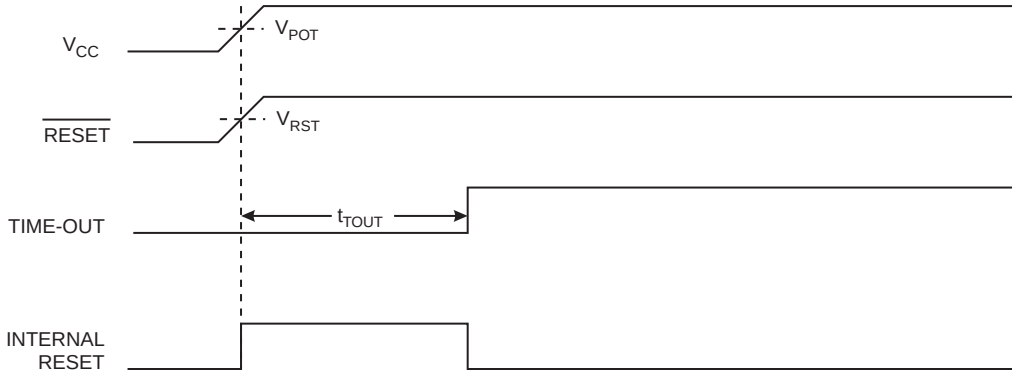
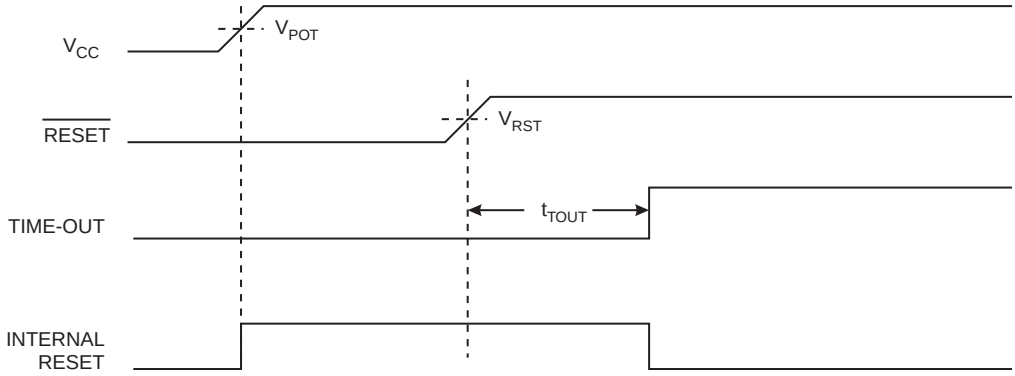


Figure 9-3. MCU startup, $\overline{\text{RESET}}$ extended externally.



9.4.2 Brownout detection

The on-chip brownout detection (BOD) circuit monitors the V_{CC} level during operation by comparing it to a fixed, programmable level that is selected by the BODLEVEL fuses. If disabled, BOD is forced on at the lowest level during chip erase and when the PDI is enabled.

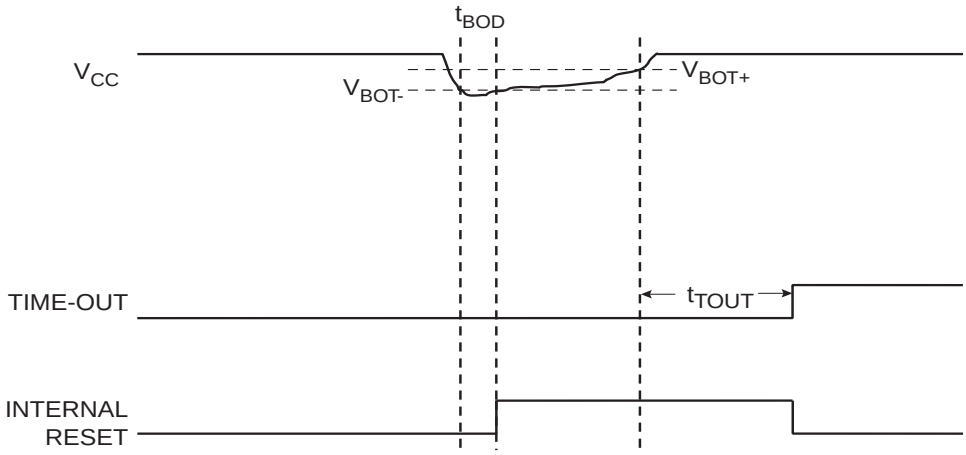
When the BOD is enabled and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-4), the brownout reset is immediately activated.

When V_{CC} increases above the trigger level (V_{BOT+} in Figure 9-4), the reset counter starts the MCU after the timeout period, t_{TOUT} has expired.

The trigger level has a hysteresis to ensure spike free brownout detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD circuit will detect a drop in V_{CC} only if the voltage stays below the trigger level for longer than t_{BOD}.

Figure 9-4. Brownout detection reset.



For BOD characterization data consult the device datasheet. The programmable BODLEVEL setting is shown in Table 9-2.

Table 9-2. Programmable BODLEVEL setting.

BOD level	Fuse BODLEVEL[2:0](2)	V_{BOT}(1)	Unit
BOD level 0	111	1.6	V
BOD level 1	110	1.8	
BOD level 2	101	2.0	
BOD level 3	100	2.2	
BOD level 4	011	2.4	
BOD level 5	010	2.6	
BOD level 6	001	2.8	
BOD level 7	000	3.0	

Notes: 1. The values are nominal values only. For accurate, actual numbers, consult the device datasheet.
2. Changing these fuse bits will have no effect until leaving programming mode.

The BOD circuit has three modes of operation:

- **Disabled:** In this mode, there is no monitoring of the V_{CC} level.
- **Enabled:** In this mode, the V_{CC} level is continuously monitored, and a drop in V_{CC} below V_{BOT} for a period of t_{BOD} will give a brownout reset.
- **Sampled:** In this mode, the BOD circuit will sample the V_{CC} level with a period identical to that of the 1kHz output from the ultra low power (ULP) internal oscillator. Between each sample, the BOD is turned off. This mode will reduce the power consumption compared to the enabled mode, but a fall in the V_{CC} level between two positive

edges of the 1kHz ULP oscillator output will not be detected. If a brownout is detected in this mode, the BOD circuit is set in enabled mode to ensure that the device is kept in reset until V_{CC} is above V_{BOT} again.

The BODACT fuse determines the BOD setting for active mode and idle mode, while the BODPD fuse determines the brownout detection setting for all sleep modes, except idle mode.

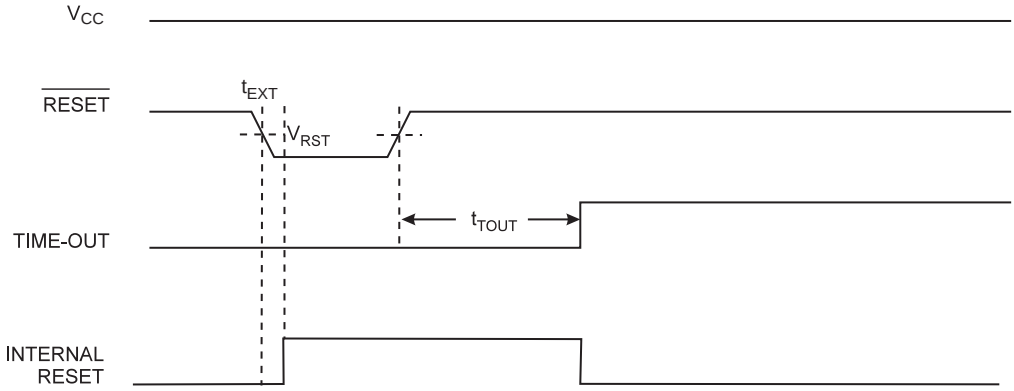
Table 9-3. BOD setting fuse decoding.

BODACT[1:0]/ BODPD[1:0]	Mode
00	Reserved
01	Sampled
10	Enabled
11	Disabled

9.4.3 External reset

The external reset circuit is connected to the external $\overline{\text{RESET}}$ pin. The external reset will trigger when the $\overline{\text{RESET}}$ pin is driven below the $\overline{\text{RESET}}$ pin threshold voltage, V_{RST} for longer than the minimum pulse period, t_{EXT}. The reset will be held as long as the pin is kept low. The $\overline{\text{RESET}}$ pin includes an internal pull-up resistor.

Figure 9-5. External reset characteristics.

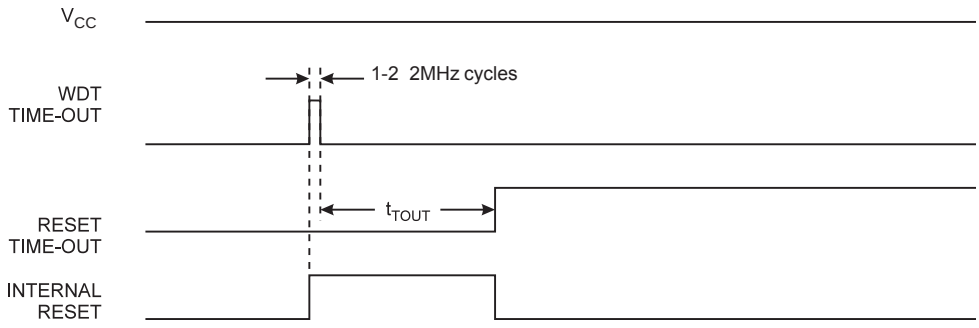


For external reset characterization data consult the device datasheet.

9.4.4 Watchdog reset

The watchdog timer (WDT) is a system function for monitoring correct program operation. If the WDT is not reset from the software within a programmable timout period, a watchdog reset will be given. The watchdog reset is active for one to two 2MHz clock cycles from the 8MHz internal oscillator.

Figure 9-6. Watchdog reset.

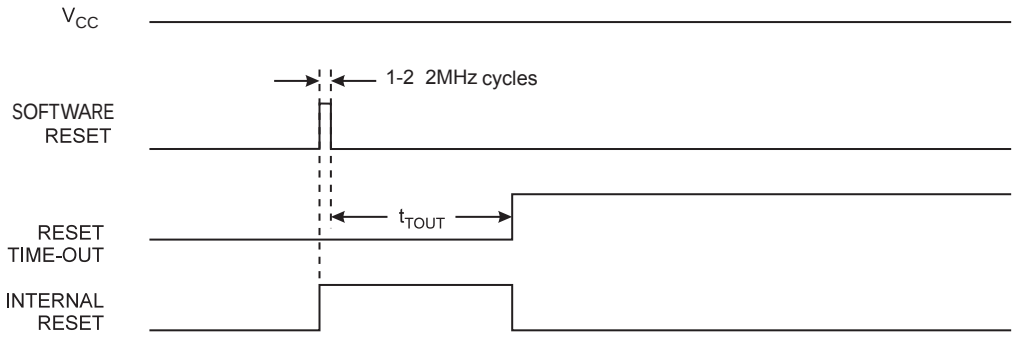


For information on configuration and use of the WDT, refer to the “WDT – Watchdog Timer” on page 127.

9.4.5 Software reset

The software reset makes it possible to issue a system reset from software by writing to the software reset bit in the reset control register.The reset will be issued within two CPU clock cycles after writing the bit. It is not possible to execute any instruction from when a software reset is requested until it is issued.

Figure 9-7. Software reset.



9.4.6 Program and debug interface reset

The program and debug interface reset contains a separate reset source that is used to reset the device during external programming and debugging. This reset source is accessible only from external debuggers and programmers.

9.5 Register description

9.5.1 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	SRF	PDIRF	WDRF	BORF	EXTRF	PORF
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	–	–	–	–	–	–	–	–

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5 – SRF: Software Reset Flag**
  This flag is set if a software reset occurs. The flag will be cleared by a power-on reset or by writing a one to the bit location.
- **Bit 4 – PDIRF: Program and Debug Interface Reset Flag**
  This flag is set if a programming interface reset occurs. The flag will be cleared by a power-on reset or by writing a one to the bit location.
- **Bit 3 – WDRF: Watchdog Reset Flag**
  This flag is set if a watchdog reset occurs. The flag will be cleared by a power-on reset or by writing a one to the bit location.
- **Bit 2 – BORF: Brownout Reset Flag**
  This flag is set if a brownout reset occurs. The flag will be cleared by a power-on reset or by writing a one to the bit location.
- **Bit 1 – EXTRF: External Reset Flag**
  This flag is set if an external reset occurs. The flag will be cleared by a power-on reset or by writing a one to the bit location.
- **Bit 0 – PORF: Power On Reset Flag**
  This flag is set if a power-on reset occurs. Writing a one to the flag will clear the bit location.

9.5.2 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	–	–	–	SWRST
Read/Write	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – SWRST: Software Reset**
  When this bit is set, a software reset will occur. The bit is cleared when a reset is issued. This bit is protected by the configuration change protection mechanism. For details, refer to “Configuration change protection” on page 13.

9.6 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	–	–	SRF	PDIRF	WDRF	BORF	EXTRF	PORF	126
+0x01	CTRL	–	–	–	–	–	–	–	SWRST	126

10. WDT – Watchdog Timer

10.1 Features

- Issues a device reset if the timer is not reset before its timeout period
- Asynchronous operation from dedicated oscillator
- 1kHz output of the 32kHz ultra low power oscillator
- 11 selectable timeout periods, from 8ms to 8s.
- Two operation modes:
  - Normal mode
  - Window mode
- Configuration lock to prevent unwanted changes

10.2 Overview

The watchdog timer (WDT) is a system function for monitoring correct program operation. It makes it possible to recover from error situations such as runaway or deadlocked code. The WDT is a timer, configured to a predefined timeout period, and is constantly running when enabled. If the WDT is not reset within the timeout period, it will issue a microcontroller reset. The WDT is reset by executing the WDR (watchdog timer reset) instruction from the application code.

The window mode makes it possible to define a time slot or window inside the total timeout period during which 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, this can also catch situations where a code error causes constant WDR execution.

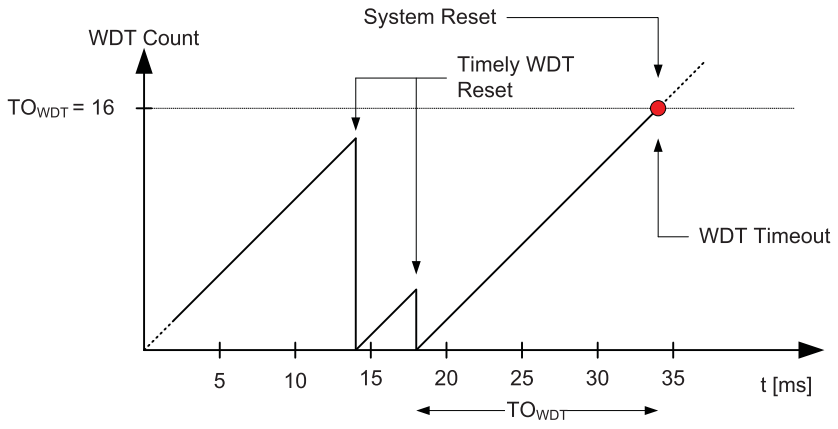
The WDT will run in active mode and all sleep modes, if enabled. It is asynchronous, runs from a CPU-independent clock source, and will continue to operate to issue a system reset even if the main clocks fail.

The configuration change protection mechanism ensures that the WDT settings cannot be changed by accident. For increased safety, a fuse for locking the WDT settings is also available.

10.3 Normal mode operation

In normal mode operation, a single timeout period is set for the WDT. If the WDT is not reset from the application code before the timeout occurs, then the WDT will issue a system reset. There are 11 possible WDT timeout (TO_{WDT}) periods, selectable from 8ms to 8s, and the WDT can be reset at any time during the timeout period. A new WDT timeout period will be started each time the WDT is reset by the WDR instruction. The default timeout period is controlled by fuses. Normal mode operation is illustrated in Figure 10-1 on page 127.

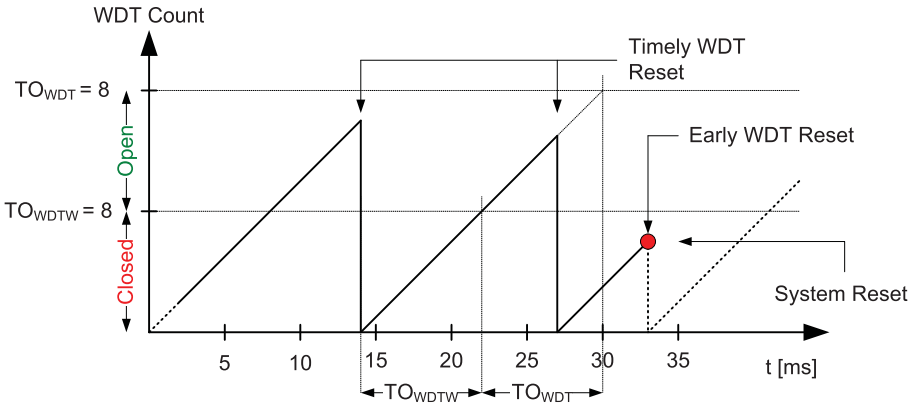
Figure 10-1. Normal mode operation.



10.4 Window mode operation

In window mode operation, the WDT uses two different timeout periods, a "closed" window timeout period (TO_{WDTW}) and the normal timeout period (TO_{WDT}). The closed window timeout period defines a duration of from 8ms to 8s where the WDT cannot be reset. If the WDT is reset during this period, the WDT will issue a system reset. The normal WDT timeout period, which is also 8ms to 8s, defines the duration of the "open" period during which the WDT can (and should) be reset. The open period will always follow the closed period, and so the total duration of the timeout period is the sum of the closed window and the open window timeout periods. The default closed window timeout period is controlled by fuses (both open and closed periods are controlled by fuses). The window mode operation is illustrated in Figure 10-2.

Figure 10-2. Window mode operation.



10.5 Watchdog timer clock

The WDT is clocked from the 1kHz output from the 32kHz ultra low power (ULP) internal oscillator. Due to the ultra low power design, the oscillator is not very accurate, and so the exact timeout period may vary from device to device. When designing software which uses the WDT, this device-to-device variation must be kept in mind to ensure that the timeout periods used are valid for all devices. For more information on ULP oscillator accuracy, consult the device datasheet.

10.6 Configuration protection and lock

The WDT is designed with two security mechanisms to avoid unintentional changes to the WDT settings.

The first mechanism is the configuration change protection mechanism, employing a timed write procedure for changing the WDT control registers. In addition, for the new configuration to be written to the control registers, the register's change enable bit must be written at the same time.

The second mechanism locks the configuration by setting the WDT lock fuse. When this fuse is set, the watchdog time control register cannot be changed; hence, the WDT cannot be disabled from software. After system reset, the WDT will resume at the configured operation. When the WDT lock fuse is programmed, the window mode timeout period cannot be changed, but the window mode itself can still be enabled or disabled.

10.7 Registers description

10.7.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	PER[3:0]				ENABLE	CEN
Read/Write (unlocked)	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Read/Write (locked)	R	R	R	R	R	R	R	R
Initial value (x = fuse)	0	0	X	X	X	X	X	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 5:2 – PER[3:0]: Timeout Period**
  These bits determine the watchdog timeout period as a number of 1kHz ULP oscillator cycles. In window mode operation, these bits define the open window period. The different typical timeout periods are found in Table 10-1. The initial values of these bits are set by the watchdog timeout period (WDP) fuses, which are loaded at power-on. In order to change these bits, the CEN bit must be written to 1 at the same time. These bits are protected by the configuration change protection mechanism. For a detailed description, refer to “Configuration change protection” on page 13.

Table 10-1. Watchdog timeout periods.

PER[3:0]	Group configuration	Typical timeout periods
0000	8CLK	8ms
0001	16CLK	16ms
0010	32CLK	32ms
0011	64CLK	64ms
0100	128CLK	0.128s
0101	256CLK	0.256s
0110	512CLK	0.512s
0111	1KCLK	1.0s
1000	2KCLK	2.0s
1001	4KCLK	4.0s
1010	8KCLK	8.0s
1011	–	Reserved
1100	–	Reserved
1101	–	Reserved
1110	–	Reserved
1111	–	Reserved

Note: Reserved settings will not give any timeout.

- **Bit 1 – ENABLE: Enable**
  This bit enables the WDT. Clearing this bit disables the watchdog timer.
  In order to change this bit, the CEN bit in "CTRL – Control register" on page 129 must be written to one at the same time. This bit is protected by the configuration change protection mechanism, For a detailed description, refer to "Configuration change protection" on page 13.
- **Bit 0 – CEN: Change Enable**
  This bit enables the ability to change the configuration of the "CTRL – Control register" on page 129. When writing a new value to this register, this bit must be written to one at the same time for the changes to take effect. This bit is protected by the configuration change protection mechanism. For a detailed description, refer to "Configuration change protection" on page 13.

10.7.2 WINCTRL – Window Mode Control register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	WPER[3:0]				WEN	WCEN
Read/Write (unlocked)	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Read/Write (locked)	R	R	R	R	R	R	R/W	R/W
Initial value (x = fuse)	0	0	X	X	X	X	X	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:2 – WPER[3:0]: Window Mode Timeout Period**
  These bits determine the closed window period as a number of 1kHz ULP oscillator cycles in window mode operation. The typical different closed window periods are found in Table 10-2. The initial values of these bits are set by the watchdog window timeout period (WDWP) fuses, and are loaded at power-on. In normal mode these bits are not in use.
  In order to change these bits, the WCEN bit must be written to one at the same time. These bits are protected by the configuration change protection mechanism. For a detailed description, refer to "Configuration change protection" on page 13.

Table 10-2. Watchdog closed window periods.

WPER[3:0]	Group configuration	Typical closed window periods
0000	8CLK	8ms
0001	16CLK	16ms
0010	32CLK	32ms
0011	64CLK	64ms
0100	128CLK	0.128s
0101	256CLK	0.256s
0110	512CLK	0.512s
0111	1KCLK	1.0s
1000	2KCLK	2.0s
1001	4KCLK	4.0s
1010	8KCLK	8.0s
1011	–	Reserved

WPER[3:0]	Group configuration	Typical closed window periods
1100	–	Reserved
1101	–	Reserved
1110	–	Reserved
1111	–	Reserved

Note: Reserved settings will not give any timeout for the window.

- **Bit 1 – WEN: Window Mode Enable**
  This bit enables the window mode. In order to change this bit, the WCEN bit in "WINCTRL – Window Mode Control register" on page 130 must be written to one at the same time. This bit is protected by the configuration change protection mechanism. For a detailed description, refer to "Configuration change protection" on page 13.
- **Bit 0 – WCEN: Window Mode Change Enable**
  This bit enables the ability to change the configuration of the "WINCTRL – Window Mode Control register" on page 130. When writing a new value to this register, this bit must be written to one at the same time for the changes to take effect. This bit is protected by the configuration change protection mechanism, but not protected by the WDT lock fuse.

10.7.3 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	–	–	SYNCBUSY
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – SYNCBUSY: Synchronization Busy Flag**
  This flag is set after writing to the CTRL or WINCTRL registers and the data are being synchronized from the system clock to the WDT clock domain. This bit is automatically cleared after the synchronization is finished. Synchronization will take place only when the ENABLE bit for the Watchdog Timer is set.

10.8 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	PER[3:0]				ENABLE	CEN	129
+0x01	WINCTRL	–	–	WPER[3:0]				WEN	WCEN	130
+0x02	STATUS	–	–	–	–	–	–	–	SYNCBUSY	131

11. PMIC – Interrupts and Programmable Multilevel Interrupt Controller

11.1 Features

- Short and predictable interrupt response time
- Separate interrupt configuration and vector address for each interrupt
- Programmable multilevel interrupt controller
  - Interrupt prioritizing according to level and vector address
  - Three selectable interrupt levels for all interrupts: low, medium and high
  - Selectable, round-robin priority scheme within low-level interrupts
  - Non-maskable interrupts for critical functions
- Interrupt vectors optionally placed in the application section or the boot loader section

11.2 Overview

Interrupts signal a change of state in peripherals, and this can be used to alter program execution. Peripherals can have one or more interrupts, and all are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when the interrupt condition is present. The programmable multilevel interrupt controller (PMIC) controls the handling and prioritizing of interrupt requests. When an interrupt request is acknowledged by the PMIC, the program counter is set to point to the interrupt vector, and the interrupt handler can be executed.

All peripherals can select between three different priority levels for their interrupts: low, medium, and high. Interrupts are prioritized according to their level and their interrupt vector address. Medium-level interrupts will interrupt low-level interrupt handlers. High-level interrupts will interrupt both medium- and low-level interrupt handlers. Within each level, the interrupt priority is decided from the interrupt vector address, where the lowest interrupt vector address has the highest interrupt priority. Low-level interrupts have an optional round-robin scheduling scheme to ensure that all interrupts are serviced within a certain amount of time.

Non-maskable interrupts (NMI) are also supported, and can be used for system critical functions.

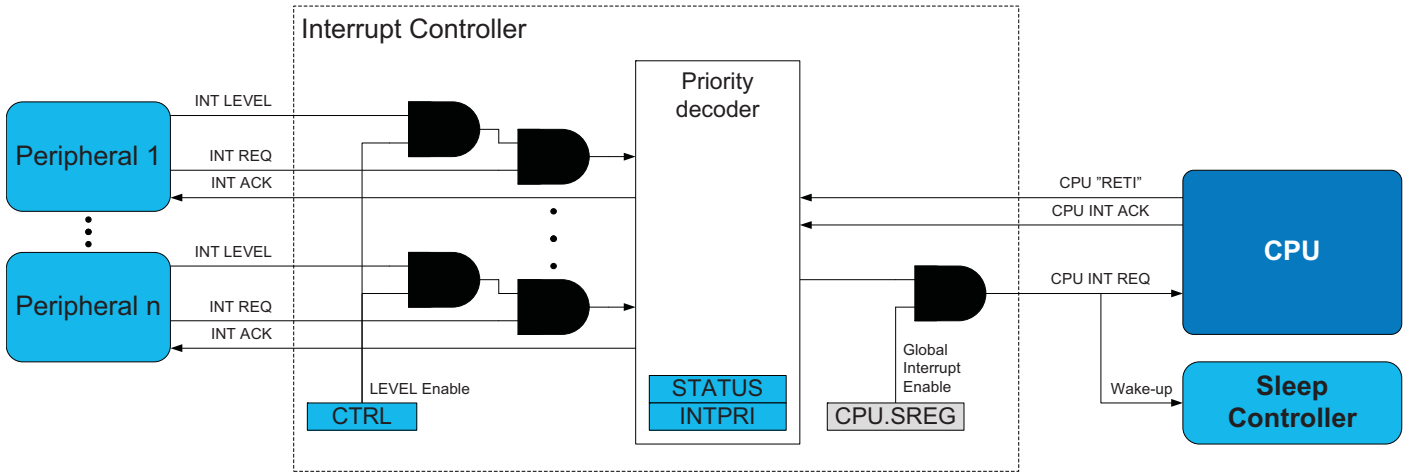
11.3 Operation

Interrupts must be globally enabled for any interrupts to be generated. This is done by setting the global interrupt enable (I) bit in the CPU “SREG – Status register” on page 17. The I bit will not be cleared when an interrupt is acknowledged. Each interrupt level must also be enabled before interrupts with the corresponding level can be generated.

When an interrupt is enabled and the interrupt condition is present, the PMIC will receive the interrupt request. Based on the interrupt level and interrupt priority of any ongoing interrupts, the interrupt is either acknowledged or kept pending until it has priority. When the interrupt request is acknowledged, the program counter is updated to point to the interrupt vector. The interrupt vector is normally a jump to the interrupt handler; the software routine that handles the interrupt. After returning from the interrupt handler, program execution continues from where it was before the interrupt occurred. One instruction is always executed before any pending interrupt is served.

The PMIC status register contains state information that ensures that the PMIC returns to the correct interrupt level when the RETI (interrupt return) instruction is executed at the end of an interrupt handler. Returning from an interrupt will return the PMIC to the state it had before entering the interrupt. The status register (SREG) is not saved automatically upon an interrupt request. The RET (subroutine return) instruction cannot be used when returning from the interrupt handler routine, as this will not return the PMIC to its correct state.

Figure 11-1. Interrupt controller overview.



11.4 Interrupts

All interrupts and the reset vector each have a separate program vector address in the program memory space. The lowest address in the program memory space is the reset vector. All interrupts are assigned with individual control bits for enabling and setting the interrupt level, and this is set in the control registers for each peripheral that can generate interrupts. Details on each interrupt are described in the peripheral where the interrupt is available.

Each interrupt has an interrupt flag associated with it. When the interrupt condition is present, the interrupt flag will be set, even if the corresponding interrupt is not enabled. For most interrupts, the interrupt flag is automatically cleared when executing the interrupt vector. Writing a logical one to the interrupt flag will also clear the flag. Some interrupt flags are not cleared when executing the interrupt vector, and some are cleared automatically when an associated register is accessed (read or written). This is described for each individual interrupt flag.

If an interrupt condition occurs while another, higher priority interrupt is executing or pending, the interrupt flag will be set and remembered until the interrupt has priority. If an interrupt condition occurs while the corresponding interrupt is not enabled, the interrupt flag will be set and remembered until the interrupt is enabled or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while global interrupts are disabled, the corresponding interrupt flag will be set and remembered until global interrupts are enabled. All pending interrupts are then executed according to their order of priority.

Interrupts can be blocked when executing code from a locked section; e.g., when the boot lock bits are programmed. This feature improves software security. Refer to “Memory Programming” on page 403 for details on lock bit settings.

Interrupts are automatically disabled for up to four CPU clock cycles when the configuration change protection register is written with the correct signature. Refer to “Configuration change protection” on page 13 for more details.

11.4.1 NMI – Non-maskable interrupts

Which interrupts represent NMI and which represent regular interrupts cannot be selected. Non-maskable interrupts must be enabled before they can be used. Refer to the device datasheet for NMI present on each device.

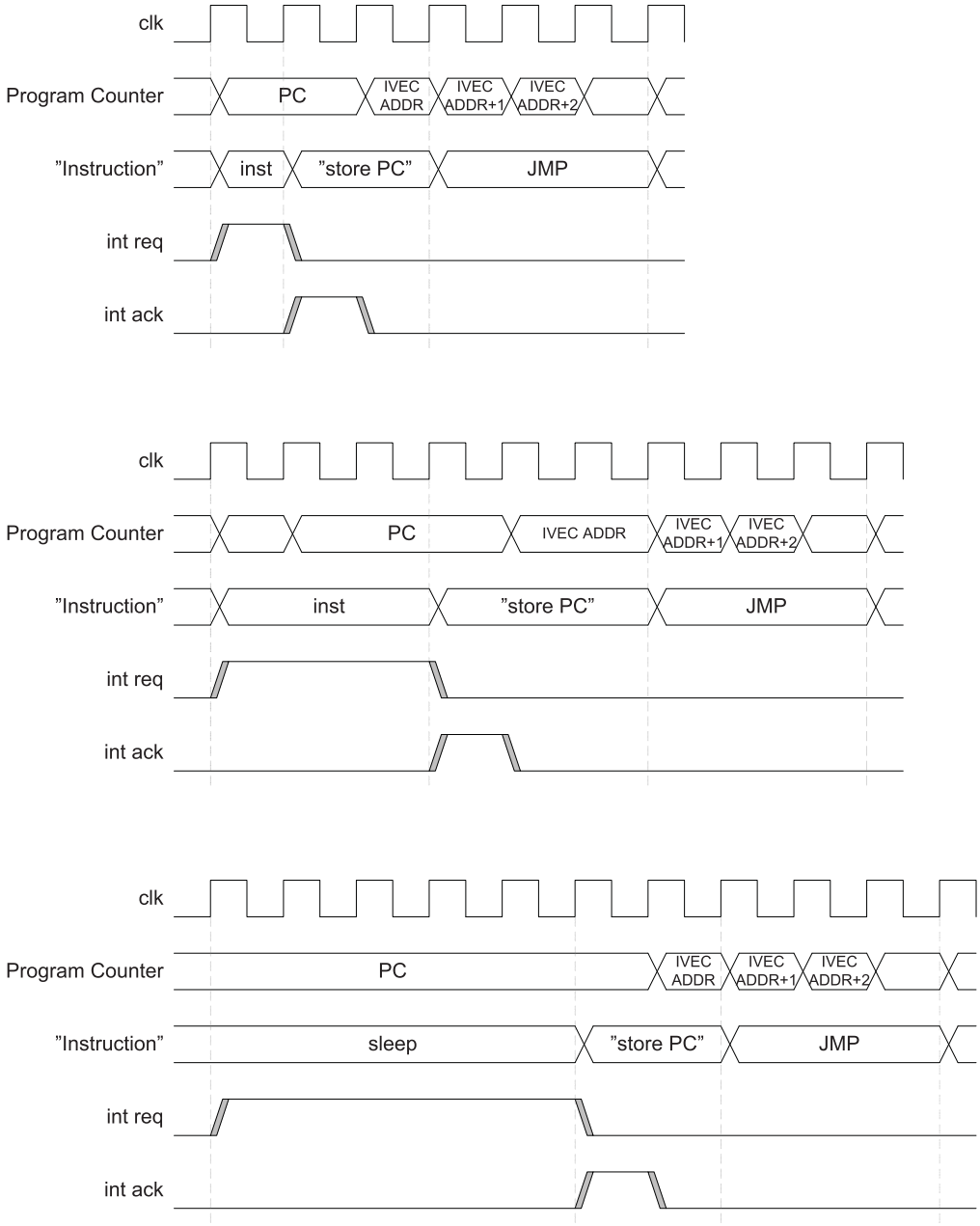
An NMI will be executed regardless of the setting of the I bit in the CPU status register, and it will never change the I bit. No other interrupts can interrupt a NMI handler. If more than one NMI is requested at the same time, priority is static according to the interrupt vector address, where the lowest address has highest priority.

11.4.2 Interrupt response time

The interrupt response time for all the enabled interrupts is three CPU clock cycles, minimum; one cycle to finish the ongoing instruction and two cycles to store the program counter to the stack. After the program counter is pushed on the stack, the program vector for the interrupt is executed. The jump to the interrupt handler takes three clock cycles.

If an interrupt occurs during execution of a multicycle instruction, this instruction is completed before the interrupt is served. See Figure 11-2 on page 134 for more details.

Figure 11-2. Interrupt execution of a multi cycle instruction.



If an interrupt occurs when the device is in sleep mode, the interrupt execution response time is increased by five clock cycles. In addition, the response time is increased by the start-up time from the selected sleep mode.

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.

11.5 Interrupt level

The interrupt level is independently selected for each interrupt source. For any interrupt request, the PMIC also receives the interrupt level for the interrupt. The interrupt levels and their corresponding bit values for the interrupt level configuration of all interrupts are shown in Table 11-1 on page 135.

Table 11-1. Interrupt levels.

Interrupt level configuration	Group configuration	Description
00	OFF	Interrupt disabled.
01	LO	Low-level interrupt
10	MED	Medium-level interrupt
11	HI	High-level interrupt

The interrupt level of an interrupt request is compared against the current level and status of the interrupt controller. An interrupt request of a higher level will interrupt any ongoing interrupt handler from a lower level interrupt. When returning from the higher level interrupt handler, the execution of the lower level interrupt handler will continue.

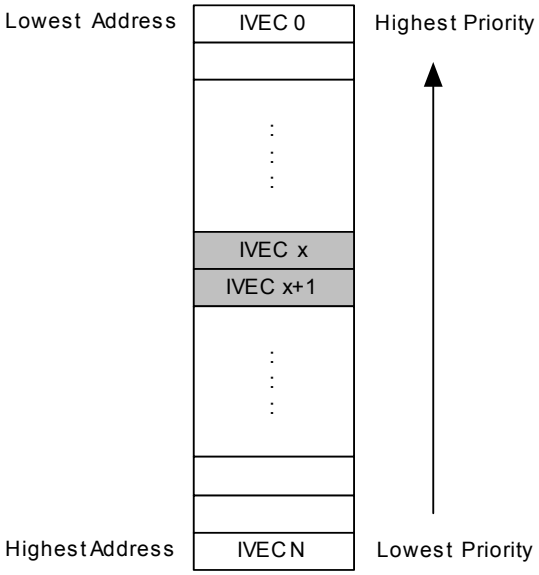
11.6 Interrupt priority

Within each interrupt level, all interrupts have a priority. When several interrupt requests are pending, the order in which interrupts are acknowledged is decided both by the level and the priority of the interrupt request. Interrupts can be organized in a static or dynamic (round-robin) priority scheme. High- and medium-level interrupts and the NMI will always have static priority. For low-level interrupts, static or dynamic priority scheduling can be selected.

11.6.1 Static priority

Interrupt vectors (IVEC) are located at fixed addresses. For static priority, the interrupt vector address decides the priority within one interrupt level, where the lowest interrupt vector address has the highest priority. Refer to the device datasheet for the interrupt vector table with the base address for all modules and peripherals with interrupt capability. Refer to the interrupt vector summary of each module and peripheral in this manual for a list of interrupts and their corresponding offset address within the different modules and peripherals.

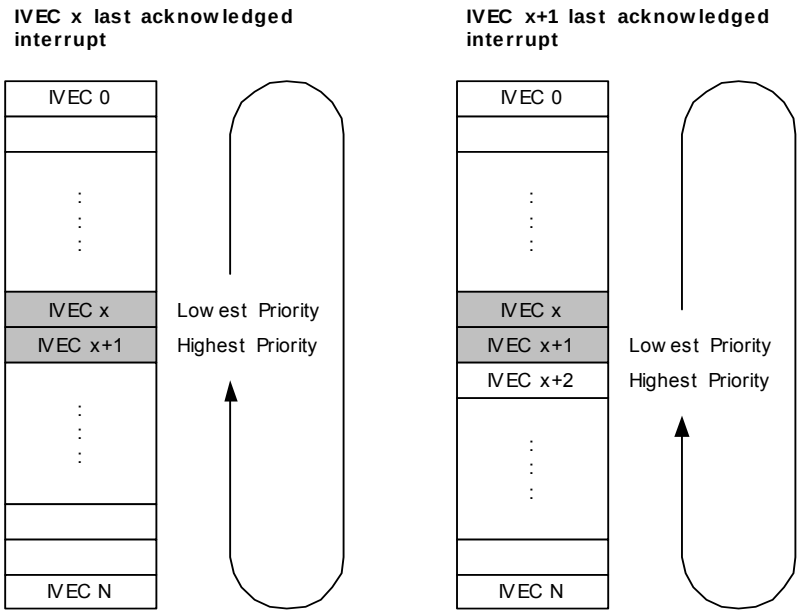
Figure 11-3. Static priority.



11.6.2 Round-robin scheduling

To avoid the possible starvation problem for low-level interrupts with static priority, where some interrupts might never be served, the PMIC offers round-robin scheduling for low-level interrupts. When round-robin scheduling is enabled, the interrupt vector address for the last acknowledged low-level interrupt will have the lowest priority the next time one or more interrupts from the low level is requested.

Figure 11-4. Round-robin scheduling.



11.7 Interrupt vector locations

Table 11-2 shows reset and Interrupt vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 11-2. Reset and interrupt vectors placement.

BOOTRST	IVSEL	Reset address	Interrupt vectors start address
1	0	0x0000	0x0002
1	1	0x0000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x0002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

11.8 Register description

11.8.1 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x00	NMIEX	–	–	–	–	HILVLEX	MEDLVLEX	LOLVLEX
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – NMIEX: Non-Maskable Interrupt Executing**
  This flag is set if a non-maskable interrupt is executing. The flag will be cleared when returning (RETI) from the interrupt handler.
- **Bit 6:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – HILVLEX: High-level Interrupt Executing**
  This flag is set when a high-level interrupt is executing or when the interrupt handler has been interrupted by an NMI. The flag will be cleared when returning (RETI) from the interrupt handler.
- **Bit 1 – MEDLVLEX: Medium-level Interrupt Executing**
  This flag is set when a medium-level interrupt is executing or when the interrupt handler has been interrupted by an interrupt from higher level or an NMI. The flag will be cleared when returning (RETI) from the interrupt handler.
- **Bit 0 – LOLVLEX: Low-level Interrupt Executing**
  This flag is set when a low-level interrupt is executing or when the interrupt handler has been interrupted by an interrupt from higher level or an NMI. The flag will be cleared when returning (RETI) from the interrupt handler.

11.8.2 INTPRI – Interrupt priority register

Bit	7	6	5	4	3	2	1	0
+0x01	INTPRI[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – INTPRI: Interrupt Priority**
  When round-robin scheduling is enabled, this register stores the interrupt vector of the last acknowledged low-level interrupt. The stored interrupt vector will have the lowest priority the next time one or more low-level interrupts are pending. The register is accessible from software to change the priority queue. This register is not reinitialized to its initial value if round-robing scheduling is disabled, and so if default static priority is needed, the register must be written to zero.

11.8.3 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x02	RREN	IVSEL	–	–	–	HILVLEN	MEDLVLEN	LOLVLEN
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – RREN: Round-robin Scheduling Enable**
  When the RREN bit is set, the round-robin scheduling scheme is enabled for low-level interrupts. When this bit is cleared, the priority is static according to interrupt vector address, where the lowest address has the highest priority.

- **Bit 6 – IVSEL: Interrupt Vector Select**
  When the IVSEL bit is cleared (zero), the interrupt vectors are placed at the start of the application section in flash. When this bit is set (one), the interrupt vectors are placed in the beginning of the boot section of the flash. Refer to the device datasheet for the absolute address.
  This bit is protected by the configuration change protection mechanism. Refer to "Configuration change protection" on page 13 for details.
- **Bit 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – HILVLEN: High-level Interrupt Enable**(1)
  When this bit is set, all high-level interrupts are enabled. If this bit is cleared, high-level interrupt requests will be ignored.
- **Bit 1 – MEDLVLEN: Medium-level Interrupt Enable**(1)
  When this bit is set, all medium-level interrupts are enabled. If this bit is cleared, medium-level interrupt requests will be ignored.
- **Bit 0 – LOLVLEN: Low-level Interrupt Enable**(1)
  When this bit is set, all low-level interrupts are enabled. If this bit is cleared, low-level interrupt requests will be ignored.

Note: 1. Ignoring interrupts will be effective one cycle after the bit is cleared.

11.9 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	NMIEX	–	–	–	–	HILVLEX	MEDLVLEX	LOLVLEX	137
+0x01	INTPRI	INTPRI[7:0]								137
+0x02	CTRL	RREN	IVSEL	–	–	–	HILVLEN	MEDLVLEN	LOLVLEN	137

12. I/O Ports

12.1 Features

- General purpose input and output pins with individual configuration
- Output driver with configurable driver and pull settings:
  - Totem-pole
  - Wired-AND
  - Wired-OR
  - Bus-keeper
  - Inverted I/O
- Input with synchronous and/or asynchronous sensing with interrupts and events
  - Sense both edges
  - Sense rising edges
  - Sense falling edges
  - Sense low level
- Optional pull-up and pull-down resistor on input and Wired-OR/AND configurations
- Optional slew rate control per I/O port
- Asynchronous pin change sensing that can wake the device from all sleep modes
- Port interrupt with pin masking
- Efficient and safe access to port pins
  - Hardware read-modify-write through dedicated toggle/clear/set registers
  - Configuration of multiple pins in a single operation
  - Mapping of port registers into bit-accessible I/O memory space
- Peripheral clocks output on port pin
- Real-time counter clock output to port pin
- Event channels can be output on port pin
- Remapping of digital peripheral pin functions
  - Selectable USART, and timer/counter input/output pin locations
  - Selectable analog comparator outputs pins locations

12.2 Overview

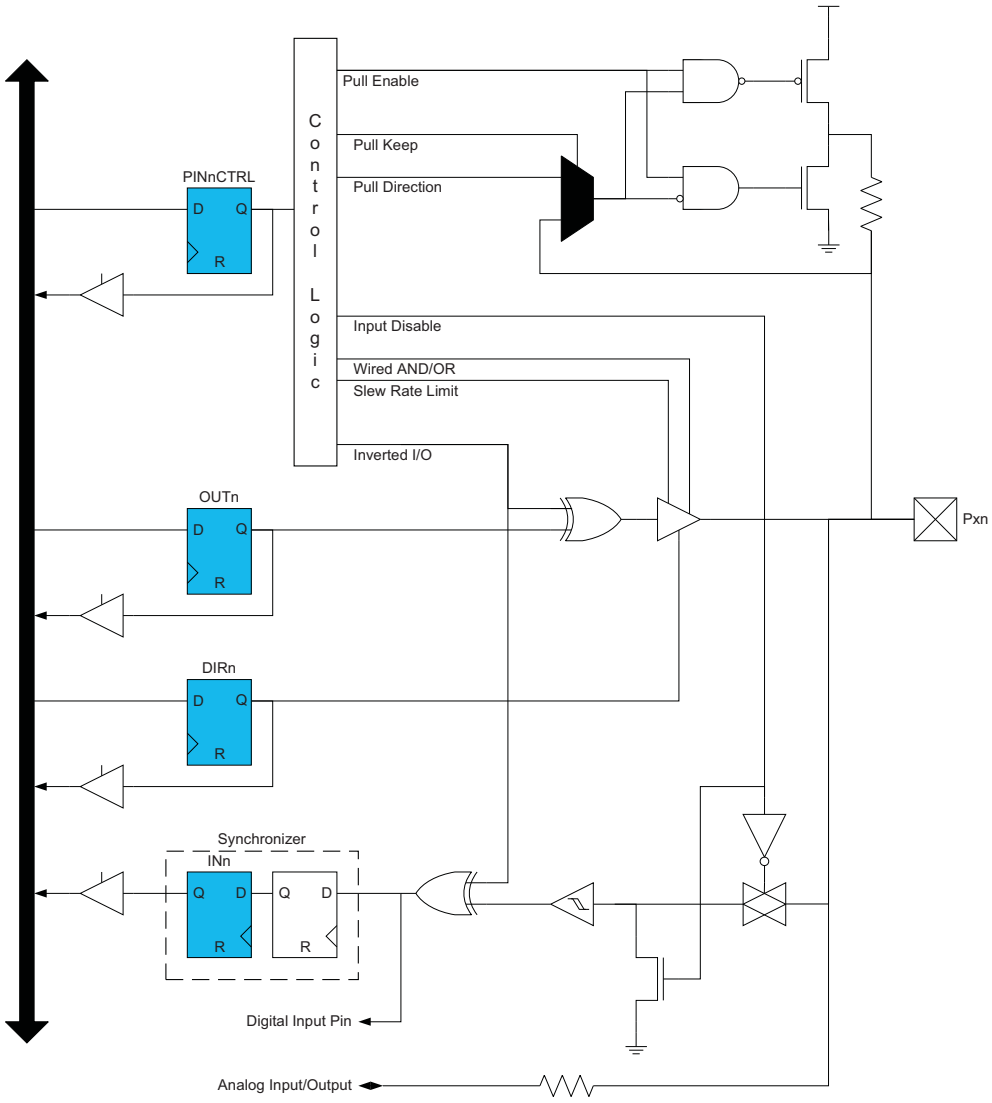
AVR XMEGA microcontrollers have flexible general purpose I/O ports. One port consists of up to eight port pins: pin 0 to 7. Each port pin can be configured as input or output with configurable driver and pull settings. They also implement synchronous and asynchronous input sensing with interrupts and events for selectable pin change conditions. Asynchronous pin-change sensing means that a pin change can wake the device from all sleep modes, included the modes where no clocks are running.

All functions are individual and configurable per pin, but several pins can be configured in a single operation. The pins have hardware read-modify-write (RMW) functionality for safe and correct change of drive value and/or pull resistor configuration. The direction of one port pin can be changed without unintentionally changing the direction of any other pin.

The port pin configuration also controls input and output selection of other device functions. It is possible to have both the peripheral clock and the real-time clock output to a port pin, and available for external use. The same applies to events from the event system that can be used to synchronize and control external functions. Other peripherals, such as analog comparator outputs, USART and timer/counters, can be remapped to selectable pin locations in order to optimize pin-out versus application needs.

Figure 12-1 on page 140 shows the I/O pin functionality and the registers that are available for controlling a pin.

Figure 12-1. General I/O pin functionality.



12.3 I/O pin use and configuration

Each port has one data direction (DIR) register and one data output value (OUT) register that are used for port pin control. The data input value (IN) register is used for reading the port pins. In addition, each pin has a pin configuration (PINnCTRL) register for additional pin configuration.

Direction of the pin is decided by the DIRn bit in the DIR register. If DIRn is written to one, pin n is configured as an output pin. If DIRn is written to zero, pin n is configured as an input pin.

When direction is set as output, the OUTn bit in OUT is used to set the value of the pin. If OUTn is written to one, pin n is driven high. If OUTn is written to zero, pin n is driven low.

The IN register is used for reading pin values. A pin value can always be read regardless of whether the pin is configured as input or output, except if digital input is disabled.

The I/O pins are tri-stated when a reset condition becomes active, even if no clocks are running.

The pin n configuration (PINnCTRL) register is used for additional I/O pin configuration. A pin can be set in a totem-pole, wired-AND, or wired-OR configuration. It is also possible to enable inverted input and output for a pin.

A totem-pole output has four possible pull configurations: totem-pole (push-pull), pull-down, pull-up, and bus-keeper. The bus-keeper is active in both directions. This is to avoid oscillation when disabling the output. The totem-pole

configurations with pull-up and pull-down have active resistors only when the pin is set as input. This feature eliminates unnecessary power consumption. For wired-AND and wired-OR configuration, the optional pull-up and pull-down resistors are active in both input and output directions.

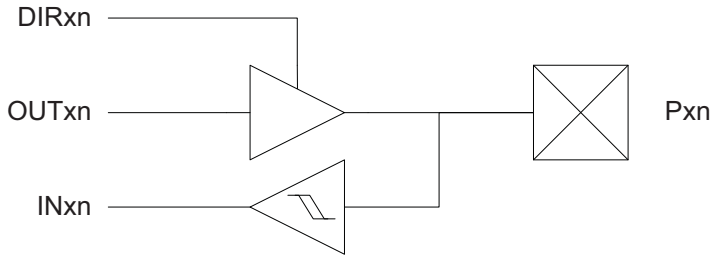
Since pull configuration is configured through the pin configuration register, all intermediate port states during switching of the pin direction and pin values are avoided.

The I/O pin configurations are summarized with simplified schematics in Figure 12-2 to Figure 12-7 on page 143.

12.3.1 Totem-pole

In the totem-pole (push-pull) configuration, the pin is driven low or high according to the corresponding bit setting in the OUT register. In this configuration, there is no current limitation for sink or source other than what the pin is capable of. If the pin is configured for input, the pin will float if no external pull resistor is connected.

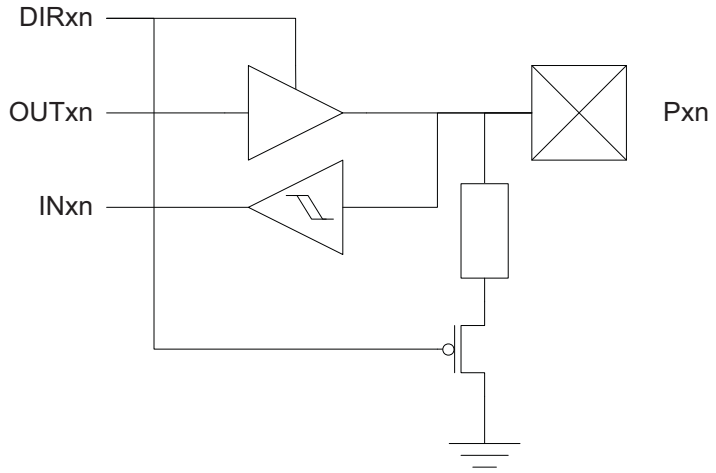
Figure 12-2. I/O pin configuration - Totem-pole (push-pull).



12.3.1.1 Totem-pole with pull-down

In this mode, the configuration is the same as for totem-pole mode, expect the pin is configured with an internal pull-down resistor when set as input.

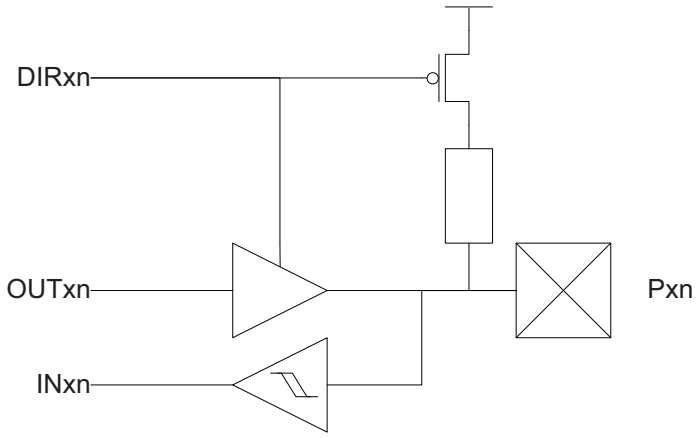
Figure 12-3. I/O pin configuration - Totem-pole with pull-down (on input).



12.3.1.2 Totem-pole with pull-up

In this mode, the configuration is as for totem-pole, expect the pin is configured with internal pull-up when set as input.

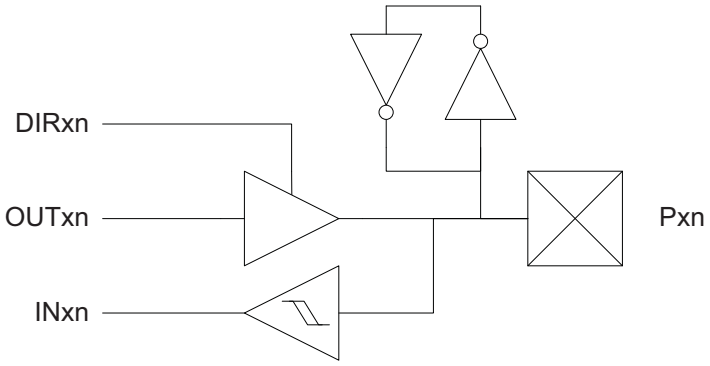
Figure 12-4. I/O pin configuration - Totem-pole with pull-up (on input).



12.3.2 Bus-keeper

In the bus-keeper configuration, it provides a weak bus-keeper that will keep the pin at its logic level when the pin is no longer driven to high or low. If the last level on the pin/bus was 1, the bus-keeper configuration will use the internal pull resistor to keep the bus high. If the last logic level on the pin/bus was 0, the bus-keeper will use the internal pull resistor to keep the bus low.

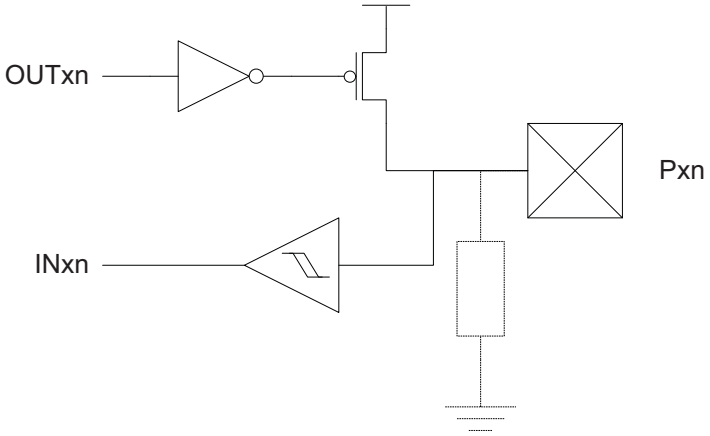
Figure 12-5. I/O pin configuration - Totem-pole with bus-keeper.



12.3.3 Wired-OR

In the wired-OR configuration, the pin will be driven high when the corresponding bits in the OUT and DIR registers are written to one. When the OUT register is set to zero, the pin is released, allowing the pin to be pulled low with the internal or an external pull-resistor. If internal pull-down is used, this is also active if the pin is set as input.

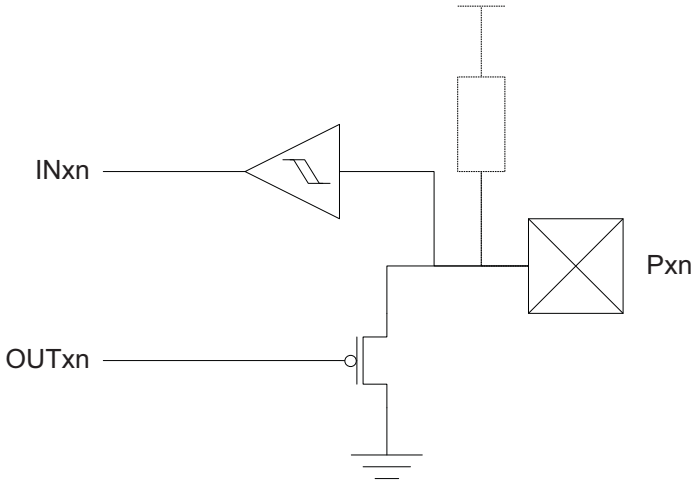
Figure 12-6. Output configuration - Wired-OR with optional pull-down.



12.3.4 Wired-AND

In the wired-AND configuration, the pin will be driven low when the corresponding bits in the OUT and DIR registers are written to zero. When the OUT register is set to one, the pin is released allowing the pin to be pulled high with the internal or an external pull-resistor. If internal pull-up is used, this is also active if the pin is set as input.

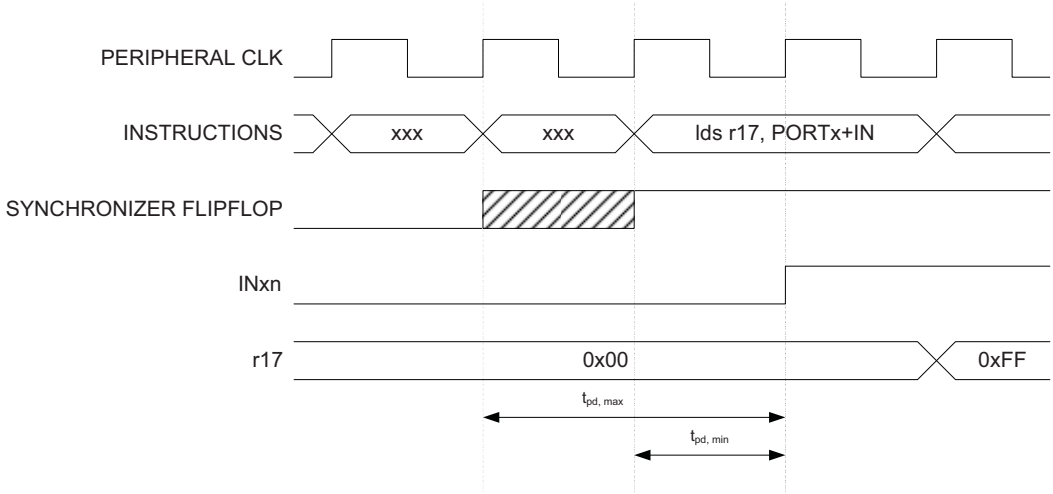
Figure 12-7. Output configuration - Wired-AND with optional pull-up.



12.4 Reading the pin value

Independent of the pin data direction, the pin value can be read from the IN register, as shown in Figure 12-1 on page 140. If the digital input is disabled, the pin value cannot be read. The IN register bit and the preceding flip-flop constitute a synchronizer. The synchronizer introduces a delay on the internal signal line. Figure 12-8 on page 144 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted as $t_{pd,max}$ and $t_{pd,min}$, respectively.

Figure 12-8. Synchronization when reading a pin value.

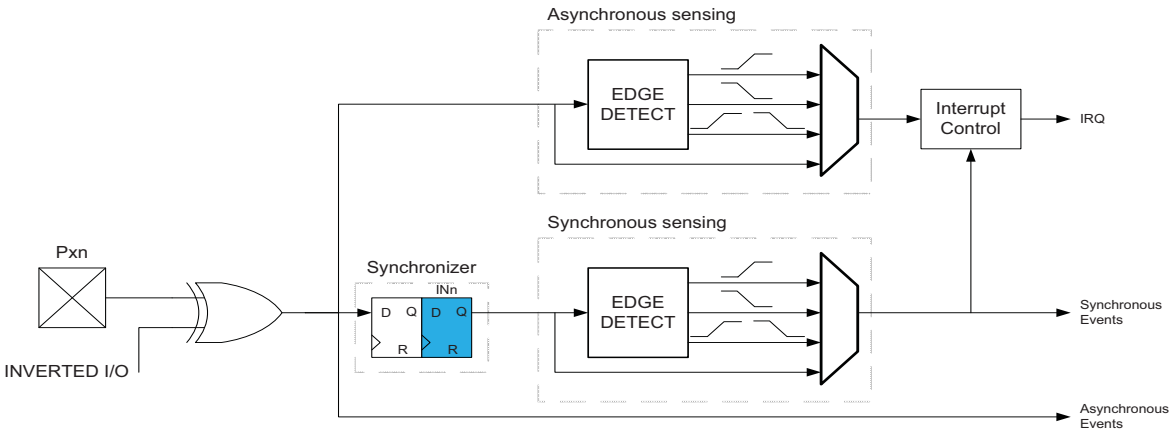


12.5 Input sense configuration

Input sensing is used to detect an edge or level on the I/O pin input. The different sense configurations that are available for each pin are detection of a rising edge, falling edge, or any edge or detection of a low level. High level can be detected by using the inverted input configuration. Input sensing can be used to trigger interrupt requests (IREQ) or events when there is a change on the pin.

The I/O pins support synchronous and asynchronous input sensing. Synchronous sensing requires the presence of the peripheral clock, while asynchronous sensing does not require any clock.

Figure 12-9. Input sensing.



12.6 Port interrupt

Each port has one interrupt vector, and it is configurable which pins on the port will trigger it. Port interrupt must be enabled before it can be used. Which sense configurations can be used to generate interrupt is dependent on whether synchronous or asynchronous input sensing is available for the selected pin.

For synchronous sensing, all sense configurations can be used to generate interrupts. For edge detection, the changed pin value must be sampled once by the peripheral clock for an interrupt request to be generated.

For asynchronous sensing, only port pin 2 on each port has full asynchronous sense support. This means that for edge detection, pin 2 will detect and latch any edge and it will always trigger an interrupt request. The other port pins have limited asynchronous sense support. This means that for edge detection, the changed value must be held until the device wakes up and a clock is present. If the pin value returns to its initial value before the end of the device wake-up time, the device will still wake up, but no interrupt request will be generated.

A low level can always be detected by all pins, regardless of a peripheral clock being present or not. If a pin is configured for low-level sensing, the interrupt will trigger as long as the pin is held low. In active mode, the low level must be held until the completion of the currently executing instruction for an interrupt to be generated. In all sleep modes, the low level must be kept until the end of the device wake-up time for an interrupt to be generated. If the low level disappears before the end of the wake-up time, the device will still wake up, but no interrupt will be generated.

Table 12-1, Table 12-2, and Table 12-3 summarize when interrupt can be triggered for the various input sense configurations.

Table 12-1. Synchronous sense support.

Sense settings	Supported	Interrupt description
Rising edge	Yes	Always triggered
Falling edge	Yes	Always triggered
Any edge	Yes	Always triggered
Low level	Yes	Pin level must be kept unchanged during wake up

Table 12-2. Full asynchronous sense support.

Sense settings	Supported	Interrupt description
Rising edge	Yes	Always triggered
Falling edge	Yes	Always triggered
Any edge	Yes	Always triggered
Low level	Yes	Pin level must be kept unchanged during wake up

Table 12-3. Limited asynchronous sense support.

Sense settings	Supported	Interrupt description
Rising edge	No	-
Falling edge	No	-
Any edge	Yes	Pin level must be kept unchanged during wake up
Low level	Yes	Pin level must be kept unchanged during wake up

12.7 Port event

Port pins can generate a synchronous event when there is a change on the pin, or an asynchronous event, where the pin level is transferred internally without any delay. The sense configurations decide the conditions for each pin to generate synchronous events. Synchronous event generation requires the presence of a peripheral clock, while asynchronous event generation does not requires any clock. For edge sensing, the changed pin value must be sampled once by the peripheral clock for a synchronous event to be generated.

For level sensing, a low-level pin value will not generate synchronous events, and a high-level pin value will continuously generate synchronous events. For synchronous events to be generated on a low level, the pin configuration must be set to inverted I/O.

For asynchronous event generation in all sleep modes where the clock is not present, the digital input buffer of the selected pin must be forced enable. Table 12-6 on page 153 for details.

Table 12-4. Synchronous event sense support.

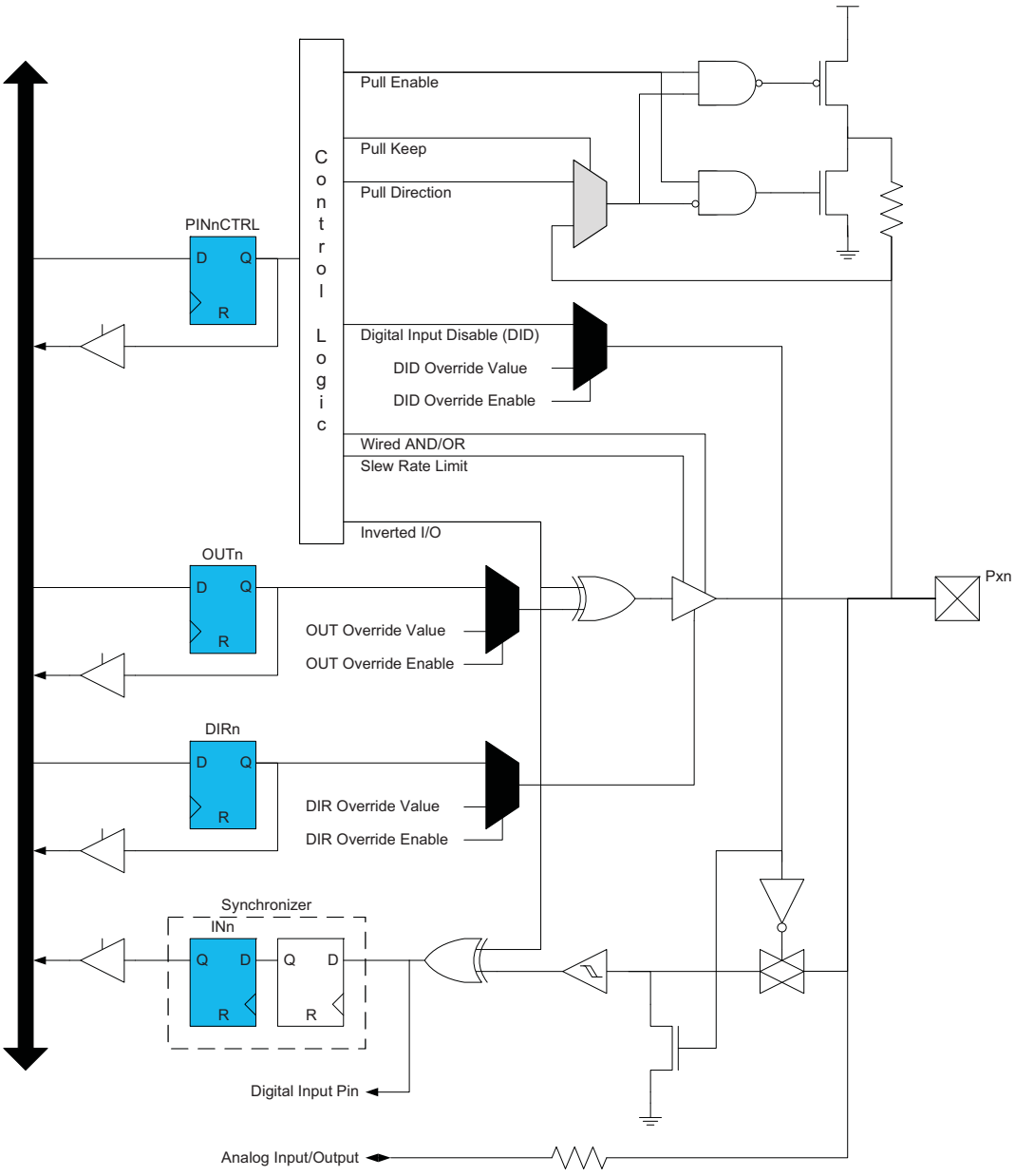
Sense settings	Signal event	Data event
Rising edge	Rising edge	Pin value
Falling edge	Falling edge	Pin value
Both edge	Any edge	Pin value
Low level	Pin value	Pin value

12.8 Alternate port functions

Most port pins have alternate pin functions in addition to being a general purpose I/O pin. When an alternate function is enabled, it might override the normal port pin function or pin value. This happens when other peripherals that require pins are enabled or configured to use pins. If and how a peripheral will override and use pins is described in the section for that peripheral.

The port override signals and related logic (grey) are shown in Figure 12-10 on page 147. These signals are not accessible from software, but are internal signals between the overriding peripheral and the port pin.

Figure 12-10. Port override signals and related logic.



12.9 Slew rate control

Slew rate control can be enabled for each I/O port individually. Enabling the slew rate limiter will typically increase the rise/fall time by 50% to 150%, depending on operating conditions and load. For information about the characteristics of the slew rate limiter, please refer to the device datasheet.

12.10 Clock and event output

It is possible to output the peripheral clock and event channel 0 events to a pin. This can be used to clock, control, and synchronize external functions and hardware to internal device timing. The output port pin is selectable. If an event occurs, it remains visible on the port pin as long as the event lasts, normally one peripheral clock cycle.

12.11 Multi-pin configuration

The multi-pin configuration function is used to configure multiple port pins using a single write operation to only one of the port pin configuration registers. A mask register decides which port pin is configured when one port pin register is written, while avoiding several pins being written the same way during identical write operations.

12.12 Virtual ports

Virtual port registers allow the port registers to be mapped virtually in the bit-accessible I/O memory space. When this is done, writing to the virtual port register will be the same as writing to the real port register. This enables the use of I/O memory-specific instructions, such as bit-manipulation instructions, on a port register that normally resides in the extended I/O memory space. There are four virtual ports, and so four ports can be mapped at the same time.

12.13 Register descriptions – Ports

12.13.1 DIR – Data Direction register

Bit	7	6	5	4	3	2	1	0
+0x00	DIR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DIR[7:0]: Data Direction**
  This register sets the data direction for the individual pins of the port. If DIRn is written to one, pin n is configured as an output pin. If DIRn is written to zero, pin n is configured as an input pin.

12.13.2 DIRSET – Data Direction Set register

Bit	7	6	5	4	3	2	1	0
+0x01	DIRSET[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DIRSET[7:0]: Port Data Direction Set**
  This register can be used instead of a read-modify-write to set individual pins as output. Writing a one to a bit will set the corresponding bit in the DIR register. Reading this register will return the value of the DIR register.

12.13.3 DIRCLR – Data Direction Clear register

Bit	7	6	5	4	3	2	1	0
+0x02	DIRCLR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DIRCLR[7:0]: Port Data Direction Clear**
  This register can be used instead of a read-modify-write to set individual pins as input. Writing a one to a bit will clear the corresponding bit in the DIR register. Reading this register will return the value of the DIR register.

12.13.4 DIRTGL – Data Direction Toggle register

Bit	7	6	5	4	3	2	1	0
+0x03	DIRTGL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DIRTGL[7:0]: Port Data Direction Toggle**
  This register can be used instead of a read-modify-write to toggle the direction of individual pins. Writing a one to a bit will toggle the corresponding bit in the DIR register. Reading this register will return the value of the DIR register.

12.13.5 OUT – Data Output Value register

Bit	7	6	5	4	3	2	1	0
+0x04	OUT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – OUT[7:0]: Port Data Output Value**
  This register sets the data output value for the individual pins of the port. If OUTn is written to one, pin n is driven high. If OUTn is written to zero, pin n is driven low. For this setting to have any effect, the pin direction must be set as output.

12.13.6 OUTSET – Data Output Value Set register

Bit	7	6	5	4	3	2	1	0
+0x05	OUTSET[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – OUTSET[7:0]: Data Output Value Set**
  This register can be used instead of a read-modify-write to set the output value of individual pins to one. Writing a one to a bit will set the corresponding bit in the OUT register. Reading this register will return the value in the OUT register.

12.13.7 OUTCLR – Data Output Value Clear register

Bit	7	6	5	4	3	2	1	0
+0x06	OUTCLR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – OUTCLR[7:0]: Data Output Value Clear**
  This register can be used instead of a read-modify-write to set the output value of individual pins to zero. Writing a one to a bit will clear the corresponding bit in the OUT register. Reading this register will return the value in the OUT register.

12.13.8 OUTTGL – Data Output Value Toggle register

Bit	7	6	5	4	3	2	1	0
+0x07	OUTTGL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – OUTTGL[7:0]: Port Data Output Value Toggle**
  This register can be used instead of a read-modify-write to toggle the output value of individual pins. Writing a one to a bit will toggle the corresponding bit in the OUT register. Reading this register will return the value in the OUT register.

12.13.9 IN – Data Input Value register

Bit	7	6	5	4	3	2	1	0
+0x08	IN[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – IN[7:0]: Data Input Value**
  This register shows the value present on the pins if the digital input driver is enabled. INn shows the value of pin n of the port. The input is not sampled and cannot be read if the digital input buffers are disabled.

12.13.10 INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	–	–	INTLVL[1:0]	
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1:0 – INTLVL[1:0]: Interrupt Level**
  These bits enable port interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.

12.13.11 INTMASK – Interrupt Mask register

Bit	7	6	5	4	3	2	1	0
+0x0A	INTMASK[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – INTMASK[7:0]: Interrupt Mask Bits**
  These bits are used to mask which pins can be used as sources for port interrupt. If INTMASKn is written to one, pin n is used as source for port interrupt.The input sense configuration for each pin is decided by the PINnCTRL registers.

12.13.12 INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x0C	INT7IF	INT6IF	INT5IF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – INTnIF: Interrupt Pin n Flag**
  The INTnIF flag is set when a pin change/state matches the pin's input sense configuration, and the pin is set as source for port interrupt. Writing a one to this flag's bit location will clear the flag.
  For enabling and executing the interrupt, refer to the interrupt level description.

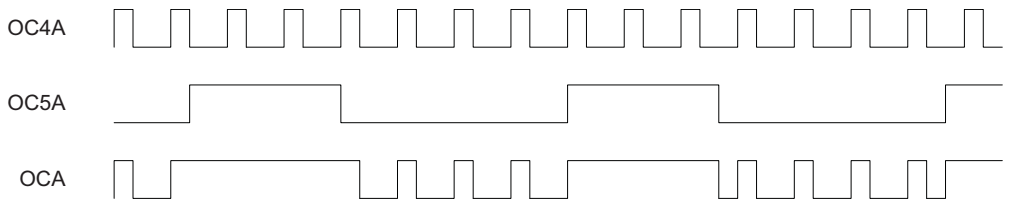
12.13.13 REMAP – Pin Remap register

The pin remap functionality is available for PORTC and PORTD only.

Bit	7	6	5	4	3	2	1	0
+0x0E	–	–	–	USART0	TC4D	TC4C	TC4B	TC4A
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – USART0: USART0 Remap**
  Setting this bit to one will move the pin location of USART0 from Px[3:0] to Px[7:4].
- **Bit 3 – TC4D: Timer/Counter 4 Output Compare D**
  Setting this bit will move the location of OC4D from Px3 to Px7.
- **Bit 2 – TC4C: Timer/Counter 4 Output Compare C**
  Setting this bit will move the location of OC4C from Px2 to Px6.
- **Bit 1 – TC4B: Timer/Counter 4 Output Compare B**
  Setting this bit will move the location of OC4B from Px1 to Px5. If this bit is set and PWM from both timer/counter 4 and timer/counter 5 is enabled, the resulting PWM will be an OR-modulation between the two PWM outputs.
- **Bit 0 – TC4A: Timer/Counter 4 Output Compare A**
  Setting this bit will move the location of OC4A from Px0 to Px4. If this bit is set and PWM from both timer/counter 4 and timer/counter 5 is enabled, the resulting PWM will be an OR-modulation between the two PWM outputs. See Figure 12-11.

Figure 12-11.I/O timer/counter.



12.13.14 PINnCTRL – Pin n Control register

Bit	7	6	5	4	3	2	1	0
+0x10 +n	–	INVEN	OPC[2:0]			ISC[2:0]		
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6 – INVEN: Inverted I/O Enable**
  Setting this bit will enable inverted output and input data on pin n.
- **Bit 5:3 – OPC: Output and Pull Configuration**
  These bits set the output/pull configuration on pin n according to Table 12-5 on page 153.

Table 12-5. Output/pull configuration.

OPC[2:0]	Group configuration	Description	
		Output configuration	Pull configuration
000	TOTEM	Totem-pole	(N/A)
001	BUSKEEPER	Totem-pole	Bus-keeper
010	PULLDOWN	Totem-pole	Pull-down (on input)
011	PULLUP	Totem-pole	Pull-up (on input)
100	WIREDOR	Wired-OR	(N/A)
101	WIREDAND	Wired-AND	(N/A)
110	WIREDORPULL	Wired-OR	Pull-down
111	WIREDANDPULL	Wired-AND	Pull-up

- **Bit 2:0 – ISC[2:0]: Input/Sense Configuration**
  These bits set the input and sense configuration on pin n according to Table 12-6. The sense configuration decides how the pin can trigger port interrupts and events. If the input buffer is not disabled, the input cannot be read in the IN register.

Table 12-6. Input/sense configuration.

ISC[2:0]	Group configuration	Input/Sense configuration
000	BOTHEDGES	Sense both edges
001	RISING	Sense rising edge
010	FALLING	Sense falling edge
011	LEVEL	Sense low level(1)
100	–	Reserved
101	–	Reserved
110	FORCE_ENABLE	Digital input buffer forced enable(2)
111	INTPUT_DISABLE	Digital input buffer disabled(3)

Notes:
1. A low-level pin value will not generate events, and a high-level pin value will continuously generate events.
2. Only PORTA - PORTD support the input buffer force enable option. If the pin is not used for asynchronous event generation, it is recommended to not use this configuration.
3. Only PORTA - PORTD support the input buffer disable option. If the pin is used for analog functionality, such as AC or ADC, it is recommended to configure the pin to INPUT_DISABLE.

12.14 Register descriptions – Port configuration

12.14.1 MPCMASK – Multi-Pin Configuration Mask register

Bit	7	6	5	4	3	2	1	0
+0x00	MPCMASK[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – MPCMASK[7:0]: Multi-Pin Configuration Mask**
  The MPCMASK register enables configuration of several pins of a port at the same time. Writing a one to bit n makes pin n part of the multi-pin configuration. When one or more bits in the MPCMASK register is set, writing any of the PINnCTRL registers will update only the PINnCTRL registers matching the mask in the MPCMASK register for that port. The MPCMASK register is automatically cleared after any PINnCTRL register is written.

12.14.2 CLKOUT – Clock Output register

Bit	7	6	5	4	3	2	1	0
+0x04	CLKEVPIN	RTCOUT[1:0]		-	CLKOUTSEL[1:0]		CLKOUT[1:0]	
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – CLKEVPIN: Clock and Event Output Pin Select**
  Setting this pin enables output of clock and event pins on port pin 4 instead of port pin 7.
- **Bit 6:5 – RTCOUT[1:0]: RTC Clock Output Enable**
  Setting this bit enables output of the RTC clock source according to Table 12-7.

Table 12-7. Event output pin selection.

RTCOUT[1:0]	Group configuration	Description
00	OFF	RTC clock output disabled
01	PC	RTC clock output on PORTC, pin 6
10	PD	RTC clock output on PORTD, pin 6
1x	PR	RTC clock output on PORTR, pin 0

- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bits 3:2 – CLKOUTSEL[1:0]: Clock Output Select**
  These bits are used to select which of the peripheral clocks will be output to the port pin if CLKOUT is configured.

Table 12-8. Event output clock selection.

CLKOUTSEL[1:0]	Group configuration	Description
00	CLK1X	CLK_{PER} output to pin
01	CLK2X	CLK_{PER2} output to pin
10	CLK4X	CLK_{PER4} output to pin
11	–	Reserved

- **Bit 1:0 – CLKOUT[1:0]: Clock Output Port**
  These bits decide which port the peripheral clock will be output to. Pin 7 on the selected port is the default used. The CLKOUT setting will override the EVOUT setting. Thus, if both are enabled on the same port pin, the peripheral clock will be visible. The port pin must be configured as output for the clock to be available on the pin. Table 12-9 shows the possible configurations.

Table 12-9. Clock output port configurations.

CLKOUT[1:0]	Group configuration	Description
00	OFF	Clock output disabled
01	PC	Clock output on PORTC
10	PD	Clock output on PORTD
11	PE	Clock output on PORTE

12.14.3 ACEVOUT – Analog Comparator and Event Output register

Bit	7	6	5	4	3	2	1	0
+0x06	ACOUT[1:0]		EVOUT[1:0]		EVASYEN	EVOUTSEL[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – ACOUT[1:0]: Analog Comparator Output Port**
  These bits decide which port analog comparator will be output to, according to Table 12-10.
  The analog compare outputs are enabled in the module itself.

Table 12-10. Analog Comparator output port selection.

ACOUT[1:0]	Group configuration	Description
00	PA	Analog Comparator outputs on PORTA
01	PC	Analog Comparator outputs on PORTC
10	PD	Analog Comparator outputs on PORTD
11	PR	Analog Comparator outputs on PORTR

- **Bit 5:4 – EVOUT[1:0]: Event Output Port**
  These bits decide which port event channel 0 from the event system will be output to, according to Table 12-11. Pin 7 on the PORTC and PORTD, or pin 0 on the PORTR, is the default used, and the CLKOUT bits must be set differently from those of EVOUT. The port pin must be configured as output for the event to be available on the pin.

Table 12-11. Event output pin selection.

EVOUT[1:0]	Group configuration	Description
00	OFF	Event output disabled
01	PC	Event channel 0 output on PORTC
10	PD	Event channel 0 output on PORTD
11	PR	Event channel 0 output on PORTR

- **Bit 3 – EVASYEN: Asynchronous Event Enabled**
  Setting this bit enables the asynchronous event output. The event channel selected by EVOUTSEL bits must be set accordingly.
- **Bit 2:0 - EVOUTSEL[2:0]: Event Channel Output Selection**
  These bits define which channel from the event system is output to the port pin, according to Table 12-12.

Table 12-12. Event channel output selection.

EVOUTSEL[2:0]	Group configuration	Description
000	0	Event channel 0 output to pin
001	1	Event channel 1 output to pin
010	2	Event channel 2 output to pin
011	3	Event channel 3 output to pin
100	4	Event channel 4 output to pin
101	5	Event channel 5 output to pin
110	6	Event channel 6 output to pin
111	7	Event channel 7 output to pin

12.14.4 SRLCTRL – Slew Rate Limit Control register

Bit	7	6	5	4	3	2	1	0
+0x07	SRLENR	–	–	–	SRLEND	SRLENC	–	SRLENA
Read/Write	R/W	R	R	R	R/W	R/W	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – SRLENR: Slew Rate Limit Enable on PORTR**
  Setting this bit will enable slew rate limiting on port R.
- **Bit 6:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3 – SRLENRD: Slew Rate Limit Enable on PORTD**
  Setting this bit will enable slew rate limiting on port D.
- **Bit 2 – SRLENRC: Slew Rate Limit Enable on PORTC**
  Setting this bit will enable slew rate limiting on port C.
- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

- **Bit 0 – SRLENRA: Slew Rate Limit Enable on PORTA**
  Setting this bit will enable slew rate limiting on port A.

12.15 Register descriptions – Virtual port

12.15.1 DIR – Data Direction register

Bit	7	6	5	4	3	2	1	0
+0x00	DIR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DIR[7:0]: Data Direction**
  This register sets the data direction for the individual pins in the port. When a port is mapped as virtual, accessing this register is identical to accessing the actual DIR register for the port.

12.15.2 OUT – Data Output Value register

Bit	7	6	5	4	3	2	1	0
+0x01	OUT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – OUT[7:0]: Data Output Value**
  This register sets the data output value for the individual pins in the port. When a port is mapped as virtual, accessing this register is identical to accessing the actual OUT register for the port.

12.15.3 IN – Data Input Value register

Bit	7	6	5	4	3	2	1	0
+0x02	IN[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – IN[7:0]: Data Input Value**
  This register shows the value present on the pins if the digital input buffer is enabled. This register sets the data direction for the individual pins in the port. When a port is mapped as virtual, accessing this register is identical to accessing the actual IN register for the port.

12.15.4 INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x03	INT7IF	INT6IF	INT5IF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – INTnIF: Interrupt Pin n Flag**
  The INTnIF flag is set when a pin change/state matches the pin's input sense configuration, and the pin is set as source for port interrupt. Writing a one to this flag's bit location will clear the flag. For enabling and executing the interrupt, refer to the interrupt level description. When a port is mapped as virtual, accessing this register is identical to accessing the actual INTFLAGS register for the port.

12.16 Register summary – Ports

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DIR	DIR[7:0]								149
+0x01	DIRSET	DIRSET[7:0]								149
+0x02	DIRCLR	DIRCLR[7:0]								149
+0x03	DIRTGL	DIRTGL[7:0]								149
+0x04	OUT	OUT[7:0]								150
+0x05	OUTSET	OUTSET[7:0]								150
+0x06	OUTCLR	OUTCLR[7:0]								150
+0x07	OUTTGL	OUTTGL[7:0]								150
+0x08	IN	IN[7:0]								151
+0x09	INTCTRL	-	-	-	-	-	-	INTLVL[1:0]		151
+0x0A	INTMASK	INTMASK[7:0]								151
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	INTFLAGS	INT7IF	INT6IF	INT5IF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF	151
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	REMAP	-	-	-	USART0	TC4D	TC4C	TC4B	TC4A	152
+0x10	PIN0CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x11	PIN1CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x12	PIN2CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x13	PIN3CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x14	PIN4CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x15	PIN5CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x16	PIN6CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x17	PIN7CTRL	-	INVEN	OPC[2:0]			ISC[2:0]			152
+0x18	Reserved	-	-	-	-	-	-	-	-	
+0x19	Reserved	-	-	-	-	-	-	-	-	
+0x1A	Reserved	-	-	-	-	-	-	-	-	
+0x1B	Reserved	-	-	-	-	-	-	-	-	
+0x1C	Reserved	-	-	-	-	-	-	-	-	
+0x1D	Reserved	-	-	-	-	-	-	-	-	
+0x1E	Reserved	-	-	-	-	-	-	-	-	
+0x1F	Reserved	-	-	-	-	-	-	-	-	

12.17 Register summary – Port configuration

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	MPCMASK	MPCMASK[7:0]								154
+0x01	Reserved	-	-	-	-	-	-	-	-	
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	Reserved	-	-	-	-	-	-	-	-	
+0x04	CLKOUT	CLKEVPIN	RTCOUT[1:0]		-	CLKOUTSEL[1:0]		CLKOUT[1:0]		154
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	ACEVOUT	ACOUT[1:0]		EVOUT[1:0]		EVASYEN	EVCTRL[2:0]			155
+0x07	SRLCTRL	SRLENR	-	-	-	SRLEND	SRLENC	-	SRLENA	156

12.18 Register summary – Virtual ports

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DIR	DIR[7:0]								157
+0x01	OUT	OUT[7:0]								157
+0x02	IN	IN[7:0]								157
+0x03	INTFLAGS	INT7IF	INT6IF	INT5IF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF	157

12.19 Interrupt vector summary – Ports

Table 12-13. USART interrupt vectors and their word offset address

Offset	Source	Interrupt description
0x00	INT_vect	Port Interrupt vector offset

13. TC4/5 – 16-bit Timer/Counter Type 4 and 5

13.1 Features

- 16-bit timer/counter
- 32-bit timer/counter support by cascading two timer/counters
- Up to four compare or capture (CC) channels
  - Four CC channels for timer/counters of type 4
  - Two CC channels for timer/counters of type 5
- Double buffered timer period setting
- Double buffered capture or compare channels
- Waveform generation:
  - Frequency generation
  - Single-slope pulse width modulation
  - Dual-slope pulse width modulation
- Input capture:
  - Input capture with noise cancelling
  - Frequency capture
  - Pulse width capture
  - 32-bit input capture
- Timer overflow and error interrupts/events
- One compare match or input capture interrupt/event per CC channel
- Can be used with event system for:
  - Quadrature decoding
  - Count and direction control
  - Capture
- Can be used with EDMA and to trigger EDMA transactions
- High-resolution extension
  - Increases frequency and waveform resolution by 4x (2-bit) or 8x (3-bit)
- Waveform extension:
  - Low- and high-side output with programmable dead-time insertion (DTI)
- Fault extension
  - Event controlled fault protection for safe disabling of drivers

13.2 Overview

Atmel AVR XMEGA devices have a set of flexible, 16-bit timer/counters (TC). Their capabilities include accurate program execution timing, frequency and waveform generation, and input capture with time and frequency measurement of digital signals. Two timer/counters can be cascaded to create a 32-bit timer/counter with optional 32-bit capture.

A timer/counter consists of a base counter and a set of compare or capture (CC) channels. The base counter can be used to count clock cycles or events. It has direction control and period setting that can be used for timing. The CC channels can be used together with the base counter to do compare match control, frequency generation, and pulse width waveform modulation (PWM) generation, as well as various input capture operations. A timer/counter can be configured for either capture or compare functions, but cannot perform both at the same time.

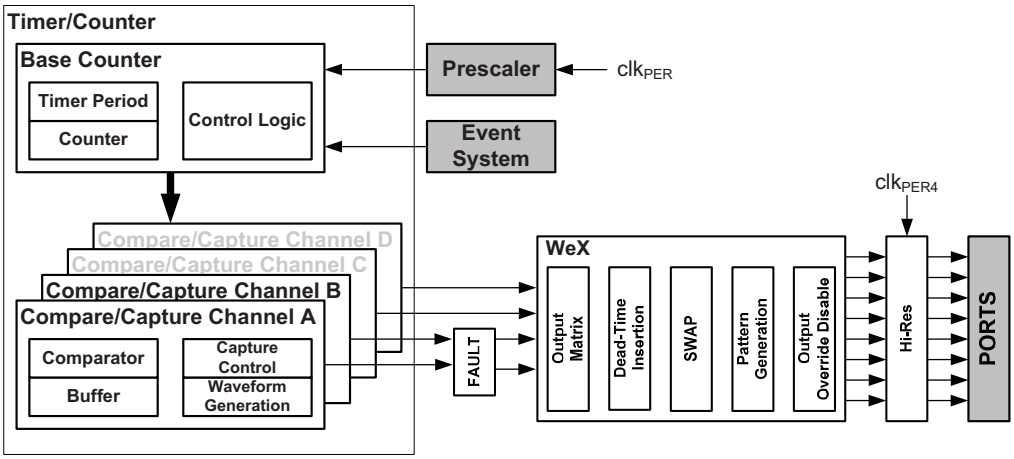
A timer/counter can be clocked and timed from the peripheral clock with optional prescaling or from the event system. The event system can also be used for direction control and capture trigger or to synchronize operations.

There are two differences between timer/counter type 4 and type 5. Timer/counter 4 has four CC channels, and timer/counter 5 has two CC channels. All information related to CC channels 3 and 4 is valid only for timer/counter 4. Both timer/counter 4 and 5 can be in 8-bit mode, allowing the application to double the number of compare and capture channels that then get 8-bit resolution.

Some timer/counters have extensions to enable more specialized waveform and frequency generation. The waveform extension (WeX) is intended for motor control, ballast, LED, H-bridge, power converters, and other types of power control applications. It enables low- and high-side output with optional dead-time insertion. It can also generate a synchronized bit pattern across the port pins. The high-resolution (Hi-Res) extension can increase the waveform resolution by four or eight times by using an internal clock source running four times faster than the peripheral clock. The fault extension (FAULT) enables fault protection for safe and deterministic handling, disabling and/or shut down of external drivers.

A block diagram of the 16-bit timer/counter with extensions and closely related peripheral modules (in grey) is shown in Figure 13-1 on page 161.

Figure 13-1. 16-bit timer/counter and closely related peripherals.



13.2.1 Definitions

The following definitions are used throughout the documentation:

Table 13-1. Timer/counter definitions.

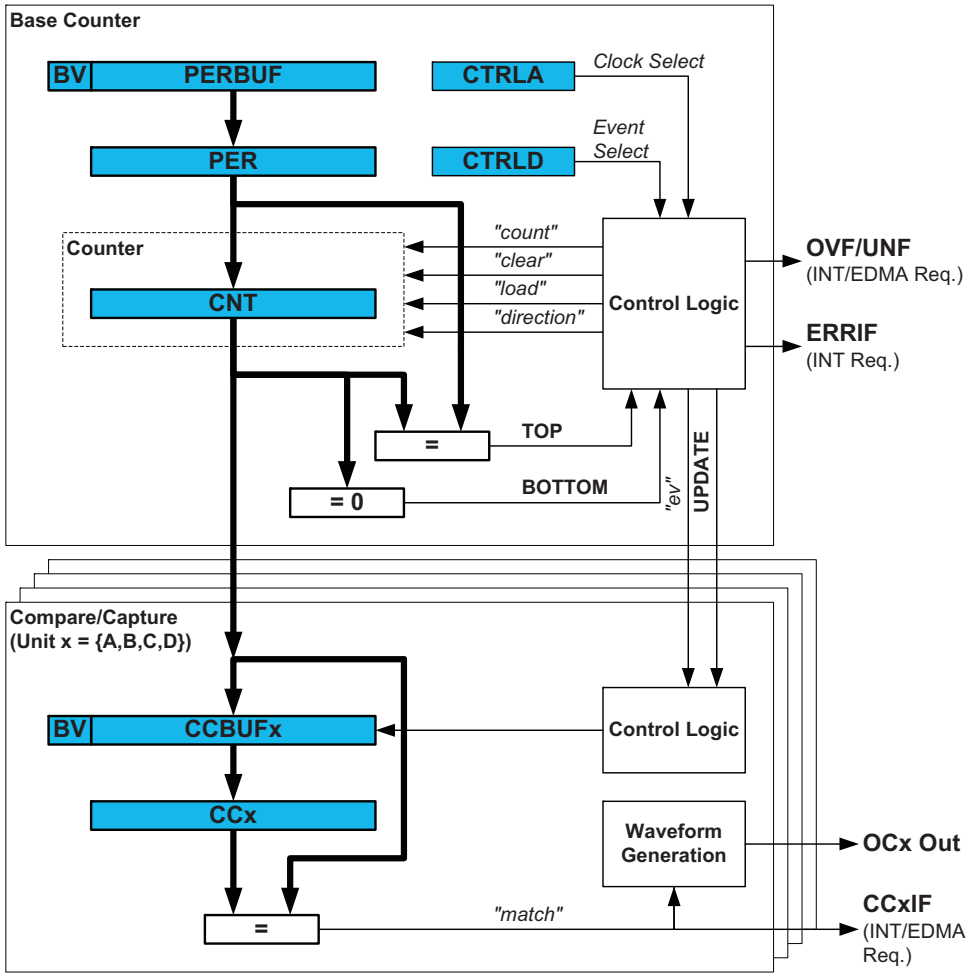
Name	Description
BOTTOM	The counter reaches BOTTOM when it becomes zero (one in single slope counting-up mode).
MAX	The counter reaches MAXimum when it becomes all ones.
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be equal to the period (PER) or the compare channel A (CCA) register setting. This is selected by the waveform generator mode.
UPDATE	The timer/counter signals an update when it reaches BOTTOM or TOP, depending on the direction settings.

In general, the term “timer” is used when the timer/counter clock control is handled by an internal source, and the term “counter” is used when the clock control is handled externally (e.g. counting external events). When used for compare operations, the CC channels are referred to as “compare channels.” When used for capture operations, the CC channels are referred to as “capture channels.”

13.3 Block diagram

Figure 13-2 shows a detailed block diagram of the timer/counter without the extensions.

Figure 13-2. Timer/counter block diagram.



The counter register (CNT), period registers with buffer (PER and PERBUF), and compare and capture registers with buffers (CCx and CCxBUF) are 16-bit registers. All buffer register have a buffer valid (BV) flag that indicates when the buffer contains a new value.

During normal operation, the counter value is continuously compared to zero and the period (PER) value to determine whether the counter has reached TOP or BOTTOM.

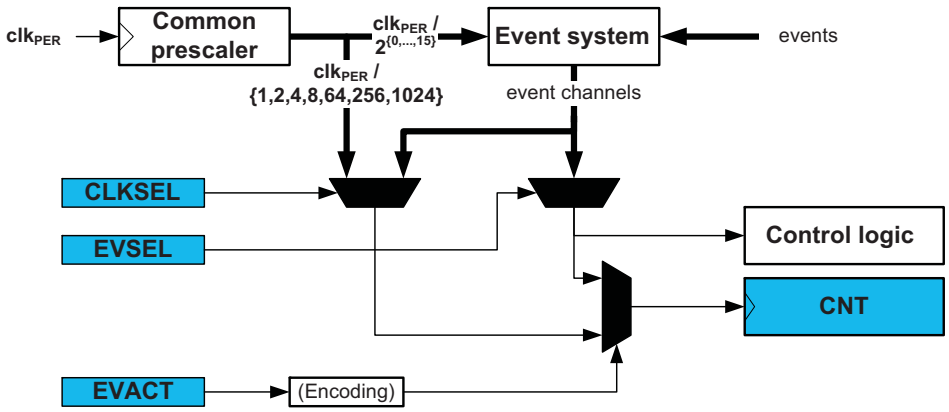
The counter value is also compared to the CCx registers. These comparisons can be used to generate interrupt requests, request EDMA transactions or generate events for the event system. The waveform generator modes use these comparisons to set the waveform period or pulse width.

A prescaled peripheral clock and events from the event system can be used to control the counter. The event system is also used as a source to the input capture. Combined with the quadrature decoding functionality in the event system (QDEC), the timer/counter can be used for quadrature decoding.

13.4 Clock and event sources

The timer/counter can be clocked from the peripheral clock (clk_{PER}), or the event system, and Figure 13-3 shows the clock and event selection.

Figure 13-3. Clock and event selection.



The peripheral clock is fed into a common prescaler (common for all timer/counters in a device). Prescaler outputs from 1 to 1/1024 are directly available for selection by the timer/counter. In addition, the whole range of prescaling from 1 to 2^{15} times are available through the event system.

Clock selection (CLKSEL) selects one of the prescaler outputs directly or an event channel as the counter (CNT) input. This is referred to as normal operation of the counter. For details, refer to "Normal operation" on page 164. By using the event system, any event source, such as an external clock signal on any I/O pin, may be used as the clock input.

In addition, the timer/counter can be controlled via the event system. The event selection (EVSEL) and event action (EVACT) settings are used to trigger an event action from one or more events. This is referred to as event action controlled operation of the counter. For details, refer to "Event action controlled operation" on page 165. When event action controlled operation is used, the clock selection must be set to use an event channel as the counter input.

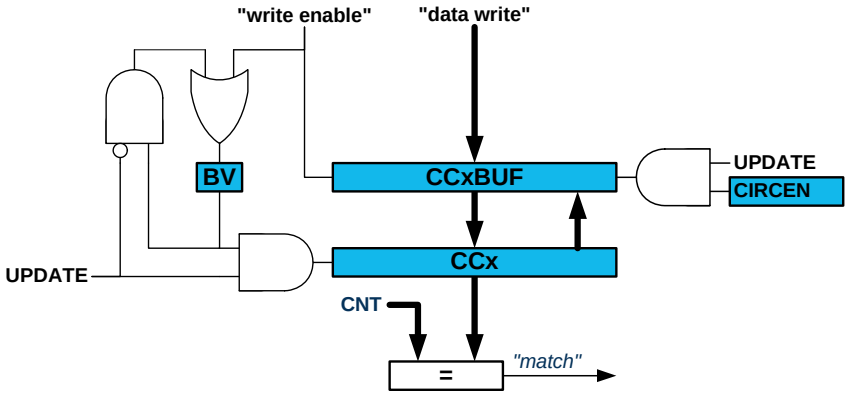
13.5 Double buffering

The period register and the CC registers are all double buffered, with circular buffer option on compare channel A. Each buffer register has a buffer valid (BV) flag, which indicates that the buffer register contains a valid, i.e. new value that can be copied into the corresponding period or CC register. When the period register and CC channels are used for a compare operation, the buffer valid flag is set when data is written to the buffer register and cleared on an UPDATE condition.

Circular buffer option can be enabled for both compare and waveform generation modes. On update condition, the period and CCA registers can be optionally stored in their corresponding buffers. In the same way, the corresponding buffers registers values are stored in period and CCA registers on the same update condition.

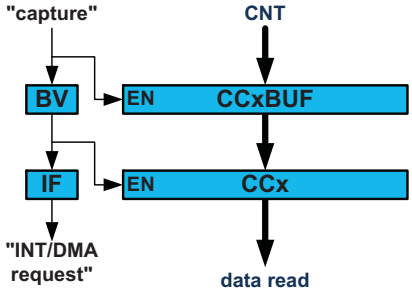
This is shown for a compare register in Figure 13-4 on page 164.

Figure 13-4. Period and compare double buffering.



When the CC channels are used for a capture operation, a similar double buffering mechanism is used, but in this case the buffer valid flag is set on the capture event, as shown in Figure 13-5. For input capture, the buffer register and the corresponding CCx register act like a FIFO. When the CC register is empty or read, any content in the buffer register is passed to the CC register. The buffer valid flag is passed to set the CCx interrupt flag (IF) and generate the optional interrupt.

Figure 13-5. Capture double buffering.



Both the CCx and CCxBUF registers are available as an I/O register. This allows initialization and bypassing of the buffer register and the double buffering function.

13.6 Counter operation

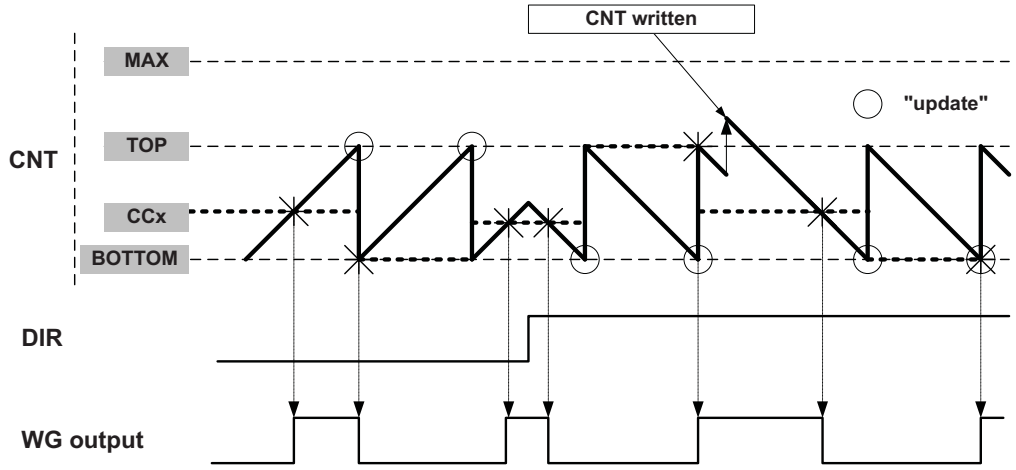
Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each timer/counter clock input.

The timer/counter can be enabled in counting up or down in normal, single-slope or dual-slop operation.

13.6.1 Normal operation

In normal operation, the counter will count in the direction set by the direction (DIR) bit for each clock until it reaches TOP or BOTTOM. When up-counting and TOP is reached, the counter will be set to zero when the next clock is given. When down-counting, the counter is reloaded with the period register value when BOTTOM is reached.

Figure 13-6. Normal operation.



As shown in Figure 13-6, it is possible to change the counter value when the counter is running. The write access has higher priority than count, clear, or reload, and will be immediate. The direction of the counter can also be changed during normal operation.

Normal operation must be used when using the counter as timer base for the input capture. When a waveform generation (WG) mode is enabled, the waveform is output to a pin. For details, refer to "Waveform generation" on page 169.

13.6.2 Event action controlled operation

The event selection and event action settings can be used to control the counter from the event system. For the counter, the following event actions can be selected:

- Event system controlled up/down counting
  - Event n will be used as count enable
  - Event n+1 will be used to select between up (1) and down (0). The pin configuration must be set to low level sensing
- Event system controlled quadrature decode counting

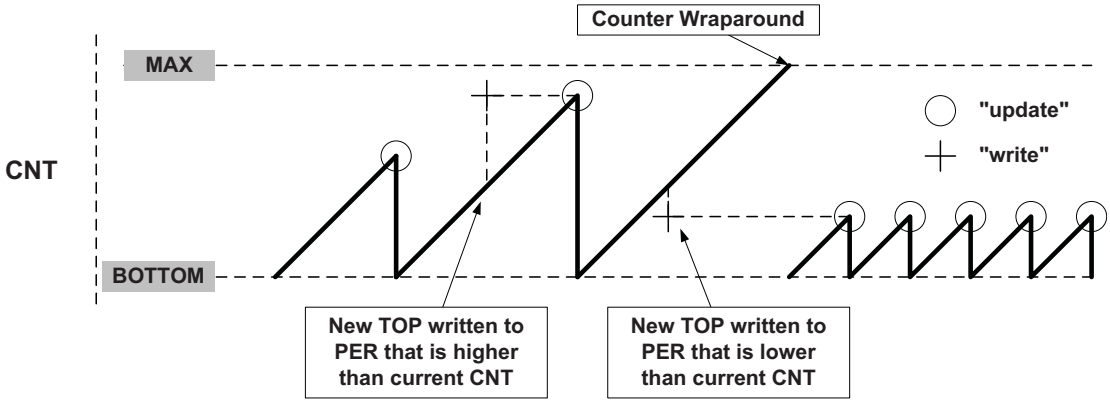
13.6.3 32-bit operation

Two timer/counters can be used together to enable 32-bit counter operation. By using two timer/counters, the overflow event from one timer/counter (least-significant timer) can be routed via the event system and used as the clock input for another timer/counter (most-significant timer).

13.6.4 Changing the period

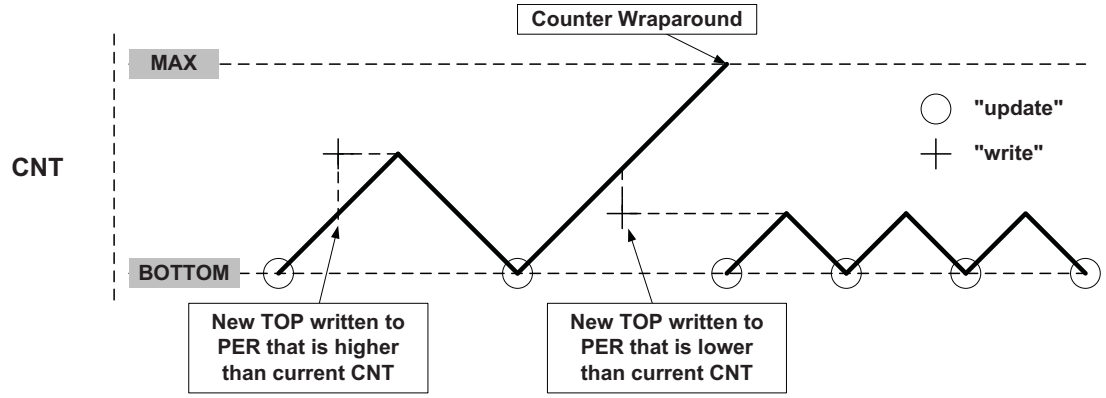
The counter period is changed by writing a new TOP value to the period register. If double buffering is not used, any period update is immediate, as shown in Figure 13-7 on page 166.

Figure 13-7. Unbuffered single-slope operation.



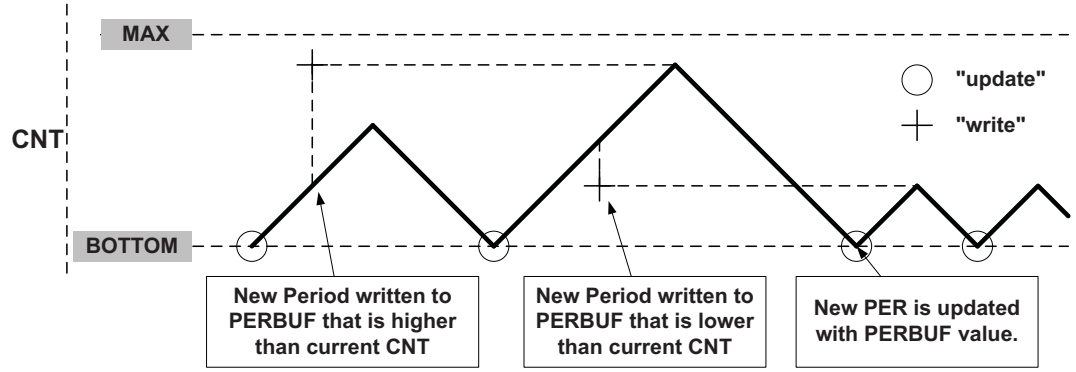
A counter wraparound can occur in any mode of operation when up-counting without buffering, as shown in Figure 13-8. This due to the fact that CNT and PER are continuously compared, and if a new TOP value that is lower than current CNT is written to PER, it will wrap before a compare match happen.

Figure 13-8. Unbuffered dual-slope operation.



When double buffering is used, the buffer can be written at any time and still maintain correct operation. The period register is always updated on the UPDATE condition, as shown for dual-slope operation in Figure 13-9. This prevents wraparound and the generation of odd waveforms.

Figure 13-9. Changing the period using buffering.



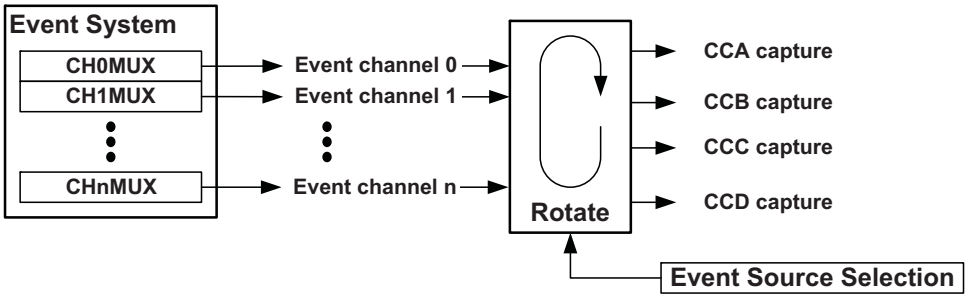
13.7 Capture channel

The CC channels can be used as capture channels to capture external events and give them a timestamp. For input capture, the counter must be set for normal operation.

Events are used to trigger the input capture; i.e., any events from the event system, including pin change from any pin, can trigger am input capture. The event source select setting selects which event channel will trigger CC channel A. The subsequent event channels then trigger input capture on subsequent CC channels, if configured. For example, setting the event source select to event channel 2 results in CC channel A capture being triggered by event channel 2, CC channel B triggered by event channel 3, and so on.

For timer/counters with fault extension, an input channel capture can also be triggered by a fault condition. If the CAPTA or CAPTB fault action is enabled in fault extension unit, a fault will trigger a CC channel capture.

Figure 13-10. Event source selection for input capture.



The event action setting in the timer/counter will determine the type of capture that is done.

The CC channels must be enabled individually before capture can be done. When the capture condition occur, the timer/counter will time-stamp the event by copying the current CNT value in the count register into the enabled CC channel register.

When an I/O pin is used as an event source for the capture, the pin must be configured for edge sensing. For details on sense configuration on I/O pins, refer to “Input sense configuration” on page 144.

For details on event channels source selection, refer to Table 13-10 on page 177.

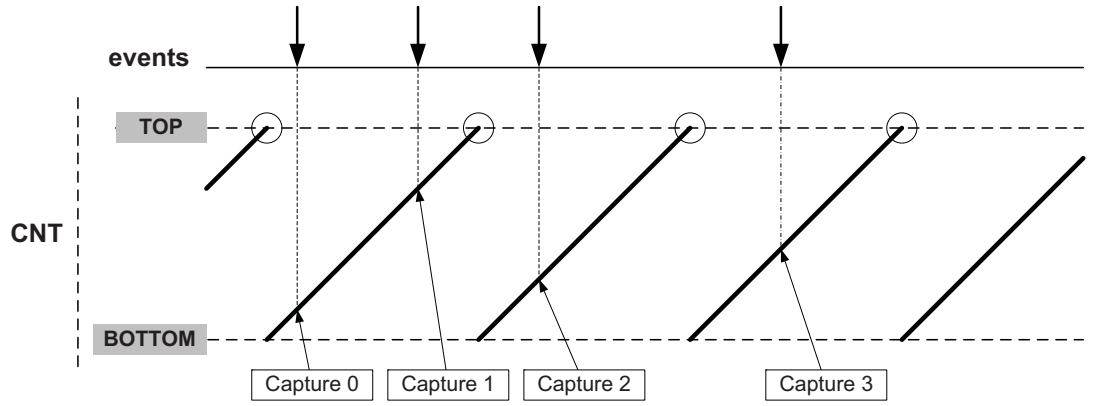
13.7.1 Input capture

Selecting the input capture event action makes the enabled capture channel perform an input capture on an event. The interrupt flags will be set and indicate that there is a valid capture result in the corresponding CC register. At the same time, the buffer valid flags indicate valid data in the buffer registers.

A capture is enabled by enabling the corresponding CC channel in capture mode. The capture can be enabled in any timer/counter operation mode. The Figure 13-11 shows four capture events for one capture channel when the timer/counter is counting from BOTTOM to TOP.

A special case occurs when the timer/counter is set in dual slope mode. When DSBOTH operation is enabled, the DIR is stored as most-significant bit of the captured value. In all other cases, the MSB bit of the timer/counter is stored as most-significant bit of the captured value.

Figure 13-11. Input capture timing.



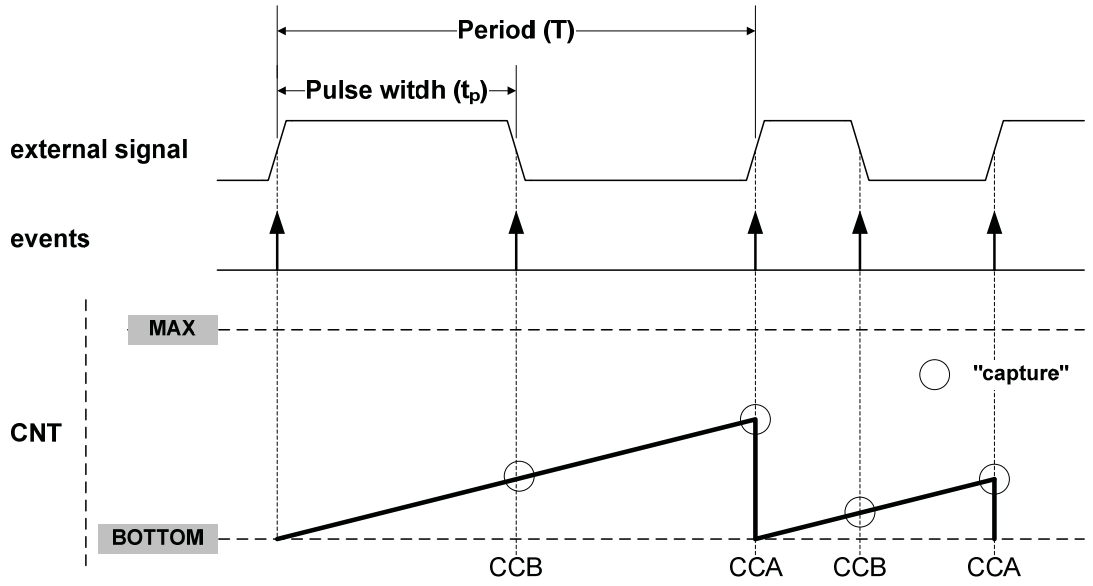
When selecting the pulse width and frequency capture event action, the enabled CCA or CCB channels perform input captures on positive edge event on CCA channel and on negative edge event on CCB channel. Counter restart is performed on positive edge event. This enables measurement of signal pulse width and frequency directly. The CCA capture result will be the period (T) from the previous timer/counter restart until the event occurred. This can be used to calculate the frequency (f) of the signal:

$$f = \frac{1}{T}$$

The CCB capture result will be pulse width (t_p) of the signal. The event source must be an I/O pin, and the sense configuration for the pin must be set to generate an event on both edges.

Figure 13-12 on page 168 shows an example where the period and pulse width of an external signal is measured twice.

Figure 13-12. Frequency and pulse width capture of an external signal.



13.7.2 32-bit input capture

Two timer/counters can be used together to enable true 32-bit input capture. In a typical 32-bit input capture setup, the overflow event of the least-significant timer is connected via the event system and used as the clock input for the most-significant timer.

The most-significant timer will be updated one peripheral clock period after an overflow occurs for the least-significant timer. To compensate for this, the capture event for the most-significant timer must be equally delayed by setting the event delay bit for this timer.

13.7.3 Capture overflow

The timer/counter can detect buffer overflow of the input capture channels. When both the buffer valid flag and the capture interrupt flag are set and a new capture event is detected, there is nowhere to store the new timestamp. If a buffer overflow is detected, the new value is rejected, the error interrupt flag is set, and the optional interrupt is generated.

13.8 Compare channel

The CC channels can be used to compare the counter (CNT) value with the CC channels (CCx) register value. If CNT equals CCx, the comparator signals a compare match. The compare match will set the CC channel's interrupt flag at the next timer/counter clock cycle, and the event and the optional interrupt are generated.

The compare buffer register provides double buffer capability equivalent to that for the period buffer. The double buffering synchronizes the update of the CCx register with the buffer value on the UPDATE condition. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensure glitch-free output.

13.8.1 Waveform generation

The compare channels can be used for waveform generation on the corresponding port pins. To make the waveform visible on the connected port pin, the following requirements must be fulfilled:

1. A waveform generation mode must be selected.
2. Event actions must be disabled.
3. The CC channels used must be enabled. This will override the corresponding port pin output register.
4. The direction for the associated port pin must be set to output.

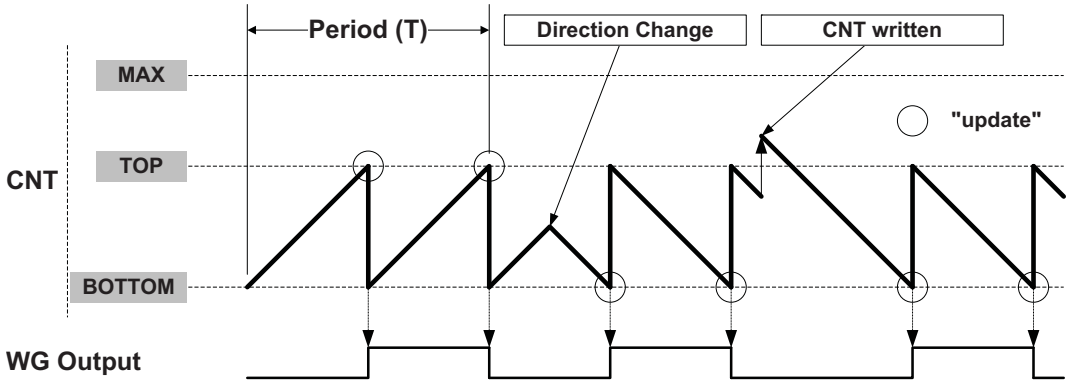
Inverted waveform output is achieved by setting the invert output bit for the port pin.

For timer/counter with fault extension, edge aligned pulses (left or right) is achieved by setting the polarity bits. For more details, refer to the fault extension unit description.

13.8.2 Frequency (FRQ) waveform generation

For frequency generation the period time (T) is controlled by the CCA register instead of PER. The waveform generation (WG) output is toggled on each compare match between the CNT and CCA registers, as shown in Figure 13-13.

Figure 13-13. Frequency waveform generation.



The waveform frequency (f_{FRQ}) is defined by the following equation:

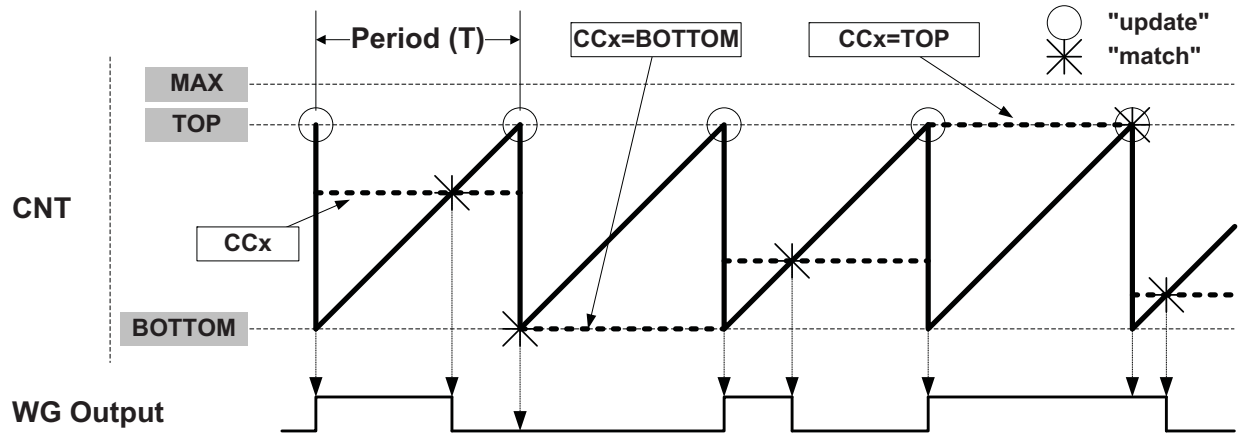
$$f_{FRQ} = \frac{fclk_{PER}}{2N(CCA+1)}$$

N represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the peripheral clock frequency ($fclk_{PER}$) when CCA is set to zero (0x0000) and no prescaling is used. This also applies when using the hi-res extension, since this increases the resolution and not the frequency.

13.8.3 Single-slope PWM generation

For single-slope PWM generation, the period (T) is controlled by PER, while CCx registers control the duty cycle of the WG output. Figure 13-14 shows how the counter counts from BOTTOM to TOP and then restarts from BOTTOM. The waveform generator (WG) output is set on the compare match between the CNT and CCx registers and cleared at TOP.

Figure 13-14.Single-slope pulse width modulation.



The PER register defines the PWM resolution. The minimum resolution is 2 bits (PER=0x0003), and the maximum resolution is 16 bits (PER=MAX).

The following equation calculate the exact resolution for single-slope PWM (R_{PWM_SS}):

$$R_{PWM\_SS} = \frac{\log(PER + DIR)}{\log(2)}$$

The single-slope PWM frequency (f_{PWM_SS}) depends on the period setting (PER) and the peripheral clock frequency ($fclk_{PER}$), and can be calculated by the following equation:

$$f_{PWM\_SS} = \frac{fclk_{PER}}{N(PER + DIR)}$$

where N represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the peripheral clock frequency ($fclk_{PER}$) when CCA is set to zero (0x0000) and no prescaling is used. This also applies when using the hi-res extension, since this increases the resolution and not the frequency.

The pulse width (P_{PWM_SS}) depends on the compare channel settings (CCx), the direction (DIR) and the peripheral clock frequency ($fclk_{PER}$), and can be calculated by the following equation:

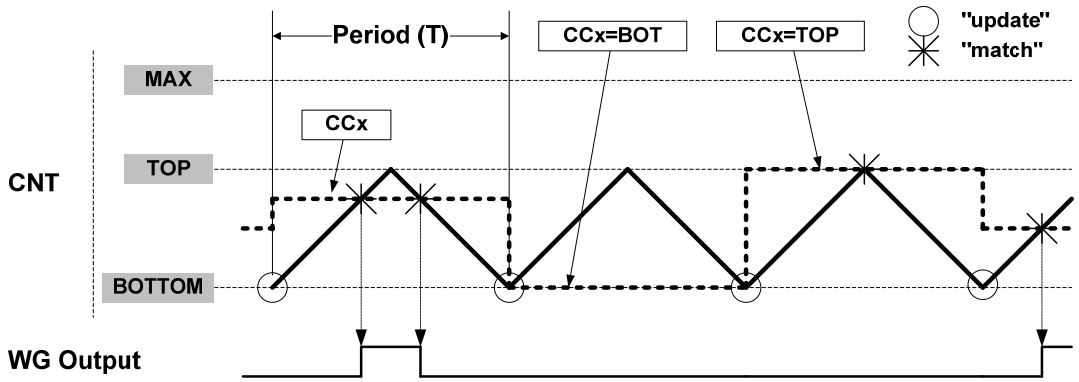
$$P_{PWM\_SS} = \frac{(N*CCx)}{fclk_{PER}}$$

where N represents the prescaler divider used. When counting up, the minimum pulse width is one peripheral clock.

13.8.4 Dual-slope PWM

For dual-slope PWM generation, the period (T) is controlled by PER, while CCx registers control the duty cycle of the WG output. Figure 13-15 shows how for dual-slope PWM the counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. The waveform generator output is set on BOTTOM, cleared on compare match when up-counting, and set on compare match when down-counting.

Figure 13-15.Dual-slope pulse width modulation.



Using dual-slope PWM results in a lower maximum operation frequency compared to the single-slope PWM operation.

The period register (PER) defines the PWM resolution. The minimum resolution is 2 bits (PER=0x0003), and the maximum resolution is 16 bits (PER=MAX).

The following equation calculate the exact resolution for dual-slope PWM (R_{PWM_DS}):

$$R_{PWM\_DS} = \frac{\log(PER+1)}{\log(2)}$$

The PWM frequency depends on the period setting (PER) and the peripheral clock frequency ($fclk_{PER}$), and can be calculated by the following equation:

$$f_{PWM\_DS} = \frac{fclk_{PER}}{2N*PER}$$

N represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the peripheral clock frequency ($fclk_{PER}$) when PER is set to one (0x0001) and no prescaling is used. This also applies when using the hi-res extension, since this increases the resolution and not the frequency.

The pulse width (P_{PWM_DS}) depends on the compare channel settings (CCx) and the peripheral clock frequency ($fclk_{PER}$), and can be calculated by the following equation:

$$P_{PWM\_DS} = \frac{2N*(PER-CCx)}{fclk_{PER}}$$

where N represents the prescaler divider used. In this mode, the pulse can be inhibited.

13.8.5 Output polarity

The polarity option is available in both single-slope and dual-slope PWM operation. In these modes, it is possible to invert the pulse edge alignment on start or end of PWM cycle. The Table 13-2 on page 172 shows the waveform output set/clear conditions, depending of timer/counter settings, direction and polarity setting.

Table 13-2. Waveform generation set/clear conditions.

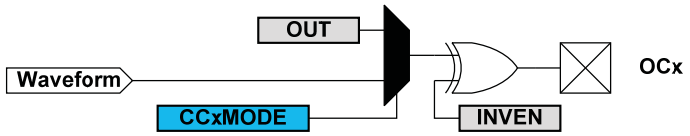
			WG output updates	
WG Mode	**DIR**	**Polarity**	**Set**	**Clear**
Single-Slope PWM	0	0	Timer/counter update	Timer/counter match
		1	Timer/counter match	Timer/counter update
	1	0	Timer/counter match	Timer/counter update
		1	Timer/counter update	Timer/counter match
Dual-Slope PMW	x	0	Timer/counter match when counting up	Timer/counter match when counting down
		1	Timer/counter match when counting down	Timer/counter match when counting up

13.8.6 Port override for waveform generation

To make the waveform generation available on the port pins, the corresponding port pin direction must be set as output. The timer/counter will override the port pin values when the CC channel is enabled (CCxMODE) and a waveform generation mode is selected.

Figure 13-16 shows the port override for a timer/counter. The timer/counter CC channel will override the port pin output value (OUT) on the corresponding port pin. Enabling inverted I/O on the port pin (INVEN) inverts the corresponding WG output.

Figure 13-16. Port override for timer/counter 4 and 5.



13.9 Interrupts and events

The timer/counter can generate both interrupts and events. The counter can generate an interrupt on overflow/underflow, and each CC channel has a separate interrupt that is used for compare or capture. In addition, an error interrupt can be generated if any of the CC channels is used for capture and a buffer overflow condition occurs on a capture channel.

Events will be generated for all conditions that can generate interrupts. For details on event generation and available events, refer to “Event System ” on page 79.

13.10 EDMA support

The interrupt flags can be used to trigger EDMA transactions. Table 13-3 on page 173 lists the transfer triggers available from the timer/counter and the EDMA action that will clear the transfer trigger. For more details on using EDMA, refer to “EDMA – Enhanced Direct Memory Access” on page 50.

Table 13-3. EDMA request sources.

Request	Acknowledge	Comment
OVFIF/UNFIF	EDMA controller writes to CNT EDMA controller writes to PER EDMA controller writes to PERBUF	
ERRIF	N/A	
CCxIF	EDMA controller access of CCx EDMA controller access of CCxBUF	Input capture operation Output compare operation

13.11 Timer/Counter commands

A set of commands can be given to the timer/counter by software to immediately change the state of the module. These commands give direct control of the UPDATE, RESTART, and RESET signals.

An update command has the same effect as when an update condition occurs. The update command is ignored if the lock update bit is set.

The software can force a restart of the current waveform period by issuing a restart command. In this case the counter, direction, and all compare outputs are set to zero.

A reset command will set all timer/counter registers to their initial values. A reset can be given only when the timer/counter is not running (OFF).

13.12 Register description – Standard configuration

13.12.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	SYNCHEN	EVSTART	UPSTOP	CLKSEL[3:0]			
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6 – SYNCHEN: Synchronization Enabled**
  When this bit is set, the event actions and software commands are synchronized with the internal timer/counter clock. When the bit is cleared, the event actions and software commands are synchronized with the peripheral clock (CLK_{PER}).
- **Bit 5 – EVSTART: Start on Next Event**
  Setting this bit will enable the timer/counter on the next event from event line selected by EVSEL bits. If the bit is cleared, the timer/counter can be enabled only by software, by clearing the STOP bit in CTRLGSET register.
- **Bit 4 – UPSTOP: Stop on Next Update**
  Setting this bit will disable the timer/counter on next update condition (overflow/underflow or retrigger). The bit has no effect if the timer/counter has been disabled by software.
- **Bit 3:0 – CLKSEL[3:0]: Clock Select**
  These bits select the clock source for the timer/counter according to Table 13-4.
  Setting CLKSEL to a no null value will start the timer, if EVSTART is written to 0 at the same time.
  DIV1 configuration must be set to ensure a correct output from the waveform generator when the Hi-Res extension is enabled.

Table 13-4. Clock select options.

CLKSEL[3:0]	Group configuration	Command action
0000	OFF	Prescaler: OFF
0001	DIV1	Prescaler: Clk
0010	DIV2	Prescaler: Clk/2
0011	DIV4	Prescaler: Clk/4
0100	DIV8	Prescaler: Clk/8
0101	DIV64	Prescaler: Clk/64
0110	DIV256	Prescaler: Clk/256
0111	DIV1024	Prescaler: Clk/1024
1nnn	EVCHn	Event channel n, n={0,…,7}

13.12.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	BYTEM[1:0]		CIRCEN[1:0]		–	WGMODE[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – BYTEM[1:0]: Byte Mode**
  These bits select the timer/counter configuration mode according to Table 13-5.

Table 13-5. Byte mode selection.

BYTEM[1:0]	Group configuration	Description
00	NORMAL	Timer/counter is set to normal mode (timer/counter type 4/5)
01	BYTEMODE	One 8-bit timer/counter with doubled CC channels. Upper byte of the counter (CNTH) will be set to zero after each counter clock cycle.
10	-	Reserved
11	-	Reserved

- **Bit 5:3 – CIRCEN[1:0]: Circular Buffer Enable**
  Setting these bits enable the circular buffer options according to Table 13-6.

Table 13-6. Circular buffer selection.

CIRCEN[1:0]	Group configuration	Description
00	DISABLE	Circular buffer disabled
01	PER	Circular buffer enabled on PER/PERBUF registers
10	CCA	Circular buffer enabled on CCA/CCABUF registers
11	BOTH	Circular buffer enabled on both PER/PERBUF and CCA/CCABUF registers

- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – WGMODE[2:0]: Waveform Generation Mode**
  These bits select the waveform generation mode, and control the counting sequence of the counter, TOP value, UPDATE condition, interrupt/event condition, and type of waveform that is generated according to Table 13-7. The result from the waveform generator can be directed to the port pins if the corresponding CCxMODE bits have been set to enable this. The port pin direction must be set as output.

Table 13-7. Waveform generation mode.

WGMODE[2:0]	Group configuration	Mode of operation	Top	Update	OVFIF/Event
000	NORMAL	Normal	PER	TOP/BOTTOM(1)	TOP/BOTTOM(1)
001	FRQ	Frequency	CCA	TOP/BOTTOM(1)	TOP/BOTTOM(1)
010	–	Reserved	N/A	N/A	N/A
011	SINGLESLOPE	Single-slope PWM	PER	TOP/BOTTOM(1)	TOP/BOTTOM(1)

WGMODE[2:0]	Group configuration	Mode of operation	Top	Update	OVFIF/Event
100	–	Reserved	N/A	N/A	N/A
101	DSTOP	Dual-slope PWM	PER	BOTTOM	TOP
110	DSBOTH	Dual-slope PWM	PER	BOTTOM	TOP and BOTTOM
111	DSBOTTOM	Dual-slope PWM	PER	BOTTOM	BOTTOM

Note: 1. Depends on DIR settings.

13.12.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	POLD	POLC	POLB	POLA	CMPD	CMPC	CMPB	CMPA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – POLx: Output Polarity x**
  Setting these bits enable the output polarity. For more details, refer to “Output polarity” on page 171.
- **Bit 3:0 – CMPx: Compare Output Value x**
  These bits allow direct access to the waveform generator's output compare value when the timer/counter is set in the OFF state. This is used to set or clear the WG output value when the timer/counter is not running.

13.12.4 CTRLD – Control register D

Bit	7	6	5	4	3	2	1	0
+0x03	EVACT[2:0]			EVDLY	EVSEL[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – EVACT[2:0]: Event Action**
  These bits define the event action the timer will perform on an event according to Table 13-8. The EVSEL setting will decide which event source or sources have control in this case.

Table 13-8. Event action selection.

EVACT[2:0]	Group configuration	Command action
000	OFF	None
001	FMODE1(1)	Fault Mode 1 capture
010	FMODE2(1)	Fault mode 2 capture
011	UPDOWN	Externally controlled up/down count
100	QDEC	Quadrature decode
101	RESTART	Restart waveform period
110	PWF	Pulse width and frequency capture
111	–	Reserved

Note: 1. This mode is available only for timer/counter with FAULT extension. For timer/counter without FAULT extension, an input capture will be done in this mode.

- **Bit 4 – EVDLY: Timer Delay Event**
  When this bit is set, the selected event source is delayed by one peripheral clock cycle. This is intended for 32-bit input capture operation. Adding the event delay is necessary to compensate for the carry propagation delay when cascading two counters via the event system.
- **Bit 3:0 – EVSEL[3:0]:Timer Event Source Select**
  These bits select the event channel source for the timer/counter. For the selected event channel to have any effect, the event action bits (EVACT) must be set according to Table 13-9.

Table 13-9. Event source selection.

EVSEL[3:0]	Group configuration	Command action
0000	OFF	None
0001	-	Reserved
0010	-	Reserved
0011	-	Reserved
0100	-	Reserved
0101	-	Reserved
0110	-	Reserved
0111	-	Reserved
1nnn	CHn	Event channel n, n={0,…,7}

By default, the selected event channel n will be the event channel source for CC channel A, and event channel (n+1)%8, (n+2)%8, and (n+3)%8 will be the event channel source for CC channel B, C, and D.

Table 13-10 shows the event channel source for each CC channel, depending of EVACT settings.

Table 13-10. Event channel source selection.

Event channel selection					
EVACT[2:0]	OCD	CCC	CCB	CCA	Restart condition
000	(n+3)%8	(n+2)%8	(n+1)%8	n	Software
001	(n+3)%8	(n+2)%8	FAULTA/B	FAULTA/B	Software or fault
010	FAULTB	FAULTA	FAULTB	FAULTA	Software or fault
011	(n+3)%8	(n+2)%8	(n+1)%8	n	Software
100	(n+3)%8	(n+2)%8	(n+1)%8	n	Quadrature decoder
101	(n+3)%8	(n+2)%8	(n+1)%8	n	Event channel n
110	(n+3)%8	(n+2)%8	n	n	Event channel n
111	N/A	N/A	N/A	N/A	Reserved

13.12.5 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x04	CCDMODE[1:0]		CCCMODE[1:0]		CCBMODE[1:0]		CCAMODE[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCxMODE[2:0] - Compare or Capture x Mode**
  These bits enable the compare and capture operation on corresponding CCx channel, according to Table 13-11.

Table 13-11. CC mode selection.

CCMODE[1:0]	Group configuration	Command action
00	DISABLE	Compare or capture disabled
01	COMP	Output compare enabled
10	CAPT	Input capture enabled
11	BOTHCC(1)	Both compare and capture enabled

Note: 1. This mode should be used only if the Fault Unit extension is set in conditional capture fault mode. For more details, refer to "Fault Extension" on page 209 description.

13.12.6 INTCTRLA – Interrupt Control register A

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	TRGINTLVL[1:0]		ERRINTLVL[1:0]		OVFINTLVL[1:0]	
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:4 – TRGINTLVL[1:0]:Timer Trigger Restart Interrupt Level**
  These bits enable the interrupt for the timer trigger restart and select the interrupt level as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132. The enabled interrupt will trigger for the conditions when TRGIF flag is set.
- **Bit 3:2 – ERRINTLVL[1:0]:Timer Error Interrupt Level**
  These bits enable the timer error interrupt and select the interrupt level as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132.
- **Bit 1:0 – OVFINTLVL[1:0]:Timer Overflow/Underflow Interrupt Level**
  These bits enable interrupt for the timer overflow/underflow and select the interrupt level as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132. The enabled interrupt will trigger for the conditions when OVFIF flag is set.

13.12.7 INTCTRLB – Interrupt Control register B

Bit	7	6	5	4	3	2	1	0
+0x07	CCDINTLVL[1:0]		CCCINTLVL[1:0]		CCBINTLVL[1:0]		CCAINTLVL[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCxINTLVL[2:0] - Compare or Capture x Interrupt Level**
  These bits enable the timer compare or capture interrupt for channel x and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.

13.12.8 CTRLGCLR/CTRLGSET – Control register G Clear/Set

This register is mapped into two I/O memory locations, one for clearing (CTRLxCLR) and one for setting the register bits (CTRLxSET) when written. Both memory locations will give the same result when read.

The individual status bit can be set by writing a one to its bit location in CTRLxSET, and cleared by writing a one to its bit location in CTRLxCLR. This allows each bit to be set or cleared without use of a read-modify-write operation on a single register.

Bit	7	6	5	4	3	2	1	0
	–	–	STOP	–	CMD[1:0]		LUPD	DIR
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	1	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5 – STOP: Timer/Counter Stop**
  When this bit is set, the timer/counter is automatically stopped and all events and waveform outputs will be disabled. When this bit is cleared, the timer/counter is automatically restarted if CLKSEL setting is not in OFF state.
- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 3:2 – CMD[1:0]: Command**
  These bits can be used for software control of update, restart, and reset of the timer/counter. The command bits are always read as zero.

Table 13-12. Command selection

CMD[1:0]	Group configuration	Command action
00	NONE	None
01	UPDATE	Force update
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is not in OFF state)

- **Bit 1 – LUPD: Lock Update**
  When this bit is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffers, including DTI buffers, are valid before an update is performed.
  This bit has no effect when input capture operation is enabled.

- **Bit 0 – DIR: Counter Direction**
  When zero, this bit indicates that the counter is counting up (incrementing). A one indicates that the counter is in the down-counting (decrementing) state.
  Normally this bit is controlled in hardware by the waveform generation mode or by event actions, but this bit can also be changed from software.

13.12.9 CTRLHCLR/CTRLHSET – Control register H Clear/Set

Refer to "CTRLGCLR/CTRLGSET – Control register G Clear/Set" on page 179 for information on how to access this type of status register.

Bit	7	6	5	4	3	2	1	0
	–	–	–	CCDBV	CCCBV	CCBBV	CCABV	PERBV
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4:1 – CCxBV: Compare or Capture x Buffer Valid**
  These bits are set when a new value is written to the corresponding CCxBUF register. These bits are automatically cleared on an UPDATE condition.
  Note that when input capture operation is used, this bit is set on a capture event and cleared if the corresponding CCxIF is cleared.
- **Bit 0 – PERBV: Period Buffer Valid**
  This bit is set when a new value is written to the PERBUF register. This bit is automatically cleared on an UPDATE condition.

13.12.10 INTFLAGS – Interrupt Flags register

Bit	7	6	5	4	3	2	1	0
+0x0C	CCDIF	CCCIF	CCBIF	CCAIF	–	TRGIF	ERRIF	OVFIF
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – CCxIF: Compare or Capture Channel x Interrupt Flag**
  The compare or capture interrupt flag (CCxIF) is set on a compare match or on an input capture event on the corresponding CC channel.
  For all modes of operation except for capture, the CCxIF will be set when a compare match occurs between the count register (CNT) and the corresponding compare register (CCx). The CCxIF is automatically cleared when the corresponding interrupt vector is executed.
  For input capture operation, the CCxIF will be set if the corresponding compare buffer contains valid data (i.e., when CCxBV is set). The flag will be cleared when the CCx register is read. Executing the interrupt vector in this mode of operation will not clear the flag.
  The flag can also be cleared by writing a one to its bit location.
  The CCxIF can be used for requesting an EDMA transfer. An EDMA read or write access of the corresponding CCx or CCxBUF will then clear the CCxIF and release the request.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2 – TRGIF: Trigger Restart Interrupt Flag**
  This flag is set when hardware restart condition is detected. Optionally an interrupt can be generated. The flag is cleared by writing a one to its bit location.

- **Bit 1 – ERRIF: Error Interrupt Flag**
  This flag is set on multiple occasions, depending on the mode of operation.
  In the FAULT1 or FAULT2 mode of operation, ERRIF is set on a fault condition from the fault extension unit that requests for software action to resume. For timer/counters which do not have the FAULT extension available, this flag is never set in FAULT1 or FAULT2 mode of operation.
  For capture operation, ERRIF is set if a buffer overflow occurs on any of the CC channels.
  For event controlled QDEC operation, ERRIF is set when an incorrect index signal is given.
  This flag is automatically cleared when the corresponding interrupt vector is executed. The flag can also be cleared by writing a one to this location.
- **Bit 0 – OVFIF: Overflow/Underflow Interrupt Flag**
  This flag is set either on a TOP (overflow) or BOTTOM (underflow) condition, depending on the WGMODE setting. The flag is cleared by writing a one to its bit location.
  OVFIF can also be used for requesting an EDMA transfer. An EDMA write access of CNT, PER, or PERBUF will then clear the OVFIF bit.

13.12.11 TEMP – Temporary register for 16-bit Access

Bit	7	6	5	4	3	2	1	0
+0x0F	TEMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – TEMP[7:0]: Temporary Bits**
  The TEMP register is used for single-cycle, 16-bit access to the 16-bit timer/counter registers by the CPU. The EDMA controller has a separate temporary storage register. There is one common TEMP register for all the 16-bit Timer/counter registers.
  For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

13.12.12CNTL – Counter register Low

The CNTH and CNTL register pair represents the 16-bit value, CNT. CNT contains the 16-bit counter value in the timer/counter. CPU and EDMA write access has priority over count, clear, or reload of the counter.

For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x20	CNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CNT[7:0]: Counter Low Byte**
  These bits hold the LSB of the 16-bit counter register.

13.12.13 CNTH – Counter register High

Bit	7	6	5	4	3	2	1	0
+0x21	CNT[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CNT[15:8]: Counter High Byte**
  These bits hold the MSB of the 16-bit counter register.

13.12.14 PERL – Period register Low

The PERH and PERL register pair represents the 16-bit value, PER. PER contains the 16-bit TOP value in the timer/counter.

For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x26	PER[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PER[7:0]: Period Low Byte**
  These bits hold the MSB of the 16-bit period register.

13.12.15 PERH – Period register High

Bit	7	6	5	4	3	2	1	0
+0x27	PER[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PER[15:8]: Period High Byte**
  These bits hold the MSB of the 16-bit period register.

13.12.16 CCxL – Compare or Capture x register Low

The CCxH and CCxL register pair represents the 16-bit value, CCx. These 16-bit register pairs have two functions, depending of the mode of operation.

For capture operation, these registers constitute the second buffer level and access point for the CPU and EDMA.

For compare operation, these registers are continuously compared to the counter value. Normally, the outputs form the comparators are then used for generating waveforms. CCx registers are updated with the buffer value from their corresponding CCxBUF register when an UPDATE condition occurs.

For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
	CCx[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCx[7:0]: Compare or Capture x Low Byte**
  These bits hold the MSB of the 16-bit compare or capture register.

13.12.17 CCxH – Compare or Capture x register High

Bit	7	6	5	4	3	2	1	0
	CCx[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCx[15:8]: Compare or Capture x High Byte**
  These bits hold the MSB of the 16-bit compare or capture register.

13.12.18 PERBUFL – Period Buffer register Low

The PERBUFH and PERBUFL register pair represents the 16-bit value, PERBUF. This 16-bit register serves as the buffer for the period register (PER). Accessing this register using the CPU or EDMA will affect the PERBUFV flag.

For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x37	PERBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PERBUF[7:0]: Period Buffer Low Byte**
  These bits hold the LSB of the 16-bit period buffer register.

13.12.19 PERBUFH – Period Buffer High

Bit	7	6	5	4	3	2	1	0
+0x38	PERBUF[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PERBUF[15:8]: Period Buffer High Byte**
  These bits hold the MSB of the 16-bit period buffer register.

13.12.20 CCxBUFL – Compare or Capture x Buffer register Low

The CCxBUFH and CCxBUFL register pair represents the 16-bit value, CCxBUF. These 16-bit registers serve as the buffer for the associated compare or capture registers (CCx). Accessing any of these registers using the CPU or EDMA will affect the corresponding CCxBV status bit.

For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
	CCxBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCxBUF[7:0]: Compare or Capture x Buffer Low Byte**
  These bits hold the LSB of the 16-bit compare or capture buffer register.

13.12.21 CCxBUFH – Compare or Capture x Buffer register H

Bit	7	6	5	4	3	2	1	0
	CCxBUF[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CCxBUF[15:8]: Compare or Capture x Buffer High Byte**
  These bits hold the MSB of the 16-bit compare or capture buffer register.

13.13 Register description – Byte mode configuration

13.13.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	SYNCHEN	EVSTART	UPSTOP	CLKSEL[3:0]			
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6 – SYNCHEN: Synchronization Enabled**
  When this bit is set, the event actions and software commands are synchronized with the internal timer/counter clock. When the bit is cleared, the event actions and software commands are synchronized with the peripheral clock (CLK_{PER}).
- **Bit 5 – EVSTART: Start on Next Event**
  Setting this bit will enable the timer/counter on the next event from event line selected by EVSEL bits. If the bit is cleared, the timer/counter can be enabled only by software, by clearing the LSTOP bit in CTRLGSET register.
- **Bit 4 – UPSTOP: Stop on next update**
  Setting this bit will disable the timer/counter on next update condition (overflow/underflow or retrigger). The bit has no effect is the timer/counter has been disabled by software.
- **Bit 3:0 – CLKSEL[3:0]: Clock Select**
  These bits select the clock source for the timer/counter according to Table 13-13.
  DIV1 configuration must be set to ensure a correct output from the waveform generator when the hires extension is enabled.

Table 13-13. Clock select options.

CLKSEL[3:0]	Group configuration	Command action
0000	OFF	Prescaler: OFF
0001	DIV1	Prescaler: Clk
0010	DIV2	Prescaler: Clk/2
0011	DIV4	Prescaler: Clk/4
0100	DIV8	Prescaler: Clk/8
0101	DIV64	Prescaler: Clk/64
0110	DIV256	Prescaler: Clk/256
0111	DIV1024	Prescaler: Clk/1024
1nnn	EVCHn	Event channel n, n={0,…,7}

13.13.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	BYTEM[1:0]		CIRCEN[1:0]		–	WGMODE[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – BYTEM[1:0]: Byte Mode**
  These bits select the timer/counter operation mode according to Table 13-14.

Table 13-14. Timer/Counter byte mode selection.

BYTEM[1:0]	Group configuration	Description
00	NORMAL	Timer/counter is set to normal mode (timer/counter type 4/5)
01	BYTEMODE	One 8-bit timer/counter with doubled CC channels. Upper byte of the counter (CNTH) will be set to zero after each counter clock cycle.
10	–	Reserved
11	–	Reserved

- **Bit 5:4 – CIRCEN[1:0]: Circular Buffer Enable**
  Setting these bits enable the circular buffer option according to Table 13-15.

Table 13-15. Circular buffer selection.

CIRCEN[1:0]	Group configuration	Description
00	DISABLE	Circular buffer disabled
01	PER	Circular buffer enabled on PER/PERBUF registers
10	CCA	Circular buffer enabled on LCCA/LCCABUF registers
11	BOTH	Circular buffer enabled on both PER/PERBUF and LCCA/LCCABUF registers

- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – WGMODE[2:0]: Waveform Generation Mode**
  These bits select the waveform generation mode, and control the counting sequence of the counter, TOP value, UPDATE condition, interrupt/event condition, and type of waveform that is generated according to Table 13-16. The result from the waveform generator can be directed to the port pins if the corresponding CCxMODE bits have been set to enable this. The port pin direction must be set as output.

Table 13-16. Timer waveform generation mode.

WGMODE[2:0]	Group configuration	Mode of operation	Top	Update	OVFIF/Event
000	NORMAL	Normal	PER	TOP/BOTTOM(1)	TOP/BOTTOM(1)
001	FRQ	Frequency	CCA	TOP/BOTTOM(1)	TOP/BOTTOM(1)
010	–	Reserved	N/A	N/A	N/A
011	SINGLESLOPE	Single-slope PWM	PER	TOP/BOTTOM(1)	TOP/BOTTOM(1)

WGMODE[2:0]	Group configuration	Mode of operation	Top	Update	OVFIF/Event
100	–	Reserved	N/A	N/A	N/A
101	DSTOP	Dual-slope PWM	PER	BOTTOM	TOP
110	DSBOTH	Dual-slope PWM	PER	BOTTOM	TOP and BOTTOM
111	DSBOTTOM	Dual-slope PWM	PER	BOTTOM	BOTTOM

Note: 1. Depends on DIR settings.

13.13.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	HCMPD	HCMPC	HCMPB	HCMPA	LCMPD	LCMPC	LCMPB	LCMPA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – HCMPx/LCMPx: High/Low Compare x Output Value**
  These bits allow direct access to the waveform generator's output compare value when the timer/counter is OFF. This is used to set or clear the WG output value when the timer/counter is not running.

13.13.4 CTRLD – Control register D

Bit	7	6	5	4	3	2	1	0
+0x03	EVACT[2:0]			EVDLY	EVSEL[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – EVACT[2:0]: Event Action**
  These bits define the event action the timer will perform on an event according to Table 13-17. The EVSEL setting will decide which event source or sources have control in this case.

Table 13-17. Timer event action selection.

EVACT[2:0]	Group configuration	Command action
000	OFF	None
001	FMODE1(1)	Fault Mode 1 capture
010	FMODE2(1)	Fault mode 2 capture
011	UPDOWN	Externally controlled up/down count
100	QDEC	Quadrature decode
101	RESTART	Restart waveform period
110	PWF	Pulse width and frequency capture
111	-	Reserved

Note: 1. This mode is available only for timer/counter with FAULT extension. For timer/counter without FAULT extension, an input capture will be done.

- **Bit 4 – EVDLY: Timer Delay Event**
  When this bit is set, the selected event source is delayed by one peripheral clock cycle. This is intended for 32-bit input capture operation. Adding the event delay is necessary to compensate for the carry propagation delay when cascading two counters via the event system.

- **Bit 3:0 – EVSEL[3:0]:Timer Event Source Select**
  These bits select the event channel source for the timer/counter. For the selected event channel to have any effect, the event action bits (EVACT) must be set according to Table 13-18.

Table 13-18. Timer event source selection.

EVSEL[3:0]	Group configuration	Command action
0000	OFF	None
0001	–	Reserved
0010	–	Reserved
0011	–	Reserved
0100	–	Reserved
0101	–	Reserved
0110	–	Reserved
0111	–	Reserved
1nnn	CHn	Event channel n, n={0,…,7}

By default, the selected event channel n will be the event channel source for CC channel A, and event channel (n+1)%8, (n+2)%8, and (n+3)%8 will be the event channel source for CC channel B, C, and D.

Table 13-19 and Table 13-20 show the event channel source for each CC channel, depending of EVACT settings:

Table 13-19. Event low channel source selection.

Event channel selection					
EVACT[2:0]	LCCD	LCCC	LCCB	LCCA	Restart condition
000	(n+3)%8	(n+2)%8	(n+1)%8	n	Software
001	(n+3)%8	(n+2)%8	FAULTB	FAULTA	Software or fault
010	FAULTB	FAULTA	FAULTB	FAULTA	Software or fault
011	(n+3)%8	(n+2)%8	(n+1)%8	n	Software
100	(n+3)%8	(n+2)%8	(n+1)%8	n	Quadrature decoder
101	(n+3)%8	(n+2)%8	(n+1)%8	n	Event channel n
110	(n+3)%8	(n+2)%8	n	n	Event channel n
111	N/A	N/A	N/A	N/A	Reserved

Table 13-20. Event high channel source selection.

Event channel selection					
EVACT[2:0]	**HCCD**	**HCCC**	**HCCB**	**HCCA**	**Restart condition**
000	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Software
001	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Software
010	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Software
011	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Software
100	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Quadrature decoder
101	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Event channel n
110	(n+7)%8	(n+6)%8	(n+5)%8	(n+4)%8	Event channel n
111	N/A	N/A	N/A	N/A	Reserved

13.13.5 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x04	**LCCDMODE[1:0]**		**LCCCMODE[1:0]**		**LCCBMODE[1:0]**		**LCCAMODE[1:0]**	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LCCxMODE[2:0] – Low Channel Compare or Capture x Mode**
  These bits enable the compare and capture operation on corresponding CCx low channel, according to Table 13-21.

13.13.6 CTRLF – Control register F

Bit	7	6	5	4	3	2	1	0
+0x05	**HCCDMODE[1:0]**		**HCCCMODE[1:0]**		**HCCBMODE[1:0]**		**HCCAMODE[1:0]**	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – HCCxMODE[2:0] – High Channel Compare or Capture x Mode**
  These bits enable the compare and capture operation on corresponding CCx high channel, according to Table 13-21.

Table 13-21. CC mode selection

CCMODE[1:0]	Group configuration	Command action
00	DISABLE	Compare or capture disabled
01	COMP	Output compare enabled
10	CAPT	Input capture enabled
11	BOTHCC(1)	Both compare and capture enabled

Note: 1. This mode should be used only if the Fault Unit extension is set in conditional capture fault mode. For more details, refer to"Fault Extension" on page 209 for description.

13.13.7 INTCTRLA – Interrupt Control register A

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	TRGINTLVL[1:0]		ERRINTLVL[1:0]		OVFINTLVL[1:0]	
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:4 – TRGINTLVL[1:0]:Timer Trigger Restart Interrupt Level**
  These bits enable the interrupt for the timer trigger restart and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger for the conditions when TRGIF flag is set.
- **Bit 3:2 – ERRINTLVL[1:0]:Timer Error Interrupt Level**
  These bits enable the timer error interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.
- **Bit 1:0 – OVFINTLVL[1:0]:Timer Overflow/Underflow Interrupt Level**
  These bits enable interrupt for the timer overflow/underflow and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger for the conditions when OVFIF flag is set.

13.13.8 INTCTRLB – Interrupt Control register B

Bit	7	6	5	4	3	2	1	0
+0x07	LCCDINTLVL[1:0]		LCCCINTLVL[1:0]		LCCBINTLVL[1:0]		LCCAINTLVL[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LCCxINTLVL[2:0] – Low-channel Compare or Capture x Interrupt Level**
  These bits enable the timer compare or capture interrupt for low channel x and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.

13.13.9 CTRLGCLR/CTRLGSET – Control register G Clear/Set

This register is mapped into two I/O memory locations, one for clearing (CTRLxCLR) and one for setting the register bits (CTRLxSET) when written. Both memory locations will give the same result when read.

The individual status bit can be set by writing a one to its bit location in CTRLxSET, and cleared by writing a one to its bit location in CTRLxCLR. This allows each bit to be set or cleared without use of a read-modify-write operation on a single register.

Bit	7	6	5	4	3	2	1	0
	–	–	STOP	–	CMD[1:0]		LUPD	DIR
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	1	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5 – STOP: Timer/Counter Stop**
  When this bit is set, the timer/counter is automatically stopped and all events and waveform outputs will be disabled. When this bit is cleared, the timer/counter is automatically restarted if CLKSEL setting is not OFF state.

- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 3:2 – CMD[1:0]: Command**
  These bits can be used for software control of update, restart, and reset of the low-byte timer/counter, according to Table 13-22 on page 190. The command bits are always read as zero.

Table 13-22. Command selections

CMD[1:0]	Group configuration	Command action
00	NONE	None
01	UPDATE	Force update
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is not in OFF state)

- **Bit 1 – LUPD: Lock Update**
  When this bit is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffers, including DTI buffers, are valid before an update is performed.
  This bit has no effect when input capture operation is enabled.
- **Bit 0 – DIR: Counter Direction**
  When zero, this bit indicates that the counter is counting up (incrementing). A one indicates that the counter is in the down-counting (decrementing) state.
  Normally this bit is controlled in hardware by the waveform generation mode or by event actions, but this bit can also be changed from software.

13.13.10 CTRLHCLR/CTRLHSET – Control register H Clear/Set

Refer to "CTRLGCLR/CTRLGSET – Control register G Clear/Set" on page 179 for information on how to access this type of status register.

Bit	7	6	5	4	3	2	1	0
	–	–	–	LCCDBV	LCCCBV	LCCBBV	LCCABV	LPERBV
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4:1 – LCCxBV: Low Channel Compare or Capture x Buffer Valid**
  These bits are set when a new value is written to the corresponding LCCxBUF register. These bits are automatically cleared on an UPDATE condition.
  Note that when input capture operation is used, this bit is set on a capture event and cleared if the corresponding LCCxIF is cleared.
  The feature is not present on high channels.
- **Bit 0 – LPERBV: Low Period Buffer Valid**
  This bit is set when a new value is written to the LPERBUF register. This bit is automatically cleared on an UPDATE condition.

13.13.11 INTFLAGS – Interrupt Flags register

Bit	7	6	5	4	3	2	1	0
+0x0C	LCCDIF	LCCCIF	LCCBIF	LCCAIF	–	TRGIF	ERRIF	OVFIF
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – LCCxIF: Low Channel Compare or Capture x Interrupt Flag**
  The compare or capture interrupt flag (LCCxIF) is set on a compare match or on an input capture event on the corresponding LCC channel.
  For all modes of operation except for capture, the LCCxIF will be set when a compare match occurs between the count register (LCNT) and the corresponding compare register (LCCx). The LCCxIF is automatically cleared when the corresponding interrupt vector is executed.
  For input capture operation, the LCCxIF will be set if the corresponding compare buffer contains valid data (i.e., when LCCxBV is set). The flag will be cleared when the LCCx register is read. Executing the interrupt vector in this mode of operation will not clear the flag.
  The flag can also be cleared by writing a one to its bit location.
  The LCCxIF can be used for requesting an EDMA transfer. An EDMA read or write access of the corresponding LCCx or LCCxBUF will then clear the LCCxIF and release the request.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2 – TRGIF: Timer/Counter Trigger Restart Interrupt Flag**
  This flag is set when hardware restart condition is detected. Optionally an interrupt can be generated. Since the trigger restart interrupt shares the interrupt address with the overflow/underflow interrupt, TRGIF will not be cleared when the interrupt vector is executed. The flag is cleared by writing a one to its bit location.
- **Bit 1 – ERRIF: Error Interrupt Flag**
  This flag is set on multiple occasions, depending on the mode of operation.
  In the FAULT1 or FAULT2 mode of operation, ERRIF is set on a fault detect condition from the FAULT unit extension that requests for software action to resume. For timer/counters which do not have the FAULT extension available, this flag is never set in FAULT1 or FAULT2 mode of operation.
  For capture operation, ERRIF is set if a buffer overflow occurs on any of the low CC channels.
  For event controlled QDEC operation, ERRIF is set when an incorrect index signal is given.
  This flag is automatically cleared when the corresponding interrupt vector is executed. The flag can also be cleared by writing a one to this location.
- **Bit 0 – OVFIF: Timer Overflow/Underflow Interrupt Flag**
  This flag is set either on a TOP (overflow) or BOTTOM (underflow) condition, depending on the WGMODE setting. Since the overflow/underflow interrupt shares the interrupt address with the trigger restart interrupt, OVFIF will not be cleared when the interrupt vector is executed. The flag is cleared by writing a one to its bit location.
  OVFIF can also be used for requesting an EDMA transfer. An EDMA write access of LCNT, LPER, or LPERBUF will then clear the OVFIF bit.

13.13.12 LCNT – Low Counter register

Bit	7	6	5	4	3	2	1	0
+0x20	LCNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LCNT[7:0]: Low Counter Byte**
  These bits hold the 8-bit counter register.

13.13.13 LPER – Low Period register

Bit	7	6	5	4	3	2	1	0
+0x26	LPER[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LPER[7:0]: Low Period Byte**
  These bits hold the 8-bit period register.

13.13.14 LCCx – Low Channel Compare or Capture x register

Bit	7	6	5	4	3	2	1	0
	LCCx[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LCCx[7:0]: Low Channel Compare or Capture x Byte**
  These bits hold the 8-bit low channels compare or capture register.

13.13.15 HCCx – High-Channel Compare or Capture x register

Bit	7	6	5	4	3	2	1	0
	HCCx[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – HCCx[7:0]: High Channel Compare or Capture x Byte**
  These bits hold the 8-bit high-channel compare or capture register.

13.13.16 LPERBUF – Low Period Buffer register

Bit	7	6	5	4	3	2	1	0
+0x37	LPERBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LPERBUF[7:0]: Low Period Buffer Byte**
  These bits hold the 8-bit period buffer register.

13.13.17 LCCxBUF – Low Channel Compare or Capture x Buffer register

Bit	7	6	5	4	3	2	1	0
	LCCxBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – LCCxBUF[7:0]: Low Channel Compare or Capture x Buffer Byte**
  These bits hold the 8-bit low-channel compare or capture buffer register.

13.13.18 HCCxBUF – High Channel Compare or Capture x Buffer register

Bit	7	6	5	4	3	2	1	0
	HCCxBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – HCCxBUF[7:0]: High Channel Compare or Capture x Buffer Byte**
  These bits hold the 8-bit high-channel compare or capture buffer register.

13.14 Register summary – Standard configuration

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	-	SYNCHEN	EVSTART	UPSTOP	CLKSEL[3:0]				174
+0x01	CTRLB	BYTEM[1:0]		CIRCEN[1:0]		-	WGMODE[2:0]			175
+0x02	CTRLC	POLD	POLC	POLB	POLA	CMPD	CMPC	CMPB	CMPA	176
+0x03	CTRLD	EVACT[2:0]			EVDLY	EVSEL[3:0]				176
+0x04	CTRLE	CCDMODE[1:0]		CCCMODE[1:0]		CCBMODE[1:0]		CCAMODE[1:0]		178
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	INTCTRLA	-	-	TRGINTLVL[1:0]		ERRINTLVL [1:0]		OVFINTLVL[1:0]		178
+0x07	INTCTRLB	CCDINTLVL[1:0]		CCCINTLVL[1:0]		CCBINTLVL[1:0]		CCAINTLVL[1:0]		179
+0x08	CTRLGCLR	-	-	STOP	-	CMD[1:0]		LUPD	DIR	179
+0x09	CTRLGSET	-	-	STOP	-	CMD[1:0]		LUPD	DIR	179
+0x0A	CTRLHCLR	-	-	-	CCDBV	CCCBV	CCBBV	CCABV	PERBV	180
+0x0B	CTRLHSET	-	-	-	CCDBV	CCCBV	CCBBV	CCABV	PERBV	180
+0x0C	INTFLAGS	CCDIF	CCCIF	CCBIF	CCAIF	-	TRGIF	ERRIF	OVFIF	180
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	TEMP	TEMP[7:0]								181
+0x10 - +0x1F	Reserved	-	-	-	-	-	-	-	-	
+0x20	CNTL	CNT[7:0]								181
+0x21	CNTH	CNT[15:8]								181
+0x22 - +0x25	Reserved	-	-	-	-	-	-	-	-	
+0x26	PERL	PER[7:0]								182
+0x27	PERH	PER[15:8]								182
+0x28	CCAL	CCA[7:0]								182
+0x29	CCAH	CCA[15:8]								182
+0x2A	CCBL	CCB[7:0]								182
+0x2B	CCBH	CCB[15:8]								182
+0x2C	CCCL	CCC[7:0]								182
+0x2D	CCCH	CCC[15:8]								182
+0x2E	CCDL	CCD[7:0]								182
+0x2F	CCDH	CCD[15:8]								182
+0x30 - +0x35	Reserved	-	-	-	-	-	-	-	-	
+0x36	PERBUFL	PERBUF[7:0]								183
+0x37	PERBUFH	PERBUF[15:8]								183
+0x38	CCABUFL	CCABUF[7:0]								183
+0x39	CCABUFH	CCABUF[15:8]								183
+0x3A	CCBBUFL	CCBBUF[7:0]								183
+0x3B	CCBBUFH	CCBBUF[15:8]								183
+0x3C	CCCBUFL	CCCBUF[7:0]								183
+0x3D	CCCBUFH	CCCBUF[15:8]								183
+0x3E	CCDBUFL	CCDBUF[7:0]								183
+0x3F	CCDBUFH	CCDBUF[15:8]								183

13.15 Interrupt vector summary – Standard configuration

Table 13-23. Timer/counter interrupt vectors and their word offset address

Offset	Source	Interrupt description
0x00	OVF_vect	Timer/counter overflow/underflow interrupt vector offset
0x02	ERR_vect	Timer/counter error interrupt vector offset
0x04	CCA_vect	Timer/counter compare or capture channel A interrupt vector offset
0x06	CCB_vect	Timer/counter compare or capture channel B interrupt vector offset
0x08	CCC_vect(1)	Timer/counter compare or capture channel C interrupt vector offset
0x0A	CCD_vect(1)	Timer/counter compare or capture channel D interrupt vector offset

Note: 1. Available only on timer/counters with four compare or capture channels.

13.16 Register summary – Byte configuration

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	-	SYNCHEN	EVSTART	UPSTOP	CLKSEL[3:0]				184
+0x01	CTRLB	BYTEM[1:0]		CIRCEN[1:0]		-	WGMODE[2:0]			185
+0x02	CTRLC	HCMPD	HCMPC	HCMPB	HCMPA	LCMPD	LCMPC	LCMPB	LCMPA	186
+0x03	CTRLD	EVACT[2:0]			EVDLY	EVSEL[3:0]				186
+0x04	CTRLE	LCCDMODE[1:0]		LCCCMODE[1:0]		LCCBMODE[1:0]		LCCAMODE[1:0]		188
+0x05	CTRLF	HCCDMODE[1:0]		HCCCMODE[1:0]		HCCBMODE[1:0]		HCCAMODE[1:0]		188
+0x06	INTCTRLA	-	-	TRGINTLVL [1:0]		ERRINTLVL[1:0]		OVFINTLVL[1:0]		189
+0x07	INTCTRLB	LCCDINTLVL[1:0]		LCCCINTLVL[1:0]		LCCBINTLVL[1:0]		LCCAINTLVL[1:0]		189
+0x08	CTRLGCLR	-	-	STOP	-	CMD[1:0]		LUPD	DIR	189
+0x09	CTRLGSET	-	-	STOP	-	CMD[1:0]		LUPD	DIR	189
+0x0A	CTRLHCLR	-	-	-	LCCDBV	LCCCBV	LCCBBV	LCCABV	LPERBV	190
+0x0B	CTRLHSET	-	-	-	LCCDBV	LCCCBV	LCCBBV	LCCABV	LPERBV	190
+0x0C	INTFLAGS	LCCDIF	LCCCIF	LCCBIF	LCCAIF	-	TRGIF	ERRIF	OVFIF	191
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	
+0x10 - +0x1F	Reserved	-	-	-	-	-	-	-	-	
+0x20	LCNT	Low-byte Timer/Counter Count Register								191
+0x21	Reserved	-	-	-	-	-	-	-	-	
+0x22 - +0x25	Reserved	-	-	-	-	-	-	-	-	
+0x26	LPER	Low-byte Timer/Counter Period Register								192
+0x27	Reserved	-	-	-	-	-	-	-	-	
+0x28	LCCA	Low Channel Compare or Capture Register A								192
+0x29	HCCA	High Channel Compare or Capture Register A								192
+0x2A	LCCB	Low Channel Compare or Capture Register B								192
+0x2B	HCCB	High Channel Compare or Capture Register B								192
+0x2C	LCCC	Low Channel Compare or Capture Register C								192
+0x2D	HCCC	High Channel Compare or Capture Register C								192
+0x2E	LCCD	Low Channel Compare or Capture Register D								192
+0x2F	HCCD	High Channel Compare or Capture Register D								192
+0x30 - +0x35	Reserved	-	-	-	-	-	-	-	-	
+0x36	LPERBUF	Low- byte Timer/Counter Period Buffer Register								192
+0x37	Reserved	-	-	-	-	-	-	-	-	
+0x38	LCCABUF	Low Channel Compare or Capture Buffer Register A								192
+0x39	HCCABUF	High Channel Compare or Capture Buffer Register A								193
+0x3A	LCCBBUF	Low Channel Compare or Capture Buffer Register B								192
+0x3B	HCCBBUF	High Channel Compare or Capture Buffer Register B								193
+0x3C	LCCCBUF	Low Channel Compare or Capture Buffer Register C								192
+0x3D	HCCCBUF	High Channel Compare or Capture Buffer Register C								193
+0x3E	LCCDBUF	Low Channel Compare or Capture Buffer Register D								192
+0x3F	HCCDBUF	High Channel Compare or Capture Buffer Register D								193

13.17 Interrupt vector summary – Byte configuration

Table 13-24. Timer/counter interrupt vectors and their word offset address

Offset	Source	Interrupt description
0x00	OVF_vect	Timer/Counter overflow/underflow interrupt vector offset
0x02	ERR_vect	Timer/Counter error interrupt vector offset
0x04	CCA_vect	Timer/Counter compare or capture channel A interrupt vector offset
0x06	CCB_vect	Timer/Counter compare or capture channel B interrupt vector offset
0x08	CCC_vect(1)	Timer/Counter compare or capture channel C interrupt vector offset
0x0A	CCD_ vect(1)	Timer/Counter compare or capture channel D interrupt vector offset

Note: 1. Available only on timer/counters with four compare or capture channels.

14. WeX – Waveform Extension

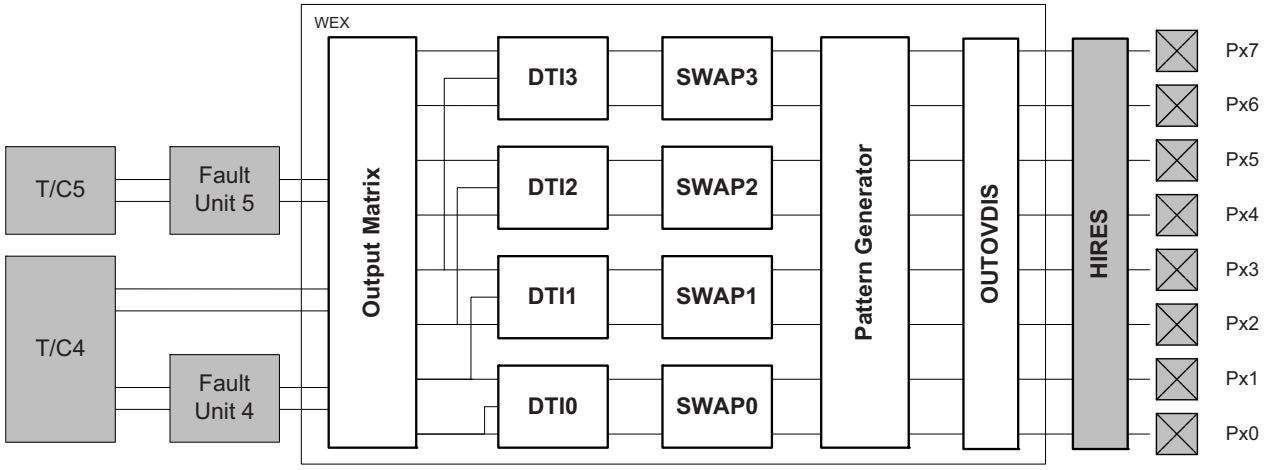
14.1 Features

- Module for more customized and advanced PWM and waveform output
  - Optimized for various types of motor, ballast and power stage control
- Output matrix for timer/counter compare channel distribution:
  - Configurable distribution of compare channel outputs across port pins
  - Redistribution of dead-time insertion resources between TC4 and TC5.
- Four dead-time insertion (DTI) units, each with:
  - Complementary high and low channel with non overlapping outputs
  - Separate dead-time setting for high and low side
  - 8-bit resolution
- Four swap (SWAP) units:
  - Separate port pair or low/high side drivers swap
  - Double buffered swap feature
- Pattern generation creating synchronized bit pattern across the port pins
  - Double buffered pattern generation

14.2 Overview

The waveform extension (WEX) provides extra functions to the timer/counter in waveform generation (WG) modes. It is primarily intended for use in different types of motor control, ballast, LED, H-bridge, power converter and other types of power control applications. The WEX consists of five independent and successive units, as shown in Figure 14-1.

Figure 14-1. Waveform extention and closely related peripherals (grey).



The output matrix (OTMX) can distribute and route out the waveform outputs from timer/counter 4 and 5 across the port pins in different configurations, each optimized for different application types.

The dead time insertion (DTI) unit splits the four lower OTMX outputs into a two non-overlapping signals, the non-inverted low side (LS) and inverted high side (HS) of the waveform output with optional dead-time insertion between LS and HS switching.

The swap (SWAP) unit can swap the LS and HS pin position. This can be used for fast decay motor control.

The pattern generation unit generates synchronized output waveform with constant logic level. This can be used for easy stepper motor and full bridge control.

The output override disable unit can disable the waveform output on selectable port pins to optimize the pins usage. This is to free the pins for other functional use, when the application does not need the waveform output spread across all the port pins as they can be selected by the OTMX configurations.

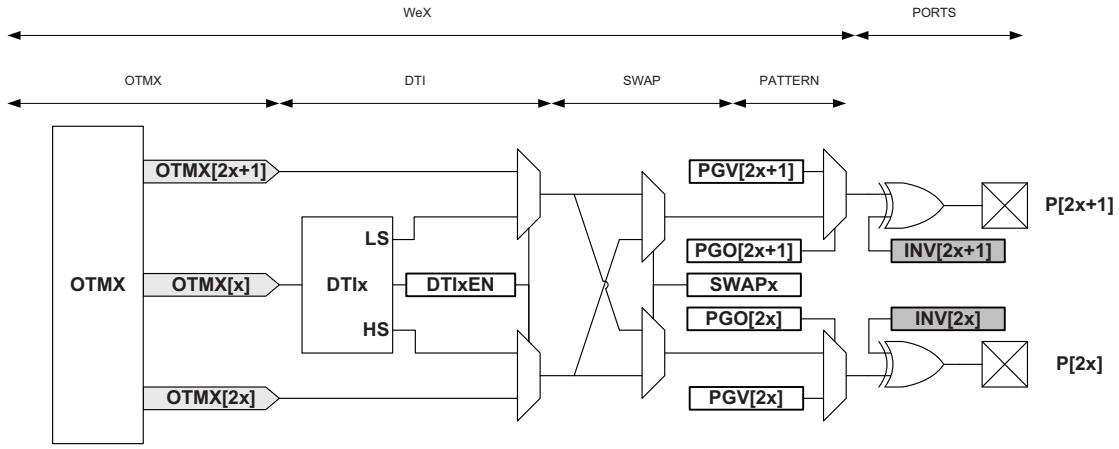
Figure 14-2 shows a schematic diagram of action of each WEX unit on a port pin pair. The WEX DTI and SWAP units can be seen as a four port pair slices:

- Slice 0, DTI0/ SWAP0 acting on port pins (Px[0],Px[1])
- Slice 1, DTI1/ SWAP1 acting on port pins (Px[2],Px[3])

And more generally:

- Slice n, DTIn/ SWAPn acting on port pins (Px[2n],Px[2n+1])

Figure 14-2. Waveform extention stage details.



14.3 Port override

The port override logic is common for all the timer/counter extensions. By default, when the dead-time enable (DTIENx) bit is set, the timer/counter extension takes control over the pin pair for the corresponding channel. The default behavior can be changed in one of the following conditions:

- PORTCTRL bit in "CTRLA – Control register A" on page 217 is set. For details on fault port control mode, refer to "Fault Extension" on page 209.
- The output is disabled by setting the corresponding bit in the output override disable register. When set, the corresponding I/O pin can be used by any other alternative pin function. For details, refer to "OUTOVDIS – Output Override Disable register" on page 205.

14.4 Output matrix

The output matrix (OTMX) unit distributes and routes waveform output across the port pins, according to the selectable configurations, as shown in Table 14-1 on page 200.

Table 14-1. Timer/counter 4 and 5 compare channel pin routing configuration.

OTMX[2:0]	PIN7	PIN 6	PIN 5	PIN 4	PIN 3	PIN 2	PIN 1	PIN 0
000			TC5CCB	TC5CCA	TC4CCD	TC4CCC	TC4CCB	TC4CCA
001	TC5CCB	TC5CCA	TC5CCB	TC5CCA	TC4CCD	TC4CCC	TC4CCB	TC4CCA
010	TC5CCB	TC5CCA	TC4CCB	TC4CCA	TC5CCB	TC5CCA	TC4CCB	TC4CCA
011	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA
100	TC4CCB	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA
101	–	–	–	–	–	–	–	–
110	–	–	–	–	–	–	–	–
111	–	–	–	–	–	–	–	–

1. Configuration 000 is default configuration. The pin location is the default one, and corresponds to the default timer/counter configuration.
2. Configuration 001 distributes the waveform outputs from timer/counter 5 compare channel A and B (TC5 CCA and TC5 CCB) on four pin locations. This provides for example, the enable control of the four transistors of a full bridge with only the use of two compare channels. Using pattern generation, some of these four outputs can be overwritten by a constant level, enabling flexible drive of a full bridge in all quadrant configurations.
3. Configuration 010 distributes the waveform outputs from compare channels A and B (CCA and CCB) from both timer/counter 4 and 5 on two other pin locations.
4. Configuration 011 distributes the waveform outputs from timer/counter 4 compare channel A (TC4CCA) to all port pins. Enabling pattern generation in this mode will control a stepper motor.
5. Configuration 100 distributes the waveform output from timer/counter 5 compare channel A (TC5 CCA) to pin 7 and the waveform output from timer/counter 4 compare channel A (TC4 CCA) to all other pins (Px0 to Px6). This, together with pattern generation and the fault extension, enable control of one to seven LED strings.

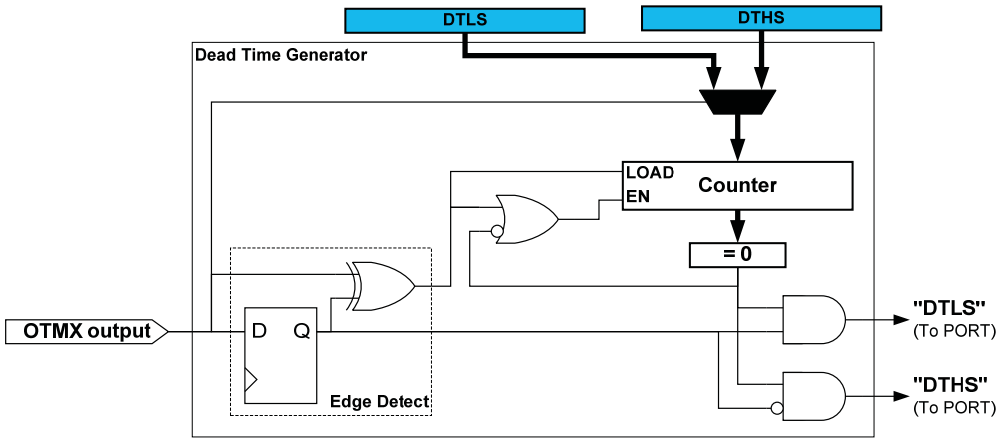
14.5 Dead-time generator

The dead-time insertion (DTI) unit generates OFF time on which the non-inverted low side (LS) and inverted high side (HS) of the WG outputs are both low. This OFF time is called dead time. Dead-time insertion ensures that the LS and HS never switch simultaneously.

The DTI stage consists of four equal dead-time insertion generators, one for each timer/counter 4 compare channels. They can be also redistributed to timer/counter 5 channels through output matrix (configuration 010).

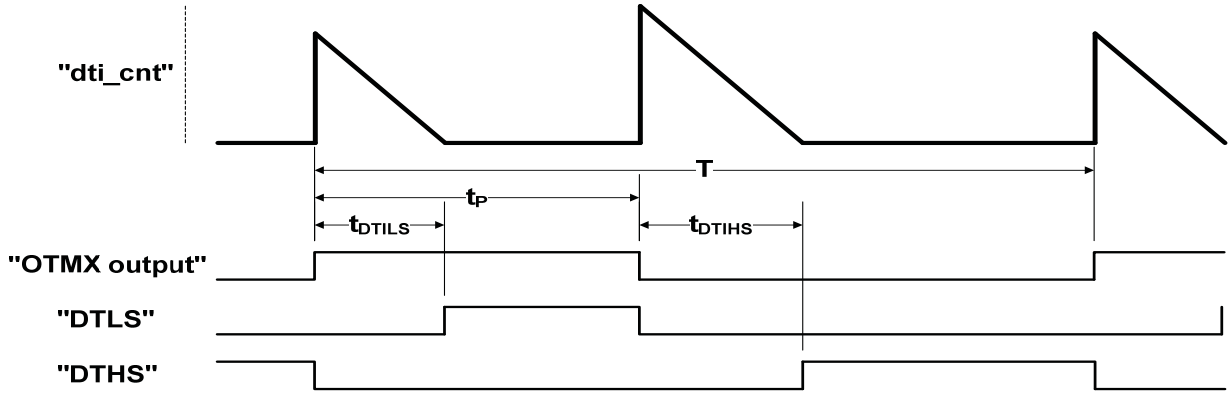
Figure 14-3 on page 201 shows the block diagram of one DTI generator. The four channels have a common register that controls independently the high side and low side dead times.

Figure 14-3. Dead-time generator block diagram.



As shown in Figure 14-4, the 8-bit dead-time counter is decremented by one at each peripheral clock cycle, until it reaches zero. A nonzero counter value will force both the low side and high side outputs into their OFF state. When the output matrix (OTMX) output changes, the dead-time counter is reloaded according to the edge of the input. When the output changes from low to high (positive edge) it initiates counter reload of the DTLS register, and when the output changes from high to low (negative edge) a reload of the DTHS register.

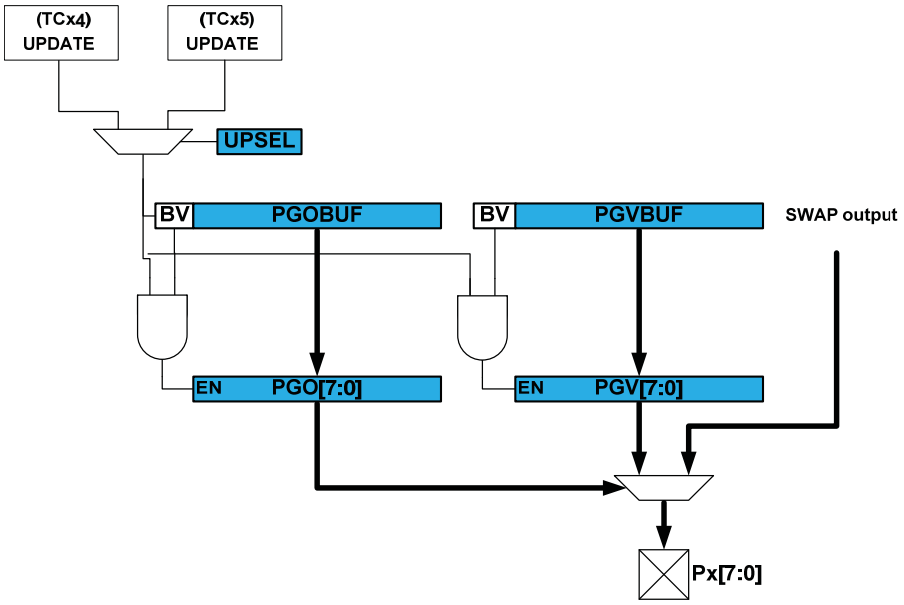
Figure 14-4. Dead-time generator timing diagram.



14.6 Pattern generator

The pattern generator unit produces a synchronized bit pattern across the port pins it is connected to. The pattern generation features are primarily intended for handling the commutation sequence in brushless DC motor (BLDC), stepper motor and full bridge control. A block diagram of the pattern generator is shown in Figure 14-5 on page 202.

Figure 14-5. Pattern generator block diagram.



As with other double buffered timer/counter registers, the register update is synchronized to the UPDATE condition, set by the timer/counter waveform generation mode. If the synchronization provided is not required by the application, the application code can simply access the PGO, PGV or PORTx registers directly.

In addition to port override condition, the timer/counter channel CCxMode associated to output port must be set to COMP or BOTHCC to make corresponding pattern generator be visible on the port.

14.7 Change protection

To avoid unintentional configuration changes, five control registers in the WEX can be protected by writing the corresponding lock bit in the waveform extension lock register. For more details, refer to “I/O memory protection” on page 25 and “WEXLOCK – Waveform Extension Lock register” on page 44.

When the lock bit is set, the CTRL, DTBOTH, DTLS, DTHS and OUTOVDIS registers cannot be changed.

14.8 Register description

14.8.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	UPSEL	OTMX[2:0]			DTI3EN	DTI2EN	DTI1EN	DTI0EN
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – UPSEL: Update Source Selection**
  By default the timer/counter 4 update condition is used by the swap and pattern generation units to update their register content. Setting this bit, makes the timer/counter 5 update condition as source of register update.
- **Bit 6:4 – OTMX[2:0]: Output Matrix**
  These bits define the matrix routing of the timer/counters waveform generation outputs to the port pins, according to Table 14-1 on page 200.
- **Bit 3:0 – DTIxEN: Dead-Time Insertion Generator x Enable**
  Setting any of these bits enables the dead-time insertion generator for the corresponding output matrix. This will override the output matrix [2x] and [2x+1], with the low side and high side waveform respectively. The bits are read zero if the fault blanking is enabled. For details on fault blanking, refer to "Fault blanking" on page 211.

14.8.2 DTBOTH – Dead-Time Concurrent Write to Both Sides register

Bit	7	6	5	4	3	2	1	0
+0x01	DTBOTH[7:0]							
Read/Write	W	W	W	W	W	W	W	W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DTBOTH[7:0]: Dead-Time Both Side Bits**
  Writing to this register will update the DTHS and DTLS registers at the same time (i.e., at the same I/O write access). Reading it, give 0x00 Value.

14.8.3 DTLS – Dead-Time Low Side register

Bit	7	6	5	4	3	2	1	0
+0x02	DTLS[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DTLS[7:0]: Dead-time Low Side Bits**
  This register holds the number of peripheral clock cycles for the dead-time low side.

14.8.4 DTHS – Dead-Time High Side register

Bit	7	6	5	4	3	2	1	0
+0x03	DTHS[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DTHS[7:0]: Dead-Time High Side Bits**
  This register holds the number of peripheral clock cycles for the dead-time high side.

14.8.5 STATUSCLR/STATUSSET – Status Clear/Set register

Bit	7	6	5	4	3	2	1	0
+0x04/0x05	–	–	–	–	–	SWAPBUFV	PGVBUFV	PGOBUFV
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – SWAPBUFV: SWAP Buffer Valid**
  If this bit is set, the swap buffer is written and contains valid data that will be copied into the SWAP register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter lock update (LUPD) flag also affects the update for the swap registers.
- **Bit 1 – PGVBUFV: Pattern Generator Value Buffer Valid**
  If this bit is set, the pattern generation value (PGV) buffer is written and contains valid data that will be copied into the PGV register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter lock update (LUPD) flag also affects the update of the PGV buffer.
- **Bit 0 – PGOBUFV: Pattern Generator Overwrite Buffer Valid**
  If this bit is set, the pattern generation overwrite (PGO) buffer is written and contains valid data that will be copied into the PGO register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter lock update (LUPD) flag also affects the update for the PGO buffers.

14.8.6 SWAP – Swap register

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	–	SWAP3	SWAP2	SWAP1	SWAP0
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:0 – SWAPx: Swap DTI Output Pair**
  Setting these bits enables output swap of DTI outputs [2x] and [2x+1]. Note the DTIxEN settings will not affect the swap operation.

14.8.7 PGO – Pattern Generation Override register

Bit	7	6	5	4	3	2	1	0
+0x07	PGO[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PGO[7:0]: Pattern Generation Override**
  This register holds the enables of pattern generation for each output. A bit position at one, overrides the corresponding SWAP output with the respective PGV bit value.

14.8.8 PGV – Pattern Generation Value register

Bit	7	6	5	4	3	2	1	0
+0x08	PGV[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PGV[7:0]: Pattern Generation Value**
  This register holds the values of pattern for each output.

14.8.9 SWAPBUF – Swap Buffer register

Bit	7	6	5	4	3	2	1	0
+0x0A	–	–	–	–	SWAP3BUF	SWAP2BUF	SWAP1BUF	SWAP0BUF
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:0 – SWAPxBUF: Swap DTI Output Pair**
  These register bits are the buffer for the SWAP register bits. If double buffering is used, valid content in these bits are copied to the corresponding SWAPx bits on an UPDATE condition.

14.8.10 PGOBUF – Pattern Generation Overwrite Buffer register

Bit	7	6	5	4	3	2	1	0
+0x0B	PGOBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PGOBUF[7:0]: Pattern Generation Override Buffer**
  This register is the buffer for the PGO register. If double buffering is used, valid content in this register is copied to the PGO register on an UPDATE condition.

14.8.11 PGVBUF – Pattern Generation Value Buffer register

Bit	7	6	5	4	3	2	1	0
+0x0C	PGVBUF[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – PGVBUF[7:0]: Pattern Generation Value Buffer**
  This register is the buffer for the PGV register. If double buffering is used, valid content in this register is copied to the PGV register on an UPDATE condition.

14.8.12 OUTOVDIS – Output Override Disable register

Bit	7	6	5	4	3	2	1	0
+0x0F	OUTOVDIS[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – OUTOVDIS[7:0]: Output Override Disable**
  These bits disable the automatic override of the corresponding output port register (i.e., one-to-one bit relation to pin position).

14.9 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	UPSEL		OTMX[2:0]		DTI3EN	DTI2EN	DTI1EN	DTI0EN	203
+0x01	DTBOTH	DTBOTH[7:0]								203
+0x02	DTLS	DTLS[7:0]								203
+0x03	DTHS	DTHS[7:0]								203
+0x04	STATUSCLR	–	–	–	–	–	SWAPBUFV	PGVBUFV	PGOBUFV	204
+0x05	STATUSSET	–	–	–	–	–	SWAPBUGV	PGVBUFV	PGOBUFV	204
+0x06	SWAP	–	–	–	–	SWAP3	SWAP2	SWAP1	SWAP0	204
+0x07	PGO	PGO[7:0]								204
+0x08	PGV	PGV[7:0]								205
+0x09	Reserved	–	–	–	–	–	–	–	–	
+0x0A	SWAPBUF	–	–	–	–	SWAP3BUF	SWAP2BUF	SWAP1BUF	SWAP0BUF	205
+0x0B	PGOBUF	PGOBUF[7:0]								205
+0x0C	PGVBUF	PGVBUF[7:0]								205
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	OUTOVDIS	OUTOVDIS[7:0]								205

15. Hi-Res – High-Resolution Extension

15.1 Features

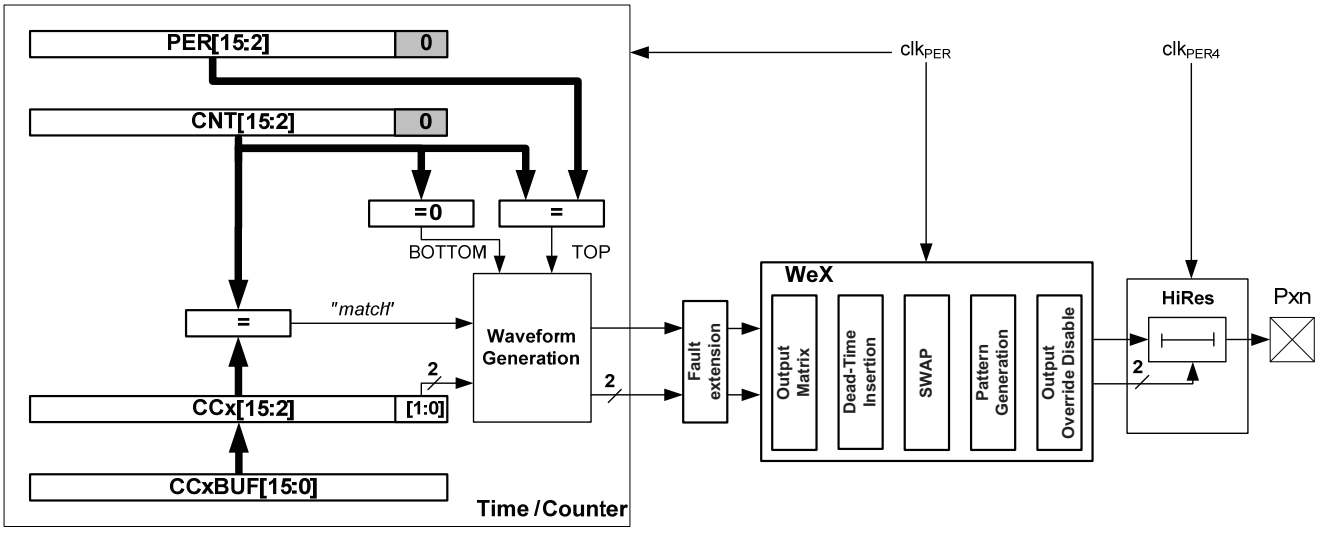
- Increases waveform generator resolution up to 8x (3 bits)
- Supports frequency, single-slope PWM, and dual-slope PWM generation
- Supports the waveform extension when this is used for the same timer/counter

15.2 Overview

The high-resolution (hi-res) extension can be used to increase the resolution of the waveform generation output from a timer/counter by four or eight. It can be used for a timer/counter doing frequency, single-slope PWM, or dual-slope PWM generation. It can also be used with the waveform extension (WeX) if this is used for the same timer/counter.

The hi-res extension uses the peripheral 4x clock (Clk_{PER4}). The system clock prescalers must be configured so the peripheral 4x clock frequency is four times higher than the peripheral and CPU clock frequency when the hi-res extension is enabled. Refer to "System clock selection and prescalers" on page 98 for more details.

Figure 15-1. Timer/counter operation with hi-res extension enabled.



When the hi-res extension is enabled, the timer/counter must run from a non-prescaled peripheral clock. The timer/counter will ignore its two least-significant bits (lsb) in the counter, and counts by four for each peripheral clock cycle. Overflow/underflow and compare match of the 14 most-significant bits (msb) is done in the timer/counter. Count and compare of the two lsb is handled and compared in the hi-res extension running from the peripheral 4x clock.

The two lsb of the timer/counter period register must be set to zero to ensure correct operation. If the count register is read from the application code, the two lsb will always be read as zero, since the timer/counter run from the peripheral clock. The two lsb are also ignored when generating events.

When the hi-res plus feature is enabled, the function is the same as with the hi-res extension, but the resolution will increase by eight instead of four. This also means that the 3 lsb are handled by the hi-res extension instead of 2 lsb, as when only hi-res is enabled. The extra resolution is achieved by counting on both edges of the peripheral 4x clock.

The hi-res extension will not output any pulse shorter than one peripheral clock cycle; i.e., a compare value lower than four will have no visible output.

15.3 Register description

15.3.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	HRPLUS[1:0]		HREN[1:0]	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:2 – HRPLUS[1:0]: High Resolution Plus**
  These bits enable the high-resolution plus mode for a timer/counter according to Table 15-1.
  Hi-res plus is the same as hi-res, but will increase the resolution by eight (3 bits) instead of four. The extra resolution is achieved by operating at both edges of the peripheral 4x clock.

Table 15-1. High resolution plus.

HRPLUS[1:0]	Group configuration	Description
00	NONE	None
01	HRP4	Hi-res plus enabled on timer/counter 4
10	HRP5	Hi-res plus enabled on timer/counter 5
11	BOTH	Hi-res plus enabled on both timer/counters 4 and 5

- **Bit 1:0 – HREN[1:0]: High Resolution Enable**
  These bits enables the high-resolution mode for a timer/counter according to Table 15-2.
  Setting one or both HREN bits will enable high-resolution waveform generation output for the entire general purpose I/O port. This means that both timer/counters connected to the same port must enable hi-res if both are used for generating PWM or FRQ output on pins.

Table 15-2. High resolution enable.

HREN[1:0]	Group configuration	Description
00	NONE	None
01	HR4	Hi-res enabled on timer/counter 4
10	HR5	Hi-res enabled on timer/counter 5
11	BOTH	Hi-res enabled on both timer/counters 4 and 5

15.4 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	HRPLUS[1:0]		HREN[1:0]		208

16. Fault Extension

16.1 Features

- Connected to timer/counter output and waveform extension input
- Event controlled fault protection for instant and predictable fault triggering
- Fast, synchronous and asynchronous fault triggering
- Flexible configuration with multiple fault sources
- Recoverable fault modes
  - Restart or halt the timer/counter on fault condition
  - Timer/counter input capture on fault condition
  - Waveform output active time reduction on fault condition
- Non-recoverable faults
  - Waveform output is forced to a pre-configured safe state on fault condition
  - Optional fuse output value configuration defining the output state during system reset
- Flexible fault filter selections
  - Digital filter to prevent false triggers from I/O pin glitches
  - Fault blanking to prevent false triggers during commutation
  - Fault input qualification to filter the fault input during the inactive output compare states

16.2 Overview

The fault extension enables event controlled fault protection by acting directly on the generated waveforms from timer/counter compare outputs. It can be used to trigger two types of faults with the following actions:

- Recoverable faults: the timer/counter can be restarted or halted as long as the fault condition is preset. The compare output pulse active time can be reduced as long as the fault condition is preset. This is typically used for current sensing regulation, zero crossing re-triggering, demagnetization re-triggering, and so on.
- Non-recoverable faults: the compare outputs are forced to a safe and pre-configured values that are safe for the application. This is typically used for instant and predictable shut down and to disable the high current or voltage drivers.

Events are used to trigger a fault condition. One or several simultaneous events are supported, both synchronously or asynchronously. By default, the fault extension supports asynchronous event operation, ensuring predictable and instant fault reaction, including system power modes where the system clock is stopped.

By using the input blanking, the fault input qualification or digital filter option in event system, the fault sources can be filtered to avoid false faults detection.

16.3 Timer/counter considerations

Each timer/counter supports the fault extension, but the fault extension may not be enabled for all timer/counters. For details on available fault extension units and corresponding timer/counters, refer to the device datasheet.

16.3.1 Polarity configuration

For details on output polarity description and settings, refer to “Output polarity” on page 171 and “CTRLC – Control register C” on page 176.

16.3.2 Waveform generation

The recoverable faults can be enabled in any waveform generation mode, except dual slope PWM. Non-recoverable faults can be enabled in any timer/counter operation mode.

16.4 Faults

A fault input is an event from the event system. When an event occurs, the fault triggers the configured fault actions.

All events from the event system can be used as fault input. For details on event system and available events, refer to “Event System ” on page 79.

16.4.1 Fault types

The fault extension defines two fault types.

- Recoverable faults can restart or halt the timer/counter. Two fault inputs, called fault A and fault B, can trigger recoverable fault actions on compare channels A and B from the corresponding timer/counter. The compare channels outputs can be clamped to inactive state as long as the fault condition is present, or from the first valid fault condition detection and until the end of the timer/counter cycle.
- Non-recoverable fault forces the compare channels outputs to a safe and pre-configured value. The fault input, called fault E, performs non-recoverable fault actions on all compare channels from the corresponding timer/counter.

16.4.2 Input selection

The fault extension uses up to three event channels. The lowest channel is selected by the EVSEL bits in the corresponding timer/counter, as described in “CTRLD – Control register D” on page 176. The next two subsequent event channels are automatically selected for the fault extension.

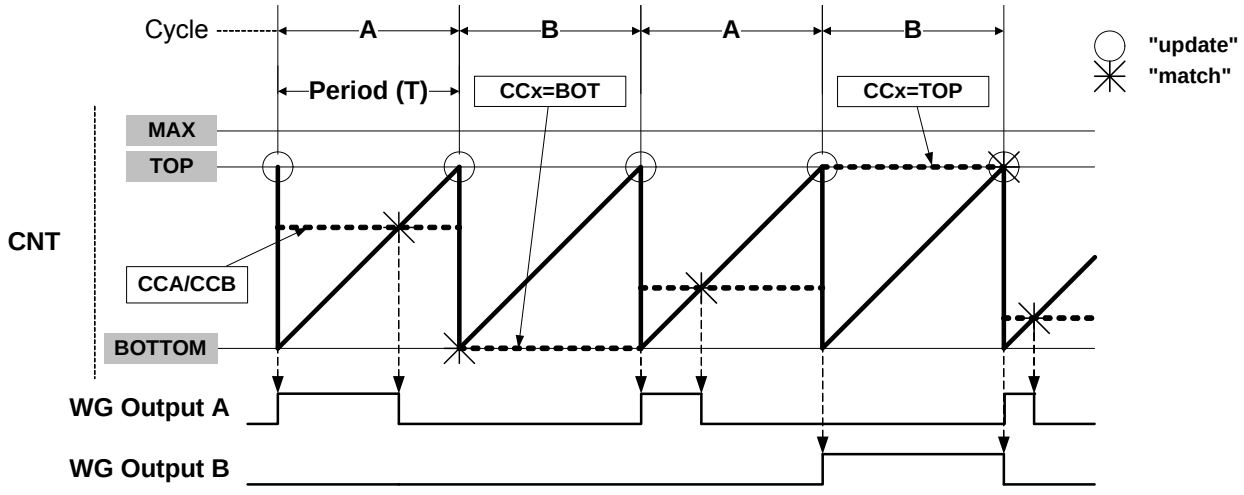
The first two event channels can be used as fault A or fault B inputs. In two ramp mode of operation, it is also possible to link the fault source for the current cycle, to the fault state from the previous cycle. This is typically used for various applications, such LED control.

The non-recoverable fault can use any of the three available even lines.

16.4.3 Ramp modes

Two ramp modes are supported and both require the timer/counter running in single slope PWM mode. By default, the timer/counter is enabled in RAMP1 mode, which is the standard timer/counter operation described in “Compare channel” on page 169. In RAMP2 mode, two consecutive timer/counter cycles are interleaved, as shown in Figure 16-1. In cycle A, the waveform output B is disabled, and in cycle B, the waveform output A is disabled. The cycle index (cycle A or cycle B) can be controlled using the cycle index commands bits. For details refer to “CTRLGSET – Control register G Set” on page 222.

Figure 16-1. Timer/counter cycle in RAMP2 mode.



16.4.4 Fault filtering

Three filtering types are available. The recoverable faults can use all three filters independently or various filters combination. The non-recoverable fault can use the input filtering method only.

The filter type or filter combination must be decided by the application.

16.4.4.1 Input filtering

By default, the event detection is asynchronous. When the event occurs, the fault system will immediately and asynchronously performs the selected fault action on the compare channel output, including system power modes where the clock is not available. To avoid false fault detection on external events (e.g. a glitch on I/O port) the digital filter can be enabled in the corresponding event channel. In this case, the event detection and routing will be synchronous, and the event action will be delayed between two and three peripheral clock cycles, plus the selected digital filter coefficient.

For details on digital filter coefficient, refer to DIGFILT description in “CHnCTRL – Event Channel n Control register” on page 90.

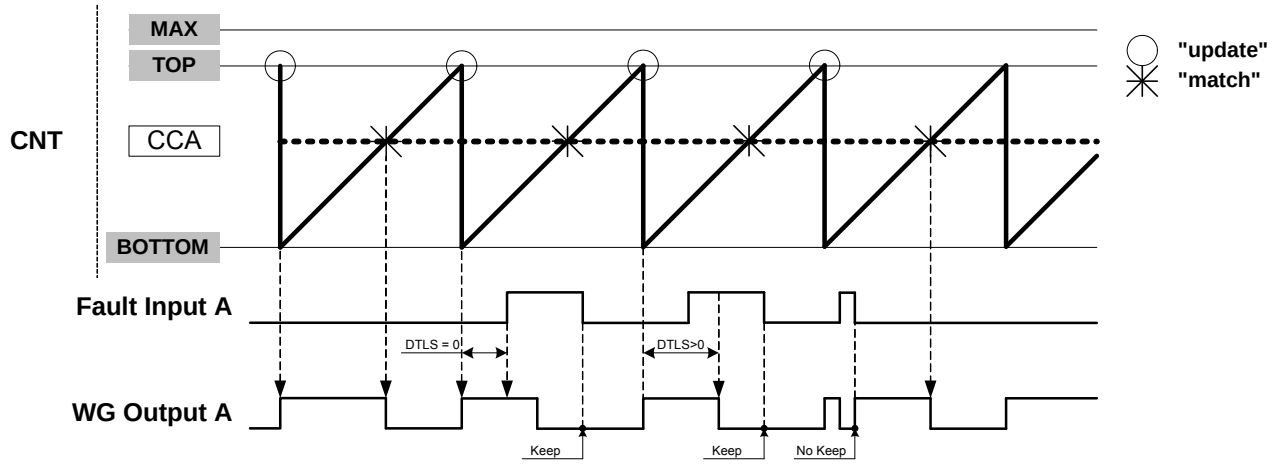
16.4.4.2 Fault blanking

Fault blanking (BLANK) provides a way to suppress fault inputs during the beginning of the active time of the waveform output. Using this method, faults can be triggered only after a configured time, and will prevent false fault triggering during commutation. The fault blanking time is set in DTLS and DTHS registers in waveform extension unit (WeX). The registers values define the number of peripheral clock cycles where the fault input is inhibited. The blanking always starts from the beginning of the cycle. As shown in Figure 16-2, the compare output value can be kept inactive until the end of the cycle, or resume the standard operation when the fault condition is no longer present.

DTLS register configures the blanking time of compare channel A from each TC4 and TC5. Both compare channels A will have the same blanking time. For details on DTLS register, refer to “DTLS – Dead-Time Low Side register” on page 203.

DTHS register configures the blanking time of compare channel B from each TC4 and TC5. Both compare channels B will have the same blanking time. For details on DTHS register, refer to “DTHS – Dead-Time High Side register” on page 203.

Figure 16-2. Waveform generation with blanking enabled.



As example, the maximum blanking time is:

- 256 / (32 ×10^6s) = 8µs when 32MHz peripheral clock frequency
- 256 / (10^6s) = 256µs when 1MHz peripheral clock frequency

16.4.4.3 Fault input qualification

If the fault input qualification (QUAL) is enabled, the fault A and B will trigger fault actions if the corresponding compare channel output has an active level, as shown in Figure 16-3 on page 212 and Figure 16-4 on page 212.

Figure 16-3. Fault input qualification in RAMP1 mode.

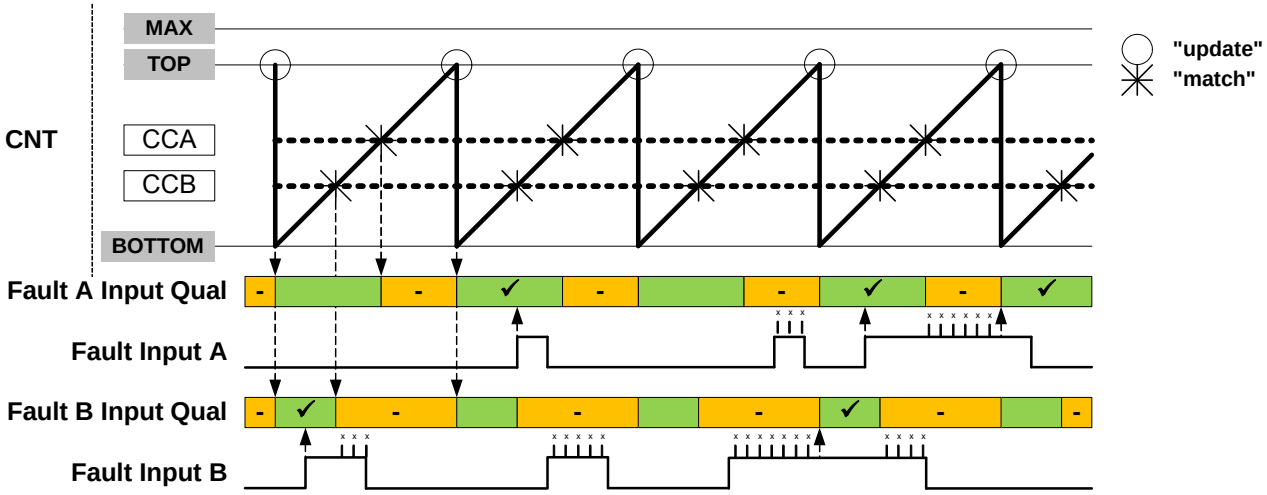
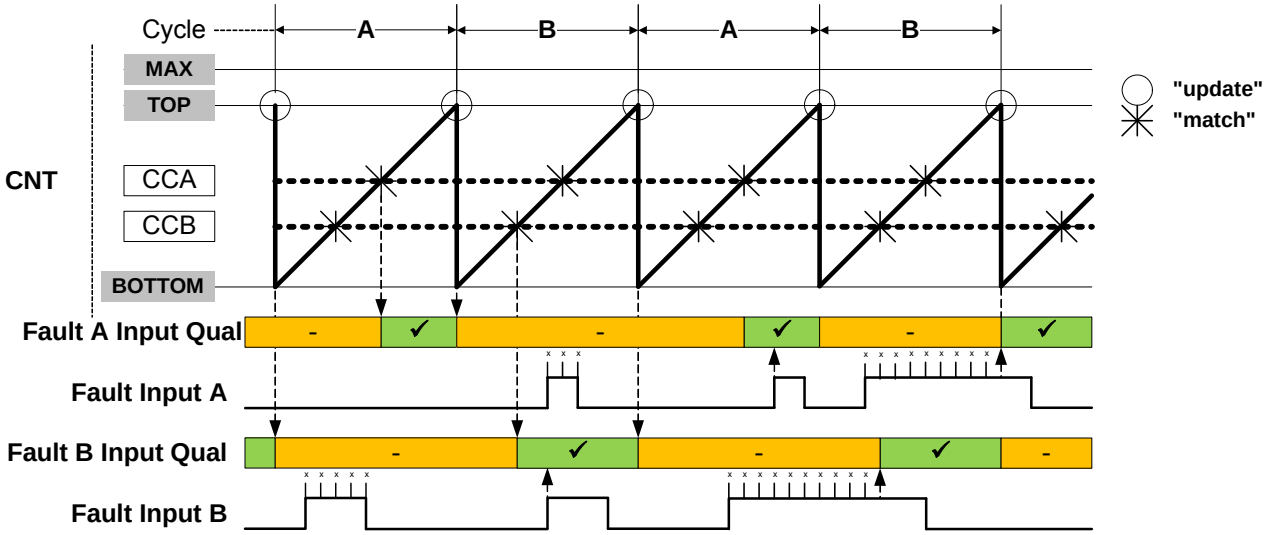


Figure 16-4. Fault input qualification in RAMP2 mode with inverted polarity.



16.4.5 Fault actions

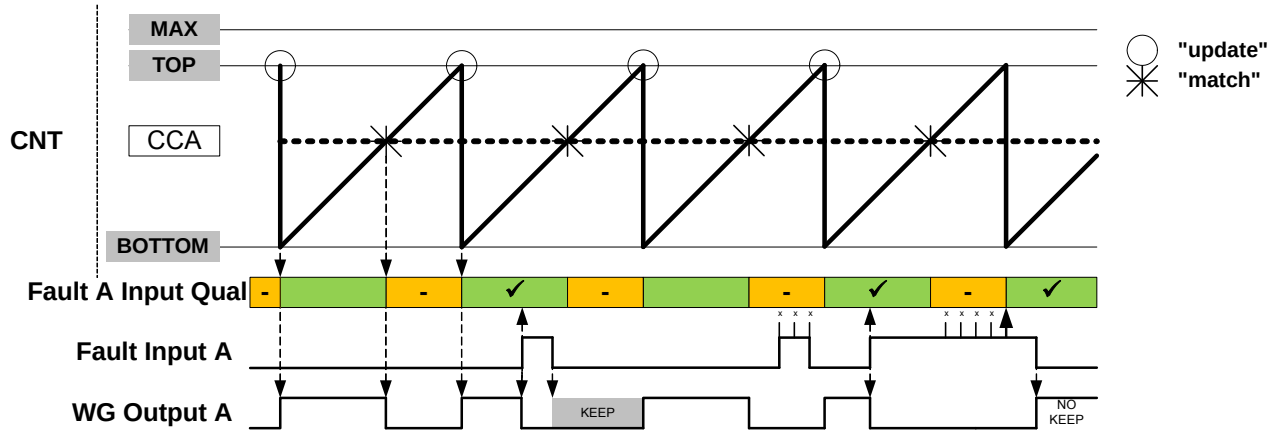
Different fault actions can be configured individually for fault A and fault B. Most fault actions are not mutually exclusive; hence two or more actions can be enabled at the same time to achieve a result that is a combination of fault actions.

16.4.5.1 Keep action

When the keep action (KEEP) is enabled, the output will be clamped to its inactive value when the fault condition is present. The clamp will be released on the start of the first cycle after the fault condition is no longer present.

Figure 16-5 on page 213 shows compare channel A output when keep action and fault input qualification are enabled for fault A.

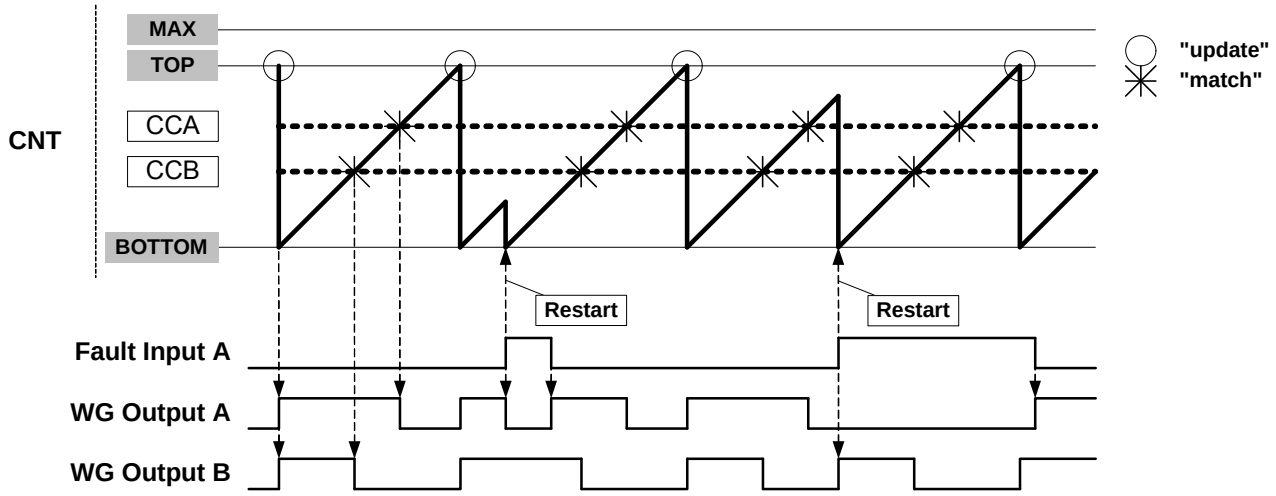
Figure 16-5. Waveform generation with fault input qualification and keep action.



16.4.5.2 Restart action

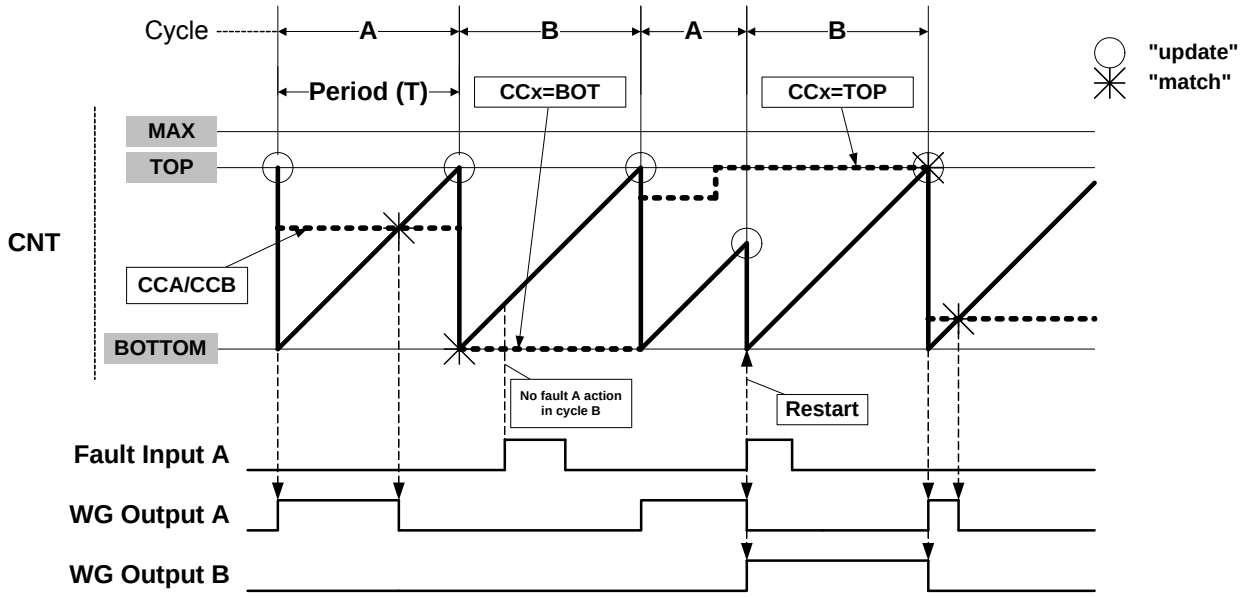
When restart action (RESTART) is enabled, the timer/counter will be restarted when a fault condition is present. The on-going cycle is stopped and the timer/counter starts a new cycle, as shown in Figure 16-6. When the new cycle starts, the compare outputs will be clamped to inactive level as long as the fault condition is present.

Figure 16-6. Waveform generation with restart action in RAMP1 mode.



Note that in RAMP2 mode of operation, when a new timer/counter cycle starts, the cycle index will change automatically, as shown in Figure 16-7 on page 214. Fault A and fault B are qualified only during the cycle A and cycle B respectively, i.e. the compare channel outputs are not forced to inactive level as long as the fault condition is present in cycle B or cycle A respectively.

Figure 16-7. Waveform generation with restart action in RAMP2 mode.



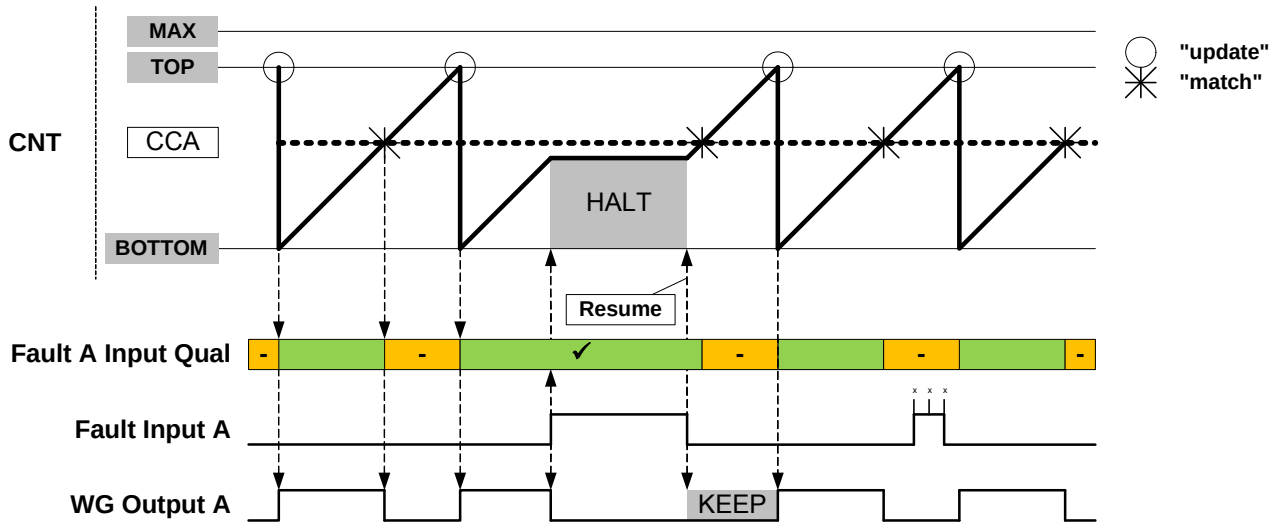
16.4.5.3 Capture action

When capture action (CAPT) is enabled, the fault can be time stamped assuming a timer/counter capture channel is enabled and the event action selection is set to fault capture (FAULTx). For details on event actions, refer to “CTRLD – Control register D” on page 176.

16.4.5.4 Hardware halt action

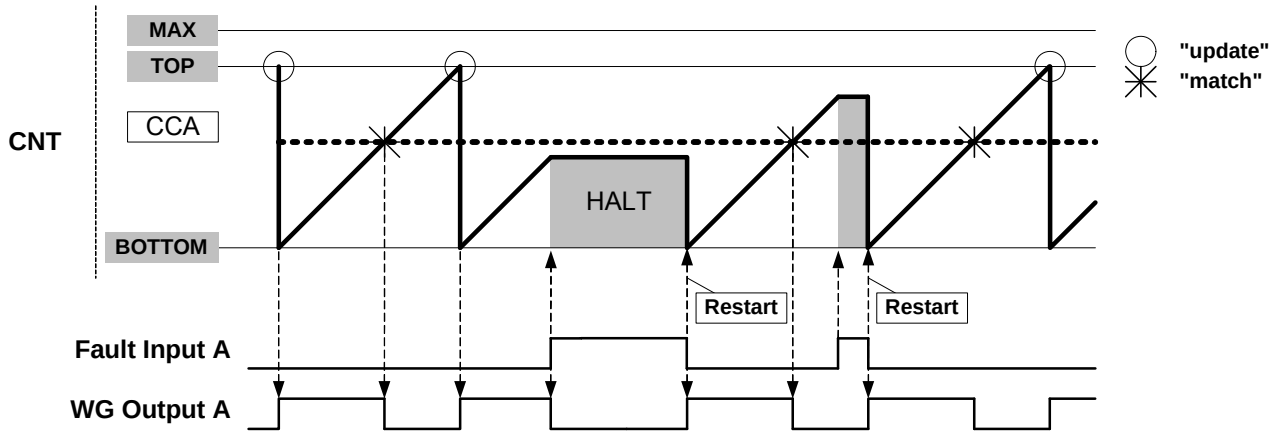
When hardware halt action (HWHALT) is enabled, the timer/counter is halted and the cycle is extended as long as the fault is present. Figure 16-8 shows an example where keep and hardware halt actions are enabled for fault A. The compare channel output A is clamped to inactive level as long as the timer/counter is halted. The timer/counter resumes the counting operation as soon as the fault condition is no longer present.

Figure 16-8. Waveform generation with hardware halt, fault input qualification and keep action.



If the restart action is enabled, the timer/counter is halted as long as the fault condition is present and restarted when the fault condition is no longer present, as shown in Figure 16-9 on page 215. The compare channel output A is clamped to inactive level as long as the timer/counter is halted. Note that in RAMP2 mode of operation, when a new timer/counter cycle starts, the cycle index will change automatically.

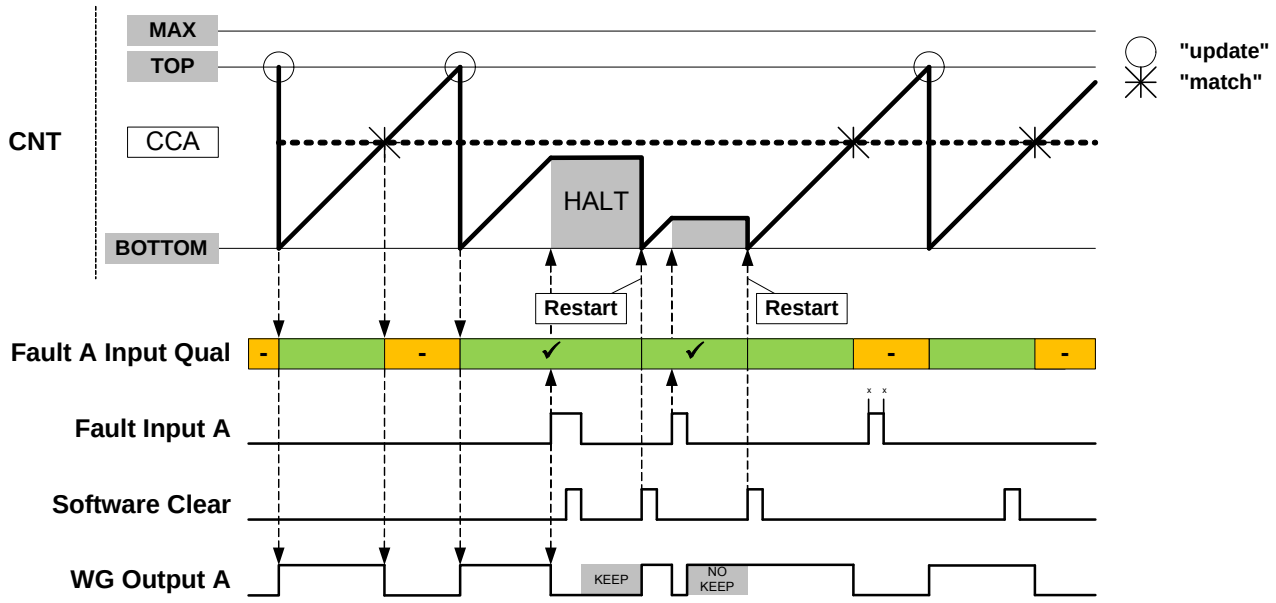
Figure 16-9. Waveform generation with hardware halt and restart action.



16.4.5.5 Software halt action

The software halt action (SWHALT) is similar to hardware halt action with one exception. To restart the timer/counter, the fault condition is no longer present and the corresponding HALTxCLR bit in "CTRLGCLR – Control register G Clear" on page 221 must be set, as shown in Figure 16-10.

Figure 16-10.Waveform generation with software halt, fault input qualification and, restart and keep actions.



16.4.5.6 Non-recoverable fault action

The non-recoverable fault action will force all the compare outputs to a pre-defined level programmed in FUSEBYTE6, as described in section "FUSEBYTE6 – Fuse Byte 6" on page 33.

If the FUSE configuration is set, the enabled compare channel output will be forced as long as the non-recoverable fault is present. When the fault condition is no longer present, timer/counter restarts when the STATEECLR bit in "CTRLGCLR – Control register G Clear" on page 221 is set. To restart the timer/counter from the beginning of a new cycle, the restart command in "CTRLGCLR/CTRLGSET – Control register G Clear/Set" on page 179 must be written before writing the STATEECLR bit.

16.4.6 Interrupts and events

If SOFTA bit in “CTRLB – Control register B” on page 218 is set, a fault A detection will set the FAULTA flag in “CTRLGCLR – Control register G Clear” on page 221 and the error interrupt flag in “ INTFLAGS – Interrupt Flags register” on page 180. If enabled, the optional timer/counter error interrupt is generated.

If SOFTB bit in “CTRLD – Control register D” on page 219 is set, a fault B detection will set the FAULTB flag in “CTRLGCLR – Control register G Clear” on page 221 and the error interrupt flag in “ INTFLAGS – Interrupt Flags register” on page 180. If enabled, the optional timer/counter error interrupt is generated.

A fault E detection will set the FAULTE flag in “CTRLGCLR – Control register G Clear” on page 221 and the error interrupt flag in “ INTFLAGS – Interrupt Flags register” on page 180. If enabled, the optional timer/counter error interrupt is generated.

The fault extension is not source of events to the event system. The fault extension uses events as described in section “Input selection” on page 210.

16.4.7 Change protection

To avoid unintentional configuration changes, the CTRLA register in the FAULT can be protected by writing the corresponding lock bit in the fault extension lock register. For more details, refer to“I/O memory protection” on page 25 and “FAULTLOCK – Fault Extension Lock register” on page 45.

When the lock bit is set, CTRLA register cannot be changed.

16.5 Register description

16.5.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	RAMP[1:0]		FDDBD	PORTCTRL	FUSE	FILTERE	SRCE[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – RAMP[1:0]: Ramp Mode Selection**
  These bits select the ramp mode (RAMP) according to Table 16-1, and define in which cycle fault A and B will trigger fault actions.

Table 16-1. RAMP mode selection.

RAMP[1:0]	Group configuration	Description
00	RAMP1	Default mode: - Fault A triggers fault action in any cycle - Fault B triggers fault action in any cycle
01		Reserved
10	RAMP2	Cycle A and cycle B are interleaved: - Fault A triggers fault action in cycle A - Fault B triggers fault action in cycle B
11		Reserved

- **Bit 5 – FDDBD: Fault Detection on Debug Break Detection**
  By default, the on-chip debug interface fault protection is enabled and a request from the debug interface will trigger an non-recoverable fault. When this bit is set, the on-chip debug interface fault protection is disabled and a break request will not trigger any fault.
- **Bit 4 – PORTCTRL: Port Control Mode**
  When this bit is set, an enabled compare channel will force the output configuration on the corresponding I/O pin. This bit has no effect if the channel is disabled or configured in capture mode of operation.
- **Bit 3 – FUSE: Fuse State**
  When this bit is set, the fuse value programmed in FUSEBYTE6 is forced on enabled compare channels outputs when a non-recoverable fault condition is present. Note that when FDACTx fuses are programmed, the same fuse value is applied during the system reset sequence. For details on fuse settings, refer to “FUSEBYTE6 – Fuse Byte 6” on page 33.
- **Bit 2 – FILTERE: Fault E Digital Filter Selection**
  Setting this bit enables the selected event channel digital filter output as event source for fault E. For details refer to “Input filtering” on page 211.
- **Bit 1:0 – SRCE[1:0]: Fault E Input Selection**
  These bits select the event channel input for fault E input, as shown in Table 16-2 on page 218. For details on event channel selection, refer to EVSEL description in “CTRLD – Control register D” on page 176.

Table 16-2. Fault E input selection.

SRCE[1:0]	Group configuration	Description
00	DISABLE	Fault protection disabled
01	CHn	Event channel n
10	CHn1	Event channel n+1
11	CHn2	Event channel n+2

16.5.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	SOFTA	HALTA[1:0]		RESTARTA	KEEPA	–	SRCA[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – SOFTA: Fault A Software Mode**
  When this bit is set, a fault A detection will set the timer/counter error interrupt flag, as described in section “Interrupts and events” on page 216.
- **Bit 6:5 – HALTA[1:0]: Fault A Halt Action**
  These bits select the halt action for fault A, as defined in Table 16-3.

Table 16-3. Fault A halt action selection.

HALTA[1:0]	Group configuration	Description
00	DISABLE	Halt action disabled
01	HW	Hardware halt action
10	SW	Software halt action
11	–	Reserved

- **Bit 4 – RESTARTA: Fault A Restart Action**
  Setting this bit enables the restart action for the fault A.
- **Bit 3 – KEEPA: Fault A Keep Action**
  Setting this bit enables keep action for the fault A.
- **Bit 2 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1:0 – SRCA[1:0]: Fault A Source Selection**
  These bits select the event channel source for the fault A, as defined in Table 16-4 on page 219. For details on event channel selection, refer to EVSEL description in “CTRLD – Control register D” on page 176.

Table 16-4. Fault A source selection.

SRCA[1:0]	Group configuration	Description
00	DISABLE	Fault disabled
01	CHn	Event channel n
10	CHn1	Event channel (n+1)
11	LINK	Fault A input source is linked to the fault B state from the end of the previous cycle. If keep action is disabled, the fault A duration is one peripheral clock cycle. Alternatively, if the keep action is enabled, the fault A duration is one complete timer/counter cycle.

16.5.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	CAPTA	–	–	FILTERA	BLANKA	QUALA
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5 – CAPTA: Fault A Capture**
  When this bit is set, the fault A detection triggers a timer/counter input capture operation. For details on event actions and event selection, refer to “CTRLD – Control register D” on page 176.
- **Bit 4:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – FILTERA: Fault A Digital Filter Selection**
  Setting this bit enables the selected event channel digital filter output as event source for fault A. For details refer to “Input filtering” on page 211.
- **Bit 1 – BLANKA: Fault A Blanking**
  Setting this bit enables input blanking for fault A. For details, refer to “Fault blanking” on page 211.
- **Bit 0 – QUALA: Fault A Qualification**
  Setting this bit enables the fault A input qualification. For details, refer to“Fault input qualification” on page 211.

16.5.4 CTRLD – Control register D

Bit	7	6	5	4	3	2	1	0
+0x03	SOFTB	HALTB[1:0]		RESTARTB	KEEPB	–	SRCB[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – SOFTB: Fault B Software Mode**
  When this bit is set, a fault B detection will set the timer/counter error interrupt flag, as described in section “Interrupts and events” on page 216.
- **Bit 6:5 – HALTB [1:0]: Fault B Halt Action**
  These bits select the halt action for fault B as defined in Table 16-5 on page 220.

Table 16-5. Fault B halt action selection.

HALTB[1:0]	Group configuration	Description
00	DISABLE	Halt action disabled
01	HW	Hardware halt action
10	SW	Software halt action
11	–	Reserved

- **Bit 4 – RESTARTB: Fault B Restart Action**
  Setting this bit enables the restart action for fault B.
- **Bit 3 – KEEPB: Fault B Keep Action**
  Setting this bit enables the keep action for fault B.
- **Bit 2 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written
- **Bit 1:0 – SRCB[1:0]: Fault B Source Selection**
  These bits select the event channel source for fault B, as defined in Table 16-6. For details on event channel selection, refer to EVSEL description in"CTRLD – Control register D" on page 176.

Table 16-6. Fault B source selection.

SRCB[1:0]	Group configuration	Description
00	DISABLE	Fault disabled
01	CHn	Event channel n
10	CHn1	Event channel (n+1)
11	LINK	Fault B input source is linked to the fault A state from the end of the previous cycle. If keep action is disabled, the fault B duration is one peripheral clock cycle. Alternatively, if the keep action is enabled, the fault B duration is one complete timer/counter cycle.

16.5.5 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	CAPTB	–	–	FILTERB	BLANKB	QUALB
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5 – CAPTB: Fault B Capture**
  When this bit is set, the fault B detection triggers a timer/counter input capture operation. For details on event actions and event selection, refer to "CTRLD – Control register D" on page 176.
- **Bit 4:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 2 – FILTERB: Fault B Digital Filter Selection**
  Setting this bit enables the selected event channel digital filter output as event source for fault B. For details, refer to “Input filtering” on page 211.
- **Bit 1 – BLANKB: Fault B Blanking**
  Setting this bit enables input blanking for fault B. For details, refer to “Fault blanking” on page 211.
- **Bit 0 – QUALB: Fault B Qualification**
  Setting this bit enables the fault B input qualification. For details, refer to“Fault input qualification” on page 211.

16.5.6 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x05	STATEB	STATEA	STATEE	–	IDX	FAULTBIN	FAULTAIN	FAULTEIN
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – STATEB: Fault B State**
  This bit is set by hardware when fault B condition is present. The bit is cleared by hardware when the fault A action is completed.
- **Bit 6 – STATEA: Fault A State**
  This bit is set by hardware when fault A condition is present. The bit is cleared by hardware when the fault B action is completed.
- **Bit 5 – STATEE: Fault E State**
  This bit is set by hardware when fault E condition is present. The bit is cleared by hardware when the fault E action is completed.
- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 3 – IDX: Channel Index Flag**
  In RAMP2 mode of operation, the bit is cleared during the cycle A and set during the cycle B. In RAMP1 mode of operation, the bit is always read zero. For details on ramp modes, refer to section “Ramp modes” on page 210.
- **Bit 2 – FAULTBIN: Fault B Flag**
  This flag is set by hardware as long as the fault B condition is present. The flag is cleared by hardware when the fault B condition is no longer present.
- **Bit 1 – FAULTAIN: Fault A Flag**
  This flag is set by hardware as long as the fault A condition is present. The flag is cleared by hardware when the fault A condition is no longer present.
- **Bit 0 – FAULTEIN: Fault E Flag**
  This flag is set by hardware when fault E condition is present. The flag is cleared by hardware when the fault E condition is no longer present.

16.5.7 CTRLGCLR – Control register G Clear

Bit	7	6	5	4	3	2	1	0
+0x06	HALTBCLR	HALTACLR	STATEECLR	–	–	FAULTB	FAULTA	FAULTE
Read/Write	R/W	R/W	R/W	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – HALTBCLR: State B Clear**
  Setting this bit will clear the STATEB bit in STATUS register when FAULTBIN is cleared. If software halt command is enabled, the timer/counter internal halt command is released.

- **Bit 6 – HALTACLR: State A Clear**
  Setting this bit will clear the STATEA bit in STATUS register when FAULTAIN is cleared. If software halt command is enabled, the timer/counter internal halt command is released.
- **Bit 5 – STATEECLR: State E Clear**
  Setting this bit will clear the STATEE bit in STATUS register when FAULTEIN is cleared. The timer/counter will restart from the last CNT value. To restart the timer/counter from BOTTOM, the timer/counter restart command must be executed before setting the STATEECLR bit. For details on timer/counter commands, refer to command description in"CTRLGCLR/CTRLGSET – Control register G Clear/Set" on page 189.
- **Bit 4:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – FAULTB: Fault B Flag**
  This bit is set by hardware when fault B input occurs and if SOFTB bit is set. The flag is cleared by writing a one to its bit location.
- **Bit 1 – FAULTA: Fault A Flag**
  This bit is set by hardware when fault A input occurs and if SOFTA bit is set. The flag is cleared by writing a one to its bit location.
- **Bit 0 – FAULTE: Fault E Flag**
  This bit is set by hardware when fault E input occurs. The flag is cleared by writing a one to its bit location.

16.5.8 CTRLGSET – Control register G Set

Bit	7	6	5	4	3	2	1	0
+0x07	FAULTBSW	FAULTASW	FAULTESW	IDXCMD[1:0]		–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – FAULTBSW: Software Fault B**
  Setting this bit will trigger and force the fault B action by software. The bit and corresponding fault input are automatically cleared by hardware in the next peripheral clock cycle (clk_{PER}).
- **Bit 6 – FAULTASW: Software Fault A**
  Setting this bit will trigger and force the fault A action by software. The bit and corresponding fault input are automatically cleared by hardware in the next peripheral clock cycle (clk_{PER}).
- **Bit 5 – FAULTESW: Software Fault E**
  Setting this bit will trigger and force the fault E action by software. The bit and corresponding fault input are automatically cleared by hardware in the next peripheral clock cycle (clk_{PER}).
- **Bit 4:3 – IDXCMD[1:0]: Cycle Index Command**
  These bits can be used to force cycle A and cycle B changes in RAMP2 mode according to Table 16-7. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.

Table 16-7. Index command selection.

IDXCMD[1:0]	Group configuration	Description
00	DISABLE	Command disabled: IDX toggles between cycles A and B
01	SET	Set IDX: cycle B will be forced in the next cycle
10	CLEAR	Clear IDX: cycle A will be forced in next cycle
11	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

- **Bit 2:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

16.6 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	RAMP[1:0]		FDDBD	PORTCTRL	FUSE	FILTERE	SRCE[1:0]		217
+0x01	CTRLB	SOFTA	HALTA[1:0]		RESTARTA	KEEPA	-	SRCA[1:0]		218
+0x02	CTRLC	-	-	CAPTA	-	-	FILTERA	BLANKA	QUALA	219
+0x03	CTRLD	SOFTB	HALTB[1:0]		RESTARTB	KEEPB	-	SRCB[1:0]		219
+0x04	CTRLE	-	-	CAPTB	-	-	FILTERB	BLANKB	QUALB	220
+0x05	STATUS	STATEB	STATEA	STATEE	-	IDX	FAULTBIN	FAULTAIN	FAULTEIN	221
+0x06	CTRLGCLR	HALTBCLR	HALTACLR	STATEECLR	-	-	FAULTB	FAULTA	FAULTE	221
+0x07	CTRLGSET	FAULTBSW	FAULTASW	FAULTESW	IDXCMD[1:0]		-	-	-	222

17. RTC – Real Time Counter

17.1 Features

- 16-bit resolution
- Selectable clock source
  - 32.768kHz external crystal
  - External clock
  - 32.768kHz internal oscillator
  - 32kHz internal ULP oscillator
- Programmable 10-bit clock prescaling
- One Compare register
- One Period register
- Clear counter on period overflow
- Optional interrupt/event on overflow and compare match
- Correction for external crystal selection

17.2 Overview

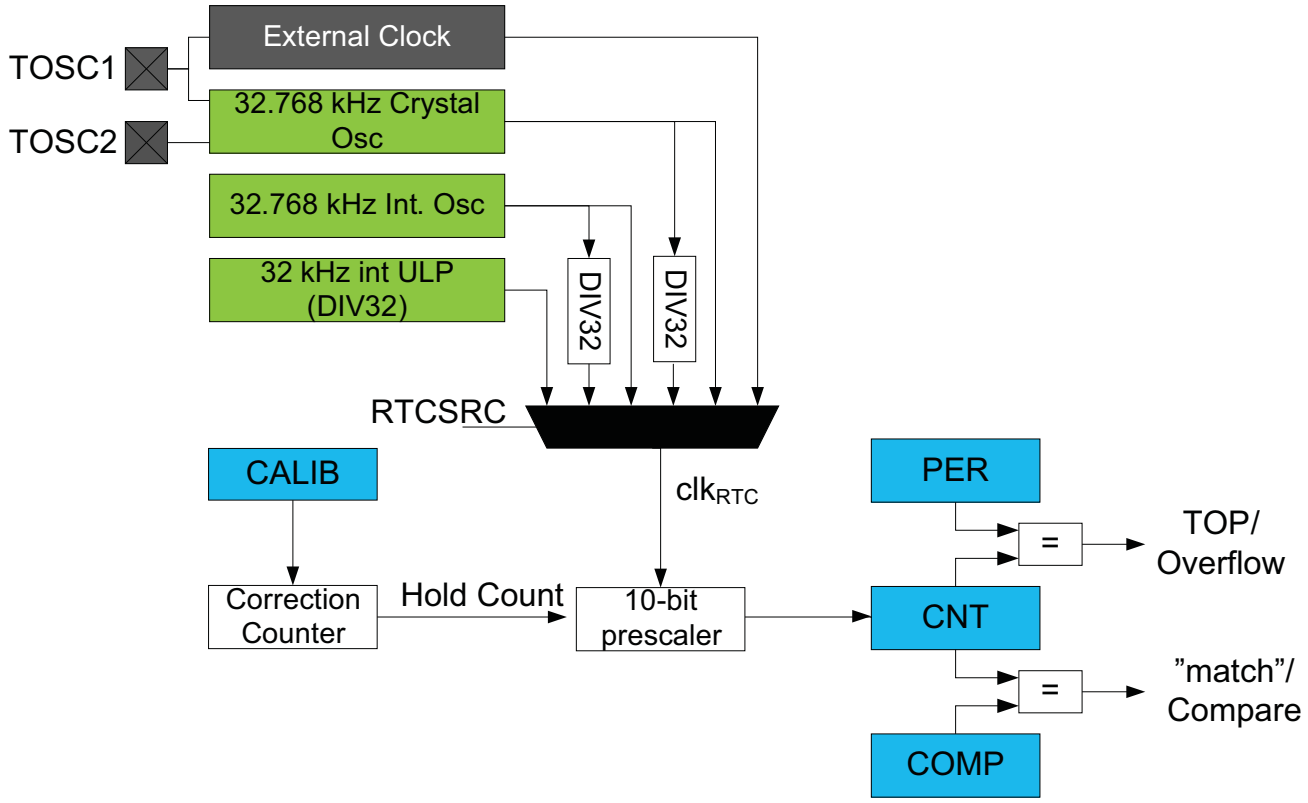
The 16-bit real-time counter (RTC) is a counter that typically runs continuously, including in low-power sleep modes, to keep track of time. It can wake up the device from sleep modes and/or interrupt the device at regular intervals.

The RTC clock is typically the 1.024kHz output from a high-accuracy crystal of 32.768kHz, and this is the configuration most optimized for low power consumption. The faster 32.768kHz output can be selected if the RTC needs a resolution higher than 1ms. The RTC can also be clocked from an external clock signal, the 32.768kHz internal oscillator or the 32kHz internal ULP oscillator.

The RTC includes a 10-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. With a 32.768kHz clock source, the maximum resolution is 30.5µs, and time-out periods can range up to 2000 seconds. With a resolution of 1s, the maximum time-out period is more than 18 hours (65536 seconds). The RTC can give a compare interrupt and/or event when the counter equals the compare register value, and an overflow interrupt and/or event when it equals the period register value.

The RTC also supports correction when operated using external crystal selection. An externally calibrated value will be used for correction. The RTC can be calibrated by software to an accuracy of ±0.5PPM relative to a minimum reference clock of 2MHz. The RTC correction operation will either speed up (by skipping count) or slow down (adding extra cycles) the prescaler to account for the clock error.

Figure 17-1. Real-time counter overview.



17.3 Clock domains

The RTC is asynchronous, operating from a different clock source independently of the main system clock and its derivative clocks, such as the peripheral clock. For control and count register updates, it will take a number of RTC clock and/or peripheral clock cycles before an updated register value is available in a register or until a configuration change has effect on the RTC. This synchronization time is described for each register.

17.4 Interrupts and events

The RTC can generate both interrupts and events. The RTC will give a compare interrupt and/or event at the first count after the counter value equals the Compare register value. The RTC will give an overflow interrupt request and/or event at the first count after the counter value equals the Period register value. The overflow will also reset the counter value to zero.

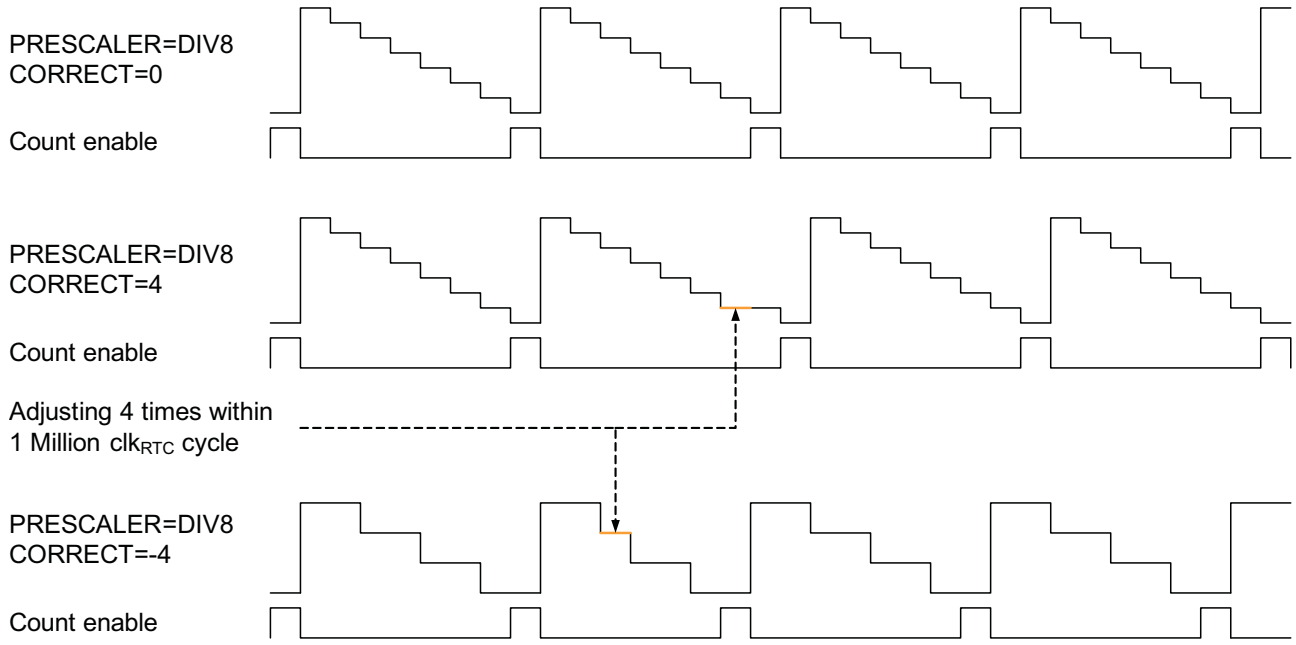
Due to the asynchronous clock domain, events will be generated only for every third overflow or compare match if the period register is zero. If the period register is one, events will be generated only for every second overflow or compare match. When the period register is equal to or above two, events will trigger at every overflow or compare match, just as the interrupt request.

17.5 Correction

The RTC can do internal correction on the RTC crystal clock by taking the PPM error value from the CALIB Register. The CALIB register will be written by software after external calibration or temperature corrections. Correction is done within an interval of approximately 1 million cycles. The correction operation is performed as a single cycle operation – adding or removing one cycle, depending on the nature of error. These single cycle operations will be performed repeatedly the error number of times (ERROR[6:0] - CALIB Register) spread through out the 1 million cycle correction interval. The correction spread over this correction interval is based on the error value. The final correction of the clock will be reflected in the RTC count value available through the CNTL and CNTH registers. When the required correction is speeding up the

clock by skipping cycles, it is required to run the prescaler at a minimum setting of DIV2 (RTC clock/2). When the system correction is disabled by software, the correction is disabled internally when the ongoing one million correction cycle is completed.

Figure 17-2. Real-time counter clock/count correction.



17.6 Register description

17.6.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	CORREN	PRESCALER[2:0]		
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3 – CORREN: Correction Enable**
  Setting this bit enables the correction process. When this bit is written low, the correction is disabled when the ongoing correction cycle is completed. Refer to CALIB Register for the correction value and type.
- **Bits 2:0 – PRESCALER[2:0]: RTC Clock Prescaling Factor**
  These bits define the prescaling factor for the RTC clock according to Table 17-1 on page 227.

Table 17-1. Real Time Counter clock prescaling factor.

PRESCALER[2:0]	Group configuration	RTC clock prescaling
000	OFF	No clock source, RTC stopped
001	DIV1	RTC clock / 1 (No prescaling)
010	DIV2	RTC clock / 2
011	DIV8	RTC clock / 8
100	DIV16	RTC clock / 16
101	DIV64	RTC clock / 64
110	DIV256	RTC clock / 256
111	DIV1024	RTC clock / 1024

17.6.2 STATUS - Status register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	–	–	–	SYNCBUSY
Read/Write	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – SYNCBUSY: Synchronization Busy Flag**
  This flag is set when the CNT, CTRL, PER, or COMP register is busy synchronizing between the RTC clock and system clock domains. This flag is automatically cleared when the synchronization is complete.

17.6.3 INTCTRL - Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	COMPINTLVL[1:0]		OVFINTLVL[1:0]	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:2 – COMPINTLVL[1:0]: Compare Match Interrupt Enable**
  These bits enable the RTC compare match interrupt and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger when COMPIF in the INTFLAGS register is set.
- **Bits 1:0 – OVFINTLVL[1:0]: Overflow Interrupt Enable**
  These bits enable the RTC overflow interrupt and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger when OVFIF in the INTFLAGS register is set.

17.6.4 INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	–	–	COMPIF	OVFIF
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written
- **Bit 1 – COMPIF: RTC Compare Match Interrupt Flag**
  This flag is set on the next count after a compare match condition occurs. It is cleared automatically when the RTC compare match interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.
- **Bit 0 – OVFIF: RTC Overflow Match Interrupt Flag**
  This flag is set on the next count after an overflow condition occurs. It is cleared automatically when the RTC overflow interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

17.6.5 TEMP - Temporary register

Bit	7	6	5	4	3	2	1	0
+0x04	TEMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – TEMP[7:0]: Temporary Bits**
  This register is used for 16-bit access to the counter value, compare value, and TOP value registers. The low byte of the 16-bit register is stored here when it is written by the CPU. The high byte of the 16-bit register is stored when the low byte is read by the CPU.

17.6.6 CALIB – Calibration register

This register stores the error value and the type of correction to be done. This register is written by software with any error value based on external calibration and/or temperature correction/s.

Bit	7	6	5	4	3	2	1	0
+0x06	SIGN	ERROR[6:0]						
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – SIGN: Correction Sign**
  This bit indicates the direction of correction.
  If this bit is LOW then the RTC counter will be slowed down by adding clocks.
  If this bit is HIGH, then the RTC counter will be speeded up by removing clocks. For this setting it is required to set the prescaler to minimum setting of DIV2 (RTC clock/2).
- **Bit 6:0 – ERROR[6:0]: Error Value**
  These bits hold the error value for correction operation.

17.6.7 CNTL – Count register Low

The CNTH and CNTL register pair represents the 16-bit value, CNT. CNT counts positive clock edges on the prescaled RTC clock. Reading and writing 16-bit values requires special attention.

Due to synchronization between the RTC clock and system clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. Application software needs to check that the SYNCBUSY flag in the “STATUS - Status register” on page 227 is cleared before writing to this register.

Bit	7	6	5	4	3	2	1	0
+0x08	CNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – CNT[7:0]: Counter Value Low Byte**
  These bits hold the LSB of the 16-bit Real Time Counter value.

17.6.8 CNTH – Count register High

Bit	7	6	5	4	3	2	1	0
+0x09	CNT[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – CNT[15:8]: Counter Value High Byte**
  These bits hold the MSB of the 16-bit Real Time Counter value.

17.6.9 PERL – Period register Low

The PERH and PERL register pair represents the 16-bit value, PER. PER is constantly compared with the counter value (CNT). A match will set OVFIF in the INTFLAGS register and clear CNT. Reading and writing 16-bit values requires special attention.

Due to synchronization between the RTC clock and system clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. Application software needs to check that the SYNCBUSY flag in the “STATUS - Status register” on page 227 is cleared before writing to this register.

Bit	7	6	5	4	3	2	1	0
+0x0A	PER[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – PER[7:0]: Period Low Byte**
  These bits hold the LSB of the 16-bit RTC TOP value.

17.6.10 PERH - Period register High

Bit	7	6	5	4	3	2	1	0
+0x0B	PER[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – PER[15:8]: Period High Byte**
  These bits hold the MSB of the 16-bit RTC TOP value.

17.6.11 COMPL – Compare register Low

The COMPH and COMPL register pair represent the 16-bit value, COMP. COMP is constantly compared with the counter value (CNT). A compare match will set COMPIF in the INTFLAGS register. Reading and writing 16-bit values requires special attention.

Due to synchronization between the RTC clock and system clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. Application software needs to check that the SYNCBUSY flag in the “STATUS - Status register” on page 227 is cleared before writing to this register.

If the COMP value is higher than the PER value, no RTC compare match interrupt requests or events will ever be generated.

Bit	7	6	5	4	3	2	1	0
+0x0C	COMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – COMP[7:0]: Compare Value Low Byte**
  These bits hold the LSB of the 16-bit RTC compare value.

17.6.12 COMPH – Compare register High

Bit	7	6	5	4	3	2	1	0
+0x0D	COMP[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – COMP[15:8]: Compare Value High Byte**
  These bits hold the MSB of the 16-bit RTC compare value.

17.7 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x00	CTRL	–	–	–	–	CORREN	PRESCALER[2:0]			227
0x01	STATUS	–	–	–	–	–	–	–	SYNCBUSY	227
0x02	INTCTRL	–	–	–	–	COMPINTLVL[1:0]		OVFINTLVL[1:0]		228
0x03	INTFLAGS	–	–	–	–	–	–	COMP	OVFIF	228
0X04	TEMP	TEMP[7:0]								228
0X05	Reserved	–	–	–	–	–	–	–	–	
0X06	CALIB	SIGN	ERROR[6:0]							229
0X07	Reserved	–	–	–	–	–	–	–	–	
0x08	CNTL	CNT[7:0]								229
0x09	CNTH	CNT[15:8]								229
0x0A	PERL	PER[7:0]								230
0x0B	PERH	PER[15:8]								230
0x0C	COMPL	COMP[7:0]								230
0x0D	COMPH	COMP[15:8]								231

17.8 Interrupt vector summary

Table 17-2. RTC interrupt vectors and their word offset.

Offset	Source	Interrupt description
0x00	OVF_vect	Real-time counter overflow interrupt vector
0x02	COMP_vect	Real-time counter compare match interrupt vector

18. TWI – Two-Wire Interface

18.1 Features

- Bidirectional, two-wire communication interface
  - Phillips I^2C compatible
  - System Management Bus (SMBus) compatible
- Bus master and slave operation supported
  - Slave operation
  - Single bus master operation
  - Bus master in multi-master bus environment
  - Multi-master arbitration
  - Bridge mode with independent and simultaneous master and slave operation
- Flexible slave address match functions
  - 7-bit and general call address recognition in hardware
  - 10-bit addressing supported
  - Address mask register for dual address match or address range masking
  - Optional software address recognition for unlimited number of addresses
- Slave can operate in all sleep modes, including power-down
- Slave address match can wake device from all sleep modes
- Up to 1MHz bus frequency support
- Slew-rate limited output drivers
- Input filter for bus noise and spike suppression
- Support arbitration between start/repeated start and data bit (SMBus)
- Slave arbitration allows support for address resolve protocol (ARP) (SMBus)
- Supports SMBUS Layer 1 timeouts
- Configurable timeout values
- Independent timeout counters in master and slave (Bridge mode support)

18.2 Overview

The two-wire interface (TWI) is a bidirectional, two-wire communication interface. It is I^2C and System Management Bus (SMBus) compatible. The only external hardware needed to implement the bus is one pull-up resistor on each bus line.

A device connected to the bus must act as a master or a slave. The master initiates a data transaction by addressing a slave on the bus and telling whether it wants to transmit or receive data. One bus can have many slaves and one or several masters that can take control of the bus. An arbitration process handles priority if more than one master tries to transmit data at the same time. Mechanisms for resolving bus contention are inherent in the protocol.

The TWI module supports master and slave functionality. The master and slave functionality are separated from each other, and can be enabled and configured separately. The master module supports multi-master bus operation and arbitration. It contains the baud rate generator. All 100kHz, 400kHz and 1MHz bus frequencies are supported. Quick command and smart mode can be enabled to auto-trigger operations and reduce software complexity.

The slave module implements 7-bit address match and general address call recognition in hardware. 10-bit addressing is also supported. A dedicated address mask register can act as a second address match register or as a register for address range masking. The slave continues to operate in all sleep modes, including power-down mode. This enables the slave to wake up the device from all sleep modes on TWI address match. It is possible to disable the address matching to let this be handled in software instead.

The TWI module will detect START and STOP conditions, bus collisions, and bus errors. Arbitration lost, errors, collision, and clock hold on the bus are also detected and indicated in separate status flags available in both master and slave modes.

It is possible to disable the TWI drivers in the device, and enable a four-wire digital interface for connecting to an external TWI bus driver. This can be used for applications where the device operates from a different V_{CC} voltage than used by the TWI bus.

It is also possible to enable the bridge mode. In this case, the slave I/O pins are selected from an alternative port, enabling independent and simultaneous master and slave operation.

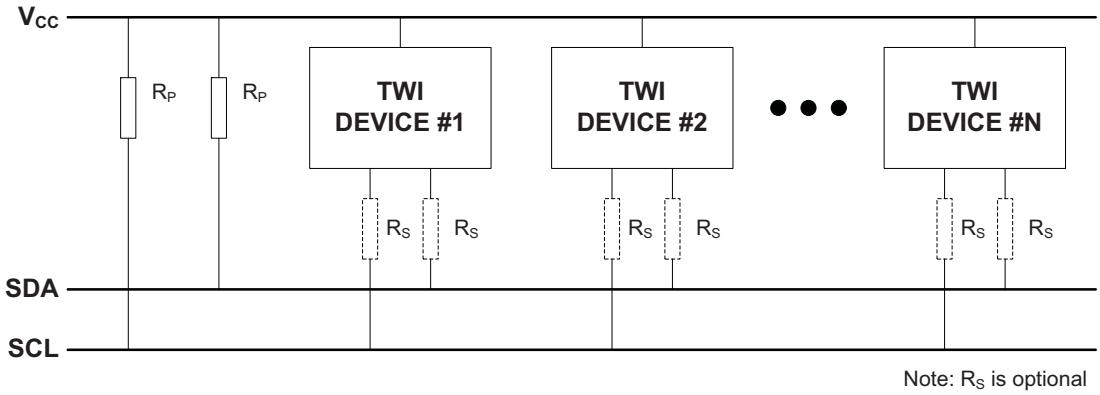
18.3 General TWI bus concepts

The TWI provides a simple, bidirectional, two-wire communication bus consisting of a serial clock line (SCL) and a serial data line (SDA). The two lines are open-collector lines (wired-AND), and pull-up resistors (Rp) are the only external components needed to drive the bus. The pull-up resistors provide a high level on the lines when none of the connected devices are driving the bus.

The TWI bus is a simple and efficient method of interconnecting multiple devices on a serial bus. A device connected to the bus can be a master or slave, where the master controls the bus and all communication.

Figure 18-1 on page 234 illustrates the TWI bus topology.

Figure 18-1. TWI bus topology.



An unique address is assigned to all slave devices connected to the bus, and the master will use this to address a slave and initiate a data transaction.

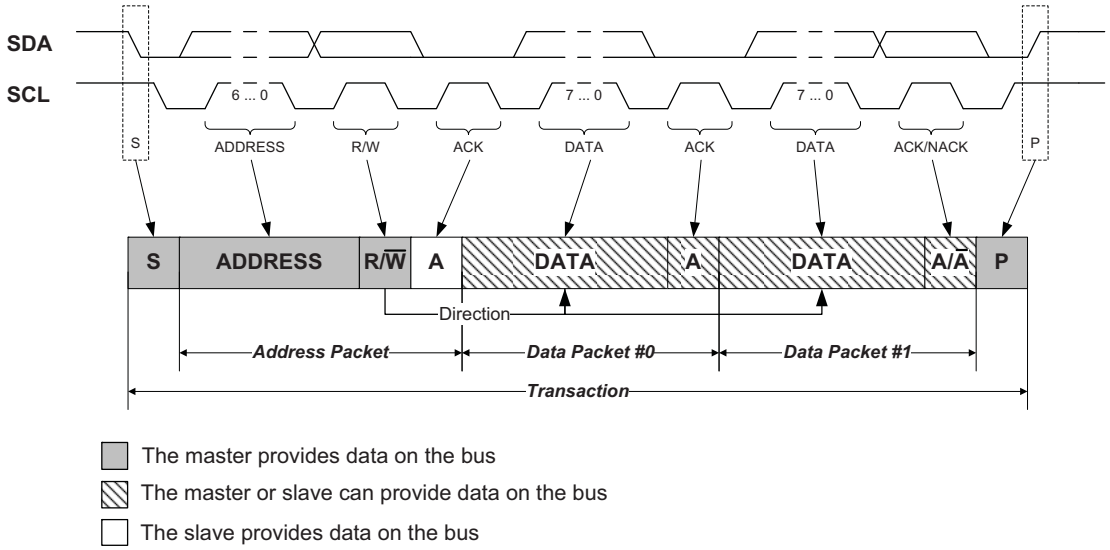
Several masters can be connected to the same bus, called a multi-master environment. An arbitration mechanism is provided for resolving bus ownership among masters, since only one master device may own the bus at any given time.

A device can contain both master and slave logic, and can emulate multiple slave devices by responding to more than one address.

A master indicates the start of a transaction by issuing a START condition (S) on the bus. An address packet with a slave address (ADDRESS) and an indication whether the master wishes to read or write data (R/$\overline{W}$) are then sent. After all data packets (DATA) are transferred, the master issues a STOP condition (P) on the bus to end the transaction. The receiver must acknowledge (A) or not-acknowledge ($\overline{A}$) each byte received.

Figure 18-2 on page 235 shows a TWI transaction.

Figure 18-2. Basic TWI transaction diagram topology for a 7-bit address bus.



The master provides the clock signal for the transaction, but a device connected to the bus is allowed to stretch the low-level period of the clock to decrease the clock speed.

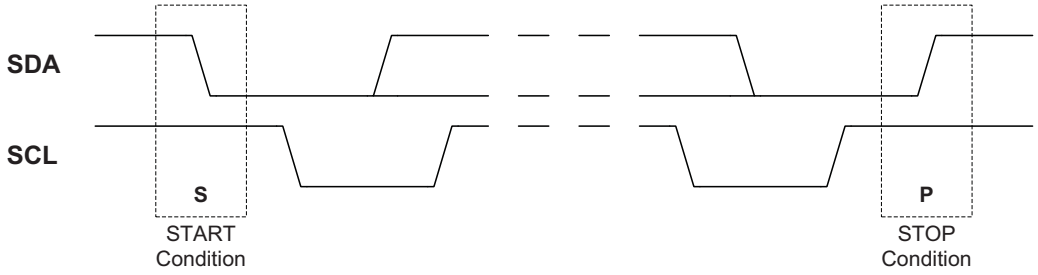
18.3.1 Electrical characteristics

The TWI module in XMEGA devices follows the electrical specifications and timing of I^2C bus and SMBus. These specifications are not 100% compliant, and so to ensure correct behavior, the inactive bus timeout period should be set in TWI master mode. Refer to "TWI master operation" on page 240 for more details.

18.3.2 START and STOP conditions

Two unique bus conditions are used for marking the beginning (START) and end (STOP) of a transaction. The master issues a START condition (S) by indicating a high-to-low transition on the SDA line while the SCL line is kept high. The master completes the transaction by issuing a STOP condition (P), indicated by a low-to-high transition on the SDA line while SCL line is kept high.

Figure 18-3. START and STOP conditions.

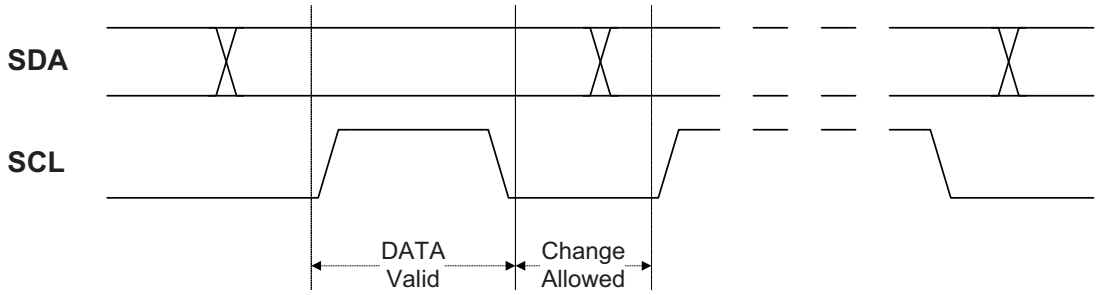


Multiple START conditions can be issued during a single transaction. A START condition that is not directly following a STOP condition is called a repeated START condition (Sr).

18.3.3 Bit transfer

As illustrated by Figure 18-4, a bit transferred on the SDA line must be stable for the entire high period of the SCL line. Consequently the SDA value can only be changed during the low period of the clock. This is ensured in hardware by the TWI module.

Figure 18-4. Data validity.



Combining bit transfers results in the formation of address and data packets. These packets consist of eight data bits (one byte) with the most-significant bit transferred first, plus a single-bit not-acknowledge (NACK) or acknowledge (ACK) response. The addressed device signals ACK by pulling the SCL line low during the ninth clock cycle, and signals NACK by leaving the line SCL high.

18.3.4 Address packet

After the START condition, a 7-bit address followed by a read/write (R/$\overline{W}$) bit is sent. This is always transmitted by the master. A slave recognizing its address will ACK the address by pulling the data line low for the next SCL cycle, while all other slaves should keep the TWI lines released and wait for the next START and address. The address, R/$\overline{W}$ bit, and acknowledge bit combined is the address packet. Only one address packet for each START condition is allowed, also when 10-bit addressing is used.

The R/$\overline{W}$ bit specifies the direction of the transaction. If the R/$\overline{W}$ bit is low, it indicates a master write transaction, and the master will transmit its data after the slave has acknowledged its address. If the R/$\overline{W}$ bit is high, it indicates a master read transaction, and the slave will transmit its data after acknowledging its address.

18.3.5 Data packet

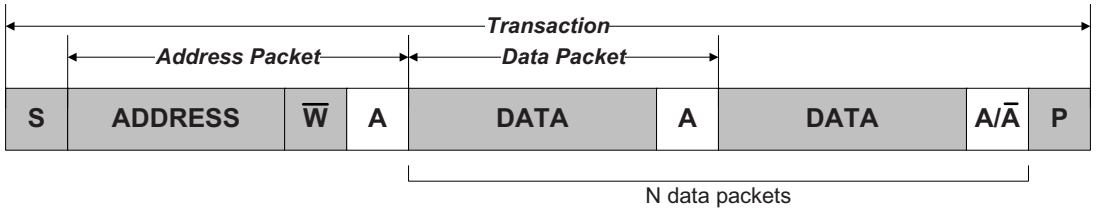
An address packet is followed by one or more data packets. All data packets are nine bits long, consisting of one data byte and an acknowledge bit. The direction bit in the previous address packet determines the direction in which the data are transferred.

18.3.6 Transaction

A transaction is the complete transfer from a START to a STOP condition, including any repeated START conditions in between. The TWI standard defines three fundamental transaction modes: Master write, master read, and a combined transaction.

Figure 18-5 on page 236 illustrates the master write transaction. The master initiates the transaction by issuing a START condition (S) followed by an address packet with the direction bit set to zero (ADDRESS+$\overline{W}$).

Figure 18-5. Master write transaction.

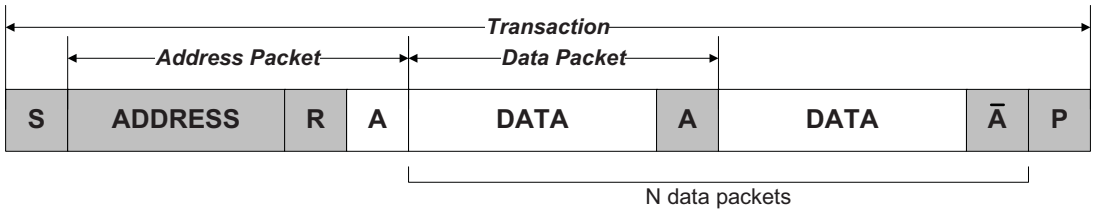


Assuming the slave acknowledges the address, the master can start transmitting data (DATA) and the slave will ACK or NACK (A/$\overline{A}$) each byte. If no data packets are to be transmitted, the master terminates the transaction by issuing a STOP condition (P) directly after the address packet. There are no limitations to the number of data packets that can be

transferred. If the slave signals a NACK to the data, the master must assume that the slave cannot receive any more data and terminate the transaction.

Figure 18-6 on page 237 illustrates the master read transaction. The master initiates the transaction by issuing a START condition followed by an address packet with the direction bit set to one (ADDRESS+R). The addressed slave must acknowledge the address for the master to be allowed to continue the transaction.

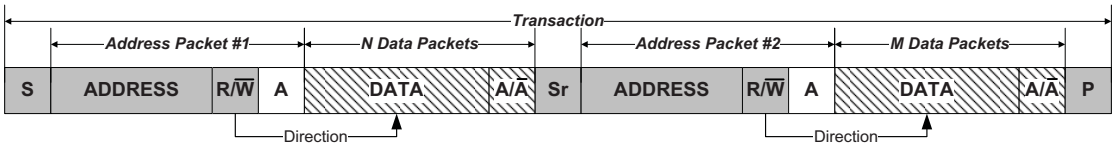
Figure 18-6. Master read transaction.



Assuming the slave acknowledges the address, the master can start receiving data from the slave. There are no limitations to the number of data packets that can be transferred. The slave transmits the data while the master signals ACK or NACK after each data byte. The master terminates the transfer with a NACK before issuing a STOP condition.

Figure 18-7 illustrates a combined transaction. A combined transaction consists of several read and write transactions separated by repeated START conditions (Sr).

Figure 18-7. Combined transaction.

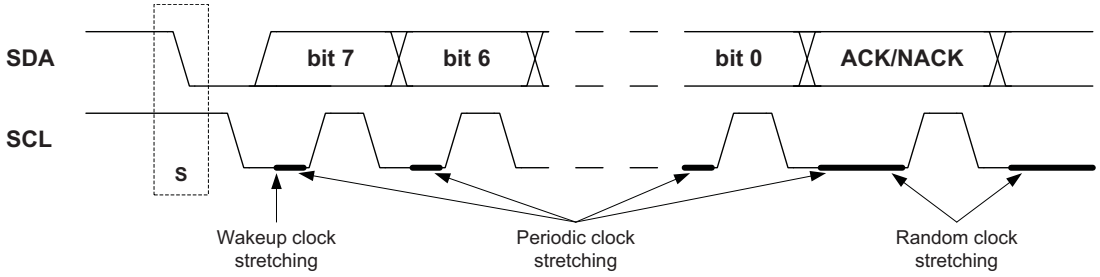


18.3.7 Clock and clock stretching

All devices connected to the bus are allowed to stretch the low period of the clock to slow down the overall clock frequency or to insert wait states while processing data. A device that needs to stretch the clock can do this by holding/forcing the SCL line low after it detects a low level on the line.

Three types of clock stretching can be defined, as shown in Figure 18-8.

Figure 18-8. Clock stretching [1].



Note: 1. Clock stretching is not supported by all I^2C slaves and masters.

If a slave device is in sleep mode and a START condition is detected, the clock stretching normally works during the wake-up period. For AVR XMEGA devices, the clock stretching will be either directly before or after the ACK/NACK bit, as AVR XMEGA devices do not need to wake up for transactions that are not addressed to it.

A slave device can slow down the bus frequency by stretching the clock periodically on a bit level. This allows the slave to run at a lower system clock frequency. However, the overall performance of the bus will be reduced accordingly. Both

the master and slave device can randomly stretch the clock on a byte level basis before and after the ACK/NACK bit. This provides time to process incoming or prepare outgoing data, or perform other time-critical tasks.

In the case where the slave is stretching the clock, the master will be forced into a wait state until the slave is ready, and vice versa.

18.3.8 Arbitration

A master can start a bus transaction only if it has detected that the bus is idle. As the TWI bus is a multi-master bus, it is possible that two devices may initiate a transaction at the same time. This results in multiple masters owning the bus simultaneously. This is solved using an arbitration scheme where the master loses control of the bus if it is not able to transmit a high level on the SDA line. The masters who lose arbitration must then wait until the bus becomes idle (i.e., wait for a STOP condition) before attempting to reacquire bus ownership. Slave devices are not involved in the arbitration procedure.

Figure 18-9. TWI arbitration.

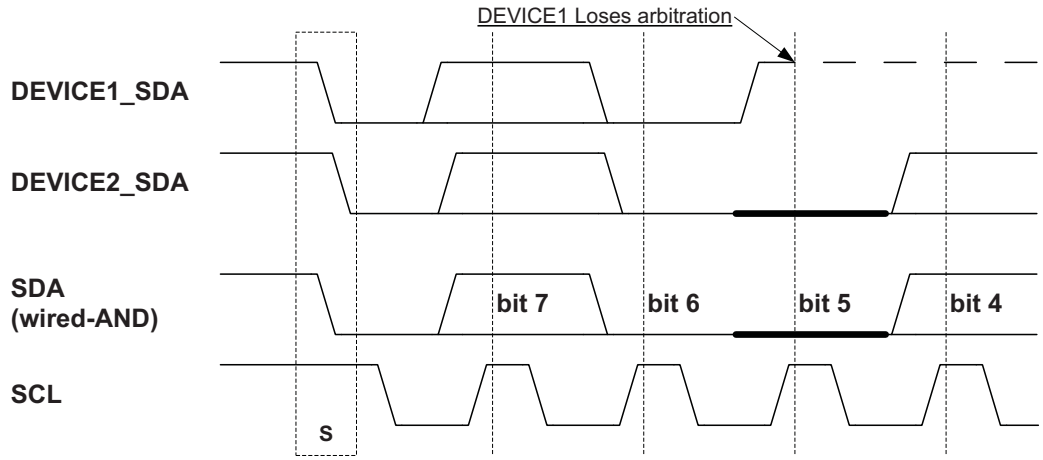


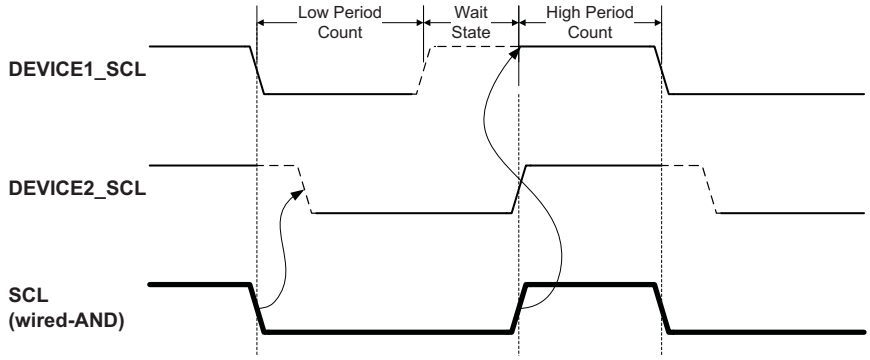
Figure 18-9 shows an example where two TWI masters are contending for bus ownership. Both devices are able to issue a START condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

Arbitration between a repeated START condition and a data bit, a STOP condition and a data bit, or a repeated START condition and a STOP condition are not allowed and will require special handling by software.

18.3.9 Synchronization

A clock synchronization algorithm is necessary for solving situations where more than one master is trying to control the SCL line at the same time. The algorithm is based on the same principles used for the clock stretching previously described. Figure 18-10 shows an example where two masters are competing for control over the bus clock. The SCL line is the wired-AND result of the two masters clock outputs.

Figure 18-10.Clock synchronization.



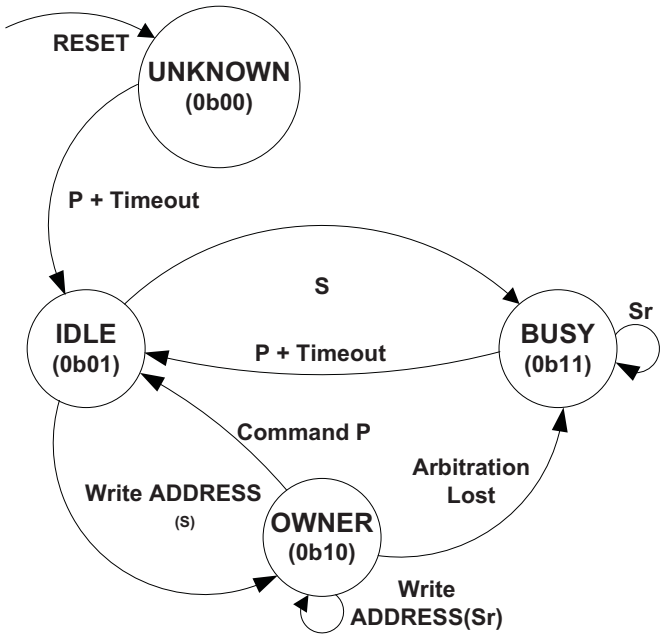
A high-to-low transition on the SCL line will force the line low for all masters on the bus, and they will start timing their low clock period. The timing length of the low clock period can vary among the masters. When a master (DEVICE1 in this case) has completed its low period, it releases the SCL line. However, the SCL line will not go high until all masters have released it. Consequently, the SCL line will be held low by the device with the longest low period (DEVICE2). Devices with shorter low periods must insert a wait state until the clock is released. All masters start their high period when the SCL line is released by all devices and has gone high. The device which first completes its high period (DEVICE1) forces the clock line low, and the procedure is then repeated. The result is that the device with the shortest clock period determines the high period, while the low period of the clock is determined by the device with the longest clock period.

18.4 TWI bus state logic

The bus state logic continuously monitors the activity on the TWI bus lines when the master is enabled. It continues to operate in all sleep modes, including power-down.

The bus state logic includes START and STOP condition detectors, collision detection, inactive bus timeout detection, and a bit counter. These are used to determine the bus state. Software can get the current bus state by reading the bus state bits in the master status register. The bus state can be unknown, idle, busy, or owner, and is determined according to the state diagram shown in Figure 18-11. The values of the bus state bits according to state are shown in binary in the figure.

Figure 18-11.Bus state, state diagram.



After a system reset and/or TWI master enable, the bus state is unknown. The bus state machine can be forced to enter idle by writing to the bus state bits accordingly. If no state is set by application software, the bus state will become idle when the first STOP condition is detected. If the master inactive bus timeout is enabled, the bus state will change to idle on the occurrence of a timeout. After a known bus state is established, only a system reset or disabling of the TWI master will set the state to unknown.

When the bus is idle, it is ready for a new transaction. If a START condition generated externally is detected, the bus becomes busy until a STOP condition is detected. The STOP condition will change the bus state to idle. If the master inactive bus timeout is enabled, the bus state will change from busy to idle on the occurrence of a timeout.

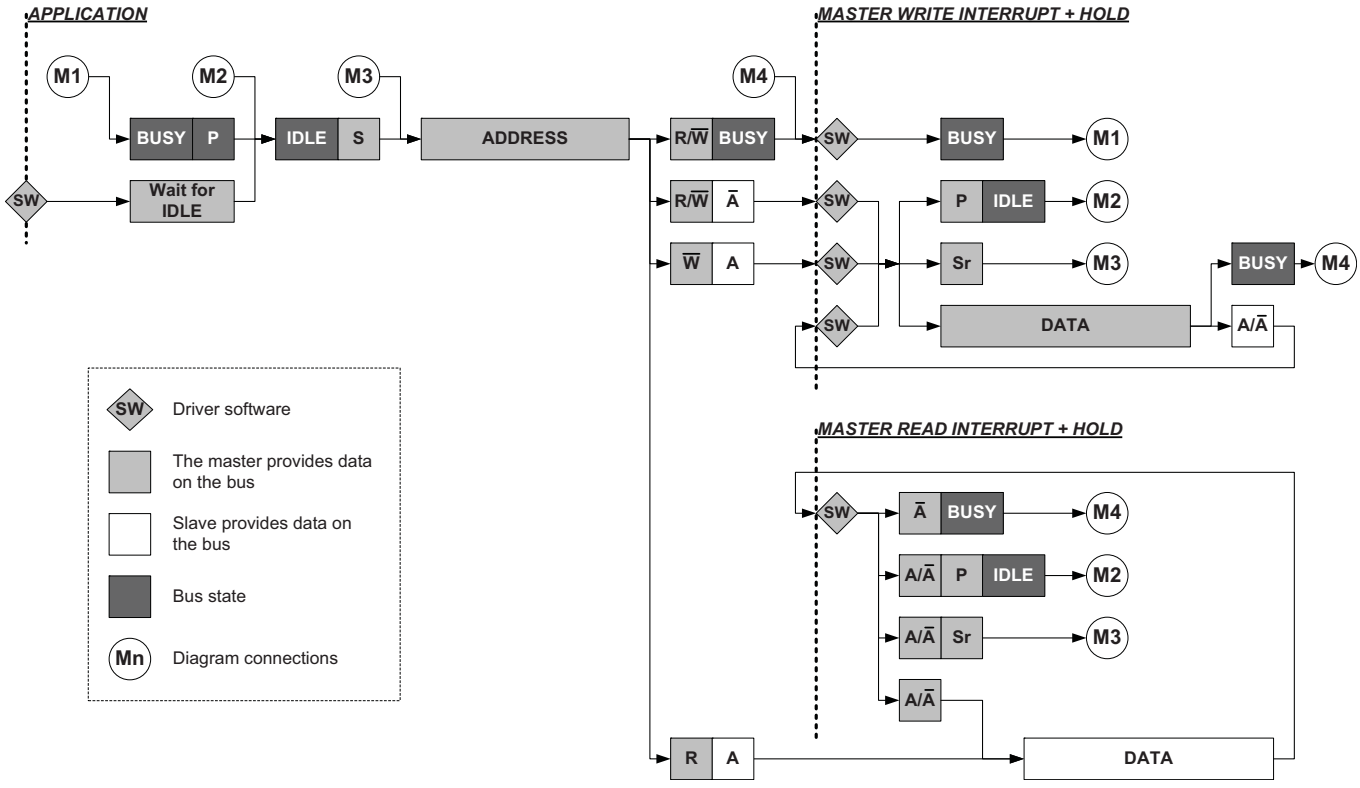
If a START condition is generated internally while in idle state, the owner state is entered. If the complete transaction was performed without interference, i.e., no collisions are detected, the master will issue a STOP condition and the bus state will change back to idle. If a collision is detected, the arbitration is assumed lost and the bus state becomes busy until a STOP condition is detected. A repeated START condition will only change the bus state if arbitration is lost during the issuing of the repeated START. Arbitration during repeated START can be lost only if the arbitration has been ongoing since the first START condition. This happens if two masters send the exact same ADDRESS+DATA before one of the masters issues a repeated START (Sr).

18.5 TWI master operation

The TWI master is byte-oriented, with an optional interrupt after each byte. There are separate interrupts for master write and master read. Interrupt flags can also be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, bus error, arbitration lost, clock hold, and bus state.

When an interrupt flag is set, the SCL line is forced low. This will give the master time to respond or handle any data, and will in most cases require software interaction. Figure 18-12 shows the TWI master operation. The diamond shaped symbols (SW) indicate where software interaction is required. Clearing the interrupt flags releases the SCL line.

Figure 18-12.TWI master operation.



The number of interrupts generated is kept to a minimum by automatic handling of most conditions. Quick command and smart mode can be enabled to auto-trigger operations and reduce software complexity.

18.5.1 Transmitting address packets

After issuing a START condition, the master starts performing a bus transaction when the master address register is written with the 7-bit slave address and direction bit. If the bus is busy, the TWI master will wait until the bus becomes idle before issuing the START condition.

Depending on arbitration and the R/$\overline{W}$ direction bit, one of four distinct cases (M1 to M4) arises following the address packet. The different cases must be handled in software.

18.5.1.1 Case M1: Arbitration lost or bus error during address packet

If arbitration is lost during the sending of the address packet, the master write interrupt flag and arbitration lost flag are both set. Serial data output to the SDA line is disabled, and the SCL line is released. The master is no longer allowed to perform any operation on the bus until the bus state has changed back to idle.

A bus error will behave in the same way as an arbitration lost condition, but the error flag is set in addition to the write interrupt and arbitration lost flags.

18.5.1.2 Case M2: Address packet transmit complete - Address not acknowledged by slave

If no slave device responds to the address, the master write interrupt flag and the master received acknowledge flag are set. The clock hold is active at this point, preventing further activity on the bus.

18.5.1.3 Case M3: Address packet transmit complete - Direction bit cleared

If the master receives an ACK from the slave, the master write interrupt flag is set and the master received acknowledge flag is cleared. The clock hold is active at this point, preventing further activity on the bus.

18.5.1.4 Case M4: Address packet transmit complete - Direction bit set

If the master receives an ACK from the slave, the master proceeds to receive the next byte of data from the slave. When the first data byte is received, the master read interrupt flag is set and the master received acknowledge flag is cleared. The clock hold is active at this point, preventing further activity on the bus.

18.5.2 Transmitting data packets

Assuming case M3 above, the master can start transmitting data by writing to the master data register. If the transfer was successful, the slave will signal with ACK. The master write interrupt flag is set, the master received acknowledge flag is cleared, and the master can prepare new data to send. During data transfer, the master is continuously monitoring the bus for collisions.

The received acknowledge flag must be checked by software for each data packet transmitted before the next data packet can be transferred. The master is not allowed to continue transmitting data if the slave signals a NACK.

If a collision is detected and the master loses arbitration during transfer, the arbitration lost flag is set.

18.5.3 Receiving data packets

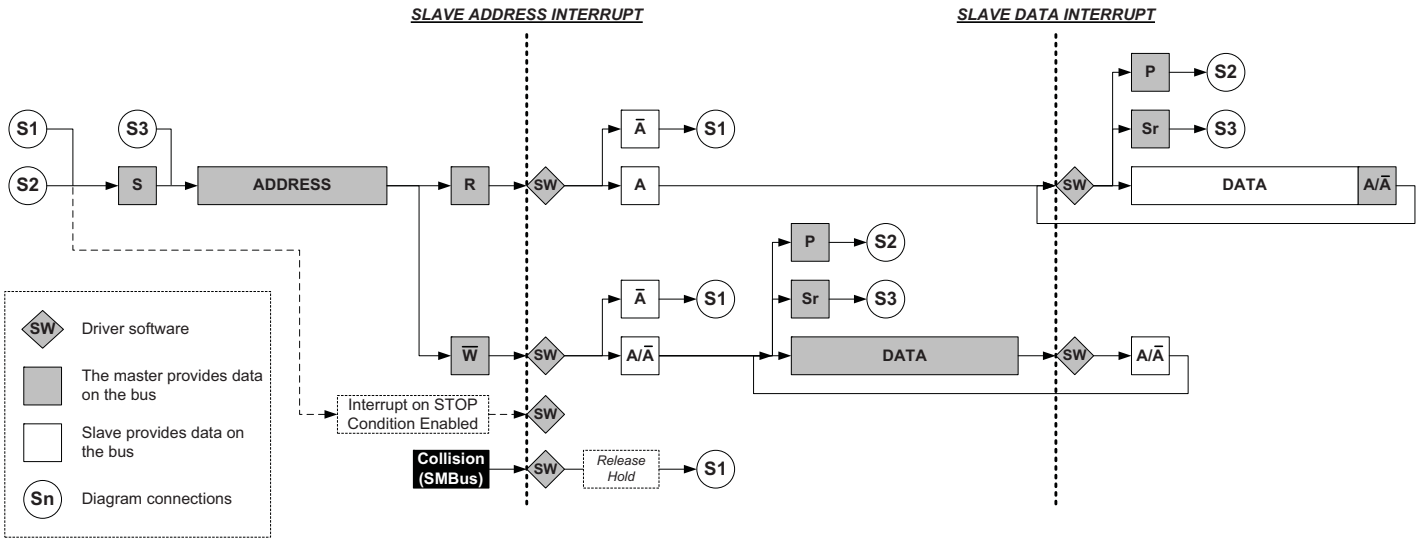
Assuming case M4 above, the master has already received one byte from the slave. The master read interrupt flag is set, and the master must prepare to receive new data. The master must respond to each byte with ACK or NACK. Indicating a NACK might not be successfully executed, as arbitration can be lost during the transmission. If a collision is detected, the master loses arbitration and the arbitration lost flag is set.

18.6 TWI slave operation

The TWI slave is byte-oriented with optional interrupts after each byte. There are separate slave data and address/stop interrupts. Interrupt flags can also be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, clock hold, collision, bus error, and read/write direction.

When an interrupt flag is set, the SCL line is forced low. This will give the slave time to respond or handle data, and will in most cases require software interaction. Figure 18-13. shows the TWI slave operation. The diamond shapes symbols (SW) indicate where software interaction is required.

Figure 18-13.TWI slave operation.



The number of interrupts generated is kept to a minimum by automatic handling of most conditions. Quick command can be enabled to auto-trigger operations and reduce software complexity.

Promiscuous mode can be enabled to allow the slave to respond to all received addresses.

18.6.1 Receiving address packets

When the TWI slave is properly configured, it will wait for a START condition to be detected. When this happens, the successive address byte will be received and checked by the address match logic, and the slave will ACK a correct address and store the address in the DATA register. If the received address is not a match, the slave will not acknowledge and store address, and will wait for a new START condition.

The slave address/stop interrupt flag is set when a START condition succeeded by a valid address byte is detected. A general call address will also set the interrupt flag.

A START condition immediately followed by a STOP condition is an illegal operation, and the bus error flag is set.

The R/$\overline{W}$ direction flag reflects the direction bit received with the address. This can be read by software to determine the type of operation currently in progress.

Depending on the R/$\overline{W}$ direction bit and bus condition, one of four distinct cases (S1 to S4) arises following the address packet. The different cases must be handled in software.

18.6.1.1 Case S1: Address packet accepted - Direction bit set

If the R/$\overline{W}$ direction flag is set, this indicates a master read operation. The SCL line is forced low by the slave, stretching the bus clock. If ACK is sent by the slave, the slave hardware will set the data interrupt flag indicating data is needed for transmit. Data, repeated START, or STOP can be received after this. If NACK is sent by the slave, the slave will wait for a new START condition and address match.

18.6.1.2 Case S2: Address packet accepted - Direction bit cleared

If the R/$\overline{W}$ direction flag is cleared, this indicates a master write operation. The SCL line is forced low, stretching the bus clock. If ACK is sent by the slave, the slave will wait for data to be received. Data, repeated START, or STOP can be received after this. If NACK is sent, the slave will wait for a new START condition and address match.

18.6.1.3 Case S3: Collision

If the slave is not able to send a high level or NACK, the collision flag is set, and it will disable the data and acknowledge output from the slave logic. The clock hold is released. A START or repeated START condition will be accepted.

18.6.1.4 Case S4: STOP condition received

When the STOP condition is received, the slave address/stop flag will be set, indicating that a STOP condition, and not an address match, occurred.

18.6.2 Receiving data packets

The slave will know when an address packet with R/$\overline{W}$ direction bit cleared has been successfully received. After acknowledging this, the slave must be ready to receive data. When a data packet is received, the data interrupt flag is set and the slave must indicate ACK or NACK. After indicating a NACK, the slave must expect a STOP or repeated START condition.

18.6.3 Transmitting data packets

The slave will know when an address packet with R/$\overline{W}$ direction bit set has been successfully received. It can then start sending data by writing to the slave data register. When a data packet transmission is completed, the data interrupt flag is set. If the master indicates NACK, the slave must stop transmitting data and expect a STOP or repeated START condition.

18.7 Enabling external driver interface

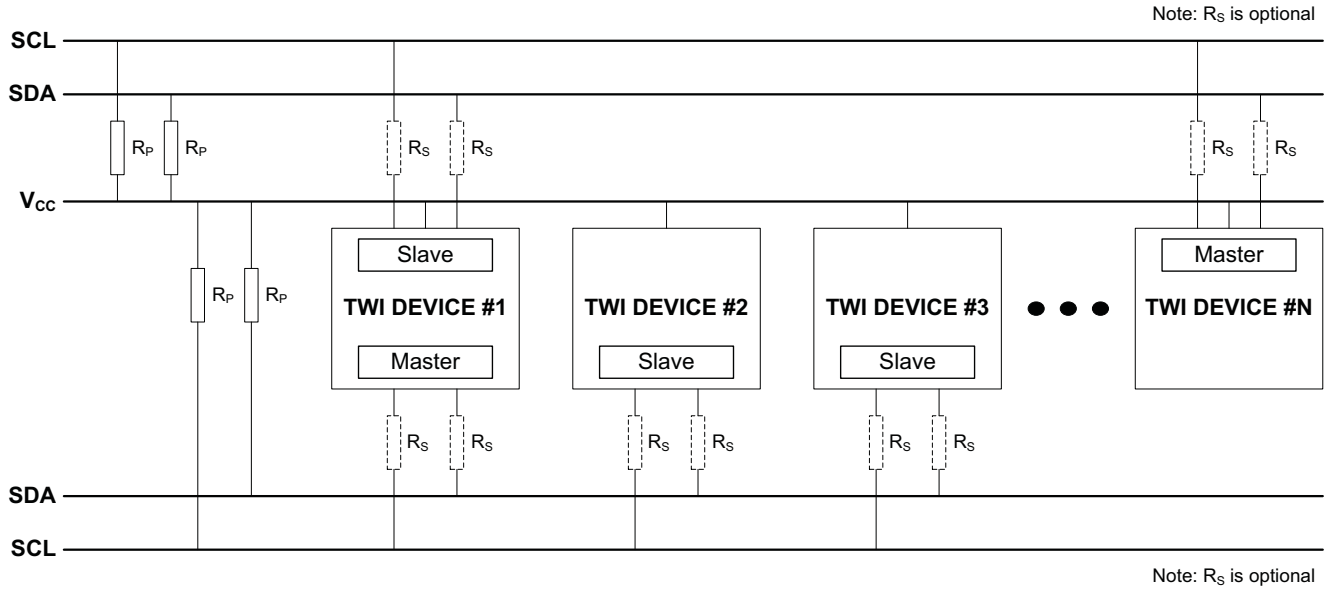
An external driver interface can be enabled. When this is done, the internal TWI drivers with input filtering and slew rate control are bypassed. The normal I/O pin function is used, and the direction must be configured by the user software. When this mode is enabled, an external TWI compliant tri-state driver is needed for connecting to a TWI bus.

By default, port pins 0 (Pn0) and 1 (Pn1) are used for SDA and SCL. The external driver interface uses port pins 0 to 3 for the SDA_IN, SCL_IN, SDA_OUT, and SCL_OUT signals.

18.8 Bridge mode

When enabling the bridge mode, both master and slave can be active at the same time, each with its specific IO pins. Refer to the device datasheet to see which actual I/O port is used as alternative port selection for the slave in bridge mode.

Figure 18-14.TWI bus topology example when one device is enabled in bridge mode.



18.9 SMBUS L1 Compliance

The Industry Standard SMBus-2.0 specifies three timeouts for Layer 1 Compliance.

- Ttimeout – Continuous SCL low for 25 ms
- Tlowsext – Cumulative slave SCL extend from START to STOP for 25ms
- Tlowmext – Cumulative master SCL extend from START- ACK or ACK-ACK or ACK – STOP for 10ms

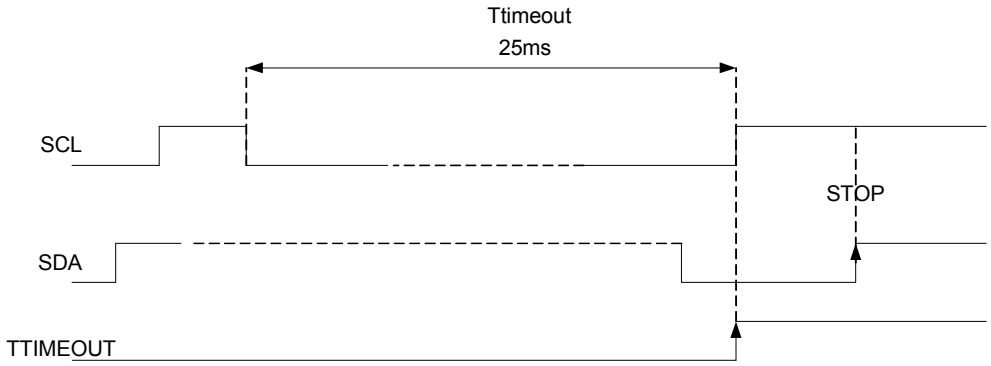
All these timeouts are supported in the TWI module.

18.9.1 Overview

18.9.1.1 TTIMEOUT Specification

The TTIMEOUT,MIN (25ms) parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. It is highly recommended that a slave device release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than TTIMEOUT,MIN. Devices that have detected this condition must reset their communication and be able to receive a new START condition in no later than TIMEOUT,MAX (35ms).

Table 18-1. Ttimeout condition



18.9.1.2 Tlowsext Specification

This is the cumulative time the slave device is allowed to extend the SCL from START to STOP. The value for this timeout is 25ms.

18.9.1.3 Tlowmext Specification

This is the cumulative time a master device is allowed to extend its clock cycles within one byte in a message as measured from

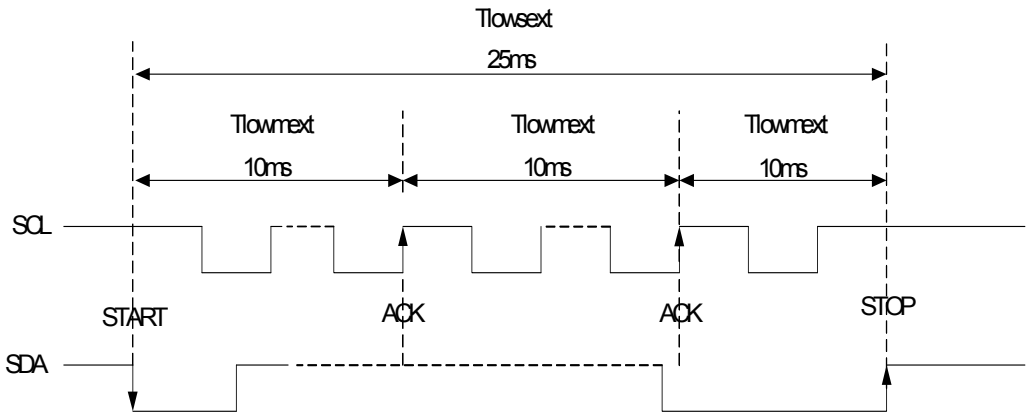
START to ACK

ACK to ACK

ACK to STOP

The value for this timeout is 10ms.

Figure 18-15.Tlowsext/Tlowmext condition



18.9.2 Operation

For the operation of SMBUS timeout counters, the 2MHZ internal oscillator should be in the oscillator control register, “CTRL – Oscillator Control register” on page 104.

18.9.2.1 Ttimeout implementation

When this Timeout occurs in slave mode, slave resets the communication and lines are released by hardware. The Slave is ready to receive a new start pulse. When this Timeout occurs in master mode, stop condition is sent by hardware. The master is ready to start a new transaction on the bus. If TWI is configured as master, but lost arbitration on the bus then TTIMEOUT counter is disabled.

18.9.2.2 Tlowsext implementation

When this timeout occurs the master sends a STOP pulse immediately or at the end of current byte in progress, depending on whether the master is in control of SDA. The slave resets its communication on receiving a STOP pulse from master on timeout. If TWI is configured as master, but lost arbitration on the bus, the TLOWSEXT counter is disabled.

18.9.2.3 Tlowmext implementation

When this timeout occurs the master sends a STOP pulse immediately or at the end of current byte in progress, depending on whether the master is in control of SDA. The slave resets its communication on receiving a STOP pulse from master on timeout. If TWI is configured as master but lost arbitration on the bus, the TLOWMEXT counter is disabled.

18.9.2.4 Timeouts Summary

Table 18-2. Summary of SMBUS timeout implementation.

SMBUS L1 Timeouts	Master	Slave
Ttimeout	Stop condition sent by hardware. The master is ready to start a new transaction on the bus.	Slave resets the communication and releases the bus. Ready to receive a START condition in no later than TIMEOUT,MAX
Tlowsext	Stop condition sent by hardware. The master is ready to start a new transaction on the bus.	Slave resets the communication after detecting a stop pulse after timeout. Ready to receive a START condition.
Tlowmext	Stop condition sent by hardware. The master is ready to start a new transaction on the bus.	Slave resets the communication after detecting a stop pulse after timeout. Ready to receive a START condition.
Arbitration lost	All timeouts disabled	-

18.9.2.5 Timeout Enable and Status indication

- Separate Timeout enable for three timeouts (Ttimeout, Tlowsext, Tlowmext) in CTRLB register.
- Common interrupt enable for all timeouts in CTRLB register.
- Separate Status bit for three timeouts (Ttimeout, Tlowsext, Tlowmext) in “TOS – Timeout Status Register” on page 257.
- Timeout configuration registers (“TOCONF – Timeout configuration register” on page 258) to program positive and negative offsets in timeout value.
- All status registers are reset on timeout except bus error flag (indicates incomplete transaction on bus) and configuration registers are left unchanged.

18.10 Register description – TWI

18.10.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	BRIDGEEN	SFMPEN	SSDAHOLD[1:0]		FMPEN	SDAHOLD[1:0]		EDIEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – BRIDGEEN: Bridge Enable**
  Setting this bit to one enables the TWI Bridge Mode.
- **Bit 6 – SFMPEN: Slave Fast Mode Plus Enable**
  Setting this bit to one enables the slave 1MHz bus speed operation. This bit setting is ignored if bridge mode is disabled.
- **Bit 5:4 – SSDAHOLD[1:0]: Slave SDA Hold Time Enable**
  Setting these bits to one enables an internal hold time on slave SDA with respect to the negative edge of SCL, as defined by Table 18-3 on page 247. These bits settings are ignored if bridge mode is disabled.
- **Bit 3 – FMPEN: FM Plus Enable**
  Setting this bit to one enables the 1MHz bus speed operation. By default, the setting applies to the master/slave node. If the bridge mode is enabled, the setting applies to the master node only.
- **Bit 2:1 – SDAHOLD[1:0]: SDA Hold Time Enable**
  Setting these bits to one enables an internal hold time on SDA with respect to the negative edge of SCL. By default, the setting applies to the master/slave node. If the bridge mode is enabled, the setting applies to the master node only.

Table 18-3. SDA hold time.

SDAHOLD[1:0]	Group configuration	Description
00	OFF	SDA hold time off
01	50NS	Typical 50ns hold time
10	300NS	Typical 300ns hold time
11	400NS	Typical 400ns hold time

- **Bit 0 – EDIEN: External Driver Interface Enable**
  Setting this bit enables the use of the external driver interface, and clearing this bit enables normal two-wire mode. See Table 18-4 on page 247 for details. If bridge mode is enabled, this bit setting applies to both master and slave nodes.

Table 18-4. External driver interface enable.

EDIEN	Mode	Comment
0	Normal TWI	Two-pin interface, slew rate control, and input filter
1	External driver interface	Four-pin interface, standard I/O, no slew rate control, and no input filter

18.11 Register description – TWI master

18.11.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	INTLVL[1:0]		RIEN	WIEN	ENABLE	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – INTLVL[1:0]: Interrupt Level**
  These bits select the interrupt level for the TWI master interrupt, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.
- **Bit 5 – RIEN: Read Interrupt Enable**
  Setting the read interrupt enable (RIEN) bit enables the read interrupt when the read interrupt flag (RIF) in the STATUS register is set. In addition the INTLVL bits must be nonzero for TWI master interrupts to be generated.
- **Bit 4 – WIEN: Write Interrupt Enable**
  Setting the write interrupt enable (WIEN) bit enables the write interrupt when the write interrupt flag (WIF) in the STATUS register is set. In addition the INTLVL bits must be nonzero for TWI master interrupts to be generated.
- **Bit 3 – ENABLE: Enable TWI Master**
  Setting the enable TWI master (ENABLE) bit enables the TWI master.
- **Bit 2:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

18.11.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	TOIE	TMEXTEN	TSEXTEN	TTOUTEN	TIMEOUT[1:0]		QCEN	SMEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – TOIE: Timeout Interrupt Enable**
  Setting this bit to one enables interrupt on all master timeout interrupt flags (TTOUTMIF, TSEXTIF, TMEXTIF). A TWI master timeout interrupt would be generated only if any one of the master timeout interrupt flags (TTOUTMIF/TSEXTIF/TMEXTIF), TOIE and Global Interrupt Enable are all set to one, and master interrupt level (INTLVL[1:0]) is not equal to zero.
- **Bit 6 – TMEXTEN: Tlowmext Enable**
  When tlowmext (TMEXTEN) is enabled, the master monitors the bus for tlowmext condition and the corresponding interrupt flag (TMEXTIF) is set immediately after the tlowmext condition occurs.
- **Bit 5 –TSEXTEN: Tlowsext Enable**
  When tlowsext (TSEXTEN) is enabled, the master monitors the bus for tlowsext condition and the corresponding interrupt flag (TSEXTIF) is set immediately after the tlowsext condition occurs.
- **Bit 4 – TTOUTEN : Ttimeout Enable**
  When ttimeout (TTOUTEN) is enabled, the master monitors the bus for ttimeout condition and the corresponding interrupt flag (TTOUTMIF) is set immediately after the ttimeout condition occurs.
- **Bit 3:2 – TIMEOUT[1:0]: Inactive Bus Timeout**
  Setting the inactive bus timeout (TIMEOUT) bits to a nonzero value will enable the inactive bus timeout supervisor. If the bus is inactive for longer than the TIMEOUT setting, the bus state logic will enter the idle state.

Table 18-5 on page 249 lists the timeout settings.

Table 18-5. TWI master inactive bus timeout settings.

TIMEOUT[1:0]	Group configuration	Description
00	DISABLED	Disabled, normally used for I^2C
01	50US	50µs, normally used for SMBus at 100kHz
10	100US	100µs
11	200US	200µs

- **Bit 1 – QCEN: Quick Command Enable**
  When quick command is enabled, the corresponding interrupt flag is set immediately after the slave acknowledges the address (read or write interrupt). At this point, software can issue either a STOP or a repeated START condition.
- **Bit 0 – SMEN: Smart Mode Enable**
  Setting this bit enables smart mode. When smart mode is enabled, the acknowledge action, as set by the ACKACT bit in the CTRLC register, is sent immediately after reading the DATA register.

18.11.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	ACKACT	CMD[1:0]	
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – ACKACT: Acknowledge Action**
  This bit defines the master's acknowledge behavior in master read mode. The acknowledge action is executed when a command is written to the CMD bits. If SMEN in the CTRLB register is set, the acknowledge action is performed when the DATA register is read.

Table 18-6 lists the acknowledge actions.

Table 18-6. ACKACT bit description.

ACKACT	Action
0	Send ACK
1	Send NACK

- **Bit 1:0 – CMD[1:0]: Command**
  Writing the command (CMD) bits triggers a master operation as defined by Table 18-7 on page 250. The CMD bits are strobe bits, and always read as zero. The acknowledge action is only valid in master read mode (R). In master write mode ($\overline{W}$), a command will only result in a repeated START or STOP condition. The ACKACT bit and the CMD bits can be written at the same time, and then the acknowledge action will be updated before the command is triggered.

Table 18-7. CMD bits description.

CMD[1:0]	Group configuration	MODE	Operation
00	NOACT	X	Reserved
01	START	X	Execute acknowledge action succeeded by repeated START condition
10	BYTEREC	$\overline{W}$	No operation
		R	Execute acknowledge action succeeded by a byte receive
11	STOP	X	Execute acknowledge action succeeded by issuing a STOP condition

Writing a command to the CMD bits will clear the master interrupt flags and the CLKHOLD flag.

18.11.4 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x03	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]	
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – RIF: Read Interrupt Flag**
  This flag is set when a byte is successfully received in master read mode; i.e., no arbitration was lost or bus error occurred during the operation. Writing a one to this bit location will clear RIF. When this flag is set, the master forces the SCL line low, stretching the TWI clock period. Clearing the interrupt flags will release the SCL line.
  This flag is also cleared automatically when:
  - Writing to the ADDR register
  - Writing to the DATA register
  - Reading the DATA register
  - Writing a valid command to the CMD bits in the CTRLC register
- **Bit 6 – WIF: Write Interrupt Flag**
  This flag is set when a byte is transmitted in master write mode. The flag is set regardless of the occurrence of a bus error or an arbitration lost condition. WIF is also set if arbitration is lost during sending of a NACK in master read mode, and if issuing a START condition when the bus state is unknown. Writing a one to this bit location will clear WIF. When this flag is set, the master forces the SCL line low, stretching the TWI clock period. Clearing the interrupt flags will release the SCL line.
  The flag is also cleared automatically for the same conditions as RIF.
- **Bit 5 – CLKHOLD: Clock Hold**
  This flag is set when the master is holding the SCL line low. This is a status flag and a read-only flag that is set when RIF or WIF is set. Clearing the interrupt flags and releasing the SCL line will indirectly clear this flag.
  The flag is also cleared automatically for the same conditions as RIF.
- **Bit 4 – RXACK: Received Acknowledge**
  This flag contains the most recently received acknowledge bit from the slave. This is a read-only flag. When read as zero, the most recent acknowledge bit from the slave was ACK, and when read as one the most recent acknowledge bit was NACK.
- **Bit 3 – ARBLOST: Arbitration Lost**
  This flag is set if arbitration is lost while transmitting a high data bit or a NACK bit, or while issuing a START or repeated START condition on the bus. Writing a one to this bit location will clear ARBLOST.
  Writing the ADDR register will automatically clear ARBLOST.
- **Bit 2 – BUSERR: Bus Error**
  This flag is set if an illegal bus condition has occurred. An illegal bus condition occurs if a repeated START or a STOP condition is detected, and the number of received or transmitted bits from the previous START condition is not a multiple of nine. Writing a one to this bit location will clear BUSERR.

Writing the ADDR register will automatically clear BUSERR.

- **Bit 1:0 – BUSSTATE[1:0]: Bus State**
  These bits indicate the current TWI bus state as defined in Table 18-8. The change of bus state is dependent on bus activity. Refer to the "TWI bus state logic" on page 239.

Table 18-8. TWI master bus state.

BUSSTATE[1:0]	Group configuration	Description
00	UNKNOWN	Unknown bus state
01	IDLE	Idle bus state
10	OWNER	Owner bus state
11	BUSY	Busy bus state

Writing 01 to the BUSSTATE bits forces the bus state logic into the idle state. The bus state logic cannot be forced into any other state. When the master is disabled, and after reset, the bus state logic is disabled and the bus state is unknown.

18.11.5 BAUD – Baud Rate register

Bit	7	6	5	4	3	2	1	0
+0x04	BAUD[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

The baud rate (BAUD) register defines the relation between the system clock and the TWI bus clock (SCL) frequency. The frequency relation can be expressed by using the following equation:

$$f_{TWI} = \frac{f_{sys}}{2(5 + (BAUD))}[\text{Hz}] \quad [1]$$

The BAUD register must be set to a value that results in a TWI bus clock frequency (f_{TWI}) equal or less than 100kHz or 400kHz, depending on which standard the application should comply with. The following equation [2] expresses equation [1] solved for the BAUD value:

$$BAUD = \frac{f_{sys}}{2f_{TWI}} - 5 \quad [2]$$

The BAUD register should be written only while the master is disabled.

18.11.6 ADDR – Address register

Bit	7	6	5	4	3	2	1	0
+0x05	ADDR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

When the address (ADDR) register is written with a slave address and the R/$\overline{W}$ bit while the bus is idle, a START condition is issued and the 7-bit slave address and the R/$\overline{W}$ bit are transmitted on the bus. If the bus is already owned

when ADDR is written, a repeated START is issued. If the previous transaction was a master read and no acknowledge is sent yet, the acknowledge action is sent before the repeated START condition.

After completing the operation and the acknowledge bit from the slave is received, the SCL line is forced low if arbitration was not lost. WIF is set.

If the bus state is unknown when ADDR is written, WIF is set and BUSERR is set.

All TWI master flags are automatically cleared when ADDR is written. This includes BUSERR, ARBLOST, RIF, and WIF. The master ADDR can be read at any time without interfering with ongoing bus activity.

18.11.7 DATA – Data register

Bit	7	6	5	4	3	2	1	0
+0x06	DATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

The data (DATA) register is used when transmitting and receiving data. During data transfer, data are shifted from/to the DATA register and to/from the bus. This implies that the DATA register cannot be accessed during byte transfers, and this is prevented by hardware. The DATA register can only be accessed when the SCL line is held low by the master; i.e., when CLKHOLD is set.

In master write mode, writing the DATA register will trigger a data byte transfer followed by the master receiving the acknowledge bit from the slave. WIF and CLKHOLD are set.

In master read mode, RIF and CLKHOLD are set when one byte is received in the DATA register. If smart mode is enabled, reading the DATA register will trigger the bus operation as set by the ACKACT bit. If a bus error occurs during reception, WIF and BUSERR are set instead of RIF.

Accessing the DATA register will clear the master interrupt flags and CLKHOLD.

18.12 Register description – TWI slave

18.12.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	INTLVL[1:0]		DIEN	APIEN	ENABLE	PIEN	PMEN	SMEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – INTLVL[1:0]: Interrupt Level**
  These bits select the interrupt level for the TWI master interrupt, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.
- **Bit 5 – DIEN: Data Interrupt Enable**
  Setting the data interrupt enable (DIEN) bit enables the data interrupt when the data interrupt flag (DIF) in the STATUS register is set. The INTLVL bits must be nonzero for the interrupt to be generated.
- **Bit 4 – APIEN: Address/Stop Interrupt Enable**
  Setting the address/stop interrupt enable (APIEN) bit enables the address/stop interrupt when the address/stop interrupt flag (APIF) in the STATUS register is set. The INTLVL bits must be nonzero for interrupt to be generated.
- **Bit 3 – ENABLE: Enable TWI Slave**
  Setting this bit enables the TWI slave.
- **Bit 2 – PIEN: Stop Interrupt Enable**
  Setting the this bit will cause APIF in the STATUS register to be set when a STOP condition is detected.
- **Bit 1 – PMEN: Promiscuous Mode Enable**
  By setting the this bit, the slave address match logic responds to all received addresses. If this bit is cleared, the address match logic uses the ADDR register to determine which address to recognize as its own address.
- **Bit 0 – SMEN: Smart Mode Enable**
  This bit enables smart mode. When Smart mode is enabled, the acknowledge action, as set by the ACKACT bit in the CTRLB register, is sent immediately after reading the DATA register.

18.12.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	TOIE	–	–	TTOUTEN	–	ACKACT	CMD[1:0]	
Read/Write	R/W	R	R	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – TOIE: Timeout Interrupt Enable**
  Setting this bit to one enables interrupt on slave timeout interrupt flag (TTOUTSIF). A TWI slave timeout interrupt would be generated if TTOUTSIF, TOIE and Global Interrupt Enable are all set to one, and slave interrupt level (INTLVL[1:0]) is not equal to zero.
- **Bit 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – TTOUTEN: Ttimeout Enable**
  When ttimeout (TTOUTEN) is enabled, the slave monitors the bus for ttimeout condition and the corresponding interrupt flag (TTOUTSIF) is set immediately after the ttimeout condition occurs.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

- **Bit 2 – ACKACT: Acknowledge Action**
  This bit defines the slave's acknowledge behavior after an address or data byte is received from the master. The acknowledge action is executed when a command is written to the CMD bits. If the SMEN bit in the CTRLA register is set, the acknowledge action is performed when the DATA register is read.
  Table 18-9 on page 254 lists the acknowledge actions.

Table 18-9. TWI slave acknowledge actions.

ACKACT	Action
0	Send ACK
1	Send NACK

- **Bit 1:0 – CMD[1:0]: Command**
  Writing these bits trigger the slave operation as defined by Table 18-10. The CMD bits are strobe bits and always read as zero. The operation is dependent on the slave interrupt flags, DIF and APIF. The acknowledge action is only executed when the slave receives data bytes or address byte from the master.

Table 18-10. TWI slave command.

CMD[1:0]	Group configuration	DIR	Operation
00	NOACT	X	No action
01		X	Reserved
10	COMPLETE	Used to complete transaction	
		0	Execute acknowledge action succeeded by waiting for any START (S/Sr) condition
		1	Wait for any START (S/Sr) condition
11	RESPONSE	**Used in response to an address byte (APIF is set)**	
		0	Execute acknowledge action succeeded by reception of next byte
		1	Execute acknowledge action succeeded by DIF being set
		Used in response to a data byte (DIF is set)	
		0	Execute acknowledge action succeeded by waiting for the next byte
		1	No operation

Writing the CMD bits will automatically clear the slave interrupt flags and CLKHOLD, and release the SCL line. The ACKACT bit and CMD bits can be written at the same time, and then the acknowledge action will be updated before the command is triggered.

18.12.3 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x02	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – DIF: Data Interrupt Flag**
  This flag is set when a data byte is successfully received; i.e., no bus error or collision occurred during the operation. Writing a one to this bit location will clear DIF. When this flag is set, the slave forces the SCL line low, stretching the TWI clock period. Clearing the interrupt flags will release the SCL line.
  This flag is also cleared automatically when writing a valid command to the CMD bits in the CTRLB register.
- **Bit 6 – APIF: Address/Stop Interrupt Flag**
  This flag is set when the slave detects that a valid address has been received, or when a transmit collision is detected. If the PIEN bit in the CTRLA register is set, a STOP condition on the bus will also set APIF. Writing a one to this bit location will clear APIF. When set for an address interrupt, the slave forces the SCL line low, stretching the TWI clock period. Clearing the interrupt flags will release the SCL line.
  The flag is also cleared automatically for the same condition as DIF.
- **Bit 5 – CLKHOLD: Clock Hold**
  This flag is set when the slave is holding the SCL line low.This is a status flag and a read-only bit that is set when DIF or APIF is set. Clearing the interrupt flags and releasing the SCL line will indirectly clear this flag.
- **Bit 4 – RXACK: Received Acknowledge**
  This flag contains the most recently received acknowledge bit from the master. This is a read-only flag. When read as zero, the most recent acknowledge bit from the maser was ACK, and when read as one, the most recent acknowledge bit was NACK.
- **Bit 3 – COLL: Collision**
  This flag is set when a slave has not been able to transfer a high data bit or a NACK bit. If a collision is detected, the slave will commence its normal operation, disable data, and acknowledge output, and no low values will be shifted out onto the SDA line. Writing a one to this bit location will clear COLL.
  The flag is also cleared automatically when a START or repeated START condition is detected.
- **Bit 2 – BUSERR: TWI Slave Bus Error**
  This flag is set when an illegal bus condition occurs during a transfer. An illegal bus condition occurs if a repeated START or a STOP condition is detected, and the number of bits from the previous START condition is not a multiple of nine. Writing a one to this bit location will clear BUSERR.
  For bus errors to be detected, the bus state logic must be enabled. This is done by enabling the TWI master.
- **Bit 1 – DIR: Read/Write Direction**
  The R/W direction (DIR) flag reflects the direction bit from the last address packet received from a master. When this bit is read as one, a master read operation is in progress. When read as zero, a master write operation is in progress.
- **Bit 0 – AP: Slave Address or Stop**
  This flag indicates whether a valid address or a STOP condition caused the last setting of APIF in the STATUS register.

Table 18-11. TWI slave address or stop.

AP	Description
0	A STOP condition generated the interrupt on APIF
1	Address detection generated the interrupt on APIF

18.12.4 ADDR – Address register

The TWI slave address register should be loaded with the 7-bit slave address (in the seven most significant bits of ADDR) to which the TWI will respond. The lsb of ADDR is used to enable recognition of the general call address (0x00).

Bit	7	6	5	4	3	2	1	0
+0x03	ADDR[7:1]							ADDR[0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:1 – ADDR[7:1]: TWI Slave Address**
  This register contains the TWI slave address used by the slave address match logic to determine if a master has addressed the slave. The seven most-significant bits (ADDR[7:1]) represent the slave address.
  When using 10-bit addressing, the address match logic only supports hardware address recognition of the first byte of a 10-bit address. By setting ADDR[7:1] = 0b11110nn, ”nn” represents bits 9 and 8 of the slave address. The next byte received is bits 7 to 0 in the 10-bit address, and this must be handled by software.
  When the address match logic detects that a valid address byte is received, APIF is set and the DIR flag is updated.
  If the PMEN bit in CTRLA is set, the address match logic responds to all addresses transmitted on the TWI bus. The ADDR register is not used in this mode.
- **Bit 0 – ADDR: General Call Recognition Enable**
  When ADDR[0] is set, this enables general call address recognition logic so the device can respond to a general address call that addresses all devices on the bus.

18.12.5 DATA – Data register

Bit	7	6	5	4	3	2	1	0
+0x04	DATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

The data (DATA) register is used when transmitting and received data. During data transfer, data are shifted from/to the DATA register and to/from the bus. This implies that the DATA register cannot be accessed during byte transfers, and this is prevented by hardware. The DATA register can be accessed only when the SCL line is held low by the slave; i.e., when CLKHOLD is set.

When a master is reading data from the slave, data to send must be written to the DATA register. The byte transfer is started when the master starts to clock the data byte from the slave, followed by the slave receiving the acknowledge bit from the master. DIF and CLKHOLD are set.

When a master writes data to the slave, DIF and CLKHOLD are set when one byte has been received in the DATA register. If smart mode is enabled, reading the DATA register will trigger the bus operation as set by the ACKACT bit.

Accessing the DATA register will clear the slave interrupt flags and CLKHOLD. When an address match occurs, the received address will be stored in the DATA register.

18.12.6 ADDRMASK – Address Mask register

Bit	7	6	5	4	3	2	1	0
+0x05	ADDRMASK[7:1]							ADDREN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:1 – ADDRMASK[7:1]: Address Mask**
  These bits can act as a second address match register or as an address mask register, depending on the ADDREN setting.

If ADDREN is set to zero, ADDRMASK can be loaded with a 7-bit slave address mask. Each bit in ADDRMASK can mask (disable) the corresponding address bit in the ADDR register. If the mask bit is one, the address match between the incoming address bit and the corresponding bit in ADDR is ignored; i.e., masked bits will always match.
If ADDREN is set to one, ADDRMASK can be loaded with a second slave address in addition to the ADDR register. In this mode, the slave will match on two unique addresses, one in ADDR and the other in ADDRMASK.

- **Bit 0 – ADDREN: Address Enable**
  By default, this bit is zero, and the ADDRMASK bits acts as an address mask to the ADDR register. If this bit is set to one, the slave address match logic responds to the two unique addresses in ADDR and ADDRMASK.

18.13 Register Description – TWI Timeout

18.13.1 TOS – Timeout Status Register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	TTOUTSIF	–	TMEXTIF	TSEXTIF	TTOUTMIF
Read/Write	R	R	R	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 4 – TTOUTSIF: Slave Ttimeout Interrupt Flag**
  The slave ttimeout interrupt flag in set when ttimeout enable (TTOUTEN) is set and ttimeout condition occurs in slave. The slave releases the bus after this condition and ready to receive a new START pulse. Writing a one to its bit location will clear the TTOUTSIF flag.
- **Bit 2 – TMEXTIF: Tlowmext Interrupt Flag**
  The tlowmext interrupt flag in set when tlowmext enable (TMEXTEN) is set and tlowmext condition occurs in master. Writing a one to its bit location will clear the TMEXTIF flag. This flag is also automatically cleared by writing into master address register (ADDR[7:0]).
- **Bit 1 – TSEXTIF: Tlowsext Interrupt Flag**
  The tlowsext interrupt flag in set when tlowsext enable (TSEXTEN) is set and tlowsext condition occurs in master. Writing a one to its bit location will clear the TSEXTIF flag. This flag is also automatically cleared by writing into master address register (ADDR[7:0]).
- **Bit 0 – TTOUTMIF: Master Ttimeout Interrupt Flag**
  The master ttimeout interrupt flag in set when ttimeout enable (TTOUTEN) is set and ttimeout condition occurs in master. The master should wait for BUS IDLE condition before starting the next transaction. This ensures that all slave devices have released the bus following the ttimeout condition and ready to receive a new START pulse. Writing a one to its bit location will clear the TTOUTMIF flag. This flag is also automatically cleared by writing into master address register (ADDR[7:0]).

18.13.2 TOCONF – Timeout configuration register

Bit	7	6	5	4	3	2	1	0
+0x01	TTOUTSSEL[2:0]			TMSEXTSEL[1:0]		TTOUTMSEL[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:5 – TTOUTSSEL[2:0]: Slave Ttimeout select**
  These bits select slave ttimeout value according to Table 18-12 on page 258.
- **Bit 4:3 – TMSEXTSEL[1:0]: Master Tlowsext/Tlowmext select**
  These bits select master tlowsext/tlowmext value according Table 18-13 on page 258.
- **Bit 2:0 – TTOUTMSEL[2:0]: Master Ttimeout select**
  These bits select master ttimeout value according to Table 18-12 on page 258.

Table 18-12. Ttimeout configuration.

TTOUTMSEL[2:0]	TOUTSSEL[2:0]	Ttimeout Value
3'b000	3'b000	25ms
3'b001	3'b001	24ms
3'b010	3'b010	23ms
3'b011	3'b011	22ms
3'b100	3'b100	26ms
3'b101	3'b101	27ms
3'b110	3'b110	28ms
3'b111	3'b111	29ms

Table 18-13. Tlowsext/Tlowmext configuration.

TMSEXTSEL[1:0]	Tlowsext Value	Tlowmext Value
2'b00	25ms	10ms
2'b01	24ms	9ms
2'b10	26ms	11ms
2'b11	27ms	12ms

18.14 Register summary - TWI

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	BRIDGEEN	SFMPEN	SSDAHOLD[1:0]		FMPEN	SDAHOLD[1:0]		EDIEN	247
+0x01	MASTER	Offset address for TWI Master								
+0x08	SLAVE	Offset address for TWI Slave								

18.15 Register summary - TWI master

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	INTLVL[1:0]		RIEN	WIEN	ENABLE	–	–	–	248
+0x01	CTRLB	TOIE	TMEXTEN	TSEXTEN	TTOUTEN	TIMEOUT[1:0]		QCEN	SMEN	248
+0x02	CTRLC	–	–	–	–	–	ACKACT	CMD[1:0]		249
+0x03	STATUS	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]		250
+0x04	BAUD	BAUD[7:0]								251
+0x05	ADDR	ADDR[7:0]								251
+0x06	DATA	DATA[7:0]								252

18.16 Register summary - TWI slave

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	INTLVL[1:0]		DIEN	APIEN	ENABLE	PIEN	PMEN	SMEN	253
+0x01	CTRLB	TOIE	–	–	TTOUTEN	–	ACKACT	CMD[1:0]		253
+0x02	STATUS	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP	255
+0x03	ADDR	ADDR[7:0]								256
+0x04	DATA	DATA[7:0]								256
+0x05	ADDRMASK	ADDRMASK[7:1]							ADDREN	256

18.17 Register Summary – TWI timeout

Address	Name	7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	TOS	–	–	–	TTOUTSIF	–	TMEXTIF	TSEXTIF		257
+0x01	TOCONF	TTOUTSSEL[2:0]			TMSEXTSEL[1:0]		TTOUTMSEL[2:0]			258

18.18 Interrupt vector summary

Table 18-14. TWI interrupt vectors and their word offset addresses.

Offset	Source	Interrupt description
0x00	SLAVE_vect	TWI slave interrupt vector
0x02	MASTER_vect	TWI master interrupt vector

19. SPI – Serial Peripheral Interface

19.1 Features

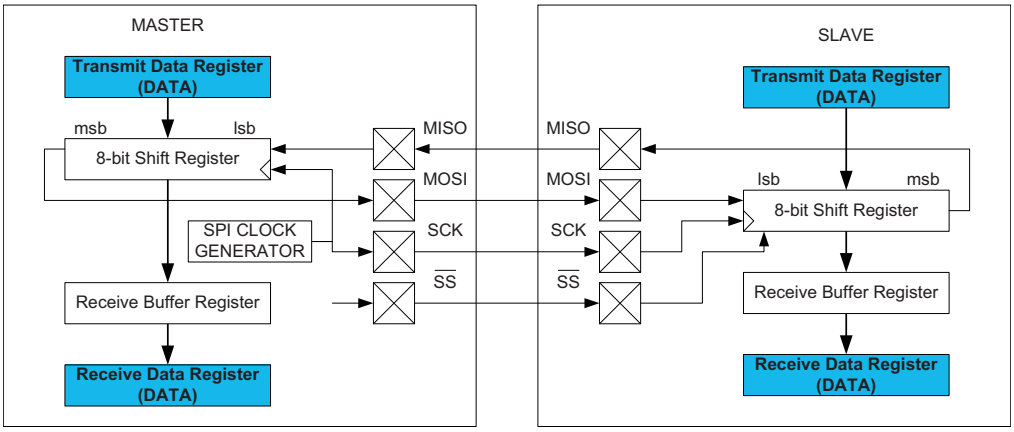
- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- Lsb first or msb first data transfer
- Eight programmable bit rates
- Optional double buffered receive
- Optional buffered transmit
- Optional separate interrupts for
  - Receive complete
  - Transmit complete
  - Transmit data register empty
  - Slave Select line pulled low
- Data overrun detection
- Wake up from idle sleep mode
- Double speed master mode

19.2 Overview

The Serial Peripheral Interface (SPI) is a high-speed synchronous data transfer interface using three or four pins. It allows fast communication between an XMEGA device and peripheral devices or between several microcontrollers. The SPI supports full-duplex communication.

A device connected to the bus must act as a master or slave.The master initiates and controls all data transactions. The interconnection between master and slave devices with SPI is shown in Figure 19-1 on page 260. The system consists of two shift registers and a master clock generator. The SPI master initiates the communication cycle by pulling the slave select ($\overline{SS}$) signal low for the desired slave. Master and slave prepare the data to be sent in their respective shift registers, and the master generates the required clock pulses on the SCK line to interchange data. Data are always shifted from master to slave on the master output, slave input (MOSI) line, and from slave to master on the master input, slave output (MISO) line. After each data packet, the master can synchronize the slave by pulling the $\overline{SS}$ line high.

Figure 19-1. SPI master-slave interconnection.



In SPI slave mode, the control logic will sample the incoming signal on the SCK pin. To ensure correct sampling of this clock signal, the minimum low and high periods must each be longer than two CPU clock cycles.

When the SPI module is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 19-1 on page 261. The pins with user-defined direction must be configured from software to have the correct direction according to the application.

Table 19-1. SPI pin override and directions.

Pin	Master mode	Slave mode
MOSI	User defined	Input
MISO	Input	User defined
SCK	User defined	Input
$\overline{SS}$	User defined	Input

19.3 Master mode

In master mode, the SPI interface has no automatic control of the $\overline{SS}$ line. If the $\overline{SS}$ pin is used, it must be configured as output and controlled by user software. If the bus consists of several SPI slaves and/or masters, a SPI master can use general purpose I/O pins to control the $\overline{SS}$ line to each of the slaves on the bus.

Writing a byte to the DATA register starts the SPI clock generator and the hardware shifts the eight bits into the selected slave. After shifting one byte and there are no pending data, Data Register Empty Interrupt flag (DREIF) is set, the SPI clock generator stops and the Transfer Complete interrupt flag (TXCIF) is set.

If there are pending data, DREIF is cleared; the master will continue to shift the next bytes and after each byte is shifted out, the new data is copied to the shift register and the DREIF is set. Only when a shift is completed and there are no more pending data, will the TXCIF be set. An end of transfer can also be signaled by pulling the $\overline{SS}$ line high. The last incoming byte will be kept in the shift register.

If the $\overline{SS}$ pin is not used it can be disabled by writing the Slave Select Disable (SSD) bit in the CTRLB register. If not disabled and is configured as input, it must be held high to ensure master operation. If the SS pin is set as input and is being driven low, the SPI module will interpret this as another master trying to take control of the bus. To avoid bus contention, the master will take the following action:

1. The master enters slave mode.
2. The slave select interrupt flag (SSIF) is set.

19.4 Slave mode

In slave mode, the SPI module will remain sleeping with the MISO line tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the DATA register, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. If $\overline{SS}$ is driven low, the slave will start to shift out data on the first SCK clock pulse. When one byte has been completely shifted, the SPI interrupt flag is set. The slave may continue placing new data to be sent into the DATA register before reading the incoming data. The last incoming byte will be kept in the buffer register.

When $\overline{SS}$ is driven high, the SPI logic is halted, and the SPI slave will not receive any new data. Any partially received packet in the shift register will be dropped.

As the $\overline{SS}$ pin is used to signal the start and end of a transfer, it is also useful for doing packet/byte synchronization, keeping the slave bit counter synchronous with the master clock generator.

To ensure that write collision never can happen, SPI module can be configured in buffered mode. Data is copied from the Transmit Register to the Shift Register only at receive complete. This means that, after data is written to the Transmit Buffer, one SPI transfer must be completed before the data is copied into the shift register.

19.5 Buffer modes

There are three buffer modes:

- Unbuffered mode:

  The default SPI module is unbuffered in the transmit direction and single buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI DATA register before the entire shift cycle is completed. When receiving data, a received character must be read from the DATA register before the next character has been completely shifted in. Otherwise, the first byte will be lost.

- Buffered mode 1:

  The SPI module is single buffered in the transmit direction and double buffered in the receive direction. A byte written to the transmit register will be copied to the shift register when a full character has been received. When receiving data, a received character must be read from the DATA register before the third character has been completely shifted in to avoid loosing data.

- Buffered mode 2:

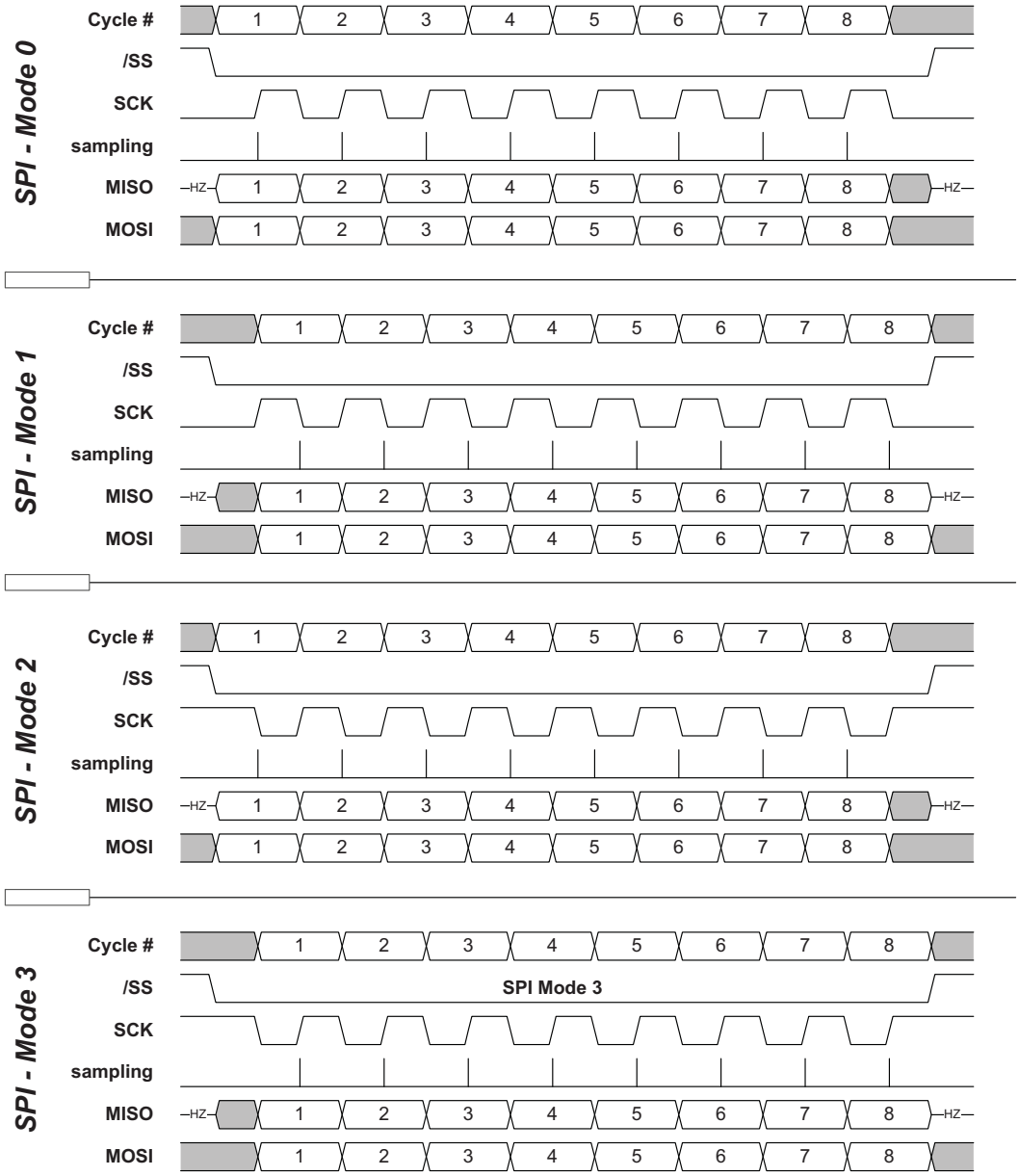
  The SPI module is single buffered in the transmit direction and double buffered in the receive direction. A byte written to the transmit register will be copied to the shift register when the SPI is enabled. Then, one SPI transfer must be completed before the data is copied to the shift register.

19.6 Data modes

There are four combinations of SCK phase and polarity with respect to serial data. The SPI data transfer formats are shown in Figure 19-2. 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 19-2. SPI transfer modes.

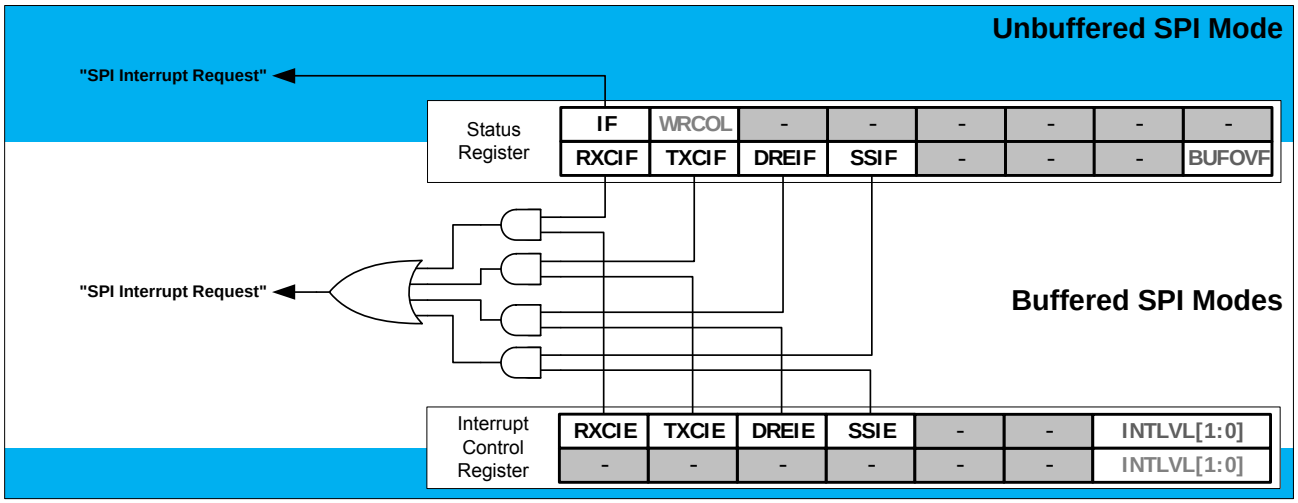


19.7 Interrupts

The SPI module has four interrupt sources. These are combined into one interrupt vector, the SPI interrupt. The interrupt is enabled by setting the interrupt level (INTLVL), while different interrupt sources are enabled individually.

Figure 19-3 on page 264 summarizes the interrupts sources for the SPI module, and shows how they are enabled.

Figure 19-3. SPI interrupt summary.

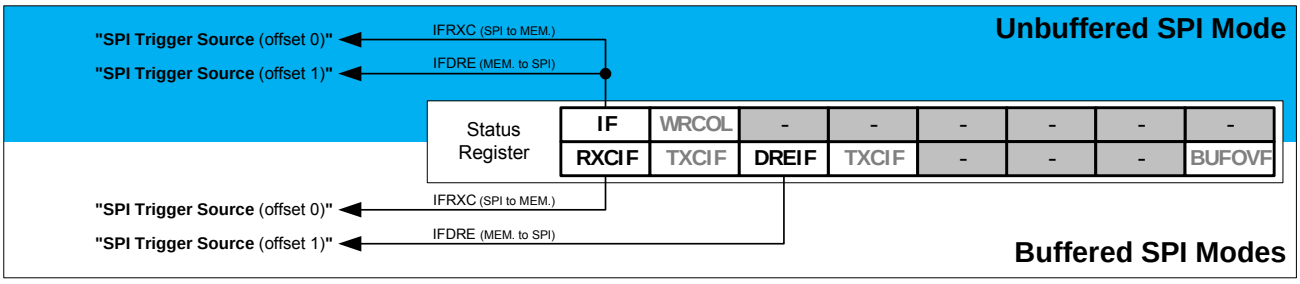


19.8 EDMA support

The SPI slave can trigger an EDMA transfer either as one byte has been moved into the DATA register or as data has been shifted out and new transmit data can be written. The EDMA can be used for half duplex SPI operation using the data register empty or receive complete EDMA triggers.

Figure 19-4 on page 264 summarizes the interrupts sources for the SPI module, and shows how they are enabled.

Figure 19-4. SPI EDMA request summary.



It is possible to use the XMEGA USART in SPI mode and then have EDMA support in master mode. For details, refer to “USART in master SPI mode” on page 285.

19.9 Register description

19.9.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	CLK2X	ENABLE	DORD	MASTER	MODE[1:0]		PRESCALER[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – CLK2X: Clock Double**
  When this bit is set, the SPI speed (SCK frequency) will be doubled in master mode.
- **Bit 6 – ENABLE: Enable**
  Setting this bit enables the SPI module. This bit must be set to enable any SPI operations.
- **Bit 5 – DORD: Data Order**
  DORD decides the data order when a byte is shifted out from the DATA register. When DORD is written to one, the least-significant bit (lsb) of the data byte is transmitted first, and when DORD is written to zero, the most-significant bit (msb) of the data byte is transmitted first.
- **Bit 4 – MASTER: Master Select**
  This bit selects master mode when written to one, and slave mode when written to zero. If $\overline{SS}$ is configured as an input and driven low while master mode is set, master mode will be cleared.
- **Bit 3:2 – MODE[1:0]: Transfer Mode**
  These bits select the transfer mode. The four combinations of SCK phase and polarity with respect to the serial data are shown in Table 19-2 on page 265. 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.

Table 19-2. SPI transfer modes.

MODE[1:0]	Group configuration	Leading edge	Trailing edge
00	0	Rising, sample	Falling, setup
01	1	Rising, setup	Falling, sample
10	2	Falling, sample	Rising, setup
11	3	Falling, setup	Rising, sample

- **Bits 1:0 – PRESCALER[1:0]: Clock Prescaler**
  These two bits control the SPI clock rate configured in master mode. These bits have no effect in slave mode. The relationship between SCK and the peripheral clock frequency (clk_{PER}) is shown in Table 19-4 on page 266. Setting the CLK2X will double the frequency as shown in Table 19-4 on page 266.

Table 19-3. Prescaler clock configuration.

PRESCALER[1:0]	Group configuration	Comment
00	DIV4	Divide clock by 4
01	DIV16	Divide clock by 16
10	DIV64	Divide clock by 64
11	DIV128	Divide clock by 128

Table 19-4. Relationship between SCK and the peripheral clock (Clk_{PER}) frequency.

CLK2X	PRESCALER[1:0]	SCK frequency
0	00	$Clk_{PER}/4$
0	01	$Clk_{PER}/16$
0	10	$Clk_{PER}/64$
0	11	$Clk_{PER}/128$
1	00	$Clk_{PER}/2$
1	01	$Clk_{PER}/8$
1	10	$Clk_{PER}/32$
1	11	$Clk_{PER}/64$

19.9.2 INTCTRL – Interrupt Control register

	Bit	7	6	5	4	3	2	1	0
Unbuffered mode	+0x01	–	–	–	–	–	–	INTLVL[1:0]	
Buffer modes		RXCIE	TXCIE	DREIE	SSIE	–	–		
Unbuffered mode	Read/Write	R	R	R	R	R	R	R/W	R/W
Buffer modes	Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
	Initial value	0	0	0	0	0	0	0	0

Note: For details on buffer modes, refer to Table 19-5 on page 269.

19.9.2.1 Unbuffered mode

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1:0 – INTLVL[1:0]: Interrupt Level**
  These bits enable the SPI interrupt and select the interrupt level, as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132. The enabled interrupt will be triggered when IF is set in the STATUS register.

19.9.2.2 Buffered modes

- **Bit 7 – RXCIE – Receive Complete Interrupt Enable**
  This bit enables the receive complete interrupt. The interrupt level is defined by INTLVL[1:0]. The enabled interrupt will be triggered when the RXCIF flag in the STATUS register is set.

- **Bit 6 – TXCIE – Transfer Complete Interrupt Enable**
  This bit enables the transfer complete interrupt. The interrupt level is defined by INTLVL[1:0]. The enabled interrupt will be triggered when the TXCIF flag in the STATUS register is set.
- **Bit 5 – DREIE – Data Register Empty Interrupt Enable**
  This bit enables the data register empty interrupt. The interrupt level is defined by INTLVL[1:0]. The enabled interrupt will be triggered when the DREIF flag in the STATUS register is set.
- **Bit 4 – SSIE – Slave Select trigger Interrupt Enable**
  This bit enables the Slave Select interrupt. The interrupt level is defined by INTLVL[1:0]. The enabled interrupt will be triggered when the SSIF flag in the STATUS register is set.
- **Bit 3:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1:0 – INTLVL[1:0]: Interrupt Level**
  These bits enable the SPI interrupt and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will be triggered when corresponding flags are set in the STATUS register.

19.9.3 STATUS – Status register

	Bit	7	6	5	4	3	2	1	0
Unbuffered mode	+0x02	**IF**	**WRCOL**	**–**	**–**	**–**	**–**	**–**	**–**
Buffer modes		**RXCIF**	**TXCIF**	**DREIF**	**SSIF**	**–**	**–**	**–**	**BUFOVF**
Unbuffered mode	Read/Write	R	R	R	R	R	R	R	R
Buffer modes	Read/Write	R/W	R/W	R/W	R/W	R	R	R	R/W
	Initial value	0	0	0	0	0	0	0	0

Note: For details on buffer modes, refer to Table 19-5 on page 269.

19.9.3.1 Unbuffered mode

- **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 DATA register. If $\overline{SS}$ is configured as input and is driven low when the SPI is in master mode, this will also set this flag. IF is cleared by hardware when executing the corresponding interrupt vector. Alternatively, the IF flag can be cleared by first reading the STATUS register when IF is set, and then accessing the DATA register.
- **Bit 6 – WRCOL: Write Collision Flag**
  The WRCOL flag is set if the DATA register is written during a data transfer. This flag is cleared by first reading the STATUS register when WRCOL is set, and then accessing the DATA register.
- **Bit 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

19.9.3.2 Buffered modes

- **Bit 7 – RXCIF: Receive Complete Interrupt Flag**
  This flag is set when there is unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data).
  When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from DATA in order to clear RXCIF. If not, a new interrupt will occur directly after the return from the current interrupt. This flag can also be cleared by writing a one 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 are no new data in the transmit buffer (DATA). The flag is cleared by writing a one to its bit location.

- **Bit 5 – DREIF: Data Register Empty Interrupt Flag**
  This flag indicates whether the transmit buffer (DATA) is ready to receive new data. The flag is one when the transmit buffer is empty and zero when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. DREIF is set after a reset to indicate that the transmitter is ready.
  DREIF is cleared by writing DATA. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to DATA in order to clear DREIF or disable the data register empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.
- **Bit 4 – SSIF: Slave Select Interrupt Flag**
  This flag indicates that the SPI has been in master mode and the SS line has been pulled low externally so the SPI is now working in slave mode. The flag will only be set if the Slave Select Disable (SSD) is not enabled. The flag is cleared by writing a one to its bit location.
- **Bit 3:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – BUFOVF: Buffer Overflow**
  This flag indicates data loss due to a receiver 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 characters) and the third byte has been received. If there is no transmit data the buffer overflow will not be set before the start of a new serial transfer. This flag is valid until the receive buffer (DATA) is read. Always write this bit location to zero when writing the STATUS register.

19.9.4 DATA – Data register

Bit	7	6	5	4	3	2	1	0
+0x03	DATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 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, and the byte written to the register will be shifted out on the SPI output line.
  Reading the register causes the first byte in the buffer FIFO to be read. Additionally received bytes will then be shifted in the FIFO.

19.9.5 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x04	BUFMODE[1:0]		–	–	–	SSD	–	–
Read/Write	R/W	R/W	R	R	R	R/W	R	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – BUFMODE[1:0]: Buffer Modes**
  Setting these bits will enable the buffer modes for SPI. Buffers for both receive and transmit are added to the SPI.

Table 19-5. SPI buffer modes.

BUFMODE[1:0]	Group configuration	Description
00	OFF	Unbuffered mode: - 1 buffer in reception, no buffer in transmission - 1 interrupt flag for both transmission and reception
01	-	Reserved
1 0	BUFMODE1	Buffer Mode 1: - 2 buffers in reception, 1 buffer in transmission - Separated interrupt flags for transmission and reception - 1 SPI transfer must be completed before the data is copied into the shift register, even after SPI enable (1st data transmitted = dummy byte)
11	BUFMODE2	Buffer Mode 2: - 2 buffers in reception, 1 buffer in transmission - Separated interrupt flags for transmission and reception - Immediate write data into shift register after SPI enable. Then, 1 SPI transfer must be completed before the data is copied into the shift register.

- **Bit 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – SSD: Slave Select Disable**
  Setting this bit will disable the Slave Select line when operating as SPI Master.
- **Bit 1:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

19.10 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	CLK2X	ENABLE	DORD	MASTER	MODE[1:0]		PRESCALER[1:0]		265
+0x01	INTCTRL	–	–	–	–	–	–	INTLVL[1:0]		266
		RXCIE	TXCIE	DREIE	SSIE	–	–			
+0x02	STATUS	IF	WRCOL	–	–	–	–	–	–	267
		RXCIF	TXCIF	DREIF	SSIF	–	–	–	BUFOVF	
+0x03	DATA	DATA[7:0]								268
+0x04	CTRLB	BUFMODE[1:0]		–	–	–	SSD	–	–	268
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	

19.11 Interrupt vector summary

Table 19-6. SPI Interrupt vectors and their word offset address

Offset	Source	Interrupt description
0x00	SPI_vect	SPI interrupt vector

20. USART

20.1 Features

- Full-duplex or one-wire half-duplex operation
- Asynchronous or synchronous operation
  - Synchronous clock rates up to 1/2 of the device clock frequency
  - Asynchronous clock rates up to 1/8 of the device clock frequency
- Supports serial frames with:
  - 5, 6, 7, 8 or 9 data bits
  - Optionally even and odd parity bits
  - 1 or 2 stop bits
- Fractional baud rate generator
  - Can generate desired baud rate from any system clock frequency
  - No need for external oscillator with certain frequencies
- Built-in error detection and correction schemes
  - Odd or even parity generation and parity check
  - Data overrun and framing error detection
  - Noise filtering includes false start bit detection and digital low-pass filter
- Separate interrupts for
  - Transmit complete
  - Transmit Data Register empty
  - Receive complete
- Multiprocessor communication mode
  - Addressing scheme to address a specific devices on a multi-device bus
  - Enable unaddressed devices to automatically ignore all frames
- Start Frame detection in UART mode
- Master SPI mode
  - Double buffered operation
  - Configurable data order
  - Operation up to 1/2 of the peripheral clock frequency
- IRCOM module for IrDA compliant pulse modulation/demodulation
- Can be linked with XMEGA Custom Logic (XCL):
  - Send and receive events from peripheral counter (PEC) to extend frame length
  - Modulate/demodulate data within the frame by using the glue logic outputs

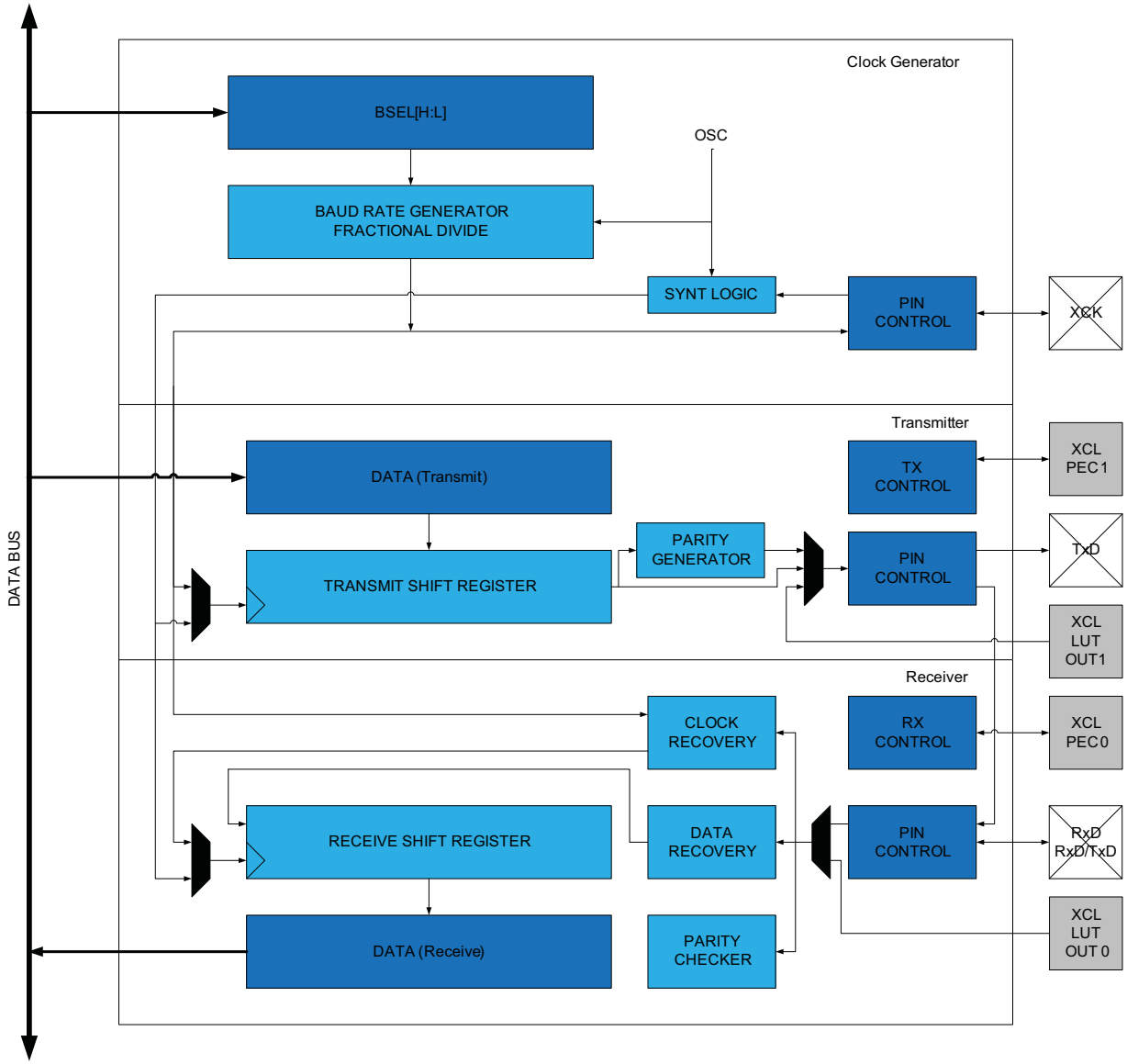
20.2 Overview

The universal synchronous and asynchronous serial receiver and transmitter (USART) is a fast and flexible serial communication module. The USART supports full-duplex communication, asynchronous and synchronous operation and one-wire configurations. The USART can be set in SPI master mode and used for SPI communication.

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 complete enable fully interrupt driven communication. Frame error and buffer overflow are detected in hardware and indicated with separate status flags. Even or odd parity generation and parity check can also be enabled.

A block diagram of the USART and closely related peripheral modules (in grey) is shown in Figure 20-1 on page 272.The main functional blocks are the clock generator, the transmitter, and the receiver, which are indicated in dashed boxes.

Figure 20-1. USART block diagram.



The clock generator includes a fractional baud rate generator that is able to generate a wide range of USART baud rates from any system clock frequencies. This removes the need to use an external crystal oscillator with a specific frequency to achieve a required baud rate. It also supports external clock input in synchronous slave operation.

The transmitter consists of a single write buffer (DATA), a Shift Register and a parity generator. The write buffer allows continuous data transmission without any delay between frames.

The receiver consists of a two-level receive buffer (DATA) and a Shift Register. Data and clock recovery units ensure robust synchronization and noise filtering during asynchronous data reception. It includes frame error, buffer overflow, and parity error detection.

When the USART is set in one-wire mode, the transmitter and the receiver share the same RxD I/O pin.

When the USART is set in master SPI mode, all USART-specific logic is disabled, leaving the transmit and receive buffers, Shift registers, and baud rate generator enabled. Pin control and interrupt generation are identical in both modes. The registers are used in both modes, but their functionality differs for some control settings.

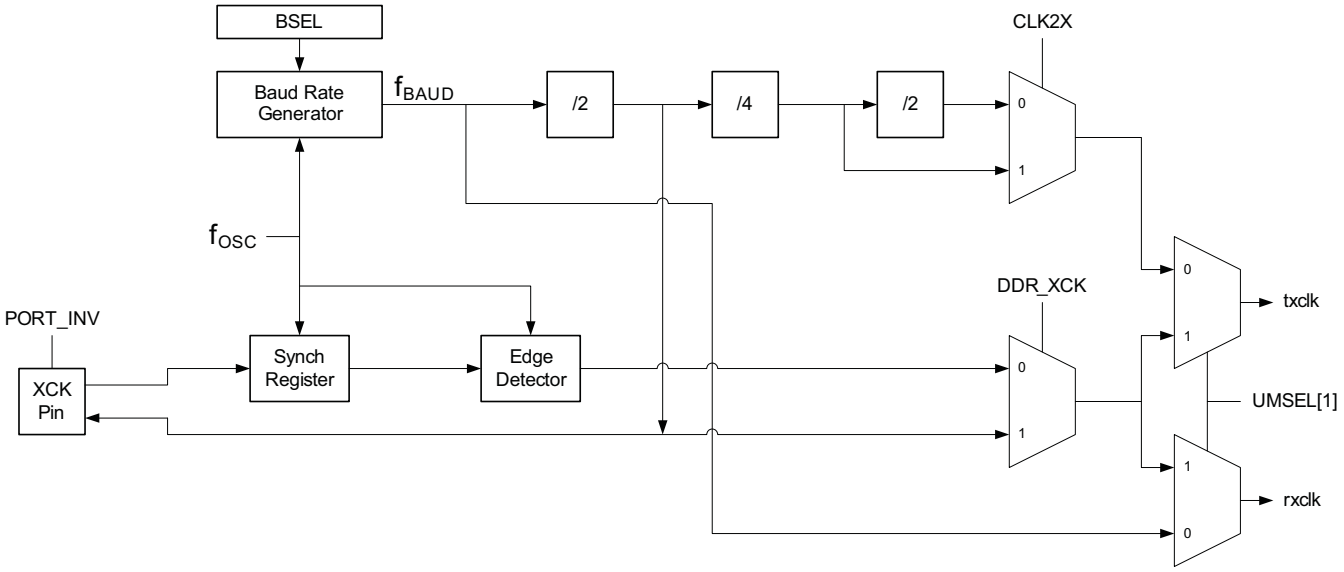
An IRCOM module can be enabled for one USART to support IrDA 1.4 physical compliant pulse modulation and demodulation for baud rates up to 115.2kbps. For details, refer to “IRCOM – IR Communication Module” on page 296.

One USART can be linked to the XMEGA Custom Logic unit (XCL). When used with the XCL, the data length within an USART/SPI frame can be controlled by the peripheral counter (PEC) within the XCL. In addition, the TxD/RxD data can be encoded/decoded before the signal is fed into the USART receiver or after the signal is output from transmitter when the USART is connected to XCL LUT outputs. For more details on how using and setting the LUT's and PEC's, refer to "XCL – XMEGA Custom Logic" on page 300 module.

20.3 Clock generation

The clock used for baud rate generation and for shifting and sampling data bits is generated internally by the fractional baud rate generator or externally from the transfer clock (XCK) pin. Five modes of clock generation are supported: normal and double-speed asynchronous mode, master and slave synchronous mode, and master SPI mode.

Figure 20-2. Clock generation logic block diagram.



20.3.1 Internal clock generation - The fractional baud rate generator

The fractional baud rate generator is used for internal clock generation for asynchronous modes, synchronous master mode, and master SPI mode operation. The output frequency generated (f_{BAUD}) is determined by the period setting (BSEL), an optional scale setting (BSCALE), and the peripheral clock frequency (f_{PER}). Table 20-1 on page 274 contains equations for calculating the baud rate (in bits per second) and for calculating the BSEL value for each mode of operation. It also shows the maximum baud rate versus peripheral clock frequency. BSEL can be set to any value between 0 and 4095. BSCALE can be set to any value between -7 and +7, and increases or decreases the baud rate slightly to provide the fractional baud rate scaling of the baud rate generator.

When BSEL is 0, BSCALE must also be 0. Also, the value $2^{ABS(BSCALE)}$ must at most be one half of the minimum number of clock cycles a frame requires. For more details, see "Internal clock generation - The fractional baud rate generator" on page 273.

Table 20-1. Equations for calculating Baud Rate Register setting.

Operating mode	Conditions	Baud rate calculation [1]	BSEL value calculation
Asynchronous Normal Speed mode (CLK2X = 0)	$BSCALE \geq 0$ $f_{BAUD} \leq \frac{f_{PER}}{16}$	$f_{BAUD} = \frac{f_{PER}}{2^{BSCALE} \cdot 16(BSEL+1)}$	$BSEL = \frac{f_{PER}}{2^{BSCALE} \cdot 16 f_{BAUD}} - 1$
	$BSCALE < 0$ $f_{BAUD} \leq \frac{f_{PER}}{16}$	$f_{BAUD} = \frac{f_{PER}}{16((2^{BSCALE} \cdot BSEL)+1)}$	$BSEL = \frac{1}{2^{BSCALE}}\left(\frac{f_{PER}}{16 f_{BAUD}} - 1\right)$
Asynchronous Double Speed mode (CLK2X = 1)	$BSCALE \geq 0$ $f_{BAUD} \leq \frac{f_{PER}}{8}$	$f_{BAUD} = \frac{f_{PER}}{2^{BSCALE} \cdot 8(BSEL+1)}$	$BSEL = \frac{f_{PER}}{2^{BSCALE} \cdot 8 f_{BAUD}} - 1$
	$BSCALE < 0$ $f_{BAUD} \leq \frac{f_{PER}}{8}$	$f_{BAUD} = \frac{f_{PER}}{8((2^{BSCALE} \cdot BSEL)+1)}$	$BSEL = \frac{1}{2^{BSCALE}}\left(\frac{f_{PER}}{8 f_{BAUD}} - 1\right)$
Synchronous and master SPI mode	$f_{BAUD} < \frac{f_{PER}}{2}$	$f_{BAUD} = \frac{f_{PER}}{2(BSEL+1)}$	$BSEL = \frac{f_{PER}}{2 f_{BAUD}} - 1$

Notes: 1. The baud rate is defined to be the transfer rate bit per second (bps).

For BSEL = 0, all baud rates be achieved by changing BSEL instead of setting BSCALE: BSEL = (2 BSCALE-1).

BSCALE	BSEL		BSCALE	BSEL
1	0	->	0	1
2	0	->	0	3
3	0	->	0	7
4	0	->	0	15
5	0	->	0	31
6	0	->	0	63
7	0	->	0	127

20.3.2 External clock

External clock (XCK) is used in synchronous slave mode operation. The XCK clock input is sampled on the peripheral clock frequency (f_{PER}), and the maximum XCK clock frequency (f_{XCK}) is limited by the following:

For each high and low period, XCK clock cycles must be sampled twice by the peripheral clock. 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.

20.3.3 Double speed operation

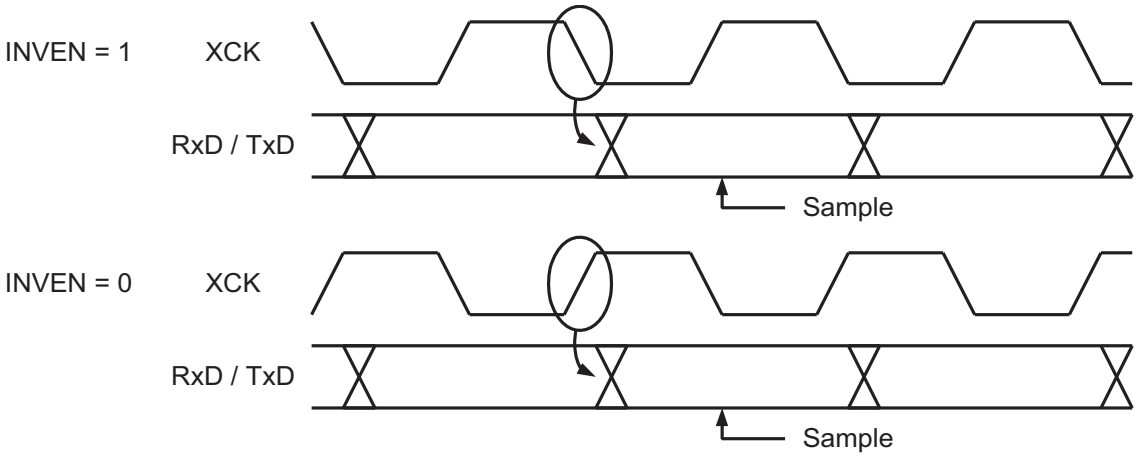
Double speed operation allows for higher baud rates under asynchronous operation with lower peripheral clock frequencies. When this is enabled, the baud rate for a given asynchronous baud rate setting shown in Table 20-1 on page 274 will be doubled. In this mode, the receiver will use half the number of samples (reduced from 16 to 8) for data

sampling and clock recovery. Due to the reduced sampling, a more accurate baud rate setting and peripheral clock are required. See “Asynchronous data reception” on page 279 for more details.

20.3.4 Synchronous clock operation

When synchronous mode is used, the XCK pin controls whether the transmission clock is input (slave mode) or output (master mode). The corresponding port pin must be set to output for master mode or to input for slave mode. The normal port operation of the XCK pin will be overridden. The dependency between the clock edges and data sampling or data change is the same. Data input (on RxD) is sampled at the XCK clock edge which is opposite the edge where data output (TxD) is changed.

Figure 20-3. Synchronous mode XCK timing.



Using the inverted I/O (INVEN) setting for the corresponding XCK port pin, the XCK clock edges used for data sampling and data change can be selected. If inverted I/O is disabled (INVEN=0), data will be changed at the rising XCK clock edge and sampled at the falling XCK clock edge. If inverted I/O is enabled (INVEN=1), data will be changed at the falling XCK clock edge and sampled at the rising XCK clock edge. For more details, see “I/O Ports” on page 139.

20.3.5 Master SPI mode clock generation

For master SPI mode operation, only internal clock generation is supported. This is identical to the USART synchronous master mode, and the baud rate or BSEL setting is calculated using the same equations (see Table 20-1 on page 274).

There are four combinations of the SPI clock (SCK) phase and polarity with respect to the serial data, and these are determined by the clock phase (UCPHA) control bit and the inverted I/O pin (INVEN) settings. The data transfer timing diagrams are shown in Figure 20-4 on page 276.

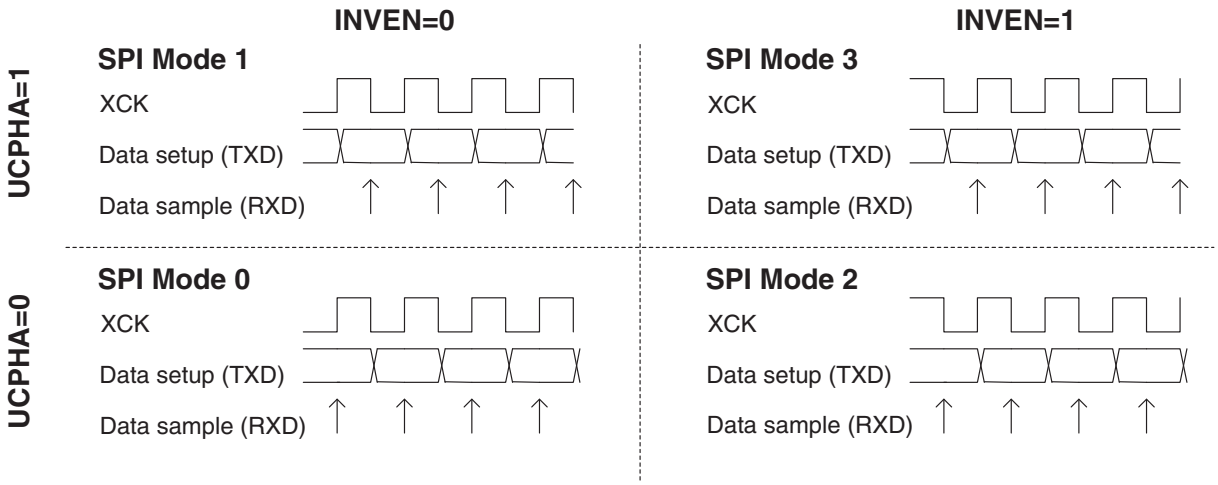
Data bits are shifted out and latched in on opposite edges of the XCK signal, ensuring sufficient time for data signals to stabilize. The UCPHA and INVEN (bit of POTRTx.PINnCTRL Register) settings are summarized in Table 20-2 on page 275. Changing the setting of any of these bits during transmission will corrupt both the receiver and transmitter.

Table 20-2. INVEN and UCPHA functionality.

SPI mode	INVEN	UCPHA	Leading edge	Trailing edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

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 20-4. UCPHA and INVEN data transfer timing diagrams.



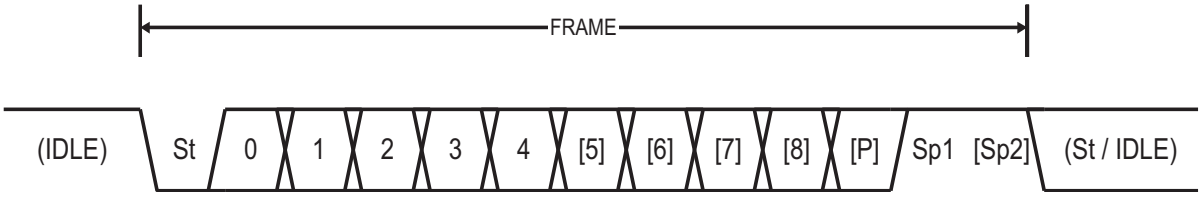
20.4 Frame formats

Data transfer is frame based, where a serial frame consists of one character of data bits with synchronization bits (start and stop bits) and an optional parity bit for error checking. Note that this does not apply to master SPI operation (See "SPI frame formats" on page 277). The USART accepts all combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8 or 9 data bits
- variable data bits, controlled by the peripheral counter from XCL (PEC)
- no, even, or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit, followed by all the data bits (least-significant bit first and most significant bit last). If enabled, the parity bit is inserted after the data bits, before the first stop bit. One frame can be directly followed by a start bit and a new frame, or the communication line can return to the idle (high) state. Figure 20-5 on page 276 illustrates the possible combinations of frame formats. Bits inside brackets are optional.

Figure 20-5. Frame formats.



St Start bit, always low.

(n) Data bits (0 to 8 in standard mode, variable when controlled by PEC).

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.

20.4.1 Parity bit calculation

Even or odd parity can be selected for error checking. If even parity is selected, the parity bit is set to one if the number of logical one data bits is odd (making the total number of ones even). If odd parity is selected, the parity bit is set to one if the number of logical one data bits is even (making the total number of ones odd).

When variable data length mode is enabled, the parity bit calculation is not supported.

20.4.2 SPI frame formats

The serial frame in SPI mode is defined to be one character of eight data bits. The USART in master SPI mode has three valid frame formats:

- 8-bit data, msb first
- 8-bit data, lsb first
- variable data bits (lsb first), controlled by the peripheral counter from XCL (PEC)

After a complete frame is transmitted, a new frame can directly follow it, or the communication line can return to the idle (high) state.

20.5 USART full-duplex initialization

For setting the USART in full-duplex mode, the following initialization sequence is recommended:

1. Set the TxD pin value high, and optionally set the XCK pin low.
2. Set the TxD and optionally the XCK pin as output.
3. Set the baud rate and frame format.
4. Set the mode of operation (enables XCK pin output in synchronous mode).
5. Optionally configure the XCL for variable data length and encoding/decoding truth table.
6. Enable the transmitter or the receiver, depending on the usage.

For interrupt-driven USART operation, global interrupts should be disabled during the initialization.

Before doing a re-initialization with a changed baud rate or frame format, be sure that there are no ongoing transmissions while the registers are changed.

20.6 USART one-wire initialization

For setting the USART in one-wire mode, the following initialization sequence is recommended:

1. Set the TxD/RxD pin value high, and optionally set the XCK pin low.
2. Optionally, set the TxD/RxD input pin as Wired-AND or Wired-OR.
3. Set the TxD/RxD and optionally the XCK pin as output.
4. Set the baud rate and frame format.
5. Set the mode of operation (enables XCK pin output in synchronous mode).
6. Optionally configure the XCL for variable data length and encoding/decoding truth table.
7. Enable the transmitter or the receiver, depending on the usage.

For interrupt-driven USART operation, global interrupts should be disabled during the initialization.

Before doing a re-initialization with a changed baud rate or frame format, be sure that there are no ongoing transmissions while the registers are changed.

20.7 Data transmission - The USART transmitter

When the transmitter has been enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the transmitter's serial output. The direction of the pin must be set as output using the direction register for the corresponding port. For details on port pin control and output configuration, refer to “I/O Ports” on page 139. If the USART is configured for one-wire operation, the USART will automatically override the RxD/TxD pin to output, when the transmitter is enabled.

20.7.1 Sending frames

A data transmission is initiated by loading the transmit buffer (DATA) with the data to be sent. The data in the transmit buffer are moved to the Shift Register when the Shift Register is empty and ready to send a new frame. The Shift Register is loaded if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with data, it will transfer one complete frame.

When the data frame length is controlled by the peripheral counter in the XCL unit, the XCL and event controlled mode in USART have to be initiated before enabling the transmitter. In variable data length mode, the minimum frame length is one bit, the maximum is 256 bits. The maximum size must be chosen according to the oscillator accuracy.

A transmission is initiated by loading the transmit buffer (DATA) with the data to be sent. When the first data is loaded in the Shift Register, the USART provides the restart command to the peripheral counter. Each bit shift will decrement the peripheral counter. A compare match is provided by the XCL when the internal counter value reaches BOTTOM (zero). While the compare match is not received, the USART continues to shift out the data bits. If the compare match occurs before completing an 8-bit data shift, the USART changes its state to stop bits. If the Shift Register is empty before the compare match is received, then new data is automatically loaded in the Shift Register and transmission continues. If there is no more data to transmit and the compare match is not received, the transmission is aborted and Data Register empty flag (DREIF) is generated. The USART returns to IDLE state and stops any event generation for peripheral counter. The user can then calculate the number of bits already sent over the line.

When not used with the EDMA, the system has to spend the minimum of time in the interrupt routine to load new data in the DATA Register.

The transmit complete interrupt flag (TXCIF) is set and the optional interrupt is generated when the entire frame in the Shift Register has been shifted out and there are no new data present in the transmit buffer.

The Transmit Data Register (DATA) can only be written when the Data Register Empty Flag (DREIF) is set, indicating that the register is empty and ready for new data.

When using frames with fewer than eight bits, the most-significant bits written to DATA are ignored. If 9-bit characters are used, the ninth bit must be written to the TXB8 bit before the low byte of the character is written to DATA.

20.7.2 Disabling the transmitter

A disabling of the transmitter will not become effective until ongoing and pending transmissions are completed; i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted and optionally while the compare match is not received from peripheral counter. In this case, it is possible to write 0x00 to the peripheral Counter Register to generate the compare match. When the transmitter is disabled, it will no longer override the TxDn pin, and the pin direction is set as input automatically by hardware, even if it was configured as output by the user.

20.8 Data reception - The USART receiver

When the receiver is enabled, the RxD pin functions as the receiver's serial input. The direction of the pin must be set as input, which is the default pin setting.

20.8.1 Receiving frames

The receiver starts data reception when it detects a valid start bit. 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 (RXCIF) is set, and the optional interrupt is generated.

When the data frame length is controlled by the peripheral counter in the XCL unit, the XCL and event controlled mode in USART have to be initiated before enabling the receiver. When a start bit is detected, the USART sends the restart command to the peripheral counter.

Each bit shift will decrement the peripheral counter. A compare match is provided by the XCL when the internal counter value reaches BOTTOM (zero). While the compare match is not received, the USART continues to shift in the data bits. If the compare match occurs before completing an 8-bit data shifts, the USART changes its state to stop bits. After each 8-bit data reception, data receive flag (DRIF) and optionally an interrupt, is generated. If the data buffer overflow condition is generated, the reception is aborted and buffer overflow flag is set. No more counter commands are generated for the peripheral counter while a new start bit condition is not detected. In such error condition, it is highly recommended to disable the receiver part, unless any falling edge will be considered as a valid start bit and the peripheral counter can automatically restart its operation. Data receive interrupt flag and receive complete interrupt flag

share the same interrupt line and interrupt settings. When not used with the EDMA, the system has to spend the minimum of time in the interrupt routine to read data from Data Register (DATA).

The receiver buffer can be read by reading the Data Register (DATA) location. DATA should not be read unless the receive complete interrupt flag is set. When using frames with fewer than eight bits, the unused most-significant bits are read as zero. If 9-bit characters are used, the ninth bit must be read from the RXB8 bit before the low byte of the character is read from DATA.

If data frame length is controlled by the peripheral timer, the RXB8 bit is unused during reception. This bit location will be used to store the data reception flag.

20.8.2 Receiver error flags

The USART receiver has three error flags. The frame error (FERR), buffer overflow (BUFOVF) and parity error (PERR) flags are accessible from the Status Register. The error flags are located in the receive FIFO buffer together with their corresponding frame. Due to the buffering of the error flags, the Status Register must be read before the receive buffer (DATA), since reading the DATA location changes the FIFO buffer.

20.8.3 Parity checker

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 is set.

When variable data length mode is enabled, the parity checker is not supported.

20.8.4 Disabling the receiver

A disabling of the receiver will be immediate. The receiver buffer will be flushed, and data from ongoing receptions will be lost.

20.8.5 Flushing the receive buffer

If the receive buffer has to be flushed during normal operation, read the DATA location until the receive complete interrupt flag is cleared.

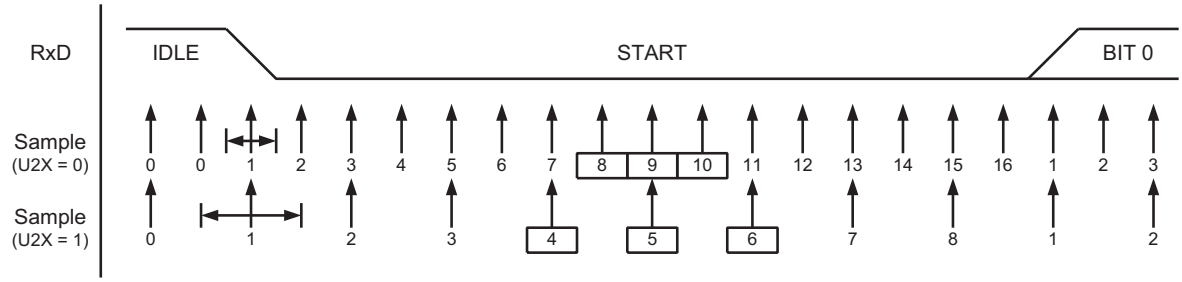
20.9 Asynchronous data reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery unit is used for synchronizing the incoming asynchronous serial frames at the RxD pin to the internally generated baud rate clock. It samples and low-pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

20.9.1 Asynchronous clock recovery

The clock recovery unit synchronizes the internal clock to the incoming serial frames. Figure 20-6 illustrates the sampling process for the start bit of an incoming frame. The sample rate is 16 times the baud rate for normal mode, and eight times the baud rate for double speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode of operation. Samples denoted as zero are samples done when the RxD line is idle; i.e., when there is no communication activity.

Figure 20-6. Start bit sampling.

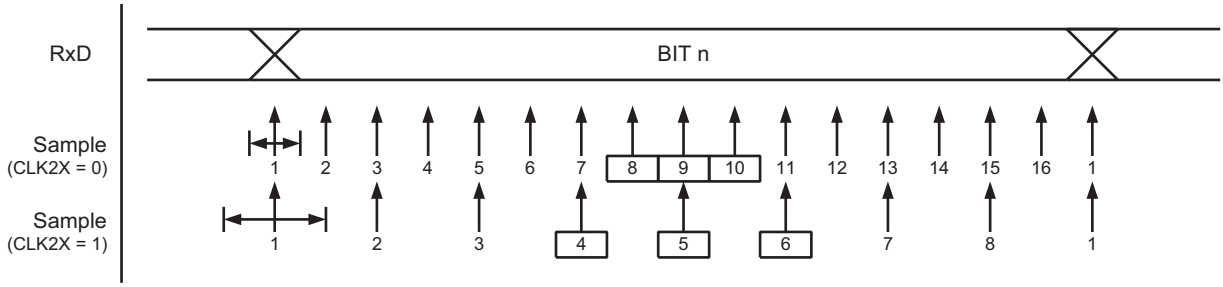


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Sample 1 denotes the first zero-sample, as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for normal mode and samples 4, 5, and 6 for double speed mode to decide if a valid start bit is received. If two or three samples have a low level, the start bit is accepted. The clock recovery unit is synchronized, and the data recovery can begin. If two or three samples have a high level, the start bit is rejected as a noise spike, and the receiver looks for the next high-to-low transition. The process is repeated for each start bit.

20.9.2 Asynchronous data recovery

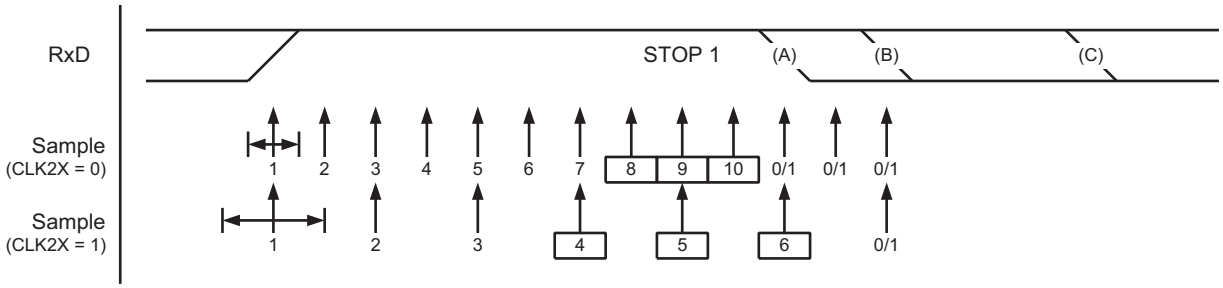
The data recovery unit uses sixteen samples in normal mode and eight samples in double speed mode for each bit. Figure 20-7 shows the sampling process of data and parity bits.

Figure 20-7. Sampling of data and parity bits.



As for start bit detection, an identical majority voting technique is used on the three center samples for deciding of the logic level of the received bit. The process is repeated for each bit until a complete frame is received. It includes the first stop bit, but excludes additional ones. If the sampled stop bit is a 0 value, the frame error (FERR) flag will be set. Figure 20-8 shows the sampling of the stop bit in relation to the earliest possible beginning of the next frame's start bit.

Figure 20-8. 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 Stop Bit Sampling and Next Start Bit Sampling. For double speed mode, the first low level must be delayed to point (B). Point (C) marks a stop bit of full length at nominal baud rate. The early start bit detection influences the operational range of the receiver.

20.9.3 Asynchronous operational range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If an external transmitter is sending using bit rates that are too fast or too slow, or if the internally generated baud rate of the receiver does not match the external source's base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)\cdot S+S_M}$$

D	Sum of character size and parity size (D = 5 to 10 bit).
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 and SF = 4 for Double Speed mode.
S_M	Middle sample number used for majority voting. SM = 9 for normal speed and SM = 5 for Double Speed mode.
R_{slow}	Is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate.
R_{fast}	Is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 20-3 and Table 20-4 on page 281 list the maximum receiver baud rate error that can be tolerated. Normal Speed mode has higher toleration of baud rate variations.

Table 20-3. Recommended maximum receiver baud rate error for normal speed mode (CLK2X = 0).

D #(Data + Parity bit)	R_{slow} [%]	R_{fast} [%]	Maximum total error [%]	Receiver max. receiver error [%]
5	93.20	106.67	+6.67/-6.80	±3.0
6	94.12	105.79	+5.79/-5.88	±2.5
7	94.81	105.11	+5.11/-5.19	±2.0
8	95.36	104.58	+4.58/-4.54	±2.0
9	95.81	104.14	+4.14/-4.19	±1.5
10	96.17	103.78	+3.78/-3.83	±1.5

Table 20-4. Recommended maximum receiver baud rate error for double speed mode (CLK2X = 1).

D #(Data + Parity bit)	R_{slow} [%]	R_{fast} [%]	Maximum total error [%]	Receiver max. receiver error [%]
5	94.12	105.66	+5.66/-5.88	±2.5
6	94.92	104.92	+4.92/-5.08	±2.0
7	95.52	104.35	+4.35/-4.48	±1.5
8	96.00	103.90	+3.90/-4.00	±1.5
9	96.39	103.53	+3.53/-3.61	±1.5
10	96.70	103.23	+3.23/-3.30	±1.0

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.

20.9.4 Start frame detection

The start frame detection is supported in UART mode only and takes place only if the system is in deeper sleep modes. The UART start frame detector can wake up the system from power save, standby or extended standby sleep modes when a start bit is detected. In power save mode, the internal 8MHz oscillator in low power mode must be used as clock source, but in standby or extended standby modes it can operate with any clock source.

When a high-to-low transition is detected on RxDn, the oscillator is powered up and the UART clock is enabled. After start-up, the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the oscillator start-up time. Start-up time of the oscillators varies with supply voltage and temperature. For details on oscillator start-up time characteristics, refer to device datasheet.

If a false start bit is detected and if the system has not been waken-up by another source, the oscillator will be automatically powered-off and the UART waits for the next transition.

The UART start frame detection works in asynchronous mode only. It is enabled by writing the Start Frame Detection bit (SFDEN) in “CTRLB – Control register B” on page 290. If the start bit is detected, the UART Start Interrupt Flag (RXSIF) bit is set.

In active and idle sleep modes, the asynchronous detection is automatically disabled. In power down sleep mode, the asynchronous detector is enabled, but the internal oscillator is never powered. In such case, it is highly recommended to disable the start detector before going in power down sleep mode.

The UART receive complete flag and UART start interrupt flag share the same interrupt line, but each has its dedicated interrupt settings. The Table 20-5 shows the USART start frame detection modes, depending of interrupt setting.

Table 20-5. USART start frame detection modes.

SFDEN	RXSIF interrupt	RXCIF interrupt	Comment
0	x	x	Standard mode
1	Disabled	Disabled	Only the oscillator is powered during the frame reception. If the interrupts are disabled and buffer overflow is ignored, all incoming frames will be lost
1 (1)	Disabled	Enabled	System/all clocks waked-up on Receive Complete interrupt
1 (1)	Enabled	x	System/all clocks waked-up on UART Start Detection

Note: 1. The SLEEP instruction will not shut down the oscillator if on going communication.

20.10 Fractional baud rate generation

Fractional baud rate generation is possible for asynchronous operation due to the relatively high number of clock cycles for each frame. Each bit is sampled sixteen times, but only the three middle samples are of importance. The total number of samples for one frame is also relatively high. Given a 1-start, 8-data, no-parity, and 1-stop-bit frame format, and assuming that normal speed mode is used, the total number of samples for a frame is (1+8+1) × 16 or 160. As stated earlier, the UART can tolerate some variation in clock cycles for each sample. The critical factor is the time from the falling edge of the start bit (i.e., the clock synchronization) until the last bit's (i.e., the first stop bit’s) value is recovered.

Standard baud rate generators have the unwanted property of having large frequency steps between high baud rate settings. The worst case is found between the BSEL values 0x000 and 0x001. Going from a BSEL value of 0x000, which has a 10-bit frame of 160 clock cycles, to a BSEL value of 0x001, with 320 clock cycles, gives a 50% change in frequency. Ideally, the step size should be small even between the fastest baud rates. This is where the advantage of the fractional baud rate generator emerges.

In principle, the fractional baud rate generator works by doing uneven counting and then distributing the error evenly over the entire frame. A typical count sequence for an ordinary baud rate generator is:

2, 1, 0, 2, 1, 0, 2, 1, 0, 2, …

which has an even period time. A baud rate clock ticks each time the counter reaches zero, and a sample of the signal received on RxD is taken for every 16th baud rate clock tick.

For the fractional baud rate generator, the count sequence can have an uneven period:

2, 1, 0, 2, 1-1, 0, 2, 1, 0, 2, 1-1, 0, ...

In this example, an extra cycle is added to every second baud clock. This gives a baud rate clock tick jitter, but the average period has been increased by a fraction of 0.5 clock cycles.

Figure 20-9 on page 283 shows an example of how BSEL and BSCALE can be used to achieve baud rates in between what is possible by just changing BSEL.

The impact of fractional baud rate generation is that the step size between baud rate settings has been reduced. Given a scale factor of -1, the worst-case step then becomes from 160 to 240 clock cycles per 10-bit frame, compared to the previous step of from 160 to 320. A higher negative scale factor gives even finer granularity. There is a limit, however, to how high the scale factor can be. The value $2^{|BSCALE|}$ must be at most half the minimum number of clock cycles of a frame. For instance, for 10-bit frames, the minimum number of clock cycles is 160. This means that the highest applicable scale factor is -6 (2^{-6} = 64 < 160/2 = 80). For higher BSEL settings, the scale factor can be increased.

Figure 20-9. Fractional baud rate example.

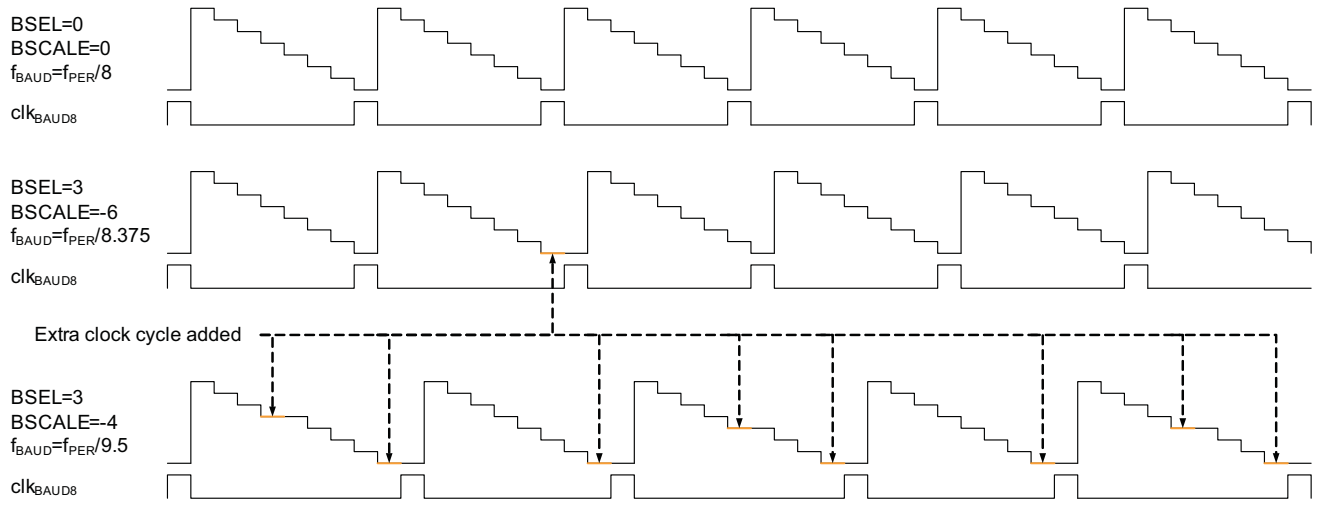


Table 20-6. USART baud rate.

Baud rate (bps)	f_{OSC} = 32.0000MHz					
	CLK2X = 0			CLK2X = 1		
	BSEL	BSCALE	Error [%]	BSEL	BSCALE	Error [%]
2400	12	6	0.2	12	7	0.2
4800	12	5	0.2	12	6	0.2
9600	12	4	0.2	12	5	0.2
14.4k	34	2	0.8	34	3	0.8
	138	0	-0.1	138	1	-0.1
19.2k	12	3	0.2	12	4	0.2
28.8k	34	1	-0.8	34	2	-0.8
	137	-1	-0.1	138	0	-0.1
38.4k	12	2	0.2	12	3	0.2
57.6k	34	0	-0.8	34	1	-0.8
	135	-2	-0.1	137	-1	-0.1
76.8k	12	1	0.2	12	2	0.2
115.2k	33	-1	-0.8	34	0	-0.8
	131	-3	-0.1	135	-2	-0.1
230.4k	31	-2	-0.8	33	-1	-0.8
	123	-4	-0.1	131	-3	-0.1
460.8k	27	-3	-0.8	31	-2	-0.8
	107	-5	-0.1	123	-4	-0.1
921.6k	19	-4	-0.8	27	-3	-0.8
	75	-6	-0.1	107	-5	-0.1
1.382M	7	-4	0.6	15	-3	0.6
	57	-7	0.1	121	-6	0.1
1.843M	3	-5	-0.8	19	-4	-0.8
	11	-7	-0.1	75	-6	-0.1
2.00M	0	0	0.0	1	0	0.0
2.304M	–	–	–	3	-2	-0.8
				47	-6	-0.1
2.5M	–	–	–	19	-4	0.4
				77	-7	-0.1

Baud	f_{OSC} = 32.0000MHz					
3.0M	–	–	–	11	-5	-0.8
				43	-7	-0.2
4.0M	–	–	–	0	0	0.0
Max	2.0Mbps			4.0Mbps		

20.11 USART in master SPI mode

Using the USART in master SPI mode requires the transmitter to be enabled. The receiver can optionally be enabled to serve as the serial input. The XCK pin will be used as the transfer clock.

As for the USART, a data transfer is initiated by writing to the DATA Register. This is the case for both sending and receiving data, since the transmitter controls the transfer clock. The data written to DATA are moved from the transmit buffer to the Shift Register when the Shift Register is ready to send a new frame.

The transmitter and receiver interrupt flags and corresponding USART interrupts used in master SPI mode are identical in function to their use in normal USART operation. The receiver error status flags are not in use and are always read as zero.

Disabling of the USART transmitter or receiver in master SPI mode is identical to their disabling in normal USART operation.

20.12 USART SPI vs. SPI

The USART in master SPI mode is fully compatible with the standalone SPI module in that:

- Timing diagrams are the same
- UCPHA bit functionality is identical to that of the SPI CPHA bit
- UDORD bit functionality is identical to that of the SPI DORD bit

When the USART is set in master SPI mode, configuration and use are in some cases different from those of the standalone SPI module. In addition, the following difference exists:

- The USART in master SPI mode does not include the SPI (Write Collision) feature

The USART in master SPI mode does not include the SPI double speed mode feature, but this can be achieved by configuring the baud rate generator accordingly:

- Interrupt timing is not compatible
- Pin control differs due to the master-only operation of the USART in SPI master mode

A comparison of the USART in master SPI mode and the SPI pins is shown in Table 20-7 on page 285.

Table 20-7. Prescaler options.

USART	SPI	Comment
TxD	MOSI	Master out only
RxD	MISO	Master in only
XCK	SCK	Functionally identical
N/A	$\overline{SS}$	Not supported by USART in master SPI mode

20.13 Multiprocessor communication mode

The multiprocessor communication mode 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. 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. When the frame type bit is one, the frame contains an address. When the frame type bit is zero, the frame is a data frame. If 5-bit to 8-bit character frames are used, the transmitter must be set to use two stop bits, since the first stop bit is used for indicating the frame type.

If a particular slave MCU has been addressed, it will receive the following data frames as usual, while the other slave MCUs will ignore the frames until another address frame is received.

20.13.1 Using multiprocessor communication mode

The following procedure should be used to exchange data in multiprocessor communication mode (MPCM):

1. All slave MCUs are in multiprocessor communication mode.
2. The master MCU sends an address frame, and all slaves receive and read this frame.
3. Each slave MCU determines if it has been selected.
4. The addressed MCU will disable MPCM and receive all data frames. The other slave MCUs will ignore the data frames.
5. When the addressed MCU has received the last data frame, it must enable MPCM again and wait for a new address frame from the master.

The process then repeats from step 2.

Using any of the 5-bit to 8-bit character frame formats is impractical, as the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult, since the transmitter and receiver must use the same character size setting.

20.14 One-wire mode

In this mode the TxD pin is connected to the RxD pin internally. If the receiver is enabled when transmitting it will receive what the transmitter is sending. This can be used to check that no one else is trying to transmit since received data will not be the same as the transmitted data.

20.15 Data encoding/decoding

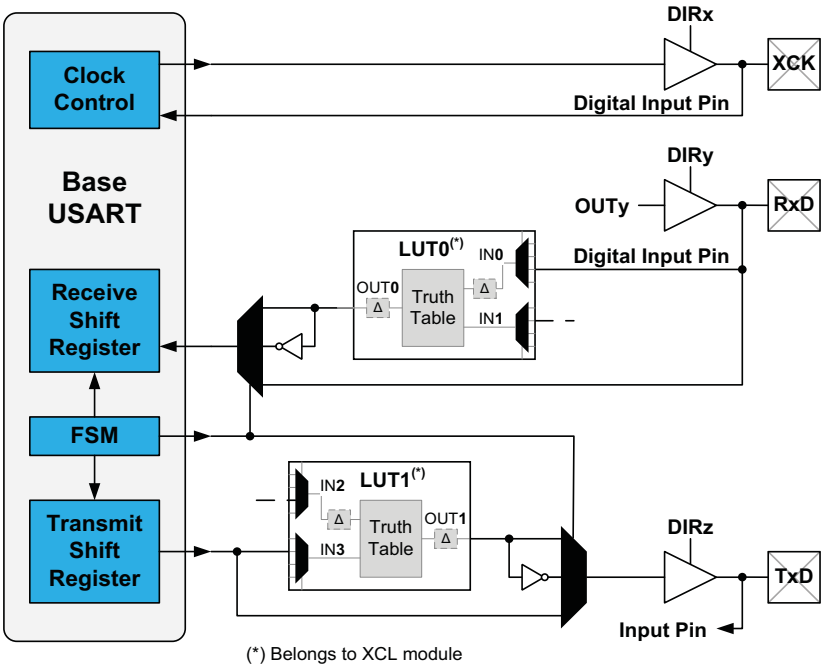
When this mode is used, the USART frame can be encoded or decoded using the LUT units in the XCL module, as shown in Figure 20-1 on page 272. For more details on how using and setting the LUT's, refer to "XCL – XMEGA Custom Logic" on page 300 module.

The USART can support independent encoding or decoding operation, each with dedicated lookup table. The USART implements different encoding and decoding types, but these options apply to both transmitter and receiver internal engines.

In transmission, and depending on encoding type settings, data sent to the pin is taken from the USART output or from XCL LUT1 output directly. In reception and depending on the decoding type settings, the data sent to the USART is taken directly from pin or from XCL LUT0 output.

For mode details on decoding/encoding types, refer to "CTRLD – Control register D" on page 293 description.

Figure 20-10.USART encoding/decoding scheme.



20.16 IRCOM mode of operation

IRCOM mode can be enabled to use the IRCOM module with the USART. This enables IrDA 1.4 compliant modulation and demodulation for baud rates up to 115.2kbps. When IRCOM mode is enabled, double speed mode cannot be used for the USART. For devices with more than one USART, IRCOM mode can be enabled for only one USART at a time, which is not linked with the XCL module. For details, refer to "IRCOM – IR Communication Module" on page 296.

20.17 EDMA support

EDMA support is available on UART, USART, and master SPI mode peripherals. For details on different USART EDMA transfer triggers, refer to "Transfer triggers" on page 53.

In variable data length receive mode, the data reception flag is used to trigger an EDMA transfer.

20.18 Register description

20.18.1 DATA – Data register

Bit	7	6	5	4	3	2	1	0
+0x00	RXB[7:0] / TXB[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register (DATA). The Transmit Data Buffer Register (TXB) will be the destination for data written to the DATA Register location. Reading the DATA Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6- or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the DREIF Flag in the STATUS Register is set. Data written to DATA when the DREIF Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. The data is then transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO and the corresponding flags in the Status Register (STATUS) will change state whenever the receive buffer is accessed (read). Always read STATUS before DATA in order to get the correct flags.

20.18.2 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x01	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	RXSIF	RXB8/DRIF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – RXCIF: Receive Complete Interrupt Flag**
  In standard mode, this flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). When the Receiver is disabled, the receive buffer will be flushed and consequently the RXCIF will become zero.
  When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from DATA in order to clear the 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 one to its bit location.
  In variable data length mode, this flag is generated when the valid STOP bit is detected. The DRIF status bit indicates if unread data in receive buffer is present or not. Since the RXCIF interrupt shares the interrupt address with the DRIF interrupt, RXCIF will not be cleared when the interrupt vector is executed. The flag is cleared by writing a one to its bit location.
- **Bit 6 – TXCIF: 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 (DATA). The TXCIF is automatically cleared when the transmit complete interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **Bit 5 – DREIF: Data Register Empty Flag**
  The DREIF indicates if the transmit buffer (DATA) is ready to receive new data. The flag is one when the transmit buffer is empty, and zero when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. DREIF is set after a reset to indicate that the Transmitter is ready. Always write this bit to zero when writing the STATUS Register.
  DREIF is cleared by writing DATA. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to DATA in order to clear DREIF or disable the Data Register Empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.
- **Bit 4 – FERR: Frame Error**
  The FERR flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The bit is set if the received character had a Frame Error, i.e. when the first stop bit was zero, and cleared when the stop bit of the received data is one. This bit is valid until the receive buffer (DATA) is read. The FERR is not affected by setting the SBMODE bit in CTRLC since the Receiver ignores all, except for the first stop bit. Always write this bit location to zero when writing the STATUS Register.
  This flag is not used in master SPI mode operation.
- **Bit 3 – BUFOVF: Buffer Overflow**
  The BUFOVF flag indicates data loss due to a receiver 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 characters), it is a new character waiting in the Receive Shift Register, a new start bit is detected in standard mode, or a new bit is received in variable data length mode. This flag is valid until the receive buffer (DATA) is read. Always write this bit location to zero when writing the STATUS Register.
  This flag is not used in master SPI mode operation.
- **Bit 2 – PERR: Parity Error**
  If parity checking is enabled and the next character in the receive buffer has a Parity Error this flag is set. If Parity Check is not enabled the PERR will always be read as zero. This bit is valid until the receive buffer (DATA) is read. Always write this bit location to zero when writing the STATUS Register. For details on parity calculation refer to “Parity bit calculation” on page 276.
  This flag is not used in master SPI mode operation.
- **Bit 1 – RXSIF: RX Start Flag**
  The RXSIF flag indicates a valid start condition on RxD line. The flag is set when the system is in power save, standby or extended standby modes and a high (IDLE) to low (START) valid transition is detected on the RxD line. If the start detection is not enabled, the RXSIF will always be read as zero.
  When interrupt-driven data reception is used, the receive complete interrupt routine must read the STATUS Register first. This flag can only be cleared by writing a one to its bit location.
  This flag is not used in master SPI mode operation.
- **Bit 0 – RXB8: Receive Bit 8**
  RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. When used, this bit must be read before reading the low bits from DATA. This bit is unused when peripheral counter controls the frame data length.
- **Bit 0 – DRIF: Data Reception Flag**
  This flag is set in variable data length mode only when there are unread data in the receive buffer. The flag is cleared when the receive buffer is empty (i.e., does not contain any unread data). When the Receiver is disabled, the receive buffer will be flushed and consequently the DRIF will become zero.
  Optionally an interrupt can be generated. The DRIF interrupt shares the interrupt address with the RXCIF interrupt. When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from DATA in order to clear the DRIF. If not, a new interrupt will occur directly after the return from the current interrupt. This flag can also be cleared by writing a one to its bit location.

20.18.3 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x02	RXSIE	DRIE	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – RXSIE: Receive Start of Frame Interrupt Enable**
  Setting this bit enables the start of frame interrupt. The interrupt level is controlled by receive complete interrupt level bits settings.
  The enabled interrupt will trigger for the conditions when RXSIF flag is set.
- **Bit 6 – DRIE: Data Reception Interrupt Enable**
  Setting this bit enables the data reception interrupt. The interrupt level is controlled by receive complete interrupt level bits settings.
  The enabled interrupt will trigger for the conditions when DRIF flag is set.
- **Bit 5:4 – RXCINTLVL[1:0]: Receive Complete Interrupt Level**
  These bits enable the Receive Complete Interrupt and select the interrupt level as described in“PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will be triggered when the RXCIF in the STATUS Register is set.
- **Bit 3:2 – TXCINTLVL[1:0]: Transmit Complete Interrupt Level**
  These bits enable the Transmit Complete Interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will be triggered when the TXCIF in the STATUS Register is set.
- **Bit 1:0 – DREINTLVL[1:0]: Data Register Empty Interrupt Level**
  These bits enable the Data Register Empty Interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will be triggered when the DREIF in the STATUS Register is set.

20.18.4 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x03	ONEWIRE	SFDEN	–	RXEN	TXEN	CLK2X	MPCM	TXB8
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – ONEWIRE: One-Wire Configuration Enabled**
  Setting this bit enables the USART TxD and RxD pins multiplexing, as described in “One-wire mode” on page 286.
- **Bit 6 – SFDEN: Start Frame Detection Enable**
  Writing this bit to one enables the USART Start Frame Detection mode. The start frame detector is able to wake up the system from power-save or standby sleep modes when a high (IDLE) to low (START) transition is detected on the RxDn line, as described in “Start frame detection” on page 282. The bit setting is ignored if the system is in the IDLE or ACTIVE modes. If the bit is set, the corresponding RXD pin has to be driven to avoid power consumption is deep sleep modes.
- **Bit 5 – Reserved**
  This bit is reserved and will always be read as zero. For compatibility with future devices, always write this bit zero when this register is written.
- **Bit 4 – RXEN: Receiver Enable**
  Setting this bit enables the USART Receiver. The receiver will override normal port operation for the RxD pin when enabled. Disabling the receiver will flush the receive buffer invalidating the FERR, BUFOVF, and PERR flags.
- **Bit 3 – TXEN: Transmitter Enable**
  Setting this bit enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. Disabling the Transmitter (writing TXEN to zero) will not become effective until ongoing and pend-

ing transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.

- **Bit 2 – CLK2X: Double Transmission Speed**
  Setting this bit will reduce the divisor of the baud rate divider from16 to 8 effectively doubling the transfer rate for asynchronous communication modes.
- **Bit 1 – MPCM: Multi-processor Communication Mode**
  This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, the USART Receiver ignores all the incoming frames that do not contain address information. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multiprocessor communication mode” on page 286.
- **Bit 0 – TXB8: Transmit Bit 8**
  TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. When used, this bit must be written before writing the low bits to DATA. This bit is ignored when peripheral counter controls the frame data length.

20.18.5 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x04	**CMODE[1:0]**		**PMODE[1:0]**		**SBMODE**	**CHSIZE[2:0]**		
+0x04 (1)	**CMODE[1:0]**		**–**	**–**	**–**	**UDORD**	**UCPHA**	**–**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Note: 1. Master SPI mode.

- **Bit 7:6 – CMODE[1:0]: Communication Mode**

These bits select the mode of operation of the USART as shown in Table 20-8.

Table 20-8. CMODE bit settings.

CMODE[1:0]	Group configuration	Mode
00	ASYNCHRONOUS	Asynchronous USART
01	SYNCHRONOUS	Synchronous USART
10	IRCOM	IRCOM (1)
11	MSPI	Master SPI (2)

Note: 1. See “IRCOM – IR Communication Module” on page 296 for full description on using IRCOM mode.
2. See “USART in master SPI mode” on page 285 for full description of master SPI operation.

- **Bit 5:4 – PMODE[1:0]: Parity Mode**
  These bits enable and set the type of parity generation according to Table 20-9. When enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and compare it to the PMODE setting and if a mismatch is detected, the PERR flag in STATUS will be set.

Table 20-9. PMODE bit settings.

PMODE[1:0]	Group configuration	Mode
00	DISABLED	Disabled
01	–	Reserved
10	EVEN	Enabled, Even Parity
11	ODD	Enabled, Odd Parity

- **Bit 3 – SBMODE: Stop Bit Mode**
  This bit selects the number of stop bits to be inserted by the transmitter according to Table 20-10 on page 292. The receiver ignores this setting.

Table 20-10. SBMODE bit settings.

SBMODE	Stop bit(s)
0	1-bit
1	2-bit

- **Bit 2:0 – CHSIZE[2:0]: Character Size**
  The CHSIZE[2:0] bits sets the number of data bits in a frame according to Table 20-11 on page 292. The receiver and transmitter use the same setting. The CHSIZE bits settings are ignored when peripheral counter controls the frame data length.

Table 20-11. CHSIZE bits settings.

CHSIZE[2:0]	Group configuration	Character size
000	5BIT	5-bit
001	6BIT	6-bit
010	7BIT	7-bit
011	8BIT	8-bit
100	–	Reserved
101	–	Reserved
110	–	Reserved
111	9BIT	9-bit

- **Bit 2 – UDORD: Data Order**
  This bit sets the frame format. When written to one, the lsb of the data word is transmitted first. When written to zero, the msb of the data word is transmitted first. The receiver and transmitter use the same setting. Changing the setting of UDORD will corrupt all ongoing communication for both receiver and transmitter. This bit is valid only for master SPI mode.
  In variable data length mode, this bit must be set to one.
- **Bit 1 – UCPHA: Clock Phase**
  The UCPHA bit setting determine whether data are sampled on the leading (first) edge or tailing (last) edge of XCKn. Refer to the "Master SPI mode clock generation" on page 275 for details.

20.18.6 CTRLD – Control register D

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	DECTYPE[1:0]		LUTACT[1:0]		PECACT[1:0]	
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits zero when this register is written.
- **Bit 5:4 – DECTYPE[1:0]: Decoding and encoding type**
  These bits decide the decoding and encoding type that is applied to both receiver and transmitter engines, as shown in Table 20-12. The settings are applied according to LUTACT settings.

Table 20-12. USART decoding and encoding types.

DECTYPE[1:0]	Group configuration	Description
00	DATA	LUT OUT applies during data field only
01	–	Reserved
10	SDATA	LUT OUT applies during start and data field
11	NOTSDATA	- Inverted LUT OUT applies during start field - LUT OUT applies during data field

- **Bit 3:2 – LUTACT[1:0]: LUT Action**
  These bits decide the action the USART performs when linked to LUT units from XCL module, according to Table 20-13.

Table 20-13. USART LUT action selection.

LUTACT[1:0]	Group configuration	Event action
00	OFF	Standard Configuration
01	RX	Enable decoding for on receiver engine
10	TX	Enable encoding on transmitter engine
11	BOTH	Enable both encoding/decoding

- **Bit 1:0 – PECACT[1:0]: Peripheral Counter Action**
  This bit decides the event action the USART performs on XCL PEC event, according to Table 20-14.

Table 20-14. USART peripheral counter action selection.

PECACT[1:0]	Group configuration	Event action
00	OFF	Standard Configuration
01	PEC0	Receiver data length controlled by peripheral counter 0
10	PEC1	Transmitter data length controlled by peripheral counter 1
11	PEC01	- Receiver data length controlled by peripheral counter 0 - Transmitter data length controlled by peripheral counter 1

20.18.7 BAUDCTRLA – Baud Rate Control register A

Bit	7	6	5	4	3	2	1	0
+0x06	BSEL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – BSEL[7:0]: Baud Rate Bits**
  This is a 12-bit value which contains the USART baud rate setting. The BAUDCTRLB contains the four most significant bits, and the BAUDCTRLA contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing BAUDCTRLA will trigger an immediate update of the baud rate prescaler.

20.18.8 BAUDCTRLB – Baud Rate Control register B

Bit	7	6	5	4	3	2	1	0
+0x07	BSCALE[3:0]				BSEL[11:8]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – BSCALE[3:0]: Baud Rate Scale Factor**
  These bits select the Baud Rate Generator scale factor. The scale factor is given in two's complement form from -7 (0b1001) to 7 (0b0111). The -8 (0b1000) setting is reserved. For positive scale values the Baud Rate Generator is prescaled by 2^{BSCALE}. For negative values the Baud Rate Generator will use fractional counting, which increases the resolution. See equations in Table 20-1 on page 274.
- **Bit 3:0 – BSEL[11:8]: Baud Rate Bits**
  This is a 12-bit value which contains the USART baud rate setting. The BAUDCTRLB contains the four most significant bits, and the BAUDCTRLA contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing BAUDCTRLA will trigger an immediate update of the baud rate prescaler.

20.19 Register summary

20.19.1 Register summary – USART

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DATA	DATA[7:0]								288
+0x01	STATUS	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	RXSIF	RXB8/DRIF	288
+0x02	CTRLA	RXSIE	DRIE	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		290
+0x03	CTRLB	ONEWIRE	SFDEN	–	RXEN	TXEN	CLK2X	MPCM	TXB8	290
+0x04	CTRLC	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]			291
+0x05	CTRLD	–	–	DECTYPE[1:0]		LUTACT[1:0]		PECACT[1:0]		293
+0x06	BAUDCTRLA	BSEL[7:0]								294
+0x07	BAUDCTRLB	BSCALE[3:0]				BSEL[11:8]				294

20.19.2 Register summary – USART in master SPI mode

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DATA	DATA[7:0]								288
+0x01	STATUS	RXCIF	TXCIF	DREIF	–	–	–	–	–	288
+0x02	CTRLA	–	–	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		290
+0x03	CTRLB	–	–	–	RXEN	TXEN	–	–	–	290
+0x04	CTRLC	CMODE[1:0]		–	–	–	UDORD	UCPHA	–	291
+0x05	CTRLD	–	–	DECTYPE[1:0]		LUTACT[1:0]		PECACT[1:0]		293
+0x06	BAUDCTRLA	BSEL[7:0]								294
+0x07	BAUDCTRLB	BSCALE[3:0]				BSEL[11:8]				294

20.20 Interrupt vector summary – USART

Table 20-15. USART Interrupt vectors and their word offset address.

Offset	Source	Interrupt description
0x00	RXC_vect	USART Receive Complete Interrupt vector
0x02	DRE_vect	USART Data Register Empty Interrupt vector
0x04	TXC_vect	USART Transmit Complete Interrupt vector

21. IRCOM – IR Communication Module

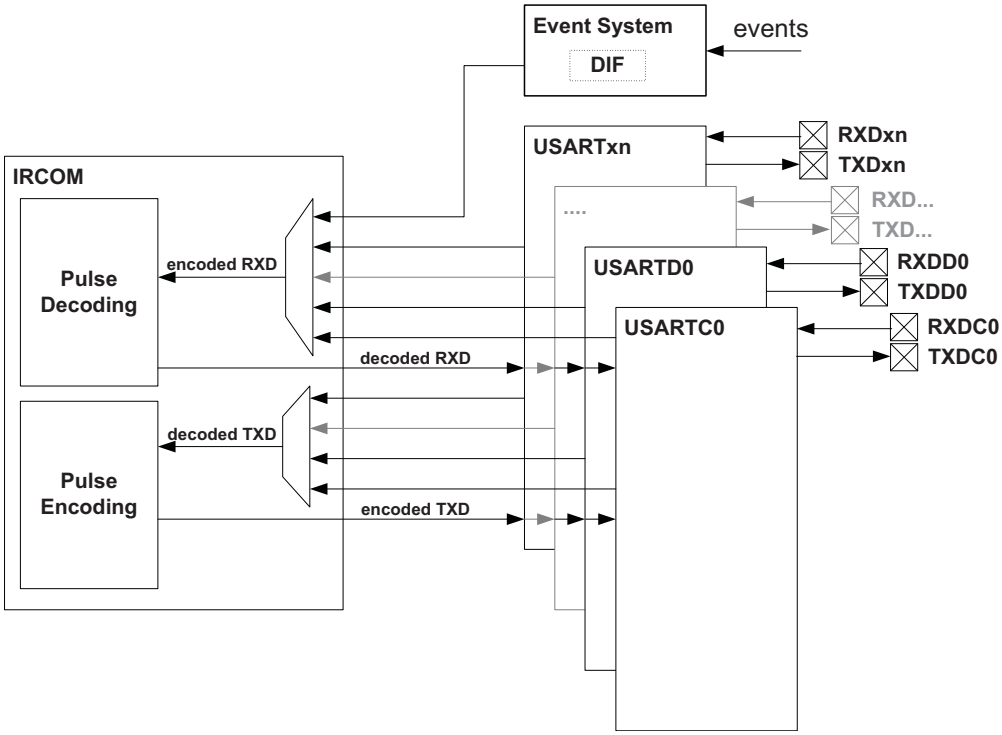
21.1 Features

- Pulse modulation/demodulation for infrared communication
- IrDA compatible for baud rates up to 115.2kbps
- Selectable pulse modulation scheme
  - 3/16 of the baud rate period
  - Fixed pulse period, 8-bit programmable
  - Pulse modulation disabled
- Built-in filtering
- Can be connected to and used by any USART

21.2 Overview

XMEGA devices contain an infrared communication module (IRCOM) that is IrDA compatible for baud rates up to 115.2kbps. It can be connected to any USART to enable infrared pulse encoding/decoding for that USART.

Figure 21-1. IRCOM connection to USARTs and associated port pins.



The IRCOM is automatically enabled when a USART is set in IRCOM mode. The signals between the USART and the RX/TX pins are then routed through the module as shown in Figure 21-1. The data on the TX/RX pins are the inverted value of the transmitted/received infrared pulse. It is also possible to select an event channel from the event system as input for the IRCOM receiver. This will disable the RX 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 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.

The module can only be used in combination with one USART at a time. Thus, IRCOM mode must not be set for more than one USART at a time. This must be ensured in the user software.

21.2.1 Event system filtering

The event system can be used as the receiver input. This enables IRCOM or USART input from I/O pins or sources other than the corresponding RX pin. If event system input is enabled, input from the USART's RX pin is automatically disabled. The event system has a digital input filter (DIF) on the event channels that can be used for filtering.

Refer to “Event System ” on page 79 for details on using the event system.

21.3 Registers description

21.3.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	EVSEL[3:0]			
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**

These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 3:0 – EVSEL [3:0]: Event Channel Selection**

These bits select the event channel source for the IRCOM receiver according to Table 21-1. If event input is selected for the IRCOM receiver, the input from the USART's RX pin is automatically disabled.

Table 21-1. Event channel selection.

EVSEL[3:0]	Group configuration	Event source
0000	–	None
0001	–	(Reserved)
0010	–	(Reserved)
0011	–	(Reserved)
0100	–	(Reserved)
0101	–	(Reserved)
0110	–	(Reserved)
0111	–	(Reserved)
1nnn	CHn	Event system channel n; n = {0, …,7}

21.3.2 TXPLCTRL – Transmitter Pulse Length Control register

Bit	7	6	5	4	3	2	1	0
+0x01	TXPLCTRL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – TXPLCTRL[7:0]: Transmitter Pulse Length Control**

This 8-bit value sets the pulse modulation scheme for the transmitter. Setting this register will have no effect if IRCOM mode is not selected by a USART.

By leaving this register value to zero, 3/16 of the baud rate period pulse modulation is used.

Setting this value from 1 to 254 will give a fixed pulse length coding. The 8-bit value sets the number of system clock periods for the pulse. The start of the pulse will be synchronized with the rising edge of the baud rate clock.

Setting the value to 255 (0xFF) will disable pulse coding, letting the 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.

TXPCTRL must be configured before the USART transmitter is enabled (TXEN).

21.3.3 RXPLCTRL – Receiver Pulse Length Control register

Bit	7	6	5	4	3	2	1	0
+0x02	RXPLCTRL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RXPLCTRL[7:0]: Receiver Pulse Length Control**

This 8-bit value sets the filter coefficient for the IRCOM transceiver. Setting this register will have no effect if IRCOM mode is not selected by a USART.

By leaving this register value at zero, filtering is disabled. Setting this value between 1 and 255 will enable filtering, where x+1 equal samples are required for the pulse to be accepted.

RXPCTRL must be configured before the USART receiver is enabled (RXEN).

21.4 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	EVSEL[3:0]				298
+0x01	TXPLCTRL	TXPLCTRL[7:0]								298
+0x02	RXPLCTRL	RXPLCTRL[7:0]								299

22. XCL – XMEGA Custom Logic

22.1 Features

- Two independent 8-bit timer/counter with:
  - Period or compare channel for each timer/counter
  - Input capture for each timer
  - Serial peripheral data length control for each timer
  - Timer underflow interrupt/event
  - Compare match or input capture interrupt/event for each timer
- One 16-bit timer/counter by cascading two 8-bit timer/counters with:
  - Period or compare channel
  - Input capture
  - Timer underflow interrupt/event
  - Compare match or input capture interrupt/event
- Programmable lookup table supporting multiple configurations:
  - Two 2-input units
  - One 3-input unit
  - RS configuration
  - Duplicate input with selectable delay on one input
  - Connection to external I/O pins or event system
- Combinatorial logic functions using programmable truth table:
  - AND, NAND, OR, NOR, XOR, XNOR, NOT, MUX
- Sequential logic functions:
  - D-Flip-Flop, D Latch, RS Latch
- Input sources:
  - From external pins or the event system
  - One input source includes selectable delay or synchronization option
  - Can be shared with selectable USART pin locations
- Outputs:
  - Available on external pins or event system
  - Includes selectable delay or synchronization option
  - Can override selectable USART pin locations
- Operates in all power modes

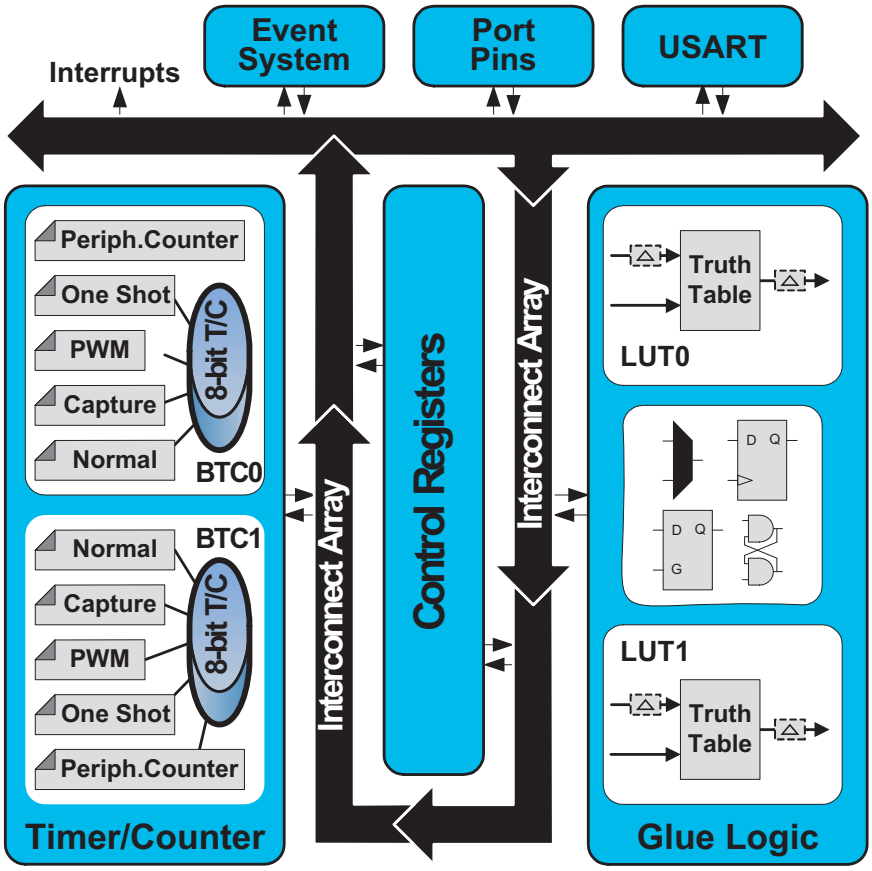
22.2 Overview

Atmel AVR XMEGA E devices include the XMEGA Custom Logic (XCL). The module consists of two main sub-units, timer/counter and glue logic.

- The timer/counter includes two 8-bit timer/counters BTCO and BTC1 respectively, allowing up to seven configuration settings. Both timer/counters can be cascaded to create a 16-bit timer/counter with optional 16-bit capture
- The glue logic is made of two truth tables with configurable delay elements and sequential logic functions such as D-type flip-flop or D-latch

An interconnect array enables a large amount of connections between a number of XCL elements and also allows working with other peripherals such as USART, port pins or event system.

Figure 22-1. XCL block diagram and closely related peripherals.



The timer/counter configuration allows for two 8-bits timer/counter usage. Each timer/counter supports normal, input capture, continuous and one shot pulse width modulations (PWM), with common flexible clock selections and event channels. By cascading the two 8-bit timer/counters, the XMEGA custom logic (XCL) offers a 16-bit timer/counter.

In peripheral counter (PEC) configuration, the XCL is directly linked to one of USART modules. The selected USART controls the counter operation, since the PEC can optionally control the data length within the USART frame.

If the glue logic configuration is enabled, the XCL implements two programmable lookup tables (LUT). Each LUT defines the truth table corresponding to the logical condition between two inputs. Any combinatorial function logic is possible.

The LUT inputs can be connected to I/O pins or to event system channels. If the LUT is connected to USART or SPI I/O pin locations (TxD/RxD/XCK or MOSI/MISO/SCK), serial data encoding/decoding is possible. Connecting together the LUT units, RS Latch or any combinatorial logic between two operands can be enabled.

A delay element (DLY) can be enabled. Each DLY has a 2-stage digital flip-flop. The position of the DLY is software selectable between either one input or the output. The size of the delay is software selectable, between 0-cycle delay (no delay), 1-cycle delay or 2-cycle delay configurations.

The LUT works in all sleep modes. Combined with event system and one I/O pin, the LUT can wake-up the system if condition on LUT inputs is true.

A block diagram of the programmable logic unit with extensions and closely related peripheral modules is shown in Figure 22-1 on page 301.

22.2.1 Definitions

Table 22-1 shows the definitions used throughout the documentation.

Table 22-1. Definitions of XMEGA custom logic.

Name	Description
XCL	XMEGA custom logic.
BTC	8-bit timer/counter.
PEC	Peripheral counter. Can work only with the serial peripheral module.
BOTTOM	The counter reaches BOTTOM when it becomes zero.
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. The TOP value can be equal to the period (PER) or the compare channel A (CCA) register setting. This is selected by the waveform generator mode.
UPDATE	The timer/counter signals an update when it reaches BOTTOM or TOP, depending on the waveform generator mode.
CLEAR	External peripheral, event system or CPU forces the (peripheral) timer/counter next value to BOTTOM.
CC	Compare or capture.
LUT	Lookup table including truth table register and decoder.
DLY	Delay element, created with a programmable number of flip-flops. Position is selectable by software, between one LUT input or LUT output
GLUE	Glue logic, including a LUT and DLY elements.

In general, the term “timer” is used when the timer/counter clock control is handled by an internal source, and the term “counter” is used when the clock control is handled externally (e.g. counting external events). When the CC channels are used for compare operations, they are referred to as “compare channels”. When used for capture operations, the CC channels are referred to as “capture channels”.

22.3 Timer/counter configuration

The XCL includes two 8-bit timer/counters. The two 8-bit timer/counters can be reconfirmed to work as:

- One 16-bit timer/counter
- One 8-bit timer/counter and one 8-bit peripheral counter
- Two 8-bit peripheral counters
- One 8-bit timer/counter and two 4-bit peripheral counters

Depending on the mode of operation, the timer/counter is reloaded or decremented at each timer/counter clock input.

22.4 Timer/counter operation

The XCL includes up to two identical 8-bit timer/counters, named BTC0 and BTC1 respectively. Each of 8-bit timer/counters has a compare channel.

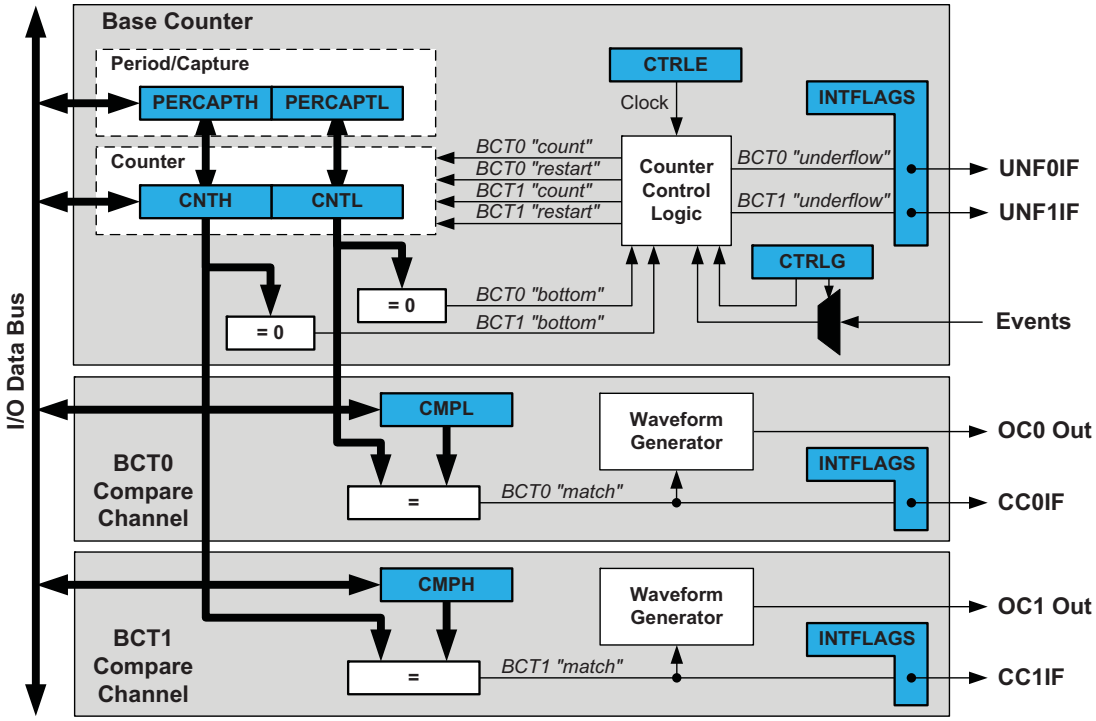
The two 8-bit timer/counters have a shared clock source and separate period and compare settings. They can be clocked and timed from the peripheral clock, with optional pre-scaling, or from the event system. The counters are always counting down.

When the XCL configuration is set to 16-bit timer/counter, BTC0 and BTC1 are cascaded to create a true 16-bit timer/counter (TC16) with 16-bit period, compare or capture registers.

A detailed block diagram of the timer/counters showing the base counter with its registers and the compare modules is shown in Figure 22-2.

The timer/counter will always operate in single slope decrementing mode. It will be counting down for each clock until it reaches BOTTOM and then reloads the counter with the period register value.

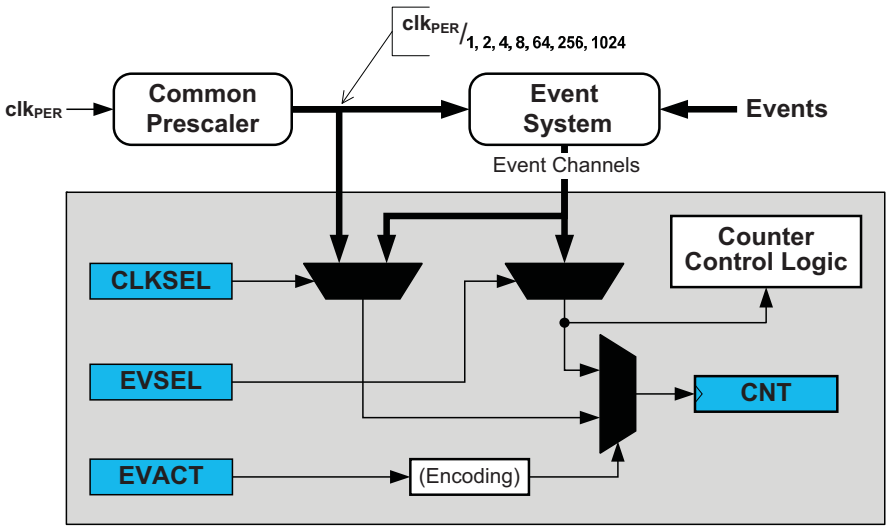
Figure 22-2. Timer/counter block diagram.



22.4.1 Clock sources

The timer/counters can be clocked from pre-scaled peripheral clock (clk_{PER}) and from the event system, Figure 22-3 shows the clock and event selection logic.

Figure 22-3. Clock and event selection.



The peripheral clock is fed into the common pre-scaler (common for all timer/counters in a device). A selection of the pre-scaler outputs is directly available for both timer/counters. In addition the whole range from 1 to 2^{15} times pre-scaling is available through the event system.

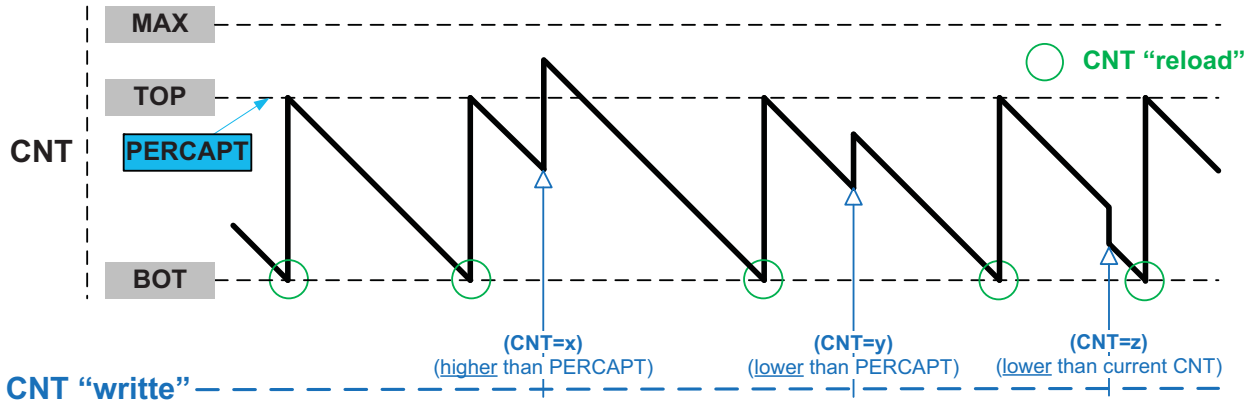
Clock selection (CLKSEL[3:0]) selects one of the pre-scaler outputs directly or an event channel as the counter (CNT) input. This is referred to as normal operation of the counter. For details, refer to the "Normal operation" on page 304. By using the event system, any event source, such as an external clock signal on any I/O pin, may be used as the clock input. In addition, the timer/counter can be controlled via the event system. The event selection (EVSEL[1:0]) and event action (EVACT[1:0]) settings are used to trigger an event action from one or more events. This is referred to as event action controlled operation of the counter. When event action controlled operation is used, the clock selection must be set to use an event channel as the counter input.

By default, no clock input is selected and the timer/counter is not running.

22.4.2 Normal operation

In normal mode, the timer/counter will always operate in single slope decrementing mode. The counter will be counting down for each clock until it reaches BOTTOM and then reloads the counter with the period register value.

Figure 22-4. Counter in normal operation.



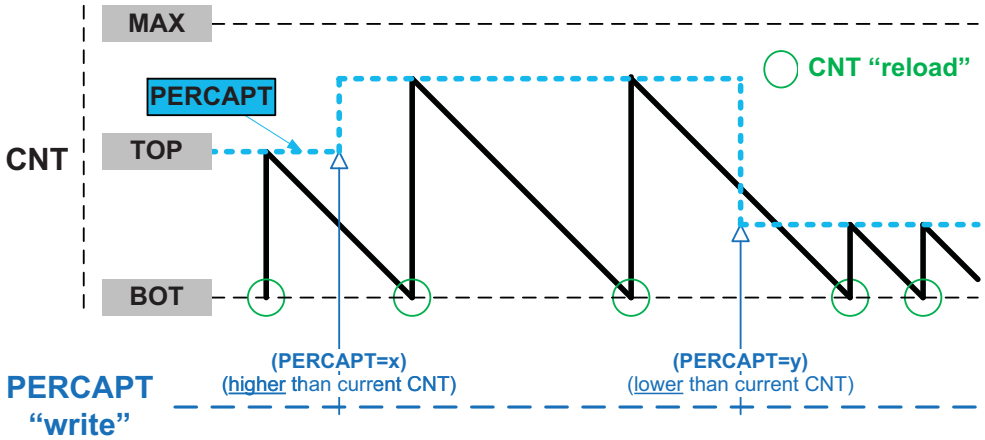
As shown in Figure 22-4, changing the counter value while the counter is running is possible. Write accesses have higher priority than count, clear, or reload and will be immediate.

22.4.2.1 Changing the period

The counter period is changed by writing a new TOP value to PERCAPT register.

Since the counter is counting down, PERCAPT register can be written at any time without affecting the current period as shown in Figure 22-5 on page 304. This prevents wraparound and generation of odd waveforms.

Figure 22-5. Changing the period.

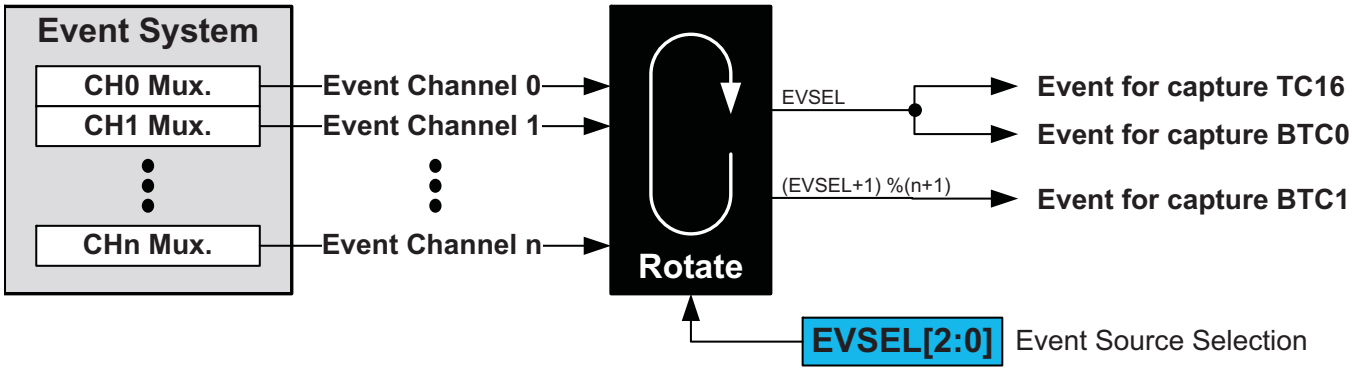


22.4.3 Capture operation

The PERCAPT register can be used as capture channel to capture external events and give them a time-stamp. To perform a capture operation, the timer/counter operation must be set in capture mode. In capture mode, the counter will count down every clock until it reaches BOTTOM and then it will be reloaded with the MAX value. The capture value will be stored in the same register as the period value (the period is fixed to MAX and doesn't need any more to be set in a register).

Events are used to trigger the capture; i.e., any events from the event system, including pin change from any device pin, can trigger a capture operation. The event source select setting selects which event channel will trigger the capture on BTC0. The subsequent event channel then trigger the capture on BTC1, if configured. For example, setting the event source select to event channel 2 results in BTC0 being triggered by event channel 2 and BTC1 triggered by event channel 3.

Figure 22-6. Event source selection for capture operation.



The event action setting in the timer/counter will determine the type of capture that is done. The capture operation must be enabled before capture can be done. When the capture condition occurs, the timer/counter will time-stamp the event by copying the current CNT value of the count register into the PERCAPT register.

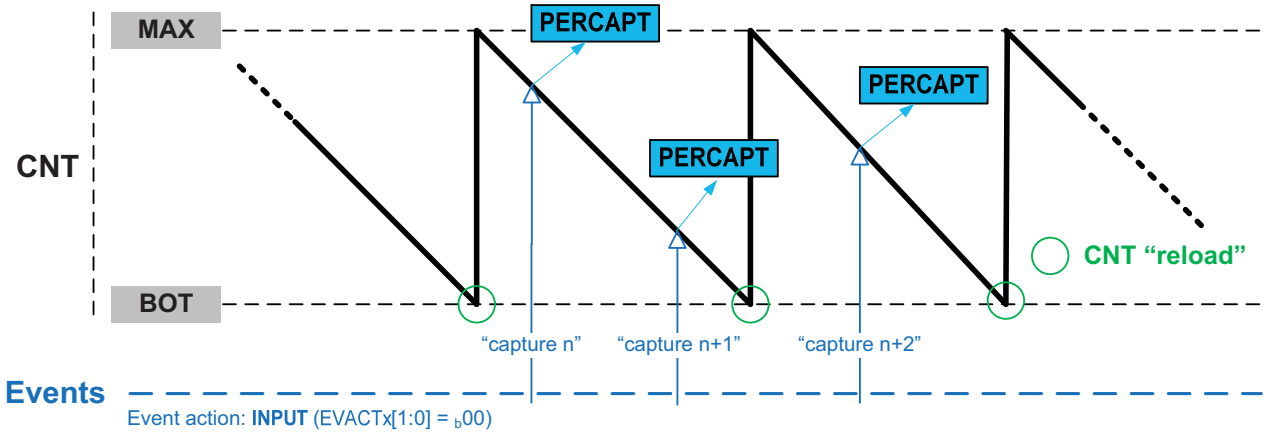
When an I/O pin is used as an event source for the capture, the pin must be configured for edge sensing. For details on sense configuration on I/O pins, refer to "Input sense configuration" on page 144.

22.4.3.1 Input capture event action

Selecting the input capture event action makes the enabled capture channel perform an input capture on an event. The interrupt flag (CCxIF) is set and indicates that there is a valid capture result in the corresponding PERCAPT register.

The counter will continuously count from MAX to BOTTOM, and then restart at MAX, as shown in Figure 22-7. The figure also shows three capture events for one capture channel.

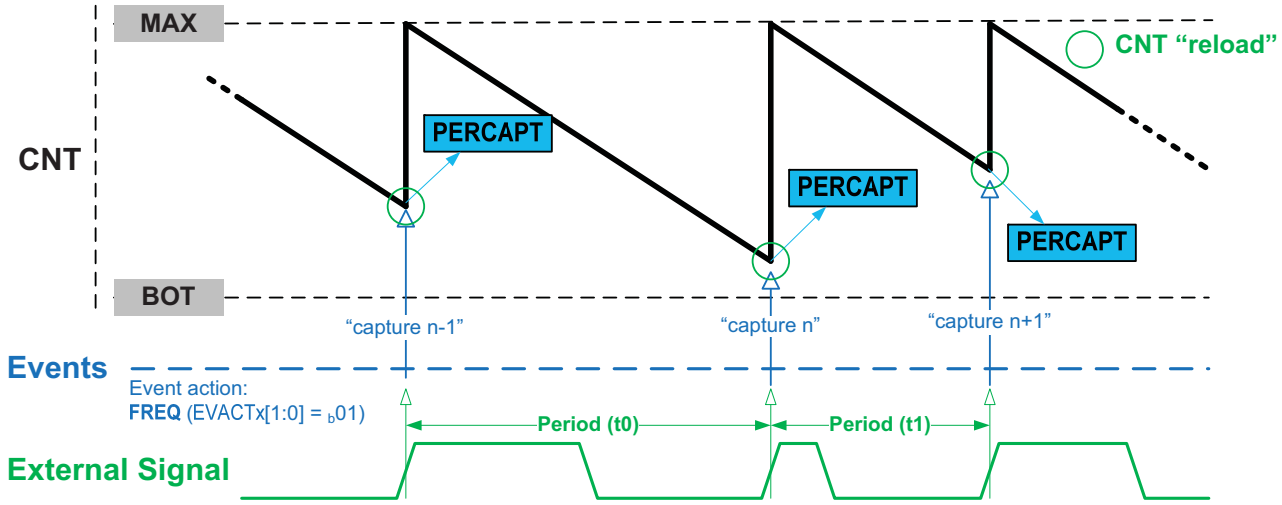
Figure 22-7. Timer/counter in CAPT configuration with INPUT command.



22.4.3.2 Frequency capture event action

Selecting the frequency capture event action makes the enabled capture timer/counter perform an input capture and restart on positive edge events. This enables the timer/counter to measure the period of a signal directly.

Figure 22-8. Timer/counter in CAPT configuration with FREQ command.



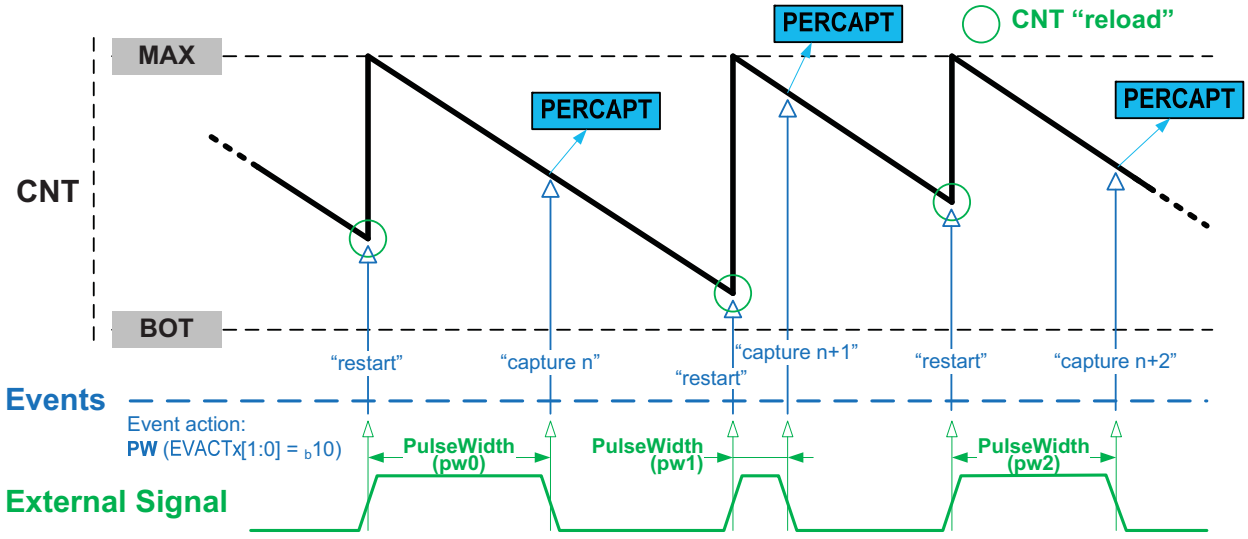
The capture result will be the time (t) from the previous timer/counter restart until the event occurred (MAX-PERCAPT).

Frequency (f): $f = \frac{1}{t}$

22.4.3.3 Pulse width capture event action

Selecting the pulse width measure event action, makes the enabled compare channel perform the input capture action on falling edge events and the restart action on rising edge events. The counter will then restart on positive edge events, and the input capture will be performed on the negative edge events. The event source must be an I/O pin, and the sense configuration for the pin must be set to generate an event on both edges. Figure 22-9 on page 306 shows and example where the high pulse width is measured three times for an external signal.

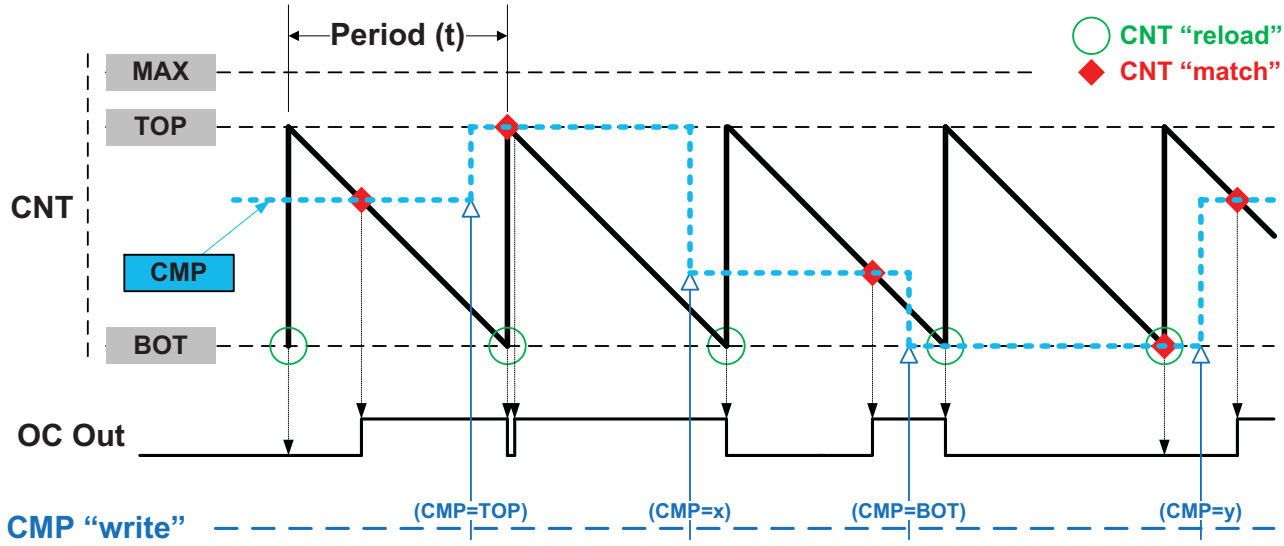
Figure 22-9. Timer/counter in CAPT configuration with PW command.



22.4.4 PWM generation

For PWM generation, the period time (t) is controlled by PERCAPT register, while the CMP register controls the duty cycle of the waveform generator (OC) output.-Figure 22-10 shows how the counter counts from TOP to BOTTOM, and then restarts from TOP. OC output is set on the compare match between the CNT and CMP register, and cleared at BOTTOM.

Figure 22-10.Single-slope pulse width modulation.



The PERCAPT register defines the PWM resolution. The minimum resolution is two bits (PERCAPT=0x03), and the maximum resolution is PERCAPT=MAX.

The following equation is used to calculate the exact resolution for a single-slope PWM (R_{PWM}) waveform:

$$R_{PWM} = \frac{\log(PERCAPT + 1)}{\log(2)}$$

The PWM frequency (f_{PWM}) depends on the period setting (PERCAPT) and the peripheral clock frequency ($f_{PERCAPT}$), and it is calculated by using the following equation:

$$f_{PWM} = \frac{f_{PERCAPT}}{N(PERCAPT + 1)}$$

Where N represents the prescaler divider used (1, 2, 4, 8, 64, 256, 1024, or event channel n).

When used in 16-bit configuration, the PERCAPT is automatically set to MAX value.

22.4.5 One-shot PWM generation

In one-shot PWM generation (1SHOT), the start and stop timer/counter operation is controlled by external events or software commands. If the operation is controlled by the external events, the event actions (EVACT) must be enabled and configured accordingly.

When timer/counter is enabled (CLKSEL), the counting operation starts only when software restart or an event is received. If no other command is provided to the timer/counter before the update condition is reached, the timer/counter will stop the operation and waits for a new command. The waveform generation is similar to PWM mode.

If the software restart or restart event command is provided before the update condition is detected, the timer/counter and waveform generation are restarted immediately.

If the stop event is provided before the update condition is detected, the timer/counter stops the operation immediately, the waveform is cleared and waits for a new command before starting again.

Figure 22-11 on page 308 and Figure 22-12 on page 308 show the timer/counter operation when software RESTART and STOPSTART commands are provided.

Figure 22-11.Timer/counter in 1SHOT configuration with RESTART command.

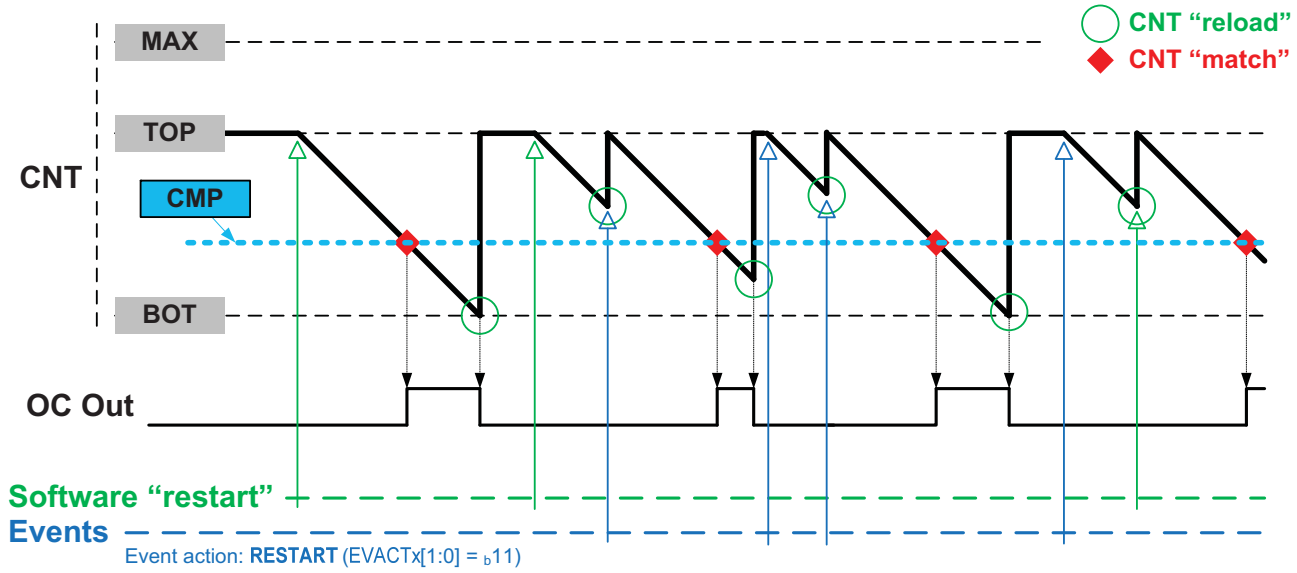
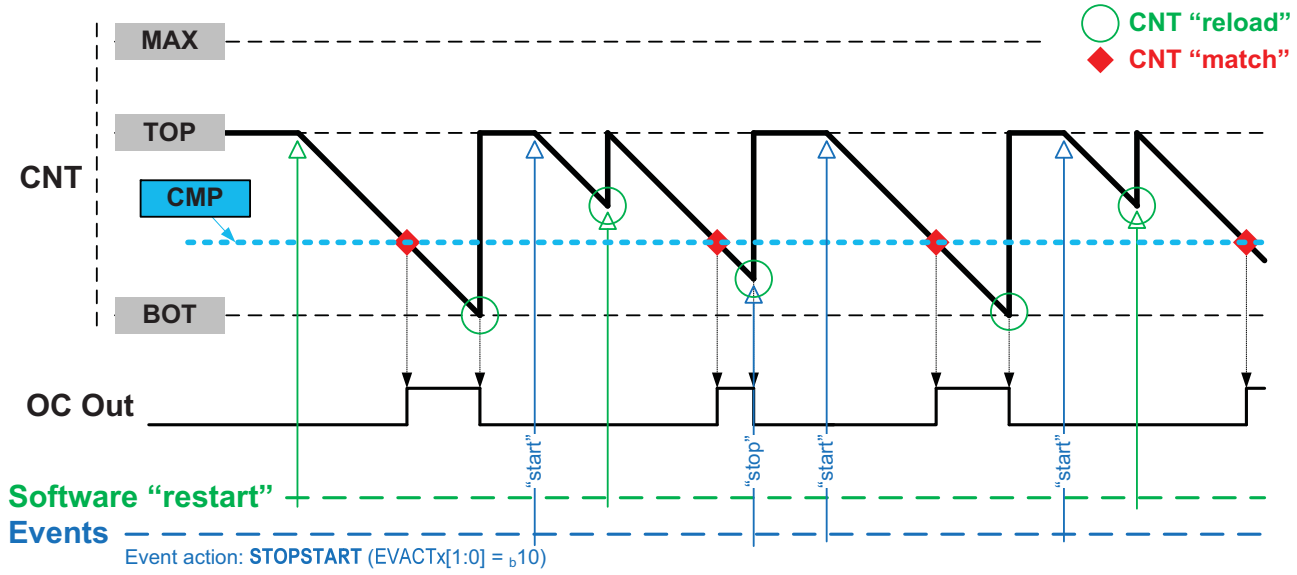


Figure 22-12.Timer/counter in 1SHOT configuration with STOPSTART command.



22.4.6 Timer/counter commands

A set of commands can be given to the timer/counter by software to immediately change the state of the module. These commands give direct control of the Restart signals.

The software can force a restart of the current waveform period by issuing a restart command. In this case the counter is set to TOP value and all compare outputs are set to zero.

22.4.7 16-bit operation

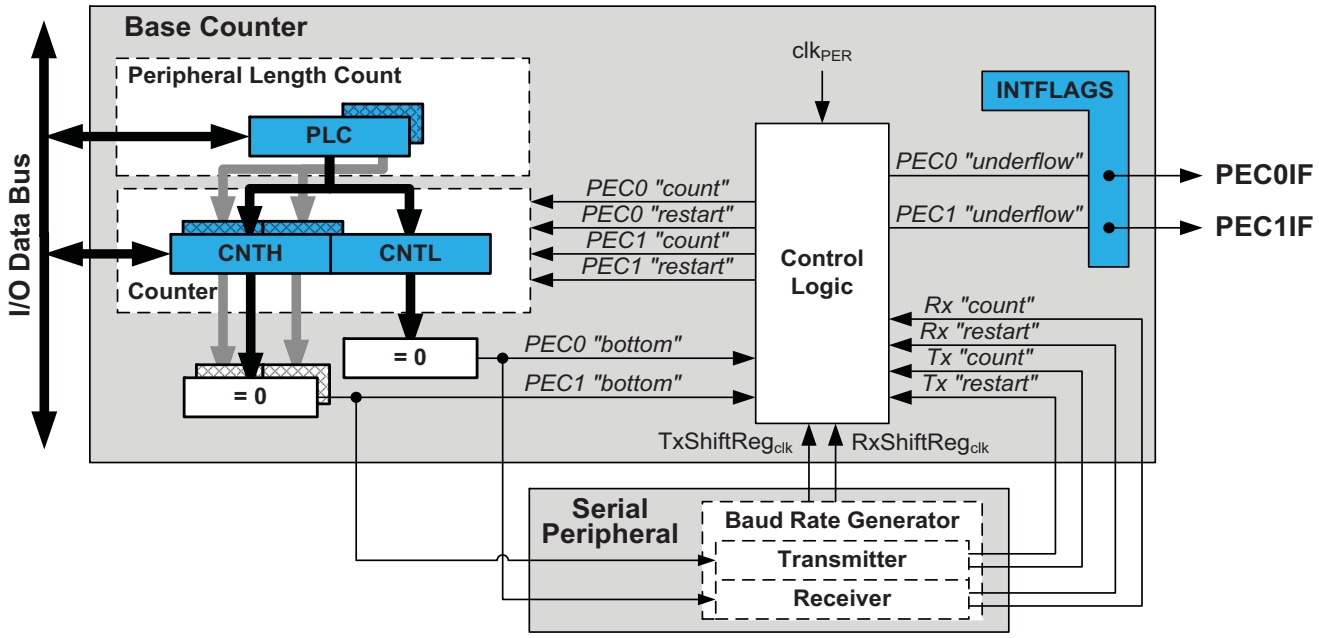
Both timer/counters can be cascaded to enable 16-bit counter operation. All timer/counter modes will be then available on a pure 16-bit timer/counter. Counter and output compare registers will be combined each into one 16-bit register.

In PWM and 1SHOT modes, the TOP of the count is always MAX.

22.4.8 Peripheral counter operation

A peripheral counter is used to customize an USART to be able to send or receive a frame with up to 256 bits. A detailed block diagram of the peripheral counters is shown in Figure 22-13 on page 309.

Figure 22-13.Peripheral counter block diagram.



In peripheral counter configuration (PEC), the peripheral length control register (PLC) represents the TOP value and the compare is always done with the BOTTOM.

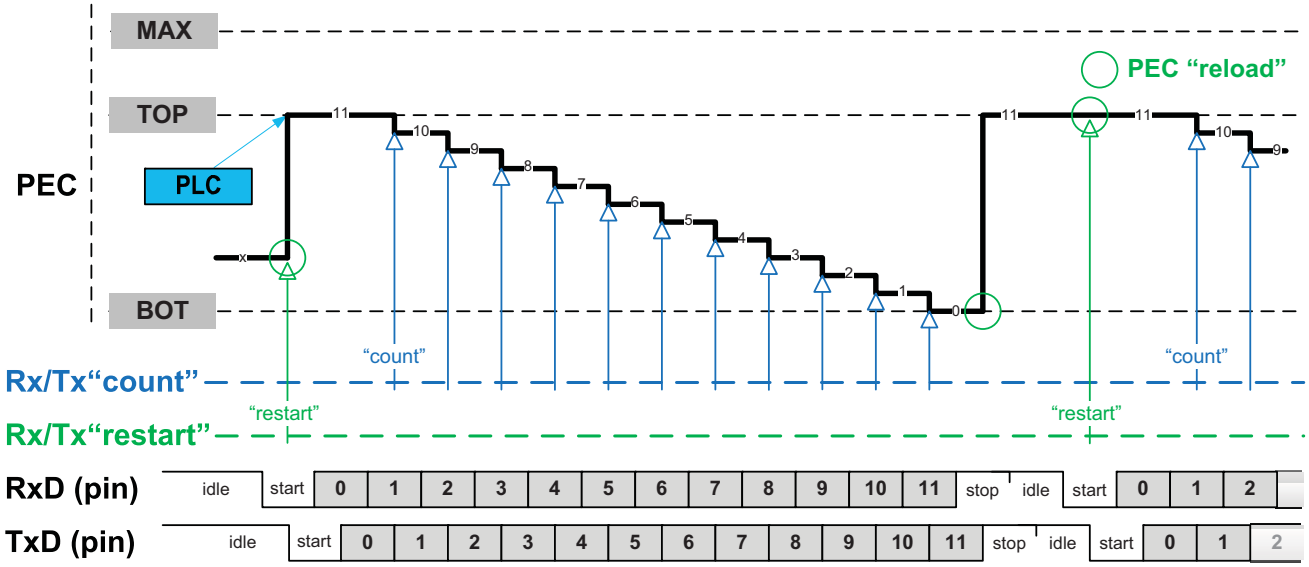
The data length within serial peripheral frame is defined using formula:

$$Frame_{Lenght} = PLC+1$$

The count (Rx/Tx ”count”) and restart (Rx/Tx ”restart”) commands are provided by each receiver and transmitter stage of the selected serial peripheral and all CLKSEL clock settings are ignored in PEC configuration. If only one PEC is used, the CLKSEL clock selection will be available for the other timer/counter.

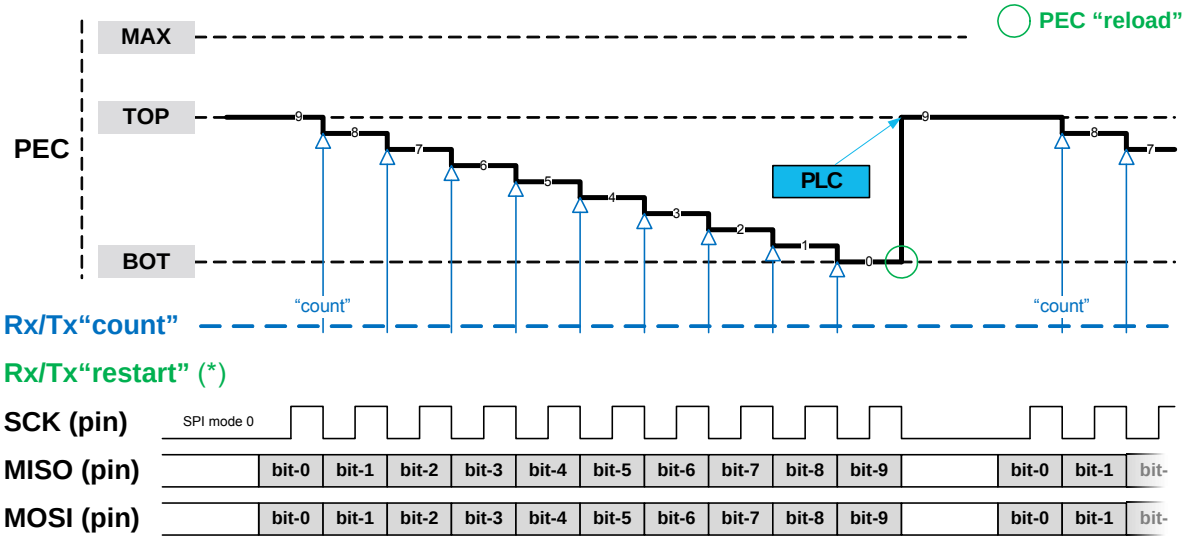
While counting, the counter restarts from PLC if the restart command is received. When it reaches the BOTTOM value, the counter restarts from PLC if restart or count command is received from the serial peripheral.

Figure 22-14.Peripheral counter with UART/USART 12-bit frame.



Since the commands are provided by the serial peripheral, the event actions are ignored. Only software restart command is available.

Figure 22-15.Peripheral counter with SPI 10-bit frame.



(*) No Rx/Tx"restart": During PEC initialization for SPI communication, set-up PEC=PLC.

The XCL supports up to two 8-bit peripheral counters, called PEC0 and PEC1 respectively.The receiver stage of serial peripheral controls PEC0 operation, since the transmitter stage of the serial peripheral controls the PEC1 operation.

The XCL also supports two 4-bit peripheral counters, called PEC20 and PEC21 respectively. Both are mapped in one 8-bit T/C. The receiver stage of serial peripheral controls PEC20 operation, since the transmitter stage of the serial peripheral controls the PEC21 operation. This lets the opportunity to control serial frames up to 16 bits in reception and in transmission, and still to have an 8-bit timer/counter available resource.

Cascading two peripheral counters is not available.

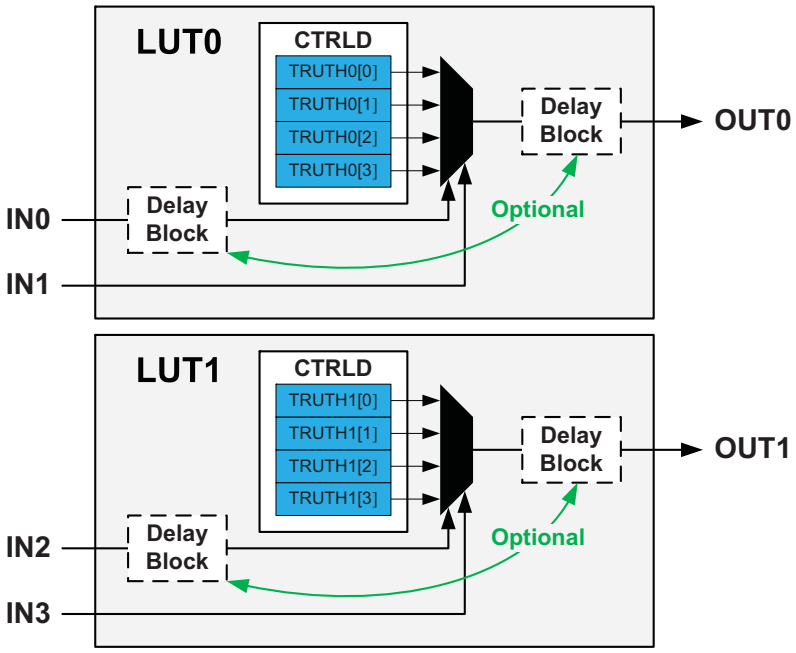
22.5 Glue logic

The glue logic is made of two lookup tables and some sequential logic functions such as D-type flip-flop or D-latch. The lookup tables can be cascaded to extend the input selection or to create advanced logic.

22.5.1 LUT description

The XCL includes two lookup table units (LUT). Each LUT is composed by a 4-bit truth table and a decoder, allowing generation of any logic expression OUT as a function of two inputs, as shown in Figure 22-16.

Figure 22-16.LUT units block diagram..



The combinatorial logic functions can be: AND, NAND, OR, NOR, XOR, XNOR, NOT.

The truth table for these functions is written to the CTRLD register of the LUT. Table 22-2 shows both truth table for 2-input LUT units.

Table 22-2. 2-input LUT truth table (2LUTxIN configuration).

IN1	IN0	OUT0
0	0	TRUTH0[0]
0	1	TRUTH0[1]
1	0	TRUTH0[2]
1	1	TRUTH0[3]

IN3	IN2	OUT1
0	0	TRUTH1[0]
0	1	TRUTH1[1]
1	0	TRUTH1[2]
1	1	TRUTH1[3]

Cascading both LUT units, the logic function OUT can be a function of three inputs. Table 22-3 shows the truth table for a 3-input LUT.

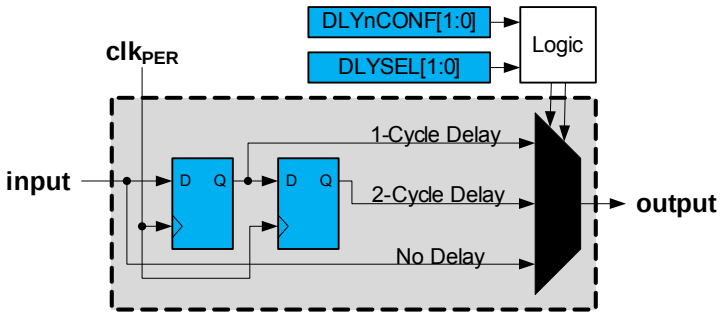
Table 22-3. 3-input LUT truth table (1LUT3IN configuration).

IN0	IN3	IN2	OUT1		OUT0
0	0	0	TRUTH1[0]	=0	TRUTH0[0]
				=1	TRUTH0[1]
0	0	1	TRUTH1[1]	=0	TRUTH0[0]
				=1	TRUTH0[1]
0	1	0	TRUTH1[2]	=0	TRUTH0[0]
				=1	TRUTH0[1]
0	1	1	TRUTH1[3]	=0	TRUTH0[0]
				=1	TRUTH0[1]
1	0	0	TRUTH1[0]	=0	TRUTH0[2]
				=1	TRUTH0[3]
1	0	1	TRUTH1[1]	=0	TRUTH0[2]
				=1	TRUTH0[3]
1	1	0	TRUTH1[2]	=0	TRUTH0[2]
				=1	TRUTH0[3]
1	1	1	TRUTH1[3]	=0	TRUTH0[2]
				=1	TRUTH0[3]

22.5.2 Delay description

The XCL has two delay units, one for each LUT. The delay can be configured from zero (no delay) and up to two peripheral clock cycles delay, as shown in Figure 22-17 on page 312.

Figure 22-17.Delay block diagram.



The insertion of a delay unit is selectable for each LUT, on the first of the two inputs or on the output. The insertion is decided by the application purpose, but some examples are provided:

- Delay on input can be used as input synchronizer or as edge detector on input signal
- Delay on output can be used to filter glitches or to synchronize the LUT output when both inputs are asynchronous

22.5.3 Glue logic configurations

Figure 22-18 to Figure 22-25 on page 314 show the different glue logic configurations. Dashed boxes show the possible places of the delay element.

Figure 22-18.Two independent 2-input LUT (2LUT2IN).

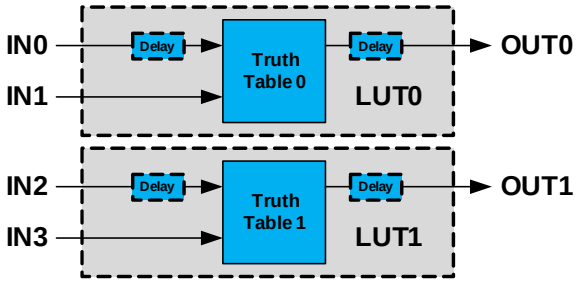


Figure 22-19.Two independent 2-input LUT with duplicated input (2LUT1IN).

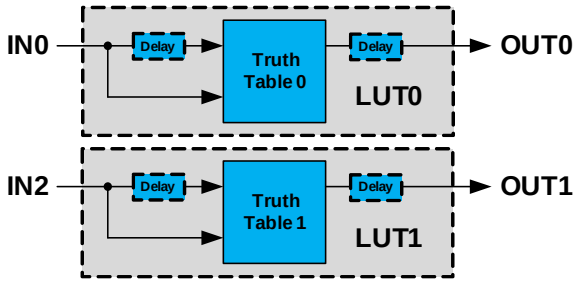


Figure 22-20.Two 2-input LUT with one common input (2LUT3IN).

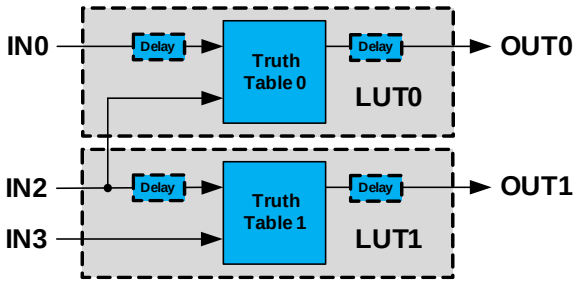


Figure 22-21.One 3-input LUT (1LUT3IN).

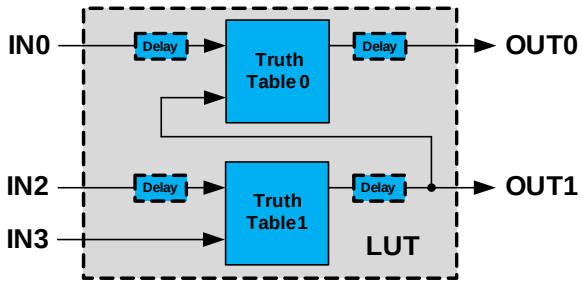


Figure 22-22.One 2-input multiplexer controlled by one 2-input LUT (MUX).

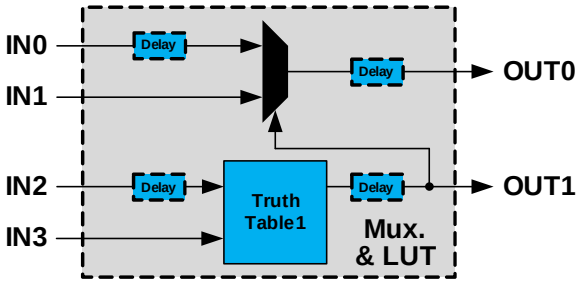


Figure 22-23.One D-Latch controlled by two 2-input LUT (DLATCH).

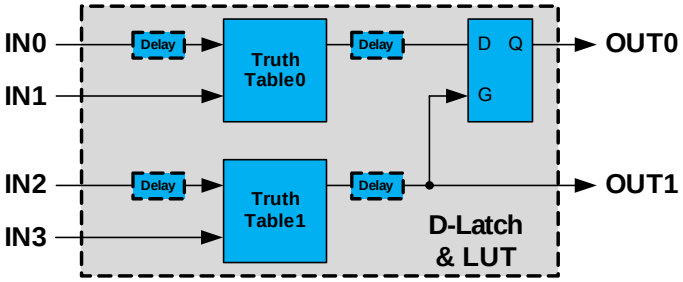


Figure 22-24.One RS-Latch LUT (RSLATCH).

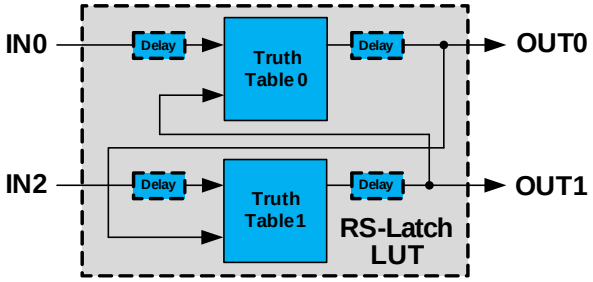
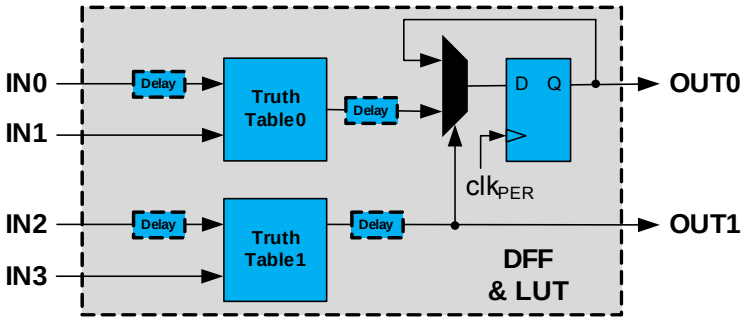


Figure 22-25.One DFF with data controlled by two independent 2-input LUT (DFF).

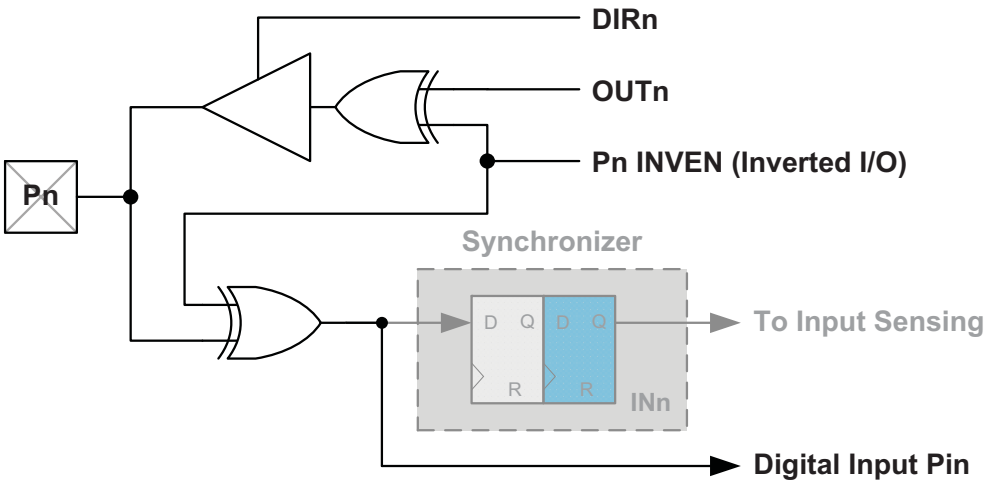


22.5.4 Glue input and output description

The input selection includes selection from I/O pins, from event system or from internal XCL sub-modules, including TxD line from the selectable USART module and timer/counter outputs.

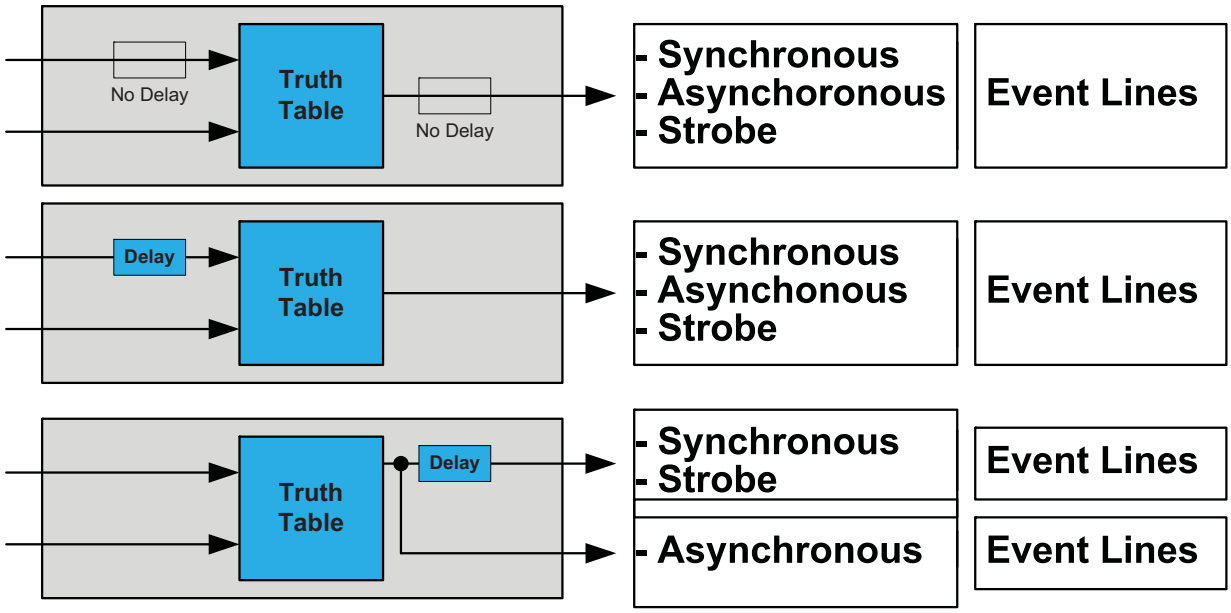
When used with I/O pins, the LUT inputs are connected to digital input pins, as shown in Figure 22-26 on page 315. For more details, refer to "I/O Ports" on page 139.

Figure 22-26.Input connection.



The LUT outputs are connected to all strobe, asynchronous and synchronous event system data lines, as shown in Figure 22-7. The connection depends on delay configuration, but it is up to the application to generate the correct waveform or to use the correct event line.

Figure 22-27.LUT output connection.



INxSEL bits in CTRLB register decide the source of each input pin for each LUT0 and LUT1. Table 22-4 on page 316 shows the input selections for the LUT units. PORTSEL bits in CTRLA register select the port associated to LUT0 and LUT1.

Table 22-4. LUT input pin location selection.

INxSEL[1:0]		IN3 (1)	IN2 (1)	IN1 (1)	IN0 (1)
Event system	EVSYS	CH7	CH1	CH0	CH6
XCL	XCL	TxD	OC0 out	OC1 out (2)	OC0 out
I/O pin low	PINL	PIN3	PIN1	PIN0	PIN2
I/O pin high	PINH	PIN7	PIN5	PIN4	PIN6

Notes: 1. Figure 22-18 on page 313 to Figure 22-25 on page 314 show the active inputs.
2. In TC16 configuration, IN1 is the 16-bit Waveform Generation (OC0 out).

EVASYSELn bits in CTRLC register decides if the event system channel line selection is the strobe or the asynchronous event line.

Only the LUT0 output can be connected to I/O pin. LUT0OUTEN bit in CTRLA register allows two LUT0 output pin locations: PIN0 or PIN4.

Table 22-5 shows the consumers of the LUT outputs.

Table 22-5. LUT output consumers.

OUT1	OUT0	
Event Channel	Event Channel	
No I/O output	PIN0	LUT0OUTEN[1:0] = $_b$01
	PIN4	LUT0OUTEN[1:0] = $_b$10
Inserted in TxD (MOSI) encoding logic of the selected USART (1)	Directly to RxD (MISO) input of the selected USART (1)	

Note: 1. Refer to "CTRLD – Control register D" on page 293 in USART.

22.6 Interrupts and events

The XCL can generate both interrupts and events.

Each timer/counter can generate an interrupt on underflow, but the interrupt line is shared between timer/counter0 and timer/counter1.

The CC channel has a separate interrupt that is used for compare or capture, but the interrupt line is shared between BTC0 and BTC1.

timer/counter events will be generated for all conditions that can generate interrupts. For details on event generation and available events refer to "Event System " on page 79. The glue logic generates only events. For details, refers to "Glue input and output description" on page 314.

22.7 Register description

22.7.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	LUT0OUTEN[1:0]		PORTSEL[1:0]		–	LUTCONF[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – LUT0OUTEN[1:0]: LUT0 Output Enable**
  Setting these bits enable LUT0 output to pin, according to table Table 22-6.

Table 22-6. LUT output pin selection.

LUT0OUTEN[1:0]	Group configuration	Description
00	DISABLE	LUT0 output disabled
01	PIN0 (1)	LUT0 output to PC0
		LUT0 output to PD0
10	PIN4 (1)	LUT0 output to PC4
		LUT0 output to PD4
11	-	Reserved

Note: 1. The port is defined by PORTSEL settings.

- **Bit 5:4 – PORTSEL[1:0]: Port Selection**
  These bits select from which port I/O pins are used as input/output for LUT(s) or USART module used with PEC, according to Table 22-7.

Table 22-7. Port source selection.

PORTSEL[1:0]	Group configuration	Description	
00	PC	LUT(s)	Port C
		PEC	USARTC0
01	PD	LUT(s)	Port D
		PEC	USARTD0
1x	-	Reserved	

- **Bit 3 - Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – LUTCONF[2:0]: LUT Configuration**
  Setting these bits enables the configuration of the glue logic cells, according to Table 22-8 on page 318.

Table 22-8. LUT configuration mode.

LUTCONF[2:0]	Group configuration	Description
000	2LUT2IN	Two independent 2-input LUT
001	2LUT1IN	Two independent 2-input LUT with duplicated input
010	2LUT3IN	Two LUT with one common input
011	1LUT3IN	One 3-input LUT
100	MUX	One 2-input multiplexer controlled by one 2-input LUT
101	DLATCH	One D-Latch controlled by two 2-input LUT
110	RSLATCH	One RS-Latch LUT
111	DFF	One DFF with data controlled by two independent 2-input LUT

22.7.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	IN3SEL[1:0]		IN2SEL [1:0]		IN1SEL [1:0]		IN0SEL [1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:0 – INxSEL[1:0]: Input Selection**
  These bits decide source of input pins of LUT0 and LUT1, according to Table 22-9 on page 318. Table 22-4 on page 316 also shows LUT input pin location selection and Table 22-5 on page 316 shows LUT output consumers.

Table 22-9. LUT input pins source selection.

INxSEL[1:0]	Group configuration	Description
00	EVSYS	Event system selected as source
01	XCL	XCL selected as source
10	PINL(1)	LSB port pins selected as source
11	PINH(1)	MSB port pins selected as source

Note: 1. The Port is defined by PORTSEL[1:0] settings.

22.7.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	EVASYSEL1	EVASYSEL0	DLYSEL [1:0]		DLY1CONF[1:0]		DLY0CONF[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 - EVASYSEL1: LUT1 Asynchronous Event Line Selection**
  Setting this bit selects the asynchronous event line as possible input for LUT1. When cleared, the event strobe line can be selected as input for LUT1.
- **Bit 6 - EVASYSEL0: LUT0 Asynchronous Event Line Selection**
  Setting this bit selects the asynchronous event line as possible input for LUT0. When cleared, the event strobe line can be selected as input for LUT0.

- **Bit 5:4 – DLYSEL[1:0]: Delay Selection**
  These bits define the configuration of the delay logic cells, according to Table 22-10 on page 319.

Table 22-10. Delay selection.

DLYSEL[1:0]	Group configuration	Description
00	DLY11	1-cycle delay for both LUT1 and LUT0
01	DLY12	1-cycle delay for LUT1 and 2-cycle delay for LUT0
10	DLY21	2-cycle delay for LUT1 and 1-cycle delay for LUT0
11	DLY22	2-cycle delay for both LUT1 and LUT0

- **Bit 3:2 – DLY1CONF[1:0]: Delay Configuration on LUT1**
  These bits define the delay configuration for LUT1, according to Table 22-11 on page 319.
- **Bit 1:0 – DLY0CONF[1:0]: Delay Configuration on LUT0**
  These bits define the delay configuration for LUT0, according to Table 22-11 on page 319.

Table 22-11. Delay configuration.

DLYxCONF[1:0]	Group configuration	Description
00	DISABLE	No delay
01	IN (1)	Delay enabled on LUT input
10	OUT (1)	Delay enabled on LUT output
11	-	Reserved

Note: 1. Figure 22-18 on page 313 to Figure 22-25 on page 314 show possible location of delay elements.

22.7.4 CTRLD – Control register D

This register defines the truth tables for LUT0 and LUT1 units, as described in "LUT description" on page 311.

Bit	7	6	5	4	3	2	1	0
+0x03	TRUTH1[3:0]				TRUTH0[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:4 – TRUTH1[3:0]: LUT1 Truth Table**
  These bits hold the truth table definition for LUT1.
- **Bit 3:0 – TRUTH0[3:0]: LUT0 Truth Table**
  These bits hold the truth table definition for LUT0.

22.7.5 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x04	CMDSEL	TCSEL[2:0]			CLKSEL[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – CMDSEL: Command Selection**
  This command bit is used for software control of timer/counter restart, according to Table 22-12 on page 320. The command bit is always read as zero. The CMD bit must be used together with CMDEN.

Table 22-12. Command selection.

CMDSEL	Group configuration	Description
0	NONE	None
1	RESTART	Force count restart

- **Bit 6:4 – TCSEL[2:0]: Timer/Counter Selection**
  Setting these bits enables the configuration of the timer/counters, according to Table 22-13 on page 320.

Table 22-13. Timer/counter selection.

TCSEL[2:0]	Group configuration	Description
000	TC16	16-bit timer/counter
001	BTC0	One 8-bit timer/counter (with period)
010	BTC01	Two 8-bit timer/counters
011	BTC0PEC1	One 8-bit timer/counter with period and one 8-bit transmitter peripheral counter
100	PEC0BTC1	One 8-bit timer/counter with period and one 8-bit receiver peripheral counter
101	PEC01	Two 8-bit transmitter/receiver peripheral counter
110	BTC0PEC2	One 8-bit timer/counter with period and two 4-bit transmitter/receiver peripheral counter
111	-	Reserved

- **Bit 3:0 – CLKSEL[3:0]: Clock Select**
  Setting these bits enables the input clock of the timer/counters, according to Table 22-14 on page 320.

Table 22-14. Clock select options.

CLKSEL[3:0]	Group configuration	Description
0000	OFF	Prescaler: OFF
0001	DIV1	Prescaler: clk_{PER}
0010	DIV2	Prescaler: $clk_{PER}/2$
0011	DIV4	Prescaler: $clk_{PER}/4$
0100	DIV8	Prescaler: $clk_{PER}/8$

CLKSEL[3:0]	Group configuration	Description
0101	DIV64	Prescaler: $clk_{PER}/64$
0110	DIV256	Prescaler: $clk_{PER}/256$
0111	DIV1024	Prescaler: $clk_{PER}/1024$
1nnn	EVCHn	Event channel n, n={0,. ., 7)

22.7.6 CTRLF – Control register F

Bit	7	6	5	4	3	2	1	0
+0x05	CMDEN[1:0]		CMP1	CMP0	CCEN1	CCEN0	MODE[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7:6 – CMDEN[1:0]: Command Enable**
  These bits are used to indicate for which timer/counter the command (CMD) is valid, according to Table 22-15 on page 321.

Table 22-15. Command selections.

CMD[1:0]	Group configuration	Description
00	DISABLE	Command ignored.
01	CMD0	Command valid for BTC0
10	CMD1	Command valid for BTC1
11	CMD01	Command valid for both BTC0 and BTC1

- **Bit 5:4 – CMPx: Compare Output Value**
  These bits allow direct access to the waveform generator's output compare value when the timer/counter is set in the OFF state. This is used to set or clear the WG output value when the timer/counter is not running.
- **Bit 3:2 – CCENx: Compare or Capture Enable**
  Setting these bits in the compare or PWM waveform generation mode of operation will override the port output register for the corresponding OCn output pin.
  When input capture operation is selected, the CCxEN bits enable the capture operation for the corresponding CC channel.
- **Bit 1:0 – MODE[1:0]: Operation Mode**
  This bit selects the operation mode for the timer/counter according to Table 22-16 on page 321. The clock select and operation mode is identical to both BTC0 and BTC1.

Table 22-16. Operation mode.

MODE[1:0]	Group configuration	Description
00	NORMAL	Normal mode
01	CAPT (1)	Capture mode
10	PWM (1)	Single-slope PWM
11	1SHOT (1)	One-shot PWM

Note: 1. Not supported in PEC01 configuration. Refer to Table 22-13 on page 320 for more details.

22.7.7 CTRLG – Control register G

Bit	7	6	5	4	3	2	1	0
+0x06	EVACTEN	EVACT1[1:0]		EVACT0[1:0]		EVSEL[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – EVACTEN: Event Action Enable**
  This command bit is used to enable the event actions for both timer/counters.
- **Bit 6:3 – EVACTx[1:0]: Event Action Selection**
  This bit defines the event action each timer/counter will perform on an event, according to Table 22-17. The EVSRC setting will decide which event source has control. The settings are ignored if the EVACTEN bit is not set.

Table 22-17. Event action selection.

EVACTx[1:0]	Group configuration	Description	
00	INPUT	CAPT(1)	Input capture
	OFF	1SHOT(1)	Event action is disabled
01	FREQ	CAPT(1)	Frequency capture
	OFF	1SHOT(1)	Event action is disabled
10	PW	CAPT	Pulse width capture
	STOPSTART	1SHOT	- Stop on event if counter is counting or - Start on event if counter is stopped
11	RESTART	All modes	Restart counter

Note: 1. Refer to Table 22-16 on page 321 for more details.

- **Bit 2:0 – EVSEL[2:0]: Event Source Selection**
  These bits select the event channel source for the timer/counter, according to Table 22-18. For the selected event channel to have any effect, the event action bits (EVACT) must be set according to Table 22-17.

Table 22-18. Event source selection.

EVSRC[2:0]	Group configuration	Description
n	CHn	Event channel n

22.7.8 INTCTRL – Interrupt Control register

Bit		7	6	5	4	3	2	1	0
+0x08	(1)	UNF1IE	UNF0IE	CC1IE	CC0IE	UNFINTLVL[1:0]		CCINTLVL[1:0]	
	(2)	PEC1IE	PEC0IE	-	-				
	(3)	PEC21IE	-	PEC20IE	-				
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		0	0	0	0	0	0	0	0

Notes: 1. Using TC16, BTC0 and/or BTC1.
2. Using PEC0 and/or PEC1.
3. Using PEC20 and/or PEC21.

- **Bit 7 – UNF1IE: Underflow Interrupt 1 Enable**
  Setting this bit enables the underflow interrupt from BTC1.
- **Bit 7 – PEC1IE: Peripheral Counter 1 Interrupt Enable**
  Setting this bit enables the interrupt from PEC1.
- **Bit 7 – PEC21IE: Peripheral Counter 21 Interrupt Enable**
  Setting this bit enables the interrupt from peripheral counter high when set in BTC0PEC2 configuration.
- **Bit 6 – UNF0IE: Underflow Interrupt 0 Enable**
  Setting this bit enables the underflow interrupt from BTC0.
- **Bit 6 – PEC0IE: Peripheral Counter 0 Interrupt Enable**
  Setting this bit enables the interrupt from PEC0.
- **Bit 5 – CC1IE: Capture or Compare 1 Interrupt Enable**
  Setting this bit enables the capture or compare interrupt on BTC1.
- **Bit 5 – PEC20IE: Peripheral Counter 20 Low Interrupt Enable**
  Setting this bit enables the interrupt from peripheral counter low when set in BTC0PEC2 configuration.
- **Bit 4 – CC0IE: Capture or Compare 0 Interrupt Enable**
  Setting this bit enables the capture or compare interrupt on BTC0.
- **Bit 3:2 – UNFINTLVL[1:0]: Timer Underflow Interrupt Level**
  These bits enable the timer/counter interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.
- **Bit 1:0 – CCINTLVL[1:0]: Timer Compare or Capture Interrupt Level**
  These bits enable the timer interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.

22.7.9 INTFLAGS – Interrupt Flag register

Bit		7	6	5	4	3	2	1	0
	(1)	UNF1IF	UNF0IF	CC1IF	CC0IF	-	-	-	-
+0x08	(2)	PEC1IF	PEC0IF	-	-	-	-	-	-
	(3)	PEC21IF	-	PEC20IF	-	-	-	-	-
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		0	0	0	0	0	0	0	0

Notes:
1. Using TC16, BTC0 and /or BTC1.
2. Using PEC0 and/or PEC1.
3. Using PEC20 and/or PEC21.

- **Bit 7:6 – UNFxIF: Timer/Counter Underflow Interrupt Flag**
  This flag is set on a BOTTOM condition. The flag can be cleared by writing a one to its bit location.
- **Bit 7:6 – PECxIF: Peripheral Counter Interrupt Flag**
  This flag is set on a BOTTOM condition on the 8-bit peripheral counter. The flag can be cleared by writing a one to its bit location.
- **Bit 7 – PEC21IF: Peripheral Counter High Interrupt Flag**
  This flag is set on a BOTTOM condition on the 4-bit peripheral high counter. The flag can be cleared by writing a one to its bit location.
- **Bit 5:4 – CCxIF: Compare or Capture Channel x Interrupt Flag**
  The compare or capture interrupt flag (CCxIF) is set on a compare match or on an input capture event on the corresponding CC channel.
  For normal mode of operation, the CCxIF will be set when a compare match occurs between the count register (CNT) and the corresponding compare register (CCx). The flag can be cleared by writing a one to its bit location.
  For input capture operation, the CCxIF will be set if the corresponding compare register contains valid data. The flag will be cleared when the CCx register is read.
  The flag can also be cleared by writing a one to its bit location.

- **Bit 5 – PEC20IF: Peripheral Counter Low Interrupt Flag**
  This flag is set on a BOTTOM condition on the 4-bit peripheral low counter. The flag can be cleared by writing a one to its bit location.
- **Bit 3:0 - Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

22.7.10 PLC – Peripheral Length Control register

When one or both peripheral counters are selected, this register is used to store the TOP value of the counter(s).

This register is used as TEMP register when reading 16-bit registers when TC16 configuration is selected.

Bit		7	6	5	4	3	2	1	0
+0x09	(1)	PLC[7:0]							
	(2)	–	–	–	–	PLC[3:0]			
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		0	0	0	0	0	0	0	0

Notes: 1. Using PEC0 and/or PEC1.
2. Using PEC20 and/or PEC21.

- **Bit 7:0 – PLC[7:0]: Peripheral Length Control Bits**
  These bits hold the TOP value of 8-bit peripheral counter(s).
- **Bit 3:0 – PLC[3:0]: Peripheral Length Control Bits**
  These bits hold the TOP value of 4-bit peripheral counters.

22.7.11 CNTL – Count register low

When the timer/counter is in 16-bit timer/counter configuration, CNTL register contains the low byte of the 16-bit counter value (CNT). For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

When the timer/counter is in a configuration that enables CNT0, CNTL register contains the 8-bit counter 0 value (BCNT0).

When the timer/counter is in a configuration that enables PCNT0, CNTL register contains the 8-bit peripheral counter 0 value (PCNT0).

CPU write access has priority over count or reload of the counter.

Bit		7	6	5	4	3	2	1	0
+0x0A	(1)	CNT[7:0]							
	(2)	BCNT0[7:0]							
	(3)	PCNT0[7:0]							
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		1	1	1	1	1	1	1	1

Notes: 1. Using TC16.
2. Using BCT0.
3. Using PEC0.

- **Bit 7:0 – CNT[7:0]: Counter Low Byte**
  These bits hold the LSB count value of the 16-bit counter register.
- **Bit 7:0 – BCNT0[7:0]: Counter 0 Byte**
  These bits hold the count value of the 8-bit timer/counter 0.
- **Bit 7:0 – PCNT0[7:0]: Peripheral Counter 0 Byte**
  These bits hold the count value of the 8-bit peripheral counter 0. Writing this register requires special attention. Any ongoing serial communication will be corrupted.

22.7.12 CNTH – Count register High

When the timer/counter is in 16-bit timer/counter configuration, CNTH register contains the high byte of the 16-bit counter value (CNT). For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

When the timer/counter is in a configuration that enables CNT1, CNTH register contains the 8-bit counter 1 value (BCNT1).

When the timer/counter is in a configuration that enables PCNT1, CNTH register contains the 8-bit peripheral counter 1 value (PCNT1).

When the timer/counter is in a configuration that enables PCNT2, CNTL register contains both values of the two 4-bit peripheral counters (PCNT21 and PCNT20).

CPU write access has priority over count or reload of the counter.

Bit		7	6	5	4	3	2	1	0
+0x0B	(1)	CNT[15:8]							
	(2)	BCNT1[7:0]							
	(3)	PCNT1[7:0]							
	(4)	PCNT21[3:0]				PCNT20[3:0]			
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		1	1	1	1	1	1	1	

Notes:
1. Using TC16.
2. Using BTC1.
3. Using PEC1.
4. Using PEC2.

- **Bit 7:0 – CNT[15:8]: Counter High Byte**
  These bits hold the LSB count value of the 16-bit counter register.
- **Bit 7:0 – BCNT1[7:0]: Counter 1 Byte**
  These bits hold the count value of the 8-bit BTC1.
- **Bit 7:0 – PCNT1[7:0]: Peripheral Counter 1 Byte**
  These bits hold the count value of the 8-bit peripheral counter 1. Writing this register requires special attention: any ongoing serial communication will be corrupted.
- **Bit 7:4 – PCNT21[3:0]: Peripheral Counter High Bits**
  These bits hold the count value of the 4-bit peripheral high counter. Writing this register requires special attention: any ongoing serial communication will be corrupted.
- **Bit 3:0 – PCNT20[3:0]: Peripheral Counter Low Bits**
  These bits hold the count value of the 4-bit peripheral low counter. Writing this register requires special attention: any ongoing serial communication will be corrupted.

22.7.13 CMPL – Compare register Low

When the timer/counter is in 16-bit timer/counter configuration, CMP register contains the low byte of the 16-bit compare value (CMP). For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

When the timer/counter is in a configuration that enables CNT0, CMPL register contains the 8-bit compare value (BCMP0).

Bit		7	6	5	4	3	2	1	0
+0x0C	(1)	CMP[7:0]							
	(2)	BCMP0[7:0]							
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		1	1	1	1	1	1	1	1

Notes: 1. Using TC16.
2. Using BCT0.

- **Bit 7:0 – CMP[7:0]: Compare Low Byte**
  These bits hold the LSB compare value of the 16-bit timer/counter when it is used in single slop PWM.
- **Bit 7:0 – BCMP0[7:0]: Compare 0 Byte**
  These bits hold the compare value of the 8-bit BTC0 when it is used in single slop PWM.

22.7.14 CMPH – Compare register High

When the timer/counter is in 16-bit timer/counter configuration, CMPH register contains the high byte of the 16-bit compare value (CMP). For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

When the timer/counter is in a configuration that enables CNT1, CMPH register contains the 8-bit compare value (BCMP1).

Bit		7	6	5	4	3	2	1	0
+0x0D	(1)	CMP[15:8]							
	(2)	BCMP1[7:0]							
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		1	1	1	1	1	1	1	1

Notes: 1. Using TC16.
2. Using BCT1.

- **Bit 7:0 – CMP[15:8]: Compare High Byte**
  These bits hold the MSB compare value of the 16-bit timer/counter when it is used in single slop PWM.
- **Bit 7:0 – BCMP1[7:0]: Compare 1 Byte**
  These bits hold the compare value of the 8-bit BTC1 when it is used in single slop PWM.

22.7.15 PERCAPTL – Period and Capture register Low

When the timer/counter is in 16-bit timer/counter configuration, PERCAPTL register contains either the low byte of the 16-bit period value (PER[7:0]) or the low byte of the 16-bit capture value (CAPT[7:0]). For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

When the timer/counter is in a configuration that enables CNT0, PERCAPTL register contains either the 8-bit period value (BPER0) or either the 8-bit capture value (BCAPT0).

Bit		7	6	5	4	3	2	1	0
+0x0E	(1)	PER[7:0]							
		CAPT[7:0]							
	(2)	BPER0[7:0]							
		BCAPT0[7:0]							
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		1	1	1	1	1	1	1	1

Notes: 1. Using TC16.
2. Using BCT0.

- **Bit 7:0 – PER[7:0]: Period Low Byte**
  These bits hold the LSB period value of the 16-bit timer/counter.
- **Bit 7:0 – CAPT[7:0]: Capture Value Low Byte**
  These bits hold the LSB capture value of the 16-bit timer/counter when an event is received.
- **Bit 7:0 – BPER0[7:0]: Period Byte 0**
  These bits hold the period value of the 8-bit BTC0.
- **Bit 7:0 – BCAPT0[7:0]: Capture Value Byte 0**
  These bits hold the capture value of the 8-bit BTC0 when an event is received.

22.7.16 PERCAPTH – Period and Capture register High

When the timer/counter is in 16-bit timer/counter configuration, PERCAPTH register contains either the high byte of the 16-bit period value (PER) or the high byte of the 16-bit capture value (CAPT). For more details on reading and writing 16-bit registers, refer to “Accessing 16-bit registers” on page 13.

When the timer/counter is in a configuration that enables CNT1, PERCAPTH register contains either the 8-bit period value (BPER1) or either the 8-bit capture value (BCAPT1).

Bit		7	6	5	4	3	2	1	0
+0x0F	(1)	PER[15:8]							
		CAPT[15:8]							
	(2)	BPER1[7:0]							
		BCAPT1[7:0]							
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value		1	1	1	1	1	1	1	1

Notes: 1. Using TC16.
2. Using BCT1.

- **Bit 7:0 – PER[15:8]: Period High Byte**
  These bits hold the MSB period value of the 16-bit timer/counter.
- **Bit 7:0 – CAPT[15:8]: Capture Value High Byte**
  These bits hold the MSB capture value of the 16-bit timer/counter when an event is received.
- **Bit 7:0 – BPER1[7:0]: Period Byte 1**
  These bits hold the period value of the 8-bit timer/counter 1.
- **Bit 7:0 – BCAPT1[7:0]: Capture Value Byte 1**
  These bits hold the capture value of the 8-bit timer/counter 1 when an event is received.

22.8 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	LUT0OUT [1:0]		PORTSEL[1:0]		-	LUTCONF[2:0]			317
+0x01	CTRLB	IN3SEL[1:0]		IN2SEL[1:0]		IN1SEL[1:0]		IN0SEL[1:0]		318
+0x02	CTRLC	EVASYSEL1	EVASYSEL0	DLYSEL[1:0]		DLYCONF1[1:0]		DLYCONF0[1:0]		318
+0x03	CTRLD	TRUTH1[3:0]				TRUTH0[3:0]				319
+0x04	CTRLE	CMDSEL	TCSEL[2:0]			CLKSEL[3:0]				320
+0x05	CTRLF	CMDEN[1:0]		CMP1	CMP0	CCEN1	CCEN0	MODE[1:0]		321
+0x06	CTRLG	EVACTEN	EVACT1[1:0]		EVACT0[1:0]		EVSRC[2:0]			322

22.8.1 Register summary – One 16-bit T/C (TC16)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	BIt 2	BIt 1	Bit 0	Page
+0x07	INTCTRL	-	UNF0IE	-	CC0IE	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	-	UNF0IF	-	CC0IF	-	-	-	-	323

22.8.1.1 T/C in normal mode with programmable period (NORMAL)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	CNT[7:0]								324
+0x0B	CNTH	CNT[15:8]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	PER[7:0]								327
+0x0F	PERCAPTH	PER[15:8]								328

22.8.1.2 T/C in capture mode (CAPT)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	CNT[7:0]								324
+0x0B	CNTH	CNT[15:8]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	CAPT[7:0]								327
+0x0F	PERCAPTH	CAPT[15:8]								328

22.8.1.3 T/C in PWM Modes with Period Fixed to MAX (PWM or SSPWM)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	CNT[7:0]								324
+0x0B	CNTH	CNT[15:8]								325
+0x0C	CMPL	CMP[7:0]								326
+0x0D	CMPH	CMP[15:8]								326
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.2 Register summary – One 8-bit T/C (BTC0)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x07	INTCTRL	-	UNF0IE	-	CC0IE	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	-	UNF0IF	-	CC0IF	-	-	-	-	323

22.8.2.1 T/C in normal mode with programmable period (NORMAL)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.2.2 T/C in capture mode (CAPT)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BCAPT0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.2.3 T/C in PWM modes with programmable period (PWM or SSPWM)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	CMPL	BCPM0[7:0]								326
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.3 Register summary – Two 8-bit T/C (BTC01)

Address	Name	Bit 7	Bit 6	BIt 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x07	INTCTRL	UNF1IE	UNF0IE	CC1IE	CC0IE	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	UNF1IF	UNF0IF	CC1IF	CC0IF	-	-	-	-	323

22.8.3.1 T/C in normal mode with programmable period (NORMAL)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	CNTH	BCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	

+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	PERCAPTH	BPER1[7:0]								328

22.8.3.2 T/C in capture mode (CAPT)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	CNTH	BCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BCAPT0[7:0]								327
+0x0F	PERCAPTH	BCAPT1[7:0]								328

22.8.3.3 T/C in PWM modes with period fixed to MAX (PWM or SSPWM)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	CNTH	BCNT1[7:0]								325
+0x0C	CMPL	BCMP0[7:0]								326
+0x0D	CMPH	BCMP1[7:0]								326
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.4 Register summary – One 8-bit T/C and one 8-bit Tx PEC (BTC0PEC1)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x07	INTCTRL	PEC1IE	UNF0IE	-	CC0IE	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	PEC1IF	UNF0IF	-	CC0IF	-	-	-	-	323

22.8.4.1 T/C in normal mode with programmable period (NORMAL)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.4.2 T/C in capture mode (CAPT)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0D	PERCAPTL	BCAPT0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.4.3 T/C in PWM modes with programmable period (PWM or SSPWM)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	CMPL	BCMP0[7:0]								326
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.4.4 Transmitter peripheral counter (PEC)

+0x09	PLC	PLC[7:0]								324
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	CNTH	PCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.5 Register summary – One 8-bit T/C and one 8-bit Rx PEC (PEC0BTC1)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x07	INTCTRL	UNF1IE	PEC0IE	CC1IE	-	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	UNF1IF	PEC0IF	CC1IF	-	-	-	-	-	323

22.8.5.1 Receiver peripheral counter (PEC)

+0x09	PLC	PLC[7:0]								324
+0x0A	CNTL	PCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.5.2 T/C in normal mode with programmable period (NORMAL)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	CNTH	BCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	PERCAPTH	BPER1[7:0]								328

22.8.5.3 T/C in capture mode (CAPT)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	CNTH	BCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	PERCAPTH	BCAPT1[7:0]								328

22.8.5.4 T/C in PWM modes with programmable period (PWM or SSPWM)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	CNTH	BCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	CMPH	BCMP1[7:0]								326
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	PERCAPTH	BPER1[7:0]								328

22.8.6 Register summary – Two 8-bit Tx/Rx PEC (PEC01)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x07	INTCTRL	PEC1IE	PEC0IE	-	-	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	PEC1IF	PEC0IF	-	-	-	-	-	-	323

22.8.6.1 Transmitter/Receiver peripheral counter (PEC)

+0x09	PLC	PLC[7:0] - (Same length for Tx & Rx)								324
+0x0A	CNTL	PCNT0[7:0]								324
+0x0B	CNTH	PCNT1[7:0]								325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.7 Register summary – One 8-bit T/C and two 4-bit Tx/Rx PEC (BTC0PEC2)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	BIt 3	Bit 2	Bit 1	Bit 0	Page
+0x07	INTCTRL	PEC2HIE	UNF0IE	PEC2LIE	CC0IE	UNFINTLVL[1:0]		CCINTLVL[1:0]		322
+0x08	INTFLAGS	PEC2HIF	UNF0IF	PEC2LIF	CC0IF	-	-	-	-	323

22.8.7.1 T/C in normal mode with programmable period (NORMAL)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.7.2 T/C in capture mode (CAPT)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BCAPT0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.8 T/C in PWM modes with programmable period (PWM or SSPWM)

+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	CNTL	BCNT0[7:0]								324
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	CMPL	BCMP0[7:0]								326
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	PERCAPTL	BPER0[7:0]								327
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.8.8.1 Transmitter/Receiver peripheral counter (PEC)

+0x09	PLC	-	-	-	-	PLC[3:0] - (Same length for Tx & Rx)				324
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	CNTH	PCNT21[3:0]				PCNT20[3:0]				325
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	

22.9 Interrupt vector summary

Table 22-19. XCL interrupt vectors and their word offset address.

Offset	Source	Interrupt description
0x00	UNF_vect	Timer/counter underflow interrupt vector offset
0x02	CC_vect	Timer/counter compare or capture interrupt vector offset

22.10 T/C and PEC register summary vs. Configuration and mode

Config.	Mode	@Off.	Register	Field	Mode	@Off.	Register	Field	Mode	@Off.	Register	Field
TC16	NORMAL	+0x09	Reserved		CAPT	+0x09	Reserved		PWM & 1SHOT	+0x09	Reserved	
		+0x0A	CNTL	CNT[7:0]		+0x0A	CNTL	CNT[7:0]		+0x0A	CNTL	CNT[7:0]
		+0x0B	CNTH	CNT[15:8]		+0x0B	CNTH	CNT[15:8]		+0x0B	CNTH	CNT[15:8]
		+0x0C	Reserved			+0x0C	Reserved			+0x0C	CMPL	CMP[7:0]
		+0x0D	Reserved			+0x0D	Reserved			+0x0D	CMPH	CMP[15:8]
		+0x0E	PERCAPTL	PER[7:0]		+0x0E	PERCAPTL	CAPT[7:0]		+0x0E	Reserved	
		+0x0F	PERCAPTH	PER[15:8]		+0x0F	PERCAPTH	CAPT[15:8]		+0x0F	Reserved	
BTC0	NORMAL	+0x09	Reserved		CAPT	+0x09	Reserved		PWM & 1SHOT	+0x09	Reserved	
		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]
		+0x0B	Reserved			+0x0B	Reserved			+0x0B	Reserved	
		+0x0C	Reserved			+0x0C	Reserved			+0x0C	CMPL	BCMP0[7:0]
		+0x0D	Reserved			+0x0D	Reserved			+0x0D	Reserved	
		+0x0E	PERCAPTL	BPER0[7:0]		+0x0E	PERCAPTL	BCAPT0[7:0]		+0x0E	PERCAPTL	BPER0[7:0]
		+0x0F	Reserved			+0x0F	Reserved			+0x0F	Reserved	
BTC01	NORMAL	+0x09	Reserved		CAPT	+0x09	Reserved		PWM & 1SHOT	+0x09	Reserved	
		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]
		+0x0B	CNTH	BCNT1[7:0]		+0x0B	CNTH	BCNT1[7:0]		+0x0B	CNTH	BCNT1[7:0]
		+0x0C	Reserved			+0x0C	Reserved			+0x0C	CMPL	BCMP0[7:0]
		+0x0D	Reserved			+0x0D	Reserved			+0x0D	CMPH	BCMP1[7:0]
		+0x0E	PERCAPTL	BPER0[7:0]		+0x0E	PERCAPTL	BCAPT0[7:0]		+0x0E	Reserved	
		+0x0F	PERCAPTH	BPER1[7:0]		+0x0F	PERCAPTH	BCAPT1[7:0]		+0x0F	Reserved	
BTC0PEC1	NORMAL	+0x09	PLC	PLC[7:0]	CAPT	+0x09	PLC	PLC[7:0]	PWM & 1SHOT	+0x09	PLC	PLC[7:0]
		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]
		+0x0B	CNTH	PCNT1[7:0]		+0x0B	CNTH	PCNT1[7:0]		+0x0B	CNTH	PCNT1[7:0]
		+0x0C	Reserved			+0x0C	Reserved			+0x0C	CMPL	BCMP0[7:0]
		+0x0D	Reserved			+0x0D	Reserved			+0x0D	Reserved	
		+0x0E	PERCAPTL	BPER0[7:0]		+0x0E	PERCAPTL	BCAPT0[7:0]		+0x0E	PERCAPTL	BPER0[7:0]
		+0x0F	Reserved			+0x0F	Reserved			+0x0F	Reserved	
PEC0BTC1	NORMAL	+0x09	PLC	PLC[7:0]	CAPT	+0x09	PLC	PLC[7:0]	PWM & 1SHOT	+0x09	PLC	PLC[7:0]
		+0x0A	CNTL	PCNT0[7:0]		+0x0A	CNTL	PCNT0[7:0]		+0x0A	CNTL	PCNT0[7:0]
		+0x0B	CNTH	BCNT1[7:0]		+0x0B	CNTH	BCNT1[7:0]		+0x0B	CNTH	BCNT1[7:0]
		+0x0C	Reserved			+0x0C	Reserved			+0x0C	Reserved	
		+0x0D	Reserved			+0x0D	Reserved			+0x0D	CMPH	BCMP1[7:0]
		+0x0E	Reserved			+0x0E	Reserved			+0x0E	Reserved	
		+0x0F	PERCAPTH	BPER1[7:0]		+0x0F	PERCAPTH	BCAPT1[7:0]		+0x0F	PERCAPTH	BPER1[7:0]
PEC01	NORMAL	+0x09	PLC	PLC[7:0]								
		+0x0A	CNT0	PCNT0[7:0]								
		+0x0B	CNT1	PCNT1[7:0]								
		+0x0C	Reserved									
		+0x0D	Reserved									
		+0x0E	Reserved									
		+0x0F	Reserved									
BTC0PEC2	NORMAL	+0x09	PLC	PLC[3:0]	CAPT	+0x09	PLC	PLC[3:0]	PWM & 1SHOT	+0x09	PLC	PLC[3:0]
		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]		+0x0A	CNTL	BCNT0[7:0]
		+0x0B	CNTH	PCNT20-21[3:0]		+0x0B	CNTH	PCNT20-21[3:0]		+0x0B	CNTH	PCNT20-21[3:0]
		+0x0C	Reserved			+0x0C	Reserved			+0x0C	CMPL	BCMP0[7:0]
		+0x0D	Reserved			+0x0D	Reserved			+0x0D	Reserved	
		+0x0E	PERCAPTL	BPER0[7:0]		+0x0E	PERCAPTL	BCAPT0[7:0]		+0x0E	PERCAPTL	BPER0[7:0]
		+0x0F	Reserved			+0x0F	Reserved			+0x0F	Reserved	

23. CRC – Cyclic Redundancy Check generator

23.1 Features

- Cyclic redundancy check (CRC) generation and checking for
  - Communication data
  - Program or data in flash memory
  - Data in SRAM and I/O memory space
- Integrated with flash memory, EDMA controller and CPU
  - Continuous CRC on data going through an EDMA channel
  - Automatic CRC of the complete or a selectable range of the flash memory
  - CPU can load data to the CRC generator through the I/O interface
- CRC polynomial software selectable to
  - CRC-16 (CRC-CCITT)
  - CRC-32 (IEEE 802.3)
- Zero remainder detection

23.2 Overview

A cyclic redundancy check (CRC) is an error detection technique test algorithm used to find accidental errors in data, and it is commonly used to determine the correctness of a data transmission, and data presence in the data and program memories. A CRC takes a data stream or a block of data as input and generates a 16- or 32-bit output that can be appended to the data and used as a checksum. When the same data are later received or read, the device or application repeats the calculation. If the new CRC result does not match the one calculated earlier, the block contains a data error. The application will then detect the error and may take a corrective action, such as requesting the data to be sent again or simply not using the incorrect data.

Typically, an n-bit CRC applied to a data block of arbitrary length will detect any single error burst not longer than n bits (any single alteration that spans no more than n bits of the data), and will detect the fraction $1-2^{-n}$ of all longer error bursts. The CRC module in XMEGA devices supports two commonly used CRC polynomials; CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3).

- **CRC-16:**

Polynomial:	$x^{16}+x^{12}+x^5+1$
Hex value:	0x1021

- **CRC-32:**

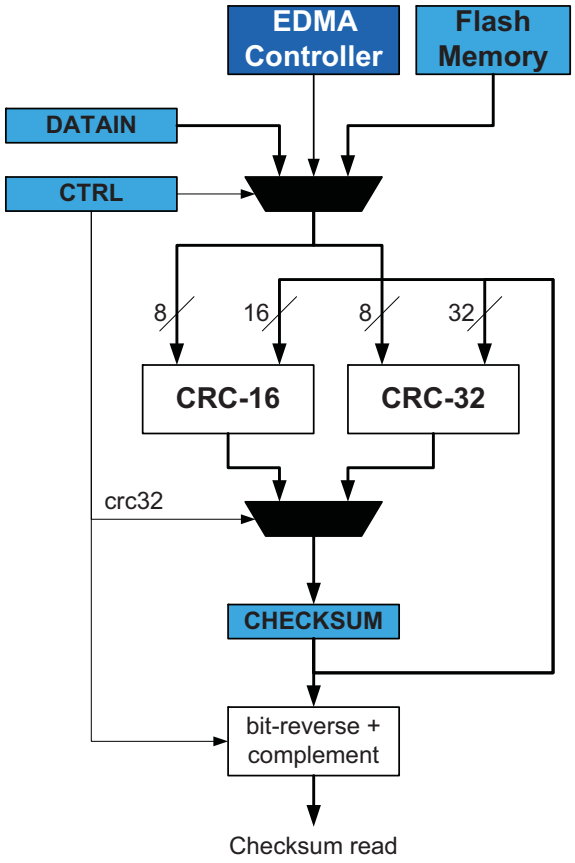
Polynomial:	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
Hex value:	0x04C11DB7

23.3 Operation

The data source for the CRC module must be selected in software as either flash memory, the EDMA channels, or the I/O interface. The CRC module then takes data input from the selected source and generates a checksum based on these data. The checksum is available in the CHECKSUM registers in the CRC module. When CRC-32 polynomial is used, the final checksum read is bit reversed and complemented (see Figure 23-1 on page 337).

For the I/O interface or EDMA controller, which CRC polynomial is used is software selectable, but the default setting is CRC-16. CRC-32 is automatically used if Flash Memory is selected as the source. The CRC module operates on bytes only.

Figure 23-1. CRC generator block diagram.



23.4 CRC on Flash memory

A CRC-32 calculation can be performed on the entire flash memory, on only the application section, on only the boot section, or on a software selectable range of the flash memory. Other than selecting the flash as the source, all further control and setup are done from the NVM controller. This means that the NVM controller configures the memory range to perform the CRC on, and the CRC is started using NVM commands. Once completed, the result is available in the checksum registers in the CRC module. For further details on setting up and performing CRC on flash memory, refer to "Memory Programming" on page 403.

23.5 CRC on EDMA Data

CRC-16 or CRC-32 calculations can be performed on data passing through any EDMA channel. Once a EDMA channel is selected as the source, the CRC module will continuously generate the CRC on the data passing through the EDMA channel. The checksum is available for readout once the EDMA transaction is completed or aborted. A CRC can be performed not only on communication data, but also on data in SRAM or I/O memory by passing these data through an EDMA channel. If the latter is done, the destination register for the EDMA data can be the data input (DATAIN) register in the CRC module. Refer to "EDMA – Enhanced Direct Memory Access" on page 50 for more details on setting up EDMA transactions.

23.6 CRC using the I/O Interface

CRC can be performed on any data by loading them into the CRC module using the CPU and writing the data to the DATAIN register. Using this method, an arbitrary number of bytes can be written to the register by the CPU, and CRC is done continuously for each byte. New data can be written for each cycle. The CRC complete is signaled by writing the BUSY bit in the STATUS register.

23.7 Register Description

23.7.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	RESET[1:0]		CRC32	–	SOURCE[3:0]			
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – RESET[1:0]: Reset**
  These bits are used to reset the CRC module, and they will always be read as zero. The CRC registers will be reset one peripheral clock cycle after the RESET[1] bit is set.

Table 23-1. CRC reset.

RESET[1:0]	Group configuration	Description
00	NO	No reset
01	–	Reserved
10	RESET0	Reset CRC with CHECKSUM to all zeros
11	RESET1	Reset CRC with CHECKSUM to all ones

- **Bit 5 – CRC32: CRC-32 Enable**
  Setting this bit will enable CRC-32 instead of the default CRC-16. It cannot be changed while the BUSY flag is set.
- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 3:0 – SOURCE[3:0]: Input Source**
  These bits select the input source for generating the CRC. The selected source is locked until either the CRC generation is completed or the CRC module is reset. CRC generation complete is generated and signaled from the selected source when used with the EDMA controller or flash memory.

Table 23-2. CRC source select.

SOURCE[3:0]	Group configuration	Description
0000	DISABLE	CRC disabled
0001	IO	I/O interface
0010	FLASH	Flash
0011	–	Reserved for future use
0100	EDMACH0	EDMA controller channel 0
0101	EDMACH1	EDMA controller channel 1
0110	EDMACH2	EDMA controller channel 2
0111	EDMACH3	EDMA controller channel 3
1xxx	–	Reserved for future use

23.7.2 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	–	–	ZERO	BUSY
Read/Write	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – ZERO: Checksum Zero**
  This flag is set if the CHECKSUM is zero when the CRC generation is complete. It is automatically cleared when a new CRC source is selected.
  When running CRC-32 and appending the checksum at the end of the packet (as little endian), the final checksum shold be 0x2144df1c, and not zero. However, if the checksum is complemented before it is appended (as little endian) to the data, the final result in the checksum register will be zero.
  See the description of CHECKSUM to read out different versions of the CHECKSUM.
- **Bit 0 – BUSY: Busy**
  This flag is read as one when a source configuration is selected and as long as the source is using the CRC module. If the I/O interface is selected as the source, the flag can be cleared by writing a one this location. If an EDMA channel if selected as the source, the flag is cleared when the EDMA channel transaction is completed or aborted. If flash memory is selected as the source, the flag is cleared when the CRC generation is completed.

23.7.3 DATAIN – Data Input register

Bit	7	6	5	4	3	2	1	0
+0x03	DATAIN[7:0]							
Read/Write	W	W	W	W	W	W	W	W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DATAIN[7:0]: Data Input**
  This register is used to store the data for which the CRC checksum is computed. A new CHECKSUM is ready one clock cycle after the DATAIN register is written.

23.7.4 CHECKSUM0 – Checksum register 0

CHECKSUM0, CHECKSUM1, CHECKSUM2, and CHECKSUM3 represent the 16- or 32-bit CHECKSUM value and the generated CRC. The registers are reset to zero by default, but it is possible to write RESET to reset all bits to one. It is possible to write these registers only when the CRC module is disabled. If NVM is selected as the source, reading CHECKSUM will return a zero value until the BUSY flag is cleared. If CRC-32 is selected and the BUSY flag is cleared (i.e., CRC generation is completed or aborted), the bit reversed (bit 31 is swapped with bit 0, bit 30 with bit 1, etc.) and complemented result will be read from CHECKSUM. If CRC-16 is selected or the BUSY flag is set (i.e., CRC generation is ongoing), CHECKSUM will contain the actual content.

Bit	7	6	5	4	3	2	1	0
+0x04	CHECKSUM[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CHECKSUM[7:0]: Checksum Byte 0**
  These bits hold byte 0 of the generated CRC.

23.7.5 CHECKSUM1 – Checksum register 1

Bit	7	6	5	4	3	2	1	0
+0x05	CHECKSUM[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CHECKSUM[15:8]: Checksum Byte 1**
  These bits hold byte 1 of the generated CRC.

23.7.6 CHECKSUM2 – Checksum register 2

Bit	7	6	5	4	3	2	1	0
+0x06	CHECKSUM[23:16]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CHECKSUM[23:16]: Checksum Byte 2**
  These bits hold byte 2 of the generated CRC when CRC-32 is used.

23.7.7 CHECKSUM3 – CRC Checksum register 3

Bit	7	6	5	4	3	2	1	0
+0x07	CHECKSUM[31:24]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CHECKSUM[31:24]: Checksum Byte 3**
  These bits hold byte 3 of the generated CRC when CRC-32 is used.

23.8 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	bit 0	Page
+0x00	CTRL	RESET[1:0]		CRC32	–	SOURCE[3:0]				338
+0x01	STATUS	–	–	–	–	–	–	ZERO	BUSY	339
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	DATAIN	DATAIN[7:0]								339
+0x04	CHECKSUM0	CHECKSUM[7:0]								339
+0x05	CHECKSUM1	CHECKSUM[15:8]								340
+0x06	CHECKSUM2	CHECKSUM[23:16]								340
+0x07	CHECKSUM3	CHECKSUM[31:24]								340

24. ADC – Analog to Digital Converter

24.1 Features

- 12-bit resolution
- Up to 300 thousand samples per second
  - Down to 2.3µs conversion time with 8-bit resolution
  - Down to 3.35µs conversion time with 12-bit resolution
- Differential and single-ended input
  - Up to 16 single-ended inputs
  - 16x8 differential input with programmable gain
- Built-in differential gain stage
  - 1/2x, 1x, 2x, 4x, 8x, 16x, 32x and 64x gain options
- Single, continuous and scan conversion options
- Four internal inputs
  - Internal temperature sensor
  - DAC output
  - V_{CC} voltage divided by 10
  - 1.1V bandgap voltage
- Internal and external reference options
- Compare function for accurate monitoring of user defined thresholds
- Offset and gain correction
- Averaging
- Over-sampling and decimation
- Optional event triggered conversion for accurate timing
- Optional interrupt/event on compare result
- Optional EDMA transfer of conversion results

24.2 Overview

The ADC converts analog signals to digital values. The ADC has 12-bit resolution and is capable of converting up to 300 thousand samples per second (ksps). The input selection is flexible, and both single-ended and differential measurements can be done. A programmable gain stage is available to increase the dynamic range. In addition, several internal signal inputs are available. The ADC can provide both signed and unsigned results.

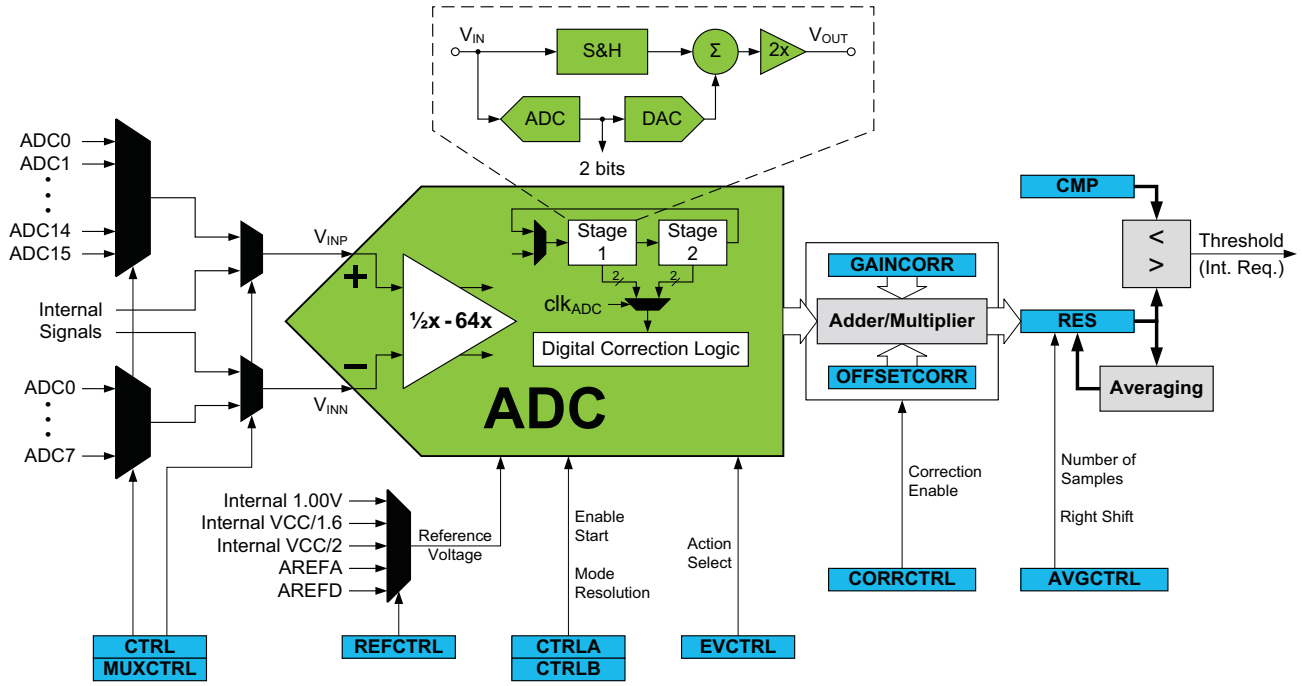
The ADC measurements can either be started by application software or an incoming event from another peripheral in the device. The ADC measurements can be started with predictable timing, and without software intervention. It is possible to use EDMA to move ADC results directly to memory or peripherals when conversions are done.

Both internal and external reference voltages can be used. An integrated temperature sensor is available for use with the ADC. The V_{CC}/10 and the bandgap voltage can also be measured by the ADC.

The ADC has a compare function for accurate monitoring of user defined thresholds with minimum software intervention required.

When operation in noisy conditions, the average feature can be enabled to increase the ADC resolution. Up to 1024 samples can be averaged, enabling up to 16-bit resolution results. In the same way, using the over-sampling and decimation mode, the ADC resolution is increased up to 16-bits, which results in up to 4-bit extra LSB resolution. The ADC includes various calibration options. In addition to standard production calibration, the user can enable the offset and gain correction to improve the absolute ADC accuracy.

Figure 24-1. ADC block diagram.



24.3 Input sources

Input sources are the voltage inputs that the ADC can measure and convert. Three types of measurements can be selected:

- Differential input with programmable gain
- Single-ended input
- Internal input

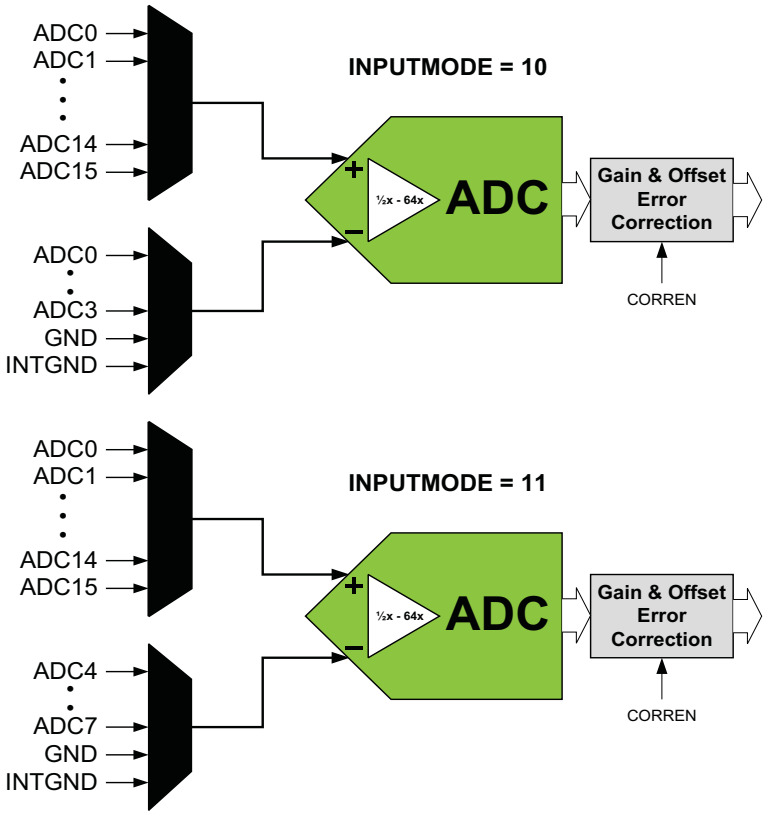
The input pins are used for single-ended and differential input, while the internal inputs are directly available inside the device. Port A and Port D pins can be input to ADC.

The ADC is differential, and so for single-ended measurements the negative input is connected to a fixed internal value. The three types of measurements and their corresponding input options are shown in Figure 24-5 on page 344 to Figure 24-6 on page 345.

24.3.1 Differential inputs

When differential inputs are enabled, all input pins can be selected as positive input, and input pins 0 to 7 can be selected as negative input. For gain settings other than 1x, the differential input is first sampled and amplified by the gain stage before the result is converted. The gain is selectable to 1/2x, 1x, 2x, 4x, 8x, 16x, 32x and 64x gain.

Figure 24-2. Differential measurement.

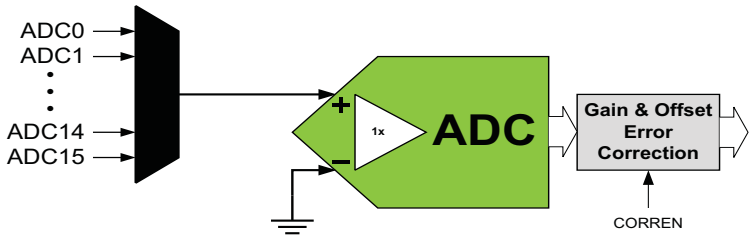


24.3.2 Single-ended input

For single-ended measurements, all input pins can be used as inputs. Single-ended measurements can be done in both signed and unsigned mode. For normal operation of the ADC, gain should be programmed by the application to 1x in this mode.

The negative input is connected to internal ground in signed mode.

Figure 24-3. Single-ended measurement in signed mode.

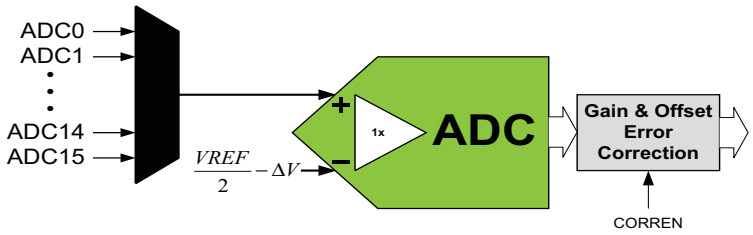


In unsigned mode, the negative input is connected to half of the voltage reference (V_{REF}) voltage minus a fixed offset. The nominal value for the offset is:

$$\Delta V = V_{REF} \times 0.05$$

Since the ADC is differential, the input range is V_{REF} to zero for the positive single-ended input. The offset enables the ADC to measure zero crossing in unsigned mode.

Figure 24-4. Single-ended measurement in unsigned mode.



24.3.3 Internal inputs

These internal signals can be measured or used by the ADC.

- Temperature sensor
- Bandgap voltage
- V_{CC} scaled
- DAC output
- Pad ground and internal ground

The temperature sensor gives an output voltage that increases linearly with the internal temperature of the device. One or more calibration points are needed to compute the temperature from a measurement of the temperature sensor. The temperature sensor is calibrated at one point in production test, and the result is stored to TEMPESENSE0 and TEMPSENSE1 in the production signature row. For more calibration condition details, refer to the device datasheet.

The bandgap voltage is an accurate internal voltage reference.

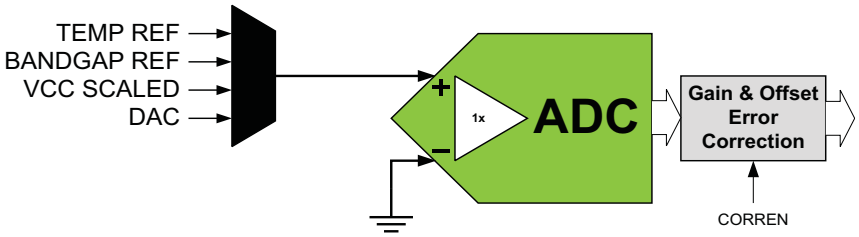
V_{CC} can be measured directly by scaling it down by a factor of 10 before the ADC input. Thus, a V_{CC} of 1.8V will be measured as 0.18V, and V_{CC} of 3.6V will be measured as 0.36V. This enables easy measurement of the V_{CC} voltage.

The internal signals need to be enabled before they can be measured. Refer to their manual sections for Bandgap and DAC for details of how to enable these. The sample rate for the internal signals is lower than that of the ADC. Refer to the ADC characteristics in the device datasheets for details.

For differential measurement pad ground (GND) and internal ground (INTGND) can be selected as negative input. Pad ground is the ground level on the pin and identical or very close to the external ground. Internal ground is the internal device ground level. For normal operation of the ADC, gain should be programmed by the application to 1x in this mode.

Internal ground is used as the negative input when other internal signals are measured in single-ended signed mode.

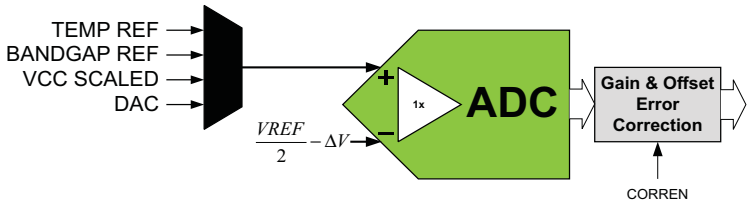
Figure 24-5. Internal measurement in single-ended signed mode.



To measure the internal signals in unsigned mode, the negative input is connected to a fixed value given by the formula below, which is half of the voltage reference (V_{REF}) minus a fixed offset, as it is for single-ended unsigned input. Refer to Table 24-2 on page 347 for details.

$$V_{INN} = \frac{V_{REF}}{2} - \Delta V$$

Figure 24-6. Internal measurement in unsigned mode.



24.4 Sampling time control

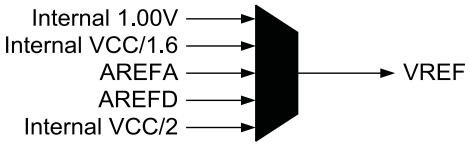
To support applications with high source output resistance, the sampling time can be increased by steps of a half ADC clock cycle.

24.5 Voltage reference selection

The following voltages can be used as the reference voltage (V_{REF}) for the ADC:

- Accurate internal 1.00V voltage generated from the bandgap
- Internal V_{CC}/1.6 voltage
- Internal V_{CC}/2 voltage
- External voltage applied to AREF pin on port A
- External voltage applied to AREF pin on port D

Figure 24-7. ADC voltage reference selection.



24.6 Conversion result

The result of the analog-to-digital conversion is written to the channel result register. The ADC is either in signed or unsigned mode. This setting is global for the ADC and for the ADC channel.

In signed mode, negative and positive results are generated. Signed mode must be used when the ADC channel is set up for differential measurements. In unsigned mode, only single-ended or internal signals can be measured. With 12-bit resolution, the TOP value of a signed result is 2047, and the results will be in the range -2048 to +2047 (0xF800 - 0x07FF).

The ADC transfer function can be written as:

$$RES = \frac{V_{INP} - V_{INN}}{V_{REF}} \times GAIN \times (TOP + 1)$$

V_{INP} and V_{INN} are the positive and negative inputs to the ADC.

For differential measurements, GAIN is software selectable from 1/2 to 64. For single-ended and internal measurements, GAIN must be set by software to 1x and V_{INP} is the internal ground.

In unsigned mode, only positive results are generated. The TOP value of an unsigned result is 4095, and the results will be in the range 0 to +4095 (0x0 - 0x0FFF).

The ADC transfer functions can be written as:

$$RES = \frac{V_{INP} - (-\Delta V)}{V_{REF}} \times (TOP + 1)$$

V_{INP} is the single-ended or internal input.

The ADC can be configured to generate either an 8-bit or a 12-bit result. A result with lower resolution will be available faster. See the “ADC clock and conversion timing” on page 351 for a description on the propagation delay.

The result register is 16 bits wide, and data are stored as right adjusted 16-bit values. Right adjusted means that the eight least-significant bits (lsb) are found in the low byte. A 12-bit result can be represented either left or right adjusted. Left adjusted means that the eight most-significant bits (msb) are found in the high byte.

When the ADC is in signed mode, the msb represents the sign bit. In 12-bit right adjusted mode, the sign bit (bit 11) is padded to bits 12-15 to create a signed 16-bit number directly. In 8-bit mode, the sign bit (bit 7) is padded to the entire high byte.

Table 24-1 on page 346 to Table 24-2 on page 347 show the different input options, the signal input range, and the result representation with 12-bit right adjusted mode.

Table 24-1. Signed differential input (with gain), input voltage versus output code.

V_{INP}	Signed	
	Read code	**Decimal value**
$V_{INN} + V_{REF} / GAIN$	0x07FF	2047
$V_{INN} + 0.9995 V_{REF} / GAIN$	0x07FE	2046
$V_{INN} + 0.9990 V_{REF} / GAIN$	0x07FD	2045
…	…	…
$V_{INN} + 0.5007 V_{REF} / GAIN$	0x0401	1025
$V_{INN} + 0.5002 V_{REF} / GAIN$	0x0400	1024
$V_{INN} + 0.4998 V_{REF} / GAIN$	0x03FF	1023
$V_{INN} + 0.4993 V_{REF} / GAIN$	0x03FE	1022
…	…	…
$V_{INN} + 0.0010 V_{REF} / GAIN$	0x0002	2
$V_{INN} + 0.0005 V_{REF} / GAIN$	0x0001	1
V_{INN}	0x0000	0
$V_{INN} - 0.0005 V_{REF} / GAIN$	0xFFFF	-1
$V_{INN} - 0.0010 V_{REF} / GAIN$	0xFFFE	-2
…	…	…
$V_{INN} - 0.9990 V_{REF} / GAIN$	0xF802	-2046
$V_{INN} - 0.9995 V_{REF} / GAIN$	0xF801	-2047
$V_{INN} - V_{REF} / GAIN$	0xF800	-2048

Table 24-2. Single-ended, input voltage versus output code.

V_{INP}	Signed		Unsigned	
	Read code	Decimal value	Read code	Decimal value
V_{REF}	0x07FF	2047	0x0FFF (saturation)	4095 (saturation)
0.9998 V_{REF}	-	-	0x0FFF (saturation)	4095 (saturation)
0.9995 V_{REF}	0x07FE	2046	0x0FFF (saturation)	4095 (saturation)
0.9993 V_{REF}	-	-	0x0FFF (saturation)	4095 (saturation)
0.9990 V_{REF}	0x07FD	2045	0x0FFF (saturation)	4095 (saturation)
…	...	...	...	...
0.9500 V_{REF}	0x0799	1945	0x0FFF	4095
0.9498 V_{REF}	-	-	0x0FFE	4094
0.9495 V_{REF}	0x0798	1944	0x0FFD	4093
0.9493 V_{REF}	-	-	0x0FFC	4092
0.9490 V_{REF}	0x0797	1943	0x0FFB	4091
…	...	...	...	...
0.5005 V_{REF}	0x0400	1024	0x08CE	2254
0.5002 V_{REF}	-	-	0x08CD	2253
0.5000 V_{REF}	0x03FF	1023	0x08CC	2252
0.4998 V_{REF}	-	-	0x08CB	2251
0.4995 V_{REF}	0x03FE	1022	0x08CA	2250
…	…	…	…	…
0.0005 V_{REF}	0x0001	1	0x00CF	207
0.0002 V_{REF}	-	-	0x00CE	206
GND	0x0000	0	0x00CD	205
-0.0002 V_{REF}	-	-	0x00CC	204
-0.0005 V_{REF}	0xFFFE	-1	0x00CB	203
…	…	…	…	…
-0.0593 V_{REF}	0xFF9B	-101	0x0003	3
-0.0495 V_{REF}	-	-	0x0002	2
-0.0498 V_{REF}	0xFF9A	-102	0x0001	1
-0.0500 V_{REF}	-	-	0x0000	0
-0.0503 V_{REF}	0xFF99	-103	0x0000 (saturation)	0 (saturation)
-0.0506 V_{REF}	-	-	0x0000 (saturation)	0 (saturation)
-0,0508 V_{REF}	0xFF98	-104	0x0000 (saturation)	0 (saturation)

24.7 Calibration and correction

24.7.1 Production test calibration

The ADC has built-in linearity calibration. Values of electronic elements in ADC stage 1 and stage 2, as shown in Figure 24-1 on page 342, are adjusted by this calibration. The value from the production test calibration must be loaded from the signature row and into the ADC calibration register (CAL) from software to achieve specified accuracy. The production test calibration must be loaded prior to any other hardware correction.

24.7.2 Offset and gain correction

Inherent gain and offset errors affect the absolute accuracy of the ADC.

The offset error is defined as the deviation of the current ADC transfer function from ideal straight line at zero input voltage. The offset error cancellation is handled by a 12-bit register (OFFSETCORR). The offset correction value is subtracted from the data converted before writing the result (RES) register. The offset error calculation must be done prior to any gain error correction.

The gain error is defined as the deviation of the last output step's midpoint from the ideal straight line, after compensating for offset error. The gain error cancellation is handled via 12-bit register (GAINCORR).

To correct these two errors, the bit CORREN must be set in CORRCTRL register. The equation that is implemented in hardware for correcting the output is:

$$RES = (V_{IN} - OFFSETCORR) \times GAINCORR$$

In single conversion, a latency of 13 peripheral clock cycles (clk_{PER}) is added for the final conversion result availability. Since the correction time is always less than propagation delay, in free running mode this latency affects only the first conversion time. All the other conversions are done within the normal sampling rate.

Figure 24-8. ADC offset error.

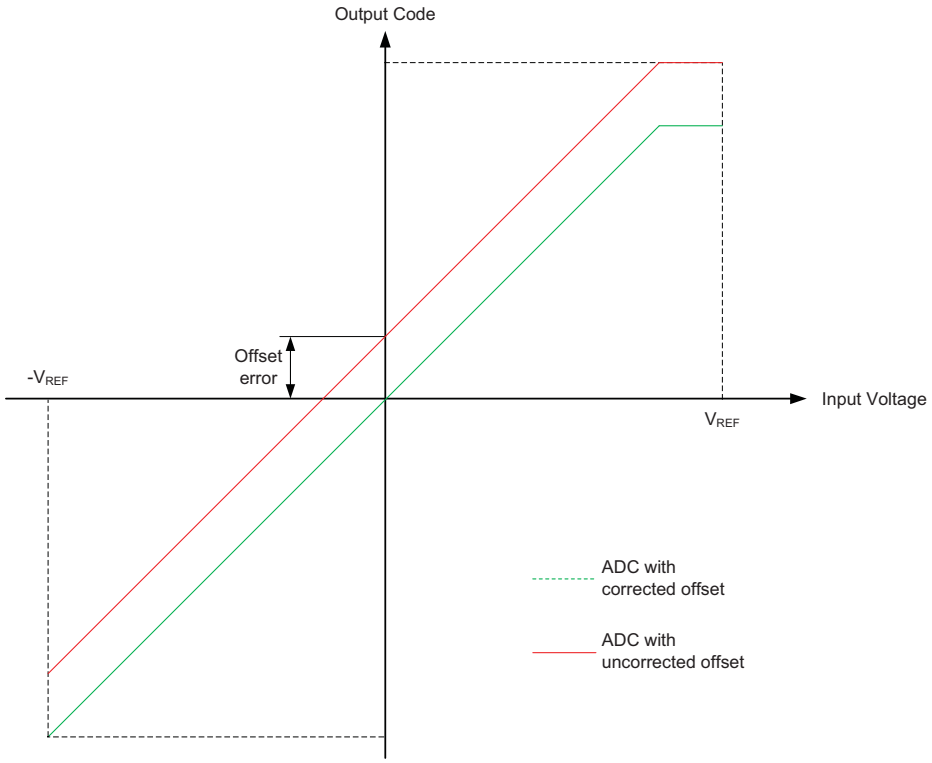
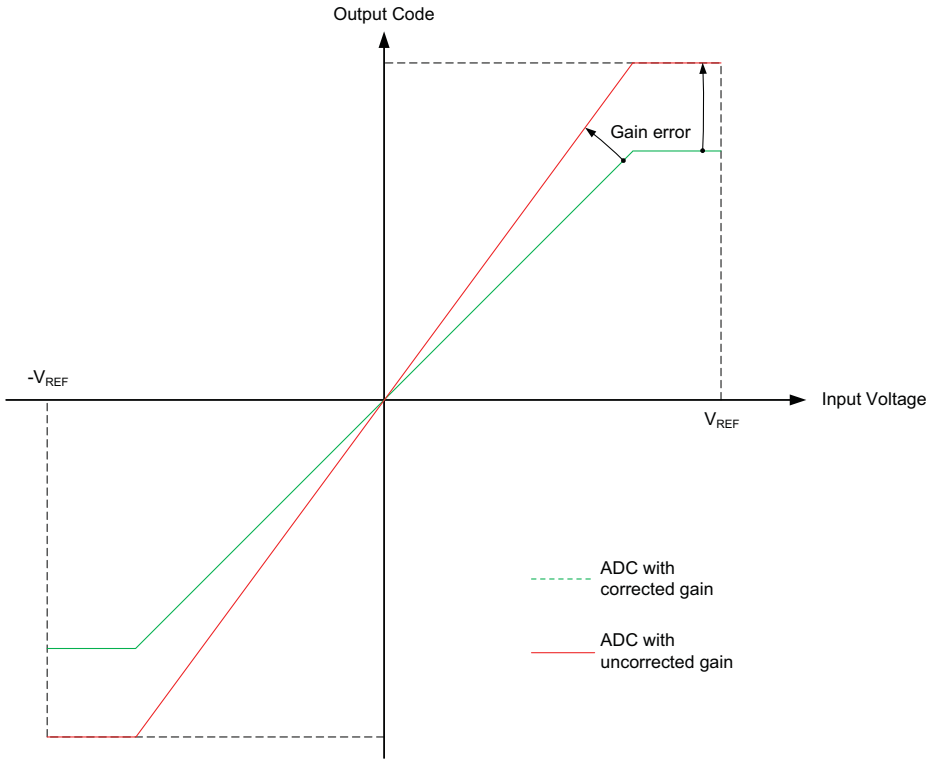


Figure 24-9. ADC gain error.



24.7.3 Offset error measurement

a. Configure MUXPOS and MUXNEG to connect both the inputs of ADC to the same value (GND is recommended)
b. Start a conversion on the channel
c. Wait for interrupt
d. Read the value from channel result register (RES) which corresponds to OFFSETCORR value

24.7.4 Gain error measurement

a. Configure MUXPOS and MUXNEG to connect each input of ADC to appropriate voltage levels to produce close to maximum code taking into account the gain factor and a possible saturation
b. Start a conversion on the channel
c. Wait for interrupt
d. Read the value from channel result register (RES) which corresponds to captured value (*CapturedValue)*.

Gain correction is coded as:

$$GAINCORR = 2048 \times \frac{ExpectedValue}{CapturedValue - OFFSETCORR}$$

Notes:
1. GAINCORR precision is 1-bit integer + 11-bit fraction, implies $0.5 \leq GAINCORR < 2.0$
2. GAINCORR range is from 0x0400 (0.5) up to 0x0FFF (1.99951171875)
3. No gain correction (1x gain) is set when GAINCORR = 0x0800 (1.0)

24.8 Starting a conversion

Before a conversion is started, the desired input channel source must be selected. An ADC conversion can be started either by the application software writing to the start conversion bit, or from any of the event triggers in the Event System.

24.8.1 Input source scan

It is possible to select a range of consecutive input sources that is automatically scanned and measured when a conversion is started. This is done by setting the first (lowest) positive ADC channel input using the MUX control register, and a number of consecutive positive input sources. When a conversion is started, the first selected input source is measured and converted, then the positive input source selection is incremented after each conversion until it reaches the specified number of sources to scan.

24.8.2 Compare function

The ADC has a built-in 12-bit compare function. The ADC compare register can hold a 12-bit value that represents a threshold voltage. The ADC channel can be configured to automatically compare its result with this compare value to give an interrupt or event only when the result is above or below the threshold.

24.8.3 Averaging

The ADC has inherently 12-bit resolution but it is possible to obtain 16-bit results by averaging up to 1024 samples. The numbers of samples to be averaged is specified in AVGCTRL register and the averaged output is written to channel output register.

The number of samples to be averaged is set by the SAMPNUM bits in “ AVGCTRL – Average Control register” on page 370. A maximum of 1024 samples can be averaged. The final result is rounded off to 16-bit value. After accumulating programmed number of bits, division is achieved by automatic right shifting and result will be available in channel result register (RES).

The output is calculated as per following formula:

$$Output = (\sum 2^{SAMPNUM} 16bitRoundOff) >> \text{RIGHTSHIFT}$$

For SAMPNUM > 0, Table 24-3 shows the number of samples which will be accumulated and the automatic number of right shifts internally performed.

Table 24-3. Final result versus number of samples to average.

Final result resolution	Number of samples	Number of automatic right shift for round-off
12-bits	1	0
13-bits	2	0
14-bits	4	0
15-bits	8	0
16-bits	16	0
16-bits	32	1
16-bits	64	2
16-bits	128	3
16-bits	256	4
16-bits	512	5
16-bits	1024	6

24.8.4 Over-sampling and decimation

Whilst averaging smooths out noise it doesn't increase the resolution. It is possible to increase the resolution of the ADC provided sufficient noise is present in the system and the sampling frequency is adequately higher than the signal frequency. Refer application note AVR1629. In over-sampling and decimation mode the ADC resolution is increased from 12-bit to programmed 13, 14, 15 or 16 bit. If n-bit resolution is to be increased, 4^n samples are accumulated (added) and the result is right shifted by n-bits. This method will result in n-bit extra LSB bit resolution

For this increased resolution to be valid, the following assumptions have to be met:

- Input to ADC is over-sampled input and the corresponding pre-scalar setting is already done
- The ADC sampling frequency fs > (4^n * 2) * f i
  (f i = highest frequency component of input signal, n = number of bits increased)
- Artificial noise addition is assumed to be added if required to input for minimum LSB toggling
- Range of over-sampled output is assumed to be between 13-bits and 16-bits

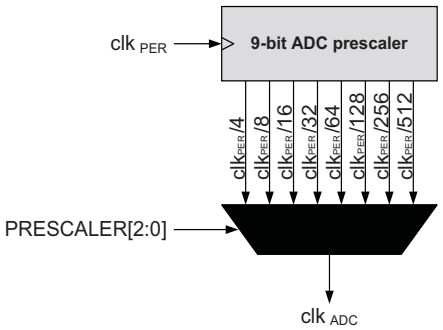
Table 24-4. Configuration required for averaging function to output corresponding over-sampled output.

Result resolution	Number of samples to average	SAMPNUM	No. of automatic right shift	RIGHTSHIFT
13-bit	4^1 = 4	0010	0	001
14-bit	4^2 = 16	0100	0	010
15-bit	4^3 = 64	0110	2	001
16-bit	4^4 = 256	1000	4	000

24.9 ADC clock and conversion timing

The ADC is clocked from the peripheral clock (clk_{PER}). The ADC can prescale the peripheral clock to provide an ADC Clock (clk_{ADC}) that matches the application requirements and is within the operating range of the ADC.

Figure 24-10.ADC prescaler.



The maximum ADC sample rate is given by:

$$SampleRate = \frac{f_{ADC}}{0.5 \cdot (RESOLUTION + SAMPVAL) + GAINFACTOR}$$

The propagation delay of an ADC measurement is given by:

$$PropagationDelay = \frac{1}{SampleRate}$$

where

- RESOLUTION is the resolution, 8 or 12 bits.
- SAMPVAL is the value programmed in the Sampling Time Control register.
- GAINFACTOR = 0 (1x gain), 1 (1/2x, 2x, 4x gain), 3 (32x, 64x gain)

The most-significant bit (msb) of the result is converted first, and the rest of the bits are converted during the next three (for 8-bit results) or five (for 12-bit results) ADC clock cycles. Converting one bit takes a half ADC clock period. During the last cycle, the result is prepared before the interrupt flag is set and the result is available in the result register for readout.

24.9.1 Single conversion with 1x gain

Figure 24-11 on page 352 shows the ADC timing for a single conversion with 1xgain. The writing of the start conversion bit, or the event triggering the conversion (START), must occur at least one peripheral clock cycle before the ADC clock cycle on which the conversion starts (indicated with the grey slope of the START trigger).

The input source is sampled in the first half of the first cycle.

Figure 24-11.ADC timing for one conversion with 1x gain.

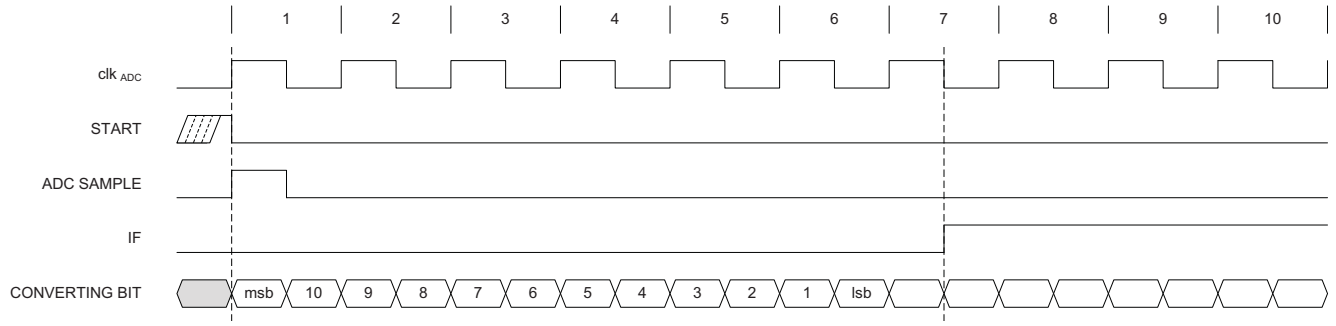
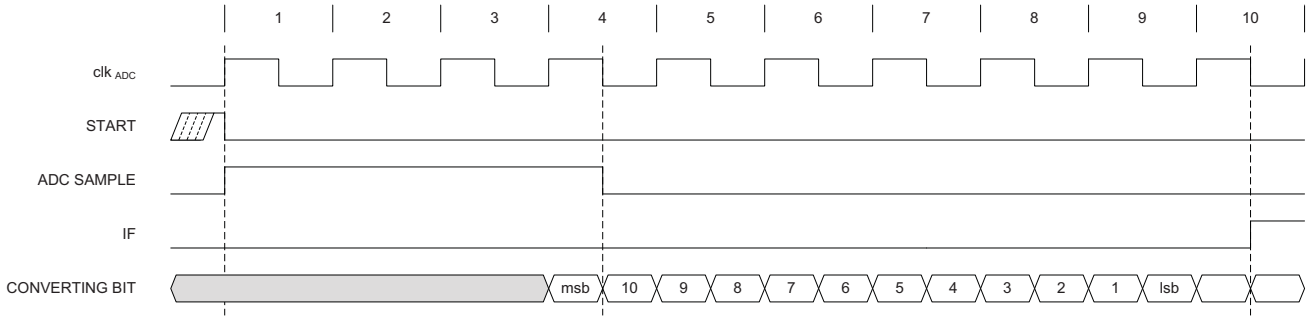


Figure 24-12.ADC timing for one conversion with increased sampling time (SAMPVAL = 6).



24.9.2 Single conversion with various gain settings

Figure 24-13 on page 353 to Figure 24-15 on page 353 show the ADC timing for one single conversion with various gain settings. As seen in the "Overview" on page 341, the gain stage is built into the ADC. Gain is achieved by running the signal through a pipeline stage without converting. Compared to a conversion without gain, each gain multiplication of 2 adds one half ADC clock cycle.

Figure 24-13.ADC timing for one single conversion with 2x gain.

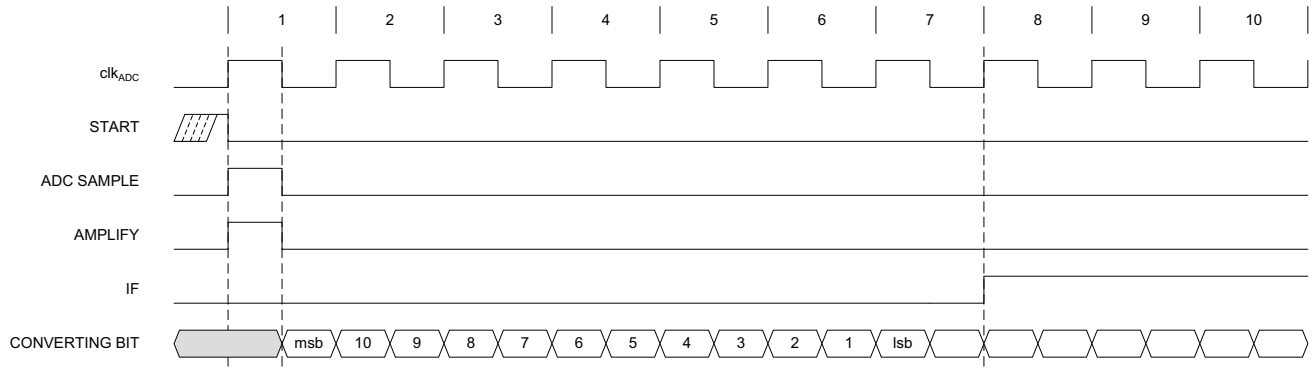


Figure 24-14.ADC timing for one single conversion with 8x gain.

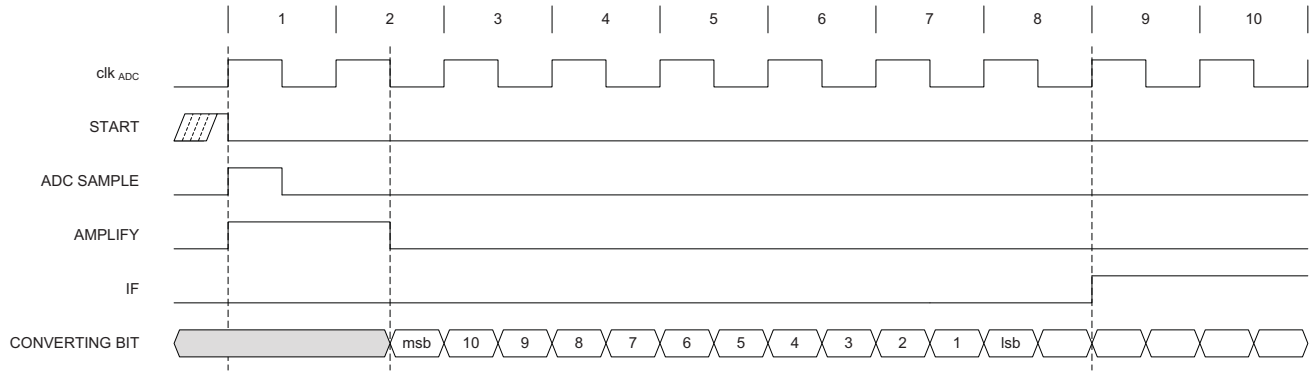
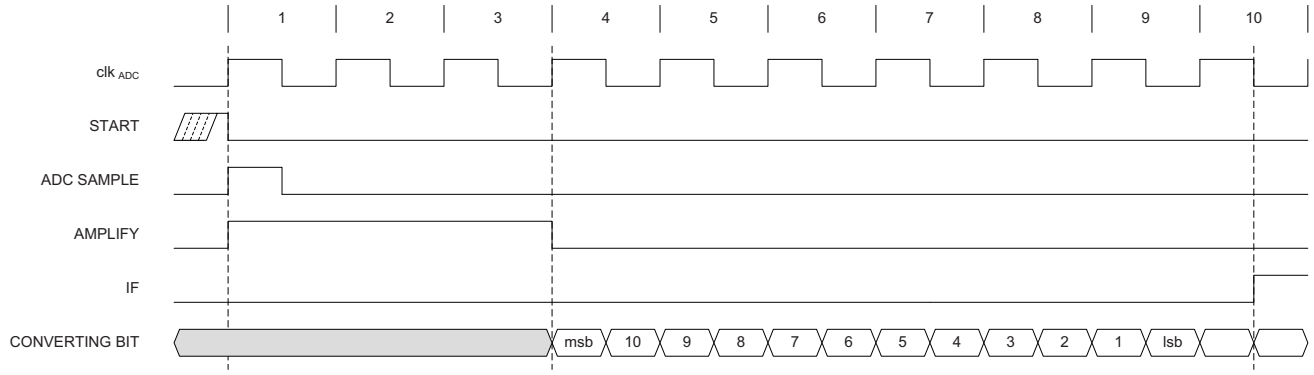


Figure 24-15.ADC timing for one single conversion with 64x gain.



24.10 ADC input model

The voltage input must charge the sample and hold (S/H) capacitor in the ADC in order to achieve maximum accuracy. Seen externally, the ADC input consists of an input resistance (R_{in} = $R_{channel}$ + R_{switch}) and the S/H capacitor (C_{sample}). Figure 24-16 on page 354 and Figure 24-17 on page 354 show the ADC input channel.

Figure 24-16.ADC input for single-ended measurements.

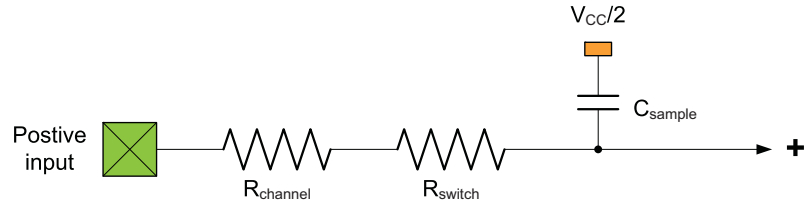
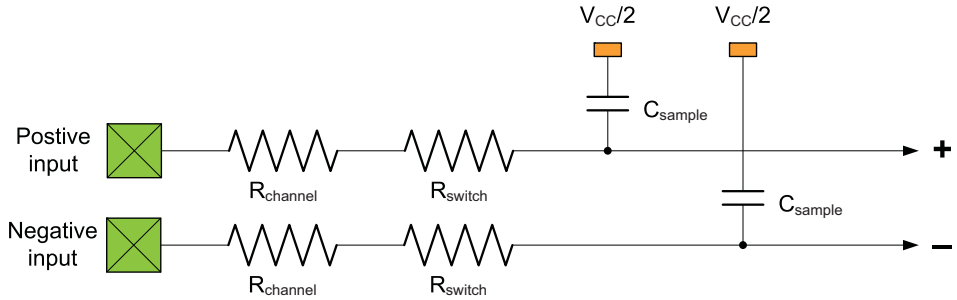


Figure 24-17.ADC input for differential measurements and differential measurement with gain.



In order to achieve *n* bits of accuracy, the source output resistance, R_{source}, must be less than the ADC input resistance on a pin:

$$R_{source} \leq \frac{T_s}{C_{sample} \times \ln(2^{n+1})} - R_{channel} - R_{switch}$$

where the ADC sample time, T_S is one-half the ADC clock cycle given by:

$$T_s \leq \frac{T_s}{2 \times f_{ADC}}$$

For details on $R_{channel}$, R_{switch}, and C_{sample}, refer to the ADC electrical characteristic in the device datasheet.

24.11 EDMA transfer

The EDMA controller can be used to transfer ADC conversion results to memory or other peripherals. A new conversion result for the ADC channel can trigger an EDMA transaction for the ADC channel. Refer to "EDMA – Enhanced Direct Memory Access" on page 50 for more details on EDMA transfers.

24.12 Interrupts and events

The ADC can generate interrupt requests and events. The ADC channel has individual interrupt settings and interrupt vectors. Interrupt requests and events can be generated when an ADC conversion is complete or when an ADC measurement is above or below the ADC compare register value.

24.13 Synchronous sampling

Starting an ADC conversion can cause an unknown delay between the start trigger or event and the actual conversion since the peripheral clock is faster than the ADC clock. To start an ADC conversion immediately on an incoming event, it is possible to flush the ADC of all measurements, reset the ADC clock, and start the conversion at the next peripheral clock cycle (which then will also be the next ADC clock cycle). If this is done, the ongoing conversions in the ADC will be lost.

The ADC can be flushed from software, or an incoming event can do this automatically. When this function is used, the time between each conversion start trigger must be longer than the ADC propagation delay to ensure that one conversion is finished before the ADC is flushed and the next conversion is started.

It is also important to clear pending events or start ADC conversion commands before doing a flush. If not, pending conversions will start immediately after the flush.

In devices with two ADC peripherals, it is possible to start two ADC samples synchronously in the two ADCs by using the same event channel to trigger both ADC.

24.14 Register description – ADC

24.14.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	START	FLUSH	ENABLE
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 2 – START: Start Single Conversion**
  Setting this bit will start an ADC conversion. This bit is cleared by hardware when the conversion has started. Writing this bit is equivalent to writing the START bit inside the ADC channel register.
- **Bit 1 – FLUSH: Flush**
  Writing this bit to one will flush the ADC. When this is done, the ADC clock will be restarted on the next peripheral clock edge and the conversion in progress is aborted and lost.
  After the flush and the ADC clock restart, the ADC will resume where it left off. I.e. if a sweep was in progress or any conversions was pending, these will enter the ADC complete.
- **Bit 0 – ENABLE: Enable**
  Setting this bit enables the ADC.

24.14.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	–	CURRLIMIT[1:0]		CONVMODE	FREERUN	RESOLUTION[1:0]		–
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bits 6:5 – CURRLIMIT[1:0]: Current Limitation**
  These bits can be used to limit the current consumption of the ADC by reducing the maximum ADC sample rate. The available settings are shown in Table 24-5 on page 356. The indicated current limitations are nominal values. Refer to the device datasheet for actual current limitation for each setting.

Table 24-5. ADC current limitations.

CURRLIMIT[1:0]	Group configuration	Description
00	NO	No limit
01	LOW	Low current limit, max. sampling rate 225kSPS
10	MED	Medium current limit, max. sampling rate 150kSPS
11	HIGH	High current limit, max. sampling rate 75kSPS

- **Bit 4 – CONVMODE: Conversion Mode**
  This bit controls whether the ADC will work in signed or unsigned mode. By default, this bit is cleared and the ADC is configured for unsigned mode. When this bit is set, the ADC is configured for signed mode.

- **Bit 3 – FREERUN: Free Run Mode**
  This bit controls the free running mode for the ADC. Once a conversion is finished, the next input will be sampled and converted.
- **Bits 2:1 – RESOLUTION[1:0]: Conversion Result Resolution**
  These bits define whether the ADC completes the conversion at 12- or 8-bit result resolution. They also define whether the 12-bit result is left or right adjusted within the 16-bit result registers. See Table 24-6 on page 357 for possible settings.

Table 24-6. ADC conversion result resolution.

RESOLUTION[1:0]	Group configuration	Description
00	12BIT	12-bit result, right adjusted
01	MT12BIT	More than 12-bit (oversampling) right adjusted result
10	8BIT	8-bit result, right adjusted
11	LEFT12BIT	12-bit result, left adjusted

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

24.14.3 REFCTRL – Reference Control register

Bit	7	6	5	4	3	2	1	0
+0x02	–	REFSEL[2:0]			–	–	BANDGAP	TEMPREF
Read/Write	R	R/W	R/W	R/W	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bits 6:4 – REFSEL[2:0]: Reference Selection**
  These bits set the reference settings and conversion range for the ADC according to Table 24-7 on page 357.

Table 24-7. ADC reference control.

REFSEL[2:0]	Group configuration	Description
000	INT1V	Internal 1.0 V
001	INTVCC	Internal V_{cc}/1.6
010 (1)	AREFA	External reference from AREF on port A
011 (2)	AREFD	External reference from AREF on port D
100	INTVCC2	Internal V_{CC}/2
101-111	–	Reserved

Notes: 1. Only available if AREF exists on port A
2. Only available if AREF exists on port D

- **Bits 3:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – BANDGAP: Bandgap Enable**
  Setting this bit enables the bandgap for ADC measurement. Note that if any other functions are already using the bandgap, this bit does not need to be set when the internal 1.00V reference is used for another ADC or if the brownout detector is enabled.
- **Bit 0 – TEMPREF: Temperature Reference Enable**
  Setting this bit enables the temperature sensor for ADC measurement.

24.14.4 EVCTRL – Event Control register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	EVSEL[2:0]			EVACT[2:0]		
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 5:3 – EVSEL[2:0]: Event Line Input Select**
  These bits define which event channel is used as trigger source for a conversion. See Table 24-8 on page 358.

Table 24-8. ADC event line select.

EVSEL[2:0]	Group configuration	Selected event lines
000	0	Event channel 0 as selected input
001	1	Event channel 1 as selected input
010	2	Event channel 2 as selected input
011	3	Event channel 3 as selected input
100	4	Event channel 4 as selected input
101	5	Event channel 5 as selected input
110	6	Event channel 6 as selected input
111	7	Event channel 7 as selected input

- **Bits 2:0 – EVACT[2:0]: Event Action**
  These bits select and limit how many of the selected event input channels are used, and also further limit the ADC channels triggers. They also define more special event triggers as defined in Table 24-9 on page 359.

Table 24-9. ADC event mode select.

EVACT[2:0]	Group configuration	Event input operation mode
000	NONE	No event inputs
001	CH0	Event channel with the lowest number defined by EVSEL triggers conversion on ADC channel
010	–	Reserved
011	–	Reserved
100	–	Reserved
101	–	Reserved
110	SYNCSWEEP	The ADC is flushed and restarted for accurate timing
111	–	Reserved

24.14.5 PRESCALER – Clock Prescaler register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	PRESCALER[2:0]		
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	1	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 2:0 – PRESCALER[2:0]: Prescaler Configuration**
  These bits define the ADC clock relative to the peripheral clock according to Table 24-10 on page 359.

Table 24-10. ADC prescaler settings.

PRESCALER[2:0]	Group configuration	Peripheral clock division factor
000	DIV4	4
001	DIV8	8
010	DIV16	16
011	DIV32	32
100	DIV64	64
101	DIV128	128
110	DIV256	256
111	DIV512	512

24.14.6 INTFLAGS – Interrupt Flags register

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	–	–	–	–	CH0IF
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – CH0IF: Interrupt Flag**
  This flag is set when the ADC conversion is complete. If the ADC is configured for compare mode, the corresponding flag will be set if the compare condition is met. CH0IF is automatically cleared when the ADC interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

24.14.7 TEMP – Temporary register

Bit	7	6	5	4	3	2	1	0
+0x07	TEMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – TEMP: Temporary Bits**
  This register is used when reading 16-bit registers in the ADC controller. The high byte of the 16-bit register is stored here when the low byte is read by the CPU. This register can also be read and written from the user software.
  For more details on 16-bit register access refer to“Accessing 16-bit registers” on page 13.

24.14.8 SAMPCTRL – Sampling Time Control register

Bit	7	6	5	4	3	2	1	0
+0x08	–	–	SAMPVAL[5:0]					
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 5:0 – SAMPVAL: Sampling Time Value**
  These bits control the ADC sampling time in number of half ADC prescaled clock cycles (depends of PRESCALER settings), thus controlling the ADC input impedance. Sampling time is set according to the formula:

$$\text{SamplingTime} = (\text{SAMPVAL} + 1) \times (clk_{\text{ADC}} / 2)$$

24.14.9 CALL – Calibration register Low

The CALH and CALL register pair represent the 12-bit value, CAL. The ADC is calibrated during production programming, and the calibration value must be read from the signature row and written to the CAL register from software, prior to gain and offset correction.

For more details on 16-bit register access refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x0C	CAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – CAL[7:0]: ADC Calibration Value Low Byte**
  These are the eight lsbs of the 12-bit CAL value.

24.14.10 CALH – Calibration register High

Bit	7	6	5	4	3	2	1	0
+0x0D	–	–	–	–	CAL[11:8]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 3:0 – CAL[11:8]: Calibration Value High Byte**
  These are the four msbs of the 12-bit CAL value.

24.14.11 CH0RESL – Channel 0 Result register Low

The CH0RESHand CH0RESL register pair represent the 12-bit value, CH0RES.

For more details on 16-bit register access refer to “Accessing 16-bit registers” on page 13.

Bit		7	6	5	4	3	2	1	0
+0x10	8-bit/12-bit, right	CH0RES[7:0]							
	12-bit, left	CH0RES[3:0]				–	–	–	–
Read/Write		R	R	R	R	R	R	R	R
Initial value		0	0	0	0	0	0	0	0

24.14.11.1 8-bit mode/12-bit mode, right adjusted

- **Bits 7:0 – CH0RES[7:0]: Channel Result Low Byte**
  These are the 8 lsbs of the ADC result.

24.14.11.2 12-bit mode, left adjusted

- **Bits 7:4 – CH0RES[3:0]: ADC Channel Result Low Byte**
  These are the 4 lsbs of the 12-bit ADC result.
- **Bits 3:0 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

24.14.12 CH0RESH – Channel 0 Result register High

Bit		7	6	5	4	3	2	1	0
+0x11	12-bit, left	CH0RES[11:4]							
	12-bit, right	–	–	–	–	CH0RES[11:8]			
	8-bit	–	–	–	–	–	–	–	–
Read/Write		R	R	R	R	R	R	R	R
Initial value		0	0	0	0	0	0	0	0

24.14.12.1 12-bit mode, left adjusted

- **Bits 7:0 – CH0RES[11:4]: ADC Channel Result High Byte**
  These are the 8 msbs of the 12-bit ADC result.

24.14.12.2 12-bit mode, right adjusted

- **Bits 7:4 – Reserved**
  These bits will in practice be the extension of the sign bit CH0RES11 when ADC works in differential mode and set to zero when ADC works in signed mode.
- **Bits 3:0 – CH0RES[11:8]: ADC Channel Result High Byte**
  These are the 4 msbs of the 12-bit ADC result.

24.14.12.3 8-bit mode

- **Bits 7:0 – Reserved**
  These bits will in practice be the extension of the sign bit CH0RES7 when ADC works in signed mode and set to zero when ADC works in single-ended mode.

24.14.13 CMPL – Compare register Low

The CMPL and CMPH register pair represent the 16-bit value, CMP.

For more details on 16-bit register access refer to “Accessing 16-bit registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x18	CMP[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – CMP[7:0]: Compare Value Low Byte**
  These are the 8 lsbs of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement.

24.14.14 CMPH – Compare register High

Bit	7	6	5	4	3	2	1	0
+0x19	CMP[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – CMP[15:0]: Compare Value High Byte**
  These are the 8 msbs of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement and the msbs is the sign bit.

24.15 Register description - ADC channel

24.15.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	START	–	–	GAIN[2:0]			INPUTMODE[1:0]	
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – START: START Conversion**
  Writing this to one will start a conversion on the channel. The bit is cleared by hardware when the conversion has started. Writing this bit to one when it already is set will have no effect. Writing this bit is equivalent to writing the START bit in " CTRLA – Control register A on page 356".
- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 4:2 – GAIN [2:0]: Gain Factor**
  These bits define the gain factor in order to amplify input signals before ADC conversion. See Table 24-11 for different gain factor settings. Gain is only valid in differential mode settings, as shown in Table 24-16 on page 365 and Table 24-17 on page 366. In single-ended mode of operation, the gain factor must be set to zero.

Table 24-11. ADC gain factor.

GAIN[2:0]	Group configuration	Gain factor
000	1X	1 x
001	2X	2 x
010	4X	4 x
011	8X	8 x
100	16X	16 x
101	32X	32 x
110	64X	64 x
111	DIV2	½ x

- **Bits 1:0 – INPUTMODE[1:0]: Channel Input Mode**
  These bits define the channel mode. This setting is independent of the ADC CONVMODE (signed/unsigned mode) setting, but differential input mode can only be done in ADC signed mode. In single ended input mode, the negative input to the ADC will be connected to a fixed value, both for ADC signed and unsigned mode.

Table 24-12. Channel input modes, CONVMODE = 0 (unsigned mode).

INPUTMODE[1:0]	Group configuration	Description
00	INTERNAL	Internal positive input signal
01	SINGLEENDED	Single-ended positive input signal
10	–	Reserved
11	–	Reserved

Table 24-13. Channel input modes, CONVMODE = 1 (signed mode).

INPUTMODE[1:0]	Group configuration	Description
00	INTERNAL	Internal positive input signal
01	SINGLEENDED	Single-ended positive input signal
10	DIFFWGAINL	Differential input signal with gain, 4 LSB pins available for MUXNEG selection
11	DIFFWGAINH	Differential input signal with gain, 4 MSB pins available for MUXNEG selection

24.15.2 MUXCTRL – MUX Control register

Bit	7	6	5	4	3	2	1	0
+0x01	–	MUXPOS[3:0]				MUXNEG[2:0]		
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bits 6:3 – MUXPOS[3:0]: MUX Selection on Positive ADC Input**
  These bits define the MUX selection for the positive ADC input. Table 24-14 on page 364 and Table 24-15 on page 365 shows the possible input selection for the different input modes.

Table 24-14. ADC MUXPOS configuration when INPUTMODE[1:0] = 00 (internal) is used.

MUXPOS[3:0]	Group configuration	Analog input
0000	TEMP	Temperature Reference
0001	BANDGAP	Bandgap
0010	SCALEDVCC	1/10 scaled VCC
0011	DAC	DAC Output
0100-1111	–	Reserved

Table 24-15. ADC MUXPOS configuration, INPUTMODE[1:0] = 01, 10, 11 (single-ended or differential with programmable gain).

MUXPOS[3:0]	Group configuration	Analog input (1)
0000	PIN0	ADC0 pin
0001	PIN1	ADC1 pin
0010	PIN2	ADC2 pin
0011	PIN3	ADC3 pin
0100	PIN4	ADC4 pin
0101	PIN5	ADC5 pin
0110	PIN6	ADC6 pin
0111	PIN7	ADC7 pin
1000	PIN8	ADC8 pin
1001	PIN9	ADC9 pin
1010	PIN10	ADC10 pin
1011	PIN11	ADC11 pin
1100	PIN12	ADC12 pin
1101	PIN13	ADC13 pin
1110	PIN14	ADC14 pin
1111	PIN15	ADC15 pin

Note: 1. Depending on the device pin count and feature configuration, the actual number of analog input pins may be less than 16. Refer to the device datasheet and pin-out description for details.

- **Bits 2:0 – MUXNEG[2:0]: MUX Selection on Negative ADC Input**
  These bits define the MUX selection for the negative ADC input when differential measurements are done. For internal or single-ended measurements, these bits are not in use. Table 24-16 on page 365 and Table 24-17 on page 366 shows the possible input sections.

Table 24-16. ADC MUXNEG configuration, INPUTMODE[1:0] = 10 (differential with programmable gain).

MUXNEG[2:0]	Group configuration	Analog input
000	PIN0	ADC0
001	PIN1	ADC1
010	PIN2	ADC2
011	PIN3	ADC3
100	–	Reserved
101	GND	PAD ground
110	–	Reserved
111	INTGND	Internal ground

Table 24-17. ADC MUXNEG configuration, INPUTMODE[1:0] = 11, (differential with programmable gain).

MUXNEG[2:0]	Group configuration	Analog input
000	PIN4	ADC4
001	PIN5	ADC5
010	PIN6	ADC6
011	PIN7	ADC7
100	–	Reserved
101	–	Reserved
110	–	Reserved
111	GND	PAD ground

24.15.3 INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	INTMODE[1:0]		INTLVL[1:0]	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 3:2 – INTMODE[1:0]: Interrupt Mode**
  INTMODE bits define the interrupt mode according to Table 24-18 on page 366.

Table 24-18. ADC interrupt mode.

INTMODE[1:0]	Group configuration	Interrupt mode
00	COMPLETE	Conversion Complete
01	BELOW	Compare Level Below Threshold
10	–	Reserved
11	ABOVE	Compare Level Above Threshold

- **Bits 1:0 – INTLVL[1:0]: Interrupt Priority Level and Enable**
  These bits enable the ADC channel interrupt and select the interrupt level as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will be triggered when IF is set in the INTFLAGS register.

24.15.4 INTFLAGS – Interrupt Flags register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	–	–	–	**IF**
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 – IF: Interrupt Flag**
  The interrupt flag is set when the ADC conversion is complete. If the channel is configured for compare mode, the flag will be set if the compare condition is met. IF is automatically cleared when the ADC channel interrupt vector is executed. The bit can also be cleared by writing a one to the bit location.

24.15.5 RESL – Result register Low

The RESH and RESL register pair represent the 12-bit value, RES.

For more details on 16-bit register access refer to “Accessing 16-bit registers” on page 13.

Bit		7	6	5	4	3	2	1	0
+0x04	8-bit/12-bit, right	**RES[7:0]**							
	12-bit, left	**RES[3:0]**				–	–	–	–
Read/Write		R	R	R	R	R	R	R	R
Initial value		0	0	0	0	0	0	0	0

24.15.5.1 8-bit mode/12-bit mode, right adjusted

- **Bits 7:0 – RES[7:0]: Check Police**
  These are the 8 lsbs of the ADC result.

24.15.5.2 12-bit mode, left adjusted

- **Bits 7:4 – RES[3:0]: Check Police**
  These are the 4 lsbs of the 12-bit ADC result.
- **Bits 3:0 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

24.15.6 RESH – Result register High

Bit		7	6	5	4	3	2	1	0
+0x05	12-bit, left	RES[11:4]							
	12-bit, right	–	–	–	–	RES[11:8]			
	8-bit	–	–	–	–	–	–	–	–
Read/Write		R	R	R	R	R	R	R	R
Initial value		0	0	0	0	0	0	0	0

24.15.6.1 12-bit mode, left adjusted

- **Bits 7:0 – RES[11:4]: Channel Result High Byte**
  These are the 8 msbs of the 12-bit ADC result.

24.15.6.2 12-bit mode, right adjusted

- **Bits 7:4 – Reserved**
  These bits will in practice be the extension of the sign bit RES11 when ADC works in differential mode and set to zero when ADC works in signed mode.
- **Bits 3:0 – RES[11:8]: Channel Result High Byte**
  These are the 4 msbs of the 12-bit ADC result.

24.15.6.3 8-bit mode

- **Bits 7:0 – Reserved**
  These bits will in practice be the extension of the sign bit RES7 when ADC works in signed mode and set to zero when ADC works in single-ended mode.

24.15.7 SCAN – Scan register

Bit	7	6	5	4	3	2	1	0
+0x06	**INPUTOFFSET[3:0]**				**INPUTSCAN[3:0]**			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:4 – INPUTOFFSET[3:0]: Positive MUX Setting Offset**
  The channel scan is enabled when INPUTSCAN is not equal to 0 and this register contains the offset for the next input source to be converted on ADC channel. The actual MUX setting for positive input equals MUXPOS + INPUTOFFSET. The value is incremented after each conversion until it reaches the maximum value given by INPUTSCAN. When INPUTOFFSET is equal to INPUTSCAN, INPUTOFFSET will be cleared on the next conversion.
- **Bits 3:0 – INPUTSCAN[3:0]: Number of Input Channels Included in Scan**
  This register gives the number of input sources included in the channel scan. The number of input sources included is INPUTSCAN + 1. The input channels included are the range from MUXPOS + INPUTOFFSET to MUXPOS + INPUTOFFSET + INPUTSCAN.

24.15.8 CORRCTRL - Correction Control register

Bit	7	6	5	4	3	2	1	0
+0x07	–	–	–	–	–	–	–	**CORREN**
Read/Write	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 – CORREN: Correction Enable**
  Writing one to this bit enables the offset and gain correction. When enabled, 13 clk_{PER} latency for the final output is added. In free running mode, the latency is added on the first conversion only.
  When disabled, the ADC output result is not corrected for offset and gain.

24.15.9 OFFSETCORR0 – Offset Correction register 0

The OFFSETCORR1 and OFFSETCORR0 register pair stores the 12-bit value, OFFSETCORR. This pair has no 16-bit register access type.

There register values are ignored if the CORREN bit is cleared.

Bit	7	6	5	4	3	2	1	0
+0x08	OFFSETCORR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – OFFSETCORR[7:0] – Offset Correction Byte 0**
  These bits are the eight lsbs of the 12-bit offset correction value.

24.15.10 OFFSETCORR1 – Offset Correction register 1

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	OFFSETCORR[11:8]			
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 3:0 – OFFSETCORR[11:8] – Offset Correction Byte 1**
  These bits are the four msbs of the 12-bit offset correction value.

24.15.11 GAINCORR0 – Gain Correction register 0

The GAINCORR1 and GAINCORR0 register pair stores the 12-bit value, GAINCORR. This pair has no 16-bit register access type.

There register values are ignored if the CORREN bit is cleared.

Bit	7	6	5	4	3	2	1	0
+0x0A	GAINCORR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:0 – GAINCORR[7:0] – Gain Correction Byte 0**
  These bits are the eight lsbs of the 12-bit gain correction value.

24.15.12 GAINCORR1 – Gain Correction register 1

Bit	7	6	5	4	3	2	1	0
+0x0B	–	–	–	–	GAINCORR[11:8]			
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bits 3:0 – GAINCORR[11:8] – Gain Correction Byte 1**
  These bits are the four msbs of the 12-bit gain correction value.

24.15.13 AVGCTRL – Average Control register

Bit	7	6	5	4	3	2	1	0
+0x0C	–	RIGHTSHIFT[2:0]			SAMPNUM3:0]			
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is reserved and will always read as zero. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6:4 – RIGHTSHIFT[2:0] – Right Shift**
  This value is effective only if SAMPNUM > 0 and if over-sampling mode is required. Output value will be in RES 16-bit register. Accumulated value will be right shifted by the value specified by this bits. Right shift is from 0-shift till 7-shift.
  In averaging mode, these bits must set to zero.
- **Bit 3:0 – SAMPNUM[3:0] - Averaged Number of Samples**
  This value is effective only if SAMPNUM > 0. The below table specify the number of samples which will be accumulated and result will be available in RES 16-bit register.

Table 24-19. Number of samples.

SAMPNUM[3:0]	Group configuration	Number of samples
0000	1X	1
0001	2X	2
0010	4X	4
0011	8X	8
0100	16X	16
0101	32X	32
0110	64X	64
0111	128X	128
1000	256X	256
1001	512X	512
1010	1024X	1024
1011-1111	–	Reserved

24.16 Register summary – ADC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	–	START	FLUSH	ENABLE	356
+0x01	CTRLB	–	CURRLIMIT[1:0]		CONVMODE	FREERUN	RESOLUTION[1:0]		–	356
+0x02	REFCTRL	–	REFSEL[2:0]			–	–	BANDGAP	TEMPREF	357
+0x03	EVCTRL	–	–	EVSEL[2:0]			EVACT[2:0]			358
+0x04	PRESCALER	–	–	–	–	–	PRESCALER[2:0]			359
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	INTFLAGS	–	–	–	–	–	–	–	CH0IF	360
+0x07	TEMP	TEMP[7:0]								360
+0x08	SAMPCTRL	–	–	SAMPVAL[5:0]						360
+0x09	Reserved	–	–	–	–	–	–	–	–	
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	CALL	CAL[7:0]								361
+0x0D	CALH	–	–	–	–	CAL[11:8]				361
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10	CH0RESL	CH0RES[7:0]								361
+0x11	CH0RESH	CH0RES[15:8]								362
+0x12	Reserved	–	–	–	–	–	–	–	–	
+0x13	Reserved	–	–	–	–	–	–	–	–	
+0x14	Reserved	–	–	–	–	–	–	–	–	
+0x15	Reserved	–	–	–	–	–	–	–	–	
+0x16	Reserved	–	–	–	–	–	–	–	–	
+0x17	Reserved	–	–	–	–	–	–	–	–	
+0x18	CMPL	CMP[7:0]								362
+0x19	CMPH	CMP[15:8]								362
+0x1A	Reserved	–	–	–	–	–	–	–	–	
+0x1B	Reserved	–	–	–	–	–	–	–	–	
+0x1C	Reserved	–	–	–	–	–	–	–	–	
+0x1D	Reserved	–	–	–	–	–	–	–	–	
+0x1E	Reserved	–	–	–	–	–	–	–	–	
+0x1F	Reserved	–	–	–	–	–	–	–	–	
+0x20	CH0 Offset	Offset address for ADC Channel								
+0x28	Reserved	–	–	–	–	–	–	–	–	
+0x30	Reserved	–	–	–	–	–	–	–	–	
+0x38	Reserved	–	–	–	–	–	–	–	–	

24.17 Register summary – ADC channel

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	START	–	–	GAIN[2:0]			INPUTMODE[1:0]		363
+0x01	MUXCTRL	–	MUXPOS[3:0]				MUXNEG[2:0]			364
+0x02	INTCTRL	–	–	–	–	INTMODE[1:0]		INTLVL[1:0]		366
+0x03	INTFLAGS	–	–	–	–	–	–	–	IF	367
+0x04	RESL	RES[7:0]								367
+0x05	RESH	RES[15:8]								368
+0x06	SCAN	INPUTOFFSET[3:0]				INPUTSCAN[3:0]				368
+0x07	CORRCTRL	–	–	–	–	–	–	–	CORREN	368
+0x08	OFFSETCORR0	OFFSETCORR[7:0]								369
+0x09	OFFSETCORR1	–	–	–	–	OFFSETCORR[11:8]				369
+0x0A	GAINCORR0	GAINCORR[7:0]								369
+0x0B	GAINCORR1	–	–	–	–	GAINCORR[11:8]				370
+0x0C	AVGCTRL	–	RIGHTSHIFT[2:0]			SAMPNUM[3:0]				370
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	

24.18 Interrupt vector summary

Table 24-20. Analog-to-digital convertor interrupt vectors and their word offset address.

Offset	Source	Interrupt description
0x00	CH0_vect	Analog-to-digital convertor channel 0 interrupt vector

25. DAC – Digital to Analog Converter

25.1 Features

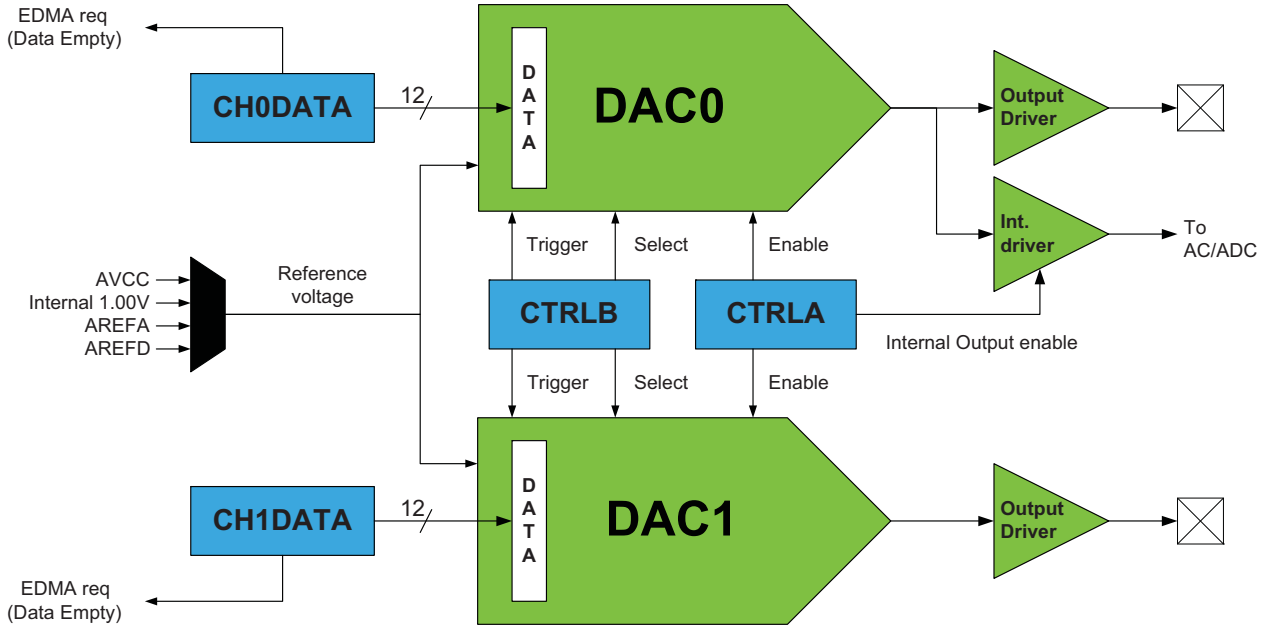
- 12-bit resolution
- Two independent, continuous-drive output channels
- Up to one million samples per second conversion rate per DAC channel
- Built-in calibration that removes:
  - Offset error
  - Gain error
- Multiple conversion trigger sources
  - On new available data
  - Events from the event system
- High drive capabilities and support for:
  - Resistive loads
  - Capacitive loads
  - Combined resistive and capacitive loads
- Internal and external reference options
- DAC output available as input to analog comparator and ADC
- Low-power mode, with reduced drive strength
- Optional EDMA transfer of data

25.2 Overview

The digital-to-analog converter (DAC) converts digital values to voltages. The DAC has two channels, each with12-bit resolution, and is capable of converting up to one million samples per second (MSPS) on each channel. The built-in calibration system can remove offset and gain error when loaded with calibration values from software.

Figure 25-1 illustrates the basic functionality of the DAC. Not all functions are shown.

Figure 25-1. DAC overview.



A DAC conversion is automatically started when new data to be converted are available. Events from the event system can also be used to trigger a conversion, and this enables synchronized and timed conversions between the DAC and other peripherals, such as a timer/counter. The EDMA controller can be used to transfer data to the DAC.

The DAC has high drive strength, and is capable of driving both resistive and capacitive loads, as well as loads which combine both. A low-power mode is available, which will reduce the drive strength of the output.

Internal and external voltage references can be used. The DAC output is also internally available for use as input to the analog comparator or ADC.

25.3 Voltage reference selection

The following can be used as the reference voltage (VREF) for the DAC:

- AV_{CC} voltage
- Accurate internal 1.00V voltage
- External voltage applied to AREF pin on PORTA
- External voltage applied to AREF pin on PORTD

25.4 Starting a conversion

By default, conversions are started automatically when new data are written to the channel data register. It is also possible to enable events from the event system to trigger conversion starts. When enabled, a new conversion is started when the DAC channel receives an event and the channel data register has been updated. This enables conversion starts to be synchronized with external events and/or timed to ensure regular and fixed conversion intervals.

25.5 Output and output channels

The two DAC channels have fully independent outputs and individual data and conversion control registers. This enables the DAC to create two different analog signals. The channel 0 output can also be made internally available as input for the Analog Comparator and the ADC.

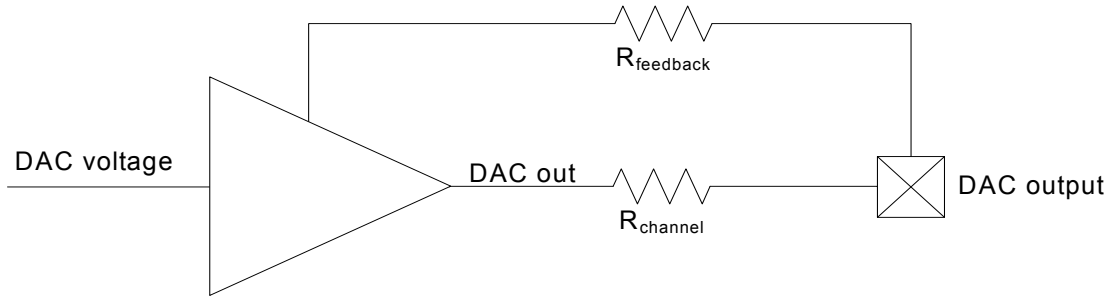
The output voltage from a DAC channel (V_{DAC}) is given as:

$$V_{DACn} = \frac{CHnDATA}{0xFFF} \times VREF$$

25.6 DAC output model

Each DAC output channel has a driver buffer with feedback to ensure that the voltage on the DAC output pin is equal to the DACs internal voltage. Section 25-2 shows the DAC output model. For details on $R_{channel}$, refer to the DAC characteristics in the device data sheet.

Figure 25-2. DAC output model



25.7 DAC clock

The DAC is clocked directly from the peripheral clock (clk_{PER}), and this puts a limitation on how fast new data can be clocked into the DAC data registers.

25.8 Low power mode

To reduce the power consumption in DAC conversions, the DAC may be set in a low power mode. Conversion time will be longer if new conversions are started in this mode. This increases the DAC conversion time per DAC channel by a factor of two.

25.9 Calibration

For improved accuracy, it is possible to calibrate for gain and offset errors in the DAC.

To get the best calibration result, it is recommended to use the same DAC configuration during calibration as will be used in the final application. The theoretical transfer function for the DAC was shown in the equation in "Output and output channels" on page 374. Including gain and offset errors, the DAC output value can be expressed as:

Equation 25-1. Calculation of DAC output value

$$V_{DAC} = VREF \cdot \left(\frac{DATA}{0xFFF} \cdot ERROR_{GAIN}\right) + V_{OFFSET}$$

To calibrate for offset error, output the DAC channel's middle code (0x800) and adjust the offset calibration value until the measured output value is as close as possible to the middle value (VREF / 2). The formula for the offset calibration is given by the Equation 25-2 on page 375, where OCAL is OFFSETCAL and GCAL is GAINCAL.

Equation 25-2. Offset calibration.

$$V_{OCAL} = VREF \cdot (2.OCAL[7] - 1) \cdot \left(\frac{OCAL[6]}{64} + \frac{OCAL[5]}{128} + \frac{OCAL[4]}{256} + \frac{OCAL[3]}{512} + \frac{OCAL[2]}{1024} + \frac{OCAL[1]}{2048} + \frac{OCAL[0]}{4096}\right)$$

To calibrate for gain error, output the DAC channel's maximum code (0xFFF) and adjust the gain calibration value until the measured output value is as close as possible to the top value (VREF x 4095 / 4096). The gain calibration controls the slope of the DAC characteristic by rotating the transfer function around the middle code. The formula for gain calibration is given by the Equation 25-3 on page 375.

Equation 25-3. Gain calibration.

$$V_{GCAL} = \left(V_{DAC} - \left(\frac{VREF}{2}\right)\right) \cdot (1 - 2.GCAL[7]) \cdot \left(\frac{GCAL[6]}{16} + \frac{GCAL[5]}{32} + \frac{GCAL[4]}{64} + \frac{GCAL[3]}{128} + \frac{GCAL[2]}{256} + \frac{GCAL[1]}{512} + \frac{GCAL[0]}{1024}\right)$$

Including calibration in the equation, the DAC output can be expressed by Equation 25-4 on page 375.

Equation 25-4. DAC output calculation

$$V_{DAC\_out} = V_{DAC} + V_{OCAL} + V_{GCAL}$$

25.10 Register description

25.10.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	IDOEN	CH1EN	CH0EN	LPMODE	ENABLE
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bite are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – IDOEN: Internal Output Enable**
  Setting this bit will enable the internal DAC channel 0 output to be used by the Analog Comparator and ADC. This will then also disable the output pin for DAC Channel 0.
- **Bit 3 – CH1EN: Channel 1 Output Enable**
  Setting this bit will make channel 1 available on the output pin.
- **Bit 2 – CH0EN: Channel 0 Output Enable**
  Setting this bit will make channel 0 available on the output pin unless IDOEN is set to 1.
- **Bit 1 – LPMODE: Low Power Mode**
  Setting this bit enables the DAC low-power mode. The DAC is turned off between each conversion to save current. Conversion time will be doubled when new conversions are started in this mode.
- **Bit 0 – ENABLE: Enable**
  This bit enables the entire DAC.

25.10.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	–	CHSEL[1:0]		–	–	–	CH1TRIG	CH0TRIG
Read/Write	R	R/W	R/W	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 6:5 – CHSEL[1:0]: Channel Selection**
  These bits control which DAC channels are enabled and operating. Table 25-1 shows the available selections.

Table 25-1. DAC channel selection.

CHSEL[1:0]	Group configuration	Description
00	SINGLE	Single-channel operation on channel 0
01	SINGLE1	Single-channel operation on channel 1
10	DUAL	Dual-channel operation
11	–	Reserved

- **Bit 4:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 – CH1TRIG: Auto Trigged Mode Channel 1**
  If this bit is set, an event on the configured event channel, set in EVCTRL, will trigger a conversion on DAC channel 1 if its data register, CH1DATA, has been updated.
- **Bit 0 – CH0TRIG: Auto Trigged Mode Channel 0**
  If this bit is set, an event on the configured event channel, set in EVCTRL, will trigger a conversion on DAC channel 0 if its data register, CH0DATA, has been updated.

25.10.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	REFSEL[1:0]		–	–	LEFTADJ
Read/Write	R	R	R	R/W	R/W	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4:3 – REFSEL[1:0]: Reference Selection**
  These bits select the reference voltage for the DAC according to Table 25-2 on page 377.

Table 25-2. DAC reference selection.

CHSEL[1:0]	Group configuration	Description
00	INT1V	Internal 1.00V
01	AVCC	AV_{CC}
10	AREFA	AREF on PORTA
11	AREFD	AREF on PORTD

- **Bit 2:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 0 - LEFTADJ: Left-Adjust Value**
  If this bit is set, CH0DATA and CH1DATA are left-adjusted.

25.10.4 EVCTRL – Event Control register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	EVSEL[3:0]			
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3 – EVSEL[3]: Event Selection Bit 3**
  Setting this bit to 1 enables event channel EVSEL[2:0]+1 as the trigger source for DAC Channel 1. When this bit is 0, the same event channel is used as the trigger source for both DAC channels.
- **Bit 2:0 – EVSEL[2:0]: Event Channel Input Selection**
  These bits select which Event System channel is used for triggering a DAC conversion. Table 25-3 on page 378 shows the available selections.

Table 25-3. DAC reference selection.

EVSEL[2:0]	Group configuration	Description
000	0	Event channel 0 as input to DAC
001	1	Event channel 1 as input to DAC
010	2	Event channel 2 as input to DAC
011	3	Event channel 3 as input to DAC
100	4	Event channel 4 as input to DAC
101	5	Event channel 5 as input to DAC
110	6	Event channel 6 as input to DAC
111	7	Event channel 7 as input to DAC

25.10.5 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	–	–	–	–	CH1DRE	CH0DRE
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – CH1DRE: Channel 1 Data Register Empty**
  This bit when set indicates that the data register for channel 1 is empty, meaning that a new conversion value may be written. Writing to the data register when this bit is cleared will cause the pending conversion data to be over-written. This bit is directly used for EDMA requests.
- **Bit 0 – CH0DRE: Channel 0 Data Register Empty**
  This bit when set indicates that the data register for channel 0 is empty, meaning that a new conversion value may be written. Writing to the data register when this bit is cleared will cause the pending conversion data to be over-written. This bit is directly used for EDMA requests.

25.10.6 CH0GAINCAL – Gain Calibration register

Bit	7	6	5	4	3	2	1	0
+0x08	CH0GAINCAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CH0GAINCAL[7:0]: Gain Calibration Value**
  These bits are used to compensate for the gain error in DAC channel 0. See “Calibration” on page 375 for details.

25.10.7 CH0OFFSETCAL – Offset Calibration register

Bit	7	6	5	4	3	2	1	0
+0x09	CH0OFFSETCAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CH0OFFSETCAL[7:0]: Offset Calibration Value**
  These bits are used to compensate for the offset error in DAC channel 0. See “Calibration” on page 375 for details.

25.10.8 CH1GAINCAL – Gain Calibration register

Bit	7	6	5	4	3	2	1	0
+0x0A	CH1GAINCAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CH1GAINCAL[7:0]: Gain Calibration Value**
  These bits are used to compensate for the gain error in DAC channel 1. See “Calibration” on page 375 for details.

25.10.9 CH1OFFSETCAL – Offset Calibration register

Bit	7	6	5	4	3	2	1	0
+0x0B	CH1OFFSETCAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CH1OFFSETCAL[7:0]: Offset Calibration Value**
  These bits are used to compensate for the offset error in DAC channel 1. See “Calibration” on page 375 for details.

25.10.10CH0DATAH – Channel 0 Data register High

These two channel data registers, CHnDATAH and CHnDATAL, are the high byte and low byte, respectively, of the 12-bit CHnDATA value that is converted to a voltage on DAC channel n. By default, the 12 bits are distributed with 8 bits in CHnDATAL and 4 bits in the four lsb positions of CHnDATAH (right-adjusted).To select left-adjusted data, set the LEFTADJ bit in the CTRLC register.

When left adjusted data is selected, it is possible to do 8-bit conversions by writing only to the high byte of CHnDATA, i.e., CHnDATAH. The TEMP register should be initialized to zero if only 8-bit conversion mode is used.

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x19	–	–	–	–	CHDATA[11:8]			
Left-adjust		CHDATA[11:4]							
Right-adjust	Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

25.10.10.1Right-adjusted

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:0 – CHDATA[11:8]: Conversion Data Channel 0, Four MSB Bits**
  These bits are the four msbs of the 12-bit value to convert to channel 0 in right-adjusted mode.

25.10.10.2Left-adjusted

- **Bits 7:0 –- CHDATA[11:4]: Conversion Data Channel 0, Eight MSB Bits**
  These bits are the eight msbs of the 12-bit value to convert to channel 0 in left-adjusted mode

25.10.11CH0DATAL – Channel 0 Data register Low

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x18	CHDATA[7:0]							
Left-adjust		CHDATA[3:0]				–	–	–	–
Right-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

25.10.11.1Right-adjusted

- **Bit 7:0 – CHDATA[7:0]: Conversion Data Channel 0, Eight LSB Bits**
  These bits are the eight lsbs of the 12-bit value to convert to channel 0 in right-adjusted mode.

25.10.11.2Left-adjusted

- **Bit 7:4 – CHDATA[3:0]: Conversion Data Channel 0, Four LSB Bits**
  These bits are the four lsbs of the 12-bit value to convert to channel 0 in left-adjusted mode.
- **Bit 3:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

25.10.12CH1DATAH – Channel 1 Data register High

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x1B	–	–	–	–	CHDATA[11:8]			
Left-adjust		CHDATA[11:4]							
Right-adjust	Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

25.10.12.1Right-adjusted

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:0 – CHDATA[11:8]: Conversion Data Channel 1, Four MSB Bits**
  These bits are the four msbs of the 12-bit value to convert to channel 1 in right-adjusted mode.

25.10.12.2Left-adjusted

- **Bit 7:0 – CHDATA[11:4]: Conversion Data Channel 1, Eight MSB Bits**
  These bits are the eight msbs of the 12-bit value to convert to channel 1 in left-adjusted mode.

25.10.13CH1DATAL – Channel 1 Data register Low

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x1A	CHDATA[7:0]							
Left-adjust		CHDATA[3:0]				–	–	–	–
Right-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

25.10.13.1Right-adjusted

- **Bit 7:0 – CHDATA[7:0]: Conversion Data Channel 1, Eight LSB Bits**
  These bits are the eight lsbs of the 12-bit value to convert to channel 1 in right-adjusted mode.

25.10.13.2Left-adjusted

- **Bits 7:4 – CHDATA[3:0]: Conversion Data Channel 1, Four LSB Bits**
  These bits are the four lsbs of the 12-bit value to convert to channel 1 in left-adjusted mode.
- **Bit 3:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

25.11 Register summary

This is the I/O summary when the DAC is configured to give standard 12-bit results. The I/O summary for 12-bit left-adjusted results will be similar, but with some changes in the CHnDATAL and CHnDATAH data registers.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	IDOEN	CH1EN	CH0EN	LPMODE	ENABLE	376
+0x01	CTRLB	–	CHSEL[1:0]		–	–	–	CH1TRIG	CH0TRIG	376
+0x02	CTRLC	–	–	–	REFSEL[1:0]		–	–	LEFTADJ	377
+0x03	EVCTRL	–	–	–	–	EVSEL[3:0]				377
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	STATUS	–	–	–	–	–	–	CH1DRE	CH0DRE	378
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	CH0GAINCAL	CH0GAINCAL[7:0]								378
+0x09	CH0OFFSETCAL	CH0OFFSETCAL[7:0]								378
+0x0A	CH1GAINCAL	CH1GAINCAL[7:0]								379
+0x0B	CH1OFFSETCAL	CH1OFFSETCAL[7:0]								379
+0x12	Reserved	–	–	–	–	–	–	–	–	
+0x13	Reserved	–	–	–	–	–	–	–	–	
+0x14	Reserved	–	–	–	–	–	–	–	–	
+0x15	Reserved	–	–	–	–	–	–	–	–	
+0x16	Reserved	–	–	–	–	–	–	–	–	
+0x17	Reserved	–	–	–	–	–	–	–	–	
+0x18	CH0DATAL	CHDATA[7:0]								380
+0x19	CH0DATAH	–	–	–	–	CHDATA[11:8]				379
+0x1A	CH1DATAL	CHDATA[7:0]								381
+0x1B	CH1DATAH	–	–	–	–	CHDATA[11:8]				380

26. AC – Analog Comparator

26.1 Features

- Selectable hysteresis
  - None
  - Small
  - Large
- Analog comparator output available on pin
- Flexible input selection
  - All pins on the port
  - Output from the DAC
  - Bandgap reference voltage
  - A 64-level programmable voltage scaler of the internal V_{CC} voltage
- Interrupt and event generation on:
  - Rising edge
  - Falling edge
  - Toggle
- Window function interrupt and event generation on:
  - Signal above window
  - Signal inside window
  - Signal below window
- Constant current source with configurable output pin selection
- Source of asynchronous event

26.2 Overview

The analog comparator (AC) compares the voltage levels on two inputs and gives a digital output based on this comparison. The analog comparator may be configured to generate interrupt requests and/or synchronous/ asynchronous events upon several different combinations of input change.

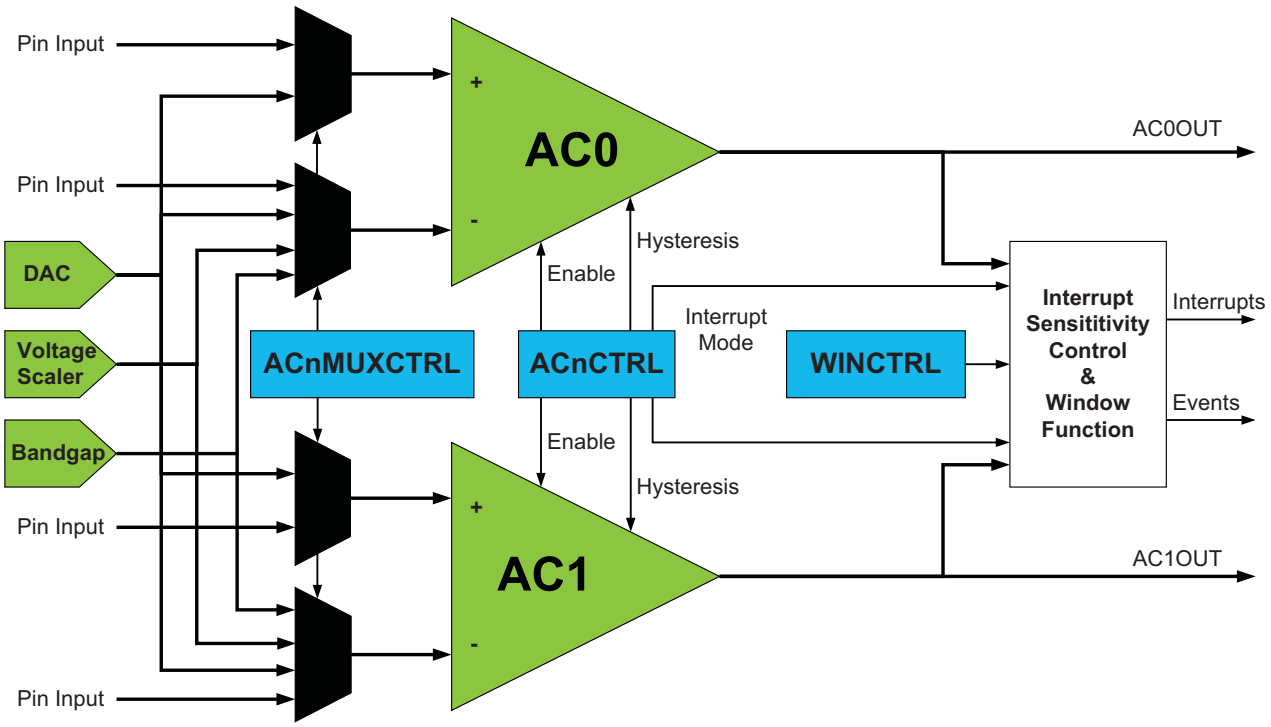
The important property of the analog comparator's dynamic behavior is the hysteresis. It can be adjusted in order to achieve the optimal operation for each application.

The input selection includes analog port pins, several internal signals, and a 64-level programmable voltage scaler. The analog comparator output state can also be output on a pin for use by external devices.

A constant current source can be enabled and output on a selectable pin. This can be used to replace, for example, external resistors used to charge capacitors in capacitive touch sensing applications.

The analog comparators are always grouped in pairs on each port. These are called analog comparator 0 (AC0) and analog comparator 1 (AC1). They have identical behavior, but separate control registers. Used as pair, they can be set in window mode to compare a signal to a voltage range instead of a voltage level.

Figure 26-1. Analog comparator overview.



26.3 Input sources

Each analog comparator has one positive and one negative input. Each input may be chosen from a selection of analog input pins and internal inputs such as a V_{CC} voltage scaler. The digital output from the analog comparator is one when the difference between the positive and the negative input voltage is positive, and zero otherwise.

26.3.1 Pin inputs

Any of analog input pins on the port can be selected as input to the analog comparator.

26.3.2 Internal inputs

Three internal inputs are available for the analog comparator:

- Output from the DAC
- Bandgap reference voltage
- Voltage scaler, which provides a 64-level scaling of the internal V_{CC} voltage

26.4 Signal compare

In order to start a signal comparison, the analog comparator must be configured with the preferred properties and inputs before the module is enabled. The result of the comparison is continuously updated and available for application software and the event system.

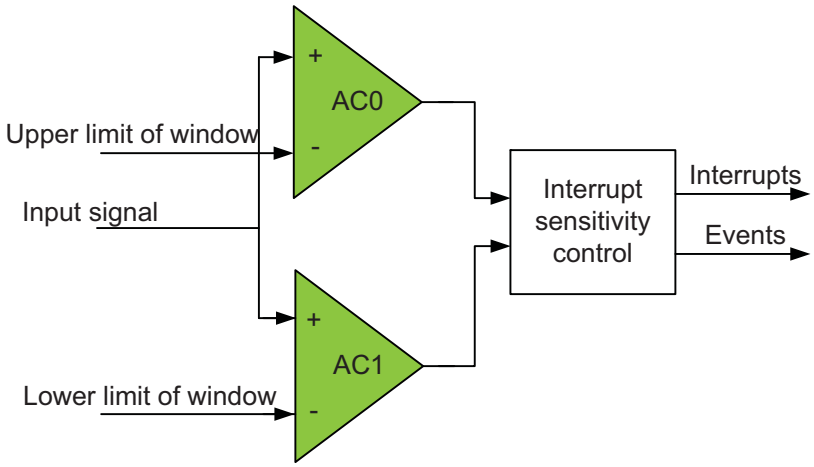
26.5 Interrupts and events

The analog comparator can be configured to generate interrupts when the output toggles, when the output changes from zero to one (rising edge), or when the output changes from one to zero (falling edge). Synchronous/ asynchronous events are generated at all times for the same condition as the interrupt, regardless of whether the interrupt is enabled or not. Each analog comparator output is source of asynchronous event.

26.6 Window mode

Two analog comparators on the same port can be configured to work together in window mode. In this mode, a voltage range is defined, and the analog comparators give information about whether an input signal is within this range or not.

Figure 26-2. Analog comparators in window mode.



26.7 Input hysteresis

Application software can select between no-, low-, and high hysteresis for the comparison. Applying a hysteresis will help prevent constant toggling of the output that can be caused by noise when the input signals are close to each other.

26.8 Register description

26.8.1 ACnCTRL – Analog Comparator n Control register

Bit	7	6	5	4	3	2	1	0
+0x00 / +0x01	INTMODE[1:0]		INTLVL[1:0]		–	HYSMODE[2:0]		ENABLE
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – INTMODE[1:0]: Interrupt Modes**
  These bits configure the interrupt mode for analog comparator n according to Table 26-1.

Table 26-1. Interrupt settings.

INTMODE[1:0]	Group configuration	Description
00	BOTHEDGES	Comparator interrupt or event on output toggle
01	–	Reserved
10	FALLING	Comparator interrupt or event on falling output edge
11	RISING	Comparator interrupt or event on rising output edge

- **Bit 5:4 – INTLVL[1:0]: Interrupt Level**
  These bits enable the analog comparator n interrupt and select the interrupt level, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132. The enabled interrupt will trigger according to the INTMODE setting.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:1 – HYSMODE[1:0]: Hysteresis Mode Select**
  These bits select the hysteresis mode according to Table 26-2. For details on actual hysteresis levels, refer to the device datasheet.

Table 26-2. Hysteresis settings.

HYSMODE[1:0]	Group configuration	Description
00	NO	No hysteresis
01	SMALL	Small hysteresis
10	LARGE	Large hysteresis
11	–	Reserved

- **Bit 0 – ENABLE: Enable**
  Setting this bit enables analog comparator n.

26.8.2 ACnMUXCTRL – Analog Comparator n Mux Control register

Bit	7	6	5	4	3	2	1	0
+0x02 / +0x03	–	–	MUXPOS[2:0]			MUXNEG[2:0]		
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:3 – MUXPOS[2:0]: Positive Input MUX Selection**
  These bits select which input will be connected to the positive input of analog comparator n according to Table 26-3.

Table 26-3. Positive input MUX selection.

MUXPOS[2:0]	Group configuration	Description
000	PIN0	Pin 0
001	PIN1	Pin 1
010	PIN2	Pin 2
011	PIN3	Pin 3
100	PIN4	Pin 4
101	PIN5	Pin 5
110	PIN6	Pin 6
111	DAC	DAC output

- **Bit 2:0 – MUXNEG[2:0]: Negative Input MUX Selection**
  These bits select which input will be connected to the negative input of analog comparator n according to Table 26-4.

Table 26-4. Negative input MUX selection.

MUXNEG[2:0]	Group configuration	Negative input MUX selection
000	PIN0	Pin 0
001	PIN1	Pin 1
010	PIN3	Pin 3
011	PIN5	Pin 5
100	PIN7	Pin 7
101	DAC	DAC output
110	BANDGAP	Internal bandgap voltage
111	SCALER	V_{CC} voltage scaler

26.8.3 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	AC1INVEN	AC0INVEN	AC1OUT	AC0OUT
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3 – AC1INVEN: Analog Comparator 1 Output Invert Enable**
  Setting this bit inverts the analog comparator 1 output. When enabled, the output to pin and event generation will be directly affected.
- **Bit 2 – AC0INVEN: Analog Comparator 0 Output Invert Enable**
  Setting this bit inverts the analog comparator 0 output. When enabled, the output to pin and event generation will be directly affected.
- **Bit 1 – AC1OUT: Analog Comparator 1 Output**
  Setting this bit makes the output of AC1 available on port pin. For details on port selection, refers to “ACEVOUT – Analog Comparator and Event Output register” on page 155. For details on available pins for each port, refer to each device datasheet.
- **Bit 0 – AC0OUT: Analog Comparator 0 Output**
  Setting this bit makes the output of AC0 available on port pin. For details on port selection, refers to “ACEVOUT – Analog Comparator and Event Output register” on page 155. For details on available pins for each port, refer to each device datasheet.

26.8.4 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	SCALEFAC[5:0]					
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 5:0 – SCALEFAC[5:0]: Voltage Scaling Factor**
  These bits define the scaling factor for the Vcc voltage scaler. The input to the analog comparator, V_{SCALE}, is:

$$V_{SCALE} = \frac{V_{CC} \cdot (SCALEFAC + 1)}{64}$$

26.8.5 WINCTRL – Window Function Control register

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	WEN	WINTMODE[1:0]		WINTLVL[1:0]	
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – WEN: Window Mode Enable**
  Setting this bit enables the analog comparator window mode.

- **Bits 3:2 – WINTMODE[1:0]: Window Interrupt Mode Settings**
  These bits configure the interrupt mode for the analog comparator window mode according to Table 26-5.

Table 26-5. Window mode interrupt settings.

WINTMODE[1:0]	Group configuration	Description
00	ABOVE	Interrupt on signal above window
01	INSIDE	Interrupt on signal inside window
10	BELOW	Interrupt on signal below window
11	OUTSIDE	Interrupt on signal outside window

- **Bits 1:0 – WINTLVL[1:0]: Window Interrupt Enable**
  These bits enable the analog comparator window mode interrupt and select the interrupt level, as described in "PMIC – Interrupts and Programmable Multilevel Interrupt Controller" on page 132. The enabled interrupt will trigger according to the WINTMODE setting.

26.8.6 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x07	WSTATE[1:0]		AC1STATE	AC0STATE	–	WIF	AC1IF	AC0IF
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bits 7:6 – WSTATE[1:0]: Window Mode Current State**
  These bits show the current state of the signal if window mode is enabled according to Table 26-6.

Table 26-6. Hysteresis settings.

WSTATE[1:0]	Group configuration	Description
00	ABOVE	Signal is above window
01	INSIDE	Signal is inside window
10	BELOW	Signal is below window
11	OUTSIDE	Signa is outside window

- **Bit 5 – AC1STATE: Analog Comparator 1 Current State**
  This bit shows the current state of the output signal from AC1.
- **Bit 4 – AC0STATE: Analog Comparator 0 Current State**
  This bit shows the current state of the output signal from AC0.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2 – WIF: Analog Comparator Window Interrupt Flag**
  This is the interrupt flag for the window mode. WIF is set according to the WINTMODE setting in the "WINCTRL – Window Function Control register" on page 388.
  This flag is automatically cleared when the analog comparator window interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **Bit 1 – AC1IF: Analog Comparator 1 Interrupt Flag**
  This is the interrupt flag for AC1. AC1IF is set according to the INTMODE setting in the corresponding “ACnCTRL – Analog Comparator n Control register” on page 386.
  This flag is automatically cleared when the analog comparator 1 interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.
- **Bit 0 – AC0IF: Analog Comparator 0 Interrupt Flag**
  This is the interrupt flag for AC0. AC0IF is set according to the INTMODE setting in the corresponding “ACnCTRL – Analog Comparator n Control register” on page 386.
  This flag is automatically cleared when the analog comparator 0 interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

26.8.7 CURRCTRL – Current Source Control register

Bit	7	6	5	4	3	2	1	0
+0x08	**CURRENT**	**CURRMODE**	–	–	–	–	**AC1CURR**	**AC0CURR**
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – CURRENT: Current Source Enable**
  Setting this bit to one will enable the constant current source.
- **Bit 6 – CURRMODE: Current Mode**
  Setting this bit to one will combine the two analog comparator current sources in order to double the output current for each analog comparator.
- **Bit 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – AC1CURR: AC1 Current Source Output Enable**
  Setting this bit to one will enable the constant current source output on the pin selected by MUXNEG in AC1MUXTRL.
- **Bit 0 – AC0CURR: AC0 Current Source Output Enable**
  Setting this bit to one will enable the constant current source output on the pin selected by MUXNEG in AC0MUXTRL.

26.8.8 CURRCALIB – Current Source Calibration register

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	**CALIB[3:0]**			
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 3:0 – CALIB[3:0]: Current Source Calibration**
  The constant current source is calibrated during production. A calibration value can be read from the signature row and written to the CURRCALIB register from software. Refer to device data sheet for default calibration values and user calibration range.

26.9 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	AC0CTRL	INTMODE[1:0]		INTLVL[1:0]		–	HYSMODE[1:0]		ENABLE	386
+0x01	AC1CTRL	INTMODE[1:0]		INTLVL[1:0]		–	HYSMODE[1:0]		ENABLE	386
+0x02	AC0MUXCTRL	–	–	MUXPOS[2:0]			MUXNEG[2:0]			387
+0x03	AC1MUXCTRL	–	–	MUXPOS[2:0]			MUXNEG[2:0]			387
+0x04	CTRLA	–	–	–	–	AC1INVEN	ACINVEN0	AC1OUT	ACOOUT	388
+0x05	CTRLB	–	–	SCALEFAC5:0]						388
+0x06	WINCTRL	–	–	–	WEN	WINTMODE[1:0]		WINTLVL[1:0]		388
+0x07	STATUS	WSTATE[1:0]		AC1STATE	AC0STATE	–	WIF	AC1IF	AC0IF	389
+0x08	CURRCTRL	CURRENT	CURRMODE	–	–	–	–	AC1CURR	AC0CURR	390
+0x09	CURRCALIB	–	–	–	–	CALIB[3:0]				390

26.10 Interrupt vector summary

Table 26-7. Analog comparator interrupt vectors.

Offset	Source	Interrupt description
0x00	COMP0_vect	Analog comparator 0 interrupt vector
0x02	COMP1_vect	Analog comparator 1 interrupt vector
0x04	WINDOW_vect	Analog comparator window interrupt vector

27. PDI – Program and Debug Interface

27.1 Features

- Programming
  - External programming through PDI interface
    - Minimal protocol overhead for fast operation
    - Built-in error detection and handling for reliable operation
  - Boot loader support for programming through any communication interface
- Debugging
  - Non-intrusive, real-time, on-chip debug system
  - No software or hardware resources required from device except pin connection
  - Program flow control
    - Go, Stop, Reset, Step Into, Step Over, Step Out, Run-to-Cursor
  - Unlimited number of user program breakpoints
  - Unlimited number of user data breakpoints, break on:
    - Data location read, write, or both read and write
    - Data location content equal or not equal to a value
    - Data location content is greater or smaller than a value
    - Data location content is within or outside a range
  - No limitation on device clock frequency
- Program and Debug Interface (PDI)
  - Two-pin interface for external programming and debugging
  - Uses the Reset pin and a dedicated pin
  - No I/O pins required during programming or debugging

27.2 Overview

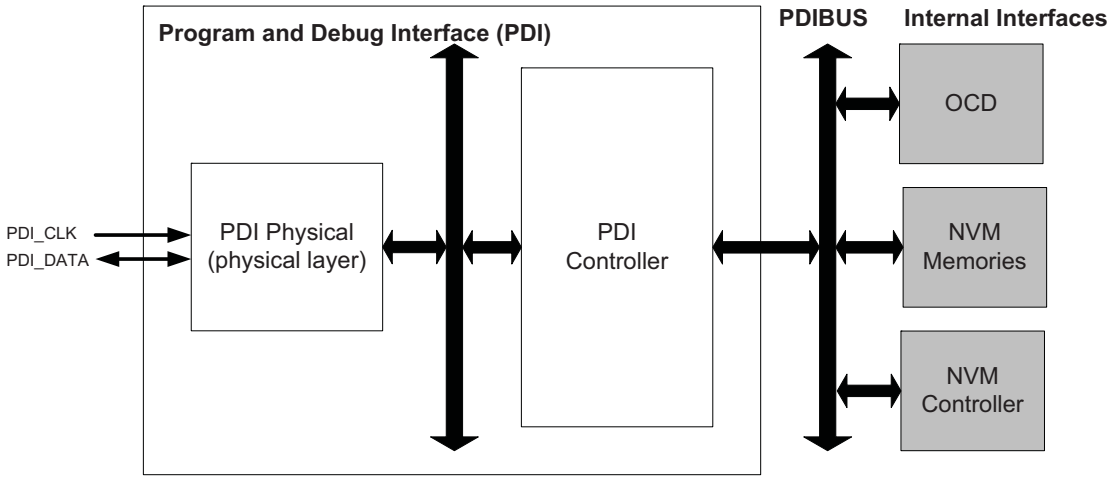
The Program and Debug Interface (PDI) is an Atmel proprietary interface for external programming and on-chip debugging of a device.

The PDI supports fast programming of nonvolatile memory (NVM) spaces; flash, EEPOM, fuses, lock bits, and the user signature row. This is done by accessing the NVM controller and executing NVM controller commands, as described in “Memory Programming” on page 403.

Debug is supported through an on-chip debug system that offers non-intrusive, real-time debug. It does not require any software or hardware resources except for the device pin connection. Using the Atmel tool chain, it offers complete program flow control and support for an unlimited number of program and complex data breakpoints. Application debug can be done from a C or other high-level language source code level, as well as from an assembler and disassembler level.

Programming and debugging can be done through two physical interfaces. The primary one is the PDI physical layer, which is available on all devices. This is a two-pin interface that uses the Reset pin for the clock input (PDI_CLK) and one other dedicated pin for data input and output (PDI_DATA). Any external programmer or on-chip debugger/emulator can be directly connected to this interface.

Figure 27-1. The PDI with PDI physical layers and closely related modules (grey).

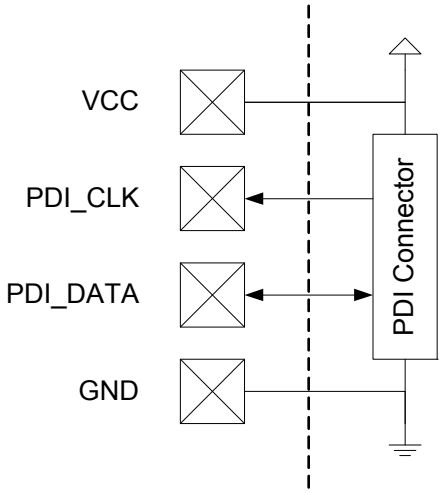


27.3 PDI physical

The PDI physical layer handles the low-level serial communication. It uses a bidirectional, half-duplex, synchronous serial receiver and transmitter (just as a USART in USRT mode). The physical layer includes start-of-frame detection, frame error detection, parity generation, parity error detection, and collision detection.

In addition to PDI_CLK and PDI_DATA, the PDI_DATA pin has an internal pull resistor, V_{CC} and GND must be connected between the External Programmer/debugger and the device. Figure 27-2 shows a typical connection.

Figure 27-2. PDI connection.



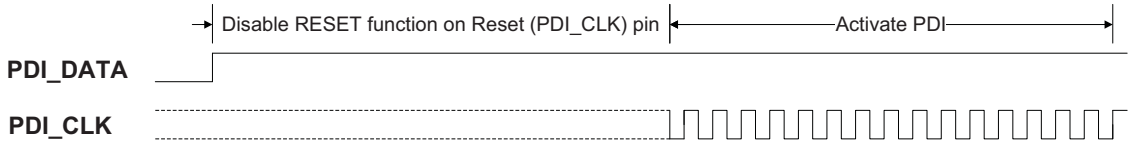
The remainder of this section is intended for use only by third parties developing programmers or programming support for Atmel AVR XMEGA devices.

27.3.1 Enabling

The PDI physical layer must be enabled before use. This is done by first forcing the PDI_DATA line high for a period longer than the equivalent external reset minimum pulse width (refer to device datasheet for external reset pulse width data). This will disable the $\overline{\text{RESET}}$ functionality of the Reset pin, if not already disabled by the fuse settings.

Next, continue to keep the PDI_DATA line high for 16 PDI_CLK cycles. The first PDI_CLK cycle must start no later than 100µs after the $\overline{\text{RESET}}$ functionality of the Reset pin is disabled. If this does not occur in time, the enabling procedure must start over again. The enable sequence is shown in Figure 27-3 on page 394.

Figure 27-3. PDI physical layer enable sequence.



The Reset pin is sampled when the PDI interface is enabled. The reset register is then set according to the state of the Reset pin, preventing the device from running code after the reset functionality of this pin is disabled.

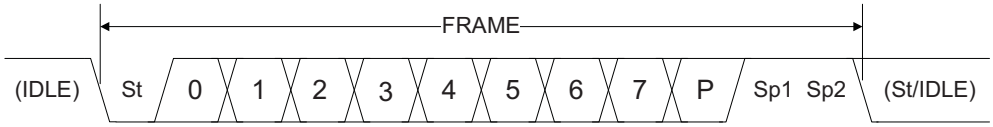
27.3.2 Disabling

If the clock frequency on PDI_CLK is lower than approximately 10kHz, this is regarded as inactivity on the clock line. This will automatically disable the PDI. If not disabled by a fuse, the reset function of the Reset (PDI_CLK) pin is enabled again. This also means that the minimum programming frequency is approximately 10kHz.

27.3.3 Frame format and characters

The PDI physical layer uses a frame format defined as one character of eight data bits, with a start bit, a parity bit, and two stop bits.

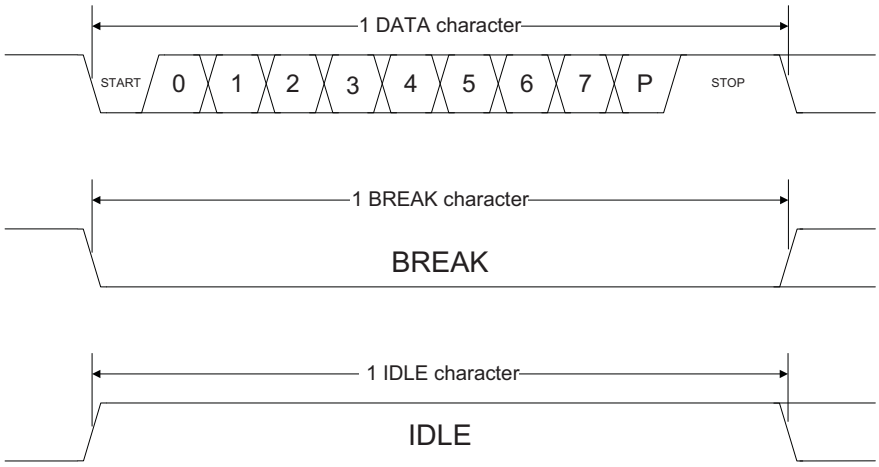
Figure 27-4. PDI serial frame format.



St	Start bit, always low
(0-7)	Data bits (0 to 7)
P	Parity bit, even parity used
Sp1	Stop bit 1, always high
Sp2	Stop bit 2, always high

Three different characters are used, DATA, BREAK, and IDLE. The BREAK character is equal to a 12-bit length of low level. The IDLE character is equal to a 12-bit length of high level. The BREAK and IDLE characters can be extended beyond the 12-bit length.

Figure 27-5. Characters and timing for the PDI physical layer.

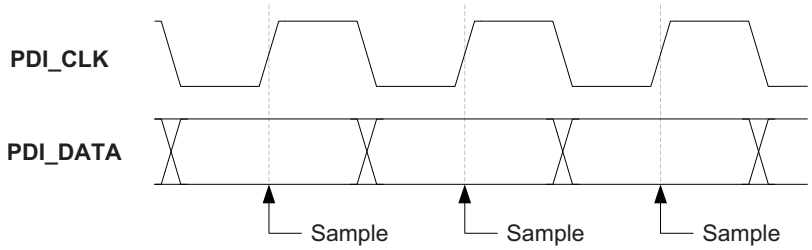


27.3.4 Serial transmission and reception

The PDI physical layer is either in transmit (TX) or receive (RX) mode. By default, it is in RX mode, waiting for a start bit.

The programmer and the PDI operate synchronously on the PDI_CLK provided by the programmer. The dependency between the clock edges and data sampling or data change is fixed. As illustrated in Figure 27-6, output data (either from the programmer or the PDI) is always set up (changed) on the falling edge of PDI_CLK and sampled on the rising edge of PDI_CLK.

Figure 27-6. Changing and sampling of data.



27.3.5 Serial transmission

When a data transmission is initiated, by the PDI controller, the transmitter simply shifts out the start bit, data bits, parity bit, and the two stop bits on the PDI_DATA line. The transmission speed is dictated by the PDI_CLK signal. While in transmission mode, IDLE bits (high bits) are automatically transmitted to fill possible gaps between successive DATA characters. If a collision is detected during transmission, the output driver is disabled, and the interface is put into RX mode waiting for a BREAK character.

27.3.6 Serial reception

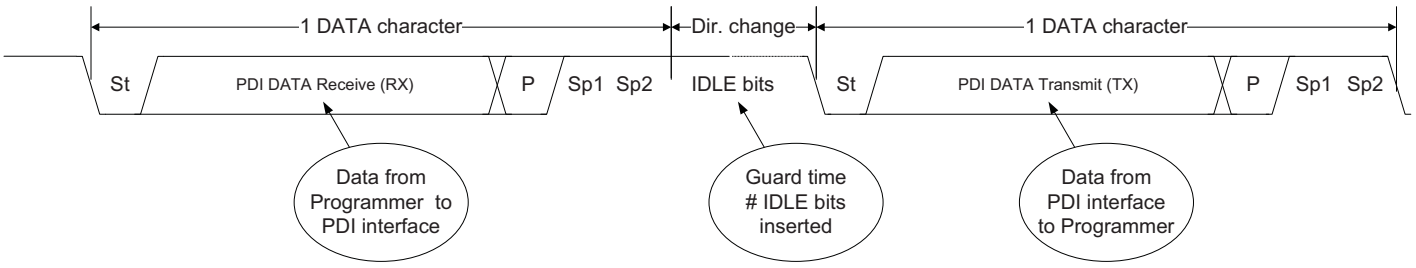
When a start bit is detected, the receiver starts to collect the eight data bits. If the parity bit does not correspond to the parity of the data bits, a parity error has occurred. If one or both of the stop bits are low, a frame error has occurred. If the parity bit is correct, and no frame error is detected, the received data bits are available for the PDI controller.

When the PDI is in TX mode, a BREAK character signaled by the programmer will not be interpreted as a BREAK, but will instead cause a generic data collision. When the PDI is in RX mode, a BREAK character will be recognized as a BREAK. By transmitting two successive BREAK characters (which must be separated by one or more high bits), the last BREAK character will always be recognized as a BREAK, regardless of whether the PDI was in TX or RX mode initially. This is because in TX mode the first BREAK is seen as a collision. The PDI then shifts to RX mode and sees the second BREAK as break.

27.3.7 Direction change

In order to ensure correct timing for half-duplex operation, a guard time mechanism is used. When the PDI changes from RX mode to TX mode, a configurable number of IDLE bits are inserted before the start bit is transmitted. The minimum transition time between RX and TX mode is two IDLE cycles, and these are always inserted. The default guard time value is 128 bits.

Figure 27-7. PDI direction change by inserting IDLE bits.

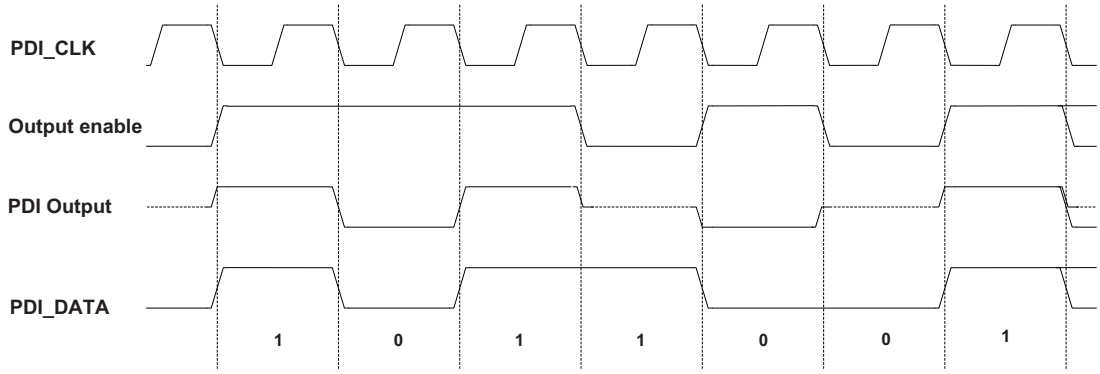


The external programmer will loose control of the PDI_DATA line at the point where the PDI changes from RX to TX mode. The guard time relaxes this critical phase of the communication. When the programmer changes from RX mode to TX mode, a single IDLE bit, at minimum, should be inserted before the start bit is transmitted.

27.3.8 Drive contention and collision detection

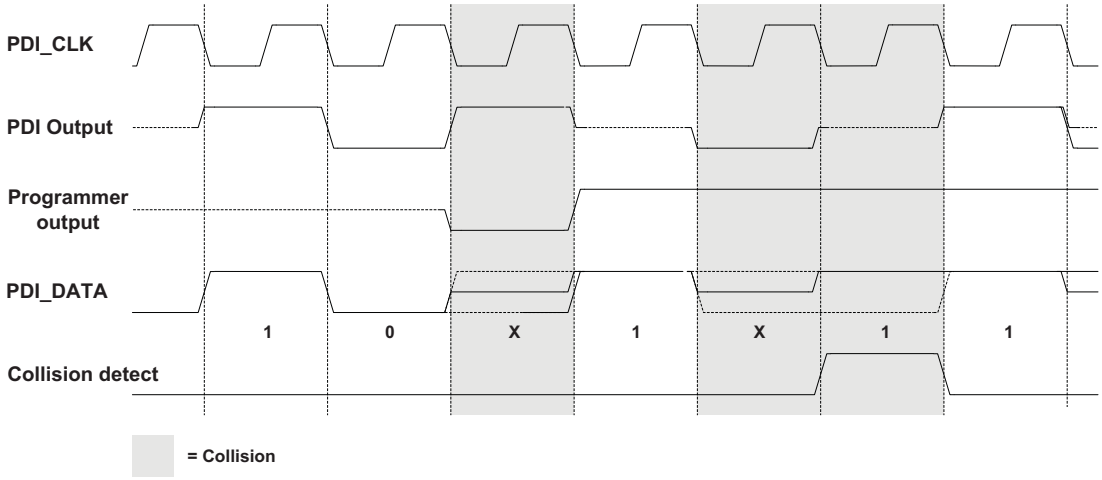
In order to reduce the effect of drive contention (the PDI and the programmer driving the PDI_DATA line at the same time), a mechanism for collision detection is used. The mechanism is based on the way the PDI drives data out on the PDI_DATA line. As shown in Figure 27-8, the PDI output driver is active only when the output value changes (from 0-1 or 1-0). Hence, if two or more successive bit values are the same, the value is actively driven only on the first clock cycle. After this point, the PDI output driver is automatically tri-stated, and the PDI_DATA pin has a bus keeper responsible for keeping the pin value unchanged until the output driver is re-enabled due to a change in the bit value.

Figure 27-8. Driving data out on the PDI_DATA using a bus keeper.



If the programmer and the PDI both drive the PDI_DATA line at the same time, drive contention will occur, as illustrated in Figure 27-9. Every time a bit value is kept for two or more clock cycles, the PDI is able to verify that the correct bit value is driven on the PDI_DATA line. If the programmer is driving the PDI_DATA line to the opposite bit value to what the PDI expects, a collision is detected.

Figure 27-9. Drive contention and collision detection on the PDI_DATA line.



As long as the PDI transmits alternating ones and zeros, collisions cannot be detected, because the PDI output driver will be active all the time, preventing polling of the PDI_DATA line. However, the two stop bits should always be transmitted as ones within a single frame, enabling collision detection at least once per frame.

27.4 PDI controller

The PDI controller performs data transmission/reception on a byte level, command decoding, high-level direction control, control and status register access, exception handling, and clock switching (PDI_CLK or TCK). The interaction between an external programmer and the PDI controller is based on a scheme where the programmer transmits various types of requests to the PDI controller, which in turn responds according to the specific request. A programmer request comes in the form of an instruction, which may be followed by one or more byte operands. The PDI controller response may be silent (e.g., a data byte is stored to a location within the device), or it may involve data being returned to the programmer (e.g., a data byte is read from a location within the device).

27.4.1 Accessing internal interfaces

After an external programmer has established communication with the PDI, the internal interfaces are not accessible, by default. To get access to the NVM controller and the nonvolatile memories for programming, a unique key must be signaled by using the KEY instruction. The internal interfaces are accessed as one linear address space using a dedicated bus (PDIBUS) between the PDI and the internal interfaces. The PDIBUS address space is shown in Figure 28-3 on page 417. The NVM controller must be enabled for the PDI controller to have any access to the NVM interface. The PDI controller can access the NVM and NVM controller in programming mode only. The PDI controller does not need to access the NVM controller's data or address registers when reading or writing NVM.

27.4.2 NVM programming key

The key that must be sent using the KEY instruction is 64 bits long. The key that will enable NVM programming is:

0x1289AB45CDD888FF.

27.4.3 Exception handling

There are several situations that are considered exceptions from normal operation. The exceptions depend on whether the PDI is in RX or TX mode.

While the PDI is in RX mode, the exceptions are:

- The physical layer detects a parity error
- The physical layer detects a frame error
- The physical layer recognizes a BREAK character (also detected as a frame error)

While the PDI is in TX mode, the exception is:

- The physical layer detects a data collision

Exceptions are signaled to the PDI controller. All ongoing operations are then aborted, and the PDI is put in ERROR state. The PDI will remain in ERROR state until a BREAK is sent from the external programmer, and this will bring the PDI back to its default RX state.

Due to this mechanism, the programmer can always synchronize the protocol by transmitting two successive BREAK characters.

27.4.4 Reset signalling

Through the reset register, the programmer can issue a reset and force the device into reset. After clearing the reset register, reset is released, unless some other reset source is active.

27.4.5 Instruction set

The PDI has a small instruction set used for accessing both the PDI itself and the internal interfaces. All instructions are byte instructions. The instructions allow an external programmer to access the PDI controller, the NVM controller and the nonvolatile memories.

27.4.5.1 LDS - Load data from PDIBUS data space using direct addressing

The LDS instruction is used to load data from the PDIBUS data space for read out. The LDS instruction is based on direct addressing, which means that the address must be given as an argument to the instruction. Even though the protocol is based on byte-wise communication, the LDS instruction supports multiple-byte addresses and data access. Four different address/data sizes are supported: single-byte, word (two bytes), three-byte, and long (four bytes). Multiple-byte access is broken down internally into repeated single-byte accesses, but this reduces protocol overhead. When using the LDS instruction, the address byte(s) must be transmitted before the data transfer.

27.4.5.2 STS - Store data to PDIBUS data space using direct addressing

The STS instruction is used to store data that are serially shifted into the physical layer shift register to locations within the PDIBUS data space. The STS instruction is based on direct addressing, which means that the address must be given as an argument to the instruction. Even though the protocol is based on byte-wise communication, the ST instruction supports multiple-bytes addresses and data access. Four different address/data sizes are supported: single-byte, word (two bytes), three-byte, and long (four bytes). Multiple-byte access is broken down internally into repeated single-byte accesses, but this reduces protocol overhead. When using the STS instruction, the address byte(s) must be transmitted before the data transfer.

27.4.5.3 LD - Load data from PDIBUS data space using indirect addressing

The LD instruction is used to load data from the PDIBUS data space into the physical layer shift register for serial read out. The LD instruction is based on indirect addressing (pointer access), which means that the address must be stored in the pointer register prior to the data access. Indirect addressing can be combined with pointer increment. In addition to reading data from the PDIBUS data space, the LD instruction can read the pointer register. Even though the protocol is based on byte-wise communication, the LD instruction supports multiple-byte addresses and data access. Four different address/data sizes are supported: single-byte, word (two bytes), three-byte, and long (four bytes). Multiple-byte access is broken down internally into repeated single-byte accesses, but this reduces the protocol overhead.

27.4.5.4 ST - Store data to PDIBUS data space using indirect addressing

The ST instruction is used to store data that is serially shifted into the physical layer shift register to locations within the PDIBUS data space. The ST instruction is based on indirect addressing (pointer access), which means that the address must be stored in the pointer register prior to the data access. Indirect addressing can be combined with pointer increment. In addition to writing data to the PDIBUS data space, the ST instruction can write the pointer register. Even though the protocol is based on byte-wise communication, the ST instruction supports multiple-bytes address - and data access. Four different address/data sizes are supported; byte, word, 3 bytes, and long (4 bytes). Multiple-bytes access is internally broken down to repeated single-byte accesses, but it reduces the protocol overhead.

27.4.5.5 LDCS - Load data from PDI control and status register space

The LDCS instruction is used to load data from the PDI control and status registers into the physical layer shift register for serial read out. The LDCS instruction supports only direct addressing and single-byte access.

27.4.5.6 STCS - Store data to PDI control and status register space

The STCS instruction is used to store data that are serially shifted into the physical layer shift register to locations within the PDI control and status registers. The STCS instruction supports only direct addressing and single-byte access.

27.4.5.7 KEY - Set activation key

The KEY instruction is used to communicate the activation key bytes required for activating the NVM interfaces.

27.4.5.8 REPEAT - Set instruction repeat counter

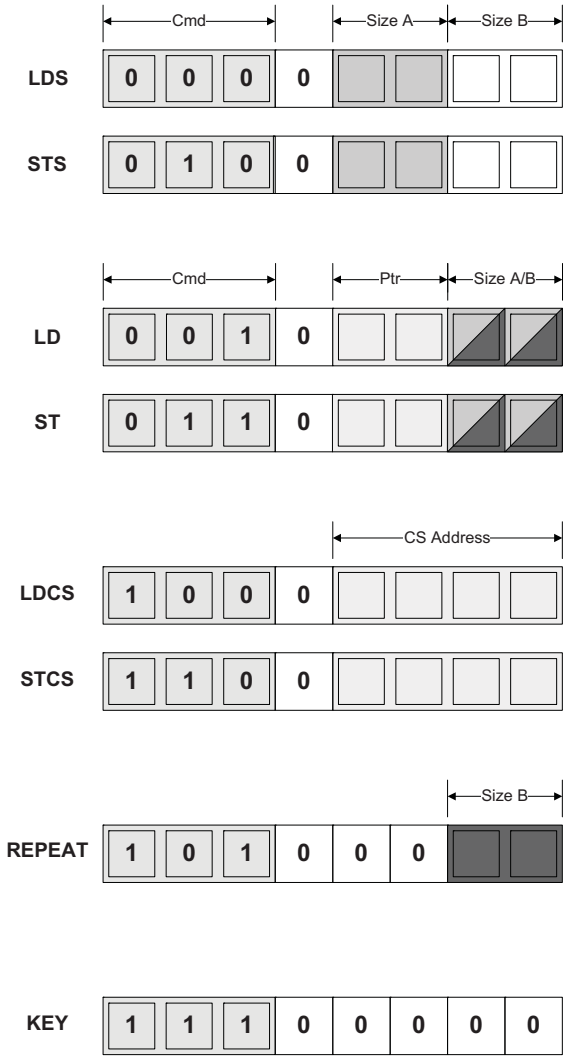
The REPEAT instruction is used to store count values that are serially shifted into the physical layer shift register to the repeat counter register. The instruction that is loaded directly after the REPEAT instruction operand(s) will be repeated a number of times according to the specified repeat counter register value. Hence, the initial repeat counter value plus one gives the total number of times the instruction will be executed. Setting the repeat counter register to zero makes the following instruction run once without being repeated.

The REPEAT instruction cannot be repeated. The KEY instruction cannot be repeated, and will override the current value of the repeat counter register.

27.4.6 Instruction set summary

The PDI instruction set summary is shown in Figure 27-10.

Figure 27-10.PDI instruction set summary.



Cmd			
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 (direct access)		
0	0	Byte
0	1	Word (2 Bytes)
1	0	3 Bytes
1	1	Long (4 Bytes)

Ptr - Pointer access (indirect access)		
0	0	*(ptr)
0	1	*(ptr++)
1	0	ptr
1	1	ptr++ - Reserved

Size B - Data size		
0	0	Byte
0	1	Word (2 Bytes)
1	0	3 Bytes
1	1	Long (4 Bytes)

CS Address (CS - Control/Status reg.)				
0	0	0	0	Register 0
0	0	0	1	Register 1
0	0	1	0	Register 2
0	0	1	1	Reserved
.....				
1	1	1	1	Reserved

27.5 Register description – PDI instruction and addressing registers

The PDI instruction and addressing registers are internal registers utilized for instruction decoding and PDIBUS addressing. None of these registers are accessible as registers in a register space.

27.5.1 Instruction register

When an instruction is successfully shifted into the physical layer shift register, it is copied into the instruction register. The instruction is retained until another instruction is loaded. The reason for this is that the REPEAT command may force the same instruction to be run repeatedly, requiring command decoding to be performed several times on the same instruction.

27.5.2 Pointer register

The pointer register is used to store an address value that specifies locations within the PDIBUS address space. During direct data access, the pointer register is updated by the specified number of address bytes given as operand bytes to an instruction. During indirect data access, addressing is based on an address already stored in the pointer register prior to the access itself. Indirect data access can be optionally combined with pointer register post-increment. The indirect access mode has an option that makes it possible to load or read the pointer register without accessing any other registers. Any register update is performed in a little-endian fashion. Hence, loading a single byte of the address register will always update the LSB while the most-significant bytes are left unchanged.

The pointer register is not involved in addressing registers in the PDI control and status register space (CSRS space).

27.5.3 Repeat counter register

The REPEAT instruction is always accompanied by one or more operand bytes that define the number of times the next instruction should be repeated. These operand bytes are copied into the repeat counter register upon reception. During the repeated executions of the instruction immediately following the REPEAT instruction and its operands, the repeat counter register is decremented until it reaches zero, indicating that all repetitions have completed. The repeat counter is also involved in key reception.

27.5.4 Operand count register

Immediately after an instruction (except the LDCS and STCS instructions) a specified number of operands or data bytes (given by the size parts of the instruction) are expected. The operand count register is used to keep track of how many bytes have been transferred.

27.6 Register description – PDI control and status registers

The PDI control and status registers are accessible in the PDI control and status register space (CSRS) using the LDCS and STCS instructions. The CSRS contains registers directly involved in configuration and status monitoring of the PDI itself.

27.6.1 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	–	NVMEN	–
Read/Write	R	R	R	R	R	R	R/W	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – NVMEN: Nonvolatile Memory Enable**
  This status bit is set when the key signalling enables the NVM programming interface. The external programmer can poll this bit to verify successful enabling. Writing the NVMEN bit disables the NVM interface.
- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

27.6.2 RESET – Reset register

Bit	7	6	5	4	3	2	1	0
+0x01	RESET[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – RESET[7:0]: Reset Signature**
  When the reset signature, 0x59, is written to RESET, the device is forced into reset. The device is kept in reset until RESET is written with a data value different from the reset signature. Reading the lsb will return the status of the reset. The seven msbs will always return the value 0x00, regardless of whether the device is in reset or not.

27.6.3 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	GUARDTIME[2:0]		
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2:0 – GUARDTIME[2:0]: Guard Time**
  These bits specify the number of IDLE bits of guard time that are inserted in between PDI reception and transmission direction changes. The default guard time is 128 IDLE bits, and the available settings are shown in Table 27-1. In order to speed up the communication, the guard time should be set to the lowest safe configuration accepted. No guard time is inserted when switching from TX to RX mode.

Table 27-1. Guard time settings.

GUARDTIME	Number of IDLE bits
000	128
001	64
010	32
011	16
100	8
101	4
110	2
111	2

27.7 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	–	–	–	–	–	–	NVMEN	–	401
+0x01	RESET	RESET[7:0]								401
+0x02	CTRL	–	–	–	–	–	GUARDTIME[2:0]			401
+0x03	Reserved	–	–	–	–	–	–	–	–	

28. Memory Programming

28.1 Features

- Read and write access to all memory spaces from
  - External programmers
  - Application software self-programming
- Self-programming and boot loader support
  - Any communication interface can be used for program upload/download
- External programming
  - Support for in-system and production programming
  - Programming through serial PDI interface
- High security with separate boot lock bits for:
  - External programming access
  - Boot loader section access
  - Application section access
  - Application table access
- Reset fuse to select reset vector address to the start of the:
  - Application section, or
  - Boot loader section

28.2 Overview

This section describes how to program the nonvolatile memory (NVM) in Atmel AVR XMEGA devices, and covers both self-programming and external programming. The NVM consists of the flash program memory, user signature and production signature rows, fuses and lock bits, and EEPROM data memory. For details on the actual memories, how they are organized, and the register description for the NVM controller used to access the memories, refer to “Memories” on page 20.

The NVM can be accessed for read and write from application software through self-programming and from an external programmer. Accessing the NVM is done through the NVM controller, and for the flash memory the two methods of programming are similar. Memory access is done by loading address and/or data to the selected memory or NVM controller and using a set of commands and triggers that make the NVM controller perform specific tasks on the nonvolatile memory.

From external programming, all memory spaces can be read and written, except for the production signature row, which can only be read. The device can be programmed in-system and is accessed through the PDI using the PDI physical interface. “External programming” on page 416 describes PDI in detail.

Self-programming and boot loader support allows application software in the device to read and write the flash, user signature row and EEPROM, write the lock bits to a more secure setting, and read the production signature row and fuses. When programming, the CPU is halted, waiting for the flash operation to complete. “Self-programming and boot loader support” on page 407 describes this in detail.

For both self-programming and external programming, it is possible to run a CRC check on the flash or a section of the flash to verify its content after programming.

The device can be locked to prevent reading and/or writing of the NVM. There are separate lock bits for external programming access and self-programming access to the boot loader section, application section, and application table section.

28.3 NVM controller

Access to the nonvolatile memories is done through the NVM controller. It controls NVM timing and access privileges, and holds the status of the NVM, and is the common NVM interface for both external programming and self-programming. For more details, refer to “Register description” on page 421.

28.4 NVM commands

The NVM controller has a set of commands used to perform tasks on the NVM. This is done by writing the selected command to the NVM command register. In addition, data and addresses must be read/written from/to the NVM data and address registers for memory read/write operations.

When a selected command is loaded and address and data are set up for the operation, each command has a trigger that will start the operation. Based on these triggers, there are three main types of commands.

28.4.1 Action-triggered commands

Action-triggered commands are triggered when the command execute (CMDEX) bit in the NVM control register A (CTRLA) is written. Action-triggered commands typically are used for operations which do not read or write the NVM, such as the CRC check.

28.4.2 NVM read-triggered commands

NVM read-triggered commands are triggered when the NVM is read, and this is typically used for NVM read operations.

28.4.3 NVM write-triggered commands

NVM write-triggered commands are triggered when the NVM is written, and this is typically used for NVM write operations.

28.4.4 Write/execute protection

Most command triggers are protected from accidental modification/execution during self-programming. This is done using the configuration change protection (CCP) feature, which requires a special write or execute sequence in order to change a bit or execute an instruction. For details on the CCP, refer to “Configuration change protection” on page 13.

28.5 NVM controller busy status

When the NVM controller is busy performing an operation, the busy flag in the NVM status register is set and the following registers are blocked for write access:

- NVM command register
- NVM control A register
- NVM control B register
- NVM address registers
- NVM data registers

This ensures that the given command is executed and the operations finished before the start of a new operation. The external programmer or application software must ensure that the NVM is not addressed when it is busy with a programming operation.

Programming any part of the NVM will automatically block:

- All programming to other parts of the NVM
- All loading/erasing of the flash and EEPROM page buffers
- All NVM reads from external programmers
- All NVM reads from the application section

During self-programming, interrupts must be disabled or the interrupt vector table must be moved to the boot loader sections, as described in “PMIC – Interrupts and Programmable Multilevel Interrupt Controller” on page 132.

28.6 Flash and EEPROM page buffers

The flash memory is updated page by page. The EEPROM can be updated on a byte-by-byte and page-by-page basis. flash and EEPROM page programming is done by first filling the associated page buffer, and then writing the entire page buffer to a selected page in flash or EEPROM.

The size of the page and page buffers depends on the flash and EEPROM size in each device, and details are described in the device's datasheet.

28.6.1 Flash page buffer

The flash page buffer is filled one word at a time, and it must be erased before it can be loaded. When loading the page buffer with new content, the result is a binary AND between the existing content of the page buffer location and the new value. If the page buffer is already loaded once after erase the location will most likely be corrupted.

Page buffer locations that are not loaded will have the value 0xFFFF, and this value will then be programmed into the corresponding flash page locations.

The page buffer is automatically erased after:

- A device reset
- Executing the write flash page command
- Executing the erase and write flash page command
- Executing the signature row write command
- Executing the write lock bit command

28.6.2 EEPROM page buffer

The EEPROM page buffer is filled one byte at a time, and it must be erased before it can be loaded. When loading the page buffer with new content, the result is a binary AND between the existing content of the page buffer location and the new value. If the EEPROM page buffer is already loaded once after erase the location will most likely be corrupted.

EEPROM page buffer locations that are loaded will get tagged by the NVM controller. During a page write or page erase, only targeted locations will be written or erased. Locations that are not targeted will not be written or erased, and the corresponding EEPROM location will remain unchanged. This means that before an EEPROM page erase, data must be loaded to the selected page buffer location to tag them. When performing an EEPROM page erase, the actual value of the tagged location does not matter.

The EEPROM page buffer is automatically erased after:

- A system reset
- Executing the write EEPROM page command
- Executing the erase and write EEPROM page command
- Executing the write lock bit and write fuse commands.

28.7 Flash and EEPROM programming sequences

For page programming, filling the page buffers and writing the page buffer into flash or EEPROM are two separate operations. The sequence is same for both self-programming and external programming.

28.7.1 Flash programming sequence

Before programming a flash page with the data in the flash page buffer, the flash page must be erased. Programming an un-erased flash page will corrupt its content.

The flash page buffer can be filled either before the erase flash Page operation or between a erase flash page and a write flash page operation:

Alternative 1:

- Fill the flash page buffer
- Perform a flash page erase
- Perform a flash page write

Alternative 2:

- Fill the flash page buffer
- Perform an atomic page erase and write

Alternative 3, fill the buffer after a page erase:

- Perform a flash page erase
- Fill the flash page buffer
- Perform a flash page write

The NVM command set supports both atomic erase and write operations, and split page erase and page write commands. This split commands enable shorter programming time for each command, and the erase operations can be done during non-time-critical programming execution. If alternative 3 is used, it is not possible to read the old data while loading, since the page is already erased. The page address must be the same for both page erase and page write operations when using alternative 1 or 3.

28.7.2 EEPROM programming sequence

Before programming an EEPROM page with the tagged data bytes stored in the EEPROM page buffer, the selected locations in the EEPROM page must be erased. Programming an un-erased EEPROM page will corrupt its content. The EEPROM page buffer must be loaded before any page erase or page write operations:

Alternative 1:

- Fill the EEPROM page buffer with the selected number of bytes
- Perform a EEPROM page erase
- Perform a EEPROM page write

Alternative 2:

- Fill the EEPROM page buffer with the selected number of bytes
- Perform an atomic EEPROM page erase and write

28.8 Protection of NVM

To protect the flash and EEPROM memories from write and/or read, lock bits can be set to restrict access from external programmers and the application software. Refer to “LOCKBITS – Lock Bit register” on page 29 for details on the available lock bit settings and how to use them.

28.9 Preventing NVM corruption

During periods when the V_{CC} voltage is below the minimum operating voltage for the device, the result from a flash memory write can be corrupt, as supply voltage is too low for the CPU and the flash to operate properly.To ensure that the voltage is sufficient enough during a complete programming sequence of the flash memory, a voltage detector using the POR threshold (V_{POT+}) level is enabled. During chip erase and when the PDI is enabled the brownout detector (BOD) is automatically enabled at its configured level.

Depending on the programming operation, if any of these V_{CC} voltage levels are reached, the programming sequence will be aborted immediately. If this happens, the NVM programming should be restarted when the power is sufficient again, in case the write sequence failed or only partly succeeded.

28.10 CRC functionality

It is possible to run an automatic cyclic redundancy check (CRC) on the flash program memory. When NVM is used to control the CRC module, an even number of bytes are read, at least in the flash range mode. If the user selects a range with an odd number of bytes, an extra byte will be read, and the checksum will not correspond to the selected range.

Refer to “CRC – Cyclic Redundancy Check generator” on page 336 for more details.

28.11 Self-programming and boot loader support

Reading and writing the EEPROM and flash memory from the application software in the device is referred to as self-programming. A boot loader (application code located in the boot loader section of the flash) can both read and write the flash program memory, user signature row, and EEPROM, and write the lock bits to a more secure setting. Application code in the application section can read from the flash, user signature row, production signature row, and fuses, and read and write the EEPROM.

28.11.1 Flash programming

The boot loader can use any available communication interface and associated protocol to read code and write (program) that code into the flash memory, or read out the program memory code. It has the capability to write into the entire flash, including the boot loader section. The boot loader can thus modify itself, and it can also erase itself from the flash if the feature is not needed anymore.

28.11.1.1 Application and boot loader sections

The application and boot loader sections in the flash are different when it comes to self-programming.

- When erasing or writing a page located inside the application section, the boot loader section can be read during the operation, but the CPU is halted during the entire operation, and cannot execute from the boot loader section
- When erasing or writing a page located inside the boot loader section, the CPU is halted during the entire operation, and code cannot execute

The user signature row section has the same properties as the boot loader section.

Table 28-1. Summary of self-programming functionality.

Section being addressed during programming	Section that can be read during programming	CPU halted?
Application section	Boot loader section	Yes
Boot loader section	None	Yes
User signature row section	None	Yes

28.11.1.2 Addressing the flash

The Z-pointer is used to hold the flash memory address for read and write access. For more details on the Z-pointer, refer to “The X-, Y-, and Z- registers” on page 11.

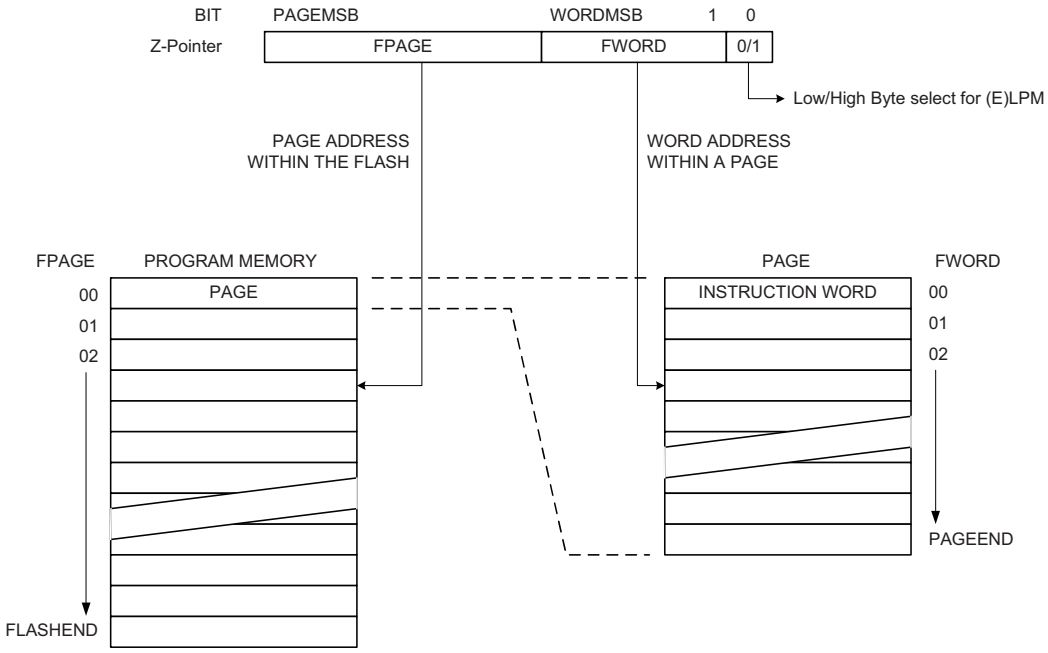
Since the flash is word accessed and organized in pages, the Z-pointer can be treated as having two sections. The least-significant bits address the words within a page, while the most-significant bits address the page within the flash. This is shown in Figure 28-1 on page 408. The word address in the page (FWORD) is held by the bits [WORDMSB:1] in the Z-pointer. The remaining bits [PAGEMSB:WORDMSB+1] in the Z-pointer hold the flash page address (FPAGE). Together FWORD and FPAGE holds an absolute address to a word in the flash.

For flash read operations (ELPM and LPM), one byte is read at a time. For this, the least-significant bit (bit 0) in the Z-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.

The size of FWORD and FPAGE will depend on the page and flash size in the device. Refer to each device’s datasheet for details.

Once a programming operation is initiated, the address is latched and the Z-pointer can be updated and used for other operations.

Figure 28-1. Flash addressing for self-programming.



28.11.2 NVM flash commands

The NVM commands that can be used for accessing the flash program memory, signature row and production signature row are listed in Table 28-2 on page 409.

For self-programming of the flash, the trigger for action-triggered commands is to set the CMDEX bit in the NVM CTRLA register (CMDEX). The read-triggered commands are triggered by executing the (E)LPM instruction (LPM). The write-triggered commands are triggered by executing the SPM instruction (SPM).

The Change Protected column indicates whether the trigger is protected by the configuration change protection (CCP) or not. This is a special sequence to write/execute the trigger during self-programming. For more details, refer to "CCP – Configuration Change Protection register" on page 15. CCP is not required for external programming. The two last columns show the address pointer used for addressing and the source/destination data register.

" Application and boot loader sections" on page 407 through " Read user signature row / production signature row" on page 412 explain in detail the algorithm for each NVM operation.

Table 28-2. Flash self-programming commands.

CMD[6:0]	Group configuration	Description	Trigger	CPU halted	NVM busy	Change	Address pointer	Data register
0x00	NO_OPERATION	No operation / read flash	-/(E)LPM	-/N	N	-/N	-/ Z-pointer	-/Rd
Flash Page Buffer								
0x23	LOAD_FLASH_BUFFER	Load flash page buffer	SPM	N	N	N	Z-pointer	R1:R0
0x26	ERASE_FLASH_BUFFER	Erase flash page buffer	CMDEX	N	Y	Y	Z-pointer	-
Flash								
0x2B	ERASE_FLASH_PAGE	Erase flash page	SPM	N/Y(2)	Y	Y	Z-pointer	-
0x02E	WRITE_FLASH_PAGE	Write flash page	SPM	N/Y(2)	Y	Y	Z-pointer	-
0x2F	ERASE_WRITE_FLASH_PAGE	Erase and write flash page	SPM	N/Y(2)	Y	Y	Z-pointer	-
0x3A	FLASH_RANGE_CRC (3)	Flash range CRC	CMDEX	Y	Y	Y	DATA/ADDR(1)	DATA
Application Section								
0x20	ERASE_APP	Erase application section	SPM	Y	Y	Y	Z-pointer	-
0x22	ERASE_APP_PAGE	Erase application section page	SPM	N	Y	Y	Z-pointer	-
0x24	WRITE_APP_PAGE	Write application section page	SPM	N	Y	Y	Z-pointer	-
0x25	ERASE_WRITE_APP_PAGE	Erase and write application section page	SPM	N	Y	Y	Z-pointer	-
0x38	APP_CRC	Application section CRC	CMDEX	Y	Y	Y	-	DATA
Boot Loader Section								
0x2C	WRITE_BOOT_PAGE	Write boot loader section page	SPM	Y	Y	Y	Z-pointer	-
0x2D	ERASE_WRITE_BOOT_PAGE	Erase and write boot loader section page	SPM	Y	Y	Y	Z-pointer	-
0x39	BOOT_CRC	Boot loader section CRC	CMDEX	Y	Y	Y	-	DATA
User Signature Row								
0x01(4)	READ_USER_SIG_ROW	Read user signature row	LPM	N	N	N	Z-pointer	Rd
0x18	ERASE_USER_SIG_ROW	Erase user signature row	SPM	Y	Y	Y	-	-
0x1A	WRITE_USER_SIG_ROW	Write user signature row	SPM	Y	Y	Y	-	-
Production Signature (Calibration) Row(5)								
0x02(4)	READ_CALIB_ROW	Read calibration row	LPM	N	N	N	Z-pointer	Rd

Notes:
1. The flash range CRC command used byte addressing of the flash.
2. Will depend on the flash section (application or boot loader) that is actually addressed.
3. This command is qualified with the lock bits, and requires that the boot lock bits are unprogrammed.
4. When using a command that changes the normal behavior of the LPM command; READ_USER_SIG_ROW and READ_CALIB_ROW; it is recommended to disable interrupts to ensure correct execution of the LPM instruction.
5. For consistency the name Calibration Row has been renamed to Production Signature Row throughout the document.

28.11.2.1 Read flash

The (E)LPM instruction is used to read one byte from the flash memory.

1. Load the Z-pointer with the byte address to read.
2. Load the NVM command register (NVM CMD) with the no operation command.
3. Execute the LPM instruction.

The destination register will be loaded during the execution of the LPM instruction.

28.11.2.2 Erase flash page buffer

The erase flash page buffer command is used to erase the flash page buffer.

1. Load the NVM CMD with the erase flash page buffer command.
2. Set the command execute bit (NVMEX) in the NVM control register A (NVM CTRLA). This requires the timed CCP sequence during self-programming.

The NVM busy (BUSY) flag in the NVM status register (NVM STATUS) will be set until the page buffer is erased.

28.11.2.3 Load flash page buffer

The load flash page buffer command is used to load one word of data into the flash page buffer.

1. Load the NVM CMD register with the load flash page buffer command.
2. Load the Z-pointer with the word address to write.
3. Load the data word to be written into the R1:R0 registers.
4. Execute the SPM instruction. The SPM instruction is not protected when performing a flash page buffer load.

Repeat step 2-4 until the complete flash page buffer is loaded. Unloaded locations will have the value 0xFFFF.

28.11.2.4 Erase flash page

The erase flash page command is used to erase one page in the flash.

1. Load the Z-pointer with the flash page address to erase. The page address must be written to FPAGE. Other bits in the Z-pointer will be ignored during this operation.
2. Load the NVM CMD register with the erase flash page command.
3. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the erase operation is finished. The flash section busy (FBUSY) flag is set as long the flash is busy, and the application section cannot be accessed.

28.11.2.5 Write flash page

The write flash page command is used to write the flash page buffer into one flash page in the flash.

1. Load the Z-pointer with the flash page to write. The page address must be written to FPAGE. Other bits in the Z-pointer will be ignored during this operation.
2. Load the NVM CMD register with the write flash page command.
3. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the write operation is finished. The FBUSY flag is set as long the flash is busy, and the application section cannot be accessed.

28.11.2.6 Flash range CRC

The flash range CRC command can be used to verify the content in an address range in flash after a self-programming.

1. Load the NVM CMD register with the flash range CRC command.
2. Load the start byte address in the NVM address register (NVM ADDR).
3. Load the end byte address in NVM data register (NVM DATA).
4. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set, and the CPU is halted during the execution of the command.

The CRC checksum will be available in the NVM DATA register.

In order to use the flash range CRC command, all the boot lock bits must be unprogrammed (no locks). The command execution will be aborted if the boot lock bits for an accessed location are set.

28.11.2.7 Erase application section / boot loader section page

The erase application section page erase and erase boot loader section page commands are used to erase one page in the application section or boot loader section.

1. Load the Z-pointer with the flash page address to erase. The page address must be written to ZPAGE. Other bits in the Z-pointer will be ignored during this operation.
2. Load the NVM CMD register with the erase application/boot section page command.
3. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the erase operation is finished. The FBUSY flag is set as long the flash is busy, and the application section cannot be accessed.

28.11.2.8 Application section / boot loader section page write

The write application section page and write boot loader section page commands are used to write the flash page buffer into one flash page in the application section or boot loader section.

1. Load the Z-pointer with the flash page to write. The page address must be written to FPAGE. Other bits in the Z-pointer will be ignored during this operation.
2. Load the NVM CMD register with the write application section/boot loader section page command.
3. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the write operation is finished. The FBUSY flag is set as long the flash is busy, and the application section cannot be accessed.

An invalid page address in the Z-pointer will abort the NVM command. The erase application section page command requires that the Z-pointer addresses the application section, and the erase boot section page command requires that the Z-pointer addresses the boot loader section.

28.11.2.9 Erase and write application section / boot loader section page

The erase and write application section page and erase and write boot loader section page commands are used to erase one flash page and then write the flash page buffer into that flash page in the application section or boot loader section in one atomic operation.

1. Load the Z-pointer with the flash page to write. The page address must be written to FPAGE. Other bits in the Z-pointer will be ignored during this operation.
2. Load the NVM CMD register with the erase and write application section/boot loader section page command.
3. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished. The FBUSY flag is set as long as the flash is busy, and the application section cannot be accessed.

An invalid page address in the Z-pointer will abort the NVM command. The erase and write application section command requires that the Z-pointer addresses the application section, and the erase and write boot section page command requires that the Z-pointer addresses the boot loader section.

28.11.2.10 Application section / boot loader section CRC

The application section CRC and boot loader section CRC commands can be used to verify the application section and boot loader section content after self-programming.

1. Load the NVM CMD register with the application section/ boot load section CRC command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set, and the CPU is halted during the execution of the CRC command. The CRC checksum will be available in the NVM data registers.

28.11.2.11 Erase user signature row

The erase user signature row command is used to erase the user signature row.

1. Load the NVM CMD register with the erase user signature row command.
2. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set, and the CPU will be halted until the erase operation is finished. The user signature row is NRWW.

28.11.2.12 Write user signature row

The write signature row command is used to write the flash page buffer into the user signature row.

1. Set up the NVM CMD register to write user signature row command.
2. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished, and the CPU will be halted during the write operation. The flash page buffer will be cleared during the command execution after the write operation, but the CPU is not halted during this stage.

28.11.2.13 Read user signature row / production signature row

The read user signature row and read production signature (calibration) row commands are used to read one byte from the user signature row or production signature (calibration) row.

1. Load the Z-pointer with the byte address to read.
2. Load the NVM CMD register with the read user signature row / production signature (calibration) row command.
3. Execute the LPM instruction.

The destination register will be loaded during the execution of the LPM instruction.

To ensure that LPM for reading flash will be executed correctly it is advised to disable interrupt while using either of these commands.

28.11.3 NVM fuse and lock bit commands

The NVM flash commands that can be used for accessing the fuses and lock bits are listed in Table 28-3 on page 412.

For self-programming of the fuses and lock bits, the trigger for action-triggered commands is to set the CMDEX bit in the NVM CTRLA register (CMDEX). The read-triggered commands are triggered by executing the (E)LPM instruction (LPM). The write-triggered commands are triggered by a executing the SPM instruction (SPM).

The Change Protected column indicates whether the trigger is protected by the configuration change protection (CCP) during self-programming or not. The last two columns show the address pointer used for addressing and the source/destination data register.

Section 28.11.3.1 “ Write lock bits” on page 413 through Section 28.11.3.2 “ Read fuses” on page 413 explain in detail the algorithm for each NVM operation.

Table 28-3. Fuse and lock bit commands.

CMD[6:0]	Group configuration	Description	Trigger	CPU halted	Change protected	NVM busy	Address pointer	Data register
0x00	NO_OPERATION	No operation	-	-	-	-	-	-
Fuses and lock bits								
0x07	READ_FUSES	Read fuses	CMDEX	Y	N	Y	ADDR	DATA
0x08	WRITE_LOCK_BITS	Write lock bits	CMDEX	N	Y	Y	ADDR	-

28.11.3.1 Write lock bits

The write lock bits command is used to program the boot lock bits to a more secure settings from software.

1. Load the NVM DATA0 register with the new lock bit value.
2. Load the NVM CMD register with the write lock bit command.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the command is finished. The CPU is halted during the complete execution of the command.

This command can be executed from both the boot loader section and the application section. The EEPROM and flash page buffers are automatically erased when the lock bits are written.

28.11.3.2 Read fuses

The read fuses command is used to read the fuses from software.

1. Load the NVM ADDR register with the address of the fuse byte to read.
2. Load the NVM CMD register with the read fuses command.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The result will be available in the NVM DATA0 register. The CPU is halted during the complete execution of the command.

28.11.4 EEPROM programming

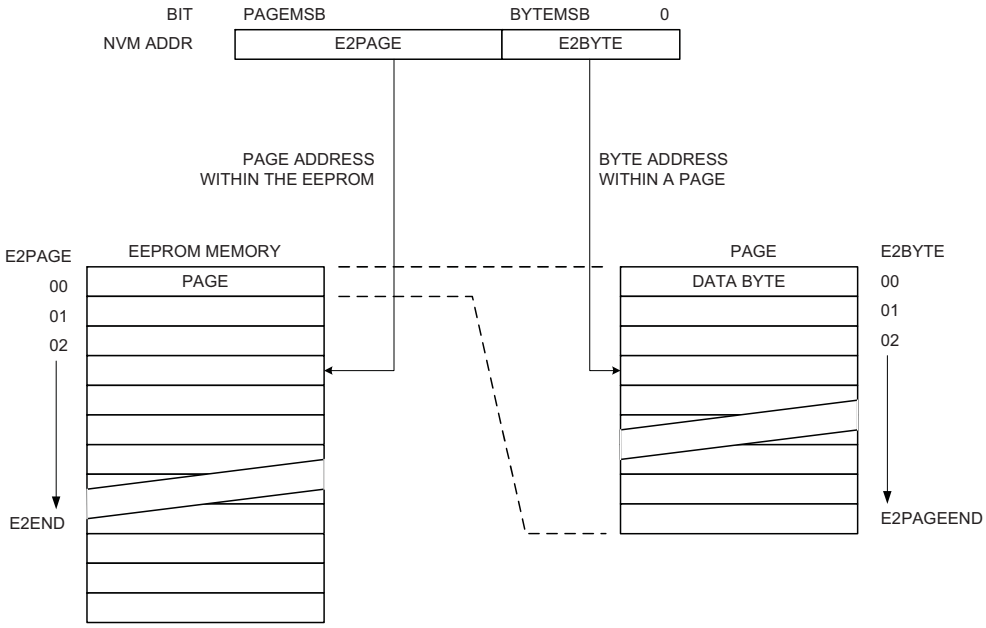
The EEPROM can be read and written from application code in any part of the flash. Its is both byte and page accessible. This means that either one byte or one page can be written to the EEPROM at once. One byte is read from the EEPROM during a read.

28.11.4.1 Addressing the EEPROM

The EEPROM is memory mapped into the data memory space to be accessed similar to SRAM.

For EEPROM page programming, the ADDR register can be treated as having two sections. The least-significant bits address the bytes within a page, while the most-significant bits address the page within the EEPROM. This is shown in Figure 28-2 on page 414. The byte address in the page (E2BYTE) is held by the bits [BYTEMSB:0] in the ADDR register. The remaining bits [PAGEMSB:BYTEMSB+1] in the ADDR register hold the EEPROM page address (E2PAGE). Together E2BYTE and E2PAGE hold an absolute address to a byte in the EEPROM. The size of E2WORD and E2PAGE will depend on the page and flash size in the device. Refer to the device datasheet for details on this.

Figure 28-2. EEPROM addressing.



Loading a data byte into the EEPROM page buffer can be performed through direct or indirect store instructions. Only the least-significant bits of the EEPROM address are used to determine locations within the page buffer, but the complete memory mapped EEPROM address is always required to ensure correct address mapping. Reading from the EEPROM can be done directly using direct or indirect load instructions. When an EEPROM page buffer load operation is performed, the CPU is halted for two cycles before the next instruction is executed.

28.11.5 NVM EEPROM commands

The NVM flash commands that can be used for accessing the EEPROM through the NVM controller are listed in Table 28-4 on page 415.

For self-programming of the EEPROM, the trigger for action-triggered commands is to set the CMDEX bit in the NVM CTRLA register (CMDEX). The read-triggered command is triggered by reading the NVM DATA0 register (DATA0).

The Change Protected column indicates whether the trigger is protected by the configuration change protection (CCP) during self-programming or not. CCP is not required for external programming. The last two columns show the address pointer used for addressing and the source/destination data register.

Section 28.11.5.1 “ Load EEPROM page buffer” on page 415 through Section 28.11.5.7 “ Read EEPROM” on page 416 explain in detail the algorithm for each EEPROM operation.

Table 28-4. EEPROM self-programming commands.

CMD[6:0]	Group configuration	Description	Trigger	CPU halted	Change protected	NVM busy	Address pointer	Data register
0x00	NO_OPERATION	No operation	-	-	-	-	-	-
EEPROM Page Buffer								
0x36	ERASE_EEPROM _BUFFER	Erase EEPROM page buffer	CMDEX	N	Y	Y	-	-
EEPROM								
0x32	ERASE_EEPROM_PAGE	Erase EEPROM page	CMDEX	N	Y	Y	ADDR	-
0x34	WRITE_EEPROM_PAGE	Write EEPROM page	CMDEX	N	Y	Y	ADDR	-
0x35	ERASE_WRITE_EEPROM_PAGE	Erase and write EEPROM page	CMDEX	N	Y	Y	ADDR	-
0x30	ERASE_EEPROM	Erase EEPROM	CMDEX	N	Y	Y	-	-

28.11.5.1 Load EEPROM page buffer

To load EEPROM page buffer, direct or indirect store instruction must be used and repeated until the arbitrary number of bytes are loaded into the page buffer.

28.11.5.2 Erase EEPROM page buffer

The erase EEPROM page buffer command is used to erase the EEPROM page buffer.

1. Load the NVM CMD register with the erase EEPROM buffer command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.11.5.3 Erase EEPROM page

The erase EEPROM page command is used to erase one EEPROM page.

1. Set up the NVM CMD register to the erase EEPROM page command.
2. Load the NVM ADDR register with the address of the EEPROM page to erase.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

The page erase commands will only erase the locations that are loaded and tagged in the EEPROM page buffer.

28.11.5.4 Write EEPROM page

The write EEPROM page command is used to write all locations loaded in the EEPROM page buffer into one page in EEPROM. Only the locations that are loaded and tagged in the EEPROM page buffer will be written.

1. Load the NVM CMD register with the write EEPROM page command.
2. Load the NVM ADDR register with the address of the EEPROM page to write.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.11.5.5 Erase and write EEPROM page

The erase and write EEPROM page command is used to first erase an EEPROM page and then write the EEPROM page buffer into that page in EEPROM in one atomic operation.

1. Load the NVM CMD register with the erase and write EEPROM page command.
2. Load the NVM ADDR register with the address of the EEPROM page to write.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.11.5.6 Erase EEPROM

The erase EEPROM command is used to erase all locations in all EEPROM pages that are loaded and tagged in the EEPROM page buffer.

1. Set up the NVM CMD register to the erase EPPROM command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.11.5.7 Read EEPROM

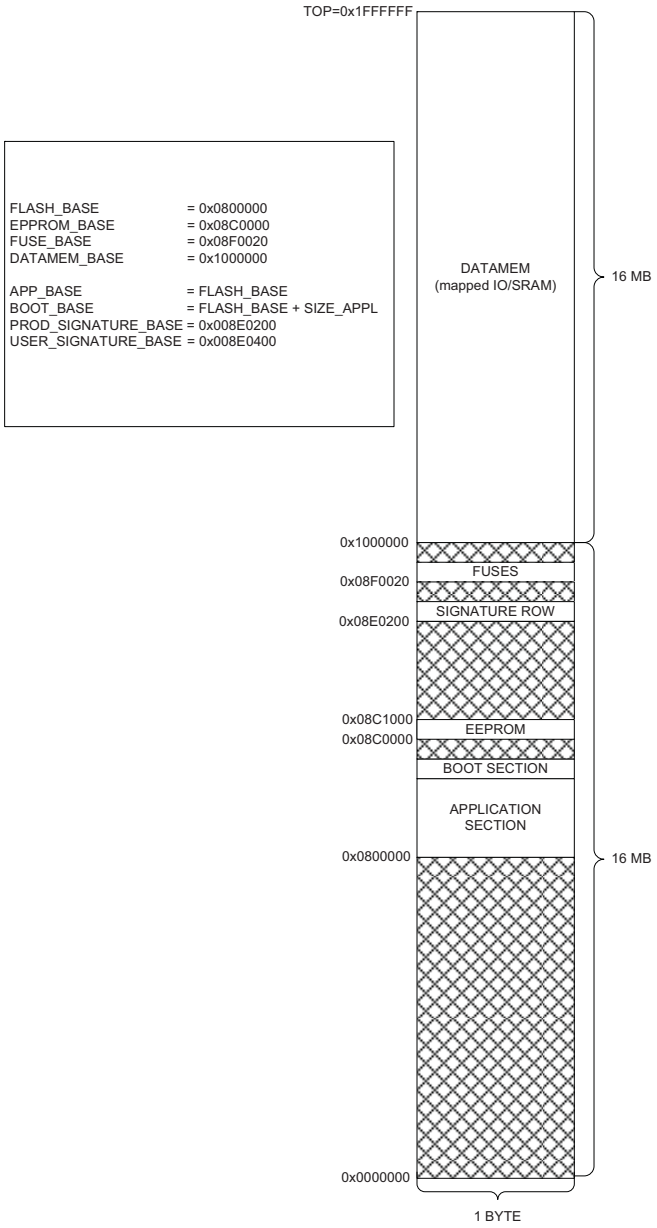
The direct or indirect load command must be used to read one byte from the EEPROM.

28.12 External programming

External programming is the method for programming code and nonvolatile data into the device from an external programmer or debugger. This can be done by both in-system or in mass production programming.

For external programming, the device is accessed through the PDI and PDI controller, and using the PDI physical connection. For details on PDI and how to enable and use the physical interface, refer to “PDI – Program and Debug Interface” on page 392. The remainder of this section assumes that the correct physical connection to the PDI is enabled. Doing this all data and program memory spaces are mapped into the linear PDI memory space. Figure 28-3 on page 417 shows the PDI memory space and the base address for each memory space in the device.

Figure 28-3. Memory map for PDI accessing the data and program memories.



28.12.1 Enabling external programming interface

NVM programming from the PDI requires enabling using the following steps:

1. Load the RESET register in the PDI with 0x59.
2. Load the NVM key in the PDI.
3. Poll NVMEN in the PDI status register (PDI STATUS) until NVMEN is set.

When the NVMEN bit in the PDI STATUS register is set, the NVM interface is enabled and active from the PDI.

28.12.2 NVM programming

When the PDI NVM interface is enabled, all memories in the device are memory mapped in the PDI address space. The PDI controller does not need to access the NVM controller's address or data registers, but the NVM controller must be loaded with the correct command (i.e., to read from any NVM, the controller must be loaded with the NVM read command before loading data from the PDIBUS address space). For the reminder of this section, all references to reading and writing data or program memory addresses from the PDI refer to the memory map shown in Figure 28-3 on page 417.

The PDI uses byte addressing, and hence all memory addresses must be byte addresses. When filling the flash or EEPROM page buffers, only the least-significant bits of the address are used to determine locations within the page buffer. Still, the complete memory mapped address for the flash or EEPROM page is required to ensure correct address mapping.

During programming (page erase and page write) when the NVM is busy, the NVM is blocked for reading.

28.12.3 NVM commands

The NVM commands that can be used for accessing the NVM memories from external programming are listed in Table 28-5 on page 418. This is a super set of the commands available for self-programming.

For external programming, the trigger for action-triggered commands is to set the CMDEX bit in the NVM CTRLA register (CMDEX). The read-triggered commands are triggered by a direct or indirect load instruction (LDS or LD) from the PDI (PDI read). The write-triggered commands are triggered by a direct or indirect store instruction (STS or ST) from the PDI (PDI write).

Section 28.12.3.1 “ Chip erase” on page 419 through Section 28.12.3.11 “ Write fuse/lock bit” on page 421 explain in detail the algorithm for each NVM operation. The commands are protected by the lock bits, and if read and write lock is set, only the chip erase and flash CRC commands are available.

Table 28-5. NVM commands available for external programming.

CMD[6:0]	Commands/operation	Trigger	Change protected	NVM busy
0x00	No operation	-	-	-
0x40	Chip erase (1)	CMDEX	Y	Y
0x43	Read NVM	PDI read	N	N
Flash Page Buffer				
0x23	Load flash page buffer	PDI write	N	N
0x26	Erase flash page buffer	CMDEX	Y	Y
Flash				
0x2B	Erase flash page	PDI write	N	Y
0x2E	Write flash page	PDI write	N	Y
0x2F	Erase and write flash page	PDI write	N	Y
0x78	Flash CRC	CMDEX	Y	Y
Application Section				
0x20	Erase application section	PDI write	N	Y
0x22	Erase application section page	PDI write	N	Y
0x24	Write application section page	PDI write	N	Y
0x25	Erase and write application section page	PDI write	N	Y

CMD[6:0]	Commands/operation	Trigger	Change protected	NVM busy
0x38	Application section CRC	CMDEX	Y	Y
Boot Loader Section				
0x68	Erase boot section	PDI write	N	Y
0x2A	Erase boot loader section page	PDI write	N	Y
0x2C	Write boot loader section page	PDI write	N	Y
0x2D	Erase and write boot loader section page	PDI write	N	Y
0x39	Boot loader section CRC	NVMAA	Y	Y
Production Signature (Calibration)[2] and User Signature Sections				
0x01	Read user signature row	PDI read	N	N
0x18	Erase user signature row	PDI write	N	Y
0x1A	Write user signature row	PDI write	N	Y
0x02	Read calibration row	PDI read	N	N
Fuses and Lock Bits				
0x07	Read fuse	PDI read	N	N
0x4C	Write fuse	PDI write	N	Y
0x08	Write lock bits	CMDEX	Y	Y
EEPROM Page Buffer				
0x33	Load EEPROM page buffer	PDI write	N	N
0x36	Erase EEPROM page buffer	CMDEX	Y	Y
EEPROM				
0x30	Erase EEPROM	CMDEX	Y	Y
0x32	Erase EEPROM page	PDI write	N	Y
0x34	Write EEPROM page	PDI write	N	Y
0x35	Erase and write EEPROM page	PDI write	N	Y
0x06	Read EEPROM	PDI read	N	N

Notes:
1. If the EESAVE fuse is programmed, the EEPROM is preserved during chip erase.
2. For consistency the name Calibration Row has been renamed to Production Signature Row throughout the document.

28.12.3.1 Chip erase

The chip erase command is used to erase the flash program memory, EEPROM and lock bits. Erasing of the EEPROM depends on EESAVE fuse setting. Refer to “FUSEBYTE5 – Fuse Byte 5” on page 32 for details. The user signature row, production signature (calibration) row, and fuses are not affected.

1. Load the NVM CMD register with the chip erase command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

Once this operation starts, the PDI bus between the PDI controller and the NVM is disabled, and the NVMEN bit in the PDI STATUS register is cleared until the operation is finished. Poll the NVMEN bit until this is set, indicating that the PDI bus is enabled.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.12.3.2 Read NVM

The read NVM command is used to read the flash, EEPROM, fuses, and signature and production signature (calibration) row sections.

1. Load the NVM CMD register with the read NVM command.
2. Read the selected memory address by executing a PDI read operation.

Dedicated read EEPROM, read fuse, read signature row and read production signature (calibration) row commands are also available for the various memory sections. The algorithm for these commands are the same as for the read NVM command.

28.12.3.3 Erase page buffer

The erase flash page buffer and erase EEPROM page buffer commands are used to erase the flash and EEPROM page buffers.

1. Load the NVM CMD register with the erase flash/EEPROM page buffer command.
2. Set the CMDEX bit in the NVM CTRLA register.

The BUSY flag in the NVM STATUS register will be set until the operation is completed.

28.12.3.4 Load page buffer

The load flash page buffer and load EEPROM page buffer commands are used to load one byte of data into the flash and EEPROM page buffers.

1. Load the NVM CMD register with the load flash/EEPROM page buffer command.
2. Write the selected memory address by doing a PDI write operation.

Since the flash page buffer is word accessed and the PDI uses byte addressing, the PDI must write the flash page buffer in the correct order. For the write operation, the low byte of the word location must be written before the high byte. The low byte is then written into the temporary register. The PDI then writes the high byte of the word location, and the low byte is then written into the word location page buffer in the same clock cycle.

The PDI interface is automatically halted before the next PDI instruction can be executed.

28.12.3.5 Erase page

The erase application section page, erase boot loader section page, erase user signature row, and erase EEPROM page commands are used to erase one page in the selected memory space.

1. Load the NVM CMD register with erase application section/boot loader section/user signature row/EEPROM page command.
2. Set the CMDEX bit in the NVM CTRLA register.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.12.3.6 Write page

The write application section page, write boot loader section page, write user signature row, and write EEPROM page commands are used to write a loaded flash/EEPROM page buffer into the selected memory space.

1. Load the NVM CMD register with write application section/boot loader section/user signature row/EEPROM page command.
2. Write the selected page by doing a PDI write. The page is written by addressing any byte location within the page.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.12.3.7 Erase and write page

The erase and write application section page, erase and write boot loader section page, and erase and write EEPROM page commands are used to erase one page and then write a loaded flash/EEPROM page buffer into that page in the selected memory space in one atomic operation.

1. Load the NVM CMD register with erase and write application section/boot loader section/user signature row/EEPROM page command.
2. Write the selected page by doing a PDI write. The page is written by addressing any byte location within the page.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.12.3.8 Erase application/ boot loader/ EEPROM section

The erase application section, erase boot loader section, and erase EEPROM section commands are used to erase the complete selected section.

1. Load the NVM CMD register with Erase Application/ Boot/ EEPROM Section command.
2. Set the CMDEX bit in the NVM CTRLA register.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

28.12.3.9 Application / boot section CRC

The application section CRC and boot loader section CRC commands can be used to verify the content of the selected section after programming.

1. Load the NVM CMD register with application/ boot loader section CRC command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished. The CRC checksum will be available in the NVM DATA register.

28.12.3.10 Flash CRC

The flash CRC command can be used to verify the content of the flash program memory after programming. The command can be executed independently of the lock bit state.

1. Load the NVM CMD register with flash CRC command.
2. Set the CMDEX bit in the NVM CTRLA register.

Once this operation starts, the PDI bus between the PDI controller and the NVM is disabled, and the NVMEN bit in the PDI STATUS register is cleared until the operation is finished. Poll the NVMEN bit until this is set again, indicting the PDI bus is enabled.

The BUSY flag in the NVM STATUS register will be set until the operation is finished. The CRC checksum will be available in the NVM DATA register.

28.12.3.11 Write fuse/lock bit

The write fuse and write lock bit commands are used to write the fuses and the lock bits to a more secure setting.

1. Load the NVM CMD register with the write fuse/ lock bit command.
2. Write the selected fuse or lock bits by doing a PDI write operation.

The BUSY flag in the NVM STATUS register will be set until the command is finished.

For lock bit write, the lock bit write command can also be used.

28.13 Register description

Refer to “Register description – NVM controller” on page 26 for a complete register description of the NVM controller.

Refer to “Register description – PDI control and status registers” on page 401 for a complete register description of the PDI.

28.14 Register summary

Refer to “Register summary – NVM controller” on page 46 for a complete register summary of the NVM controller.

Refer to “Register summary” on page 402 for a complete register summary of the PDI.

29. Peripheral Module Address Map

The address maps show the base address for each peripheral and module in XMEGA. All peripherals and modules are not present in all XMEGA devices, refer to device data sheet for the peripherals module address map for a specific device.

Table 29-1. Peripheral module address map.

Base address	Name	Description	Page
0x0000	GPIO	General purpose IO registers	42
0x0010	VPORT0	Virtual Port A	157
0x0014	VPORT1	Virtual Port C	
0x0018	VPORT2	Virtual Port D	
0x001C	VPORT3	Virtual Port R	
0x0030	CPU	CPU	19
0x0040	CLK	Clock control	111
0x0048	SLEEP	Sleep controller	116
0x0050	OSC	Oscillator control	111
0x0060	DFLLRC32M	DFLL for the 32 MHz internal RC oscillator	111
0x0070	PR	Power reduction	117
0x0078	RST	Reset controller	126
0x0080	WDT	Watch-dog timer	131
0x0090	MCU	MCU control	49
0x00A0	PMIC	Programmable multilevel interrupt controller	138
0x00B0	PORTCFG	Port configuration	158
0x00D0	CRC	CRC module	340
0x0100	EDMA	Enhanced DMA controller	75
0x0180	EVSYS	Event system	93
0x01C0	NVM	Non volatile memory (NVM) controller	46
0x0200	ADCA	Analog to digital converter on port A	371
0x0300	DACA	Digital to analog converter on port A	382
0x0380	ACA	Analog comparator pair on port A	391
0x0400	RTC	Real time counter	232
0x0460	XCL	XMEGA Custom Logic	329
0x0480	TWIC	Two wire interface on port C	259

Base address	Name	Description	Page
0x0600	PORTA	Port A	159
0x0640	PORTC	Port C	
0x0660	PORTD	Port D	
0x07E0	PORTR	Port R	
0x0800	TCC4	Timer/counter 4 on port C	194
0x0840	TCC5	Timer/counter 5 on port C	
0x0880	FAULTC4	Fault Extension on TCC4	223
0x0890	FAULTC5	Fault Extension on TCC5	
0x08A0	WEXC	Waveform extension on port C	206
0x08B0	HIRESC	High resolution extension on port C	208
0x08C0	USARTC0	USART 0 on port C	295
0x08E0	SPIC	Serial peripheral interface on port C	270
0x08F8	IRCOM	Infrared communication module	299
0x0940	TCD5	Timer/counter 5 on port D	194
0x09C0	USARTD0	USART 0 on port D	295

30. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	Rd ← Rd + 1:Rd + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd + 1:Rd ← Rd + 1:Rd - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2
Branch instructions					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2
EIJMP		Extended Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	2
JMP	k	Jump	PC ← k	None	3
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	2 / 3[1]

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ICALL		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2 / 3[1]
EICALL		Extended Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	3[1]
CALL	k	call Subroutine	PC ← k	None	3 / 4[1]
RET		Subroutine Return	PC ← STACK	None	4 / 5[1]
RETI		Interrupt Return	PC ← STACK	I	4 / 5[1]
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	2 / 3 / 4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC ← PC + 2 or 3	None	2 / 3 / 4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data transfer instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
LDS	Rd, k	Load Direct from data space	Rd ← (k)	None	2[1][2]
LD	Rd, X	Load Indirect	Rd ← (X)	None	1[1][2]
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X) X ← X + 1	None	1[1][2]
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, ← X - 1 Rd ← (X) ← (X)	None	2[1][2]
LD	Rd, Y	Load Indirect	Rd ← (Y) ← (Y)	None	1[1][2]
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y) Y ← Y + 1	None	1[1][2]
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1 Rd ← (Y)	None	2[1][2]
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2[1][2]
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	1[1][2]
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	1[1][2]
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2[1][2]
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2[1][2]
STS	k, Rr	Store Direct to Data Space	(k) ← Rd	None	2[1]
ST	X, Rr	Store Indirect	(X) ← Rr	None	1[1]
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	1[1]
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2[1]
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	1[1]
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	1[1]
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2[1]
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2[1]
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	1[1]
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr Z ← Z + 1	None	1[1]
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1	None	2[1]
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2[1]
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Increment	Rd ← (Z), Z ← Z + 1	None	3
ELPM		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z), Z ← Z + 1	None	3
SPM		Store Program Memory	(RAMPZ:Z) ← R1:R0	None	-

Mnemonics	Operands	Description	Operation	Flags	#Clocks
SPM	Z+	Store Program Memory and Post-Increment by 2	(RAMPZ:Z) ← R1:R0, Z ← Z + 2	None	-
IN	Rd, A	In From I/O Location	Rd ← I/O(A)	None	1
OUT	A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	1[(1)]
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2[(1)]
XCH	Z, Rd	Exchange RAM location	Temp ← Rd, Rd ← (Z), (Z) ← Temp	None	2
LAS	Z, Rd	Load and Set RAM location	Temp ← Rd, Rd ← (Z), (Z) ← Temp v (Z)	None	2
LAC	Z, Rd	Load and Clear RAM location	Temp ← Rd, Rd ← (Z), (Z) ← ($FFh – Rd) • (Z)	None	2
LAT	Z, Rd	Load and Toggle RAM location	Temp ← Rd, Rd ← (Z), (Z) ← Temp ⊕ (Z)	None	2
Bit and bit-test instructions					
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0, C ← Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	1
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU control instructions					
BREAK		Break	(See specific descr. for BREAK)	None	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1

Notes: 1. Cycle times for data memory accesses assume internal memory accesses, and are not valid for accesses via the external RAM interface.
2. One extra cycle must be added when accessing Internal SRAM.

31. Revision History

Please note that the referring page numbers in this section are referring to this document. The referring revision in this section are referring to the document revision.

31.1 42005C – 08/2013

1.	ADC:
	• "Optional gain" replaced with "Programmable gain" in:
	  ◦ "Features" on page 341, "Overview" on page 341, "Input sources" on page 342.
	  ◦ Headings of Table 24-15, Table 24-16 and Table 24-17.
	• "Differential inputs" on page 342 updated.
	• "Single-ended input" on page 343 updated.
	• "Internal inputs" on page 344 updated.
	• "Single conversion with 1x gain" on page 352: Heading updated from saying "Single conversion without gain".
	• "Single conversion with various gain settings" on page 352: Heading updated from saying "Single conversion with gain".

2.	DAC register and bit updates: "CH0GAINCAL – Gain Calibration register" on page 378: Removed "+0x0A" address. "CH0OFFSETCAL – Offset Calibration register" on page 378: Corrected name of Bit 7:0 to CH0OFFSETCAL[7:0]. "CH1GAINCAL – Gain Calibration register" on page 379: Corrected name of Bit 7:0 to CH1GAINCAL[7:0]

31.2 42005B – 04/2013

1.	Updated "ADC clock and conversion timing" on page 351.

31.3 42005A – 04/2013

1.	Initial revision

Table of Contents

1. About the manual 2
   - 1.1 Reading the manual 2
   - 1.2 Resources 2
   - 1.3 Recommended reading 2

2. Overview 3
   - 2.1 Block diagram 4

3. Atmel AVR CPU 7
   - 3.1 Features 7
   - 3.2 Overview 7
   - 3.3 Architectural overview 7
   - 3.4 ALU - Arithmetic logic unit 8
   - 3.5 Program flow 9
   - 3.6 Instruction execution timing 9
   - 3.7 Status register 10
   - 3.8 Stack and stack pointer 10
   - 3.9 Register file 10
   - 3.10 RAMP and extended indirect registers 12
   - 3.11 Accessing 16-bit registers 13
   - 3.12 Configuration change protection 13
   - 3.13 Fuse lock 14
   - 3.14 Register descriptions 15
   - 3.15 Register summary 19

4. Memories 20
   - 4.1 Features 20
   - 4.2 Overview 20
   - 4.3 Flash program memory 20
   - 4.4 Fuses and lockbits 22
   - 4.5 Data memory 22
   - 4.6 Internal SRAM 23
   - 4.7 EEPROM 23
   - 4.8 I/O memory 23
   - 4.9 Data memory and bus arbitration 23
   - 4.10 Memory timing 24
   - 4.11 Device ID and revision 24
   - 4.12 I/O memory protection 25
   - 4.13 Register description – NVM controller 26
   - 4.14 Register descriptions – Fuses and lock bits 30
   - 4.15 Register description – Production signature row 36
   - 4.16 Register description – General purpose I/O memory 42
   - 4.17 Register descriptions – MCU control 42
   - 4.18 Register summary – NVM controller 46
   - 4.19 Register summary – Fuses and lockbits 46
   - 4.20 Register summary – Production signature row 46
   - 4.21 Register summary – General purpose I/O registers 49
   - 4.22 Register summary – MCU control 49
   - 4.23 Interrupt vector summary 49

5. EDMA – Enhanced Direct Memory Access 50
5.1 Features 50
5.2 Overview 50
5.3 EDMA transaction 52
5.4 Transfer triggers 53
5.5 Addressing and transfer count 53
5.6 Priority between channels 54
5.7 Double buffering 54
5.8 Data processing 55
5.9 Error detection 57
5.10 Software reset 57
5.11 Protection 58
5.12 Interrupts 58
5.13 Register description – EDMA controller 59
5.14 Register description – Peripheral channel 61
5.15 Register description – Standard channel 67
5.16 Register summary – EDMA controller in PER0123 configuration. 75
5.17 Register summary – EDMA controller in STD0 configuration. 75
5.18 Register summary – EDMA controller in STD2 configuration. 76
5.19 Register summary – EDMA controller in STD02 configuration. 76
5.20 Register summary – EDMA peripheral channel. 77
5.21 Register Summary – EDMA standard channel. 77
5.22 Interrupt vector summary 78

6. Event System 79
6.1 Features 79
6.2 Overview 79
6.3 Events 80
6.4 Signaling events 81
6.5 Data events 81
6.6 Peripheral clock events 81
6.7 Software events 81
6.8 Event routing network 81
6.9 Event timing 84
6.10 Filtering 84
6.11 Quadrature decoder 85
6.12 Register description 88
6.13 Register summary 93

7. System Clock and Clock Options 94
7.1 Features 94
7.2 Overview 94
7.3 Clock distribution 95
7.4 Clock sources 96
7.5 System clock selection and prescalers 98
7.6 PLL with 1x-31x multiplication factor 99
7.7 DFLL 32MHz 99
7.8 PLL and external clock source failure monitor 100
7.9 Register description – Clock 101
7.10 Register description – Oscillator 104
7.11 Register description – DFLL32M 108

7.12 Register summary - Clock . . . 111
7.13 Register summary - Oscillator . . . 111
7.14 Register summary – DFLL32M . . . 111
7.15 Interrupt vector summary . . . 111

8. Power Management and Sleep Modes . . . 112
8.1 Features . . . 112
8.2 Overview . . . 112
8.3 Sleep modes . . . 112
8.4 Power reduction registers . . . 114
8.5 Minimizing power consumption . . . 114
8.6 Register description – Sleep . . . 116
8.7 Register description – Power reduction . . . 117
8.8 Register summary – Sleep . . . 119
8.9 Register summary – Power reduction . . . 119

9. Reset System . . . 120
9.1 Features . . . 120
9.2 Overview . . . 120
9.3 Reset sequence . . . 121
9.4 Reset sources . . . 122
9.5 Register description . . . 126
9.6 Register summary . . . 126

10. WDT – Watchdog Timer . . . 127
10.1 Features . . . 127
10.2 Overview . . . 127
10.3 Normal mode operation . . . 127
10.4 Window mode operation . . . 128
10.5 Watchdog timer clock . . . 128
10.6 Configuration protection and lock . . . 128
10.7 Registers description . . . 129
10.8 Register summary . . . 131

11. PMIC – Interrupts and Programmable Multilevel Interrupt Controller . 132
11.1 Features . . . 132
11.2 Overview . . . 132
11.3 Operation . . . 132
11.4 Interrupts . . . 133
11.5 Interrupt level . . . 134
11.6 Interrupt priority . . . 135
11.7 Interrupt vector locations . . . 136
11.8 Register description . . . 137
11.9 Register summary . . . 138

12. I/O Ports . . . 139
12.1 Features . . . 139
12.2 Overview . . . 139
12.3 I/O pin use and configuration . . . 140
12.4 Reading the pin value . . . 144
12.5 Input sense configuration . . . 144

12.6 Port interrupt . . . 145
12.7 Port event . . . 146
12.8 Alternate port functions . . . 146
12.9 Slew rate control . . . 147
12.10 Clock and event output . . . 147
12.11 Multi-pin configuration . . . 148
12.12 Virtual ports . . . 148
12.13 Register descriptions – Ports . . . 149
12.14 Register descriptions – Port configuration . . . 154
12.15 Register descriptions – Virtual port . . . 157
12.16 Register summary – Ports . . . 158
12.17 Register summary – Port configuration . . . 159
12.18 Register summary – Virtual ports . . . 159
12.19 Interrupt vector summary – Ports . . . 159

13. TC4/5 – 16-bit Timer/Counter Type 4 and 5 . . . 160
13.1 Features . . . 160
13.2 Overview . . . 160
13.3 Block diagram . . . 162
13.4 Clock and event sources . . . 163
13.5 Double buffering . . . 163
13.6 Counter operation . . . 164
13.7 Capture channel . . . 167
13.8 Compare channel . . . 169
13.9 Interrupts and events . . . 172
13.10 EDMA support . . . 172
13.11 Timer/Counter commands . . . 173
13.12 Register description – Standard configuration . . . 174
13.13 Register description – Byte mode configuration . . . 184
13.14 Register summary – Standard configuration . . . 194
13.15 Interrupt vector summary – Standard configuration . . . 195
13.16 Register summary – Byte configuration . . . 196
13.17 Interrupt vector summary – Byte configuration . . . 197

14. WeX – Waveform Extension . . . 198
14.1 Features . . . 198
14.2 Overview . . . 198
14.3 Port override . . . 199
14.4 Output matrix . . . 199
14.5 Dead-time generator . . . 200
14.6 Pattern generator . . . 201
14.7 Change protection . . . 202
14.8 Register description . . . 203
14.9 Register summary . . . 206

15. Hi-Res – High-Resolution Extension . . . 207
15.1 Features . . . 207
15.2 Overview . . . 207
15.3 Register description . . . 208
15.4 Register summary . . . 208

16. Fault Extension . . . 209

16.1 Features . . . 209
16.2 Overview . . . 209
16.3 Timer/counter considerations . . . 209
16.4 Faults . . . 210
16.5 Register description . . . 217
16.6 Register summary . . . 223

17. RTC – Real Time Counter . . . 224
17.1 Features . . . 224
17.2 Overview . . . 224
17.3 Clock domains . . . 225
17.4 Interrupts and events . . . 225
17.5 Correction . . . 225
17.6 Register description . . . 227
17.7 Register summary . . . 232
17.8 Interrupt vector summary . . . 232

18. TWI – Two-Wire Interface . . . 233
18.1 Features . . . 233
18.2 Overview . . . 233
18.3 General TWI bus concepts . . . 234
18.4 TWI bus state logic . . . 239
18.5 TWI master operation . . . 240
18.6 TWI slave operation . . . 241
18.7 Enabling external driver interface . . . 243
18.8 Bridge mode . . . 243
18.9 SMBUS L1 Compliance . . . 244
18.10 Register description – TWI . . . 247
18.11 Register description – TWI master . . . 248
18.12 Register description – TWI slave . . . 253
18.13 Register Description – TWI Timeout . . . 257
18.14 Register summary - TWI . . . 259
18.15 Register summary - TWI master . . . 259
18.16 Register summary - TWI slave . . . 259
18.17 Register Summary – TWI timeout . . . 259
18.18 Interrupt vector summary . . . 259

19. SPI – Serial Peripheral Interface . . . 260
19.1 Features . . . 260
19.2 Overview . . . 260
19.3 Master mode . . . 261
19.4 Slave mode . . . 261
19.5 Buffer modes . . . 262
19.6 Data modes . . . 262
19.7 Interrupts . . . 264
19.8 EDMA support . . . 264
19.9 Register description . . . 265
19.10 Register summary . . . 270
19.11 Interrupt vector summary . . . 270

20. USART . . . 271
20.1 Features . . . 271

20.2 Overview . . . 271
20.3 Clock generation . . . 273
20.4 Frame formats . . . 276
20.5 USART full-duplex initialization . . . 277
20.6 USART one-wire initialization . . . 277
20.7 Data transmission - The USART transmitter . . . 277
20.8 Data reception - The USART receiver . . . 278
20.9 Asynchronous data reception . . . 279
20.10 Fractional baud rate generation . . . 282
20.11 USART in master SPI mode . . . 285
20.12 USART SPI vs. SPI . . . 285
20.13 Multiprocessor communication mode . . . 286
20.14 One-wire mode . . . 286
20.15 Data encoding/decoding . . . 286
20.16 IRCOM mode of operation . . . 287
20.17 EDMA support . . . 287
20.18 Register description . . . 288
20.19 Register summary . . . 295
20.20 Interrupt vector summary – USART . . . 295

21. IRCOM – IR Communication Module . . . 296
21.1 Features . . . 296
21.2 Overview . . . 296
21.3 Registers description . . . 298
21.4 Register summary . . . 299

22. XCL – XMEGA Custom Logic . . . 300
22.1 Features . . . 300
22.2 Overview . . . 300
22.3 Timer/counter configuration . . . 302
22.4 Timer/counter operation . . . 302
22.5 Glue logic . . . 311
22.6 Interrupts and events . . . 316
22.7 Register description . . . 317
22.8 Register summary . . . 329
22.9 Interrupt vector summary . . . 335
22.10 T/C and PEC register summary vs. Configuration and mode . . . 335

23. CRC – Cyclic Redundancy Check generator . . . 336
23.1 Features . . . 336
23.2 Overview . . . 336
23.3 Operation . . . 336
23.4 CRC on Flash memory . . . 337
23.5 CRC on EDMA Data . . . 337
23.6 CRC using the I/O Interface . . . 337
23.7 Register Description . . . 338
23.8 Register summary . . . 340

24. ADC – Analog to Digital Converter . . . 341
24.1 Features . . . 341
24.2 Overview . . . 341
24.3 Input sources . . . 342

24.4 Sampling time control . . . 345
24.5 Voltage reference selection . . . 345
24.6 Conversion result . . . 345
24.7 Calibration and correction . . . 348
24.8 Starting a conversion . . . 349
24.9 ADC clock and conversion timing . . . 351
24.10 ADC input model . . . 354
24.11 EDMA transfer . . . 354
24.12 Interrupts and events . . . 354
24.13 Synchronous sampling . . . 355
24.14 Register description – ADC . . . 356
24.15 Register description - ADC channel . . . 363
24.16 Register summary – ADC . . . 371
24.17 Register summary – ADC channel . . . 372
24.18 Interrupt vector summary . . . 372

25. DAC – Digital to Analog Converter . . . 373
25.1 Features . . . 373
25.2 Overview . . . 373
25.3 Voltage reference selection . . . 374
25.4 Starting a conversion . . . 374
25.5 Output and output channels . . . 374
25.6 DAC output model . . . 374
25.7 DAC clock . . . 374
25.8 Low power mode . . . 375
25.9 Calibration . . . 375
25.10 Register description . . . 376
25.11 Register summary . . . 382

26. AC – Analog Comparator . . . 383
26.1 Features . . . 383
26.2 Overview . . . 383
26.3 Input sources . . . 384
26.4 Signal compare . . . 384
26.5 Interrupts and events . . . 384
26.6 Window mode . . . 385
26.7 Input hysteresis . . . 385
26.8 Register description . . . 386
26.9 Register summary . . . 391
26.10 Interrupt vector summary . . . 391

27. PDI – Program and Debug Interface . . . 392
27.1 Features . . . 392
27.2 Overview . . . 392
27.3 PDI physical . . . 393
27.4 PDI controller . . . 397
27.5 Register description – PDI instruction and addressing registers . . . 399
27.6 Register description – PDI control and status registers . . . 401
27.7 Register summary . . . 402

28. Memory Programming . . . 403
28.1 Features . . . 403

28.2 Overview . . . 403
28.3 NVM controller . . . 403
28.4 NVM commands . . . 404
28.5 NVM controller busy status . . . 404
28.6 Flash and EEPROM page buffers . . . 404
28.7 Flash and EEPROM programming sequences . . . 405
28.8 Protection of NVM . . . 406
28.9 Preventing NVM corruption . . . 406
28.10 CRC functionality . . . 406
28.11 Self-programming and boot loader support . . . 407
28.12 External programming . . . 416
28.13 Register description . . . 421
28.14 Register summary . . . 421

29. Peripheral Module Address Map . . . 422

30. Instruction Set Summary . . . 424

31. Revision History . . . 429
31.1 42005C – 08/2013 . . . 429
31.2 42005B – 04/2013 . . . 429
31.3 42005A – 04/2013 . . . 429

Table of Contents . . . i

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Roa
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81) (3) 6417-0300
Fax: (+81) (3) 6417-0370