XMEGA B MANUAL

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

- Atmel AVR CPU
- Memories
- DMAC - Direct memory access controller
- Event system
- System clock and clock options
- Power management and sleep modes
- System control and reset
- WDT - Watchdog timer
- Interrupts and programmable multilevel interrupt controller
- PORT - I/O ports
- TC - 16-bit timer/counters
- AWeX - Advanced waveform extension
- Hi-Res - High resolution extension
- RTC - Real-time counter
- USB - Universal serial bus interface
- TWI - Two-wire serial interface
- SPI - Serial peripheral interface
- USART - Universal synchronous and asynchronous serial receiver and transmitter
- IRCOM - Infrared communication module
- AES and DES cryptographic engine
- CRC - Cyclic redundancy check
- LCD - Liquid Crystal Display controller
- ADC - Analog-to-digital converter
- AC - Analog comparator
- IEEE 1149.1 JTAG interface
- PDI - Program and debug interface
- Memory programming
- Peripheral address map
- Register summary
- Interrupt vector summary
- Instruction set summary

1. About the Manual

This document contains in-depth documentation of all peripherals and modules available for the Atmel AVR XMEGA B 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 Atmel AVR XMEGA B 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

- Atmel AVR XMEGA B device datasheets
- AVR XMEGA application notes

This manual contains general modules and peripheral descriptions. The AVR XMEGA B 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 B 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 Atmel AVR XMEGA B 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 Atmel AVR XMEGA B devices provide the following features: in-system programmable flash with read-while-write capabilities; internal EEPROM and SRAM; two-channel DMA controller; four-channel event system and programmable multilevel interrupt controller; up to 53 general purpose I/O lines; 16-bit real-time counter (RTC); 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 interface (TWI); one full-speed USB 2.0 interface; one serial peripheral interface (SPI); one LCD controller supporting display capacity up to 4 Common and up to 40 Segment terminals; CRC module; AES and DES cryptographic engine; up to two 8-channel, 12-bit ADCs with programmable gain; up to four analog comparators with window mode; programmable watchdog timer with separate internal oscillator; accurate internal oscillators with PLL and prescaler; and programmable brown-out detection.

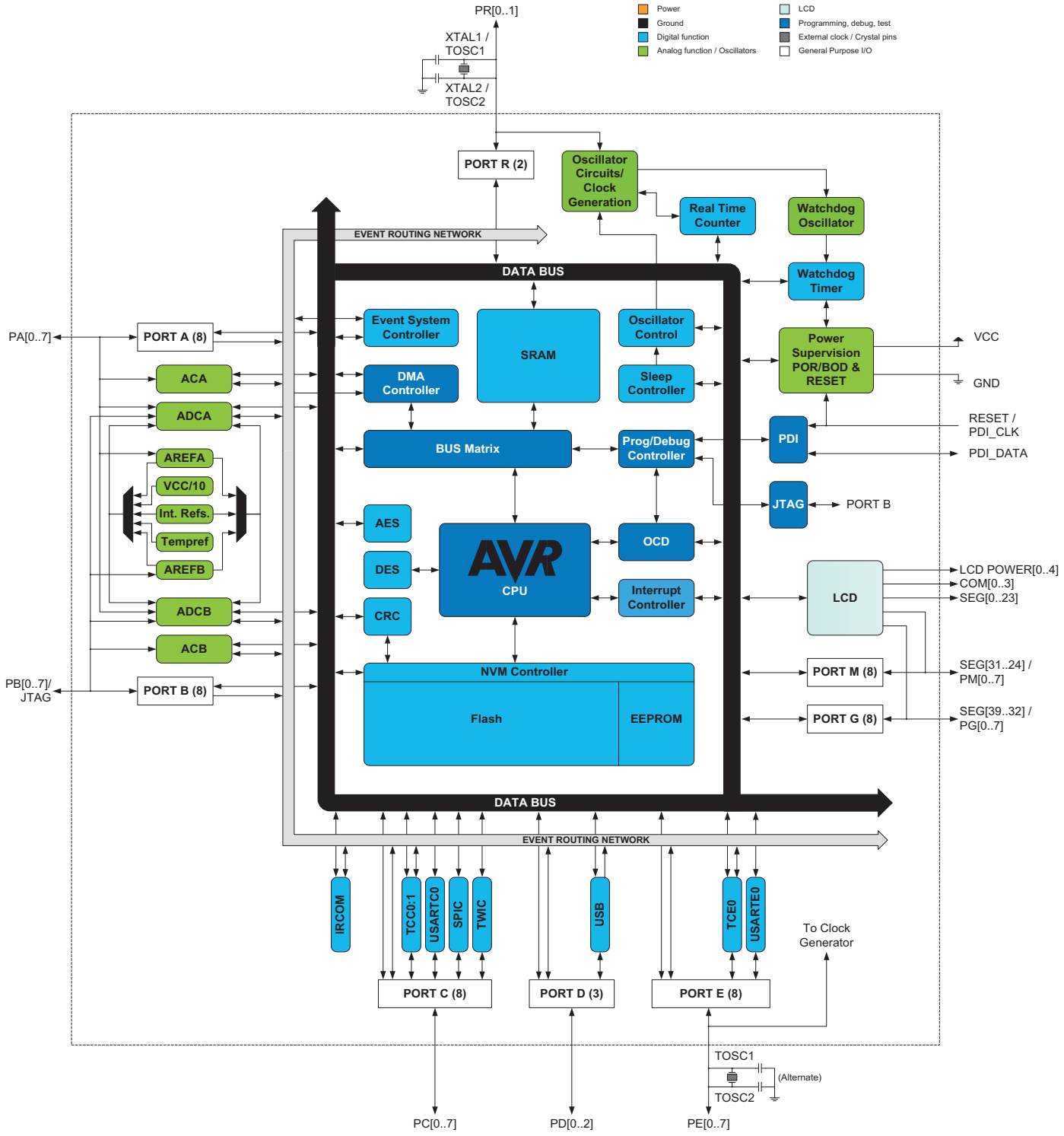
The program and debug interface (PDI), a fast, two-pin interface for programming and debugging, is available. Selected devices also have an IEEE std. 1149.1 compliant JTAG interface, and this can also be used for on-chip debug and programming.

The Atmel AVR XMEGA devices have five software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, DMA 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, USB resume, 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 this mode, the LCD controller is allowed to refresh data to the panel. 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. In this mode, the LCD controller is allowed to refresh data to the panel. To further reduce power consumption, the peripheral clock to each individual peripheral can optionally be stopped in active mode and idle sleep mode.

The devices are manufactured using Atmel high-density, nonvolatile memory technology. The program flash memory can be reprogrammed in-system through the PDI or JTAG interfaces. A boot loader running in the device can use any interface to download the application program to the flash memory. The boot loader software in the boot flash section will continue to run while the application flash section is updated, providing true read-while-write operation. 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 Atmel AVR XMEGA B 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.

Figure 2-1. Atmel AVR XMEGA B block diagram.



In Table 2-1 on page 5 a feature summary for the XMEGA B 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 B feature summary overview.

Feature	Details / sub-family	B1	B3
Pins, I/O	*Total*	100	64
	Programmable I/O pins	53	36
Memory	*Program memory (KB)*	64 - 128	64 - 128
	Boot memory (KB)	4 - 8	4 - 8
	SRAM (KB)	4 - 8	4 - 8
	EEPROM	2	2 - 4
	General purpose registers	16	16
Package	*TQFP*	100A	64A
	QFN /VQFN	–	64M2
	BGA	100C1/100C2	–
QTouch	*Sense channels*	56	56
DMA Controller	*Channels*	2	2
Event System	*Channels*	4	4
	QDEC	1	1
Crystal Oscillator	*0.4 - 16MHz XOSC*	Yes	Yes
	32.768 kHz TOSC	Yes	Yes
Internal Oscillator	*2MHz calibrated*	Yes	Yes
	32MHz calibrated	Yes	Yes
	128MHz PLL	Yes	Yes
	32.768kHz calibrated	Yes	Yes
	32kHz ULP	Yes	Yes
Timer / Counter	*TC0 - 16-bit, 4 CC*	2	1
	TC1 - 16-bit, 2 CC	1	1
	TC2 - 2x 8-bit	2	1
	Hi-Res	1	1
	AWeX	1	1
	RTC	1	1
	RTC32		
Serial Communication	*USB full-speed device*	1	1
	USART	2	1
	SPI	1	1
	TWI	1	1

Feature	Details / sub-family	B1	B3
Crypto /CRC	*AES-128*	Yes	Yes
	DES	Yes	Yes
	CRC-16	Yes	Yes
	CRC-32	Yes	Yes
Liquid Crystal Display Controller (LCD)	*Segments*	*40*	*25*
	Common terminals	*4*	*4*
Analog to Digital Converter (ADC)		2	1
	Resolution (bits)	12	12
	Sampling speed (kbps)	300	300
	Input channels per ADC	16	8
	Conversion channels	1	1
Analog Comparator (AC)		4	2
Program and Debug Interface	*PDI*	Yes	Yes
	JTAG	Yes	Yes
	Boundary scan	Yes	Yes

3. Atmel AVR CPU

3.1 Features

- 8/16-bit, high-performance Atmel AVR RISC CPU
  - 142 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

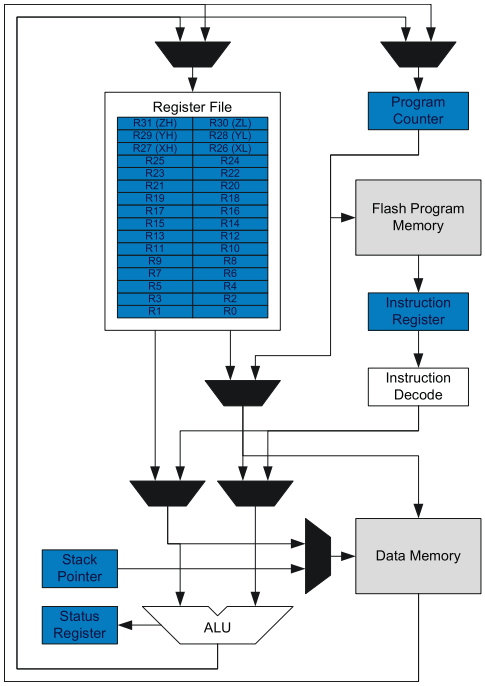
3.2 Overview

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, “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

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 395. 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 can be 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 memory mapping of 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 on page 9 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.

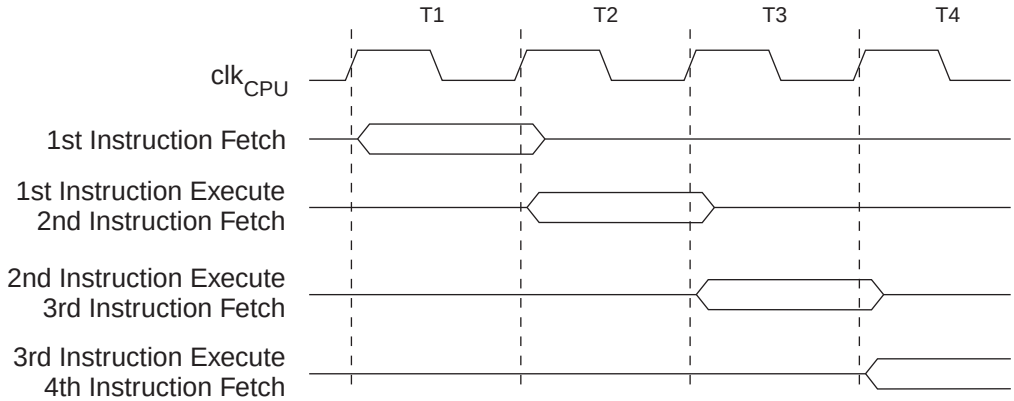
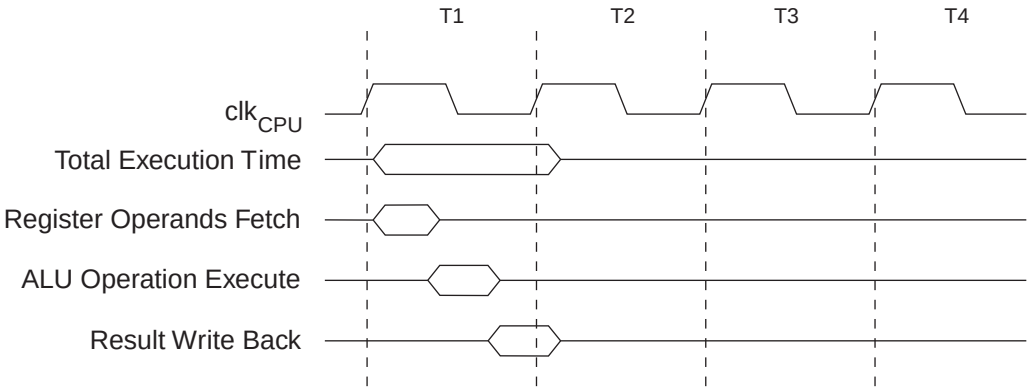


Figure 3-3 on page 9 shows the internal timing concept for the register file. In a single clock cycle, an ALU operation using two register operands is executed and the result is stored back to the destination register.

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 reference. 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.

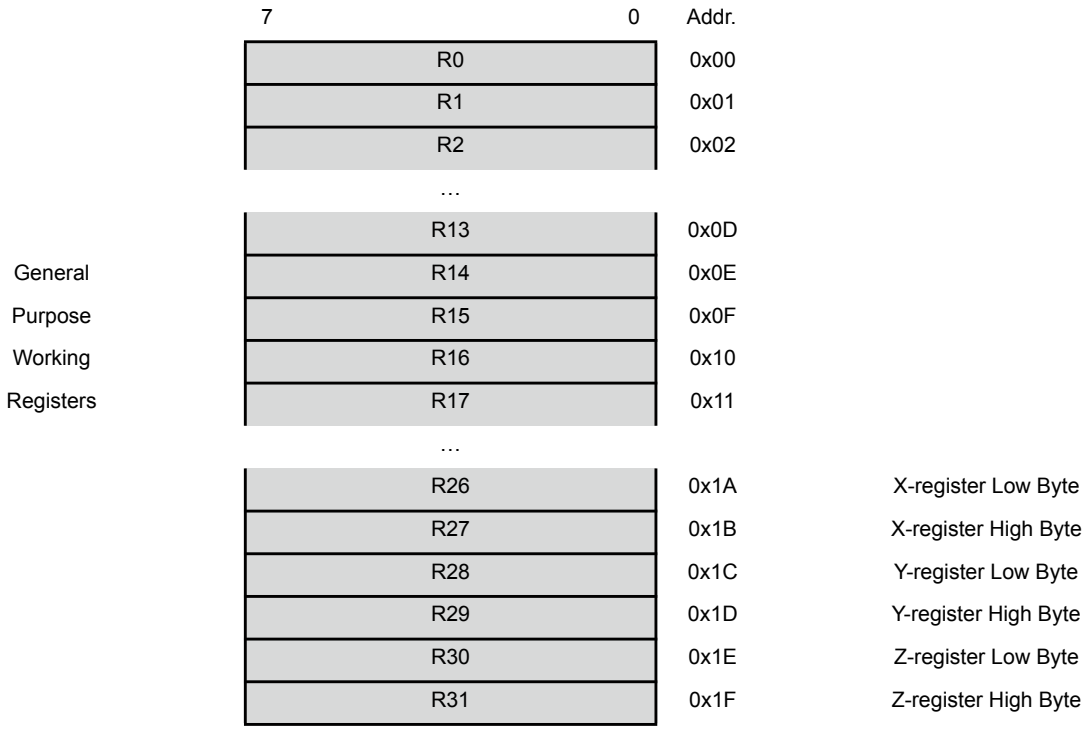
3.9 Register File

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.



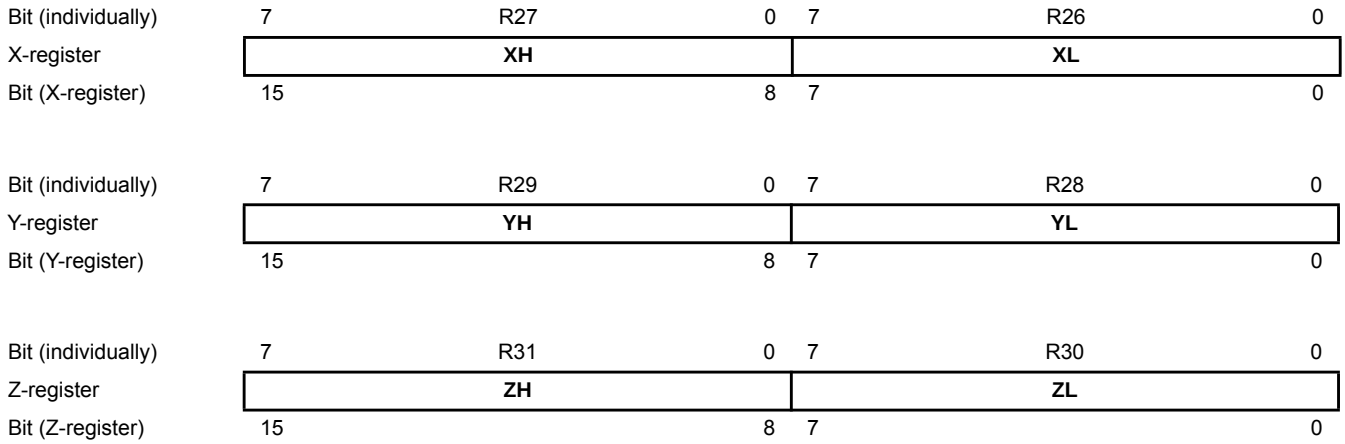
The register file is located in a separate address space, and so the registers are not accessible as data memory.

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 reference for details).

3.10 RAMP and Extended Indirect Registers

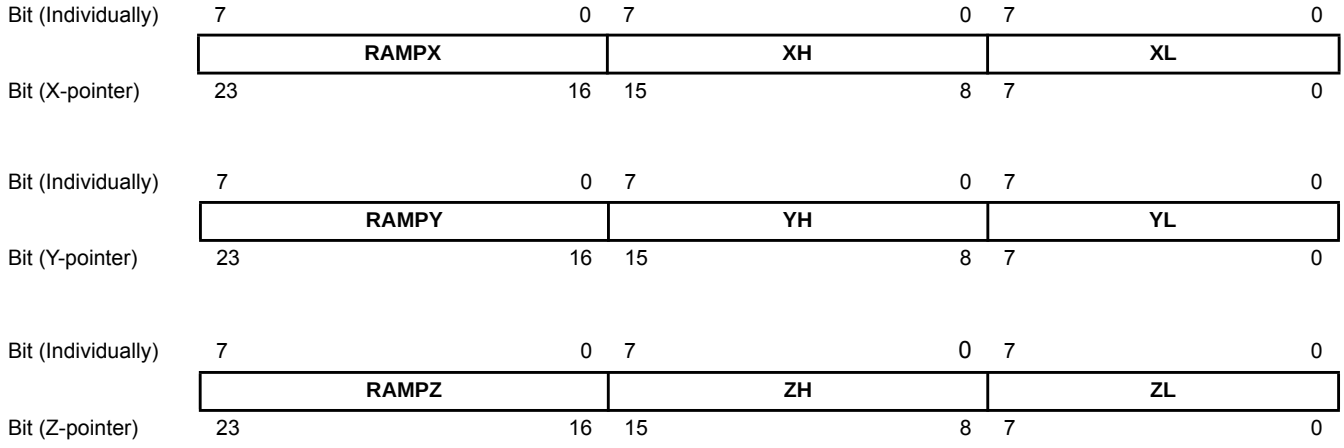
In order to access program memory or data memory above 64KB, the address pointer must be larger than 16 bits. This is done by concatenating one register to one of the X-, Y-, or Z-registers. This register then holds the most-significant byte (MSB) in a 24-bit address or address pointer.

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.

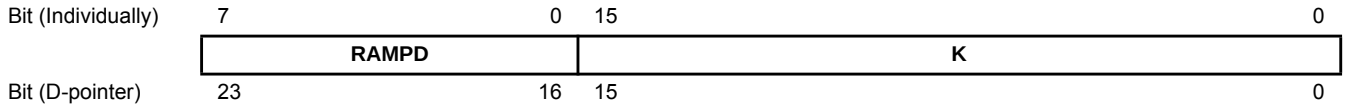


When reading (ELPM) and writing (SPM) program memory locations above the first 128KB of the program memory, RAMPZ is concatenated with the Z-register to form the 24-bit address. LPM is not affected by the RAMPZ setting.

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.

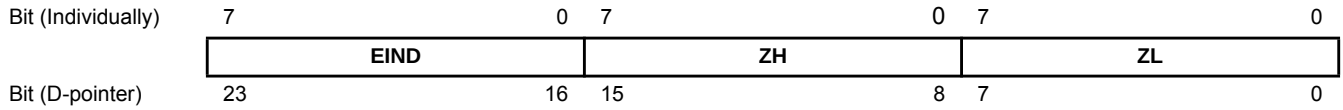
Figure 3-7. The combined RAMPD + K register.



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

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 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 register 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 initial values.

- **Bit 7:0 – SP[7:0]: Stack Pointer Register Low**

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 exact initial values.

- **Bit 7:0 – SP[15:8]: Stack Pointer Register High**

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.

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 Description.” 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. See “Instruction Set Description” for detailed information.

- **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. See “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The two’s complement overflow flag (V) supports two’s complement arithmetic. See “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The negative flag (N) indicates a negative result in an arithmetic or logic operation. See “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The zero flag (Z) indicates a zero result in an arithmetic or logic operation. See “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The carry flag (C) indicates a carry in an arithmetic or logic operation. See “Instruction Set Description” for detailed information.

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]								16
+0x0A	RAMPY	RAMPY[7:0]								16
+0x0B	RAMPZ	RAMPZ[7:0]								16
+0x0C	EIND	EIND[7:0]								17
+0x0D	SPL	SPL[7:0]								17
+0x0E	SPH	SPH[7:0]								17
+0x0F	SREG	I	T	H	S	V	N	Z	C	18

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
    - Optional memory mapping 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
    - Safe and deterministic handling of priority between CPU, DMA controller, and other bus masters
  - Separate buses for SRAM, EEPROM, I/O memory, and external memory access
    - Simultaneous bus access for CPU and DMA 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.

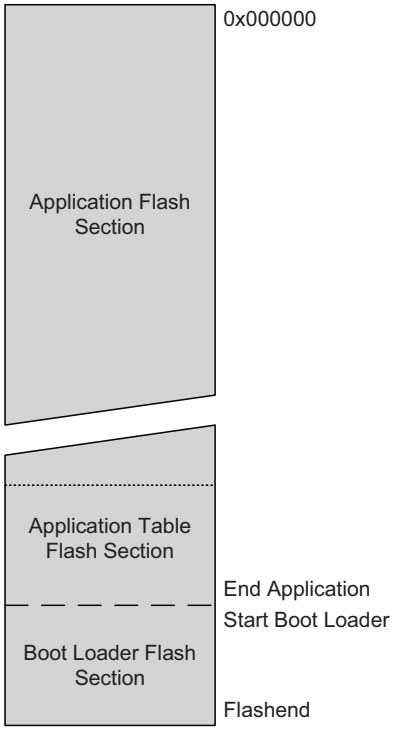
4.3 Flash Program Memory

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, which is 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 only initiate programming when executing from this section. 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.

For accessing the production signature row, refer to “NVM Flash Commands” on page 380.

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 reset sources such as brownout detector, watchdog and startup configuration.

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.

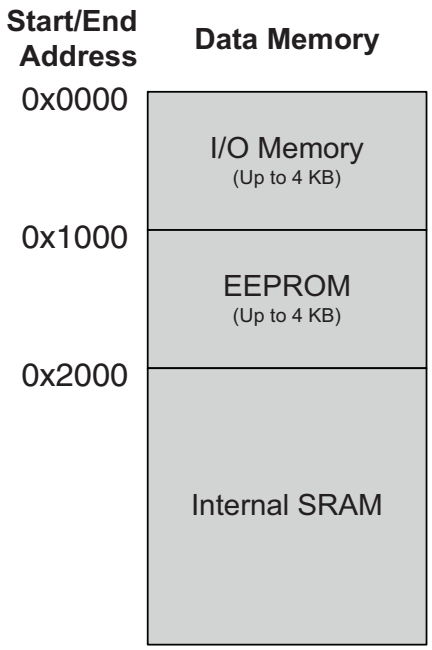
Both fuses and lock bits are reprogrammable like the flash program memory.

For some fuse bytes, leaving them unprogrammed (0xFF) will result in invalid settings. The user must ensure that the fuse bytes are programmed to values which give valid settings. Refer to the detailed description of the individual fuse bytes for further information.

4.5 Data Memory

The data memory contains the I/O memory, internal SRAM, optionally memory mapped and EEPROM. 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 data space (default) or memory mapped and accessed in normal data space. The EEPROM supports both byte and page access. Memory mapped EEPROM allows highly efficient EEPROM reading and EEPROM buffer loading. When doing this, EEPROM is accessible using load and store instructions. Memory mapped EEPROM will always start at 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

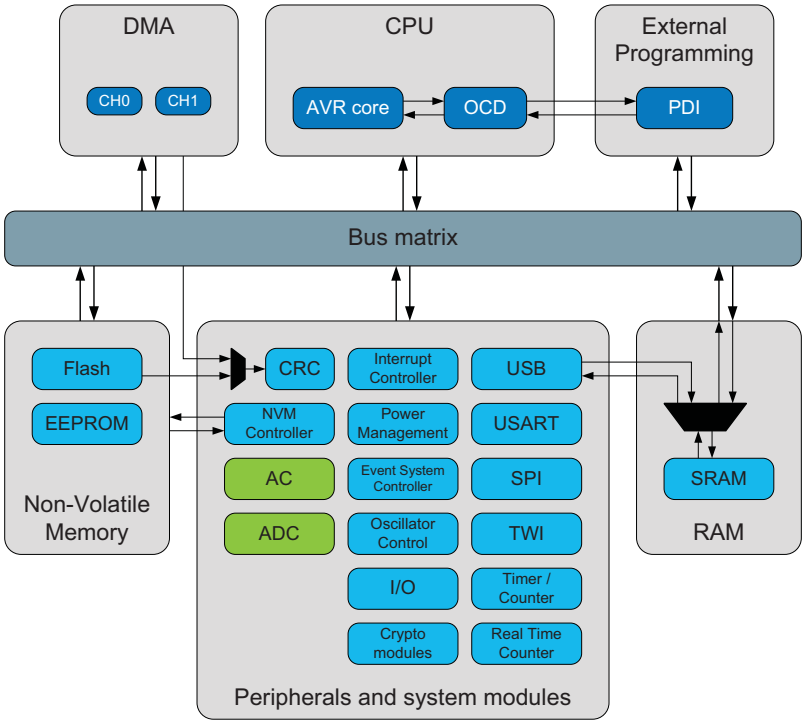
The lowest 4 I/O memory addresses are reserved as general purpose I/O registers. These registers can be used for storing global variables and flags, as they are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

4.9 Data Memory and Bus Arbitration

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

The USB module acts as a bus master, and is connected directly to internal SRAM through a pseudo-dual-port (PDP) interface.

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.
   - Alternating DMA controller read and DMA controller write when they access the same data memory section.
3. Bus Master requesting burst access.
   - CPU has priority.
4. Bus Master requesting bus access.
   - 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 (DMA), 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 375 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	–	–	–	–	EEMAPEN	FPRM	EPRM	SPMLOCK
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 – EEMAPEN: EEPROM Data Memory Mapping Enable**

Setting this bit enables data memory mapping of the EEPROM section. The EEPROM can then be accessed using load and store instructions.

- **Bit 2 – FPRM: Flash Power Reduction Mode**

Setting this bit enables power saving for the flash memory. If code is running from the application section, the boot loader section will be turned off, and vice versa. If access to the section that is turned off is required, the CPU will be halted for a time equal to the start-up time from the idle sleep mode.

- **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 equal 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. 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.

4.13.11 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x04	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 377.

- **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 377.

4.13.12 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	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 Byte1

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 113 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 112 for details.

4.14.2 FUSEBYTE2 – Fuse Byte2

Bit	7	6	5	4	3	2	1	0
+0x02	–	BOOTRST	TOSCSEL	–	–	–	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.

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

- **Bit 5 – TOSCSEL: 32.768kHz Timer Oscillator Pin Selection**

This fuse is used to select the pin location for the 32.768kHz timer oscillator (TOSC). This fuse is available only on devices where XTAL and TOSC pins by default are shared.

Table 4-2. TOSCSEL fuse.

TOSCSEL	Group configuration	Description
0	ALTERNATE[1]	TOSC1/2 on separate pins
1	XTAL	TOSC1/2 shared with XTAL

Note: 1. See the device datasheet for alternate TOSC position.

- **Bit 4:2 – 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 104.

Table 4-3. 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 Byte4

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 103 for details.

Table 4-4. 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 4-5. 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	R	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 104.

Table 4-6. 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 is 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 4-7. 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 System" on page 102 for details. For BOD level nominal values, see Table 9-2 on page 105.

4.14.5 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 by executing a chip erase command.

Table 4-8. 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 according to Table 4-9 on page 34. 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 4-9. 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 4-10. 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 4-11. 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 RCOSC2M – Internal 2MHz Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
0x00	RCOSC2M[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 – RCOSC2M[7:0]: Internal 2MHz Oscillator Calibration Value**

This byte contains the oscillator calibration value for the internal 2MHz oscillator. Calibration of the oscillator is performed during production test of the device. During reset this value is automatically loaded into calibration register B for the 2MHz DFLL. Refer to “CALB – DFLL Calibration register B” on page 92 for more details.

4.15.2 RCOSC2MA – Internal 2MHz Oscillator Calibration register

Bit	7	6	5	4	3	2	1	0
0x01	RCOSC2MA[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 – RCOSC2MA[7:0]: Internal 2MHz Oscillator Calibration Value**

This byte contains the oscillator calibration value for the internal 2MHz oscillator. Calibration of the oscillator is performed during production test of the device. During reset this value is automatically loaded into calibration register A for the 2MHz DFLL. Refer to “CALA – DFLL Calibration Register A” on page 92 for more details.

4.15.3 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 test 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 90 for more details.

4.15.4 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 test 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 92 for more details.

4.15.5 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 test 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 92 for more details.

4.15.6 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.

4.15.7 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.

4.15.8 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.

4.15.9 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.

4.15.10 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.

4.15.11 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.

4.15.12 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.

4.15.13 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.

4.15.14 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.

4.15.15 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.

4.15.16 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.

4.15.17 USBCAL0 – USB Calibration register 0

USBCAL0 and USBCAL1 contain the calibration value for the USB pins. Calibration is done during production to enable operation without requiring external components on the USB lines for the device. The calibration bytes are not loaded automatically into the USB calibration registers, and so this must be done from software.

Bit	7	6	5	4	3	2	1	0
0x1A	USBCAL0[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 – USBCAL0[7:0]: USB Pad Calibration byte 0**

This byte contains byte 0 of the USB pin calibration data, and must be loaded into the USB CALL register.

4.15.18 USBCAL1 – USB Pad Calibration register 1

Bit	7	6	5	4	3	2	1	0
0x1B	USBCAL1[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 – USBCAL1[7:0]: USB Pad Calibration byte 1**

This byte contains byte 1 of the USB pin calibration data, and must be loaded into the USB CALH register.

4.15.19 USBRCOSC – USB RCOSC Calibration

Bit	7	6	5	4	3	2	1	0
0x1C	USBRCOSC[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 – USBRCOSC[7:0]: 48MHz RSCOSC Calibration**

This byte contains a 48MHz calibration value for the internal 32MHz oscillator. When this calibration value is written to calibration register B for the 32MHz DFLL, the oscillator is calibrated to 48MHz to enable full-speed USB operation from internal oscillator.

Note: The COMP2 and COMP1 registers inside the DFLL32M must be set to B71B.

4.15.20 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 test of the device. The calibration bytes are not loaded automatically into the ADC calibration registers, 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.

4.15.21 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.

4.15.22 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 test 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.

4.15.23 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.

4.16 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 datasheets.

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	–	–	–	–	–	–	STARTUPDLYA[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 – STARTUPDLYx**

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 4-12. 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	–	–	–	–	–	–	–	EVSYS0LOCK
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 – EVSYS0LOCK:**

Setting this bit will lock all registers in the event system related to event channels 0 to 3 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. For details, refer to “Configuration Change Protection” on page 13.

4.17.7 AWEXLOCK – Advanced Waveform Extension Lock register

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	–	–	–	AWEXCLOCK
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 – AWEXCLOCK: Advanced Waveform Extension Lock for TCC0**

Setting this bit will lock all registers in the AWEXC module for timer/counter C0 for 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								25
+0x01	ADDR1	Address Byte 1								25
+0x02	ADDR2	Address Byte 2								25
+0x03	Reserved	–	–	–	–	–	–	–	–	
+0x04	DATA0	Data Byte 0								26
+0x05	DATA1	Data Byte 1								26
+0x06	DATA2	Data Byte 2								26
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	Reserved	–	–	–	–	–	–	–	–	
+0x09	Reserved	–	–	–	–	–	–	–	–	
+0x0A	CMD	–	CMD[6:0]							26
+0x0B	CTRLA	–	–	–	–	–	–	–	CMDEX	27
+0x0C	CTRLB	–	–	–	–	EEMAPEN	FPRM	EPRM	SPMLOCK	27
+0x0D	INTCTRL	–	–	–	–	SPMLVL[1:0]		EELVL[1:0]		28
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	STATUS	NVMBUSY	FBUSY	–	–	–	–	EELOAD	FLOAD	28
+0x10	LOCKBITS	BLBB[1:0]		BLBA[1:0]		BLBAT[1:0]		LB[1:0]		29

4.19 Register Summary - Fuses and Lock Bits

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	Reserved	–	–	–	–	–	–	–	–	
+0x01	FUSEBYTE1	WDWPER3:0]				WDPER[3:0]				30
+0x02	FUSEBYTE2	–	BOOTRST	TOSCSEL	–	–	–	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	Reserved	–	–	–	–	–	–	–	–	
+0x07	LOCKBITS	BLBB[1:0]		BLBA[1:0]		BLBAT[1:0]		LB[1:0]		34

4.20 Register Summary - Production Signature Row

Address	Auto Load	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x00	YES	RCOSC2M	RCOSC2M[7:0]								36
0x01	YES	RCOSC2MA	RCOSC2MA[7:0]								37
0x02	YES	RCOSC32K	RCOSC32K[7:0]								36
0x03	YES	RCOSC32M	RCOSC32M[7:0]								36
0x04	YES	RCOSC32MA	RCOSC32MA[7:0]								37
0x05		Reserved	–	–	–	–	–	–	–	–	
0x06		Reserved	–	–	–	–	–	–	–	–	
0x07		Reserved	–	–	–	–	–	–	–	–	
0x08	NO	LOTNUM0	LOTNUM0[7:0]								37
0x09	NO	LOTNUM1	LOTNUM1[7:0]								37
0x0A	NO	LOTNUM2	LOTNUM2[7:0]								37
0x0B	NO	LOTNUM3	LOTNUM3[7:0]								38
0x0C	NO	LOTNUM4	LOTNUM4[7:0]								38
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]								39
0x13	NO	COORDX1	COORDX1[7:0]								39
0x14	NO	COORDY0	COORDY0[7:0]								39
0x15	NO	COORDY1	COORDY1[7:0]								39
0x16		Reserved	–	–	–	–	–	–	–	–	
0x17		Reserved	–	–	–	–	–	–	–	–	
0x18		Reserved	–	–	–	–	–	–	–	–	
0x19		Reserved	–	–	–	–	–	–	–	–	
0x1A		USBCAL0	USBCAL0[7:0]								40
0x1B		USBCAL1	USBCAL1[7:0]								40
0x1C		USBRCOSC	USBRCOSC[7:0]								40

Address	Auto Load	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1D		Reserved	–	–	–	–	–	–	–	–	
0x0E		Reserved	–	–	–	–	–	–	–	–	
0x1E		Reserved	–	–	–	–	–	–	–	–	
0x20	NO	ADCACAL0	ADCACAL0[7:0]								40
0x21	NO	ADCACAL1	ADCACAL1{7:0]								41
0x22		Reserved	–	–	–	–	–	–	–	–	
0x23		Reserved	–	–	–	–	–	–	–	–	
0x24		Reserved	–	–	–	–	–	–	–	–	
0x25		Reserved	–	–	–	–	–	–	–	–	
0x26		Reserved	–	–	–	–	–	–	–	–	
0x27		Reserved	–	–	–	–	–	–	–	–	
0x28		Reserved	–	–	–	–	–	–	–	–	
0x29		Reserved	–	–	–	–	–	–	–	–	
0x2A		Reserved	–	–	–	–	–	–	–	–	
0x2B		Reserved	–	–	–	–	–	–	–	–	
0x2C		Reserved	–	–	–	–	–	–	–	–	
0x2D		Reserved	–	–	–	–	–	–	–	–	
0x2E	NO	TEMPSENSE0	TEMPSENSE0[7:0]								41
0x2F	NO	TEMPSENSE1	–	–	–	–	TEMPSENSE1[11:8]				41
0x38		Reserved	–	–	–	–	–	–	–	–	
0x39		Reserved	–	–	–	–	–	–	–	–	
0x3A		Reserved	–	–	–	–	–	–	–	–	
0x3B		Reserved	–	–	–	–	–	–	–	–	
0x3C		Reserved	–	–	–	–	–	–	–	–	
0x3D		Reserved	–	–	–	–	–	–	–	–	
0x3E		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	GPIOR[7:0]								42
+0x01	GPIOR1	GPIOR[7:0]								42
+0x02	GPIOR2	GPIOR[7:0]								42
+0x03	GPIOR3	GPIOR[7:0]								42

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]								42
+0x03	REVID	–	–	–	–	REVID[3:0]				43
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	ANAINIT	–	–	–	–	STARTUPDLYB[1:0]		STARTUPDLYA[1:0]		43
+0x08	EVSYSLOCK	–	–	–	–	–	–	–	EVSYS0LOC	43
+0x09	AWEXLOCK	–	–	–	–	–	–	–	AWEXCLOCK	44
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	

4.23 Interrupt Vector Summary – NVM Controller

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

5. DMAC - Direct Memory Access Controller

5.1 Features

- Allows high speed 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
- Two DMA channels with separate
  - transfer triggers
  - interrupt vectors
  - addressing modes
- Programmable channel priority
- From 1 byte to 16MB of data in a single transaction
  - Up to 64KB block transfers with repeat
  - 1, 2, 4, or 8 byte burst transfers
- Multiple addressing modes
  - Static
  - Incremental
  - Decremental
- Optional reload of source and destination addresses at the end of each
  - Burst
  - Block
  - Transaction
- Optional interrupt on end of transaction
- Optional connection to CRC generator for CRC on DMA data

5.2 Overview

The two-channel direct memory access (DMA) 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 two DMA channels enable up to two independent and parallel transfers.

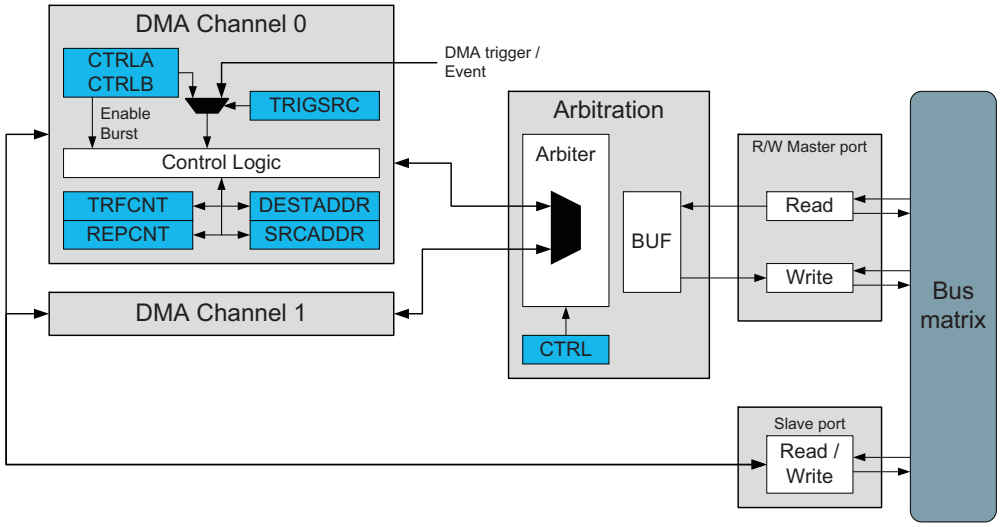
The DMA controller can move data between SRAM and peripherals, between SRAM locations and directly between peripheral registers. With access to all peripherals, the DMA controller can handle automatic transfer of data to/from communication modules. The DMA controller can also read from memory mapped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

Data transfers are done in continuous bursts of 1, 2, 4, or 8 bytes. They build block transfers of configurable size from 1 byte to 64KB. A repeat counter can be used to repeat each block transfer for single transactions up to 16MB. Source and destination addressing can be static, incremental or decremental. 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 DMA transfers.

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

To allow for continuous transfers, two channels can be interlinked so that the second takes over the transfer when the first is finished, and vice versa.

Figure 5-1. DMA Overview.



5.3 DMA Transaction

A complete DMA read and write operation between memories and/or peripherals is called a DMA 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 counter settings. Each block transfer is divided into smaller bursts.

5.3.1 Block Transfer and Repeat

The size of the block transfer is set by the block transfer count register, and can be anything from 1 byte to 64KB.

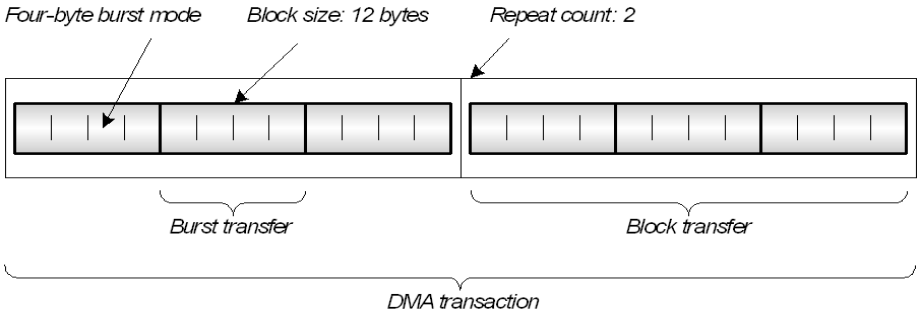
A repeat counter can be enabled to set a number of repeated block transfers before a transaction is complete. The repeat is from 1 to 255, and an unlimited repeat count can be achieved by setting the repeat count to zero.

5.3.2 Burst Transfer

Since the AVR CPU and DMA controller use the same data buses, a block transfer is divided into smaller burst transfers. The burst transfer is selectable to 1, 2, 4, or 8 bytes. This means that if the DMA acquires the data bus and a transfer request is pending, it will occupy the bus until all bytes in the burst are transferred.

A bus arbiter controls when the DMA 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, EEPROM or the external bus interface. For more details on memory access bus arbitration, refer to “Data Memory” on page 22.

Figure 5-2. DMA transaction.



5.4 Transfer Triggers

DMA transfers can be started only when a DMA transfer request is detected. A transfer request can be triggered from software, from an external trigger source (peripheral), or from an event. There are dedicated source trigger selections for each DMA 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, refer to “TRIGSRC – Trigger Source” on page 57.

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. When repeat mode is enabled, the next block transfer does not require a transfer trigger. It will start as soon as the previous block is done.

If the trigger source generates a transfer request 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 requests when one is already pending, these will be lost.

5.5 Addressing

The source and destination address for a DMA transfer can either be static or automatically incremented or decremented, with individual selections for source and destination. When address increment or decrement is used, the default behaviour is to update the address after each access. The original source and destination addresses are stored by the DMA 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

5.6 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

To allow for continuous transfer, two channels can be interlinked so that the second takes over the transfer when the first is finished, and vice versa. This leaves 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 is referred to as double buffering or chained transfers.

When double buffering is enabled for a channel pair, it is important that the two channels are configured with the same repeat count. The block sizes need not be equal, but for most applications they should be, along with the rest of the channel’s operation mode settings.

Note that the double buffering channel pairs are limited to channels 0 and 1 as the first pair and channels 2 and 3 as the second pair. However, it is possible to have one pair operate in double buffered mode while the other is left unused or operating independently.

5.8 Transfer Buffers

To avoid unnecessary bus loading when doing data transfer between memories with different access timing (for example, I/O register and external memory), the DMA controller has a four-byte buffer. Two bytes will be read from the source address and written to this buffer before a write to the destination is started.

5.9 Error detection

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

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

5.10 Software Reset

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

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

5.11 Protection

In order to ensure safe operation, some of the channel registers are protected during a transaction. When the DMA channel busy flag (CHnBUSY) is set for a channel, the user can modify only the following registers and bits:

- CTRL register
- INTFLAGS register
- TEMP registers
- CHEN, CHRST, TRFREQ, and REPEAT bits of the channel CTRL register
- TRIGSRC register

5.12 Interrupts

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

If repeat is not enabled, the transaction complete flag is set at the end of the block transfer. If unlimited repeat is enabled, the transaction complete flag is also set at the end of each block transfer.

5.13 Register Description – DMA Controller

5.13.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	RESET	–	–	–	DBUFMODE	–	PRIMODE
Read/Write	R/W	R/W	R	R	R	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – ENABLE: Enable**

Setting this bit enables the DMA controller. If the DMA 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 DMA data transfer is aborted.

- **Bit 6 – RESET: Software Reset**

Writing a one to RESET will be ignored as long as DMA is enabled (ENABLE = 1). This bit can be set only when the DMA controller is disabled (ENABLE = 0).

- **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 – DBUFMODE: Double Buffer Mode**

This bit enables the double buffer mode.

- **Bit 1 – Reserved**

This bit is unused and reserved for future use. For compatibility with future devices, always write this bits to zero when this register is written.

- **Bit 0 – PRIMODE: Channel Priority Mode**

This bit determines the internal channel priority according to Table 5-1.

Table 5-1. Channel priority settings

PRIMODE	Group Configuration	Description
0	RR01	Round robin
1	CH01	Channel0 has priority

5.13.2 INTFLAGS – Interrupt Status register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	CH1ERRIF	CH0ERRIF	–	–	CH1TRNFIF	CH0TRNFIF
Read/Write	R	R	R/W	R/W	R	R	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 – CHnERRIF[1:0]: Channel n Error Interrupt Flag**

If an error condition is detected on DMA channel n, the CHnERRIF flag will be set. Writing a one to this bit location will clear the flag.

- **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 – CHnTRNFIF[1:0]: Channel n Transaction Complete Interrupt Flag**

When a transaction on channel n has been completed, the CHnTRFIF flag will be set. If unlimited repeat count is enabled, this flag is read as one after each block transfer. Writing a one to this bit location will clear the flag.

5.13.3 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	CH1BUSY	CH0BUSY	–	–	CH1PEND	CH0PEND
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 5:4 – CHnBUSY[1:0]: Channel Busy**

When channel n starts a DMA transaction, the CHnBUSY flag will be read as one. This flag is automatically cleared when the DMA channel is disabled, when the channel n transaction complete interrupt flag is set, or if the DMA channel n error interrupt flag 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 – CHnPEND[1:0]: Channel Pending**

If a block transfer is pending on DMA channel n, 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 TEMPL – Temporary register Low

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 register 0**

This register is used when reading 16- and 24-bit registers in the DMA controller. Byte 1 of the 16/24-bit register is stored here when it is written by the CPU. Byte 1 of the 16/24-bit register is stored when byte 0 is read by the CPU. This register can also be read and written from the user software.

Reading and writing 16- and 24-bit registers requires special attention. For details, refer to “Accessing 16-bit Registers” on page 13.

5.13.5 TEMPH – Temporary Register High

Bit	7	6	5	4	3	2	1	0
+0x07	TEMP[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 – TEMP[15:8]: Temporary Register**

This register is used when reading and writing 24-bit registers in the DMA controller. Byte 2 of the 24-bit register is stored when it is written by the CPU. Byte 2 of the 24-bit register is stored here when byte 1 is read by the CPU. This register can also be read and written from the user software.

Reading and writing 24-bit registers requires special attention. For details, refer to “Accessing 16-bit Registers” on page 13.

5.14 Register Description – DMA Channel

5.14.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	RESET	REPEAT	TRFREQ	–	SINGLE	BURSTLEN[1:0]	
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 – ENABLE: Channel Enable**

Setting this bit enables the DMA channel. This bit is automatically cleared when the transaction is completed. If the DMA channel is enabled and this bit is written to zero, the CHEN bit is not cleared until the internal transfer buffer is empty and the DMA transfer is aborted.

- **Bit 6 – RESET: Software Reset**

Setting this bit will reset the DMA channel. It can only be set when the DMA channel is disabled (CHEN = 0). Writing a one to this bit will be ignored as long as the 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. In repeat mode, this bit is cleared by hardware at the beginning of the last block transfer. The REPCNT register should be configured before setting the REPEAT bit.

- **Bit 4 – TRFREQ: Transfer Request**

Setting this bit requests a data transfer on the DMA channel. This bit is automatically cleared at the beginning of the data transfer. Writing this bit does not have any effect unless the 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 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 1:0 – BURSTLEN[1:0]: Burst Mode**

These bits decide the DMA channel burst mode according to Table 5-2 on page 54. These bits cannot be changed if the channel is busy.

Table 5-2. DMA channel burst mode

BURSTLEN[1:0]	Group Configuration	Description
00	1BYTE	1 byte burst mode
01	2BYTE	2 bytes burst mode
10	4BYTE	4 bytes burst mode
11	8BYTE	8 bytes burst mode

Table 5-3. Summary of triggers, transaction complete flag and channel disable according to DMA channel configuration.

REPEAT	SINGLE	REPCNT	Trigger	Flag Set After	Channel Disabled After
0	0	0	Block	1 block	1 block
0	0	1	Block	1 block	1 block
0	0	n > 1	Block	1 block	1 block
0	1	0	BURSTLEN	1 block	1 block
0	1	1	BURSTLEN	1 block	1 block
0	1	n > 1	BURSTLEN	1 block	1 block
1	0	0	Block	Each block	Each block
1	0	1	Transaction	1 block	1 block
1	0	n > 1	Transaction	n blocks	n blocks
1	1	0	BURSTLEN	Each block	Never
1	1	1	BURSTLEN	1 block	1 block
1	1	n > 1	BURSTLEN	n blocks	n blocks

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: Channel Busy**

When the DMA channel starts a DMA transaction, the CHBUSY flag will be read as one. This flag is automatically cleared when the DMA 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 DMA 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 DMA channel, the ERRIF flag will be set and the optional interrupt is generated. Since the DMA channel error interrupt shares the interrupt address with the DMA 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: Channel n Transaction Complete Interrupt Flag**

When a transaction on the DMA channel has been completed, the TRNIF flag will be set and the optional interrupt is generated. When repeat is not enabled, the transaction is complete and TRNIFR is set after the block transfer. When unlimited repeat is enabled, TRNIF is also set after each block transfer.

Since the DMA channel transaction n complete interrupt shares the interrupt address with the DMA 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 DMA channel transfer errors and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. 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 DMA channel transaction completes and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will trigger for the conditions when TRNIF is set.

5.14.3 ADDRCTRL – Address Control register

Bit	7	6	5	4	3	2	1	0
+0x02	SRCRELOAD[1:0]		SRCDIR[1:0]		DESTRELOAD[1:0]		DESTDIR[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 – SRCRELOAD[1:0]: Channel Source Address Reload**

These bits decide the DMA channel source address reload according to Table 5-4. A write to these bits is ignored while the channel is busy.

Table 5-4. DMA channel source address reload settings

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

- **Bit 5:4 – SRCDIR[1:0]: Channel Source Address Mode**

These bits decide the DMA channel source address mode according to Table 5-5. These bits cannot be changed if the channel is busy.

Table 5-5. DMA channel source address mode settings.

SRCDIR[1:0]	Group Configuration	Description
00	FIXED	Fixed
01	INC	Increment
10	DEC	Decrement
11	-	Reserved

- **Bit 3:2 – DESTRELOAD[1:0]: Channel Destination Address Reload**

These bits decide the DMA channel destination address reload according to Table 5-6. These bits cannot be changed if the channel is busy.

Table 5-6. DMA channel destination address reload settings

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

- **Bit 1:0 – DESTDIR[1:0]: Channel Destination Address Mode**

These bits decide the DMA channel destination address mode according to Table 5-7. These bits cannot be changed if the channel is busy.

Table 5-7. DMA channel destination address mode settings

DESTDIR[1:0]	Group Configuration	Description
00	FIXED	Fixed
01	INC	Increment
10	DEC	Decrement
11	-	Reserved

5.14.4 TRIGSRC – Trigger Source

Bit	7	6	5	4	3	2	1	0
+0x03	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]: Channel Trigger Source Select**

These bits select which trigger source is used for triggering a transfer on the DMA channel. A zero value means that the trigger source is disabled. For each trigger source, the value to put in the TRIGSRC register is the sum of the module's or peripheral's base value and the offset value for the trigger source in the module or peripheral. Table 5-8 on page 57 shows the base value for all modules and peripherals. Table 5-9 on page 58 to Table 5-11 on page 58 shows the offset value for the trigger sources in the different modules and peripheral types. 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 DMA request will be lost. Since a DMA 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 how requests are generated and cleared.

Table 5-8. DMA trigger source base values for all modules and peripherals.

TRIGSRC Base Value	Group Configuration	Description
0x00	OFF	Software triggers only
0x01	SYS	Event system DMA triggers base value
0x04	AES	AES DMA trigger value
0x10	ADCA	ADCA DMA trigger value
0x40	TCC0	Timer/counter C0 DMA triggers base value
0x46	TCC1	Timer/counter C1 triggers base value
0x4A	SPIC	SPI C DMA trigger value
0x4B	USARTC0	USART C0 DMA triggers base value
0x60	TCD0	Timer/counter D0 DMA triggers base value
0x6A	SPID	SPI D DMA triggers value
0x6B	USARTD0	USART D0 DMA triggers base value
0x80	TCE0	Timer/counter E0 DMA triggers base value
0x8B	USARTE0	USART E0 DMA triggers base value
0xA0	TCF0	Timer/counter F0 DMA triggers base value
0xAB	USARTF0	USART F0 DMA triggers base value

Table 5-9. DMA trigger source offset values for event system triggers.

TRGSRC Offset Value	Group Configuration	Description
+0x00	CH0	Event channel 0
+0x01	CH1	Event channel 1
+0x02	CH2	Event channel 2

Table 5-10. DMA trigger source offset values for timer/ counter triggers.

TRGSRC 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 0.

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

TRGSRC Offset Value	Group Configuration	Description
0x00	RXC	Receive complete
0x01	DRE	Data register empty

The group configuration is the “base_offset;” for example, TCC1_CCA for the timer/counter C1 CC channel A the transfer trigger.

5.14.5 TRFCNTL – Channel 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 DMA 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
+0x04	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	0

- **Bit 7:0 – TRFCNT[7:0]: Channel n Block Transfer Count low byte**

These bits hold the LSB of the 16-bit block transfer count.

The default value of this register is 0x1. If a user writes 0x0 to this register and fires a DMA trigger, DMA will be doing 0xFFFF transfers.

5.14.6 TRFCNTH – Channel Block Transfer Count register High

Reading and writing 16-bit values requires special attention. For details, refer to “Accessing 16-bit Registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x05	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	0

- **Bit 7:0 – TRFCNT[15:8]: Channel n Block Transfer Count high byte**

These bits hold the MSB of the 16-bit block transfer count.

The default value of this register is 0x1. If a user writes 0x0 to this register and fires a DMA trigger, DMA will be doing 0xFFFF transfers.

5.14.7 REPCNT – Repeat Counter register

Bit	7	6	5	4	3	2	1	0
+0x06	REPCNT[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

REPCNT counts how many times a block transfer is performed. For each block transfer, this register will be decremented.

When repeat mode is enabled (see REPEAT bit in “ADDRCTRL – Address Control register” on page 55), this register is used to control when the transaction is complete. The counter is decremented after each block transfer if the DMA has to serve a limited number of repeated block transfers. When repeat mode is enabled, the channel is disabled when REPCNT reaches zero and the last block transfer is completed. Unlimited repeat is achieved by setting this register to zero.

5.14.8 SRCADDR0 – Source Address 0

SRCADDR0, SRCADDR1, and SRCADDR2 represent the 24-bit value SRCADDR, which is the DMA channel source address. SRCADDR2 is the most significant byte in the register. SRCADDR may be automatically incremented or decremented based on settings in the SRCDIR bits in “ADDRCTRL – Address Control register” on page 55.

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]: Channel Source Address byte 0**

These bits hold byte 0 of the 24-bit source address.

5.14.9 SRCADDR1 – Channel Source Address 1

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]: Channel Source Address byte 1**

These bits hold byte 1 of the 24-bit source address.

5.14.10 SRCADDR2 – Channel Source Address 2

Reading and writing 24-bit values require special attention. For details, refer to “Accessing 24- and 32-bit Registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x0A	SRCADDR[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 – SRCADDR[23:16]: Channel Source Address byte 2**

These bits hold byte 2 of the 24-bit source address.

5.14.11 DESTADDR0 – Channel Destination Address 0

DESTADDR0, DESTADDR1, and DESTADDR2 represent the 24-bit value DESTADDR, which is the DMA channel destination address. DESTADDR2 holds the most significant byte in the register. DESTADDR may be automatically incremented or decremented based on settings in the DESTDIR bits in “ADDRCTRL – Address Control register” on page 55.

Bit	7	6	5	4	3	2	1	0
+0x0C	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]: Channel Destination Address byte 0**

These bits hold byte 0 of the 24-bit source address.

5.14.12 DESTADDR1 – Channel Destination Address 1

Bit	7	6	5	4	3	2	1	0
+0x0D	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]: Channel Destination Address byte 1**

These bits hold byte 1 of the 24-bit source address.

5.14.13 DESTADDR2 – Channel Destination Address 2

Reading and writing 24-bit values require special attention. For details, refer to “Accessing 24- and 32-bit Registers” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x0E	DESTADDR[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 – DESTADDR[23:16]: Channel Destination Address byte 2**

These bits hold byte 2 of the 24-bit source address.

5.15 Register Summary – DMA Controller

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	ENABLE	RESET	–	–	–	DBUFMODE	–	PRIMODE	51
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	INTFLAGS	–	–	CH1ERRIF	CH0ERRIF	–	–	CH1TRNFIF	CH0TRNFIF	51
+0x04	STATUS	–	–	CH1BUSY	CH0BUSY	–	–	CH1PEND	CH0PEND	52
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	TEMPL	TEMP[7:0]								52
+0x07	TEMPH	TEMP[15:8]								53
+0x10	CH0 Offset	Offset address for DMA Channel 0								
+0x20	CH1 Offset	Offset address for DMA Channel 1								
+0x30	Reserved	–	–	–	–	–	–	–	–	
+0x40	Reserved	–	–	–	–	–	–	–	–	

5.16 Register Summary – DMA 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[1:0]		53
+0x01	CTRLB	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]		54
+0x02	ADDCTRL	SRCRELOAD[1:0]		SRCDIR[1:0]		DESTRELOAD[1:0]		DESTDIR[1:0]		55
+0x03	TRIGSRC	TRIGSRC[7:0]								57
+0x04	TRFCNTL	TRFCNT[7:0]								58
+0x05	TRFCNTH	TRFCNT[15:8]								59
+0x06	REPCNT	REPCNT[7:0]								59
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	SRCADDR0	SRCADDR[7:0]								59
+0x09	SRCADDR1	SRCADDR[15:8]								60
+0x0A	SRCADDR2	SRCADDR[23:16]								60
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	DESTADDR0	DESTADDR[7:0]								60
+0x0D	DESTADDR1	DESTADDR[15:8]								61
+0x0E	DESTADDR2	DESTADDR[23:16]								61
+0x0F	Reserved	–	–	–	–	–	–	–	–	

5.17 DMA Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	CH0_vect	DMA controller channel 0 interrupt vector
0x02	CH1_vect	DMA controller channel 1 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 DMA controller independent operation
  - 100% predictable signal timing
  - Short and guaranteed response time
- Four 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
  - Digital filtering of I/O pin state
- Works in active mode and idle sleep mode

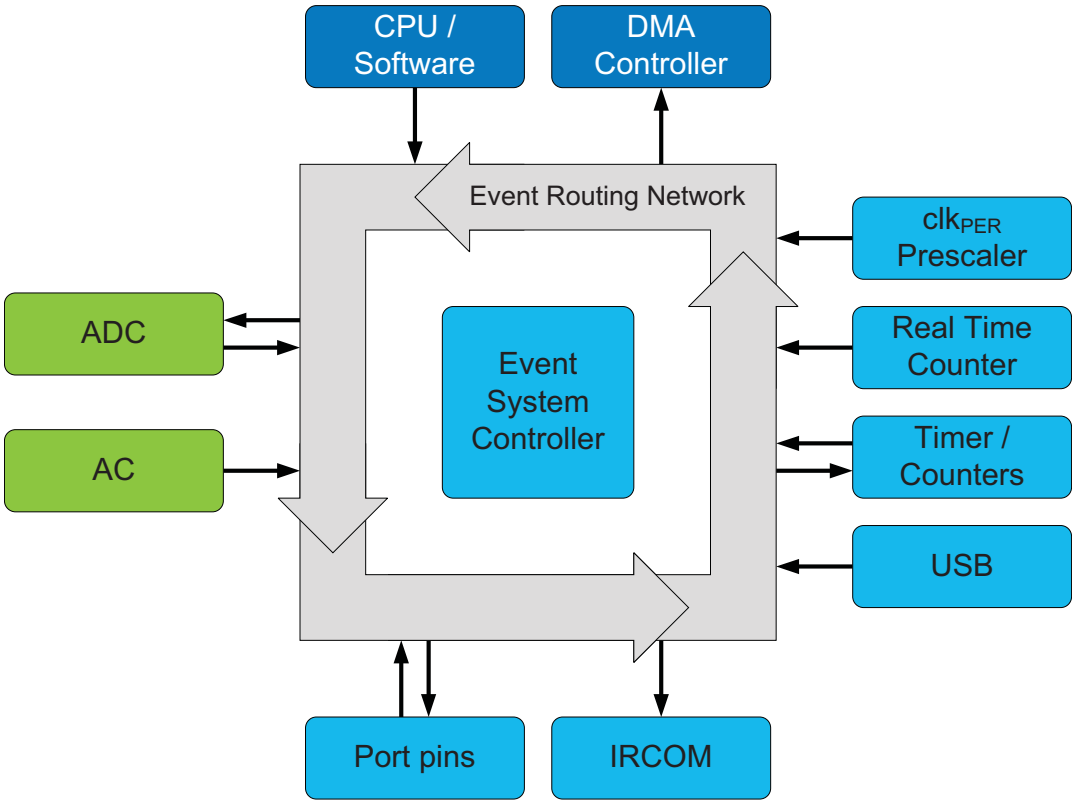
6.2 Overview

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 DMA controller resources, and is thus a powerful tool for reducing the complexity, size and execution time of application code. It also allows for synchronized timing of actions in several peripheral modules.

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 64 shows a basic diagram of all connected peripherals. The event system can directly connect together analog converters, analog comparators, I/O port pins, the real-time counter, timer/counters, IR communication module (IRCOM) and USB interface. It can also be used to trigger DMA transactions (DMA controller). Events can also be generated from software and the peripheral clock.

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



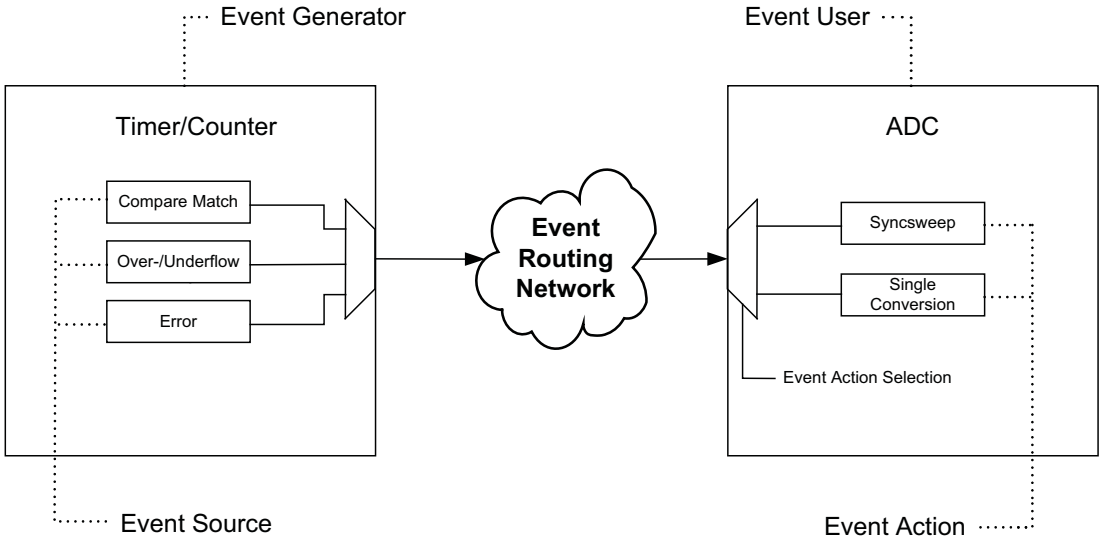
The event routing network consists of four software-configurable multiplexers that control how events are routed and used. These are called event channels, and allow for up to four parallel event configurations and routings. The maximum routing latency is two peripheral clock cycles. The event system works in both active mode and idle sleep mode.

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 two main types of events: signaling events and 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.

6.3.1 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.

6.3.2 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.

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 66.

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.

6.3.3 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.

6.3.4 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 66 shows the different events, how they can be manually generated, and how they are decoded.

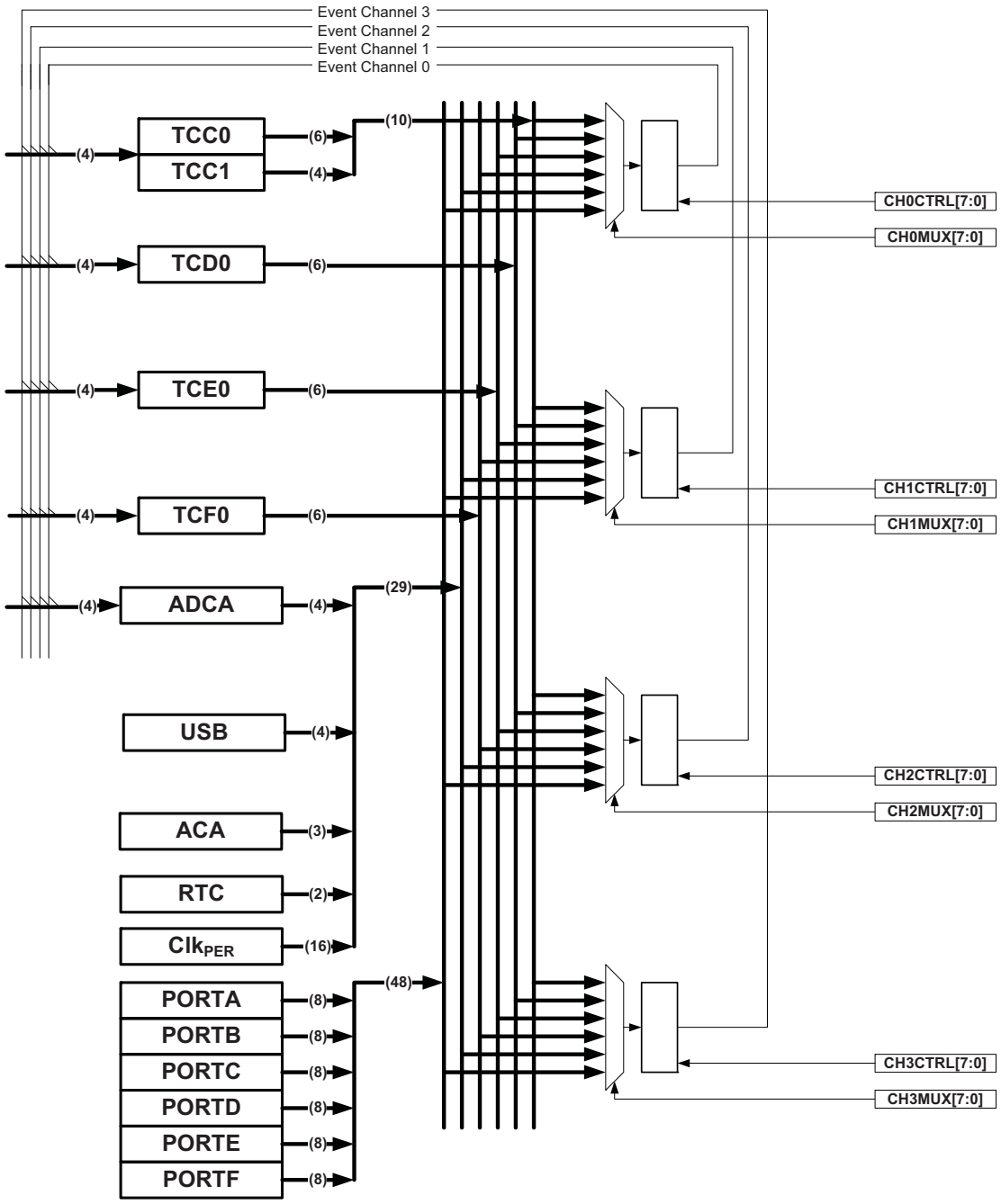
Table 6-1. Manually generated events and decoding of events.

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

6.4 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 67.

Figure 6-3. Event routing network.



Four multiplexers means that it is possible to route up to four 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.

6.5 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 DMA controller load or software revisions.

6.6 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 cycles before it is accepted. This is primarily intended for pin change events.

6.7 Quadrature Decoder

The event system includes one quadrature decoder (QDEC), 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 6-2 summarizes which quadrature decoder data events are available, how they are decoded, and how they can be generated. The QDEC and related features, control and status registers are available for event channel 0.

Table 6-2. 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

6.7.1 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 6-4. Quadrature signals from a rotary encoder.

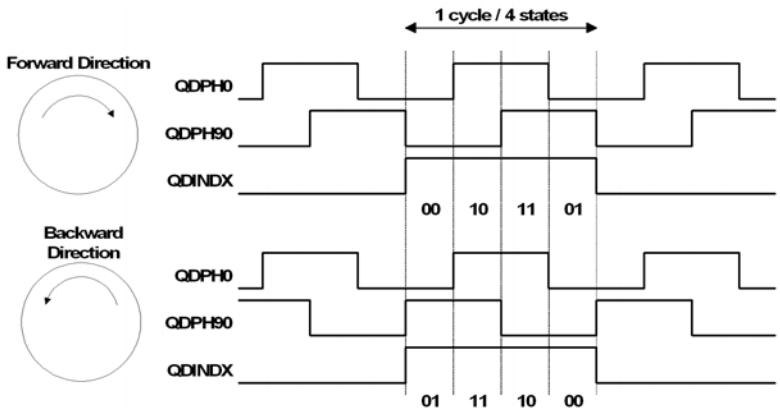


Figure 6-4 shows typical quadrature signals from a rotary encoder. The signals QDPH0 and QDPH90 are the two quadrature signals. When QDPH90 leads QDPH0, the rotation is defined as positive or forward. When QDPH0 leads QDPH90, the rotation is defined as negative or reverse. The concatenation of the two phase signals is called the quadrature state or the phase state.

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

6.7.2 QDEC Setup

For a full QDEC setup, the following is required:

- Tho 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 up a QDEC index (QINDX).
    2. Select a third pin for QINDX input.
    3. Set the pin direction for QINDX as input.
    4. Set the pin configuration for QINDX to sense both edges.
    5. Select QINDX as a multiplexer input for event channel n+1
    6. Set the quadrature index enable bit in event channel n+1.
    7. Select the index recognition mode for event channel n+1.
7. Set quadrature decoding as the event action for a timer/counter.
8. 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.

6.8 Register Description

6.8.1 CHnMUX – Event Channel n Multiplexer register

Bit	7	6	5	4	3	2	1	0
	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 6-3. 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 6-3. CHnMUX[7:0] 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	0		USB start of frame on CH0(2) USB error on CH1(2) USB overflow on CH2(2) USB setup on CH3(2)
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	0	1	1		(Reserved)
0001	0	1	X	X		(Reserved)
0001	1	X	X	X		(Reserved)
0010	0	0	0	0	ADCA_CH0	ADCA
0010	0	0	0	1		(Reserved)
0010	0	0	1	X		(Reserved)
0010	0	1	X	X		(Reserved)

CHnMUX[7:4]	CHnMUX[3:0]				Group Configuration	Event Source
0010	1	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	n			PORTB_PINn(1)	PORTB pin n (n= 0, 1, 2 ... or 7)
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	0	n			PORTE_PINn(1)	PORTE pin n (n= 0, 1, 2 ... or 7)
0111	1	n			PORTF_PINn(1)	PORTF pin n (n= 0, 1, 2 ... or 7)
1000	M				PRESCALER_M	Clk_{PER} divide by 2^M (M=0 to 15)
1001	X	X	X	X		(Reserved)
1010	X	X	X	X		(Reserved)
1011	X	X	X	X		(Reserved)
1100	0	E			See Table 6-4	Timer/counter C0 event type E
1100	1	E			See Table 6-4	Timer/counter C1 event type E
1101	0	E			See Table 6-4	Timer/counter D0 event type E
1111	1	X	X	X		(Reserved)
1110	0	E			See Table 6-4	Timer/counter E0 event type E
1111	1	X	X	X		(Reserved)
1111	0	E			See Table 6-4	Timer/counter F0 event type E
1111	1	X	X	X		(Reserved)

Notes: 1. The description of how the ports generate events is described in "Port Event" on page 130.
2. The different USB events can be selected for only event channel, 0 to 3.

Table 6-4. Timer/counter events

T/C Event E			Group Configuration	Event Type
0	0	0	TCxn_OVF	Over/Underflow (x = C, D, E or F) (n= 0 or 1)
0	0	1	TCxn_ERR	Error (x = C, D, E or F) (n= 0 or 1)
0	1	X	–	(Reserved)
1	0	0	TCxn_CCA	Capture or compare A (x = C, D, E or F) (n= 0 or 1)
1	0	1	TCxn_CCB	Capture or compare B (x = C, D, E or F) (n= 0 or 1)
1	1	0	TCxn_CCC	Capture or compare C (x = C, D, E or F) (n= 0)
1	1	1	TCxn_CCD	Capture or compare D (x = C, D, E or F) (n= 0)

6.8.2 CHnCTRL – Event Channel n Control register

Bit	7	6	5	4	3	2	1	0
	–	QDIRM[1:0](1)		QDIEN(1)	QDEN(1)	DIGFILT[2:0]		
	–	–	–	–	–	DIGFILT[2:0]		
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

Note: 1. Only available for CH0CTRL and CH2CTRL. These bits are reserved in CH1CTRL and CH3CTRL.

- **Bit 7 – Reserved**

This bit is reserved and will always be read as zero. For compatibility with future devices, always write this bit to zero when this register is written.

- **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 6-5. These bits should only be set when a quadrature encoder with a connected index signal is used.These bits are available only for CH0CTRL and CH2CTRL.

Table 6-5. 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 and CH2CTRL.

- **Bit 3 – QDEN: Quadrature Decode Enable**

Setting this bit enables QDEC operation.

This bit is available only for CH0CTRL and CH2CTRL.

- **Bit 2:0 – DIGFILT[2:0]: Digital Filter Coefficient**

These bits define the length of digital filtering used, according to Table 6-6 on page 72. 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 peripheral clock cycles defined by DIGFILT.

Table 6-6. Digital filter coefficient values .

DIGFILT[2:0]	Group Configuration	Description
000	1SAMPLE	One sample
001	2SAMPLES	Two samples
010	3SAMPLES	Three samples

Table 6-6. Digital filter coefficient values (Continued).

DIGFILT[2:0]	Group Configuration	Description
011	4SAMPLES	Four samples
100	5SAMPLES	Five samples
101	6SAMPLES	Six samples
110	7SAMPLES	Seven samples
111	8SAMPLES	Eight samples

6.8.3 STROBE – 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

6.8.4 DATA – 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 – Strobe register” on page 73.

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

6.9 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]								70
+0x01	CH1MUX	CH1MUX[7:0]								70
+0x02	CH2MUX	CH2MUX[7:0]								70
+0x03	CH3MUX	CH3MUX[7:0]								70
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	CH0CTRL	–	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			72
+0x09	CH1CTRL	–	–	–	–	–	DIGFILT[2:0]			72
+0x0A	CH2CTRL	–	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			72
+0x0B	CH3CTRL	–	–	–	–	–	DIGFILT[2:0]			72
+0x0C	Reserved	–	–	–	–	–	–	–	–	
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10	STROBE	STROBE[7:0]								73
+0x11	DATA	DATA[7:0]								73

7. System Clock and Clock Options

7.1 Features

- Fast start-up time
- Safe run-time clock switching
- Internal oscillators:
  - 32MHz run-time calibrated and tunable oscillator
  - 2MHz run-time calibrated oscillator
  - 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 1x to 31x multiplication
  - Lock detector
- Clock prescalers with 1x to 2048x division
- Fast peripheral clocks running at 2 and 4 times the CPU clock
- Automatic run-time calibration of internal oscillators
- External oscillator and PLL lock failure detection with optional non-maskable interrupt

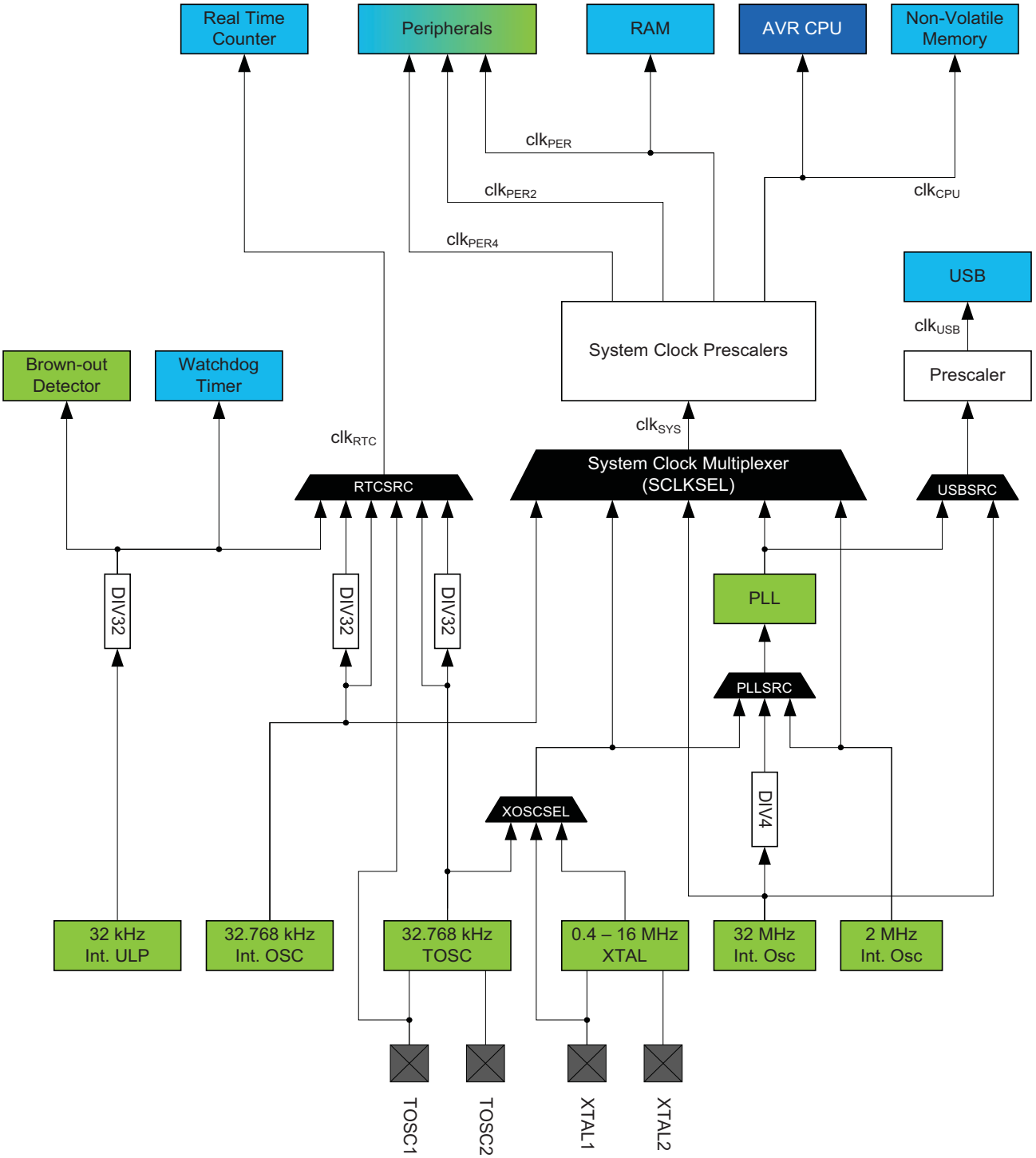
7.2 Overview

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 internal oscillator. During normal operation, the system clock source and prescalers can be changed from software at any time.

Figure 7-1 on page 76 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 94.

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



7.3 Clock Distribution

Figure 7-1 on page 76 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 and USB 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 2x and peripheral 4x 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.

7.3.6 USB Clock - Clk_{USB}

The USB device module requires a 12MHz or 48MHz clock. It has a separate clock source selection in order to avoid system clock source limitations when USB is used.

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 internal oscillator. The other clock sources, DFLLs 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, see "RTCCTRL – RTC Control register" on page 85 for details. 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, see “RTCCTRL – RTC Control register” on page 85 for details.

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. The production signature row contains 48 MHz calibration values intended used when the oscillator is used a full-speed USB clock source.

7.4.1.4 2MHz Run-time Calibrated Oscillator

The 2MHz run-time 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. A DFLL can be enabled for automatic run-time calibration of the oscillator to compensate for temperature and voltage drift and optimize the oscillator accuracy.

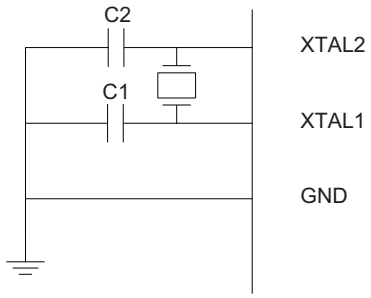
7.4.2 External Clock Sources

The XTAL1 and XTAL2 pins can be used to drive an external oscillator, either a quartz crystal or a ceramic resonator. XTAL1 can be used as input for an external clock signal. The TOSC1 and TOSC2 pins is 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.

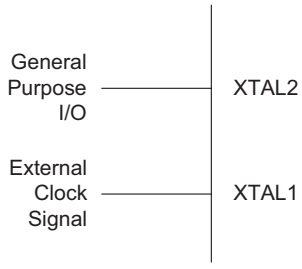


Two capacitors, C1 and C2, may be added to match the required load capacitance for the connected crystal.

7.4.2.2 External Clock Input

To drive the device from an external clock source, XTAL1 must be driven as shown in Figure 7-3 on page 78. In this mode, XTAL2 can be used as a general I/O pin.

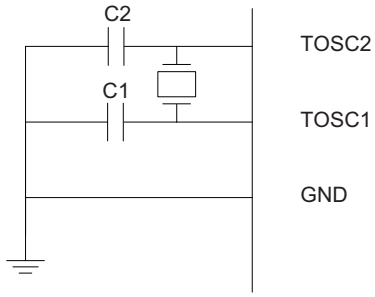
Figure 7-3. External clock drive configuration.



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 79. 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.



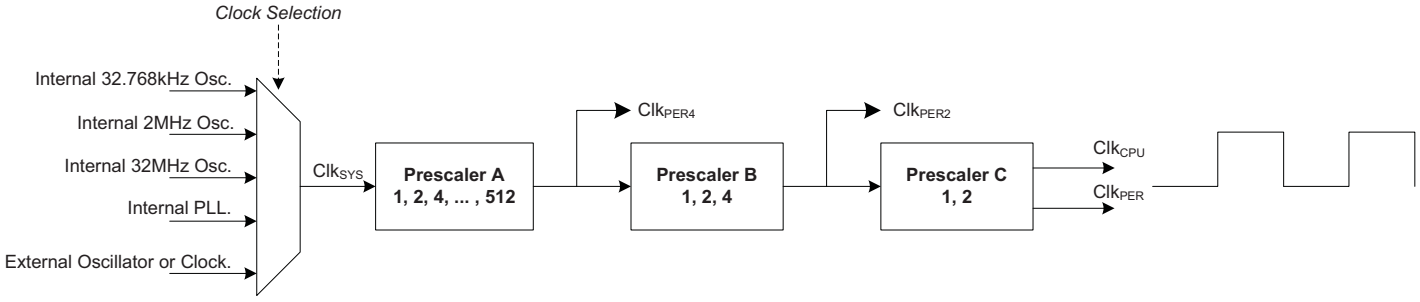
Two capacitors, C1 and C2, may be added to match the required load capacitance for the connected crystal. For details on recommended TOSC characteristics and capacitor load, refer to device datasheets.

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.

$$f_{OUT} = f_{IN} \cdot \text{PLL\_FAC}$$

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

- 2MHz 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.

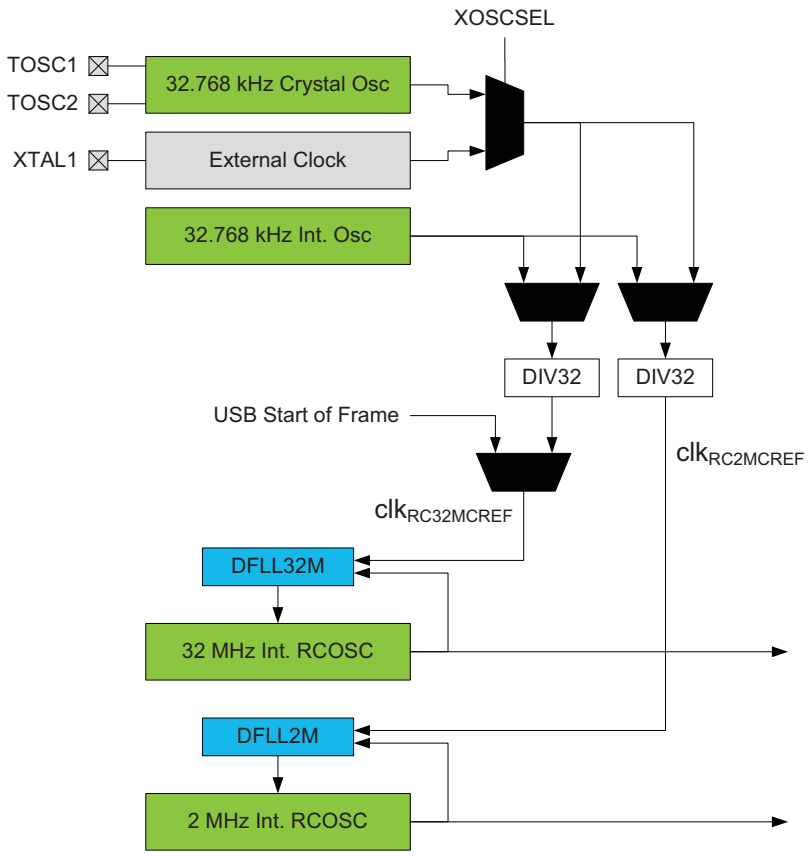
7.7 DFLL 2MHz and DFLL 32MHz

Two built-in digital frequency locked loops (DFLLs) can be used to improve the accuracy of the 2MHz and 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
- USB start of frame

The DFLLs divide the oscillator reference clock by 32 to use a 1.024kHz reference. The reference clock is individually selected for each DFLL, as shown on Figure 7-6 on page 81.

Figure 7-6. DFLL reference clock selection.



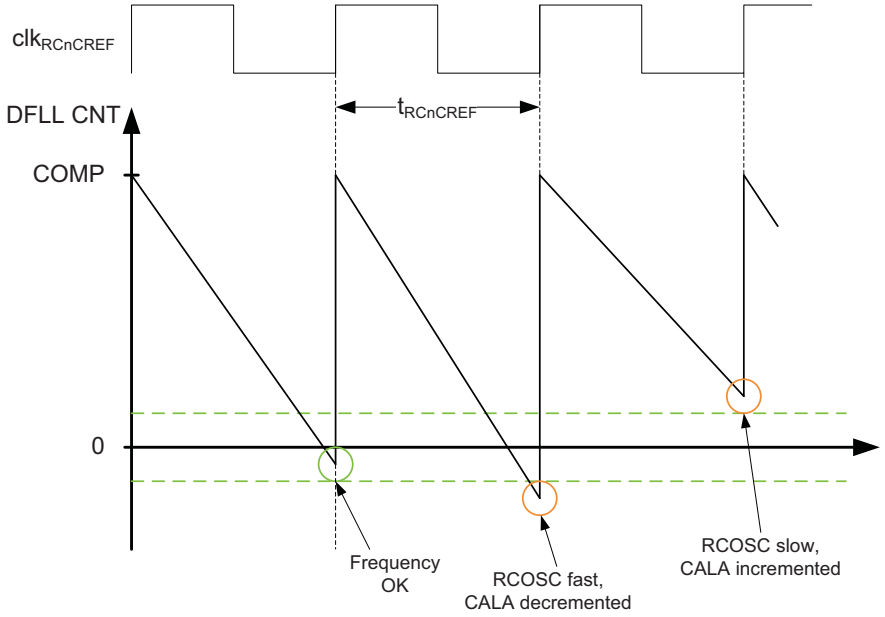
The ideal counter value representing the frequency ratio between the internal oscillator and a 1.024kHz reference clock is loaded into the DFLL oscillator compare register (COMP) during reset. For the 32MHz oscillator, this register can be written from software to make the oscillator run at a different frequency or when the ratio between the reference clock and the oscillator is different (for example when the USB start of frame is used). The 48MHz calibration values must be read from the production signature row and written to the 32MHz CAL register before the DFLL is enabled with USB SOF as reference source.

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

$$COMP = hex\left(\frac{f_{OSC}}{f_{RCnCREF}}\right)$$

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 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 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	–	Reserved
110	–	Reserved
111	–	Reserved

7.9.2 PSCTRL – Prescaler register

This register is protected by the configuration change protection mechanism. For details, refer to “Configuration Change Protection” on page 13.

Bit	7	6	5	4	3	2	1	0
+0x01	–	PSADIV[4:0]					PSBCDIV	
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
10101	–	Reserved
10111	–	Reserved
11001	–	Reserved
11011	–	Reserved
11101	–	Reserved
11111	–	Reserved

- **Bit 1:0 – PSBCDIV: Prescaler B and C Division Factors**

These bits define the division ratio of the clock prescalers B and C according to Table 7-3. 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 79 fore 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/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 – RTCSRC[2:0]: RTC Clock Source**

These bits select the clock source for the real-time counter according to Table 7-4.

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.

7.9.5 USBSCTRL – USB Control register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	USBPSDIV[2:0]			USBSRC[1:0]		USBSEN
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 – USBPSDIV[2:0]: USB Prescaler Division Factor**

These bits define the division ratio of the USB clock prescaler according to Table 7-5. These bits are locked as long as the USB clock source is enabled.

Table 7-5. USB prescaler division factor.

USBPSDIV[2:0]	Group Configuration	Description
000	1	No division
001	2	Divide by 2
010	4	Divide by 4
011	8	Divide by 8
100	16	Divide by 16
101	32	Divide by 32
110	—	Reserved
111	—	Reserved

- **Bit 2:1 – USBSRC[1:0]: USB Clock Source**

These bits select the clock source for the USB module according to Table 7-6.

Table 7-6. USB clock source.

USBSRC[1:0]	Group Configuration	Description
00	PLL	PLL
01	RC32M	32MHz internal oscillator(1)

Note: 1. The 32MHz internal oscillator must be calibrated to 48MHz before selecting this as source for the USB device module. Refer to "DFLL 2MHz and DFLL 32MHz" on page 80.

- **Bit 0 – USBSEN: USB Clock Source Enable**

Setting this bit enables the selected clock source for the USB device module.

7.10 Register Description – Oscillator

7.10.1 CTRL – Oscillator Control register

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

- **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 – 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 87.

- **Bit 3 – XOSCEN: External Oscillator Enable**

Setting this bit enables the selected external clock source. Refer to “XOSCCTRL – XOSC Control register” on page 88 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 87.

- **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 87.

- **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 87.

- **Bit 0 – RC2MEN: 2MHz Internal Oscillator Enable**

Setting this bit enables the 2MHz 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 87.

By default, the 2MHz 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	–	–	–	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 7:5 – 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 – 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 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[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 7-7.

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

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

- **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 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 7-8 on page 89 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 87. This configuration cannot be changed.

Table 7-8. External oscillator selection and start-up time..

XOSCSEL[3:0]	Group Configuration	Selected Clock Source	Start-up Time
0000	EXTCLK(3)	External Clock	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	x	x	x	x	x	x	x	x

- **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 7-9.

Table 7-9. PLL clock source.

PLLSRC[1:0]	Group Configuration	PLL Input Source
00	RC2M	2MHz internal oscillator
01	—	Reserved
10	RC32M	32MHz internal oscillator
11	XOSC	External clock source(1)

Notes: 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]		RC2MCREF
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 2:1 – RC32MCREF[1:0]: 32MHz Oscillator Calibration Reference**

These bits are used to select the calibration source for the 32MHz DFLL according to the Table 7-10. 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 7-10. 32MHz oscillator reference selection.

RC32MCREF[1:0]	Group Configuration	Description
00	RC32K	32.768kHz internal oscillator
01	XOSC32	32.768kHz crystal oscillator on TOSC
10	USBSOF	USB start of frame
11	–	Reserved

- **Bit 0 – RC2MCREF: 2MHz Oscillator Calibration Reference**

This bit is used to select the calibration source for the 2MHz DFLL. By default, this bit is zero and the 32.768kHz internal oscillator is selected. If this bit is set to one, the 32.768kHz crystal oscillator on TOSC is selected as the reference. This bit 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.

7.11 Register Description – DFLL32M/DFLL2M

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/W	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/W	R/W	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 Register value byte 2**

These bits hold byte 2 of the 16-bit compare register.

Table 7-11. Nominal DFLL32M COMP values for different output frequencies.

Oscillator Frequency (MHz)	COMP Value ($Clk_{RCnCREF}$ = 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]			83
+0x01	PSCTRL	–	PSADIV[4:0]					PSBCDIV[1:0]		83
+0x02	LOCK	–	–	–	–	–	–	–	LOCK	85
+0x03	RTCCTRL	–	–	–	–	RTCSRC[2:0]			RTCEN	85
+0x04	USBSCTR	–	–	USBPSDIV[2:0]			USBSRC[1:0]		USBSEN	85
+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	–	–	–	PLLEN	XOSCEN	RC32KEN	R32MEN	RC2MEN	87
+0x01	STATUS	–	–	–	PLLRDY	XOSCRDY	RC32KRD	R32MRDY	RC2MRDY	87
+0x02	XOSCCTR	FRQRANGE[1:0]		X32KLPM	XOSCPW	XOSCSEL[3:0]				88
+0x03	XOSCFAIL	–	–	–	–	PLLFDIF	PLLFDEN	XOSCFDIF	XOSCFDEN	89
+0x04	RC32KCAL	RC32KCAL[7:0]								90
+0x05	PLLCTRL	PLLSRC[1:0]		–	PLLFAC[4:0]					90
+0x06	DFLLCTRL	–	–	–	–	–	RC32MCREF[1:0]		RC2MCREF	90
+0x07	Reserved	–	–	–	–	–	–	–	–	

7.14 Register Summary - DFLL32M/DFLL2M

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	–	–	–	ENABLE	91
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	CALA	–	CALA[6:0]							92
+0x03	CALB	–	–	CALB[5:0]						92
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	COMP1	COMP[7:0]								92
+0x06	COMP2	COMP[15:8]								93
+0x07	Reserved	–	–	–	–	–	–	–	–	

7.15 Oscillator Failure Interrupt Vector Summary

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 96 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 and USB Clock	RTC and LCD Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	RTC and LCD Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

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 77.

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, asynchronous port interrupts, and the USB resume interrupt.

8.3.3 Power-save Mode

Power-save mode is identical to power down, with two exceptions:

1. 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.
2. If the LCD is enabled, it will keep running during sleep, and the device can wake up from LCD frame completed interrupt.

8.3.4 Standby Mode

Standby mode is identical to power down, with the exception that the enabled system clock sources are kept running while the CPU, peripheral, and RTC/LCD clocks are stopped. This reduces the wake-up time.

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.

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

When entering idle mode, the ADC should be disabled if not used. In other sleep modes, the ADC is automatically disabled. 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 326 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 352 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 109 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 115 for details on how to configure the watchdog timer.

8.5.5 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. 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.

8.5.6 On-chip Debug Systems

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.

Table 8-2. Sleep mode selection.

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	LCD	USB	–	AES	–	RTC	EVSYS	DMA
Read/Write	R/W	R/W	R	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – LCD: LCD Module**

Setting this bit stops the clock to the LCD module. When the bit is cleared the peripheral should be reinitialized to ensure proper operation.

- **Bit 6 – USB: USB Module**

Setting this bit stops the clock to the USB module. 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 – AES: AES Module**

Setting this bit stops the clock to the AES module. When this bit is cleared, the peripheral should be reinitialized to ensure proper operation.

- **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 – 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 – DMA: DMA Controller**

Setting this bit stops the clock to the DMA controller. This bit can be set only if the DMA controller is disabled.

8.7.2 PRPA/B – Power Reduction Port A/B register

Bit	7	6	5	4	3	2	1	0
+0x01/+0x02	–	–	–	–	–	–	**ADC**	**AC**
Read/Write	R	R	R	R	R	R	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 7:2 – 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 – 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/E – Power Reduction Port C/E Register

Bit	7	6	5	4	3	2	1	0
+0x03/+0x04/ +0x05/+0x06	–	**TWI**	–	**USART0**	**SPI**	**HIRES**	**TC1**	**TC0**
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 – TC1: Timer/Counter 1**

Setting this bit stops the clock to timer/counter 1. When this bit is cleared, the peripheral will continue like before the shut down.

- **Bit 0 – TC0: Timer/Counter 0**

Setting this bit stops the clock to timer/counter 0. 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	98

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	LCD	USB	–	AES	–	RTC	EVSYS	DMA	98
+0x01	PRPA	–	–	–	–	–	–	ADC	AC	99
+0x02	PRPB	–	–	–	–	–	–	ADC	AC	99
+0x03	PRPC	–	TWI	–	USART0	SPI	HIRES	TC1	TC0	99
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	PRPE	–	–	–	USART0	–	–	–	TC0	99

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 microcontroller 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.

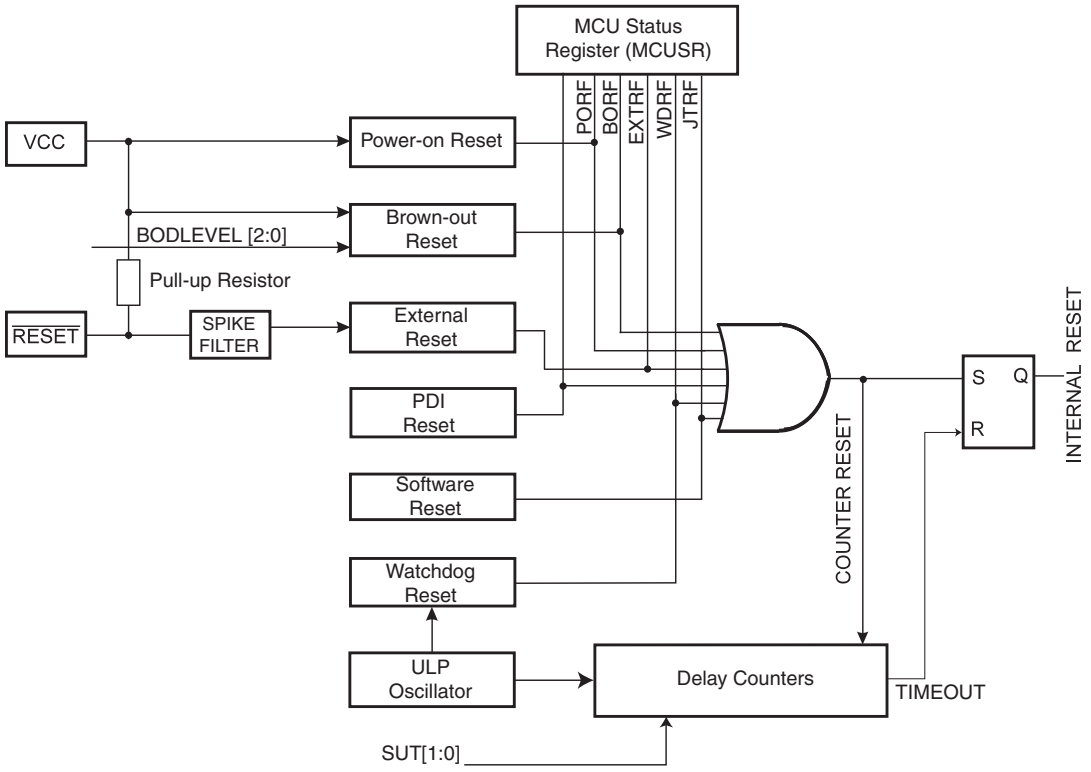
After reset is released from all reset sources, the default oscillator is started and calibrated before the device starts running from the reset vector address. By default, this is the lowest program memory address, 0, but it is possible to move the reset vector to the lowest address in the boot section.

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 103.

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 (clk_{SYS}) cycles 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.

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 internal 2MHz internal oscillator is chosen as the source for Clk_{SYS}.

9.3.2 Oscillator Startup

After the reset delay, the 2MHz internal oscillator clock is started, and its calibration values are automatically loaded from the calibration 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}.

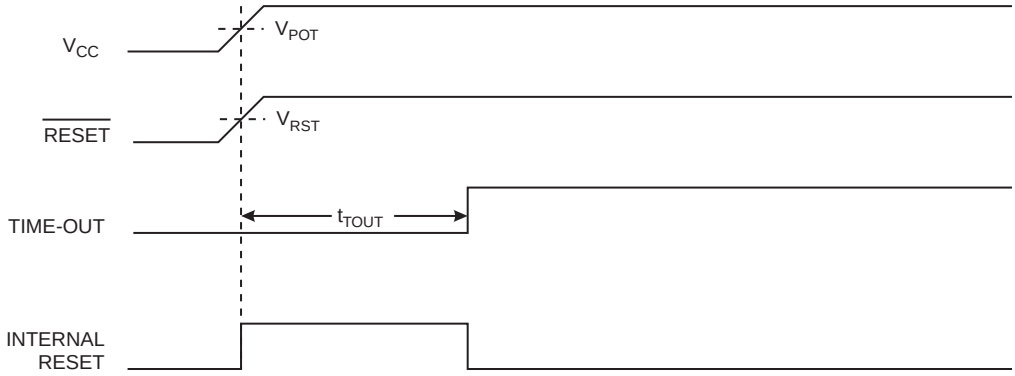
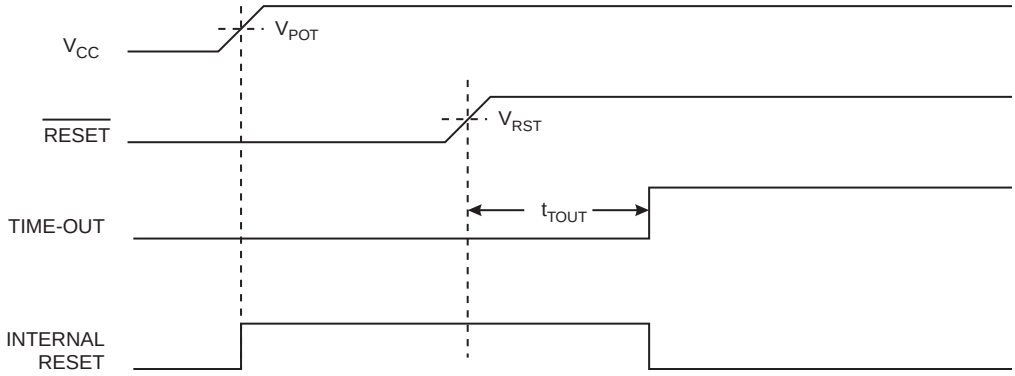


Figure 9-3. MCU startup, $\overline{\text{RESET}}$ extended externally,



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.

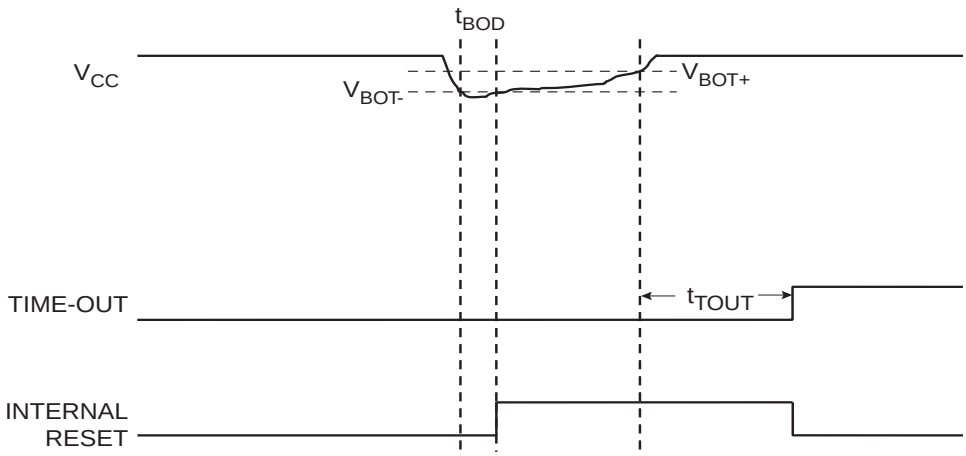
When the BOD is enabled and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-4), the brownout reset is immediately activated.

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.

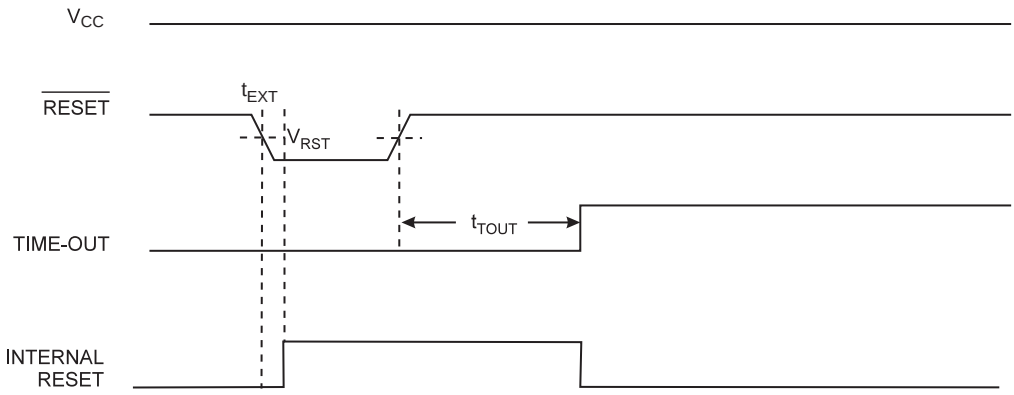
Table 9-3. BOD setting fuse decoding.

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.

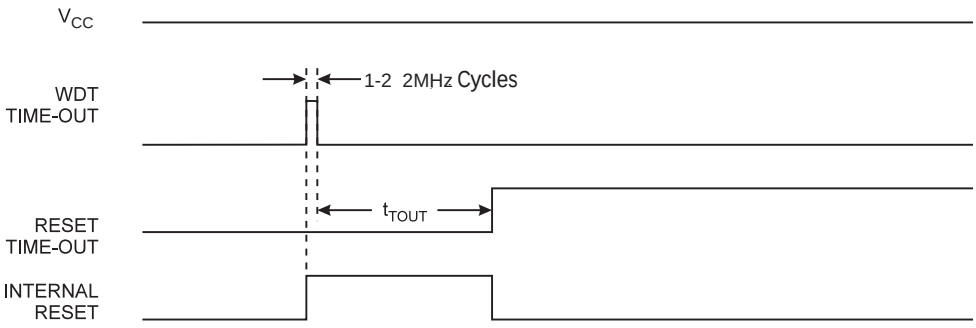


For external reset characterization data consult the device datasheet.

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 clock cycles of the 2MHz internal oscillator.

Figure 9-6. Watchdog reset.

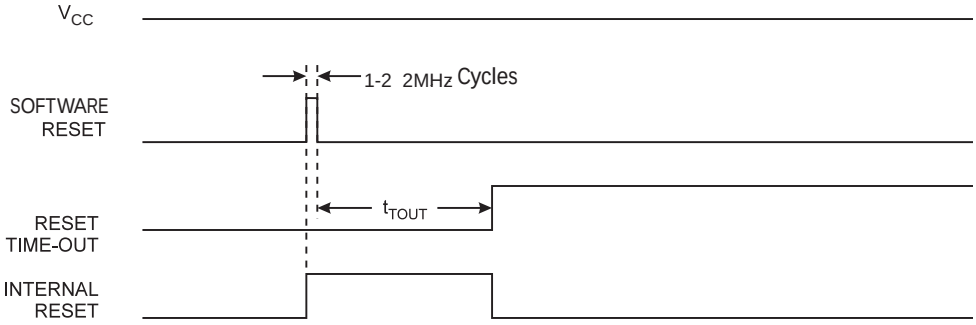


For information on configuration and use of the WDT, refer to the "WDT – Watchdog Timer" on page 110.

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	108
+0x01	CTRL	–	–	–	–	–	–	–	SWRST	108

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.

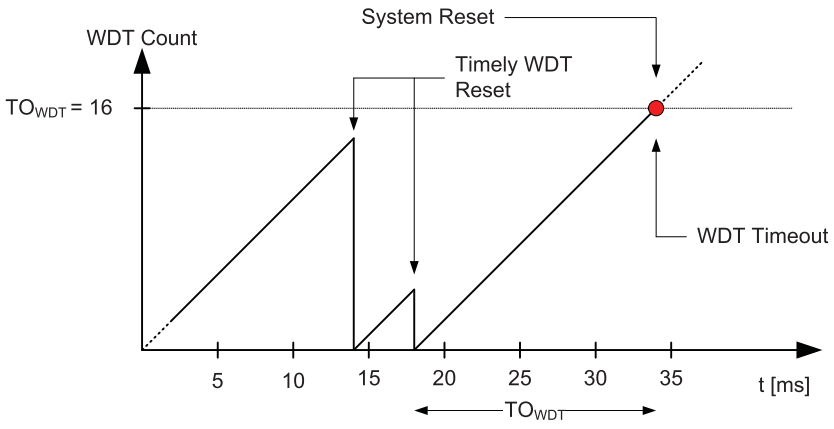
The WDT will run in active mode and all sleep modes, if enabled. It is asynchronous, runs from a CPU-independent clock source, and will continue to operate to issue a system reset even if the main clocks fail.

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 110.

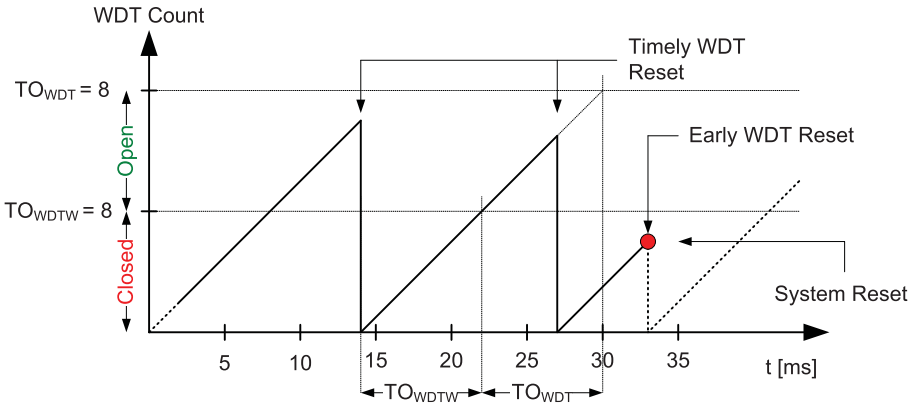
Figure 10-1. Normal mode operation.



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 112 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 112. 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

WPER[3:0]	Group Configuration	Typical Closed Window Periods
1010	8KCLK	8.0s
1011	–	Reserved
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 113 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 113. 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	112
+0x01	WINCTRL	–	–	WPER[3:0]				WEN	WCEN	113
+0x02	STATUS	–	–	–	–	–	–	–	SYNCBUSY	114

11. 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.

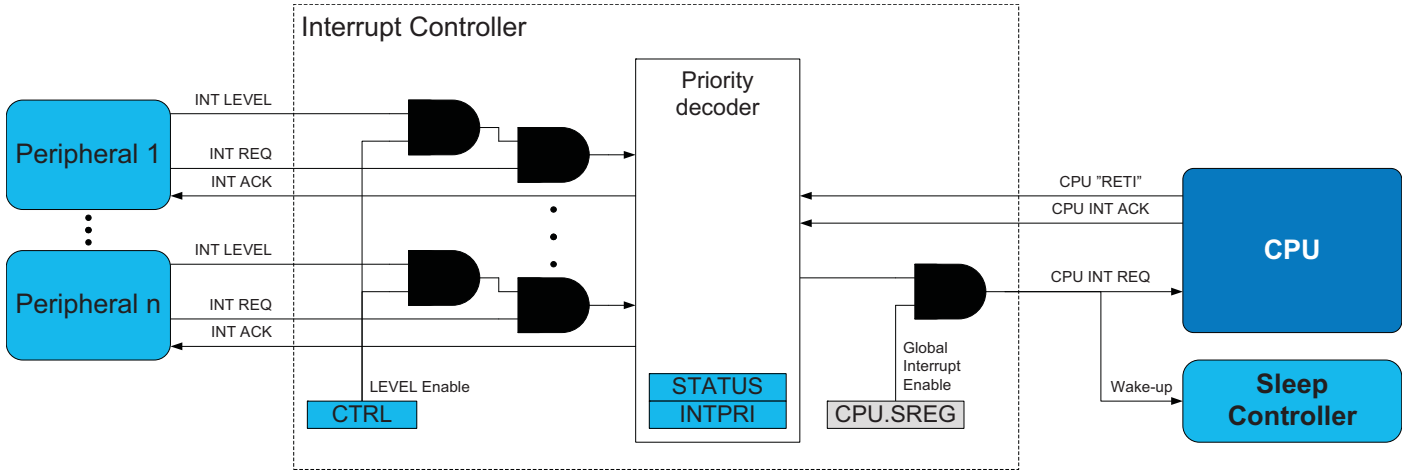
11.3 Operation

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 status register. 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 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.

All interrupts have 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 375 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.

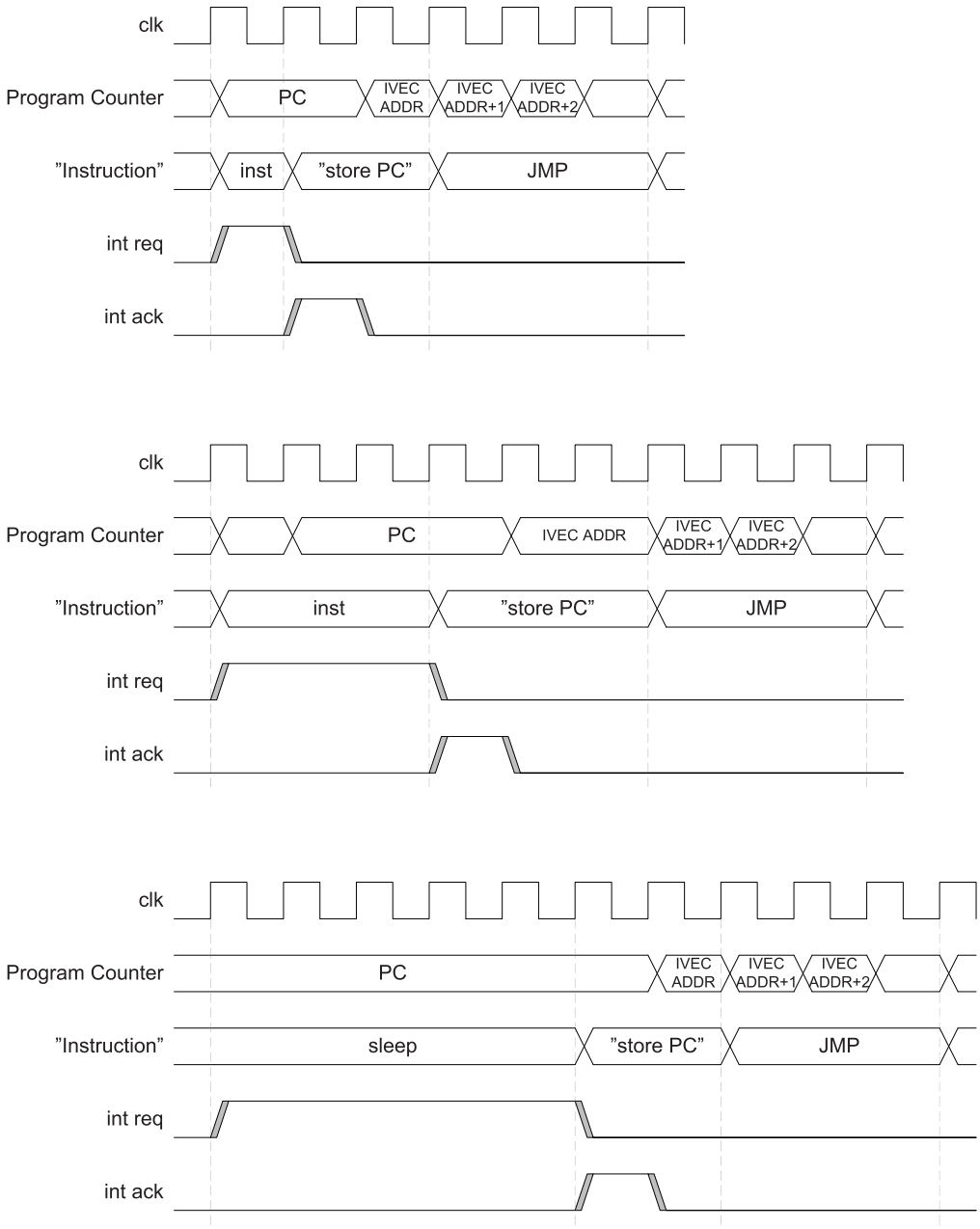
An NMI will be executed regardless of the setting of the I bit, and it will never change the I bit. No other interrupts can interrupt a NMI handler. If more than one NMI is requested at the same time, priority is static according to the interrupt vector address, where the lowest address has highest priority.

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 117 for more details.

Figure 11-2. Interrupt execution of a multicycle 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 is shown in Table 11-1.

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.

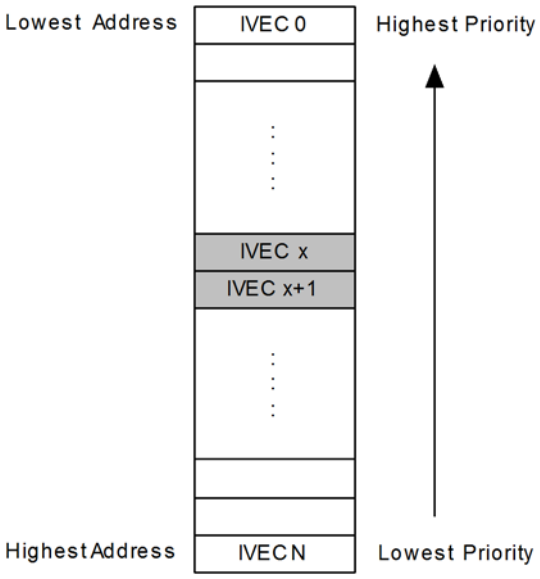
11.6 Interrupt priority

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.

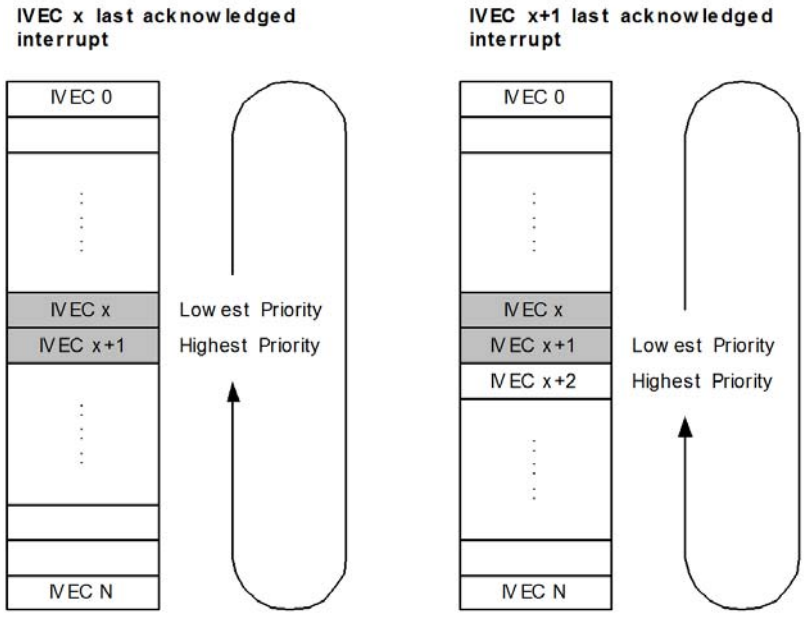
Figure 11-3. Static priority.



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	121
+0x01	INTPRI	INTPRI[7:0]								121
+0x02	CTRL	RREN	IVSEL	–	–	–	HILVLEN	MEDLVLEN	LOLVLEN	122

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
- Asynchronous pin change sensing that can wake the device from all sleep modes
- Two port interrupts with pin masking per I/O port
- 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, SPI, and timer/counter input/output pin locations

12.2 Overview

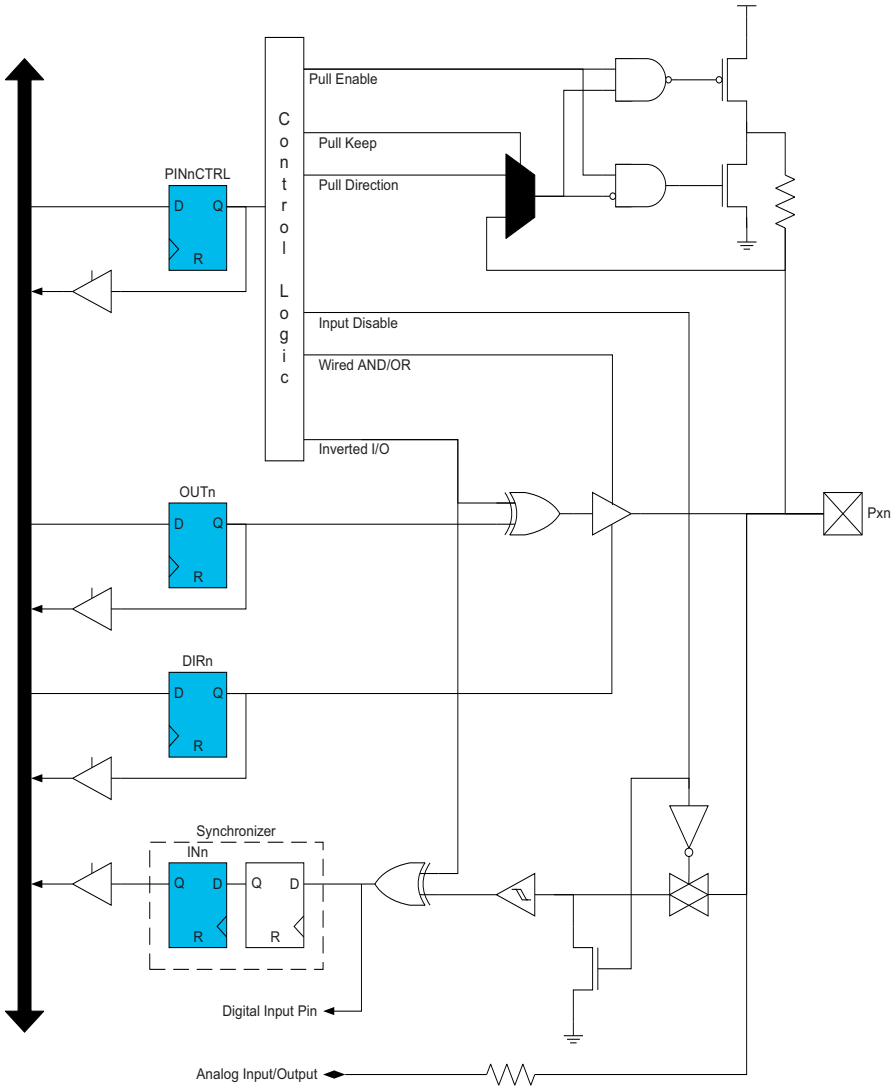
AVR XMEGA microcontrollers have flexible general purpose I/O ports. One port consists of up to eight port pins: pin 0 to 7. Each port pin can be configured as input or output with configurable driver and pull settings. They also implement synchronous and asynchronous input sensing with interrupts and 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 digital peripherals, such as USART, SPI, and timer/counters, can be remapped to selectable pin locations in order to optimize pin-out versus application needs.

Figure 12-1 on page 124 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.

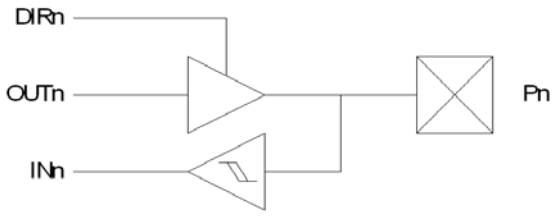
Since pull configuration is configured through the pin configuration register, all intermediate port states during switching of the pin direction and pin values are avoided.

The I/O pin configurations are summarized with simplified schematics in Figure 12-2 on page 125 to Figure 12-7 on page 127.

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.

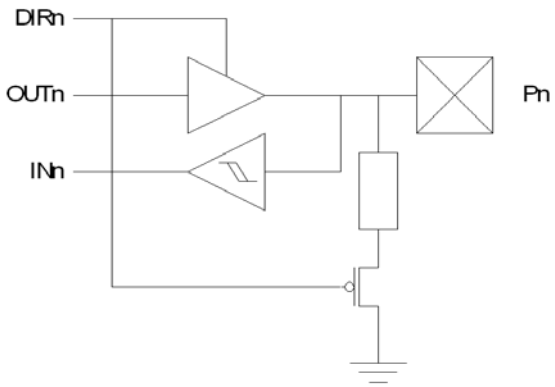
Figure 12-2. I/O pin configuration - Totem-pole (push-pull).



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.

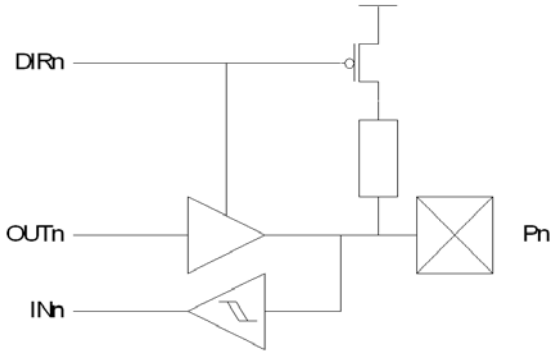
Figure 12-3. I/O pin configuration - Totem-pole with pull-down (on input).



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.

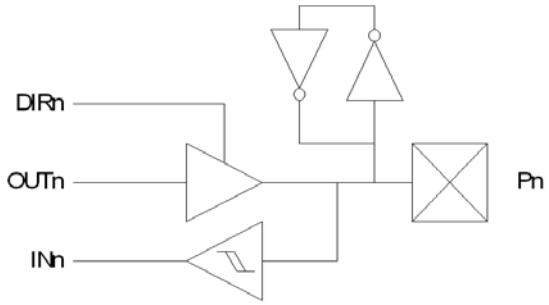
Figure 12-4. I/O pin configuration - Totem-pole with pull-up (on input).



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.

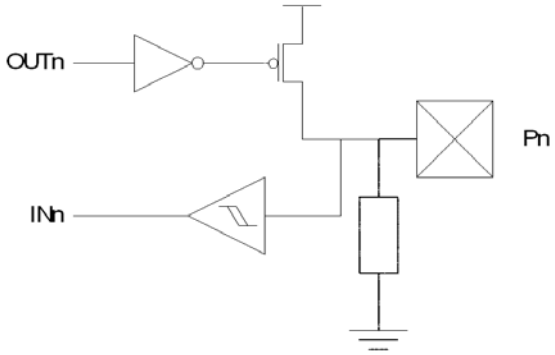
Figure 12-5. I/O pin configuration - Totem-pole with bus-keeper.



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.

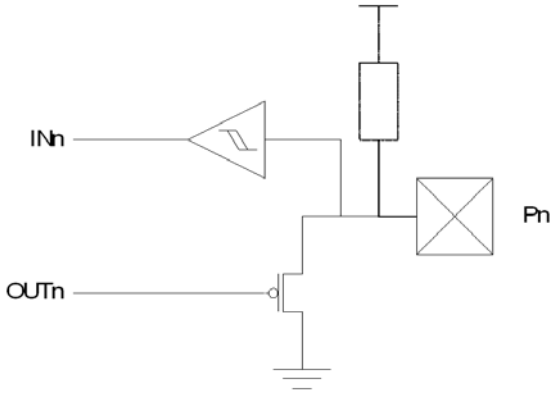
Figure 12-6. Output configuration - Wired-OR with optional pull-down.



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.

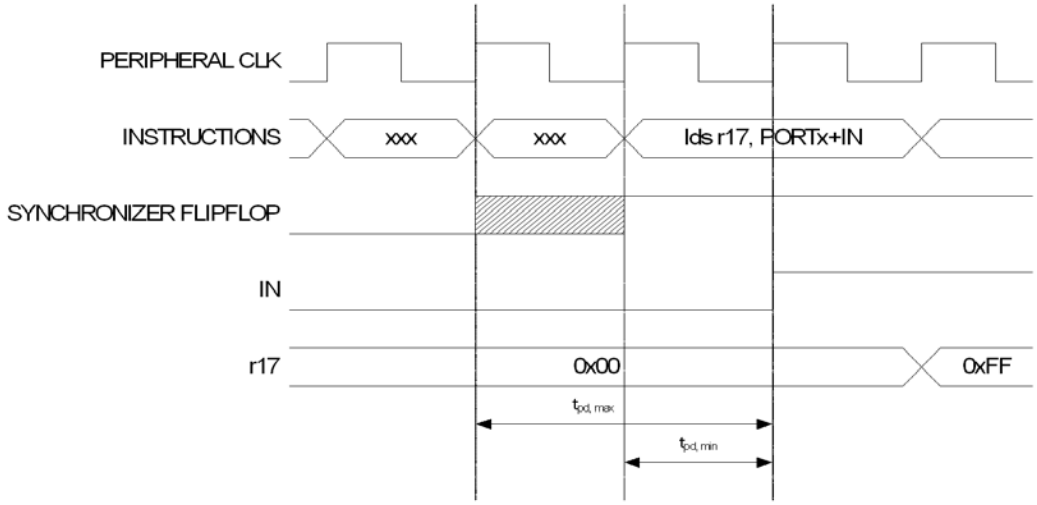
Figure 12-7. Output configuration - Wired-AND with optional pull-up.



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 124. 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 128 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.

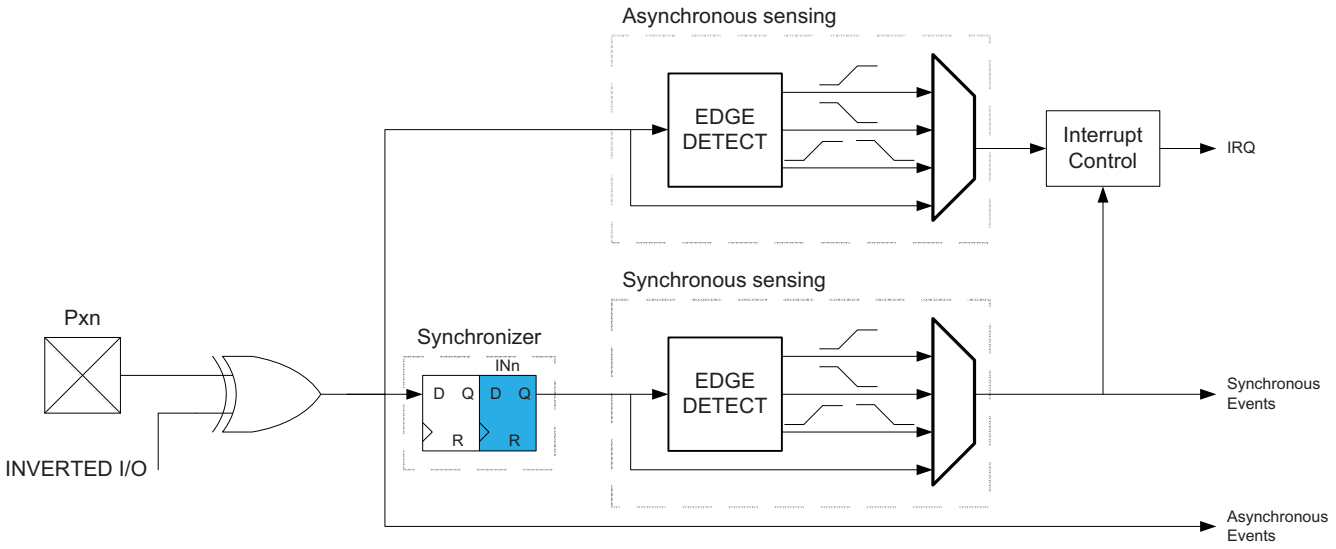


12.5 Input Sense Configuration

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 two interrupt vectors, and it is configurable which pins on the port will trigger each interrupt. Port interrupts must be enabled before they can be used. Which sense configurations can be used to generate interrupts 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 on page 130 summarize when interrupts 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
Both edges	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 value 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 an event when there is a change on the pin. The sense configurations decide the conditions for each pin to generate events. Event generation requires the presence of a peripheral clock, and asynchronous event generation is not possible. For edge sensing, the changed pin value must be sampled once by the peripheral clock for an event to be generated.

For level sensing, a low-level pin value will not generate events, and a high-level pin value will continuously generate events. For events to be generated on a low level, the pin configuration must be set to inverted I/O.

Table 12-4. Event sense support

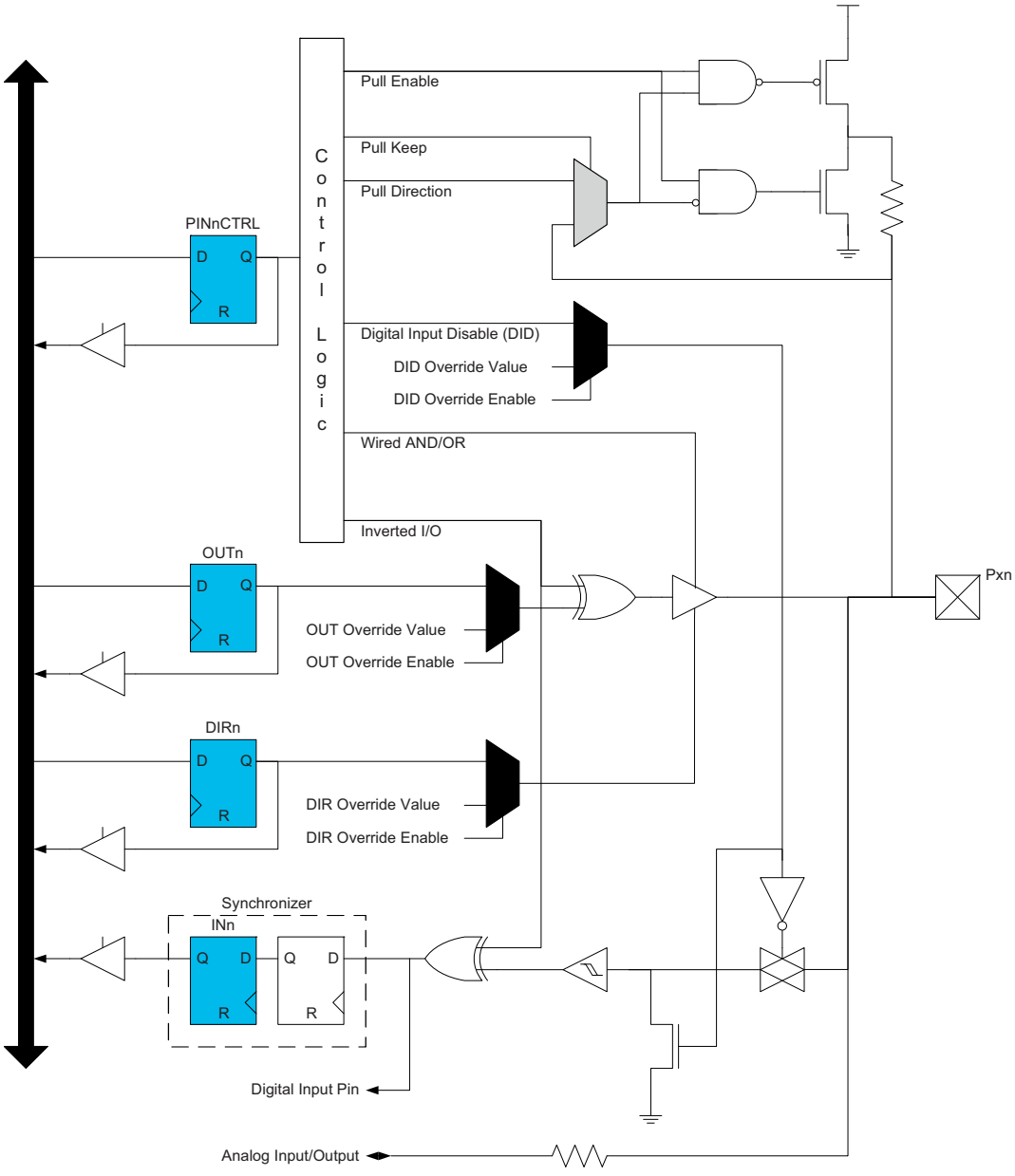
Sense Settings	Signal event	Data event
Rising edge	Rising edge	Pin value
Falling edge	Falling edge	Pin value
Both edge	Any edge	Pin value
Low level	Pin value	Pin value

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 131. 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 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.

12.10 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.

12.11 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.

12.12 Register Descriptions – Ports

12.12.1 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.

12.12.2 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.

12.12.3 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.

12.12.4 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.

12.12.5 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.

12.12.6 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.

12.12.7 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.

12.12.8 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.

12.12.9 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.

12.12.10INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	INT1LVL[1:0]		INT0LVL[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 – INTnLVL[1:0]: Interrupt n Level**

These bits enable port interrupt n and select the interrupt level as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

12.12.11INT0MASK – Interrupt 0 Mask register

Bit	7	6	5	4	3	2	1	0
+0x0A	INT0MSK[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 – INT0MSK[7:0]: Interrupt 0 Mask bits**

These bits are used to mask which pins can be used as sources for port interrupt 0. If INT0MASKn is written to one, pin n is used as source for port interrupt 0.The input sense configuration for each pin is decided by the PINnCTRL registers.

12.12.12INT1MASK – Interrupt 1 Mask register

Bit	7	6	5	4	3	2	1	0
+0x0B	INT1MSK[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 – INT1MASK[7:0]: Interrupt 1 Mask bits**

These bits are used to mask which pins can be used as sources for port interrupt 1. If INT1MASKn is written to one, pin n is used as source for port interrupt 1.The input sense configuration for each pin is decided by the PINnCTRL registers.

12.12.13INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x0C	–	–	–	–	–	–	INT1IF	INT0IF
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 – INTnIF: Interrupt 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 n. 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.

12.12.14REMAP – Pin Remap register

The pin remap functionality is available for PORTC - PORTF only.

Bit	7	6	5	4	3	2	1	0
+0x0E	–	–	SPI	USART0	TC0D	TC0C	TC0B	TC0A
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 – SPI: SPI Remap**

Setting this bit to one will swap the pin locations of the SCK and MOSI pins to have pin compatibility between SPI and USART when the USART is operating as a SPI master.

- **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 – TC0D: Timer/Counter 0 Output Compare D**

Setting this bit will move the location of OC0D from Px3 to Px7.

- **Bit 2 – TC0C: Timer/Counter 0 Output Compare C**

Setting this bit will move the location of OC0C from Px2 to Px6.

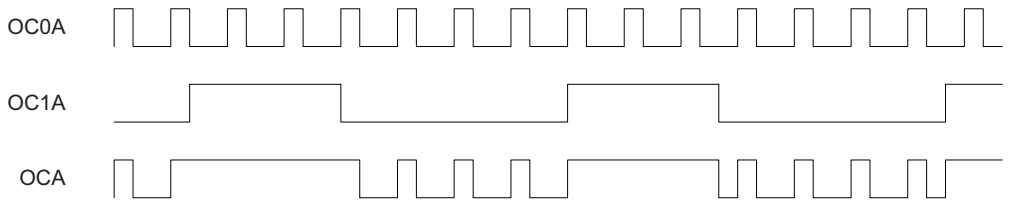
- **Bit 1 – TC0B: Timer/Counter 0 Output Compare B**

Setting this bit will move the location of OC0B from Px1 to Px5. If this bit is set and PWM from both timer/counter 0 and timer/counter 1 is enabled, the resulting PWM will be an OR-modulation between the two PWM outputs.

- **Bit 0 – TC0A: Timer/Counter 0 Output Compare A**

Setting this bit will move the location of OC0A from Px0 to Px4. If this bit is set and PWM from both timer/counter 0 and timer/counter 1 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.



12.12.15PINnCTRL – Pin n Configuration Register

Bit	7	6	5	4	3	2	1	0
	–	INVEN	OPC[2:0]			ISC[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 – 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.

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 on page 138. 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	Description
000	BOTHEDGES	Sense both edges
001	RISING	Sense rising edge
010	FALLING	Sense falling edge
011	LEVEL	Sense low level(1)

ISC[2:0]	Group Configuration	Description
100	–	Reserved
101	–	Reserved
110	–	Reserved
111	INTPUT_DISABLE	Digital input buffer disabled(2)

Note:
1. A low-level pin value will not generate events, and a high-level pin value will continuously generate events.
2. Only PORTA - PORTF 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.

12.13 Register Descriptions – Port Configuration

12.13.1 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.

12.13.2 VPCTRLA – Virtual Port-map Control register A

Bit	7	6	5	4	3	2	1	0
+0x02	VP1MAP[3:0]				VP0MAP[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 – VP1MAP: Virtual Port 1 Mapping**

These bits decide which ports should be mapped to Virtual Port 1. The registers DIR, OUT, IN, and INTFLAGS will be mapped. Accessing the virtual port registers is equal to accessing the actual port registers. See Table 12-7 on page 141 for configuration.

- **Bit 3:0 – VP0MAP: Virtual Port 0 Mapping**

These bits decide which ports should be mapped to Virtual Port 0. The registers DIR, OUT, IN, and INTFLAGS will be mapped. Accessing the virtual port registers is equal to accessing the actual port registers. See Table 12-7 on page 141 for configuration.

12.13.3 VPCTRLB – Virtual Port-map Control register B

Bit	7	6	5	4	3	2	1	0
+0x03	VP3MAP[3:0]				VP2MAP[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 – VP3MAP: Virtual Port 3 Mapping**

These bits decide which ports should be mapped to Virtual Port 3. The registers DIR, OUT, IN, and INTFLAGS will be mapped. Accessing the virtual port registers is equal to accessing the actual port registers. See Table 12-7 on page 141 for configuration.

- **Bit 3:0 – VP2MAP: Virtual Port 2 Mapping**

These bits decide which ports should be mapped to Virtual Port 2. The registers DIR, OUT, IN, and INTFLAGS will be mapped. Accessing the virtual port registers is equal to accessing the actual port registers. See Table 12-7 on page 141 for configuration.

Table 12-7. Virtual port mapping

VPnMAP[3:0]	Group Configuration	Description
0000	PORTA	PORTA mapped to Virtual Port n
0001	PORTB	PORTB mapped to Virtual Port n
0010	PORTC	PORTC mapped to Virtual Port n
0011	PORTD	PORTD mapped to Virtual Port n
0100	PORTE	PORTE mapped to Virtual Port n
0101	PORTF	PORTF mapped to Virtual Port n
0110	PORTG	PORTG mapped to Virtual Port n
0111	PORTH	PORTH mapped to Virtual Port n
1000	PORTJ	PORTJ mapped to Virtual Port n
1001	PORTK	PORTK mapped to Virtual Port n
1010	PORTL	PORTL mapped to Virtual Port n
1011	PORTM	PORTM mapped to Virtual Port n
1100	PORTN	PORTN mapped to Virtual Port n
1101	PORTP	PORTP mapped to Virtual Port n
1110	PORTQ	PORTQ mapped to Virtual Port n
1111	PORTR	PORTR mapped to Virtual Port n

12.13.4 CLKEVOUT – Clock and Event Out register

Bit	7	6	5	4	3	2	1	0
+0x04	CLKEVPIN	RTCOUT	EVOUT[1:0]		CLKOUTSEL[1:0]		CLKOUT[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 – 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 – RTCOUT: RTC Clock Output Enable**

Setting this bit enables output of the RTC clock source on PORTC pin 6.

- **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. Pin 7 on the selected port 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 12-8 on page 142 shows the possible configurations.

Table 12-8. 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	PE	Event channel 0 output on PORTE

- **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 12-9. Clock 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 12-10 shows the possible configurations.

Table 12-10. 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

12.13.5 EVCTRL – Event Control register

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	–	–	EVOUTSEL[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 – EVOUTSEL[2:0]: Event Channel Output Selection**

These bits define which channel from the event system is output to the port pin. Table 12-11 shows the available selections.

Table 12-11. 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 Register Descriptions – Virtual Port

12.14.1 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 mapped by VPCTRLA, virtual port-map control register A or VPCTRLB, virtual port-map control register B. When a port is mapped as virtual, accessing this register is identical to accessing the actual DIR register for the port.

12.14.2 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 mapped by VPCTRLA, virtual port-map control register A or VPCTRLB, virtual port-map control register B. When a port is mapped as virtual, accessing this register is identical to accessing the actual OUT register for the port.

12.14.3 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. The configuration of VPCTRLA, virtual port-map control register A or VPCTRLB, virtual port-map control register A, decides the value in the register. When a port is mapped as virtual, accessing this register is identical to accessing the actual IN register for the port.

12.14.4 INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	–	–	**INT1IF**	**INT0IF**
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 – INTnIF: Interrupt 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 n. 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. The configuration of VPCTRLA, virtual port-map control register A, or VPCTRLB, Virtual Port-map Control Register B, decides which flags are mapped. When a port is mapped as virtual, accessing this register is identical to accessing the actual INTFLAGS register for the port.

12.15 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]								133
+0x01	DIRSET	DIRSET[7:0]								133
+0x02	DIRCLR	DIRCLR[7:0]								133
+0x03	DIRTGL	DIRTGL[7:0]								134
+0x04	OUT	OUT[7:0]								134
+0x05	OUTSET	OUTSET[7:0]								134
+0x06	OUTCLR	OUTCLR[7:0]								134
+0x07	OUTTGL	OUTTGL[7:0]								135
+0x08	IN	IN[7:0]								135
+0x09	INTCTRL	–	–	–	–	INT1LVL[1:0]		INT0LVL[1:0]		135
+0x0A	INT0MASK	INT0MSK[7:0]								136
+0x0B	INT1MASK	INT1MSK[7:0]								136
+0x0C	INTFLAGS	–	–	–	–	–	–	INT1IF	INT0IF	136
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	REMAP	–	–	SPI	USART0	TC0D	TC0C	TC0B	TC0A	137
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10	PIN0CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x11	PIN1CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x12	PIN2CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x13	PIN3CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x14	PIN4CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x15	PIN5CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x16	PIN6CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x17	PIN7CTRL	–	INVEN	OPC[2:0]			ISC[2:0]			138
+0x18	Reserved	–	–	–	–	–	–	–	–	
+0x19	Reserved	–	–	–	–	–	–	–	–	
+0x1A	Reserved	–	–	–	–	–	–	–	–	
+0x1B	Reserved	–	–	–	–	–	–	–	–	
+0x1C	Reserved	–	–	–	–	–	–	–	–	
+0x1D	Reserved	–	–	–	–	–	–	–	–	
+0x1E	Reserved	–	–	–	–	–	–	–	–	
+0x1F	Reserved	–	–	–	–	–	–	–	–	

12.16 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]								140
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	VPCTRLA	VP1MAP[3:0]				VP0MAP[3:0]				140
+0x03	VPCTRLB	VP3MAP[3:0]				VP2MAP[3:0]				140
+0x04	CLKEVOU	CLKEVPIN	RTCOUT	EVOUT[1:0]		CLKOUTSEL		CLKOUT[1:0]		141
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	EVCTRL	–	–	–	–	–	EVCTRL[2:0]			143
+0x07	Reserved	–	–	–	–	–	–	–	–	

12.17 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]								144
+0x01	OUT	OUT[7:0]								144
+0x02	IN	IN[7:0]								144
+0x03	INTFLAGS	–	–	–	–	–	–	INT1IF	INT0IF	145

12.18 Interrupt Vector Summary – Ports

Offset	Source	Interrupt Description
0x00	INT0_vect	Port interrupt vector 0 offset
0x02	INT1_vect	Port interrupt vector 1 offset

13. TC0/1 – 16-bit Timer/Counter Type 0 and 1

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 0
  - Two CC channels for timer/counters of type 1
- 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 DMA and to trigger DMA transactions
- High-resolution extension
  - Increases frequency and waveform resolution by 4x (2-bit) or 8x (3-bit)
- Advanced waveform extension:
  - Low- and high-side output with programmable dead-time insertion (DTI)
  - 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, 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.

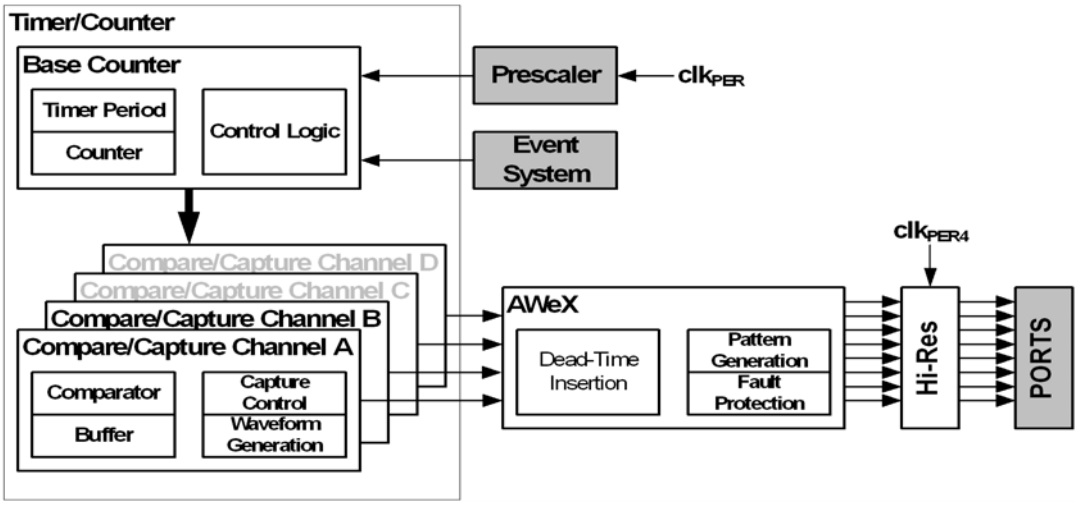
A timer/counter can be clocked and timed from the peripheral clock with optional prescaling or from the event system. The event system can also be used for direction control and capture trigger or to synchronize operations.

There are two differences between timer/counter type 0 and type 1. Timer/counter 0 has four CC channels, and timer/counter 1 has two CC channels. All information related to CC channels 3 and 4 is valid only for timer/counter 0. Only Timer/Counter 0 has the split mode feature that split it into two 8-bit Timer/Counters with four compare channels each.

Some timer/counters have extensions to enable more specialized waveform and frequency generation. The advanced waveform extension (AWeX) is intended for motor control and other power control applications. It enables low- and high-side output with dead-time insertion, as well as fault protection for disabling and shutting down external drivers. It can also generate a synchronized bit pattern across the port pins. The high-resolution (hi-res) extension can be used to increase the waveform output resolution by four or eight times by using an internal clock source running up to four times faster than the peripheral clock.

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 149.

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

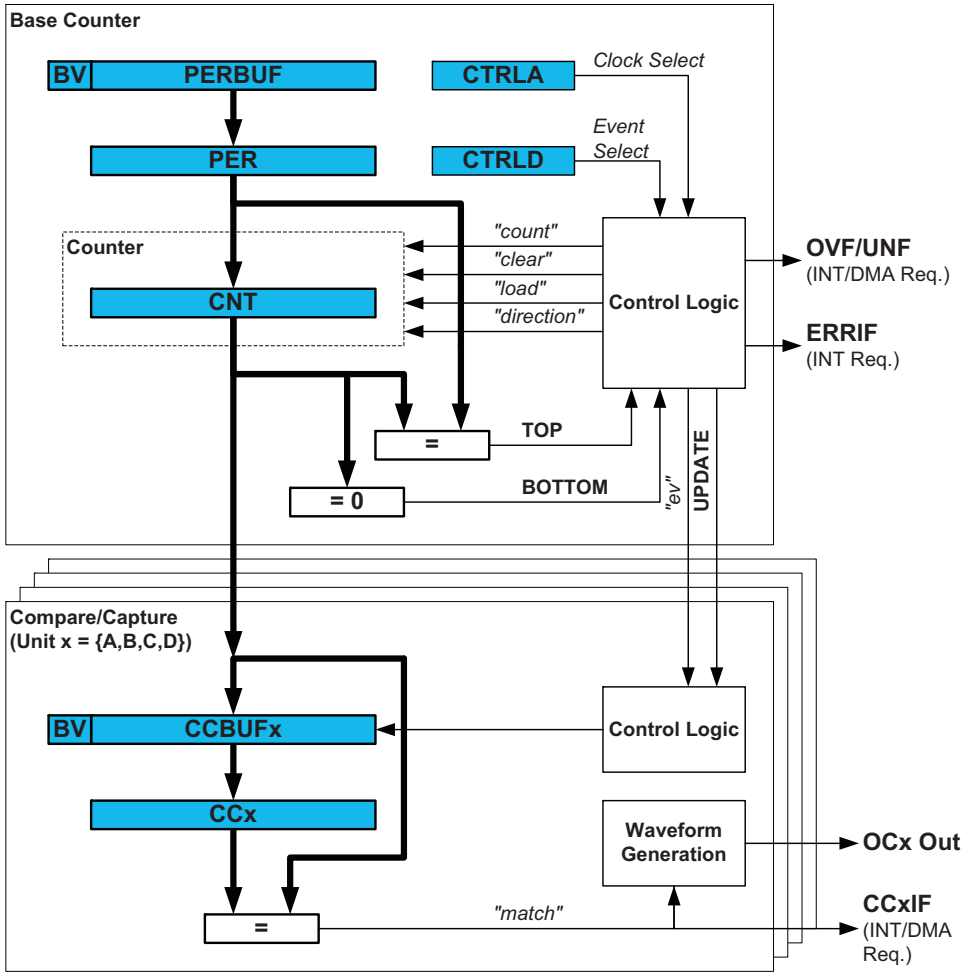
Name	Description
BOTTOM	The counter reaches BOTTOM when it becomes zero.
MAX	The counter reaches MAXimum when it becomes all ones.
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be equal to the period (PER) or the compare channel A (CCA) 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.

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 on page 150 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.

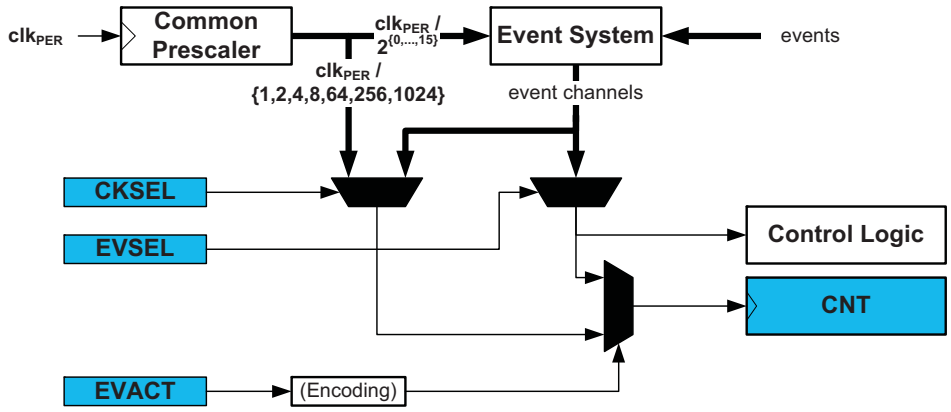
The counter value is also compared to the CCx registers. These comparisons can be used to generate interrupt requests, request DMA transactions or generate events for the event system. The waveform generator modes use these comparisons to set the waveform period or pulse width.

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 is 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 152. 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.

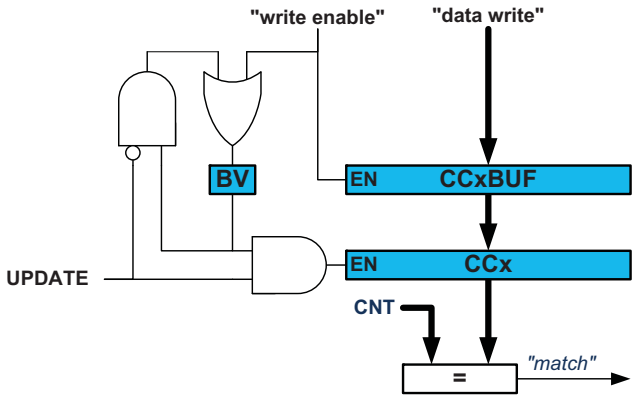
In addition, the timer/counter can be controlled via the event system. The event selection (EVSEL) and event action (EVACT) settings are used to trigger an event action from one or more events. This is referred to as event action controlled operation of the counter. For details, refer to "Event Action Controlled Operation" on page 153. 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.

13.5 Double Buffering

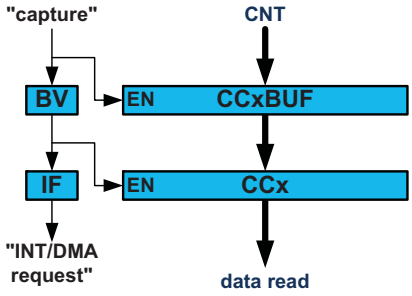
The period register and the CC registers are all double buffered. Each buffer register has a buffer valid (BV) flag, which indicates that the buffer register contains a valid, i.e. new, value that can be copied into the corresponding period or CC register. When the period register and CC channels are 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. This is shown for a compare register in Figure 13-4 on page 152.

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 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.

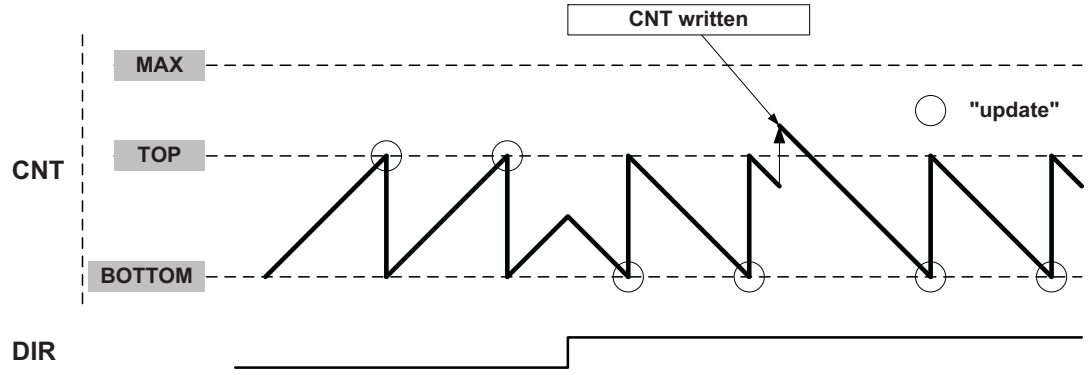
13.6 Counter Operation

Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each timer/counter clock input.

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 capture channels.

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.

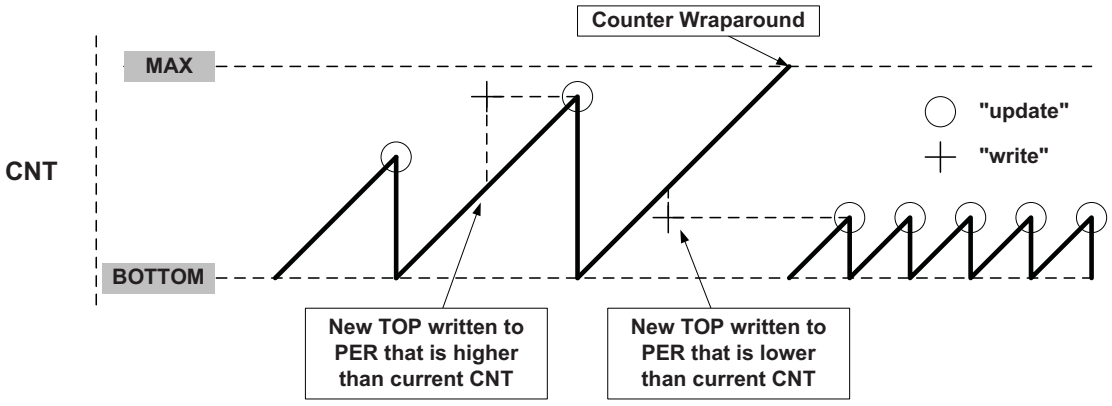
13.6.3 32-bit Operation

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

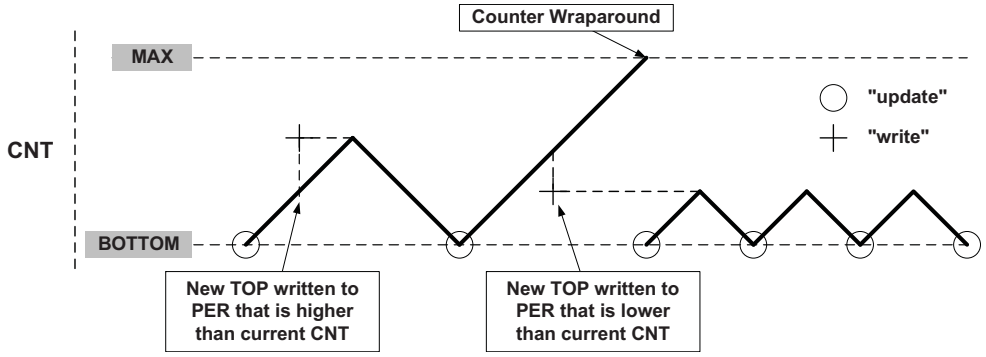
The counter period is changed by writing a new TOP value to the period register. If double buffering is not used, any period update is immediate, as shown in Figure 13-7 on page 153.

Figure 13-7. Changing the period without buffering.



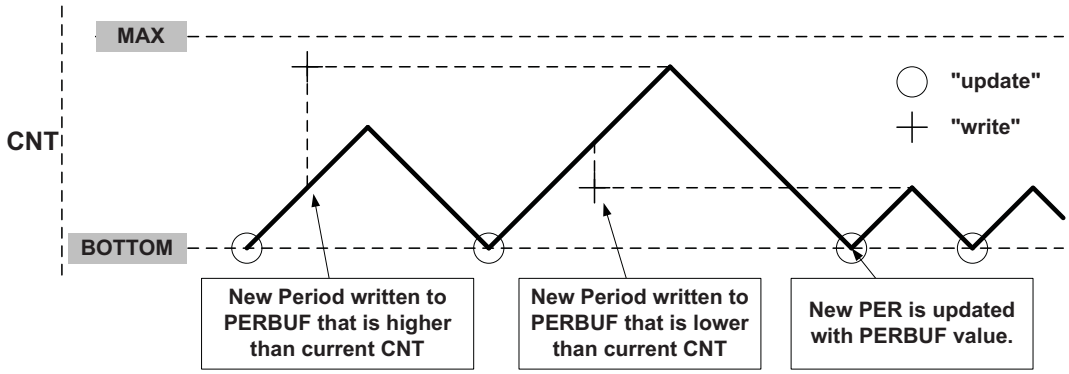
A counter wraparound can occur in any mode of operation when up-counting without buffering, as shown in Figure 13-8. This due to the fact that CNT and PER are continuously compared, and if a new TOP value that is lower than current CNT is written to PER, it will wrap before a compare match happen.

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.

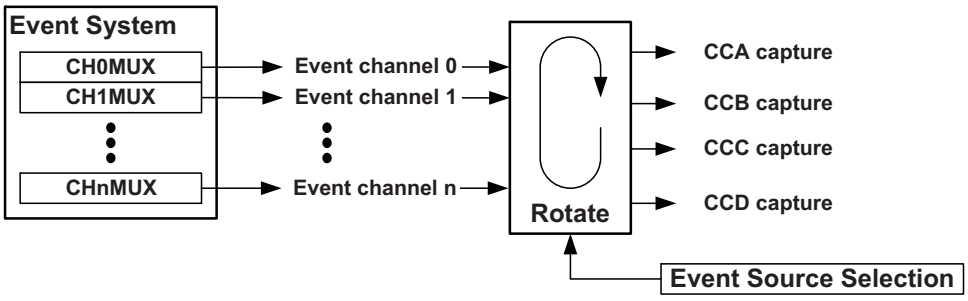


13.7 Capture Channel

The CC channels can be used as capture channels to capture external events and give them a timestamp. To use capture, the counter must be set for normal operation.

Events are used to trigger the capture; i.e., any events from the event system, including pin change from any pin, can trigger a capture operation. The event source select setting selects which event channel will trigger CC channel A. The subsequent event channels then trigger events on subsequent CC channels, if configured. For example, setting the event source select to event channel 2 results in CC channel A being triggered by event channel 2, CC channel B triggered by event channel 3, and so on.

Figure 13-10.Event source selection for capture operation.



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.

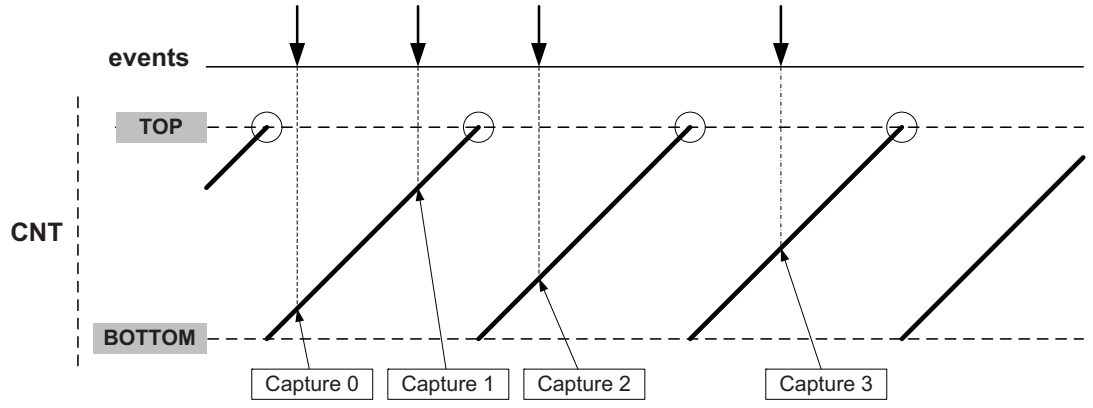
When an I/O pin is used as an event source for the capture, the pin must be configured for edge sensing. For details on sense configuration on I/O pins, refer to “Input Sense Configuration” on page 129. If the period register value is lower than 0x8000, the polarity of the I/O pin edge will be stored in the most-significant bit (msb) of the capture register. If the msb of the capture register is zero, a falling edge generated the capture. If the msb is one, a rising edge generated the capture.

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.

The counter will continuously count from BOTTOM to TOP, and then restart at BOTTOM, as shown in Figure 13-11. The figure also shows four capture events for one capture channel.

Figure 13-11.Input capture timing.



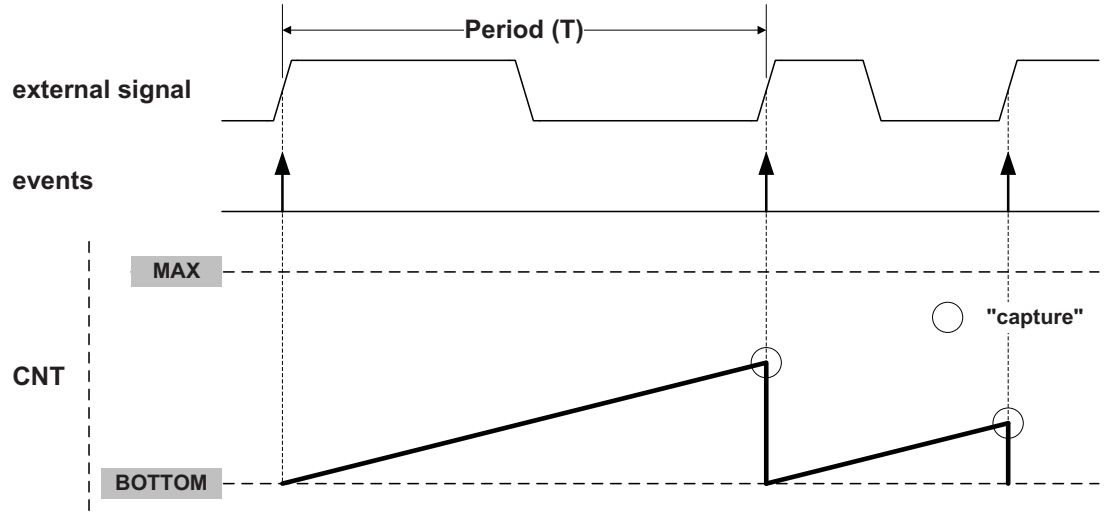
13.7.2 Frequency Capture

Selecting the frequency capture event action makes the enabled capture channel perform an input capture and restart on positive edge events. This enables the timer/counter to measure the period or frequency of a signal directly. The capture result will be the time (T) from the previous timer/counter restart until the event occurred. This can be used to calculate the frequency (f) of the signal:

$$f = \frac{1}{T}$$

Figure 13-12 on page 156 shows an example where the period of an external signal is measured twice.

Figure 13-12.Frequency capture of an external signal.

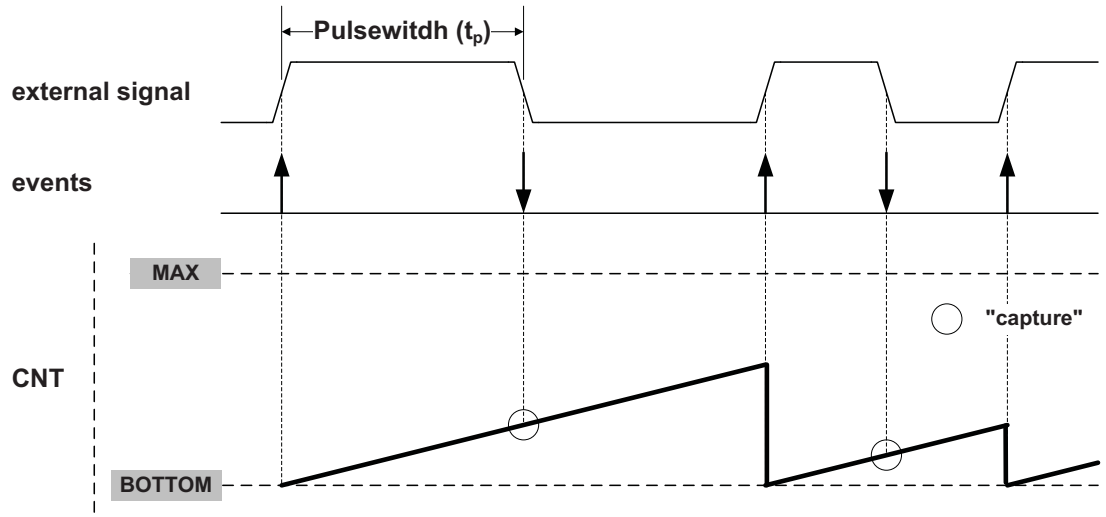


Since all capture channels use the same counter (CNT), only one capture channel must be enabled at a time. If two capture channels are used with different sources, the counter will be restarted on positive edge events from both input sources, and the result will have no meaning.

13.7.3 Pulse Width Capture

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 event. 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 13-13 on page 156 shows and example where the pulse width is measured twice for an external signal.

Figure 13-13.Pulse width capture of an external signal.



13.7.4 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.

13.7.5 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

Each compare channel continuously compares the counter value (CNT) with the CCx register. If CNT equals CCx, the comparator signals a match. The match will set the CC channel's interrupt flag at the next timer clock cycle, and the event and 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 to either the TOP or BOTTOM of the counting sequence according to the UPDATE condition. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses for glitch-free output.

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:

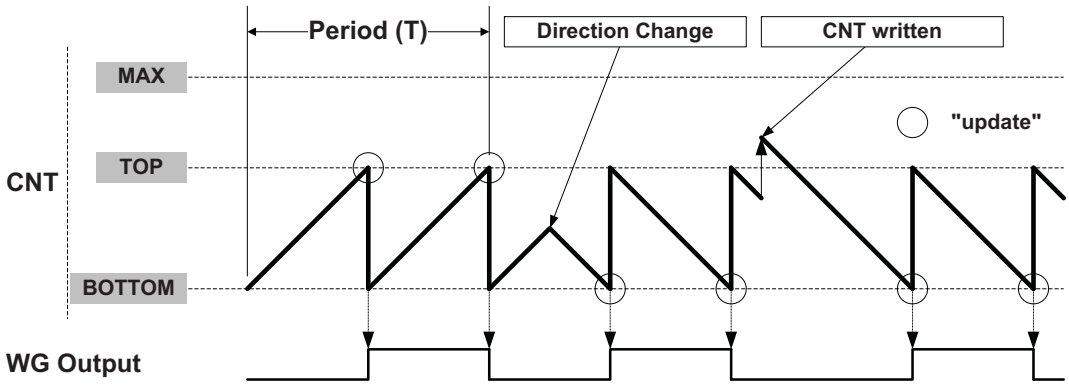
1. A waveform generation mode must be selected.
2. Event actions must be disabled.
3. The CC channels used must be enabled. This will override the corresponding port pin output register.
4. The direction for the associated port pin must be set to output.

Inverted waveform output is achieved by setting the invert output bit for the port pin.

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 13-14 on page 158.

Figure 13-14.Frequency waveform generation.



The waveform frequency (f_{FRQ}) is defined by the following equation:

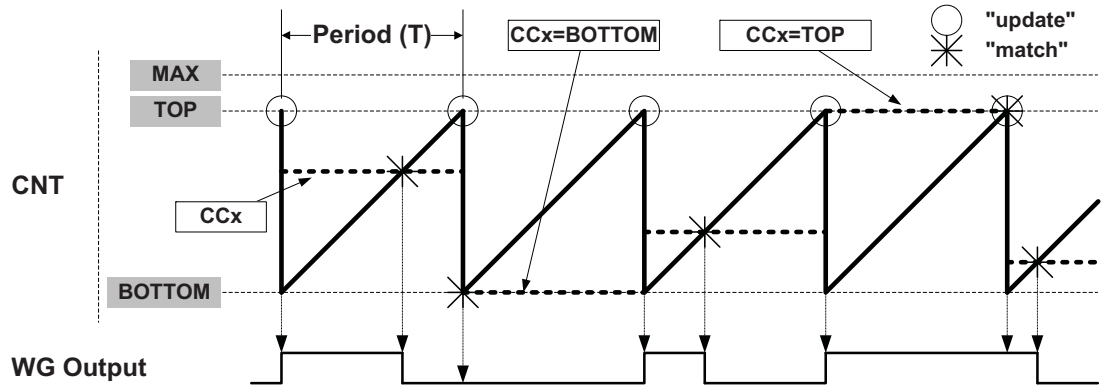
$$f_{FRQ} = \frac{fclk_{PER}}{2N(CCA + 1)}$$

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.

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 13-15 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 13-15.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}):

$$R_{PWM\_SS} = \frac{\log(PER + 1)}{\log(2)}$$

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:

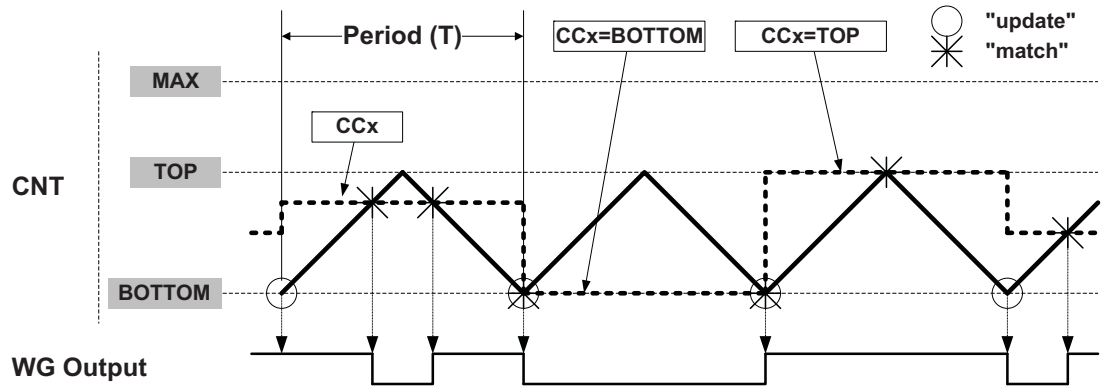
$$f_{\text{PWM\_SS}} = \frac{fclk_{PER}}{N(\text{PER} + 1)}$$

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.

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 13-16 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 13-16.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}):

$$R_{\text{PWM\_DS}} = \frac{\log(\text{PER} + 1)}{\log(2)}$$

The PWM frequency depends on the period setting (PER) and the peripheral clock frequency ($fclk_{PER}$), and can be calculated by the following equation:

$$f_{\text{PWM\_DS}} = \frac{fclk_{PER}}{2N\text{PER}}$$

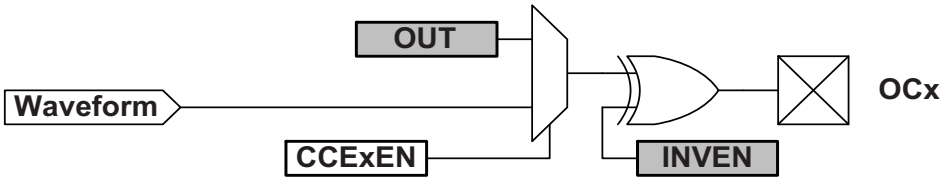
N represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the peripheral clock frequency ($fclk_{PER}$) when CCA is set to zero (0x0000) and no prescaling is used. This also applies when using the hi-res extension, since this increases the resolution and not the frequency.

13.8.5 Port Override for Waveform Generation

To make the waveform generation available on the port pins, the corresponding port pin direction must be set as output. The timer/counter will override the port pin values when the CC channel is enabled (CCENx) and a waveform generation mode is selected.

Figure 13-17 on page 160 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 13-17.Port override for timer/counter 0 and 1.



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 63.

13.10 DMA Support

The interrupt flags can be used to trigger DMA transactions. Table 13-2 lists the transfer triggers available from the timer/counter and the DMA action that will clear the transfer trigger. For more details on using DMA, refer to “DMAC - Direct Memory Access Controller” on page 47.

Table 13-2. DMA request sources

Request	Acknowledge	Comment
OVFIF/UNFIF	DMA controller writes to CNT DMA controller writes to PER DMA controller writes to PERBUF DMA controller writes to DTHSBUF or DTLSBUF in AWeX when in Pattern Generation Mode	
ERRIF	N/A	
CCxIF	DMA controller access of CCx DMA 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

13.12.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	CLKSEL[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 – CLKSEL[3:0]: Clock Select**

These bits select the clock source for the timer/counter according to Table 13-3.

CLKSEL=0001 must be set to ensure a correct output from the waveform generator when the hi-res extension is enabled.

Table 13-3. Clock select options

CLKSEL[3:0]	Group Configuration	Description
0000	OFF	None (i.e, timer/counter in OFF state)
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	CCDEN	CCCEN	CCBEN	CCAEN	–	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 7:4 – CCxEN: Compare or Capture Enable**

Setting these bits in the FRQ 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 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 13-4.

No waveform generation is performed in the normal mode of operation. For all other modes, the result from the waveform generator will only be directed to the port pins if the corresponding CCxEN bit has been set to enable this. The port pin direction must be set as output.

Table 13-4. Timer waveform generation mode.

WGMODE[2:0]	Group Configuration	Mode of Operation	Top	Update	OVFIF/Event
000	NORMAL	Normal	PER	TOP	TOP
001	FRQ	Frequency	CCA	TOP	TOP
010		Reserved	-	-	-
011	SINGLESLOPE	Single-slope PWM	PER	BOTTOM	BOTTOM
100		Reserved	-	-	-
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

13.12.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	CMPD	CMPC	CMPB	CMPA
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 – 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 13-5 on page 163.

The EVSEL setting will decide which event source or sources have control in this case.

Table 13-5. Timer event action selection.

EVACT[2:0]	Group Configuration	Event Action
000	OFF	None
001	CAPT	Input capture
010	UPDOWN	Externally controlled up/ down count
011	QDEC	Quadrature decode
100	RESTART	Restart waveform period
101	FRQ	Frequency capture
110	PW	Pulse width capture
111		Reserved

Selecting any of the capture event actions changes the behavior of the CCx registers and related status and control bits to be used for capture. The error status flag (ERRIF) will indicate a buffer overflow in this configuration. See “Event Action Controlled Operation” on page 153 for further details.

- **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 13-6. When the event action is set to a capture operation, 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 13-6. Timer event source selection

EVSEL[3:0]	Group Configuration	Event Source
0000	OFF	None
0001	–	Reserved
0010	–	Reserved
0011	–	Reserved

EVSEL[3:0]	Group Configuration	Event Source
0100	–	Reserved
0101	–	Reserved
0110	–	Reserved
0111	–	Reserved
1nnn	CHn	Event channel n, n={0,...,7}

13.12.5 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	–	BYTEM[1:0]	
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 1:0 – BYTEM[1:0]: Byte Mode**

These bits select the timer/counter operation mode according to Table 13-7.

Table 13-7. Clock select

BYTEM[1:0]	Group Configuration	Description
00	NORMAL	Timer/counter is set to normal mode (timer/counter type 0)
01	BYTEMODE	Upper byte of the counter (CNTH) will be set to zero after each counter clock cycle
10	SPLITMODE	Timer/counter 0 is split into two 8-bit timer/counters (timer/counter type 2)
11	–	Reserved

13.12.6 INTCTRLA – Interrupt Enable register A

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	–	ERRINTLVL[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

- **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 – ERRINTLVL[1:0]:Timer Error Interrupt Level**

These bits enable the timer error interrupt and select the interrupt level as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

- **Bit 1:0 – OVFINTLVL[1:0]:Timer Overflow/Underflow Interrupt Level**

These bits enable the timer overflow/underflow interrupt and select the interrupt level as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

13.12.7 INTCTRLB – Interrupt Enable 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[7:0] - 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

13.12.8 CTRLFCLR/CTRLFSET – Control register F 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.

13.12.8.1 CTRLFCLR

Bit	7	6	5	4	3	2	1	0
+0x08	–	–	–	–	CMD[1:0]		LUPD	DIR
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

13.12.8.2 CTRLFSET

Bit	7	6	5	4	3	2	1	0
+0x09	–	–	–	–	CMD[1:0]		LUPD	DIR
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 – 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 13-8. Command selections

CMD	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 OFFstate)

- **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 CTRLGCLR/CTRLGSET – Control register G Clear/Set

Bit	7	6	5	4	3	2	1	0
+0x0A/ +0x0B	–	–	–	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

Refer to “CTRLFCLR/CTRLFSET – Control register F Clear/Set” on page 165 for information on how to access this type of status register.

- **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.

13.12.10INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x0C	CCDIF	CCCIF	CCBIF	CCAIF	–	–	ERRIF	OVFIF
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

- **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 a DMA transfer. A DMA read or write access of the corresponding CCx or CCxBUF will then clear the CCxIF and release the request.

- **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 – ERRIF: Error Interrupt Flag**

This flag is set on multiple occasions, depending on the mode of operation.

In the FRQ or PWM waveform generation mode of operation, ERRIF is set on a fault detect condition from the fault protection feature in the AWeX extention. For timer/counters which do not have the AWeX extention available, this flag is never set in FRQ or PWM waveform generation mode.

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. OVFIF is automatically cleared when the corresponding interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

OVFIF can also be used for requesting a DMA transfer. A DMA write access of CNT, PER, or PERBUF will then clear the OVFIF bit.

13.12.11TEMP – Temporary bits for 16-bit Access

The TEMP register is used for single-cycle, 16-bit access to the 16-bit timer/counter registers by the CPU. The DMA controller has a separate temporary storage register. There is one common TEMP register for all the 16-bit Timer/counter registers.

For more details, refer to “Accessing 16-bit Registers” on page 13.

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

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 DMA 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.

13.12.13CNTH – 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.

13.12.14PERL – 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.

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	1	1	1	1	1	1	1	1

- **Bit 7:0 – PER[7:0]: Period low byte**

These bits hold the LSB of the 16-bit period register.

13.12.15PERH – 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	1	1	1	1	1	1	1	1

- **Bit 7:0 – PER[15:8]: Period high byte**

These bits hold the MSB of the 16-bit period register.

13.12.16CCxL – 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 DMA.

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.

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 LSB of the 16-bit compare or capture register.

13.12.17CCxH – 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.

13.12.18PERBUFL – Timer/Counter Period Buffer 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 DMA will affect the PERBUFV flag.

Bit	7	6	5	4	3	2	1	0
+0x36	PERBUF[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 – PERBUF[7:0]: Period Buffer low byte**

These bits hold the LSB of the 16-bit period buffer register.

13.12.19PERBUFH – Timer/Counter Period Buffer High

Bit	7	6	5	4	3	2	1	0
+0x37	PERBUF[15:8]							
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 – PERBUF[15:8]: Period Buffer high byte**

These bits hold the MSB of the 16-bit period buffer register.

13.12.20CCxBUFL – 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 DMA will affect the corresponding CCxBV status bit.

Bit	7	6	5	4	3	2	1	0
	CCxBUFx[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 Buffer low byte**

These bits hold the LSB of the 16-bit compare or capture buffer register.

13.12.21CCxBUFH – Compare or Capture x Buffer register High

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 Buffer high byte**

These bits hold the MSB of the 16-bit compare or capture buffer register.

13.13 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	CLKSEL[3:0]				161
+0x01	CTRLB	CCDEN	CCCEN	CCBEN	CCAEN	–	WGMODE[2:0]			161
+0x02	CTRLC	–	–	–	–	CMPD	CMPC	CMPB	CMPA	162
+0x03	CTRLD	EVACT[2:0]			EVDLY	EVSEL[3:0]				163
+0x04	CTRLE	–	–	–	–	–	–	BYTEM		164
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	INTCTRLA	–	–	–	–	ERRINTLVL[1:0]		OVINTLVL[1:0]		164
+0x07	INTCTRLB	CCCINTLVL[1:0]		CCCINTLVL[1:0]		CCBINTLVL[1:0]		CCAINTLVL[1:0]		164
+0x08	CTRLFCLR	–	–	–	–	CMD[1:0]		LUPD	DIR	165
+0x09	CTRLFSET	–	–	–	–	CMD[1:0]		LUPD	DIR	166
+0x0A	CTRLGCLR	–	–	–	CCDBV	CCCBV	CCBBV	CCABV	PERBV	166
+0x0B	CTRLGSET	–	–	–	CCDBV	CCCBV	CCBBV	CCABV	PERBV	166
+0x0C	INTFLAGS	CCDIF	CCCIF	CCBIF	CCAIF	–	–	ERRIF	OVFIF	167
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	TEMP	TEMP[7:0]								168
+0x10 to	Reserved	–	–	–	–	–	–	–	–	
+0x20	CNTL	CNT[7:0]								168
+0x21	CNTH	CNT[15:8]								168
+0x22 to	Reserved	–	–	–	–	–	–	–	–	
+0x26	PERL	PER[7:0]								168
+0x27	PERH	PER[8:15]								169
+0x28	CCAL	CCA[7:0]								169
+0x29	CCAH	CCA[15:8]								169
+0x2A	CCBL	CCB[7:0]								169
+0x2B	CCBH	CCB[15:8]								169
+0x2C	CCCL	CCC[7:0]								169
+0x02D	CCCH	CCC[15:8]								169
+0x2E	CCDL	CCD[7:0]								169
+0x2F	CCDH	CCD[15:8]								169
+0x30 to	Reserved	–	–	–	–	–	–	–	–	
+0x36	PERBUFL	PERBUF[7:0]								170
+0x37	PERBUFH	PERBUF[15:8]								170
+0x38	CCABUFL	CCABUF[7:0]								170
+0x39	CCABUFH	CCABUF[15:8]								171
+0x3A	CCBBUFL	CCBBUF[7:0]								170
+0x3B	CCBBUFH	CCBBUF[15:8]								171
+0x3C	CCCBUFL	CCCBUF[7:0]								170
+0x3D	CCCBUFH	CCCBUF[15:8]								171
+0x3E	CCDBUFL	CCDBUF[7:0]								170
+0x3F	CCDBUFH	CCDBUF[15:8]								171

13.14 Interrupt Vector Summary

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. TC2 – 16-bit Timer/Counter Type 2

14.1 Features

- A system of two eight-bit timer/counters
  - Low-byte timer/counter
  - High-byte timer/counter
- Eight compare channels
  - Four compare channels for the low-byte timer/counter
  - Four compare channels for the high-byte timer/counter
- Waveform generation
  - Single slope pulse width modulation
- Timer underflow interrupts/events
- One compare match interrupt/event per compare channel for the low-byte timer/counter
- Can be used with the event system for count control
- Can be used to trigger DMA transactions

14.2 Overview

A timer/counter 2 is realized when a timer/counter 0 is set in split mode. It is a system of two eight-bit timer/counters, each with four compare channels. This results in eight configurable pulse width modulation (PWM) channels with individually controlled duty cycles, and is intended for applications that require a high number of PWM channels.

The two eight-bit timer/counters in this system are referred to as the low-byte timer/counter and high-byte timer/counter, respectively. The difference between them is that only the low-byte timer/counter can be used to generate compare match interrupts, events and DMA triggers.

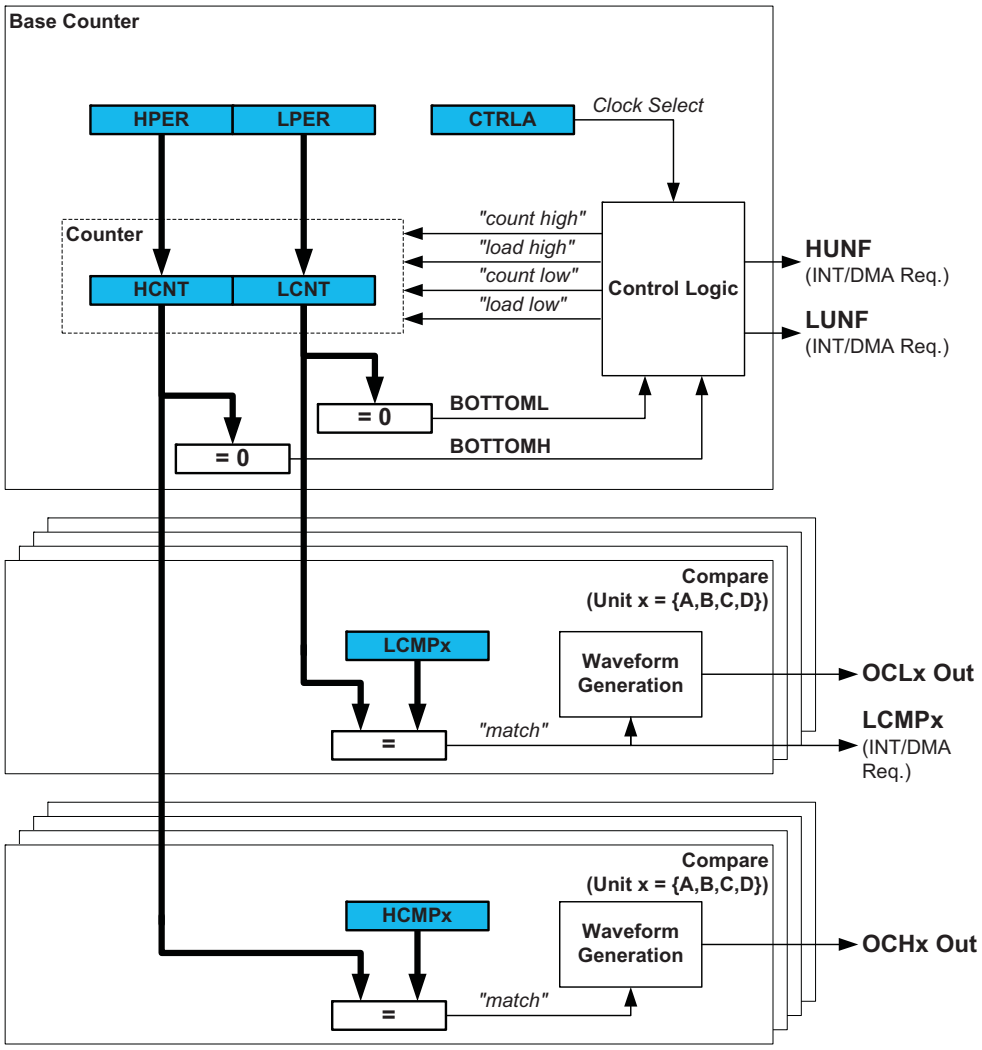
The two eight-bit timer/counters have a shared clock source and separate period and compare settings. They can be clocked and timed from the peripheral clock, with optional prescaling, or from the event system. The counters are always counting down.

The timer/counter 2 is set back to timer/counter 0 by setting it in normal mode; hence, one timer/counter can exist only as either type 0 or type 2.

A detailed block diagram of the timer/counter 2 showing the low-byte (L) and high-byte (H) timer/counter register split and compare modules is shown in Figure 14-1 on page 174.

14.3 Block Diagram

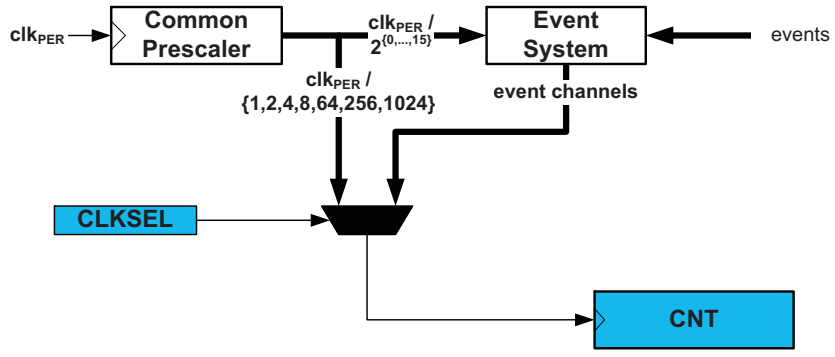
Figure 14-1. Block diagram of the 16-bit timer/counter 0 with split mode.



14.4 Clock Sources

The timer/counter can be clocked from the peripheral clock (clk_{PER}) and from the event system. Figure 14-2 shows the clock and event selection.

Figure 14-2. Clock selection.



The peripheral clock (clk_{PER}) is fed into the common prescaler (common for all timer/counters in a device). A selection of prescaler outputs from 1 to 1/1024 is directly available. In addition, the whole range of time prescalings from 1 to 2^{15} is available through the event system.

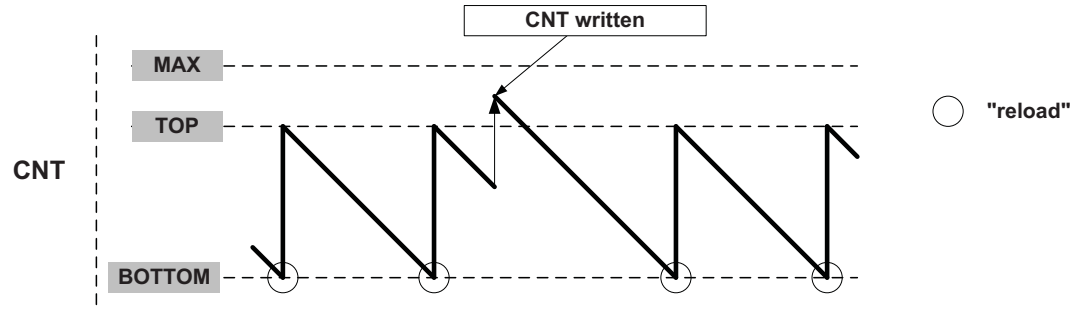
The clock selection (CLKSEL) selects one of the clock prescaler outputs or an event channel for the high-byte counter (HCNT) and low-byte counter (LCNT). By using the event system, any event source, such as an external clock signal, on any I/O pin can be used as the clock input.

By default, no clock input is selected, and the counters are not running.

14.5 Counter Operation

The counters will always count in single-slope mode. Each counter counts down for each clock cycle until it reaches BOTTOM, and then reloads the counter with the period register value at the following clock cycle.

Figure 14-3. Counter operation.

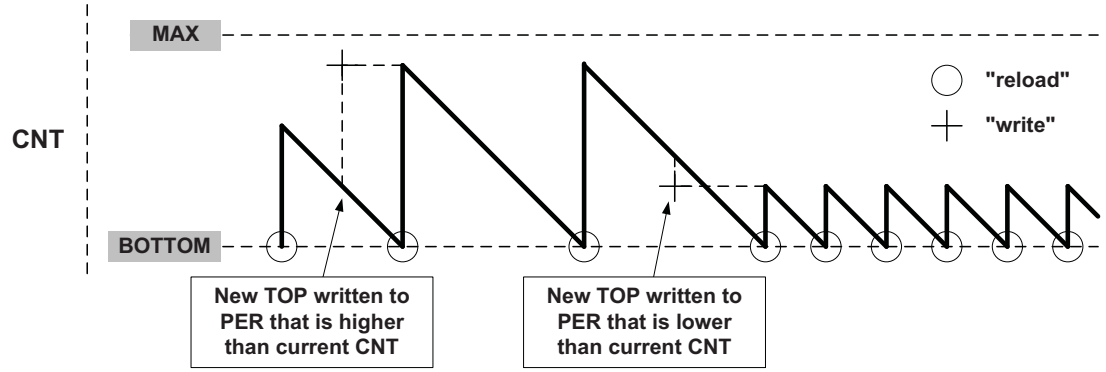


As shown in Figure 14-3, the counter can change the counter value while running. The write access has higher priority than the count clear, and reloads and will be immediate.

14.5.1 Changing the Period

The counter period is changed by writing a new TOP value to the period register. Since the counter is counting down, the period register can be written at any time without affecting the current period, as shown in Figure 14-4 on page 175. This prevents wraparound and generation of odd waveforms.

Figure 14-4. Changing the period.



14.6 Compare Channel

Each compare channel continuously compares the counter value with the CMPx register. If CNT equals CMPx, the comparator signals a match. For the low-byte timer/counter, the match will set the compare channel's interrupt flag at the next timer clock cycle, and the event and optional interrupt is generated. The high-byte timer/counter does not have compare interrupt/event.

14.6.1 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:

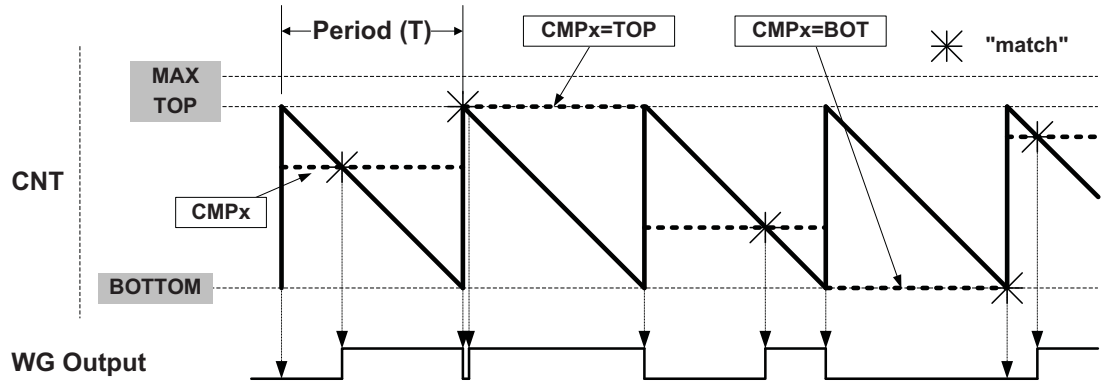
1. The compare channels to be used must be enabled. This will override the corresponding port pin output register.
2. The direction for the associated port pin must be set to output.

Inverted waveform output can be achieved by setting invert I/O on the port pin. Refer to “I/O Ports” on page 123 for more details.

14.6.2 Single-slope PWM Generation

For PWM generation, the period (T) is controlled by the PER register, while the CMPx registers control the duty cycle of the waveform generator (WG) output. Figure 14-5 on page 176 shows how the counter counts from TOP to BOTTOM, and then restarts from TOP. The WG output is set on the compare match between the CNT and CMPx registers, and cleared at BOTTOM.

Figure 14-5. Single-slope pulse width modulation.



The PER register defines the PWM resolution. The minimum resolution is two bits (PER=0x0003), and the maximum resolution is eight bits (PER=MAX).

The following equation is used to calculate the exact resolution for a single-slope PWM (R_{PWM_SS}) waveform:

$$R_{PWM\_SS} = \frac{\log(PER + 1)}{\log(2)}$$

The single, slow PWM frequency (f_{PWM_SS}) depends on the period setting (PER) and the peripheral clock frequency (f_{PER}), and it is calculated by using the following equation:

$$f_{PWM\_SS} = \frac{f_{PER}}{N(PER + 1)}$$

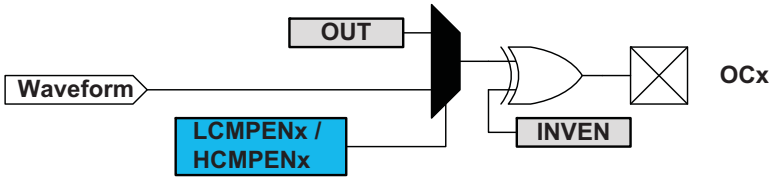
where N represents the prescaler divider used (1, 2, 4, 8, 64, 256, 1024, or event channel n).

14.6.3 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 CMP channel is enabled (LCMPENx/HCMPENx).

Figure 14-6 on page 177 shows the port override for the low- and high-byte timer/counters. For the low-byte timer/counter, CMP channels A to D will override the output value (OUTxn) of port pins 0 to 3 on the corresponding port pins (Pxn). For the high-byte timer/counter, CMP channels E to H will override port pins 4 to 7. Enabling inverted I/O on the port pin (INVENxn) inverts the corresponding WG output.

Figure 14-6. Port override for low- and high-byte timer/counters.



14.7 Interrupts and Events

The timer/counters can generate interrupts and events. The counter can generate an interrupt on underflow, and each CMP channel for the low-byte counter has a separate compare interrupt.

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 63.

14.8 DMA Support

Timer/counter underflow and compare interrupt flags can trigger a DMA transaction. The acknowledge condition that clears the flag/request is listed in Table 14-1.

Table 14-1. DMA request sources.

Request	Acknowledge	Comment
LUNFIF	DMAC writes to LCNT DMAC writes to LPER	
HUNFIF	DMAC writes to HCNT DMAC writes to HPER	
CCIF{D,C,B,A}	DMAC access of LCMP{D,C,B,A}	Output compare operation

14.9 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.

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 only be given when the timer/counter is not running (OFF).

14.10 Register Description

14.10.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	CLKSEL[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 – CLKSEL[3:0]: Clock Select**

These bits select clock source for the timer/counter according to Table 14-2. The clock select is identical for both high- and low-byte timer/counters.

Table 14-2. Clock select

CLKSEL[3:0]	Group Configuration	Description
0000	OFF	None (i.e., timer/counter in OFF state)
0001	DIV1	Prescaler: Clk_{PER}
0010	DIV2	Prescaler: $Clk_{PER}/2$
0011	DIV4	Prescaler: $Clk_{PER}/4$
0100	DIV8	Prescaler: $Clk_{PER}/8$
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]

14.10.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	HCMPEND	HCMPENC	HCMPENB	HCMPENA	LCMPEND	LCMPENC	LCMPENB	LCMPENA
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 – HCMPENx/LCMPENx: High/Low Byte Compare Enable x**

Setting these bits will enable the compare output and override the port output register for the corresponding OCn output pin.

14.10.3 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.

14.10.4 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	–	BYTEM[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 0:1 – BYTEM[1:0]: Byte Mode**

These bits select the timer/counter operation mode according to Table 14-3.

Table 14-3. Byte mode.

BYTEM[1:0]	Group Configuration	Description
00	NORMAL	Timer/counter is set to normal mode (timer/counter type 0)
01	BYTEMODE	Upper byte of the counter (HCNT) will be set to zero after each counter clock.
10	SPLITMODE	Timer/counter is split into two eight-bit timer/counters (timer/counter type 2)
11	—	Reserved

14.10.5 INTCTRLA – Interrupt Enable register A

Bit	7	6	5	4	3	2	1	0
+0x06	–	–	–	–	HUNFINTLVL[1:0]		LUNFINTLVL[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 – HUNFINTLVL[1:0]: High-byte Timer Underflow Interrupt Level**

These bits enable the high-byte timer underflow interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will be triggered when HUNFIF in the INTFLAGS register is set.

- **Bit 1:0 – LUNFINTLVL[1:0]: Low-byte Timer Underflow Interrupt Level**

These bits enable the low-byte timer underflow interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will be triggered when LUNFIF in the INTFLAGS register is set.

14.10.6 INTCTRLB – Interrupt Enable register B

Bit	7	6	5	4	3	2	1	0
+0x07	LCMPDINTLVL[1:0]		LCMPCINTLVL[1:0]		LCMPBINTLVL[1:0]		LCMPAINTLVL[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 – LCMPxINTLVL[1:0]: Low-byte Compare x Interrupt Level**

These bits enable the low-byte timer compare interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will be triggered when LCMPxIF in the INTFLAGS register is set.

14.10.7 CTRLF – Control register F

Bit	7	6	5	4	3	2	1	0
+0x08	–	–	–	–	CMD[1:0]		CMDEN[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 – CMD[1:0]: Timer/Counter Command**

These command bits are used for software control of timer/counter update, restart, and reset. The command bits are always read as zero. The CMD bits must be used together with CMDEN.

Table 14-4. Command selections.

CMD	Group Configuration	Description
00	NONE	None
01	—	Reserved
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is not in OFF state)

- **Bit 1:0 – CMDEN[1:0]: Command Enable**

These bits are used to indicate for which timer/counter the command (CMD) is valid.

Table 14-5. Command enable selections.

CMDEN	Group Configuration	Description
00	–	Reserved
01	LOW	Command valid for low-byte T/C
10	HIGH	Command valid for high-byte T/C
11	BOTH	Command valid for both low-byte and high-byte T/C

14.10.8 INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x0C	LCMPDIF	LCMPCIF	LCMPBIF	LCMPAIF	–	–	HUNFIF	LUNFIF
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

- **Bit 7:4 – LCMPxIF: Compare Channel x Interrupt Flag**

The compare interrupt flag (LCMPxIF) is set on a compare match on the corresponding CMP channel.

For all modes of operation, LCMPxIF will be set when a compare match occurs between the count register (LCNT) and the corresponding compare register (LCMPx). The LCMPxIF is automatically cleared when the corresponding interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **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 – HUNFIF: High-byte Timer Underflow Interrupt Flag**

HUNFIF is set on a BOTTOM (underflow) condition. This flag is automatically cleared when the corresponding interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **Bit 0 – LUNFIF: Low-byte Timer Underflow Interrupt Flag**

LUNFIF is set on a BOTTOM (underflow) condition. This flag is automatically cleared when the corresponding interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

14.10.9 LCNT – Low-byte Count 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]**

LCNT contains the eight-bit counter value for the low-byte timer/counter. The CPU and DMA write accesses have priority over count, clear, or reload of the counter.

14.10.10HCNT – High-byte Count register

Bit	7	6	5	4	3	2	1	0
+0x21	HCNT[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 – HCNT[7:0]**

HCNT contains the eight-bit counter value for the high-byte timer/counter. The CPU and DMA write accesses have priority over count, clear, or reload of the counter.

14.10.11LPER – Low-byte Period register

Bit	7	6	5	4	3	2	1	0
+0x27	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]**

LPER contains the eight-bit period value for the low-byte timer/counter.

14.10.12HPER – High-byte Period register

Bit	7	6	5	4	3	2	1	0
+0x26	HPER[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 – HPER[7:0]**

HPER contains the eight-bit period for the high-byte timer/counter.

14.10.13LCMPx – Low-byte Compare register x

Bit	7	6	5	4	3	2	1	0
	LCMPx[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 – LCMPx[7:0], x=[A, B, C, D]**

LCMPx contains the eight-bit compare value for the low-byte timer/counter.

These registers are all continuously compared to the counter value. Normally, the outputs from the comparators are then used for generating waveforms.

14.10.14HCMPx – High-byte Compare register x

Bit	7	6	5	4	3	2	1	0
	HCMPx[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 – HCMPx[7:0], x=[A, B, C, D]**

HCMPx contains the eight-bit compare value for the high-byte timer/counter.

These registers are all continuously compared to the counter value. Normally the outputs from the comparators are then used for generating waveforms.

14.11 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	CLKSEL[3:0]				178
+0x01	CTRLB	HCMPDEN	HCMPCEN	HCMPBEN	HCMPAEN	LCMPDEN	LCMPCEN	LCMPBEN	LCMPAEN	178
+0x02	CTRLC	HCMPD	HCMPC	HCMPB	HCMPA	LCMPD	LCMPC	LCMPB	LCMPA	179
+0x03	Reserved	–	–	–	–	–	–	–	–	
+0x04	CTRLE	–	–	–	–	–	–	BYTEM[1:0]		179
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	INTCTRLA	–	–	–	–	HUNFINTLVL[1:0]		LUNFINTLVL[1:0]		180
+0x07	INTCTRLB	LCMPDINTLVL[1:0]		LCMPCINTLVL[1:0]		LCMPBINTLVL[1:0]		LCMPAINTLVL[1:0]		180
+0x08	Reserved	–	–	–	–	–	–	–	–	
+0x09	CTRLF	–	–	–	–	CMD[1:0]		CMDEN[1:0]		180
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	INTFLAGS	LCMPDIF	LCMPCIF	LCMPBIF	LCMPAIF	–	–	HUNFIF	LUNFIF	181
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10 to	Reserved	–	–	–	–	–	–	–	–	
+0x20	LCNT	Low-byte Timer/Counter Count Register								182
+0x21	HCNT	High-byte Timer/Counter Count Register								182
+0x22 to	Reserved	–	–	–	–	–	–	–	–	
+0x26	LPER	Low-byte Timer/Counter Period Register								182
+0x27	HPER	High-byte Timer/Counter Period Register								183
+0x28	LCMPA	Low-byte Compare Register A								183
+0x29	HCMPA	High-byte Compare Register A								183
+0x2A	LCMPB	Low-byte Compare Register B								183
+0x2B	HCMPB	High-byte Compare Register B								183
+0x2C	LCMPC	Low-byte Compare Register C								183
+0x02D	HCMPC	High-byte Compare Register C								183
+0x2E	LCMPD	Low-byte Compare Register D								183
+0x2F	HCMPD	High-byte Compare Register D								183
+0x30 to	Reserved	–	–	–	–	–	–	–	–	

14.12 Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	LUNF_vect	Low-byte Timer/counter underflow interrupt vector offset
0x02	HUNF_vect	High-byte Timer/counter underflow interrupt vector offset
0x4	LCMPA_vect	Low-byte Timer/counter compare channel A interrupt vector offset
0x6	LCMPB_vect	Low-byte Timer/counter compare channel B interrupt vector offset
0x8	LCMPC_vect	Low-byte Timer/counter compare channel C interrupt vector offset
0x0A	LCMPD_vect	Low-byte Timer/counter compare channel D interrupt vector offset

15. AWeX – Advanced Waveform Extension

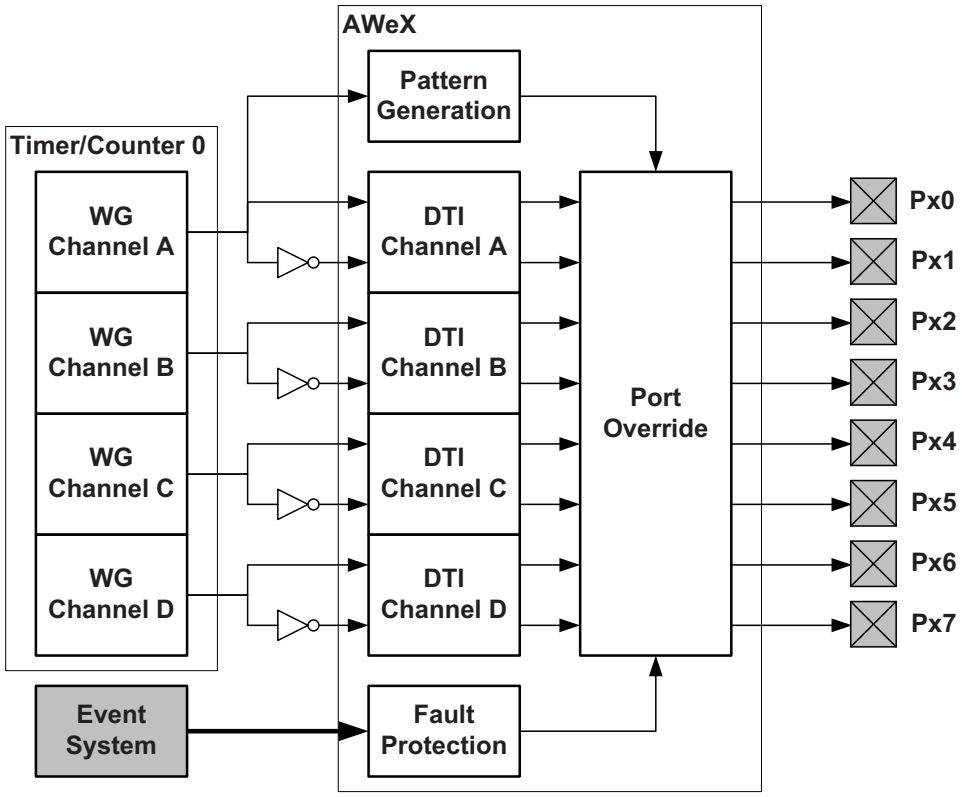
15.1 Features

- Waveform output with complementary output from each compare channel
- Four dead-time insertion (DTI) units
  - 8-bit resolution
  - Separate high and low side dead-time setting
  - Double buffered dead time
  - Optionally halts timer during dead-time insertion
- Pattern generation unit creating synchronised bit pattern across the port pins
  - Double buffered pattern generation
  - Optional distribution of one compare channel output across the port pins
- Event controlled fault protection for instant and predictable fault triggering

15.2 Overview

The advanced waveform extension (AWeX) provides extra functions to the timer/counter in waveform generation (WG) modes. It is primarily intended for use with different types of motor control and other power control applications. It enables low- and high side output with dead-time insertion and fault protection for disabling and shutting down external drivers. It can also generate a synchronized bit pattern across the port pins.

Figure 15-1. Advanced waveform extention and closely related peripherals (grey).



As shown in Figure 15-1 on page 185, each of the waveform generator outputs from timer/counter 0 are split into a complimentary pair of outputs when any AWeX features are enabled. These output pairs go through a dead-time insertion (DTI) unit that generates the non-inverted low side (LS) and inverted high side (HS) of the WG output with dead-time insertion between LS and HS switching. The DTI output will override the normal port value according to the port

override setting. Refer to “I/O Ports” on page 123 for more details.

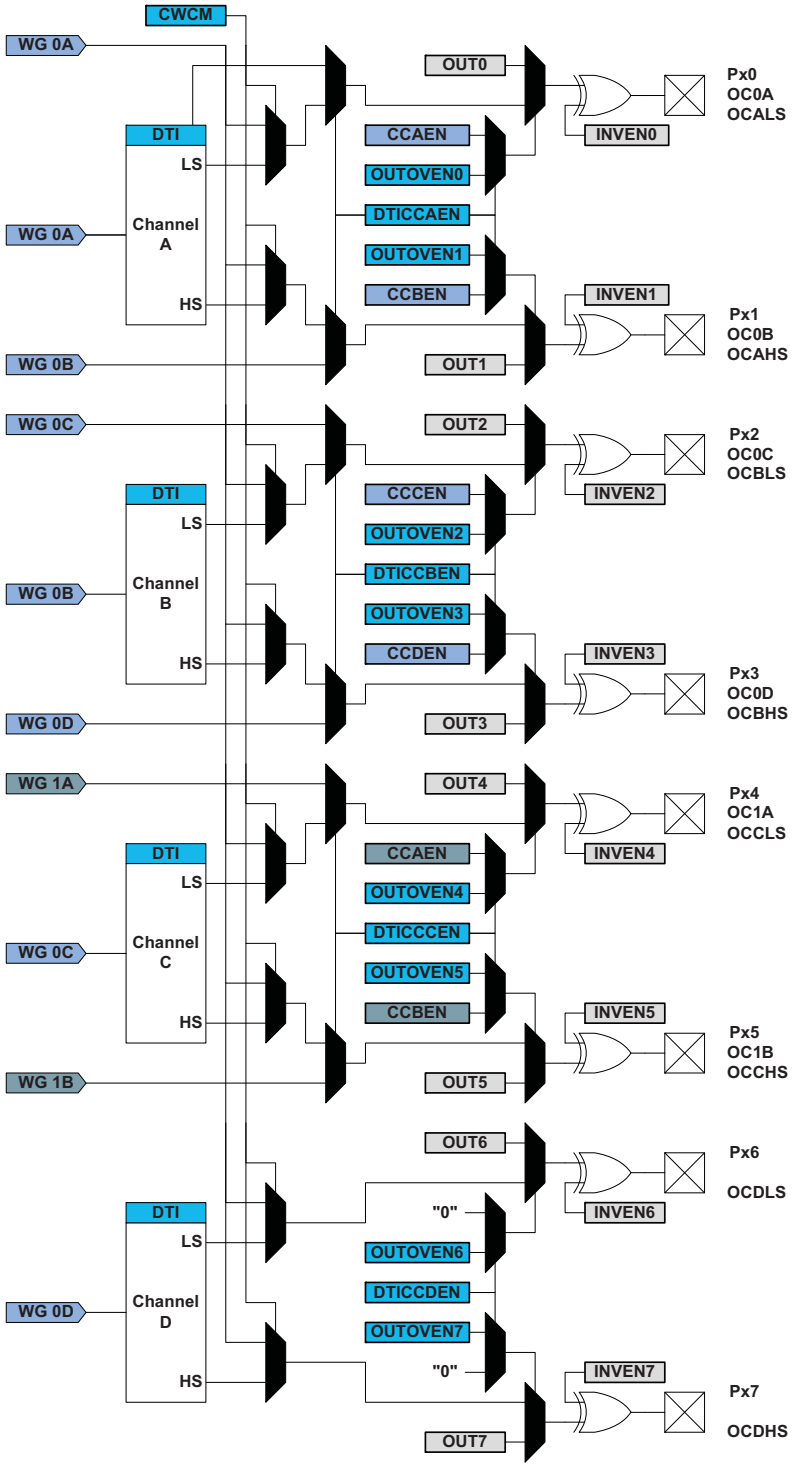
The pattern generation unit can be used to generate a synchronized bit pattern on the port it is connected to. In addition, the WG output from compare channel A can be distributed to and override all the port pins. When the pattern generator unit is enabled, the DTI unit is bypassed.

The fault protection unit is connected to the event system, enabling any event to trigger a fault condition that will disable the AWeX output. The event system ensures predictable and instant fault reaction, and gives flexibility in the selection of fault triggers.

15.3 Port Override

The port override logic is common for all the timer/counter extensions. Figure 15-2 on page 187 shows a schematic diagram of the port override logic. When the dead-time enable (DTIENx) bit is set, the timer/counter extension takes control over the pin pair for the corresponding channel. Given this condition, the output override enable (OOE) bits take control over the CCxEN bits.

Figure 15-2. Timer/counter extensions and port override logic.



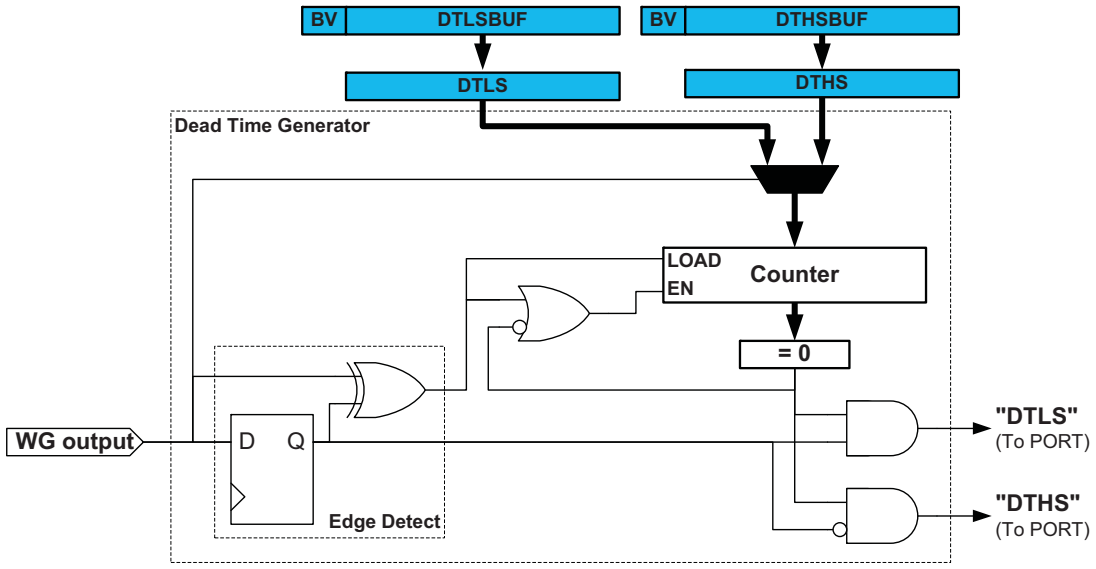
15.4 Dead-time Insertion

The dead-time insertion (DTI) unit generates OFF time where the non-inverted low side (LS) and inverted high side (HS) of the WG output are both low. This OFF time is called dead time, and dead-time insertion ensures that the LS and HS never switch simultaneously.

The DTI unit consists of four equal dead-time generators, one for each compare channel in timer/counter 0. Figure 15-3 on page 188 shows the block diagram of one DTI generator. The four channels have a common register that controls the

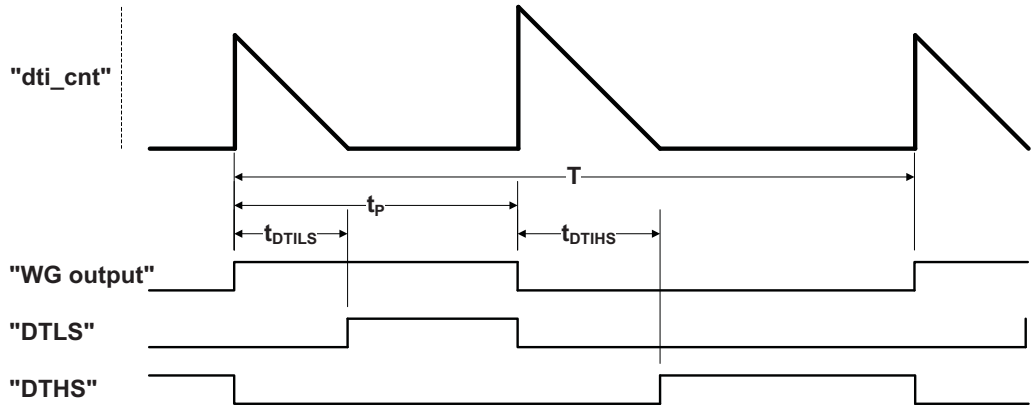
dead time. The high side and low side have independent dead-time setting, and the dead-time registers are double buffered.

Figure 15-3. Dead-time generator block diagram.



As shown in Figure 15-4 on page 188, the 8-bit dead-time counter is decremented by one for 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 a change is detected on the WG output, the dead-time counter is reloaded according to the edge of the input. A positive edge initiates a counter reload of the DTLS register, and a negative edge a reload of DTHS register.

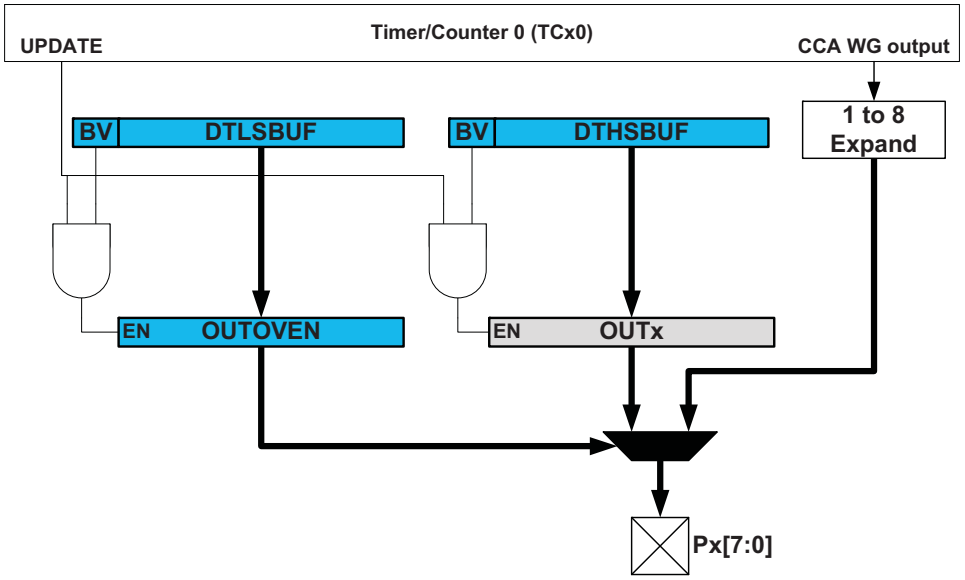
Figure 15-4. Dead-time generator timing diagram.



15.5 Pattern Generation

The pattern generator unit reuses the DTI registers to produce a synchronized bit pattern across the port it is connected to. In addition, the waveform generator output from compare channel A (CCA) can be distributed to and override all the port pins. These features are primarily intended for handling the commutation sequence in brushless DC motor (BLDC) and stepper motor applications. A block diagram of the pattern generator is shown in "Pattern generator block diagram." on page 189. For each port pin where the corresponding OOE bit is set, the multiplexer will output the waveform from CCA.

Figure 15-5. Pattern generator block diagram.



As with the other timer/counter double buffered registers, the register update is synchronized to the UPDATE condition set by the waveform generation mode. If the synchronization provided is not required by the application, the application code can simply access the DTIOE and PORTx registers directly.

The pin directions must be set for any output from the pattern generator to be visible on the port.

15.6 Fault Protection

The fault protection feature enables fast and deterministic action when a fault is detected. The fault protection is event controlled. Thus, any event from the event system can be used to trigger a fault action, such as over-current indication from analog comparator or ADC measurements.

When fault protection is enabled, an incoming event from any of the selected event channels can trigger the event action. Each event channel can be separately enabled as a fault protection input, and the specified event channels will be ORed together, allowing multiple event sources to be used for fault protection at the same time.

15.6.1 Fault Actions

When a fault is detected, the direction clear action will clear the direction (DIR) register in the associated port, setting all port pins as tri-stated inputs.

The fault detection flag is set, the timer/counter's error interrupt flag is set, and the optional interrupt is generated.

There is maximum of two peripheral clock cycles from when an event occurs in a peripheral until the fault protection triggers the event action. Fault protection is fully independent of the CPU and DMA, but requires the peripheral clock to run.

15.6.2 Fault Restore Modes

How the AWeX and timer/counter return from the fault state to normal operation after a fault, when the fault condition is no longer active, can be selected from one of two different modes:

- In latched mode, the waveform output will remain in the fault state until the fault condition is no longer active and the fault detect flag has been cleared by software. When both of these conditions are met, the waveform output will return to normal operation at the next UPDATE condition.
- In cycle-by-cycle mode the waveform output will remain in the fault state until the fault condition is no longer active. When this condition is met, the waveform output will return to normal operation at the next UPDATE condition.

When returning from a fault state the DIR[7:0] bits corresponding to the enabled DTI channels are restored. OUTOVEN is unaffected by the fault except that writing to the register from software is blocked.

The UPDATE condition used to restore normal operation is the same as the one in the timer/counter.

15.6.3 Change Protection

To avoid unintentional changes in the fault protection setup, all the control registers in the AWeX extension can be protected by writing the corresponding lock bit in the advanced waveform extension lock register. For more details, refer to “I/O Memory Protection” on page 25 and “AWEXLOCK – Advanced Waveform Extension Lock register” on page 44.

When the lock bit is set, control register A, the output override enable register, and the fault detection event mask register cannot be changed.

To avoid unintentional changes in the fault event setup, it is possible to lock the event system channel configuration by writing the corresponding event system lock register. For more details, refer to “I/O Memory Protection” on page 25 and “EVSYSLOCK – Event System Lock register” on page 43.

15.6.4 On-Chip Debug

When fault detection is enabled, an on-chip debug (OCD) system receives a break request from the debugger, which will by default function as a fault source. When an OCD break request is received, the AWeX and corresponding timer/counter will enter a fault state, and the specified fault action will be performed.

After the OCD exits from the break condition, normal operation will be started again. In cycle-by-cycle mode, the waveform output will start on the first UPDATE condition after exit from break, while in latched mode, the fault condition flag must be cleared in software before the output will be restored. This feature guarantees that the output waveform enters a safe state during a break.

It is possible to disable this feature.

15.7 Register Description

15.7.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	PGM	CWCM	DTICCDEN	DTICCCEN	DTICCBEN	DTICCAEN
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 – PGM: Pattern Generation Mode**

Setting this bit enables the pattern generation mode. This will override the DTI, and the pattern generation reuses the dead-time registers for storing the pattern.

- **Bit 4 – CWCM: Common Waveform Channel Mode**

If this bit is set, the CC channel A waveform output will be used as input for all the dead-time generators. CC channel B, C, and D waveforms will be ignored.

- **Bit 3:0 – DTICCxEN: Dead-Time Insertion CCx Enable**

Setting these bits enables the dead-time generator for the corresponding CC channel. This will override the timer/counter waveform outputs.

15.7.2 FDEMASK – Fault Detect Event Mask register

Bit	7	6	5	4	3	2	1	0
+0x02	FDEVMASK[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 – FDEVMASK[7:0]: Fault Detect Event Mask**

These bits enable the corresponding event channel as a fault condition input source. Events from all event channels will be ORed together, allowing multiple sources to be used for fault detection at the same time. When a fault is detected, the fault detect flag (FDF) is set and the fault detect action (FDACT) will be performed.

15.7.3 FDCTRL - Fault Detection Control register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	FDDBD	–	FDMODE	FDACT[1:0]	
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 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 – FDDBD: Fault Detection on Debug Break Detection**

By default, when this bit is cleared and fault protection is enabled, and OCD break request is treated as a fault. When this bit is set, an OCD break request will not trigger a fault condition.

- **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 – FDMODE: Fault Detection Restart Mode**

This bit sets the fault protection restart mode. When this bit is cleared, latched mode is used, and when it is set, cycle-by-cycle mode is used.

In latched mode, the waveform output will remain in the fault state until the fault condition is no longer active and the FDF has been cleared by software. When both conditions are met, the waveform output will return to normal operation at the next UPDATE condition.

In cycle-by-cycle mode, the waveform output will remain in the fault state until the fault condition is no longer active. When this condition is met, the waveform output will return to normal operation at the next UPDATE condition.

- **Bit 1:0 – FDACT[1:0]: Fault Detection Action**

These bits define the action performed, according to Table 15-1, when a fault condition is detected.

Table 15-1. Fault action.

FDACT[1:0]	Group Configuration	Description
00	NONE	None (fault protection disabled)
01	–	Reserved
10	–	Reserved
11	CLEARDIR	Clear all direction (DIR) bits which correspond to the enabled DTI channel(s); i.e., tri-state the outputs

15.7.4 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	FDF	DTHSBUFV	DTLSBUFV
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 – FDF: Fault Detect Flag**

This flag is set when a fault detect condition is detected; i.e., when an event is detected on one of the event channels enabled by FDEVMASK. This flag is cleared by writing a one to its bit location.

- **Bit 1 – DTHSBUFV: Dead-time High Side Buffer Valid**

If this bit is set, the corresponding DT buffer is written and contains valid data that will be copied into the DTLS register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter unit's lock update (LUPD) flag also affects the update for dead-time buffers.

- **Bit 0 – DTLSBUFV: Dead-time Low Side Buffer Valid**

If this bit is set, the corresponding DT buffer is written and contains valid data that will be copied into the DTHS register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter unit's lock update (LUPD) flag also affects the update for dead-time buffers.

15.7.5 DTBOTH – Dead-time Concurrent Write to Both Sides

Bit	7	6	5	4	3	2	1	0
+0x06	DTBOTH[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 – DTBOTH: Dead-time Both Sides**

Writing to this register will update the DTHS and DTLS registers at the same time (i.e., at the same I/O write access).

15.7.6 DTBOTHBUF – Dead-time Concurrent Write to Both Sides Buffer register

Bit	7	6	5	4	3	2	1	0
+0x07	DTBOTHBUF[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 – DTBOTHBUF: Dead-time Both Sides Buffer**

Writing to this memory location will update the DTHSBUF and DTLSBUF registers at the same time (i.e., at the same I/O write access).

15.7.7 DTLS – Dead-time Low Side register

Bit	7	6	5	4	3	2	1	0
+0x08	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: Dead-time Low Side**

This register holds the number of peripheral clock cycles for the dead-time low side.

15.7.8 DTHS – Dead-time High Side register

Bit	7	6	5	4	3	2	1	0
+0x09	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: Dead-time High Side**

This register holds the number of peripheral clock cycles for the dead-time high side.

15.7.9 DTLSBUF – Dead-time Low Side Buffer register

Bit	7	6	5	4	3	2	1	0
+0x0A	DTLSBUF[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 – DTLSBUF: Dead-time Low Side Buffer**

This register is the buffer for the DTLS register. If double buffering is used, valid content in this register is copied to the DTLS register on an UPDATE condition.

15.7.10 DTHSBUF – Dead-time High Side Buffer register

Bit	7	6	5	4	3	2	1	0
+0x0B	DTHSBUF[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 – DTHSBUF: Dead-time High Side Buffer**

This register is the buffer for the DTHS register. If double buffering is used, valid content in this register is copied to the DTHS register on an UPDATE condition.

15.7.11 OUTOVEN – Output Override Enable register

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

Note: 1. Can be written only if the fault detect flag (FDF) is zero.

- **Bit 7:0 – OUTOVEN[7:0]: Output Override Enable**

These bits enable override of the corresponding port output register (i.e., one-to-one bit relation to pin position). The port direction is not overridden.

15.8 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Pag
+0x00	CTRL	–	–	PGM	CWCM	DTICDAE	DTICCCE	DTICCBEN	DTICCAEN	191
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	FDEMASK	FDEVMASK[7:0]								191
+0x03	FDCTRL	–	–	–	FDDBD	–	FDMODE	FDACT[1:0]		192
+0x04	STATUS	–	–	–	–	–	FDF	DTBHSV	DTBLSV	193
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	DTBOTH	DTBOTH[7:0]								193
+0x07	DTBOTHBUF	DTBOTHBUF[7:0]								193
+0x08	DTLS	DTLS[7:0]								194
+0x09	DTHS	DTHS[7:0]								194
+0x0A	DTLSBUF	DTLSBUF[7:0]								194
+0x0B	DTHSBUF	DTHSBUF[7:0]								194
+0x0C	OUTOVEN	OUTOVEN[7:0]								195

16. Hi-Res – High-Resolution Extension

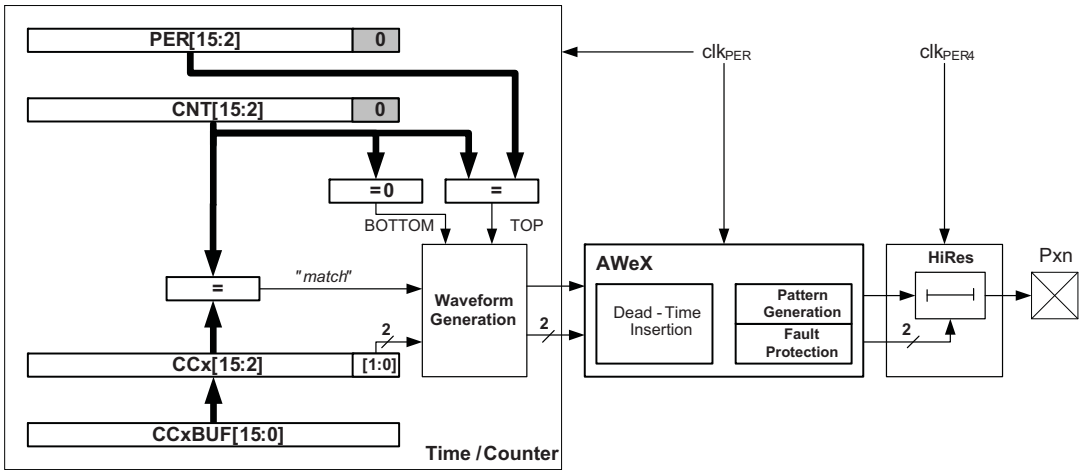
16.1 Features

- Increases waveform generator resolution up to 8x (3 bits)
- Supports frequency, single-slope PWM, and dual-slope PWM generation
- Supports the AWeX when this is used for the same timer/counter

16.2 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 AWeX 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 79 for more details.

Figure 16-1. 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.

16.3 Register Description

16.3.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	HRPLUS	HREN[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

- **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 – HRPLUS: High Resolution Plus**

Setting this bit enables high resolution plus. 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.

- **Bit 1:0 – HREN[1:0]: High Resolution Enable**

These bits enables the high-resolution mode for a timer/counter according to Table 16-1.

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 16-1. High resolution enable.

HREN[1:0]	High Resolution Enabled
00	None
01	Timer/counter 0
10	Timer/counter 1
11	Both timer/counters

16.4 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	–	HRPLUS	HREN[1:0]		197

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

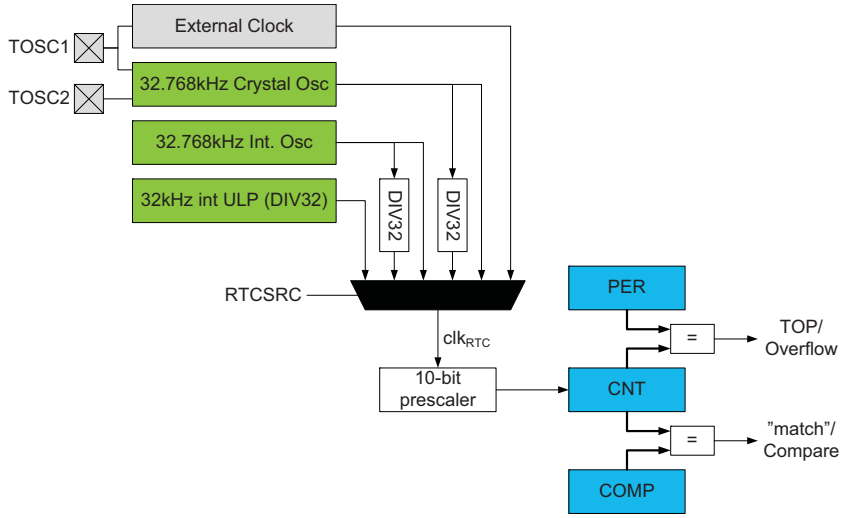
17.2 Overview

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 reference 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 timeout period is more than18 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.

Figure 17-1. Real-time counter overview.



17.2.1 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. Refer to “RTCCTRL – RTC Control register” on page 85 for selecting the asynchronous clock source for the RTC.

17.2.2 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.

17.3 Register Descriptions

17.3.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	PRESCALER[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 – PRESCALER[2:0]: Clock Prescaling factor**

These bits define the prescaling factor for the RTC clock according to Table 17-1.

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

17.3.2 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

- **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 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 synchronisation is complete

17.3.3 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

- **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 – COMPINTLVL[1:0]: Compare Match Interrupt Enable**

These bits enable the RTC compare match interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will trigger when COMPIF in the INTFLAGS register is set.

- **Bit 1:0 – OVFINTLVL[1:0]: Overflow Interrupt Enable**

These bits enable the RTC overflow interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will trigger when OVFIF in the INTFLAGS register is set.

17.3.4 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

- **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 – COMPIF: 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: Overflow 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.

17.3.5 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

- **Bit 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. For more details, refer to “Accessing 16-bit Registers” on page 13.

17.3.6 CNTL – Counter 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. Refer to “Accessing 16-bit Registers” on page 13 for details.

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 200 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

- **Bit 7:0 – CNT[7:0]: Counter Value low byte**

These bits hold the LSB of the 16-bit real-time counter value.

17.3.7 CNTH – Counter 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

- **Bit 7:0 – CNT[15:8]: Counter Value high byte**

These bits hold the MSB of the 16-bit real-time counter value.

17.3.8 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. Refer to “Accessing 16-bit Registers” on page 13 for details.

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 200 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	1	1	1	1	1	1	1	1

- **Bit 7:0 – PER[7:0]: Period low byte**

These bits hold the LSB of the 16-bit RTC TOP value.

17.3.9 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	1	1	1	1	1	1	1	1

- **Bits 7:0 – PER[15:8]: Period high byte**

These bits hold the MSB of the 16-bit RTC TOP value.

17.3.10 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. Refer “Accessing 16-bit Registers” on page 13 for details.

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 200 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

- **Bit 7:0 – COMP[7:0]: Compare value low byte**

These bits hold the LSB of the 16-bit RTC compare value.

17.3.11 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

- **Bit 7:0 – COMP[15:8]: Compare value high byte**

These bits hold the MSB of the 16-bit RTC compare value.

17.4 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	–	PRESCALER[2:0]			200
+0x01	STATUS	–	–	–	–	–	–	–	SYNCBUSY	200
+0x02	INTCTRL	–	–	–	–	COMPINTLVL[1:0]		OVFINTLVL[1:0]		201
+0x03	INTFLAGS	–	–	–	–	–	–	COMPIF	OVFIF	201
+0x04	TEMP	TEMP[7:0]								202
+0x08	CNTL	CNT[7:0]								202
+0x09	CNTH	CNT[15:8]								202
+0x0A	PERL	PER[7:0]								203
+0x0B	PERH	PER[15:8]								203
+0x0C	COMPL	COMP[7:0]								204
+0x0D	COMPH	COMP[15:8]								203

17.5 Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	OVF_vect	Real-time counter overflow interrupt vector
0x02	COMP_vect	Real-time counter compare match interrupt vector

18. USB – Universal Serial Bus Interface

18.1 Features

- USB 2.0 full speed (12Mbps) and low speed (1.5Mbps) device compliant interface
- Integrated on-chip USB transceiver, no external components needed
- 16 endpoint addresses with full endpoint flexibility for up to 31 endpoints
  - One input endpoint per endpoint address
  - One output endpoint per endpoint address
- Endpoint address transfer type selectable to
  - Control transfers
  - Interrupt transfers
  - Bulk transfers
  - Isochronous transfers
- Configurable data payload size per endpoint, up to 1023 bytes
- Endpoint configuration and data buffers located in internal SRAM
  - Configurable location for endpoint configuration data
  - Configurable location for each endpoint's data buffer
- Built-in direct memory access (DMA) to internal SRAM for:
  - Endpoint configurations
  - Reading and writing endpoint data
- Ping-pong operation for higher throughput and double buffered operation
  - Input and output endpoint data buffers used in a single direction
  - CPU/DMA controller can update data buffer during transfer
- Multipacket transfer for reduced interrupt load and software intervention
  - Data payload exceeding maximum packet size is transferred in one continuous transfer
  - No interrupts or software interaction on packet transaction level
- Transaction complete FIFO for workflow management when using multiple endpoints
  - Tracks all completed transactions in a first-come, first-served work queue
- Clock selection independent of system clock source and selection
- Minimum 1.5MHz CPU clock required for low speed USB operation
- Minimum 12MHz CPU clock required for full speed operation
- Connection to event system
- On chip debug possibilities during USB transactions

18.2 Overview

The USB module is a USB 2.0 full speed (12Mbps) and low speed (1.5Mbps) device compliant interface.

The USB supports 16 endpoint addresses. All endpoint addresses have one input and one output endpoint, for a total of 31 configurable endpoints and one control endpoint. Each endpoint address is fully configurable and can be configured for any of the four transfer types: control, interrupt, bulk, or isochronous. The data payload size is also selectable, and it supports data payloads up to 1023 bytes.

No dedicated memory is allocated for or included in the USB module. Internal SRAM is used to keep the configuration for each endpoint address and the data buffer for each endpoint. The memory locations used for endpoint configurations and data buffers are fully configurable. The amount of memory allocated is fully dynamic, according to the number of endpoints in use and the configuration of these. The USB module has built-in direct memory access (DMA), and will read/write data from/to the SRAM when a USB transaction takes place.

To maximize throughput, an endpoint address can be configured for ping-pong operation. When done, the input and output endpoints are both used in the same direction. The CPU or DMA controller can then read/write one data buffer while the USB module writes/reads the others, and vice versa. This gives double buffered communication.

Multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without software intervention. This reduces the CPU intervention and the interrupts needed for USB transfers.

For low-power operation, the USB module can put the microcontroller into any sleep mode when the USB bus is idle and a suspend condition is given. Upon bus resumes, the USB module can wake up the microcontroller from any sleep mode.

Figure 18-1. USB OUT transfer: data packet from host to USB device.

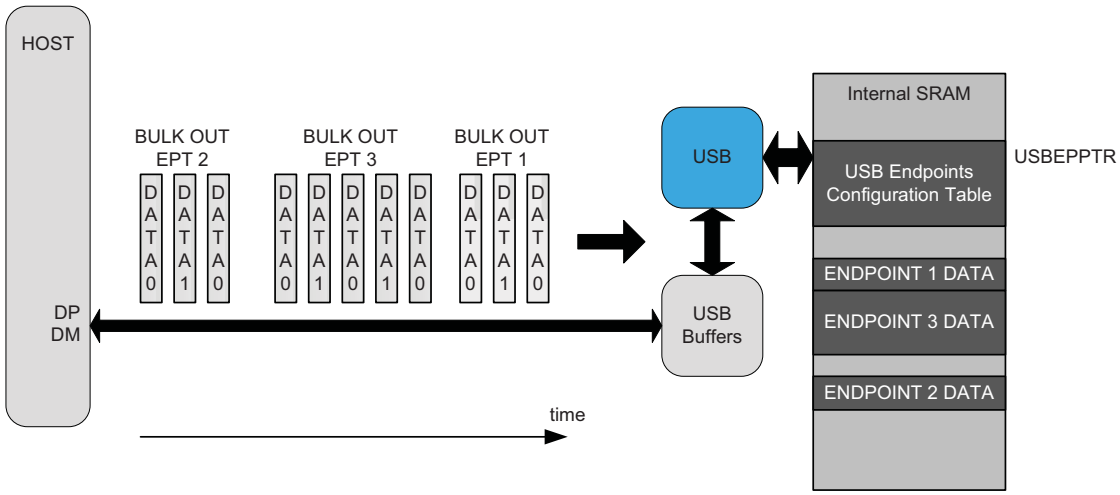
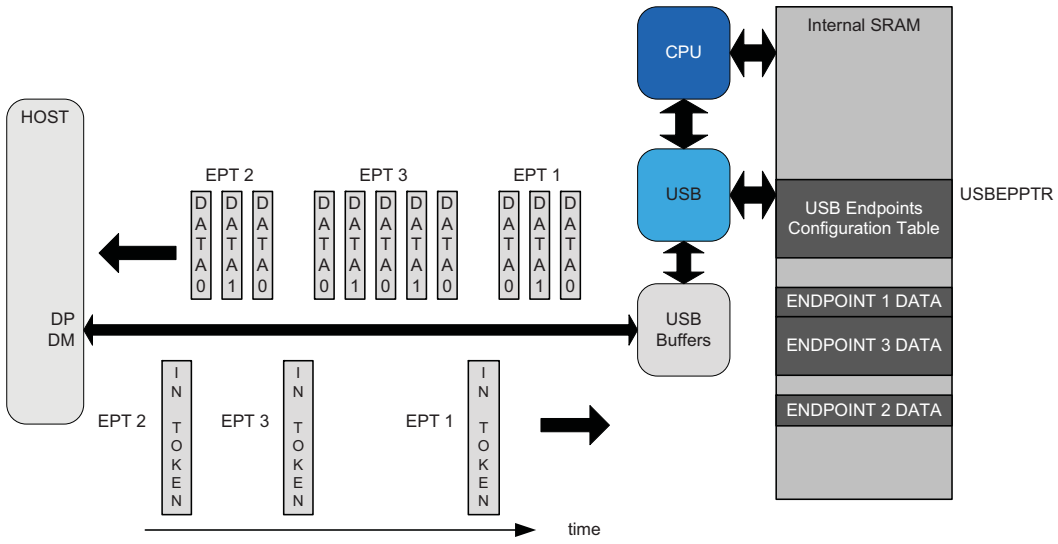


Figure 18-2. USB IN transfer: data packet from USB device to host after request from host.



18.3 Operation

This section gives an overview of the USB module operation during normal transactions. For general details on USB and the USB protocol, please refer to http://www.usb.org and the USB specification documents.

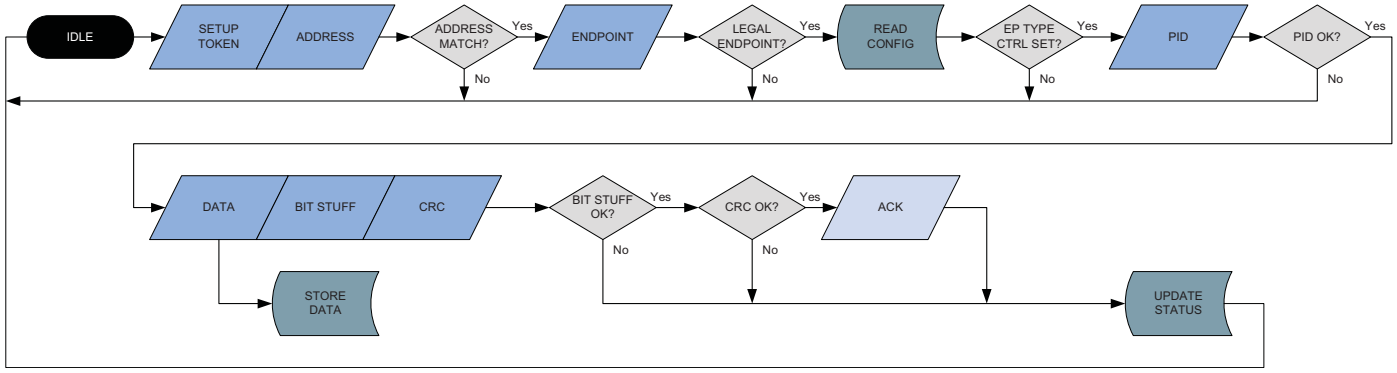
18.3.1 Start of Frame

When a start of frame (SOF) token is detected and storing of the frame numbers is enabled, the frame number from the token is stored in the frame number register (FRAMENUM) and the start of frame interrupt flag (SOFIF) in the interrupt flag B clear/set register (INTFLAGSBCLR/SET) is set. If there was a CRC or bit-stuff error, the frame error (FRAMEERR) flag in FRAMENUM is set.

18.3.2 SETUP

When a SETUP token is detected, the USB module fetches the endpoint control register (CTRL) from the addressed output endpoint in the endpoint configuration table. If the endpoint type is not set to control, the USB module returns to idle and waits for the next token packet.

Figure 18-3. SETUP transaction.



The USB module then fetches the endpoint data pointer register (DATAPTR) and waits for a DATA0 packet. If a PID error or any other PID than DATA0 is detected, the USB module returns to idle and waits for the next token packet.

The incoming data are written to the data buffer pointed to by DATAPTR. If a bit-stuff error is detected in the incoming data, the USB module returns to idle and waits for the next token packet. If the number of received data bytes exceeds the endpoint's maximum data payload size, as specified by the data size (SIZE) in the endpoint CTRL register, the remaining received data bytes are discarded. The packet will still be checked for bit-stuff and CRC errors. Software must never report a maximum data payload size to the host that is greater than specified in SIZE. If there was a bit-stuff or CRC error in the packet, the USB module returns to idle and waits for the next token packet.

If data was successfully received, an ACK handshake is returned to the host, and the number of received data bytes, excluding the CRC, is written to the endpoint byte counter (CNT). If the number of received data bytes is the maximum data payload specified by SIZE, no CRC data are written in the data buffer. If the number of received data bytes is the maximum data payload specified by SIZE minus one, only the first CRC data byte is written in the data buffer. If the number of received data bytes is equal or less than the data byte payload specified by SIZE minus two, the two CRC data bytes are written in the data buffer.

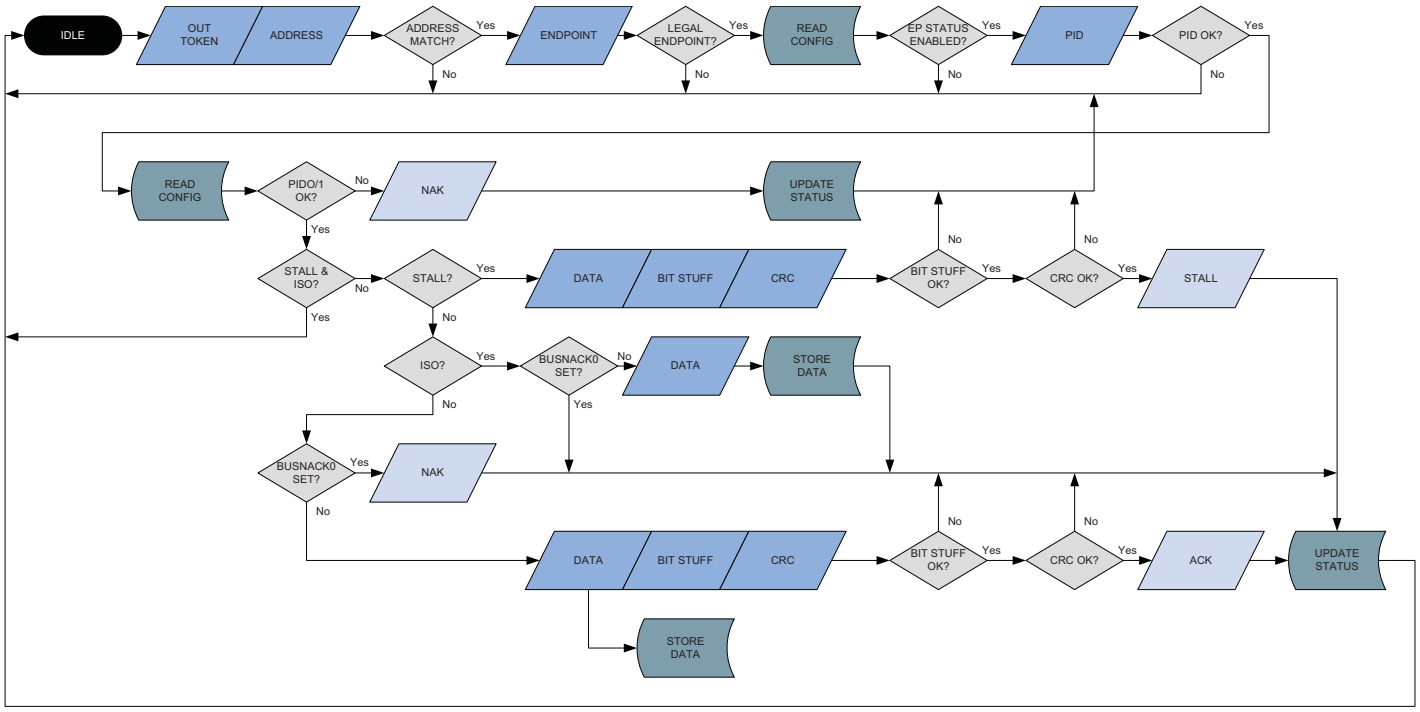
Finally, the setup transaction complete flag (SETUP), data buffer 0 not acknowledge flag (NACK0), and data toggle flag (TOGGLE) are set, while the remaining flags in the endpoint status register (STATUS) are cleared for the addressed input and output endpoints. The setup transaction complete interrupt flag (SETUPIF) in INTFLAGSBCLR/SET is set. The STALL flag in the endpoint CTRL register is cleared for the addressed input and output endpoints.

When a SETUP token is detected and the device address of the token packet does not match that of the endpoint, the packet is discarded, and the USB module returns to idle and waits for the next token packet.

18.3.3 OUT

When an OUT token is detected, the USB module fetches the endpoint CTRL and STATUS register data from the addressed output endpoint in its endpoint configuration table. If the endpoint is disabled, the USB module returns to idle and waits for the next token packet.

Figure 18-4. OUT transaction.



The USB module then fetches the endpoint DATAPTR register and waits for a DATA0 or DATA1 packet. If a PID error or any other PID than DATA0 or DATA1 is detected, the USB module returns to idle and waits for the next token packet.

If the STALL flag in the endpoint CTRL register is set, the incoming data are discarded. If the endpoint is not isochronous, and the bit stuffing and CRC of the received data are OK, a STALL handshake is returned to the host, and the STALL interrupt flag is set.

For isochronous endpoints, data from both a DATA0 and DATA1 packet will be accepted. For other endpoint types, the PID is checked against TOGGLE. If they don't match, the incoming data are discarded and a NAK handshake is returned to the host. If BUSNACK0 is set, the incoming data are discarded. The overflow flag (OVF) in the endpoint STATUS register and the overflow interrupt flag (OVFIF) in the INTFLAGSASET/CLR register are set. If the endpoint is not isochronous, a NAK handshake is returned to the host.

The incoming data are written to the data buffer pointed to by DATAPTR. If a bit-stuff error is detected in the incoming data, the USB module returns to idle and waits for the next token packet. If the number of received data bytes exceeds the maximum data payload specified by SIZE, the remaining received data bytes are discarded. The packet will still be checked for bit-stuff and CRC errors. If there was a bit-stuff or CRC error in the packet, the USB module returns to idle and waits for the next token packet.

If the endpoint is isochronous and there was a bit-stuff or CRC error in the incoming data, the number of received data bytes, excluding CRC, is written to the endpoint CNT register. Finally, CRC and BUSNACK0 in the endpoint and STATUS and CRCIF in INTFLAGSASET/CLR are set.

If data was successfully received, an ACK handshake is returned to the host if the endpoint is not isochronous, and the number of received data bytes, excluding CRC, is written to CNT. If the number of received data bytes is the maximum data payload specified by SIZE no CRC data are written in the data buffer. If the number of received data bytes is the maximum data payload specified by SIZE minus one, only the first CRC data byte is written in the data buffer If the number of received data bytes is equal or less than the data payload specified by SIZE minus two, the two CRC data bytes are written in the data buffer.

Finally, the transaction complete flag (TRNCOMPL0) and BUSNACK0 are set and TOGGLE is toggled if the endpoint is not isochronous. The transaction complete interrupt flag (TRNIF) in INTFLAGSBCLR/SET is set. The endpoint's configuration table address is written to the FIFO if the transaction complete FIFO mode is enabled.

When an OUT token is detected and the device address of the token packet does not match that of the endpoint, the packet is discarded and the USB module returns to idle and waits for the next token packet.

18.3.4 IN

If an IN token is detected the, the USB module fetches the endpoint CTRL and STATUS register data from the addressed input endpoint in the endpoint configuration table. If the endpoint is disabled, the USB module returns to idle and waits for the next token packet.

If the STALL flag in endpoint CTRL register is set, and the endpoint is not isochronous, a STALL handshake is returned to the host, the STALL flag in the endpoint STATUS register and the STALL interrupt flag (STALLIF) in INTFLAGSACLR/SET are set.

If BUSNACK0 is set, OVF in the endpoint STATUS register and OVFIF in the INTFLAGSACLR/SET register are set. If the endpoint is not isochronous, a NAK handshake is returned to the host.

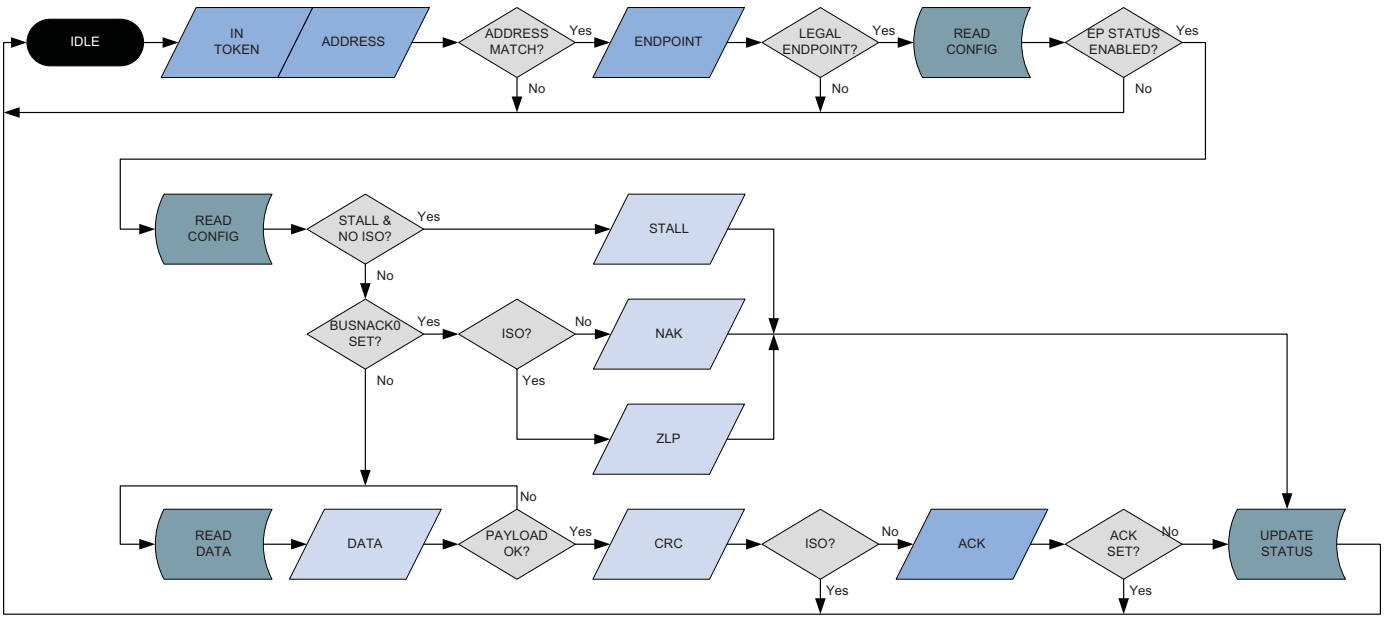
The data in the data buffer pointed to by the endpoint DATAPTR register are sent to the host in a DATA0 packet if the endpoint is isochronous; otherwise, a DATA0 or DATA1 packet according to TOGGLE is sent. When the number of data bytes specified in endpoint CNT is sent, the CRC is appended and sent to the host. If not, a ZLP handshake is returned to the host.

For isochronous endpoints, BUSNACK0 and TRNCOMPL0 in the endpoint STATUS register are set. TRNIF is set, and the endpoint's configuration table address is written to the FIFO if the transaction complete FIFO mode is enabled.

For all non-isochronous endpoints, the USB module waits for an ACK handshake from the host. If an ACK handshake is not received within 16 USB clock cycles, the USB module returns to idle and waits for the next token packet. If an ACK handshake was successfully received, BUSNACK0 and TRNCOMPL0 are set and TOGGLE is toggled. TRNIF is set and the endpoint's configuration table address is written to the FIFO if the transaction complete FIFO mode is enabled.

When an IN token is detected and the device address of the token packet does not match that of the endpoint, the packet is discarded and the USB module returns to idle and waits for the next token packet.

Figure 18-5. IN transaction.



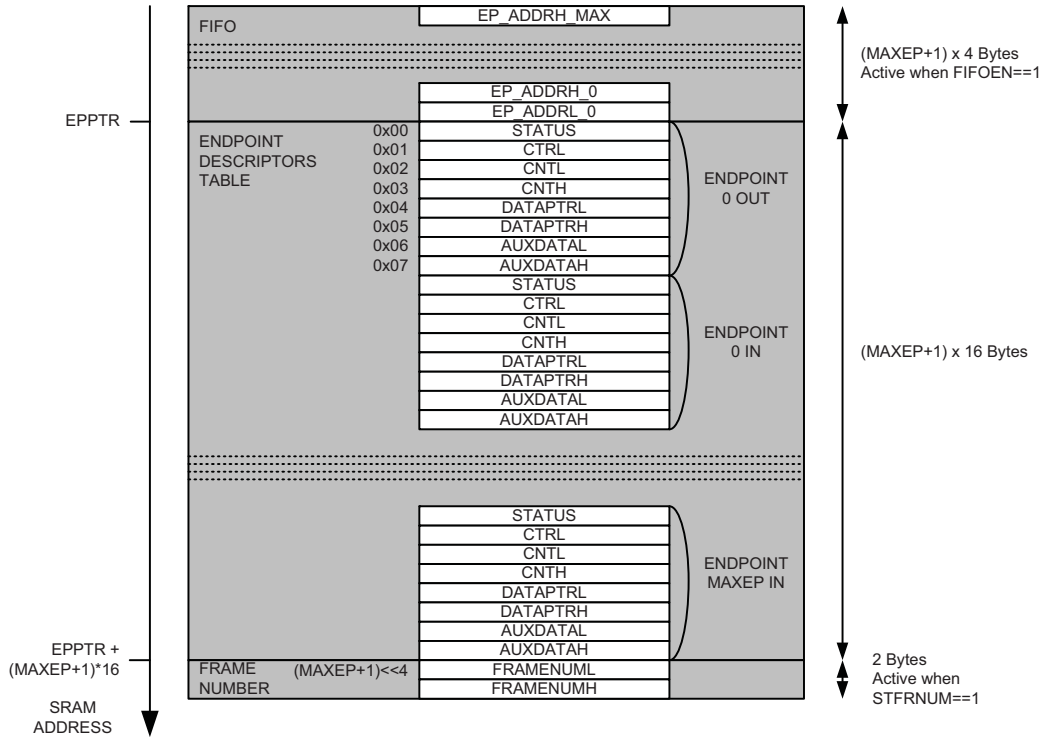
18.4 SRAM Memory Mapping

The USB module uses internal SRAM to store the:

- Endpoint configuration table
- USB frame number
- Transaction complete FIFO

The endpoint pointer register (EPPTR) is used to set the SRAM address for the endpoint configuration table. The USB frame number (FRAMENUM) and transaction complete FIFO (FIFO) locations are derived from this. The locations of these areas are selectable inside the internal SRAM. Figure on page 211 gives the relative memory location of each area.

Figure 18-6. SRAM memory mapping.



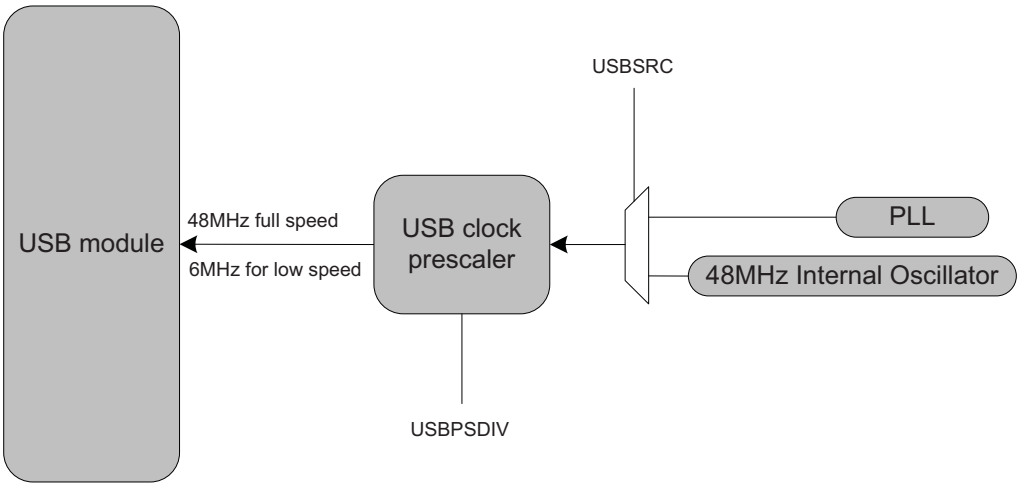
18.5 Clock Generation

The USB module requires a minimum 6MHz clock for USB low speed operation, and a minimum 48MHz clock for USB full speed operation. It can be clocked from internal or external clock sources by using the internal PLL, or directly from the 32MHz internal oscillator when it is tuned and calibrated to 48MHz. The CPU and peripherals clocks must run at a minimum of 1.5MHz for low speed operation, and a minimum of 12MHz for full speed operation.

The USB module clock selection is independent of and separate from the main system clock selection. Selection and setup are done using the main clock control settings. For details, refer to “System Clock and Clock Options” on page 75.

The Figure 18-7 on page 212 shows an overview of the USB module clock selection.

Figure 18-7. Clock generation configuration.



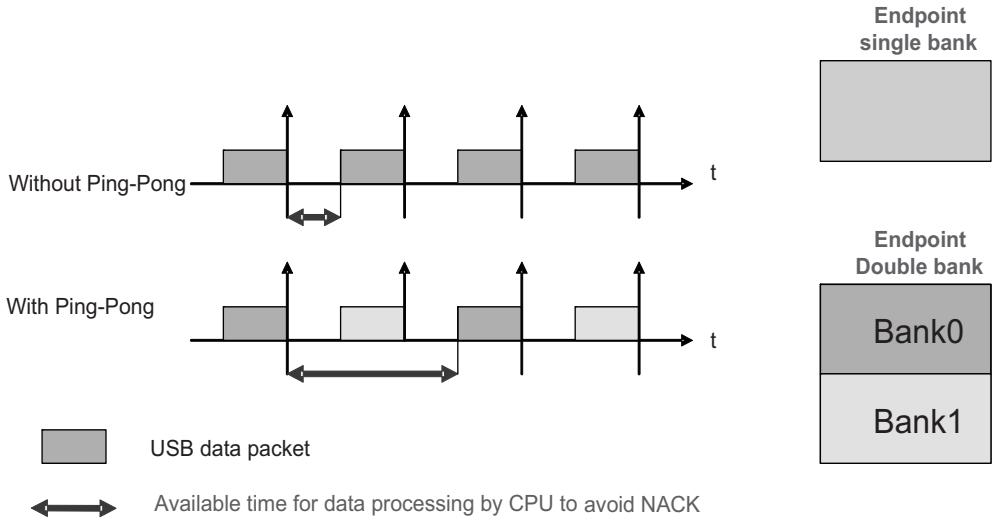
18.6 Ping-pong Operation

When an endpoint is configured for ping-pong operation, it uses the input and output data buffers to create a single, double-buffered endpoint that can be set to input or output direction. This provides double-buffered communication, as the CPU or DMA controller can access one of the buffers, while the other buffer is processing an ongoing transfer. Ping-pong operation is identical to the IN and OUT transactions described above, unless otherwise noted in this section. Ping-pong operation is not possible for control endpoints.

When ping-pong operation is enabled for an endpoint, the endpoint in the opposite direction must be disabled. The data buffer, data pointer, byte counter, and auxiliary data from the enabled endpoint are used as bank 0, and, correspondingly, bank 1 for the opposite endpoint direction.

The bank select (BANK) flag in the endpoint STATUS register indicates which data bank will be used in the next transaction. It is updated after each transaction. The TRNCOMPL0/TRNCOMPL1, underflow/overflow (UDF/OVF), and CRC flags in the STATUS register are set for either the enabled or the opposite endpoint direction according to the BANK flag. The data toggle (TOGGLE), data buffer 0/1 not acknowledge (BUSNACK0 and BUSNACK1), and BANK flags are updated for the enabled endpoint direction only.

Figure 18-8. Ping-pong operation overview.

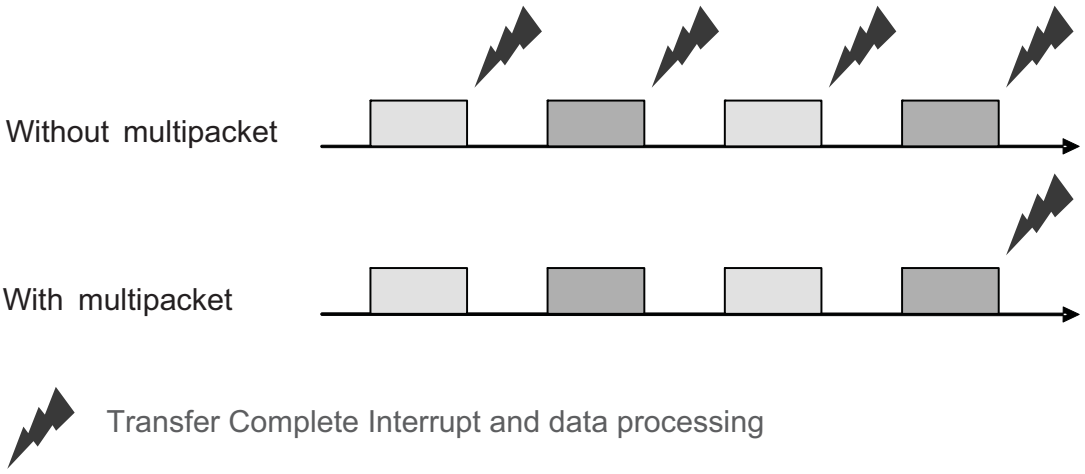


18.7 Multipacket Transfers

Multipacket transfer enables a data payload exceeding the maximum data payload size of an endpoint to be transferred as multiple packets without any software intervention. This reduces interrupts and software intervention to the higher level USB transfer, and frees up significant CPU time. Multipacket transfer is identical to the IN and OUT transactions described above, unless otherwise noted in this section.

The application software provides the size and address of the SRAM buffer to be processed by the USB module for a specific endpoint, and the USB module will then split the buffer in the required USB data transfer.

Figure 18-9. Multipacket overview.



18.7.1 For Input Endpoints

The total number of data bytes to be sent is written to CNT, as for normal operation. The auxiliary data register (AUXDATA) is used to store the number of bytes that will be sent, and must be written to zero for a new transfer.

When an IN token is received, the endpoint's CNT and AUXDATA are fetched. If CNT minus AUXDATA is less than the endpoint SIZE, endpoint CNT minus endpoint AUXDATA number bytes are transmitted; otherwise, SIZE number of bytes are transmitted. If endpoint CNT is a multiple of SIZE and auto zero length packet (AZLP) is enabled, the last packet sent will be zero length.

If a maximum payload size packet was sent (i.e., not the last transaction), AUXDATA is incremented by SIZE. TOGGLE will be toggled after the transaction has completed if the endpoint is not isochronous. If a short packet was sent (i.e., the last transaction), AUXDATA is incremented by the data payload. TOGGLE will be toggled if the endpoint is not isochronous, and BUSNACK, TRNIF, and TRNCOMPL0 will be set.

18.7.2 For Output Endpoints

The number of data bytes received is stored in the endpoint's CNT register, as for normal operation. Since the endpoint's CNT is updated after each transaction, it must be set to zero when setting up a new transfer. The total number of bytes to be received must be written to AUXDATA. This value must be a multiple of SIZE, except for ISO 1023 bytes endpoints; otherwise, excess data may be written to SRAM locations used by other parts of the application.

TOGGLE management is as for non-isochronous packets, and BUSNACK0/BUSNACK1 management is as for normal operation.

If a maximum payload size packet is received, CNT is incremented by SIZE after the transaction has completed, and TOGGLE toggles if the endpoint is not isochronous. If the updated endpoint CNT is equal to AUXDATA, then BUSNACK0/BUSNACK1, TRNIF, and TRNCOMPL0/TRNCOMPL1 will be set.

If a short or oversized packet is received, the endpoint's CNT register will be incremented by the data payload after the transaction has completed. TOGGLE will be toggled if the endpoint is not isochronous, and BUSNACK0/BUSNACK1, TRNIF, and TRNCOMPL0/TRNCOMPL1 will be set.

18.8 Auto Zero Length Packet

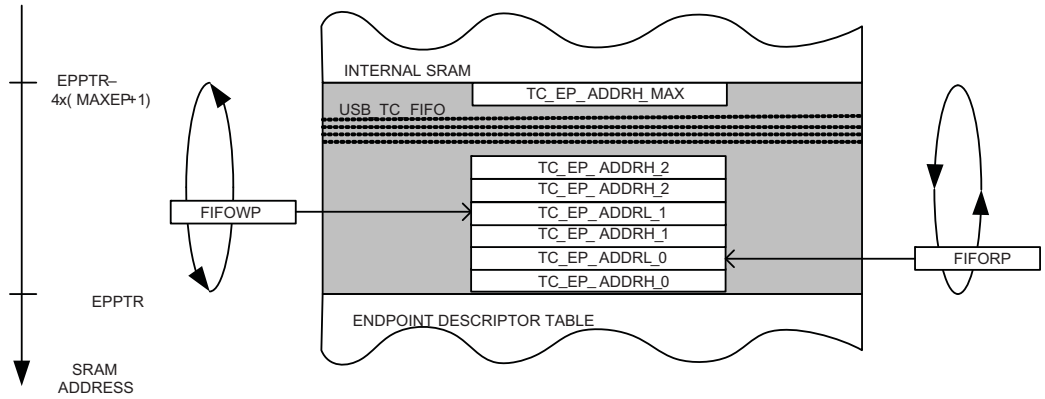
Some IN transfer requires a zero length packet to be generated in order to signal end of transfer to the host. The auto zero length packet (AZLP) function can be enabled to perform this generation automatically, thus removing the need for application software or CPU intervention to perform this task.

18.9 Transaction Complete FIFO

The transaction complete FIFO provides a convenient way to keep track of the endpoints that have completed IN or OUT transactions and need firmware intervention. It creates a first-come, first-served work queue for the application software.

The FIFO size is (MAXEP[3:0] + 1) × 4 bytes, and grows downward, starting from EPPTR - 1. This SRAM memory is allocated only when the FIFO is enabled.

Figure 18-10.Transfer complete FIFO.

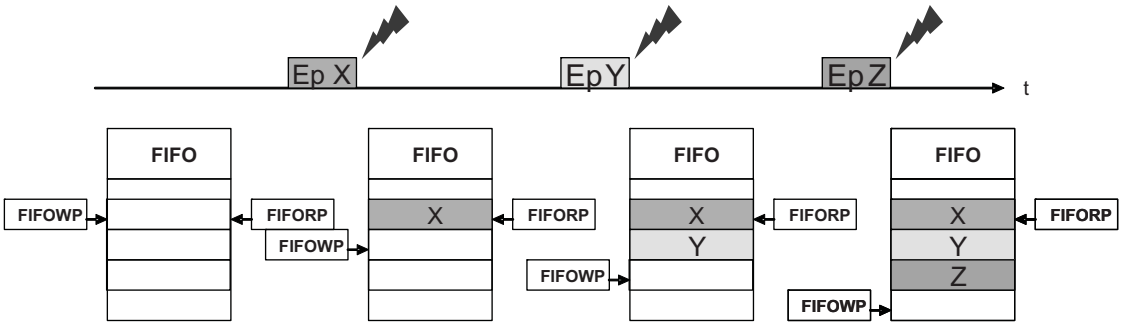


To manage the FIFO, a five-bit write pointer (FIFOWP) and five-bit read pointer (FIFORP) are used by the USB module and application software, respectively. FIFORP and FIFOWP are one's complemented, and thus hold negative values. The SRAM location of the data is the sum of EPPTR and the read or write pointer. The number of items in the FIFO is the difference between FIFOWP and FIFORP. For the programmer, the FIFORP and FIFOWP values have to be cast to a signed 8-bit integer, and then the offset into the FIFO from this signed integer must be deducted.

The transaction complete interrupt flag (TRNIF) in the INFLAGSB[CLR,SET] register is set to indicate a non-empty FIFO when FIFORP != FIFOWP, cleared when they are equal, and also set when the FIFO is full.

Each time an endpoint IN or OUT transaction completes successfully, its endpoint configuration table address is stored in the FIFO at the current write pointer position (i.e., EPPTR + 2 × FIFOWP) and FIFOWP is decremented. When the pointer reaches the FIFO size, it wraps to zero. When application software reads FIFORP, this is decremented in the same way. Reading the write pointer has no effect. The endpoint configuration table address can then be read directly from (EPPTR + 2 × FIFORP).

Figure 18-11.USB transaction complete FIFO example.

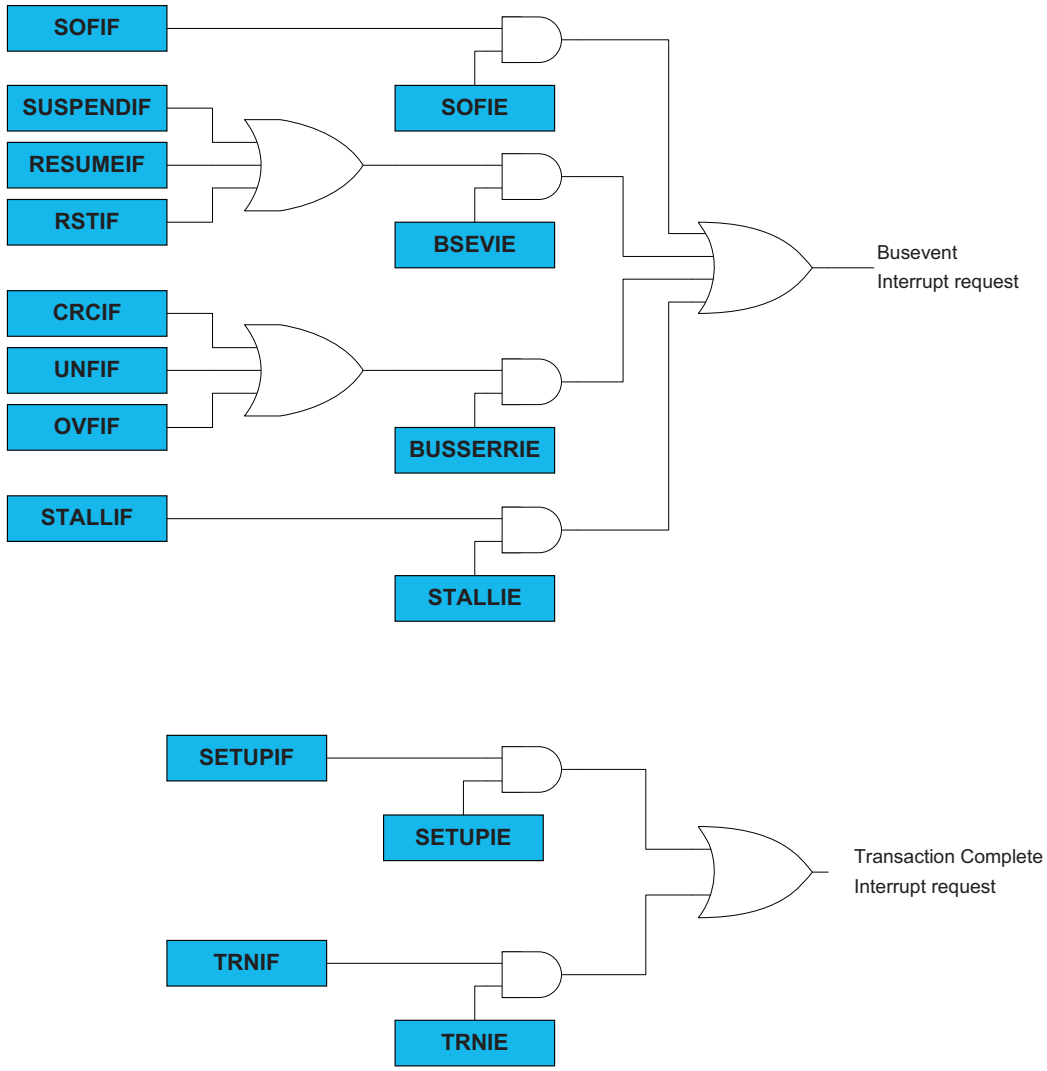


18.10 Interrupts and Events

The USB module can generate interrupts and events. The module has 10 interrupt sources. These are split between two interrupt vectors, the transaction complete (TRNCOMPL) interrupt and the bus event (BUSEVENT) interrupt. An interrupt group is enabled by setting its interrupt level (INTLVL), while different interrupt sources are enabled individually or in groups.

Figure 18-12 on page 215 summarizes the interrupts and event sources for the USB module, and shows how they are enabled.

Figure 18-12.Interrupts and events scheme summary.



18.10.1 Transaction Complete Interrupt

The transaction complete interrupt is generated per endpoint. When an interrupt occurs, the associated endpoint number is registered and optionally added to the FIFO. The following two interrupt sources use the interrupt vector:

Table 18-1. Transaction complete interrupt sources.

Interrupt source	Description
Transfer complete (TRNIF)	An IN or OUT transaction is completed
Setup complete (SETUPIF)	A SETUP transaction is completed

18.10.2 Bus Event Interrupt

The bus event (BUSEVENT) interrupt is used for all interrupts that signal various types of USB line events or error conditions. These interrupts are related to the USB lines, and are generated for the USB module and per endpoint. The following eight interrupts use the interrupt vector:

Table 18-2. Bus event interrupt source.

Interrupt source	Description
Start of frame (SOFIF)	A SOF token has been received
Suspend (SUSPENDIF)	The bus has been idle for 3ms
Resume (RESUMEIF)	A non-idle state is detected when the bus is suspended. The interrupt is asynchronous and can wake the device from all sleep modes
Reset (RSTIF)	A reset condition has been detected on the bus
Isochronous CRC error (CRCIF)	A CRC or bit-stuff error has been detected in an incoming packet to an isochronous endpoint
Underflow (UNFIF)	An endpoint is unable to return data to the host
Overflow (OVFIF)	An endpoint is unable to accept data from the host
STALL (STALLIF)	A STALL handshake has been returned to the host

18.10.3 Events

The USB module can generate several events, and these are available to the event system, allowing latency-free signaling to other peripherals or performance analysis of USB operation.

Table 18-3. Event sources.

Event source	Description
SETUP	SETUPIF
Start of Frame	SOFIF
CRC error	CRCIF
Underflow/overflow	UNFIF and OVFIF

18.11 VBUS Detection

Atmel AVR XMEGA devices can use any general purpose I/O pin to implement a VBUS detection function, and do not use a dedicated VBUS detect pin.

18.12 On-chip Debug

When a break point is reached during on-chip debug (OCD) sessions, the CPU clock can be below 12MHz. If this happens, the USB module will behave as follows:

USB OCD break mode disabled: The USB module immediately acknowledges any OCD break request. The USB module will not be able to follow up on transactions received from the USB host, and its behaviour from the host point of view is not predictable.

USB OCD break mode enabled: The USB module will immediately acknowledge any OCD break request only if there are no ongoing USB transactions. If there is an ongoing USB transaction, the USB module will acknowledge any OCD break request only when the ongoing USB transaction has been completed. The USB module will NACK any further transactions received from the USB host, whether they are SETUP, IN (ISO, BULK), or OUT (ISO, BULK).

18.13 Register Description – USB

18.13.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	SPEED	FIFOEN	STFRNUM	MAXEP[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 – ENABLE: USB Enable**

Setting this bit enables the USB interface. Clearing this bit disables the USB interface and immediately aborts any ongoing transactions.

- **Bit 6 – SPEED: Speed Select**

This bit selects between low and full speed operation. By default, this bit is zero, and low speed operation is selected. Setting this bit enables full speed operation.

- **Bit 5 – FIFOEN: USB FIFO Enable**

Setting this bit enables the USB transaction complete FIFO, and the FIFO stores the endpoint configuration table address of each endpoint that generates a transaction complete interrupt. Clearing this bit disables the FIFO and frees the allocated SRAM memory.

- **Bit 4 – STFRNUM: Store Frame Number Enable**

Setting this bit enables storing of the last SOF token frame number in the frame number (FRAMENUM) register. Clearing this bit disables the function.

- **Bit 3:0 – MAXEP[3:0]: Maximum Endpoint Address**

These bits select the number of endpoint addresses used by the USB module. Incoming packets with a higher endpoint number than this address will be discarded. Packets with endpoint addresses lower than or equal to this address will cause the USB module to look up the addressed endpoint in the endpoint configuration table.

18.13.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	PULLRST	–	RWAKEUP	GNACK	ATTACH
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 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 – PULLRST: Pull during Reset**

Setting this bit enables the pull-up on the USB lines to also be held when the device enters reset. The bit will be cleared on a power-on reset.

- **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 – RWAKEUP: Remote Wake-up**

Setting this bit sends an upstream resume on the USB lines if the bus is in the suspend state for at least 5 ms.

- **Bit 1 – GNACK: Global NACK**

When this bit is set, the USB module will NACK all incoming transactions. Expect for a SETUP packet, this prevents the USB module from performing any on-chip SRAM access, giving all SRAM bandwidth to the CPU and/or DMA controller.

- **Bit 0 – ATTACH: Attach**

Setting this bit enables the internal D+ or D- pull-up (depending on the USB speed selection), and attaches the device to the USB lines. Clearing this bit disconnects the device from the USB lines.

18.13.3 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	**URESUME**	**RESUME**	**SUSPEND**	**BUSRST**
Read/Write	R	R	R	R	R	R	R	R
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 – URESUME: Upstream Resume**

This flag is set when an upstream resume is sent.

- **Bit 2 – RESUME: Resume**

This flag is set when a downstream resume is received.

- **Bit 1 – SUSPEND: Bus Suspended**

This flag is set when the USB lines are in the suspended state (the bus has been idle for at least 3ms).

- **Bit 0 – BUSRST: Bus Reset**

This flag is set when a reset condition has been detected (the bus has been driven to SE0 for at least 2.5µs).

18.13.4 ADDR – Address register

Bit	7	6	5	4	3	2	1	0
+0x03	–	**ADDR[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 – ADDR[6:0]: Device Address**

These bits contain the USB address the device will respond to.

18.13.5 FIFOWP – FIFO Write Pointer register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	FIFOWP[4:0]				
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 4:0 – FIFOWP[4:0]: FIFO Write Pointer**

These bits contain the transaction complete FIFO write pointer. This register must be read only by the CPU or DMA controller. Writing this register will flush the FIFO write and read pointers.

18.13.6 FIFORP – FIFO Read Pointer register

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	–	FIFORP[4:0]				
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 4:0 – FIFORP[4:0]: FIFO Read Pointer**

These bits contain the transaction complete FIFO read pointer. This register must only be read by the CPU or DMA controller. Writing this register will flush the FIFO write and read pointer.

18.13.7 EPPTRL – Endpoint Configuration Table Pointer Low

The EPPTRL and EPPTRH registers represent the 16-bit value, EPPTR, that contains the address to the endpoint configuration table. The pointer to the endpoint configuration table must be aligned to a 16-bit word; i.e., EPPTR[0] must be zero. Only the number of bits required to address the available internal SRAM memory is implemented for each device. Unused bits will always be read as zero.

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

- **Bit 7:0 – EPPTR[7:0]: Endpoint Configuration Table Pointer low byte**

This register contains the eight lsbs of the endpoint configuration table pointer (EPPTR).

18.13.8 EPPTRH – Endpoint Configuration Table Pointer High

Bit	7	6	5	4	3	2	1	0
+0x07	EPPTR[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 – EPPTR[15:8]: Endpoint Configuration Table Pointer high byte**

This register contains the eight msbs of the endpoint configuration table pointer (EPPTR).

18.13.9 INTCTRLA – Interrupt Control register A

Bit	7	6	5	4	3	2	1	0
+0x06	SOFIE	BUSEVIE	BUSERRIE	STALLIE	–	–	INTLVL[1:0]	
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

- **Bit 7 – SOFIE: Start Of Frame Interrupt Enable**

Setting this bit enables the start of frame (SOF) interrupt for the conditions that set the start of frame interrupt flag (SOFIF) in the INTFLAGSACLR/ INTFLAGSASET register. The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 6 – BUSEVIE: Bus Event Interrupt Enable**

Setting this bit will enable the interrupt for the following three bus events:

1. *Suspend*: An interrupt will be generated for the conditions that set the suspend interrupt flag (SUSPENDIF) in the INTFLAGSACLR/SET register.
2. *Resume*: An interrupt will be generated for the conditions that set the resume interrupt flag (RESUMEIF) in the INTFLAGSACLR/SET register.
3. *Reset*: An interrupt will be generated for the conditions that set the reset interrupt flag (RESETIF) in the INTFLAG-SACLR/SET register.

The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 5 – BUSERRIE: Bus Error Interrupt Enable**

Setting this bit will enable the interrupt for the following three bus error events:

1. *Isochronous CRC Error:* An interrupt will be generated for the conditions that set the CRC interrupt flag (CRCIF) in the INTFLAGSACLR/SET register during isochronous transfers.
2. *Underflow*: An interrupt will be generated for the conditions that set the underflow interrupt flag (UNFIF) in the INTFLAGSACLR/SET register.
3. *Overflow*: An interrupt will be generated for the conditions that set the overflow interrupt flag (OVFIF) in the INTFLAGSACLR/SET register.

The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 4 – STALLIE: STALL Interrupt Enable**

Setting this bit enables the STALL interrupt for the conditions that set the stall interrupt flag (STALLIF) in the INTFLAGSACLR/SET register. The INTLVL bits must be nonzero for the interrupts to be generated.

- **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 USB interrupts and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. In addition, each USB interrupt source must be separately enabled.

18.13.10INTCTRLB – Interrupt Control register B

Bit	7	6	5	4	3	2	1	0
+0x07	–	–	–	–	–	–	TRNIE	SETUPIE
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 – TRNIE: Transaction Complete Interrupt Enable**

Setting this bit enables the transaction complete interrupt for IN and OUT transactions. The INTLVL bits must be nonzero for interrupts to be generated.

- **Bit 0 – SETUPIE: SETUP Transaction Complete Interrupt Enable**

Setting this bit enables the SETUP Transaction Complete Interrupt for SETUP transactions. The INTLVL bits must be non-zero for the interrupts to be generated.

18.13.11INTFLAGSACLR/ INTFLAGSASET – Clear/ Set Interrupt Flag register A

This register is mapped into two I/O memory locations, one for clearing (INTFLAGSACLR) and one for setting (INTFLAGSASET) the flags. The individual flags can be set by writing a one to their bit locations in INFLAGSASET, and cleared by writing a one to their bit locations in INT-FLAGSACLR. Both memory locations will provide the same result when read, and writing zero to any bit location has no effect.

Bit	7	6	5	4	3	2	1	0
+0x0A/ +0x0B	SOFIF	SUSPENDIF	RESUMEIF	RESETIF	CRCIF	UNFIF	OVFIF	STALLIF
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 – SOFIF: Start Of Frame Interrupt Flag**

This flag is set when a start of frame packet has been received.

- **Bit 6 – SUSPENDIF: Suspend Interrupt Flag**

This flag is set when the bus has been idle for 3ms.

- **Bit 5 – RESUMEIF: Resume Interrupt Flag**

This flag is set when a non-idle state has been detected on the bus while the USB module is in the suspend state. This interrupt is asynchronous, and is able to wake the CPU from sleep modes where the system clock is stopped, such as power-down and power-save sleep modes.

- **Bit 4 – RSTIF: Reset Interrupt Flag**

This flag is set when a reset condition has been detected on the bus.

- **Bit 3 – CRCIF: Isochronous CRC Error Interrupt Flag**

This flag is set when a CRC error has been detected in an incoming data packet to an isochronous endpoint.

- **Bit 2 – UNFIF: Underflow Interrupt Flag**

This flag is set when the addressed endpoint in an IN transaction does not have data to send to the host.

- **Bit 1 – OVFIF: Overflow Interrupt Flag**

This flag is set when the addressed endpoint in an OUT transaction is not ready to accept data from the host.

- **Bit 0 – STALLIF: STALL Interrupt Flag**

This flag is set when the USB module has responded with a STALL handshake to either an IN or an OUT transaction.

18.13.12INTFLAGSBCLR/INTFLAGSBSET – Clear/Set Interrupt Flag register B

This register is mapped into two I/O memory locations, one for clearing (INTFLAGSBCLR) and one for setting (INTFLAGSBSET) the flags. The individual flags can be set by writing a one to their bit locations in INFLAGSBSET, and cleared by writing a one to their bit locations in INTFLAGSBCLR. Both memory locations will provide the same result when read, and writing zero to any bit location has no effect.

Bit	7	6	5	4	3	2	1	0
+0x0C/ +0x0D	–	–	–	–	–	–-	**TRNIF**	**SETUPIF**
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 – TRNIF: Transaction Complete Interrupt Flag**

This flag is when there is a pending packet interrupt in the FIFO.

- **Bit 0 – SETUPIF: SETUP Transaction Complete Interrupt Flag**

This flag is set when a SETUP transaction has completed successfully.

18.13.13CALL – Calibration register Low

CALL and CALH hold the 16-bit value, CAL. The USB PADs (D- and D+) are calibrated during production to enable operation without requiring external components on the USB lines. The calibration value is stored in the signature row of the device, and must be read from there and written to the CAL registers from software.

Bit	7	6	5	4	3	2	1	0
++0x3A	**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

- **Bit 7:0 – CAL[7:0]: PAD Calibration low byte**

This byte holds the eight lsbs of CAL.

18.13.14CALH – Calibration register High

Bit	7	6	5	4	3	2	1	0
+0x3B	CAL[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 – CAL[15:8]: PAD Calibration high byte**

This byte holds the eight msbs of CAL.

18.14 Register Description – USB Endpoint

Each of the 16 endpoint addresses have one input and one output endpoint. Each endpoint has eight bytes of configuration/status data located in internal SRAM.

The address to the first configuration byte is (EPPTR[15:0] + 16 × endpoint address) for output endpoints and (EPPTR[15:0] + 16 × endpoint address + 8) for input endpoints.

Some bit locations have different functions, depending on endpoint configuration type or direction, and this is reflected by using two different names for the bit locations.

18.14.1 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x00	STALL / CRC(1)	UNF/ OVF	TRNCOMPL0	SETUP / TRNCOMPL1	BANK	BUSNACK1	BUSNACK0	TOGGLE
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. For isochronous endpoints.

- **Bit 7 – STALL: STALL Flag**

This flag is set when an IN or OUT transaction has been responded to with a STALL handshake. This flag is cleared by writing a one to its bit location.

- **Bit 7 – CRC: CRC Error Flag**

This flag is set for isochronous output endpoints when a CRC error has been detected in an incoming data packet. This flag is cleared by writing a one to its bit location.

- **Bit 6 – UNF/OVF: Underflow/Overflow Flag**

UNF: For input endpoints, the UNF flag is set when an input endpoint is not ready to send data to the host in response of an IN token.

OVF: For output endpoints, the OVF flag is set when an output endpoint is not ready to accept data from the host following an OUT token.

- **Bit 5 – TRNCOMPL0: Transaction Complete Flag**

This flag is set when an IN or OUT transaction has completed successfully. This flag is cleared by writing a one to its bit location.

- **Bit 4 – SETUP: SETUP Transaction Complete Flag**

This flag is set when a SETUP, IN, or OUT transaction has completed successfully. This flag is cleared by writing a one to its bit location.

- **Bit 4 – TRNCOMPL1: Transaction Complete Flag**

This flag is set when a SETUP, IN, or OUT transaction has completed successfully. This flag is cleared by writing a one to its bit location.

- **Bit 3 – BANK: Bank Select Flag**

When ping-pong mode is enabled, this bit indicates which bank will be used for the next transaction. BANK is toggled each time a transaction has completed successfully. This bit is not sed when ping-pong is disabled. This flag is cleared by writing a one to its bit location.

- **Bit 2 – BUSNACK1: Data Buffer 1 Not Acknowledge Flag**

When this flag is set, the USB module will discard incoming data to data buffer 1 in an OUT transaction, and will not return any data from data buffer 1 in an IN transaction. For control, bulk, and interrupt endpoints, a NAK handshake is returned. This flag is cleared by writing a one to its bit location.

- **Bit 1 – BUSNACK0: Data Buffer 0 Not Acknowledge Flag**

When this flag is set, the USB module will discard incoming data to data buffer 0 in an OUT transaction, and will not return any data from data buffer 0 in an IN transaction. For control, bulk, and interrupt endpoints, a NAK handshake is returned. This flag is cleared by writing a one to its bit location.

- **Bit 0 – TOGGLE: Data Toggle Flag**

This indicates if a DATA0 or DATA1 PID is expected in the next data packet for an output endpoint, and if a DATA0 or DATA1 PID will be sent in the next transaction for an input endpoint. This bit has no effect for isochronous endpoints, where both DATA0 and DATA1 PIDs are accepted for output endpoint, and only DATA0 PIDs are sent for input endpoints.

18.14.2 CTRL – Control

Bit	7	6	5	4	3	2	1	0
+0x01	TYPE[1:0]		MULTIPKT	PINGPONG	INTDSBL	STALL	SIZE[1:0]	
						SIZE[2:0](1)		
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. For isochronous endpoints.

- **Bit 7:6 – TYPE[1:0]: Endpoint Type**

These bits are used to enable and select the endpoint type. If the endpoint is disabled, the remaining seven endpoint configuration bytes are never read or written by the USB module, and their SRAM locations are free to use for other application data.

Table 18-4. Endpoint type.

TYPE[1:0]	Group Configuration	Description
00	DISABLE	Endpoint enabled
01	CONTROL	Control
10	BULK	Bulk/interrupt
11	ISOCHRONOUS	Isochronous

- **Bit 5 – MULTIPKT: Multipacket Transfer Enable**

Setting this bit enables multipacket transfers. Multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without interrupts or software intervention. See “Multipacket Transfers” on page 213 for details on multipacket transfers.

- **Bit 4 – PINGPONG: Ping-pong Enable**

Setting this bit enables ping-pong operation. Ping-pong operation enables both endpoints (IN and OUT) with same address to be used in the same direction to allow double buffering and maximize throughput. The endpoint in the opposite direction must be disabled when ping-pong operation is enabled. Ping-pong operation is not possible for control endpoints. See “Ping-pong Operation” on page 212 for details.

- **Bit 3 – INTDSBL: Interrupt Disable**

Setting this bit disables all enabled interrupts from the endpoint. Hence, only the interrupt flags in the STATUS register are updated when interrupt conditions occur. The FIFO does not store this endpoint configuration table address upon transaction complete for the endpoint when interrupts are disabled for an endpoint. Clearing this bit enables all previously enables interrupts again.

- **Bit 2 – STALL: Endpoint STALL**

This bit controls the STALL behavior if the endpoint.

- **Bit 1:0 – BUFSIZE[1:0]: Data Size**

These bits configure the maximum data payload size for the endpoint. Incoming data bytes exceeding the maximum data payload size are discarded.

- **Bit 2:0 – BUFSIZE[2:0]: Data Size**

These bits configure the maximum data payload size for the endpoint when configured for isochronous operation.

Table 18-5. BUFSIZE configuration

BUFSIZE[2:0]	Group Configuration	Description
000	8	8-byte buffer size
001	16	16-byte buffer size
010	32	32-byte buffer size
011	64	64-byte buffer size
100(1)	128	128-byte buffer size
101(1)	256	256-byte buffer size
110(1)	512	512-byte buffer size
111(1)	1023	1023-byte buffer size

Note: 1. Setting only available for isochronous endpoints.

18.14.3 CNTL – Counter Low register

The CNTL and CNTH registers represent the 10-bit value, CNT, that contains the number of bytes received in the last OUT or SETUP transaction for an OUT endpoint, or the number of bytes to be sent in the next IN transaction for an IN endpoint.

Bit	7	6	5	4	3	2	1	0
+0x02	CNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 7:0 – CNT[7:0]: Endpoint Byte Counter**

This byte contains the eight lsbs of the USB endpoint counter (CNT).

18.14.4 CNTH – Counter High register

Bit	7	6	5	4	3	2	1	0
+0x03	AZLP	–	–	–	–	–	CNT[9:8]	
Read/Write	R/W	R	R	R	R	R	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 6 – AZLP: Automatic Zero Length Packet**

When this bit is set, the USB module will manage the ZLP handshake by hardware. This applies to IN endpoints only. When this bit is zero, the ZLP handshake must be managed by firmware.

- **Bit 6:2 – 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 – CNT[9:8]: Endpoint Byte Counter**

These bits contain the two msbs of the USB endpoint counter (CNT).

18.14.5 DATAPTRL – Data Pointer Low register

The DATAPTRL and DATAPTRH registers represent the 16-bit value, DATAPTR, that contains the SRAM address to the endpoint data buffer.

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

- **Bit 7:0 – DATAPTR[7:0]: Endpoint Data Pointer Low**

This byte contains the eight lsbs of the endpoint data pointer (DATAPTR).

18.14.6 DATAPTRH – Data Pointer High register

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

- **Bit 15:0 - DPTR[15:8]: Endpoint Data Pointer High**

This byte contains the eight msbs of the endpoint data pointer (DATAPTR).

18.14.7 AUXDATAL – Auxiliary Data Low register

The AUXDATAL and AUXDATAH registers represent the 16-bit value, AUXDATA, that is used for multipacket transfers.

For IN endpoints, AUXDATA holds the total number of bytes sent. AUXDATA should be written to zero when setting up a new transfer. For OUT endpoints, AUXDATA holds the total data size for the complete transfer. This value must be a multiple of the maximum packet size, except for ISO 1023-byte endpoints.

See “Multipacket Transfers” on page 213 for more details on setting up and using multipacket transfers.

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

- **Bit 7:0 – AUXDATA[7:0]: Auxiliary Data Low**

This byte contains the eight lsbs of the auxiliary data (AUXDATA). When multipacket transfer is not used, this SRAM location is free to use for other application data.

18.14.8 AUXDATAH – Auxiliary Data High register

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

- **Bit 7:0 – AUXDATA[15:8]: Auxiliary Data High**

This byte contains the eight msbs of the auxiliary data (AUXDATA). When multipacket transfer is not used, this SRAM location is free to use for other application data.

18.15 Register Description - Frame

18.15.1 FRAMENUML – Frame Number Low register

The FRAMENUML and FRAMENUMH registers represent the 11-bit value, FRAMENUM, that holds the frame number from the most recently received start of frame packet.

Bit	7	6	5	4	3	2	1	0
+0x00	FRAMENUM[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 – FRAMENUM[7:0]: Frame Number**

This byte contains the eight lsbs of the frame number (FRAMENUM).

18.15.2 FRAMENUMH – Frame Number High register

Bit	7	6	5	4	3	2	1	0
+0x01	FRAMEERR	–	–	–	–	FRAMENUM[10:8]		
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – FRAMEERR: Frame Error**

This flag is set if a CRC or bit-stuffing error was detected in the most recently received start of frame packet.

- **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 2:0 – FRAMENUM[10:8]: Frame Number**

This byte contains the three msbs of the frame number (FRAMENUM).

18.16 Register Summary – USB Module

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	ENABLE	SPEED	FIFOEN	STFRNUM	MAXEP[3:0]				218
+0x01	CTRLB	–	–	–	PULLRST	–	RWAKEUP	GNACK	ATTACH	218
+0x02	STATUS	–	–	–	–	UPRESUM	RESUME	SUSPEND	BUSRST	219
+0x03	ADDR	–	ADDR[6:0]							219
+0x04	FIFOWP	–	–	–	FIFOWP[4:0]					220
+0x05	FIFORP	–	–	–	FIFORP[4:0]					220
+0x06	EPPTRL	EPPTR[7:0]								220
+0x07	EPPTRH	EPPTR[15:8]								221
+0x08	INTCTRLA	SOFIE	BUSEVIE	BUSERRIE	STALLIE	–	–	INTLVL[1:0]		221
+0x09	INTCTRLB	–	–	–	–	–	–	TRNIE	SETUPIE	222
+0x0A	INFLAGSACL	SOFIF	SUSPENDI	RESUMEIF	RSTIF	CRCIF	UNFIF	OVFIF	STALLIF	222
+0x0B	INFLAGSASE	SOFIF	SUSPENDI	RESUMEIF	RSTIF	CRCIF	UNFIF	OVFIF	STALLIF	222
+0x0C	INFLAGSBCL	–	–	–	–	–	–	TRNIF	SETUPIF	223
+0x0D	INFLAGSBSE	–	–	–	–	–	–	TRNIF	SETUPIF	223
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10-0X39	Reserved	–	–	–	–	–	–	–	–	
+0x3A	CALL	CAL[7:0]								223
+0x3B	CALH	CAL[15:8]								224

18.17 Register Summary – USB Endpoint

The address to the first configuration byte is (EPPTR[15:0] + 16 × endpoint address) for OUT endpoints and (EPPTR[15:0] + 16 × endpoint address + 8) for IN endpoints.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	STALL	OVF/UNF	TRNCOMPL0	SETUP	BANK	BUSNACK1	BUSNACK0	TOGGLE	225
		CRC			TRNCOMP					Isochronous
+0x01	CTRL	TYPE[1:0]		MULTIPKT	PINGPONG	INTDSBL	STALL	BUFSIZE[1:0]		226
							BUFSIZE[2:0]			Isochronous
+0x02	CNTL	CNT[7:0]								227
+0x03	CNTH	AZLP	–	–	–	–	–	CNT[9:8]		228
+0x04	DATAPTR	DATAPTR[7:0]								228
+0x05	DATAPTR	DATAPTR[15:8]								228
+0x06	AUXDATA	AUXDATA[7:0]								229
+0x07	AUXDATA	AUXDATA[15:8]								229

18.18 Register Summary – Frame

The address to the frame configuration byte is (MAXEP + 1) << 4. For instance with MAXEP = 3, the first address would be located at offset address 0x40.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	FRAMENUM	FRAMENUM[7:0]								230
+0x01	FRAMENUM	FRAMEER	–	–	–	–	FRAMENUM[10:8]			230

18.19 USB Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	BUSEVENT_vect	SOF, suspend, resume, bus reset, CRC, underflow, overflow, and stall error interrupts
0x02	TRNCOMPL_vect	Transaction complete interrupt

19. TWI – Two-Wire Interface

19.1 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
- 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
- 100kHz and 400kHz 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)

19.2 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. Both 100kHz and 400kHz bus frequency is 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.

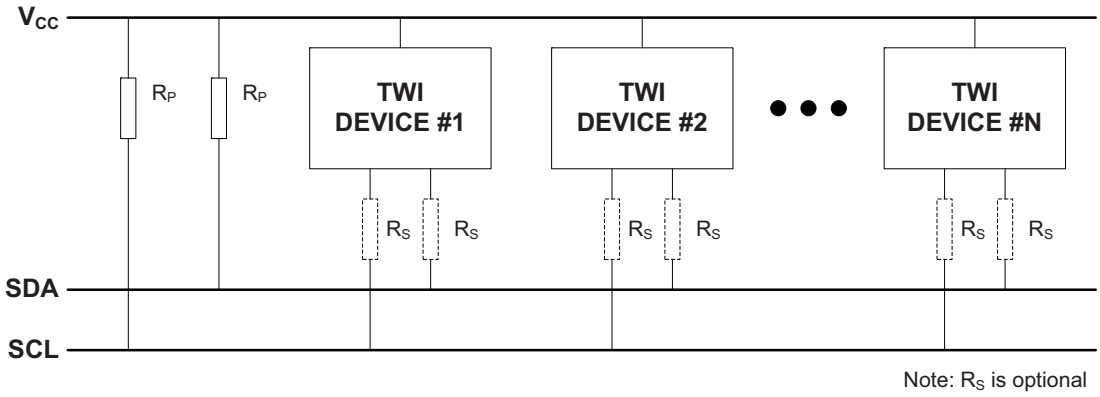
19.3 General TWI Bus Concepts

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 19-1 on page 233 illustrates the TWI bus topology.

Figure 19-1. TWI bus topology.



A 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.

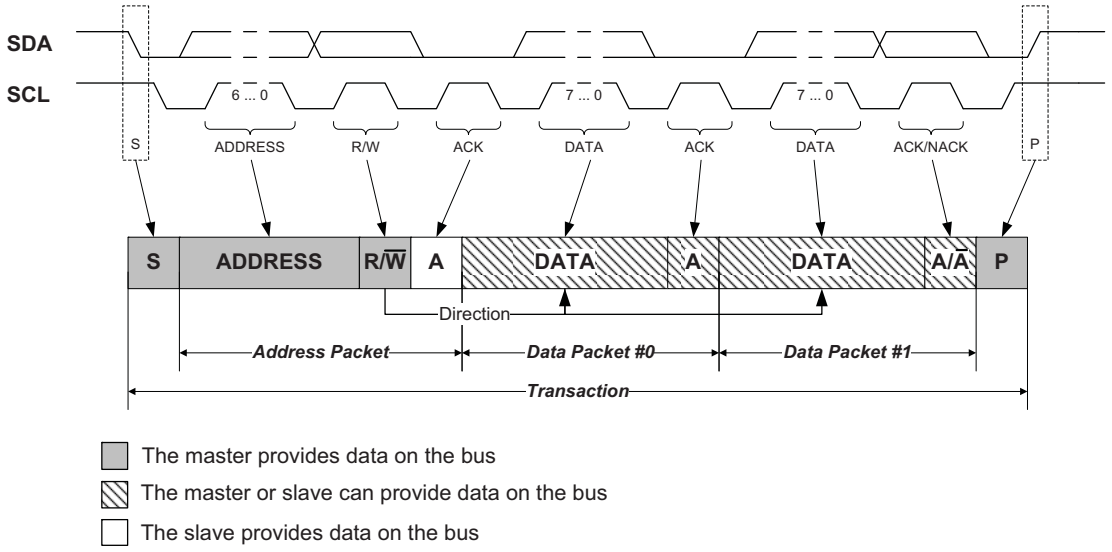
Several masters can be connected to the same bus, called a multi-master environment. An arbitration mechanism is provided for resolving bus ownership among masters, since only one master device may own the bus at any given time.

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 19-2 on page 234 shows a TWI transaction.

Figure 19-2. 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.

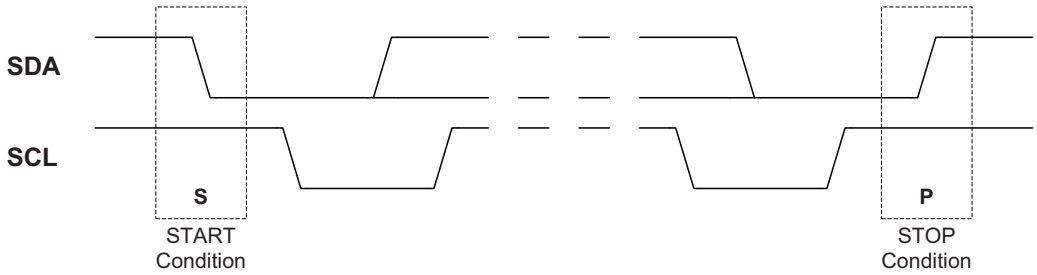
19.3.1 Electrical Characteristics

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 239 for more details.

19.3.2 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 19-3. START and STOP conditions.

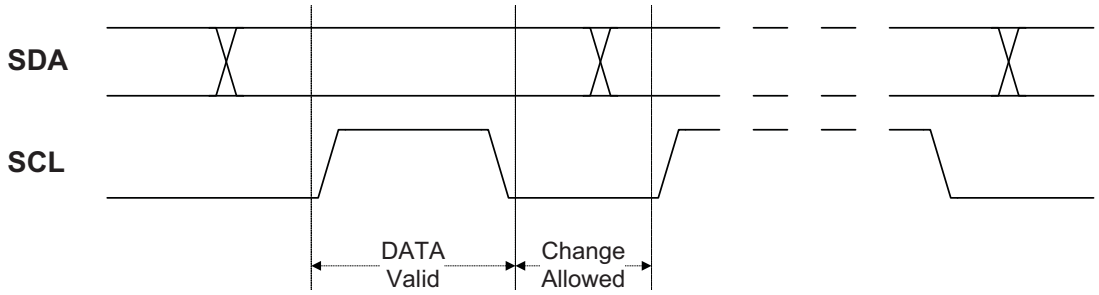


Multiple START conditions can be issued during a single transaction. A START condition that is not directly following a STOP condition is called a repeated START condition (Sr).

19.3.3 Bit Transfer

As illustrated by Figure 19-4, 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 19-4. 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.

19.3.4 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.

19.3.5 Data Packet

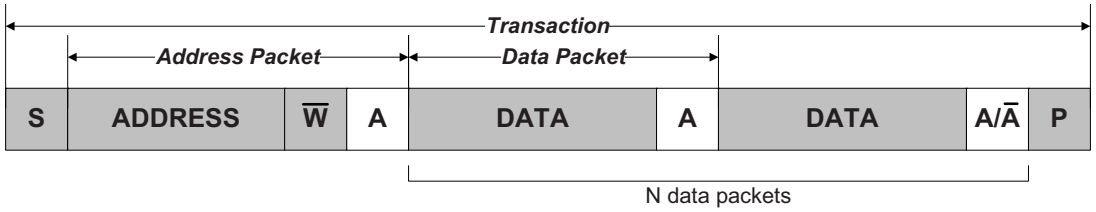
An address packet is followed by one or more data packets. All data packets are nine bits long, consisting of one data byte and an acknowledge bit. The direction bit in the previous address packet determines the direction in which the data are transferred.

19.3.6 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 19-5 on page 235 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 19-5. Master write transaction.

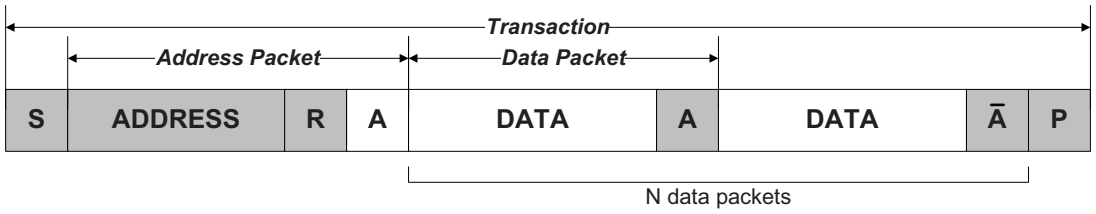


Assuming the slave acknowledges the address, the master can start transmitting data (DATA) and the slave will ACK or NACK (A/$\overline{A}$) each byte. If no data packets are to be transmitted, the master terminates the transaction by issuing a STOP condition (P) directly after the address packet. There 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 19-6 on page 236 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.

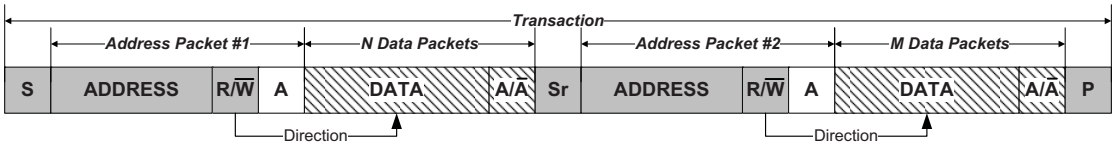
Figure 19-6. Master read transaction.



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 19-7 illustrates a combined transaction. A combined transaction consists of several read and write transactions separated by repeated START conditions (Sr).

Figure 19-7. Combined Transaction.

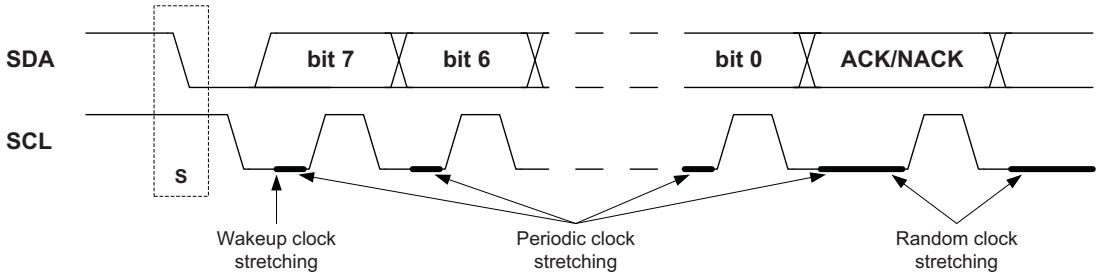


19.3.7 Clock and Clock Stretching

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 19-8.

Figure 19-8. 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.

19.3.8 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 19-9. TWI arbitration.

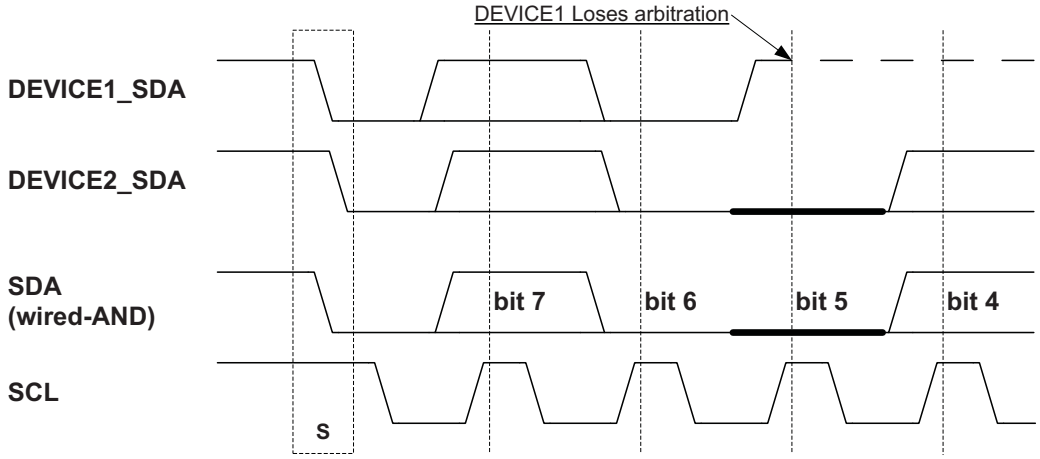


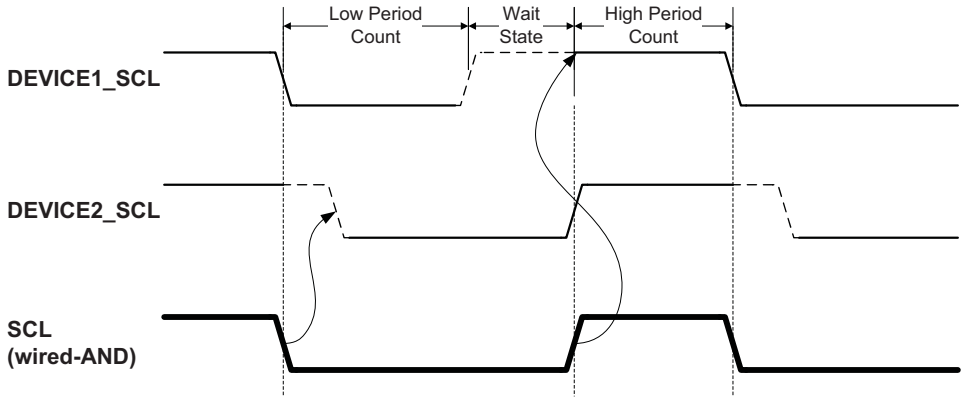
Figure 19-9 shows an example where two TWI masters are contending for bus ownership. Both devices are able to issue a START condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

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.

19.3.9 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 19-10 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 19-10.Clock synchronization.



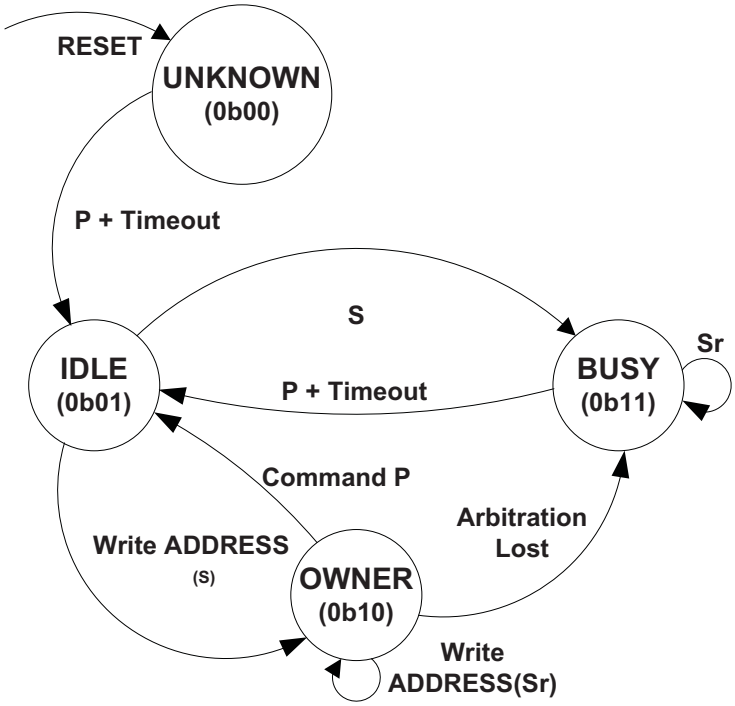
A high-to-low transition on the SCL line will force the line low for all masters on the bus, and they will start timing their low clock period. The timing length of the low clock period can vary among the masters. When a master (DEVICE1 in this case) has completed its low period, it releases the SCL 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.

19.4 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 19-11. The values of the bus state bits according to state are shown in binary in the figure.

Figure 19-11.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.

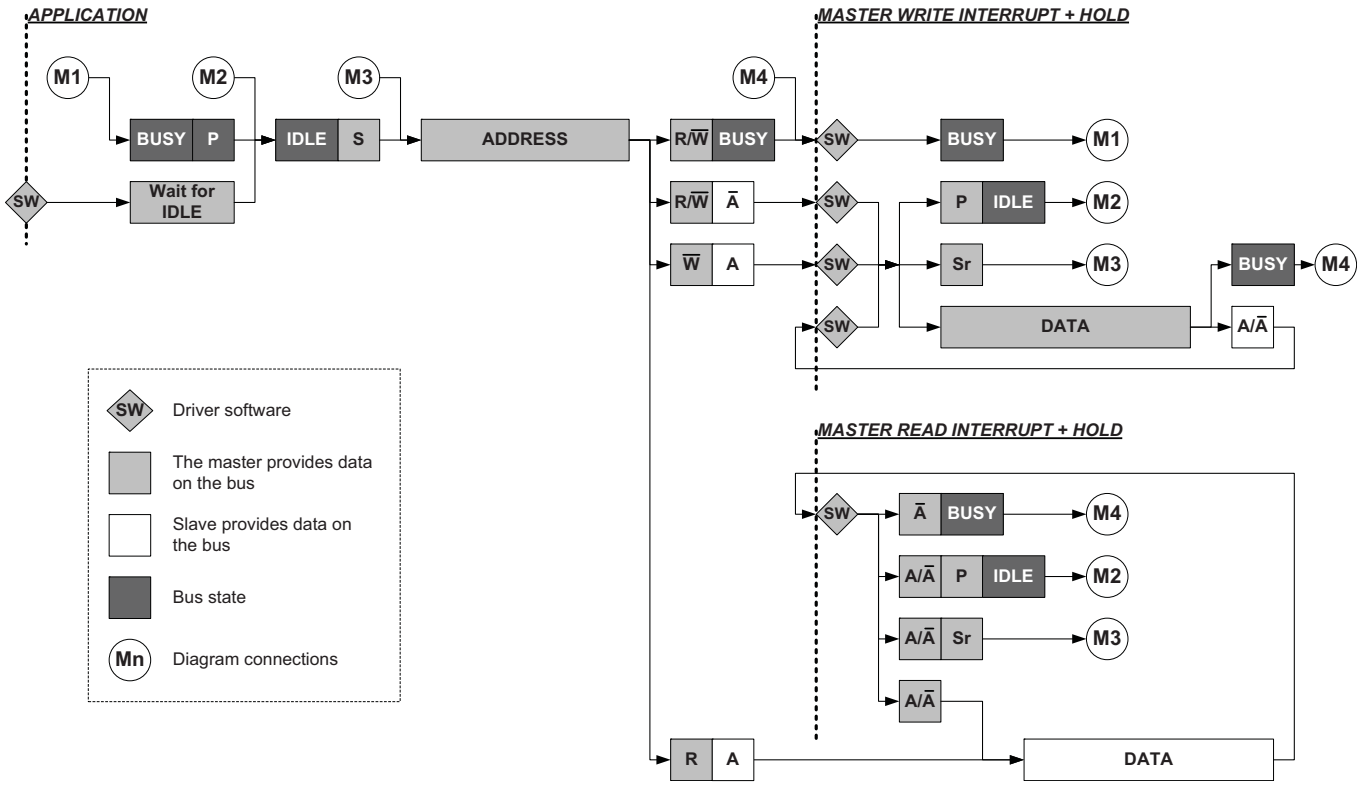
If a START condition is generated internally while in idle state, the owner state is entered. If the complete transaction was performed without interference, i.e., no collisions are detected, the master will issue a STOP condition and the bus state will change back to idle. If a collision is detected, 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).

19.5 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 19-12 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 19-12.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.

19.5.1 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.

19.5.1.1 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.

19.5.1.2 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.

19.5.1.3 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.

19.5.1.4 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.

19.5.2 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.

19.5.3 Receiving Data Packets

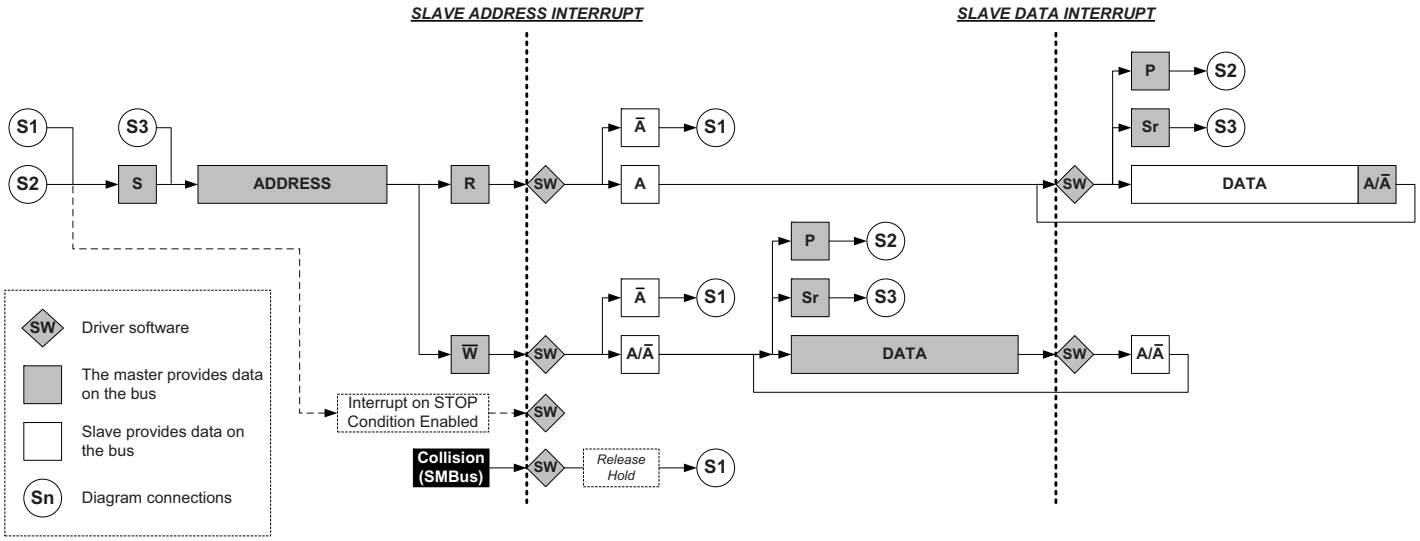
Assuming case M4 above, the master has already received one byte from the slave. The master read interrupt flag is set, and the master must prepare to receive new data. The master must respond to each byte with ACK or NACK. Indicating a NACK might not be successfully executed, as arbitration can be lost during the transmission. If a collision is detected, the master loses arbitration and the arbitration lost flag is set.

19.6 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 19-13. shows the TWI slave operation. The diamond shapes symbols (SW) indicate where software interaction is required.

Figure 19-13.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.

19.6.1 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.

19.6.1.1 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.

19.6.1.2 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.

19.6.1.3 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.

19.6.1.4 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.

19.6.2 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.

19.6.3 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.

19.7 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.

19.8 Register Description – TWI

19.8.1 CTRL – Common Control Register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	SDAHOLD[1:0]		EDIEN
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 2:1 – 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.

Table 19-1. SDA hold time.

SDAHOLD[1:0]	Group Configuration	Description
00	OFF	SDA hold time off
01	50NS	Typical 50ns hold time
10	300NS	Typical 100ns 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 19-2 for details.

Table 19-2. 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.

19.9 Register Description – TWI Master

19.9.1 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

- **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.

19.9.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	TIMEOUT[1:0]		QCEN	SMEN
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 – 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 19-3 lists the timeout settings.

Table 19-3. 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.

19.9.3 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 19-4 lists the acknowledge actions.

Table 19-4. 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 19-5. 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 19-5. CMD bit 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.

19.9.4 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 19-6. The change of bus state is dependent on bus activity. Refer to the "TWI Bus State Logic" on page 238.

Table 19-6. 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.

19.9.5 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.

19.9.6 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.

19.9.7 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.

19.10 Register Description – TWI Slave

19.10.1 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

- **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.

19.10.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	–	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

- **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 – 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 19-7 lists the acknowledge actions.

Table 19-7. 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 19-8. 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 19-8. 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.

19.10.3 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 19-9. TWI slave address or stop.

AP	Description
0	A STOP condition generated the interrupt on APIF
1	Address detection generated the interrupt on APIF

19.10.4 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.

19.10.5 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.

19.10.6 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.

19.11 Register Summary - TWI

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	–	SDAHOLD[1:0]		EDIEN	243
+0x01	MASTER	Offset address for TWI Master								
+0x08	SLAVE	Offset address for TWI Slave								

19.12 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	–	–	–	244
+0x01	CTRLB	–	–	–	–	TIMEOUT[1:0]		QCEN	SMEN	244
+0x02	CTRLC	–	–	–	–	–	ACKACT	CMD[1:0]		245
+0x03	STATUS	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]		246
+0x04	BAUD	BAUD[7:0]								247
+0x05	ADDR	ADDR[7:0]								248
+0x06	DATA	DATA[7:0]								248

19.13 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	TPMEN	SMEN	249
+0x01	CTRLB	–	–	–	–	–	ACKACT	CMD[1:0]		249
+0x02	STATUS	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP	251
+0x03	ADDR	ADDR[7:0]								252
+0x04	DATA	DATA[7:0]								252
+0x05	ADDRMAS	ADDRMASK[7:1]							ADDREN	253

19.14 Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	SLAVE_vect	TWI slave interrupt vector
0x02	MASTER_vect	TWI master interrupt vector

20. SPI – Serial Peripheral Interface

20.1 Features

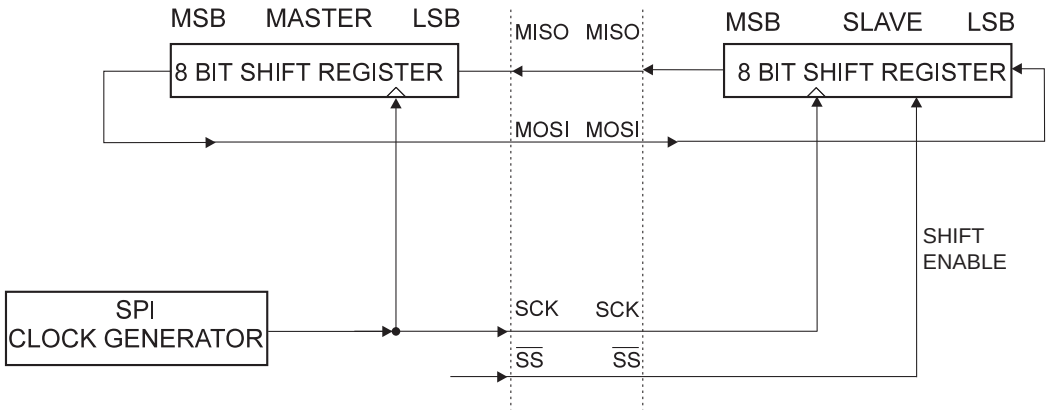
- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- Lsb first or msb first data transfer
- Eight programmable bit rates
- Interrupt flag at the end of transmission
- Write collision flag to indicate data collision
- Wake up from idle sleep mode
- Double speed master mode

20.2 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 20-1 on page 255. 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 20-1. SPI master-slave interconnection.



The 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.

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 20-1. The pins with user-defined direction must be configured from software to have the correct direction according to the application.

Table 20-1. 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

20.3 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, the SPI clock generator stops and the SPI interrupt flag is set. The master may continue to shift the next byte by writing new data to the DATA register, or can signal the end of the transfer by pulling the $\overline{SS}$ line high. The last incoming byte will be kept in the buffer register.

If the $\overline{SS}$ pin is not used and is configured as input, it must be held high to ensure master operation. If the $\overline{SS}$ 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 SPI interrupt flag is set.

20.4 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 reset, and the SPI slave will not receive any new data. Any partially received packet in the shift register will be dropped.

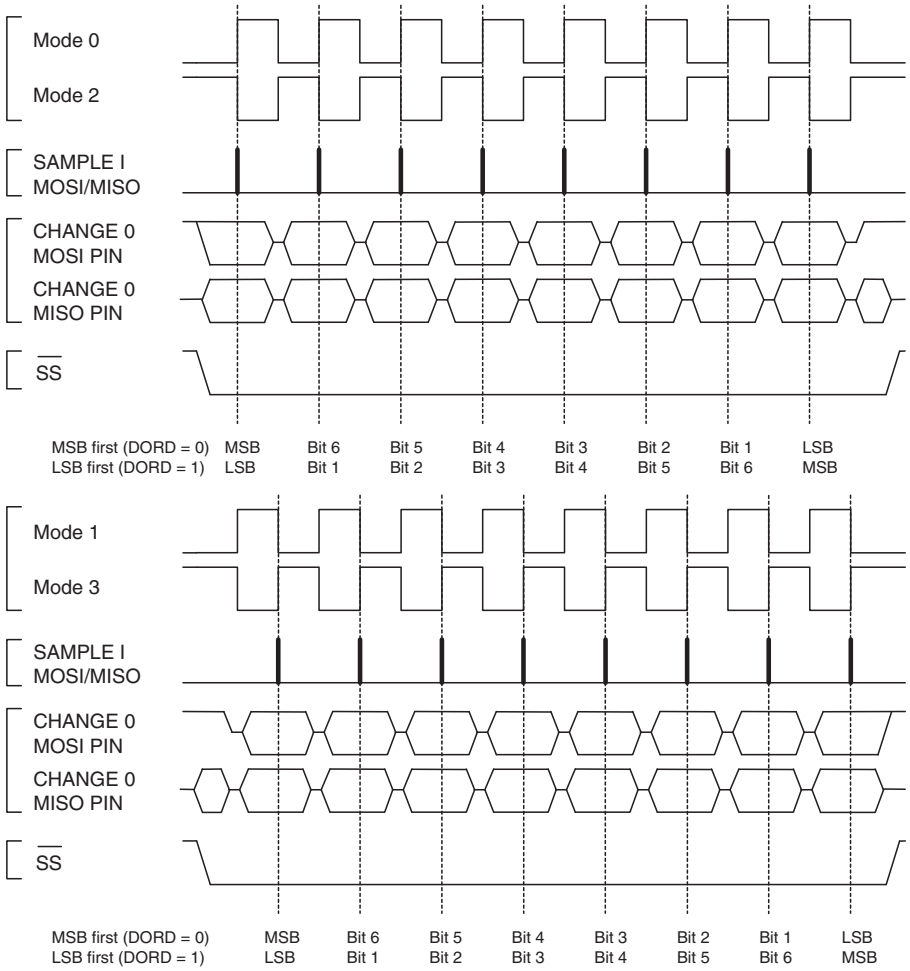
As the $\overline{SS}$ pin is used to signal the start and end of a transfer, it is also useful for doing packet/byte synchronization, keeping the slave bit counter synchronous with the master clock generator.

20.5 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 20-2. 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 20-2. SPI transfer modes.



20.6 DMA Support

DMA support on the SPI module is available only in slave mode. The SPI slave can trigger a DMA transfer as one byte has been shifted into the DATA register. It is possible, however, to use the XMEGA USART in SPI mode and then have DMA support in master mode. For details, refer to “USART in Master SPI Mode” on page 273.

20.7 Register Description

20.7.1 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 (see Table 20-3 on page 259).

- **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 20-2. 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 20-2. SPI transfer mode

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 20-3.

Table 20-3. 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	$Clk_{PER}/64$

20.7.2 INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	–	–	–	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 the SPI interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will be triggered when IF in the STATUS register is set.

20.7.3 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x02	IF	WRCOL	–	–	–	–	–	–
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – IF: Interrupt Flag**

This flag is set when a serial transfer is complete and one byte is completely shifted in/out of the 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 5:0 – 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.

20.7.4 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

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 shift register receive buffer to be read, returning the last byte successfully received.

20.8 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]		258
+0x01	INTCTRL	–	–	–	–	–	–	INTLVL[1:0]		259
+0x02	STATUS	IF	WRCOL	–	–	–	–	–	–	259
+0x03	DATA	DATA[7:0]								260

20.9 Interrupt vector Summary

Offset	Source	Interrupt Description
0x00	SPI_vect	SPI interrupt vector

21. USART

21.1 Features

- Full-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 and 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
- 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

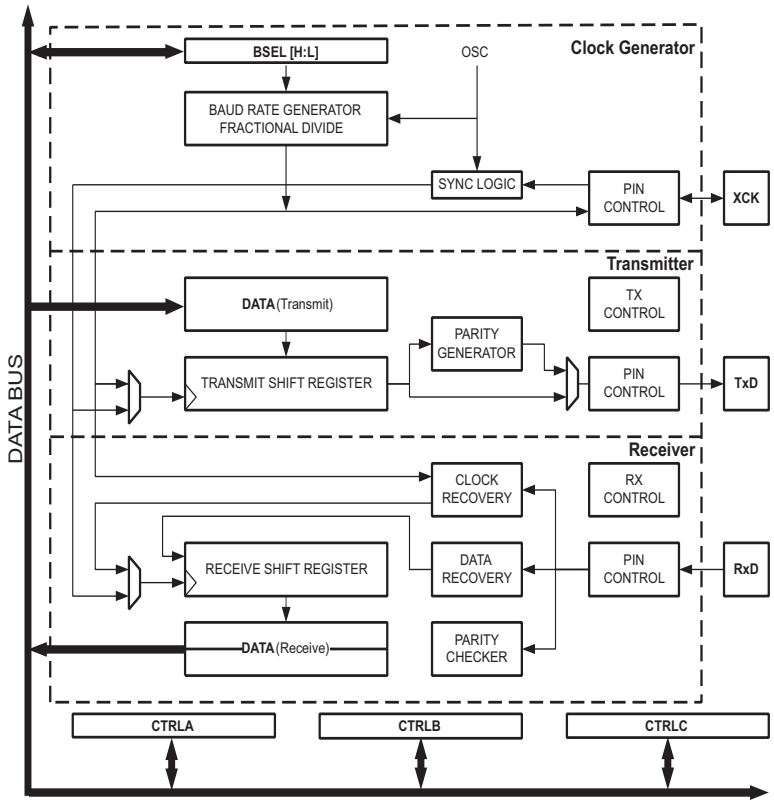
21.2 Overview

The universal synchronous and asynchronous serial receiver and transmitter (USART) is a fast and flexible serial communication module. The USART supports full-duplex communication and asynchronous and synchronous operation. The USART can be configured to operate 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 is shown in Figure 21-1 on page 262. The main functional blocks are the clock generator, the transmitter, and the receiver, which are indicated in dashed boxes.

Figure 21-1. 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.

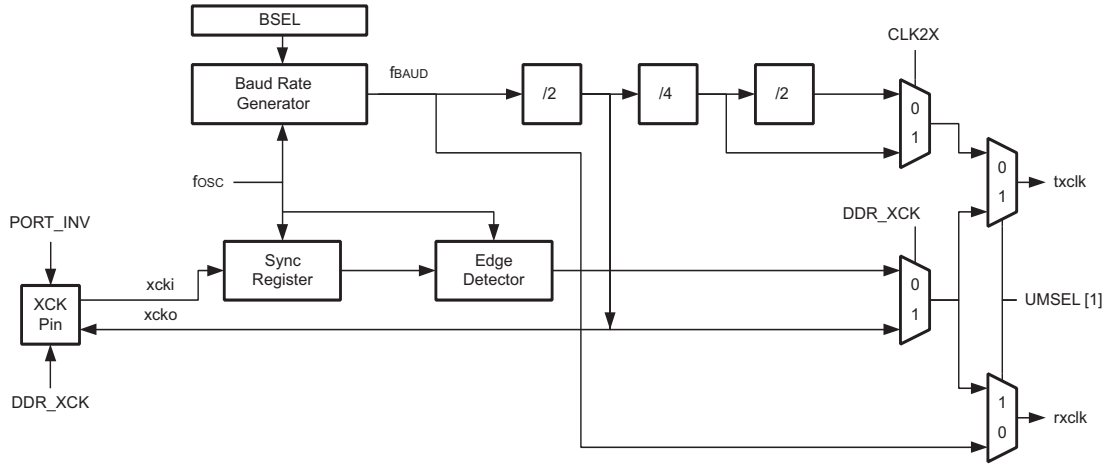
When the USART is set in master SPI mode, all USART-specific logic is disabled, leaving the transmit and receive buffers, shift registers, and baud rate generator enabled. Pin control and interrupt generation are identical in both modes. The registers are used in both modes, but their functionality differs 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 282.

21.3 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 21-2. Clock generation logic, block diagram.



21.3.1 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 21-1 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 "Fractional Baud Rate Generation" on page 271.

Table 21-1. Equations for calculating baud rate register settings.

Operating Mode	Conditions	Baud Rate[1] Calculation	BSEL Value Calculation
Asynchronous normal speed mode (CLK2X = 0)	BSCALE ≥ 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 ≥ 0 $f_{BAUD} \leq \frac{f_{PER}}{8}$	$f_{BAUD} = \frac{f_{PER}}{2^{BSCALE} \cdot 8 \cdot (BSEL + 1)}$	$BSEL = \frac{f_{PER}}{2^{BSCALE} \cdot 8 f_{BAUD}} - 1$
	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	$f_{BAUD} < \frac{f_{PER}}{2}$	$f_{BAUD} = \frac{f_{PER}}{2 \cdot (BSEL + 1)}$	$BSEL = \frac{f_{PER}}{2 f_{BAUD}} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bits per second (bps)

For BSEL=0, all baud rates must 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

21.3.2 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:

$$f_{XCK} < \frac{f_{PER}}{4}$$

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 or the peripheral clock must be increased accordingly.

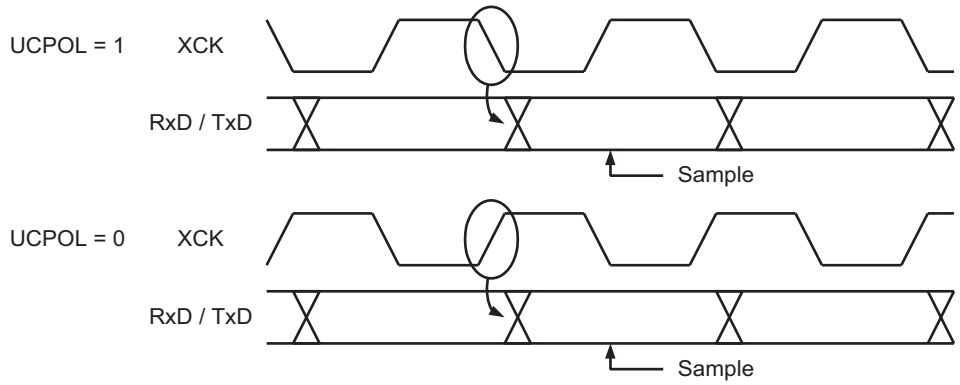
21.3.3 Double Speed Operation

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 21-1 on page 263 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 268 for more details.

21.3.4 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 21-3. 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 123.

21.3.5 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 21-1 on page 263).

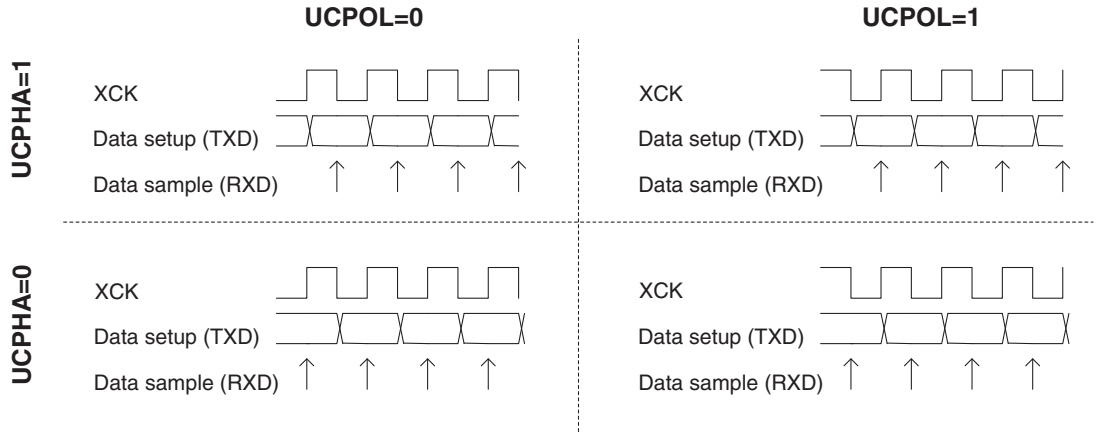
There are four combinations of the SPI clock (SCK) phase and polarity with respect to the serial data, and these are determined by the clock phase (UCPHA) control bit and the inverted I/O pin (INVEN) settings. The data transfer timing diagrams are shown in Figure 21-4 on page 266. Data bits are shifted out and latched in on opposite edges of the XCK signal, ensuring sufficient time for data signals to stabilize. The UCPHA and INVEN settings are summarized in Table 21-2. Changing the setting of any of these bits during transmission will corrupt both the receiver and transmitter

Table 21-2. 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 21-4. UCPHA and INVEN data transfer timing diagrams.



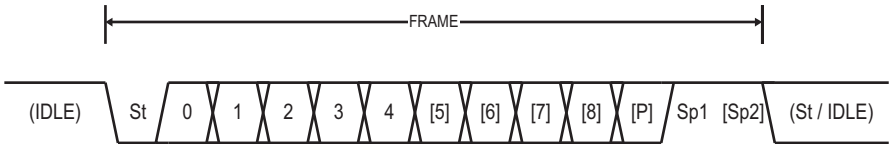
21.4 Frame Formats

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 267). The USART accepts all combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even, or odd parity bit
- 1 or 2 stop bits

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 21-5 on page 266 illustrates the possible combinations of frame formats. Bits inside brackets are optional.

Figure 21-5. Frame formats.



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit, may be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). The IDLE state is always high.

21.4.1 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).

21.4.2 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 two selectable frame formats:

- 8-bit data, msb first
- 8-bit data, lsb first

After a complete, 8-bit frame is transmitted, a new frame can directly follow it, or the communication line can return to the idle (high) state.

21.5 USART Initialization

USART initialization should use the following sequence:

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. 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.

21.6 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 123.

21.6.1 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.

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.

21.6.2 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. 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.

21.7 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.

21.7.1 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.

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.

21.7.2 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.

21.7.3 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.

21.7.4 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.

21.7.5 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.

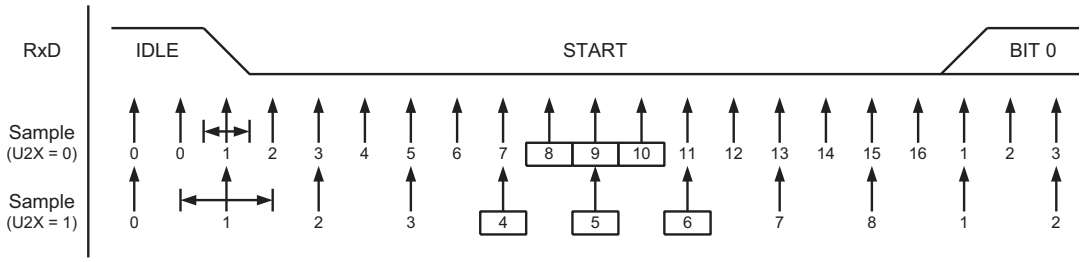
21.8 Asynchronous Data Reception

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.

21.8.1 Asynchronous Clock Recovery

The clock recovery unit synchronizes the internal clock to the incoming serial frames. Figure 21-6 on page 269 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 21-6. Start bit sampling.

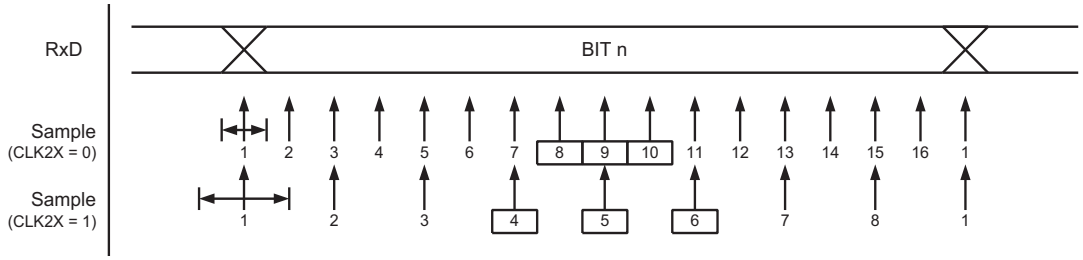


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Sample 1 denotes the first zero-sample, as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for normal mode and samples 4, 5, and 6 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.

21.8.2 Asynchronous Data Recovery

The data recovery unit uses sixteen samples in normal mode and eight samples in double speed mode for each bit. Figure 21-7 on page 269 shows the sampling process of data and parity bits.

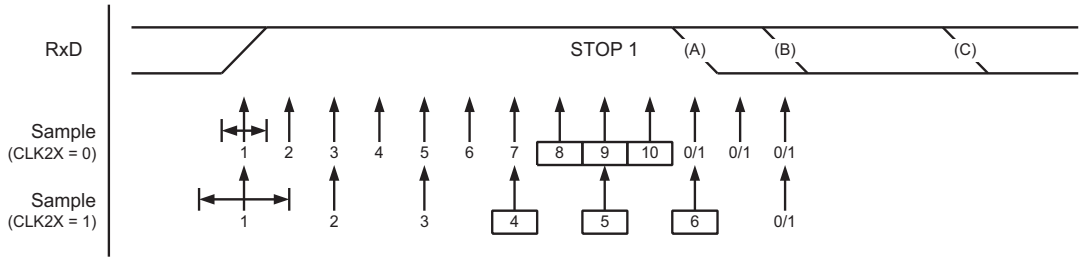
Figure 21-7. Sampling of data and parity bits.



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 21-8 on page 269 shows the sampling of the stop bit in relation to the earliest possible beginning of the next frame's start bit.

Figure 21-8. 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.

21.8.3 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.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D	Sum of character size and parity size (D = 5 to 10 bits).
S	Samples per bit. S = 16 for normal speed mode and S = 8 for double speed mode.
S_F	First sample number used for majority voting. S_F = 8 for normal speed mode and S_F = 4 for double speed mode.
S_M	Middle sample number used for majority voting. S_M = 9 for normal speed mode and S_M = 5 for double speed mode.
R_{slow}	The ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate.
R_{fast}	The ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 21-3 and Table 21-4 on page 270 list the maximum receiver baud rate error that can be tolerated. Normal speed mode has higher tolerance of baud rate variations.

Table 21-3. Recommended maximum receiver baud rate error for normal speed mode.

D #(Data + Parity Bit)	R_{slow} [%]	R_{fast} [%]	Max Total Error [%]	Recommended 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 21-4. Recommended maximum receiver baud rate error for double speed mode.

D #(Data + Parity Bit)	R_{slow} [%]	R_{fast} [%]	Max Total Error [%]	Recommended 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

D #(Data + Parity Bit)	R_{slow} [%]	R_{fast} [%]	Max Total Error [%]	Recommended Max Receiver Error [%]
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 for the maximum receiver baud rate error assume that the receiver and transmitter equally divide the maximum total error.

21.9 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 21-9 on page 272 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.

Table 21-5 on page 272 shows BSEL and BSCALE settings when using the internal oscillators to generate the most commonly used baud rates for asynchronous operation and how reducing the BSCALE can be used to reduce the baud rate error even further.

Figure 21-9. Fractional baud rate example.

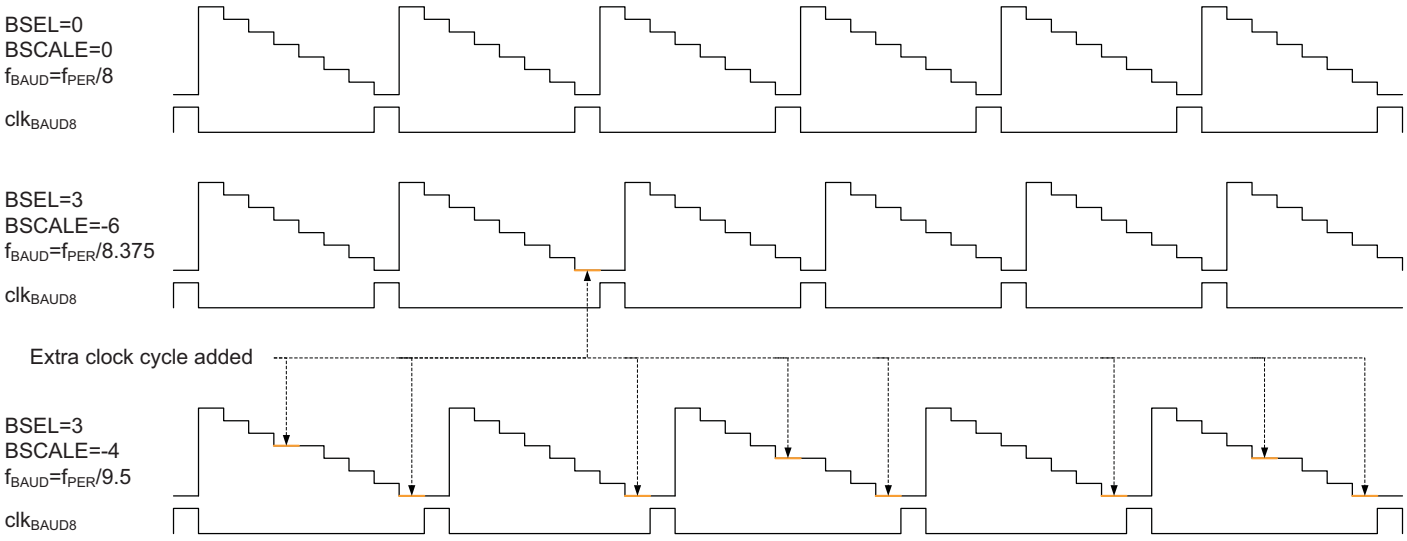


Table 21-5. USART baud rate.

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

Baud rate (bps)	f_{osc} = 32.0000MHz					
	CLK2X = 0			CLK2X = 1		
	BSEL	BSCALE	Error [%]	BSEL	BSCALE	Error [%]
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
3.0M	–	–	–	11	-5	-0.8
				43	-7	-0.2
4.0M	–	–	–	0	0	0.0
Max	2.0Mbps			4.0Mbps		

21.10 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.

21.11 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 differences exist:

- The USART transmitter in master SPI mode includes buffering, but the SPI module has no transmit buffer
- The USART receiver in master SPI mode includes an additional buffer level
- 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 Table 21-6.

Table 21-6. Comparison of USART in master SPI mode and SPI pins.

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

21.12 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.

21.12.1 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.

21.13 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. For details, refer to “IRCOM - IR Communication Module” on page 282.

21.14 DMA Support

DMA support is available on UART, USRT, and master SPI mode peripherals. For details on different USART DMA transfer triggers, refer to “Transfer Triggers” on page 49.

21.15 Register Description

21.15.1 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 (TXB) and USART receive data buffer register (RXB) share the same I/O address and is referred to as USART data register (DATA). The TXB register is the destination for data written to the DATA register location. Reading the DATA register location returns the contents of the RXB register.

For 5-bit, 6-bit, 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 be written only when DREIF in the STATUS register is set. Data written to the DATA register when DREIF is not set will be ignored by the USART transmitter. When data are 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 are then transmitted on the TxD pin.

The receive buffer consists of a two-level FIFO. Always read STATUS before DATA in order to get the correct status of the receive buffer.

21.15.2 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x01	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	–	RXB8
Read/Write	R	R/W	R	R	R	R	R	R/W
Initial Value	0	0	1	0	0	0	0	0

- **Bit 7 – RXCIF: Receive Complete Interrupt Flag**

This flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). When the receiver is disabled, the receive buffer will be flushed, and consequently 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 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: 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). 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**

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. 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., 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. FERR is not affected by setting the number of stop bits used, as it always uses only 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**

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) with a new character waiting in the receive shift register and a new start bit is detected. 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, this flag 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 266.

This flag is not used in master SPI mode operation.

- **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 – 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 in master SPI mode operation.

21.15.3 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[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 – RXCINTLVL[1:0]: Receive Complete Interrupt Level**

These bits enable the receive complete interrupt and select the interrupt level, as described in "Interrupts and Programmable Multilevel Interrupt Controller" on page 115. The enabled interrupt will be triggered when the RXCIF flag 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 "Interrupts and Programmable Multilevel Interrupt Controller" on page 115. The enabled interrupt will be triggered when the TXCIF flag 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 "Interrupts and Programmable Multilevel Interrupt Controller" on page 115. The enabled interrupt will be triggered when the DREIF flag in the STATUS register is set.

21.15.4 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	RXEN	TXEN	CLK2X	MPCM	TXB8
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 – 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 pending 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. For synchronous operation, this bit has no effect and should always be written to zero. This bit must be zero when the USART communication mode is configured to IRCOM.

This bit is unused in master SPI mode operation.

- **Bit 1 – MPCM: Multiprocessor Communication Mode**

This bit enables the multiprocessor 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 274.

This bit is unused in master SPI mode operation.

- **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 unused in master SPI mode operation.

21.15.5 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x05	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]		
+0x05[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	1	1	0

Note: 1. Master SPI mode.

- **Bits 7:6 – CMODE[1:0]: Communication Mode**

These bits select the mode of operation of the USART as shown in Table 21-7.

Table 21-7. 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)

Notes: 1. See "IRCOM - IR Communication Module" on page 282 for full description on using IRCOM mode.
2. See "USART in Master SPI Mode" on page 273 for full description of the master SPI operation.

- **Bits 5:4 – PMODE[1:0]: Parity Mode**

These bits enable and set the type of parity generation according to Table 21-8. 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.

These bits are unused in master SPI mode operation.

Table 21-8. PMODE bit settings.

PMODE[1:0]	Group Configuration	Parity 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 21-9. The receiver ignores this setting.

This bit is unused in master SPI mode operation.

Table 21-9. SBODE bit settings.

SBMODE	Stop Bit(s)
0	1
1	2

- **Bit 2:0 – CHSIZE[2:0]: Character Size**

The CHSIZE[2:0] bits set the number of data bits in a frame according to Table 21-10 on page 280. The receiver and transmitter use the same setting.

Table 21-10. CHSIZE bit 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 is only for master SPI mode, and 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.

- **Bit 1 – UCPHA: Clock Phase**

This bit is only for master SPI mode, and the bit 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 265 for details.

21.15.6 BAUDCTRLA – Baud Rate 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**

These are the lower 8 bits of the 12-bit BSEL value used for USART baud rate setting. BAUDCTRLB contains the four most-significant bits. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing BSEL will trigger an immediate update of the baud rate prescaler. See the equations in Table 21-1 on page 263.

21.15.7 BAUDCTRLB – Baud Rate 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. See the equations in Table 21-1 on page 263.

- **Bit 3:0 – BSEL[11:8]: Baud Rate bits**

These are the upper 4 bits of the 12-bit value used for USART baud rate setting. BAUDCTRLA contains the eight least-significant bits. 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.

21.16 Register Summary

21.16.1 Register Description - 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]								276
+0x01	STATUS	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	–	RXB8	276
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	CTRLA	–	–	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		277
+0x04	CTRLB	–	–	–	RXEN	TXEN	CLK2X	MPCM	TXB8	278
+0x05	CTRLC	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]			278
+0x06	BAUDCTRL	BSEL[7:0]								280
+0x07	BAUDCTRL	BSCALE[3:0]				BSEL[11:8]				280

21.16.2 Register Description - USART in SPI Master 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]								276
+0x01	STATUS	RXCIF	TXCIF	DREIF	–	–	–	–	–	276
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	CTRLA	–	–	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		277
+0x04	CTRLB	–	–	–	RXEN	TXEN	–	–	–	278
+0x05	CTRLC	CMODE[1:0]		–	–	–	UDORD	UCPHA	–	278
+0x06	BAUDCTRL	BSEL[7:0]								280
+0x07	BAUDCTRL	BSCALE[3:0]				BSEL[11:8]				280

21.17 Interrupt Vector Summary

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

22. IRCOM - IR Communication Module

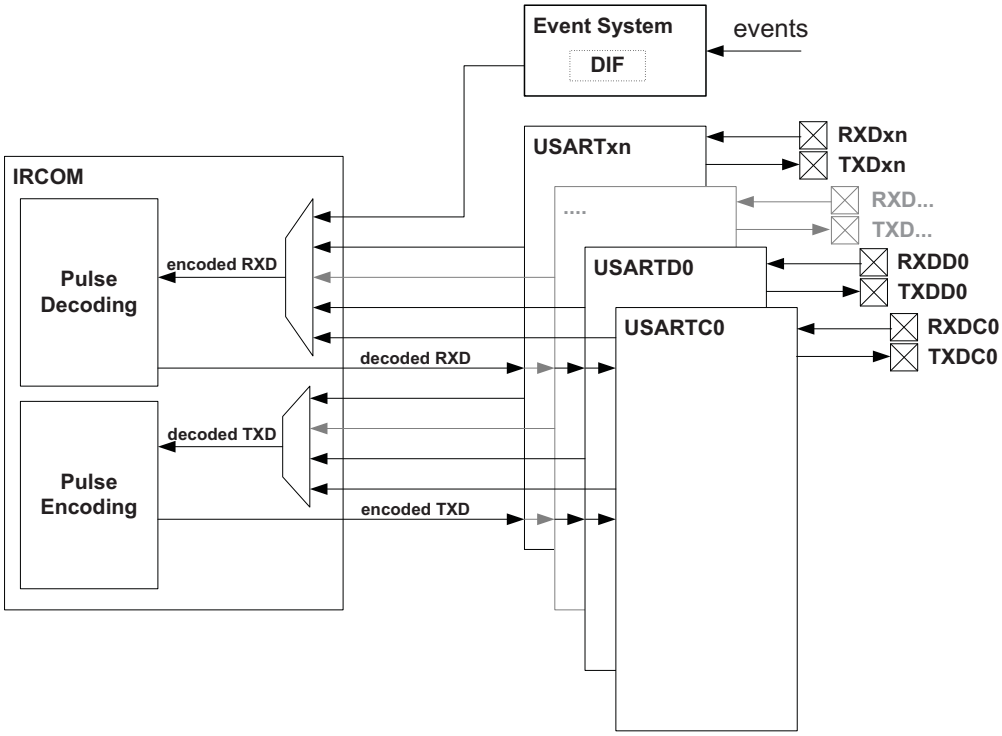
22.1 Features

- 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

22.2 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 22-1. 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 22-1 on page 282. 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.

22.2.1 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 63” for details on using the event system.

22.3 Registers Description

22.3.1 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).

22.3.2 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).

22.3.3 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 22-1. If event input is selected for the IRCOM receiver, the input from the USART's RX pin is automatically disabled.

Table 22-1. 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}

22.4 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]				284
+0x01	TXPLCTRL	TXPLCTRL[7:0]								284
+0x02	RXPLCTRL	RXPLCTRL[7:0]								284

23. AES and DES Crypto Engines

23.1 Features

- Data Encryption Standard (DES) CPU instruction
- Advanced Encryption Standard (AES) crypto module
- DES Instruction
  - Encryption and decryption
  - DES supported
  - Encryption/decryption in 16 CPU clock cycles per 8-byte block
- AES crypto module
  - Encryption and decryption
  - Supports 128-bit keys
  - Supports XOR data load mode to the state memory
  - Encryption/decryption in 375 clock cycles per 16-byte block

23.2 Overview

The Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are two commonly used standards for cryptography. These are supported through an AES peripheral module and a DES CPU instruction, and the communication interfaces and the CPU can use these for fast, encrypted communication and secure data storage.

DES is supported by an instruction in the AVR CPU. The 8-byte key and 8-byte data blocks must be loaded into the register file, and then the DES instruction must be executed 16 times to encrypt/decrypt the data block.

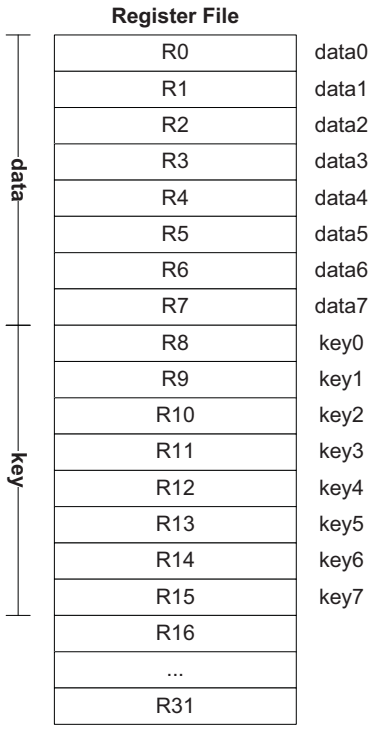
The AES crypto module encrypts and decrypts 128-bit data blocks with the use of a 128-bit key. The key and data must be loaded into the key and state memory in the module before encryption/decryption is started. It takes 375 peripheral clock cycles before the encryption/decryption is done. The encrypted/encrypted data can then be read out, and an optional interrupt can be generated. The AES crypto module also has DMA support with transfer triggers when encryption/decryption is done and optional auto-start of encryption/decryption when the state memory is fully loaded.

23.3 DES Instruction

The DES instruction is a single cycle instruction. In order to decrypt or encrypt a 64-bit (8-byte) data block, the instruction has to be executed 16 times.

The data and key blocks must be loaded into the register file before encryption/decryption is started. The 64-bit data block (plaintext or ciphertext) is placed in registers R0-R7, where the LSB of data is placed in R0 and the MSB of data is placed in R7. The full 64-bit key (including parity bits) is placed in registers R8-R15, with the LSB of the key in R8 and the MSB of the key in R15.

Figure 23-1. Register file usage during DES encryption/decryption.



Executing one DES instruction performs one round in the DES algorithm. Sixteen rounds must be executed in increasing order to form the correct DES ciphertext or plaintext. Intermediate results are stored in the register file (R0-R15) after each DES instruction. After sixteen rounds, the key is located in R8-R16 and the encrypted/decrypted ciphertext/plaintext is located in R0-R7. The instruction's operand (K) determines which round is executed, and the half carry flag (H) in the CPU status register determines whether encryption or decryption is performed. If the half carry flag is set, decryption is performed, and if the flag is cleared, encryption is performed.

For more details on the DES instruction, refer to the AVR instruction set manual.

23.4 AES Crypto Module

The AES crypto module performs encryption and decryption according to the Advanced Encryption Standard (FIPS-197). The 128-bit key block and 128-bit data block (plaintext or ciphertext) must be loaded into the key and state memories in the AES crypto module. This is done by writing the AES KEY register and STATE register sequentially with 16 bytes.

It is software selectable whether the module should perform encryption or decryption. It is also possible to enable XOR mode, where all new data loaded to the state key is XORed with the current data in the state memory.

The AES module uses 375 clock cycles before the encrypted/decrypted plaintext/ciphertext is available for readout in the state memory.

The following setup and use procedure is recommended:

1. Enable the AES interrupt (optional).
2. Select the AES direction to encryption or decryption.
3. Load the key data block into the AES key memory.
4. Load the data block into the AES state memory.
5. Start the encryption/decryption operation.

If more than one block is to be encrypted or decrypted, repeat the procedure from step 3.

When the encryption/decryption procedure is complete, the AES interrupt flag is set and an optional interrupt is generated.

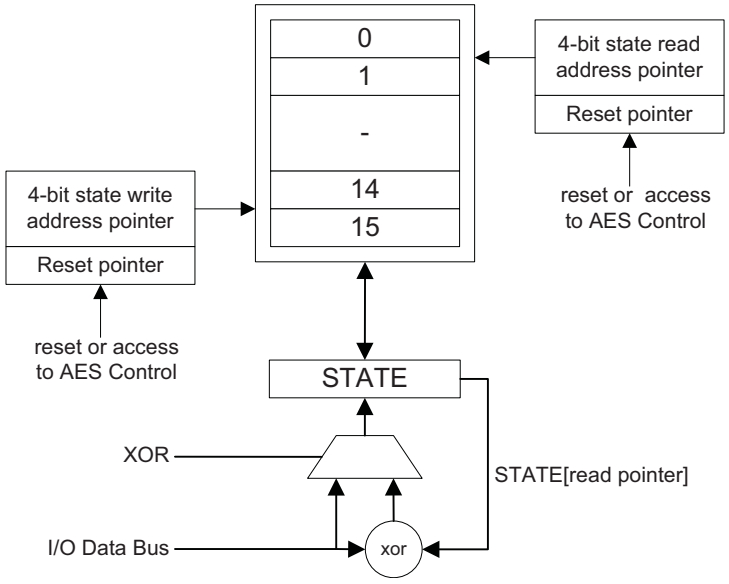
23.4.1 Key and State Memory

The AES key and state memory are both 16 x 8-bit memories that are accessible through the KEY and STATE registers, respectively.

Each memory has two 4-bit address pointers used to address the memory for read and write, respectively. The initial value of the pointers is zero. After a read or write operation to the STATE or KEY register, the appropriate pointer is automatically incremented. Accessing (read or write) the control register (CTRL) will reset all pointers to zero. A pointer overflow (a sequential read or write done more than 16 times) will also set the affected pointer to zero. The pointers are not accessible from software. Read and write memory pointers are both incremented during write operations in XOR mode.

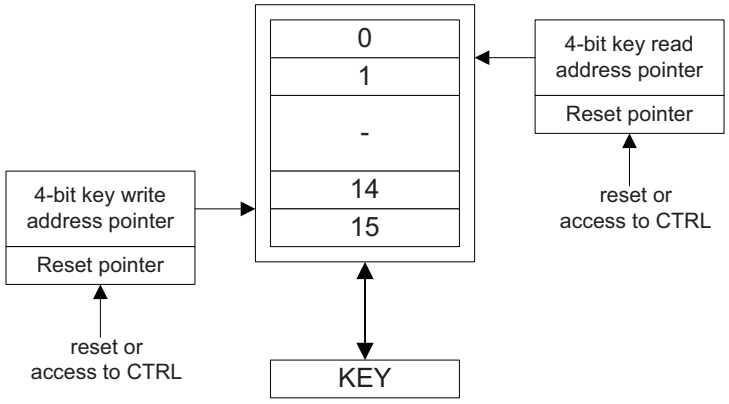
Access to the KEY and STATE registers is possible only when encryption/decryption is not in progress.

Figure 23-2. The state memory with pointers and register.



The state memory contains the AES state throughout the encryption/decryption process. The initial value of the state is the initial data (i.e., plaintext in the encryption mode, and ciphertext in the decryption mode). The last value of the state is the encrypted/decrypted data.

Figure 23-3. The key memory with pointers and register.



In the AES crypto module, the following definition of the key is used:

- In encryption mode, the key is the one defined in the AES standard.

- In decryption mode, the key is the last subkey of the expanded key defined in the AES standard.

In decryption mode, the key expansion procedure must be executed by software before operation with the AES crypto module so that the last subkey is ready to be loaded through the KEY register. Alternatively, this procedure can be run in hardware by using the AES crypto module to process a dummy data block in encryption mode using the same key. After the end of the encryption, reading from the key memory allows the last subkey to be obtained; i.e., get the result of the key expansion procedure. Table 23-1 shows the results of reading the key, depending on the mode (encryption or decryption) and status of the AES crypto module.

Table 23-1. The result of reading the key memory at different stages.

Encryption		Decryption	
Before data processing	**After data processing**	**Before data processing**	**After Data Processing**
Same key as loaded	The last subkey generated from the loaded key	Same key as loaded	The initial key generated from the last loaded subkey

23.4.2 DMA Support

The AES module can trigger a DMA transfer when the encryption/decryption procedure is complete. For more details on DMA transfer triggers, refer to “Transfer Triggers” on page 49.

23.5 Register Description – AES

23.5.1 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x00	START	AUTO	RESET	DECRYPT	–	XOR	–	–
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – START: Start/Run**

Setting this bit starts the encryption/decryption procedure, and this bit remains set while the encryption/decryption is ongoing. Writing this bit to zero will stop/abort any ongoing encryption/decryption process. This bit is automatically cleared if the SRIF or the ERROR flags in STATUS are set.

- **Bit 6 – AUTO: Auto Start Trigger**

Setting this bit enables the auto-start mode. In auto-start mode, the START bit will trigger automatically and start the encryption/decryption when all of the following conditions are met:

- The AUTO bit is set before the state memory is loaded
- All memory pointers (state read/write and key read/write) are zero
- State memory is fully loaded

If all of these conditions are not met, the encryption/decryption will be started with an incorrect key.

- **Bit 5 – RESET: Software Reset**

Setting this bit will reset the AES crypto module to its initial status on the next positive edge of the peripheral clock. All registers, pointers, and memories in the module are set to their initial value. When written to one, the bit stays high for one clock cycle before it is reset to zero by hardware.

- **Bit 4 – DECRYPT: Decryption / Direction**

This bit sets the direction for the AES crypto module. Writing this bit to zero will set the module in encryption mode. Writing one to this bit sets the module in decryption mode.

- **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 – XOR: State XOR Load Enable**

Setting this bit enables a XOR data load to the state memory. When this bit is set, the data loaded to the state memory are bitwise XORed with the data currently in the state memory. Writing this bit to zero disables XOR load mode, and new data written to the state memory will overwrite the current data.

- **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.

23.5.2 STATUS – AES Status register

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

- **Bit 7 – ERROR: Error**

The ERROR flag indicates an illegal handling of the AES crypto module. The flag is set in the following cases:

- Setting START in the control register while the state memory and/or key memory are not fully loaded or read. This error occurs when the total number of read/write operations from/to the STATE and KEY registers is not a multiple of 16 before an AES start.
- Accessing (read or write) the control register while the START bit is one.

This flag can be cleared by software by writing one to its bit location.

- **Bit 6:1 – 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 – SRIF: State Ready Interrupt flag**

This flag is the interrupt/DMA request flag, and is set when the encryption/decryption procedure is completed and the state memory contains valid data. As long as the flag is zero, this indicates that there is no valid encrypted/decrypted data in the state memory.

The flag is cleared by hardware when a read access is made to the state memory (the first byte is read). Alternatively, the bit can be cleared by writing a one to its bit location.

23.5.3 STATE – AES State register

Bit	7	6	5	4	3	2	1	0
+0x02	**STATE[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 STATE register is used to access the state memory. Before encryption/decryption can take place, the state memory must be written sequentially, byte-by-byte, through the STATE register. After encryption/decryption is done, the ciphertext/plaintext can be read sequentially, byte-by-byte, through the STATE register.

Loading the initial data to the STATE register should be done after setting the appropriate AES mode and direction. This register can not be accessed during encryption/decryption.

23.5.4 KEY – Key register

Bit	7	6	5	4	3	2	1	0
+0x03	**KEY[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 KEY register is used to access the key memory. Before encryption/decryption can take place, the key memory must be written sequentially, byte-by-byte, through the KEY register. After encryption/decryption is done, the last subkey can be read sequentially, byte-by-byte, through the KEY register.

Loading the initial data to the KEY register should be done after setting the appropriate AES mode and direction.

23.5.5 INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	–	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 priority and enable**

These bits enable the AES interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will be triggered when the SRIF in the STATUS register is set.

23.6 Register summary – AES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	bit 0	Page
+0x00	CTRL	START	AUTO	RESET	DECRYPT	–	XOR	–	–	290
+0x01	STATUS	ERROR	–	–	–	–	–	–	SRIF	291
+0x02	STATE	STATE[7:0]								291
+0x03	KEY	KEY[7:0]								291
+0x04	INTCTRL	–	–	–	–	–	–	INTLVL[1:0]		292
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	

23.7 Interrupt vector summary

Table 23-2. AES interrupt vector and its offset word address.

Offset	Source	Interrupt Description
0x00	AES_vect	AES interrupt vector

24. CRC – Cyclic Redundancy Check Generator

24.1 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, DMA controller and CPU
  - Continuous CRC on data going through a DMA 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

24.2 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 present 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 this 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: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

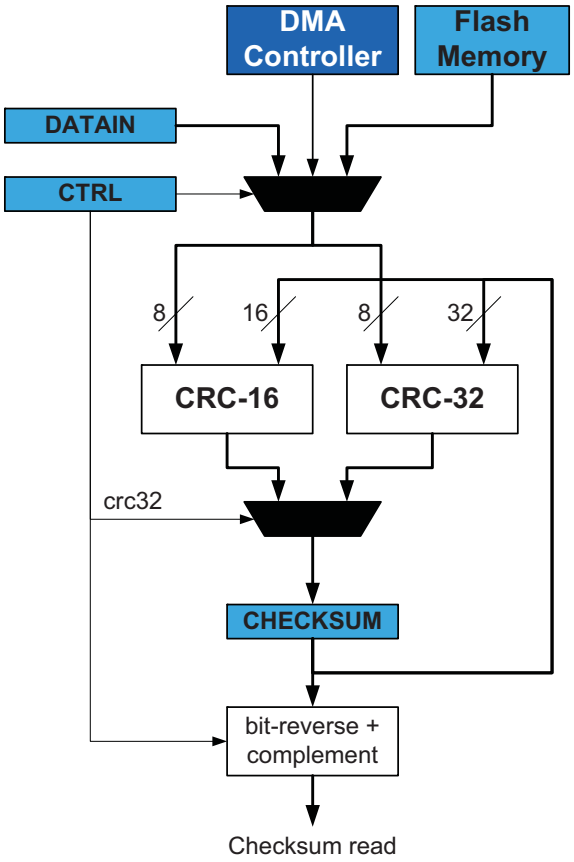
  Hex value: 0x04C11DB7

24.3 Operation

The data source for the CRC module must be selected in software as either flash memory, the DMA 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 24-1).

For the I/O interface or DMA 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 24-1. CRC generator block diagram.



24.4 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 375.

24.5 CRC on DMA Data

CRC-16 or CRC-32 calculations can be performed on data passing through any DMA channel. Once a DMA channel is selected as the source, the CRC module will continuously generate the CRC on the data passing through the DMA channel. The checksum is available for readout once the DMA transaction is completed or aborted. A CRC can be

performed not only on communication data, but also on data in SRAM or I/O memory by passing these data through a DMA channel. If the latter is done, the destination register for the DMA data can be the data input (DATAIN) register in the CRC module.

24.6 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.

24.7 Register Description

24.7.1 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 24-1. 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 DMA controller or flash memory.

Table 24-2. 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	DMACH0	DMA controller channel 0
0101	DMACH1	DMA controller channel 1
0110	DMACH2	DMA controller channel 2
0111	DMACH3	DMA controller channel 3
1xxx	–	Reserved for future use

24.7.2 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	–	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 should 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 a DMA channel if selected as the source, the flag is cleared when the DMA channel transaction is completed or aborted. If flash memory is selected as the source, the flag is cleared when the CRC generation is completed.

24.7.3 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.

24.7.4 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.

24.7.5 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.

24.7.6 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.

24.7.7 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.

24.8 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]				296
+0x01	STATUS	–	–	–	–	–	–	ZERO	BUSY	297
+0x02	Reserved	–	–	–	–	–	–	–	–	
+0x03	DATAIN	DATAIN[7:0]								297
+0x04	CHECKSU	CHECKSUM[7:0]								298
+0x05	CHECKSU	CHECKSUM[15:8]								298
+0x06	CHECKSU	CHECKSUM[23:16]								298
+0x07	CHECKSU	CHECKSUM[31:24]								298

25. LCD – Liquid Crystal Display

25.1 Features

- Display Capacity up to 40 Segment and up to 4 Common Terminals
- Supports up to 16 GPIO's
- Shadow Display Memory Gives Full Freedom in Segment Update
- ASCII Character Mapping
- Swap Capability Option on Common and/or Segment Terminal Buses
- Supports from Static up to 1/4 Duty
- Supports Static and 1/3 Bias
- LCD Driver Active in Power Save Mode for Low Power Operation
- Software Selectable Low Power Waveform
- Flexible Selection of Frame Frequency
- Programmable Blink Mode and Frequency
- Blink on two Segment Terminals
- Uses Only 32kHz RTC Clock Source
- On-chip LCD Power Supply
- Software Contrast Adjustment Control
- Equal Source and Sink Capability to Increase LCD Life Time
- Extended Interrupt Mode for Display Update or Wake-up from Sleep Mode

25.2 Overview

An LCD display is made of several segments (pixels or complete symbols) which can be visible or invisible. A segment has two electrodes with liquid crystal between them. These electrodes are the common terminal (COM pin) and the segment terminal (SEG pin). When a voltage above a threshold voltage is applied across the liquid crystal, the segment becomes visible. The voltage must alternate to avoid an electrophoresis effect in the liquid crystal, this effect degrades the display. Hence the voltage waveform across a segment must not have a DC-component.

The LCD controller is intended for monochrome passive liquid crystal display (LCD) with up to 4 Common terminals and up to 40 Segments terminals. If the application does not need all the LCD segments available on the XMEGA, up to 16 of the unused LCD pins can be used as general purpose I/O pins.

The LCD controller can be clocked by an internal or an external asynchronous 32kHz clock source. This 32kHz oscillator source selection is the same as for the Real Time Counter (RTC).

Dedicated Low Power Waveform, Contrast Control, Extended Interrupt Mode, Selectable Frame Frequency and Blink functionality are supported to offload the CPU, reduce interrupts and reduce power consumption.

To reduce hardware design complexity, the LCD includes integrated LCD buffers, an integrated power supply voltage and an innovative SWAP mode. Using SWAP mode, the hardware designers have more flexibility during board layout as they can rearrange the pin sequence on Segment and/or Common Terminal Buses.

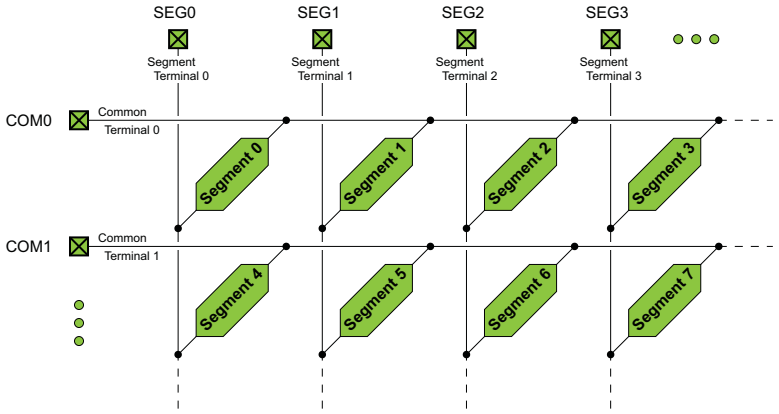
25.2.1 Definitions

Several terms are used when describing LCD. The definitions in Table 25-1 are used throughout this document.

Table 25-1. LCD definitions.

LCD	A passive display panel with terminals leading directly to a segment
Segment (or pixel)	A LCD panel active area within the display which can be turned “ON or “OFF”. This can be a single segment of a 7-segment character or a specific symbol (icon).
COM	Common terminal
SEG	Segment terminal
1 / Duty	1 / Number of common terminals on an actual LCD display
1 / Bias	1 / Number of voltage levels used driving a LCD display -1
Frame Rate	Number of times the LCD segments are energized per second

Figure 25-1. LCD Typical Connections



25.2.2 LCD Clock Sources

The LCD controller can be clocked by an internal or an external asynchronous 32kHz clock source. This 32kHz oscillator source selection is the same as for the Real Time Counter, RTCSRC bit-field in RTC control register (see Table 7-4 on page 87).

The clock source must be stable to obtain accurate LCD timing and hence minimize DC voltage offset across LCD segments.

25.2.3 LCD Prescaler

The prescaler consists of a 3-bit ripple counter and a 1 to 8-clock divider (see Figure 25-2 on page 301). The PRESC bit selects clk_{LCD} divided by 8 or 16 from the ripple counter.

If a finer resolution in frame rate is required, the CLKDIV bit-field can be used to divide the clock further by 1 to 8.

Output from the clock divider clk_{LCD_PS} is used as clock source for the LCD timing.

25.2.4 LCD Display Memory

The Display Memory is available through I/O registers grouped for each common terminal.

A start of new frame triggers an update of the Shadow Display Memory. The content of Display Memory is saved into the Shadow Display Memory. A Display Memory refresh is possible without affecting data that is sent to the panel.

When a bit in the Display Memory is written to one, the corresponding segment will be energized (“ON”), and de-energized (“OFF”) when this bit is written to zero.

To energize a segment, an absolute voltage above a certain threshold must be applied. This is done by setting the SEG pin to opposite phase when the corresponding COM pin is active. For a display with more than one common terminal, two (1/3 bias) additional voltage levels must be applied. Otherwise, non-energized segments on COM0 would be energized for all non-selected common terminals.

Addressing COM0 starts a frame by driving an opposite phase with large amplitude on COM0 as against non addressed COM lines. Non-energized segments are in phase with the addressed COM0, and energized segments have opposite phase and large amplitude. For waveform figures refer to “Mode of Operation” on page 302.

DATA4 - DATA0 from Shadow Display Memory is multiplexed into the decoder. The decoder is controlled from the LCD timing and sets up signals controlling the analog switches to produce an output waveform.

Next, COM1 is addressed, and DATA9 - DATA5 from Shadow Display Memory is input to the decoder. Addressing continues until all COM lines are addressed according to the number of selected common terminals (duty).

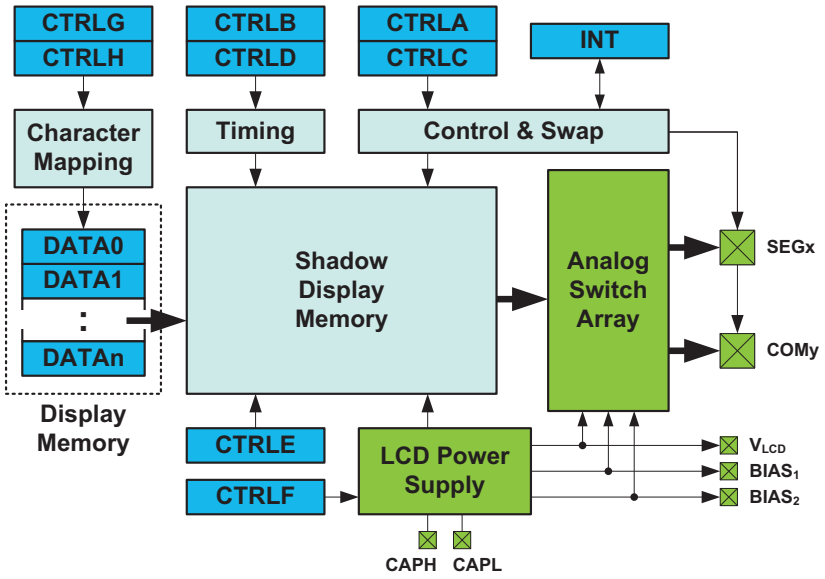
25.2.5 Minimizing Power Consumption

The power consumption of the LCD controller can be minimized by:

1. Using the lowest acceptable frame rate - Refer to the LCD glass technical characteristics.
2. Using the low power waveform - “Low Power Waveform” on page 303
3. Programming the lowest possible contrast value - “CTRLF – Control register F” on page 313.

25.3 Block Diagram

Figure 25-2. LCD Controller Block Diagram

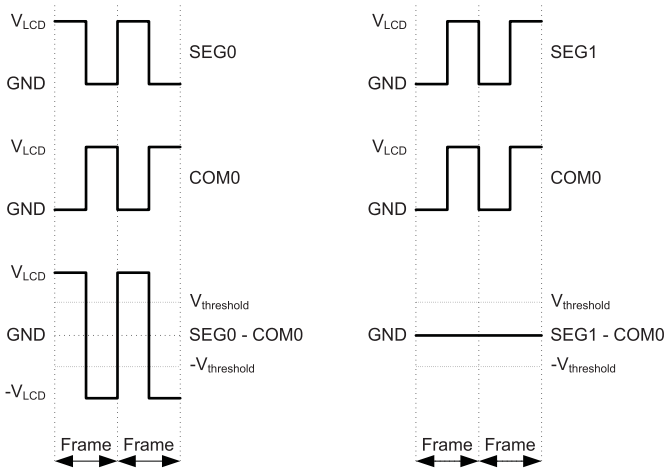


25.4 Mode of Operation

25.4.1 Static Duty and Static Bias

If all segments on an LCD have one common electrode, then, each segment must have a unique segment terminal. This kind of display is driven with the waveform shown in Figure 25-3 on page 302. SEG0-COM0 is the voltage across a segment that is “ON”, and SEG1-COM0 is the voltage across a segment that is “OFF”.

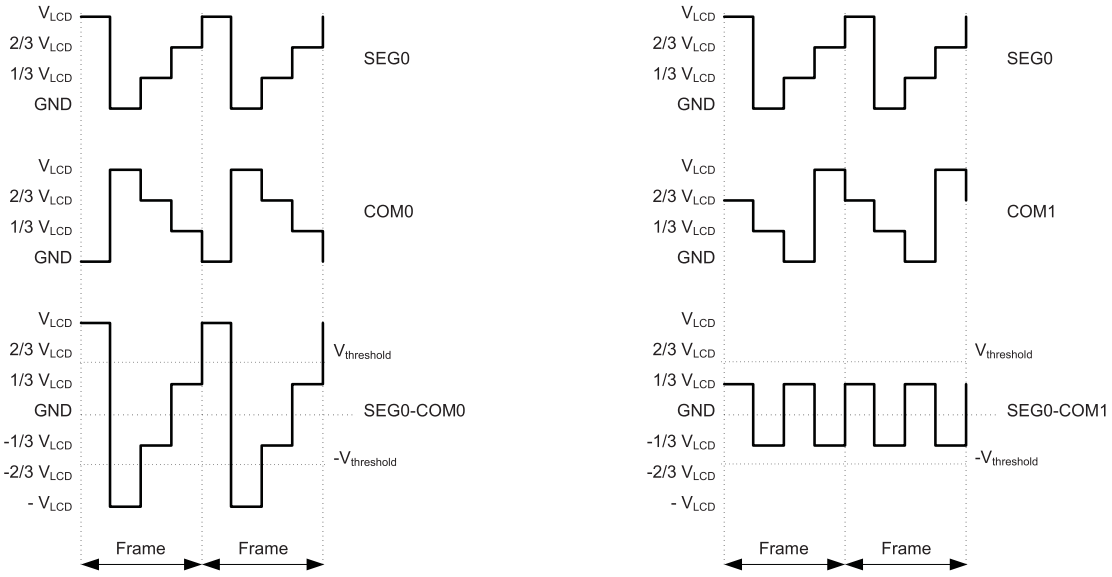
Figure 25-3. Driving an LCD With One Common Terminal



25.4.2 1/2 Duty and 1/3 Bias

For an LCD with two common terminals (1/2 duty) a more complex waveform must be used to individually control segments. The waveform is shown in Figure 25-4 on page 302. SEG0-COM0 is the voltage across a segment that is “ON”, and SEG0-COM1 is the voltage across a segment that is “OFF”.

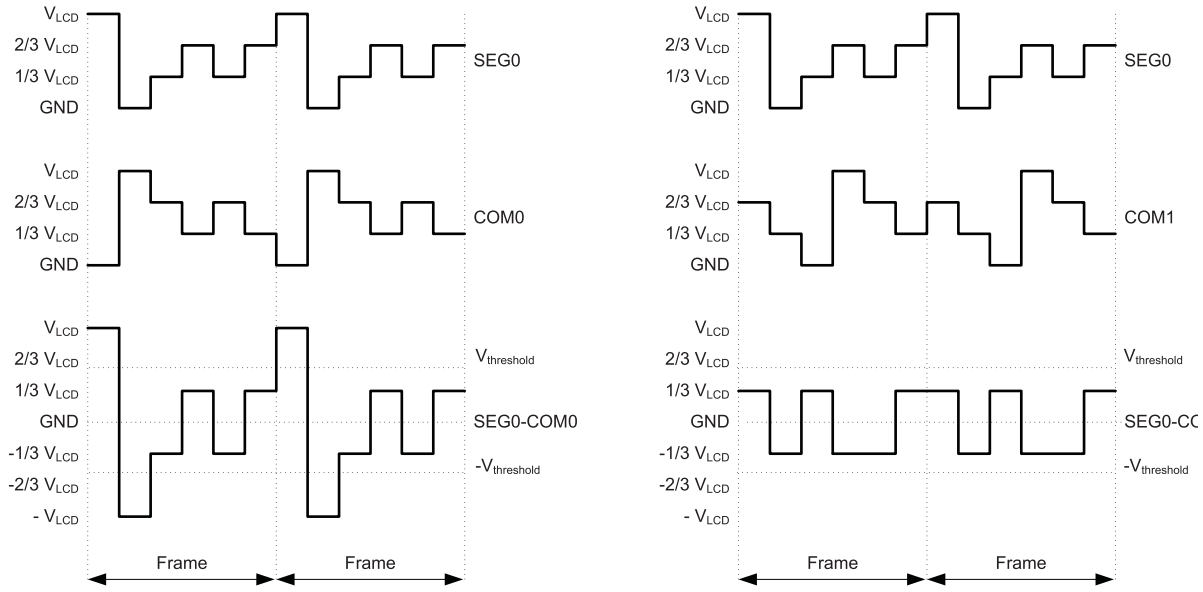
Figure 25-4. Driving an LCD With Two Common Terminals



25.4.3 1/3 Duty and 1/3 Bias

1/3 bias is usually recommended for an LCD with three common terminals (1/3 duty). The waveform is shown in Figure 25-5 on page 303. SEG0-COM0 is the voltage across a segment that is “ON” and SEG0-COM1 is the voltage across a segment that is “OFF”.

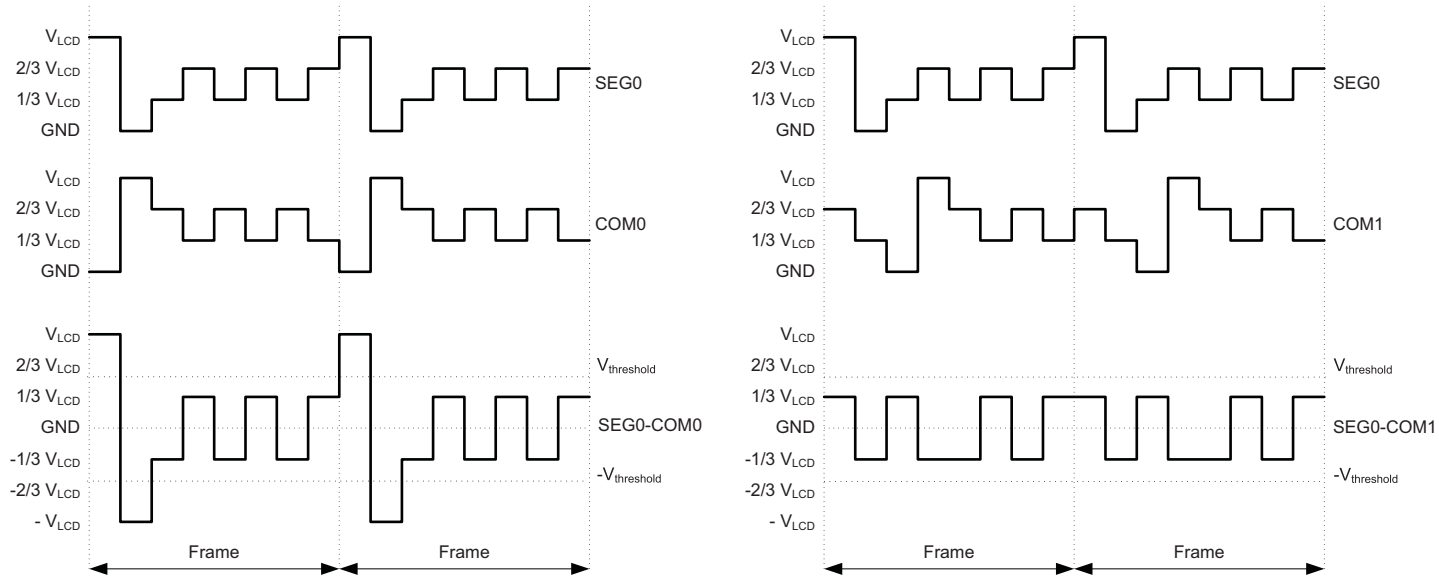
Figure 25-5. Driving an LCD With Three Common Terminals



25.4.4 1/4 Duty and 1/3 Bias

1/3 bias is optimal for an LCD displays with four common terminals (1/4 duty). The waveform is shown in Figure 25-6 on page 303. SEG0-COM0 is the voltage across a segment that is ON" and SEG0-COM1 is the voltage across a segment that is "OFF".

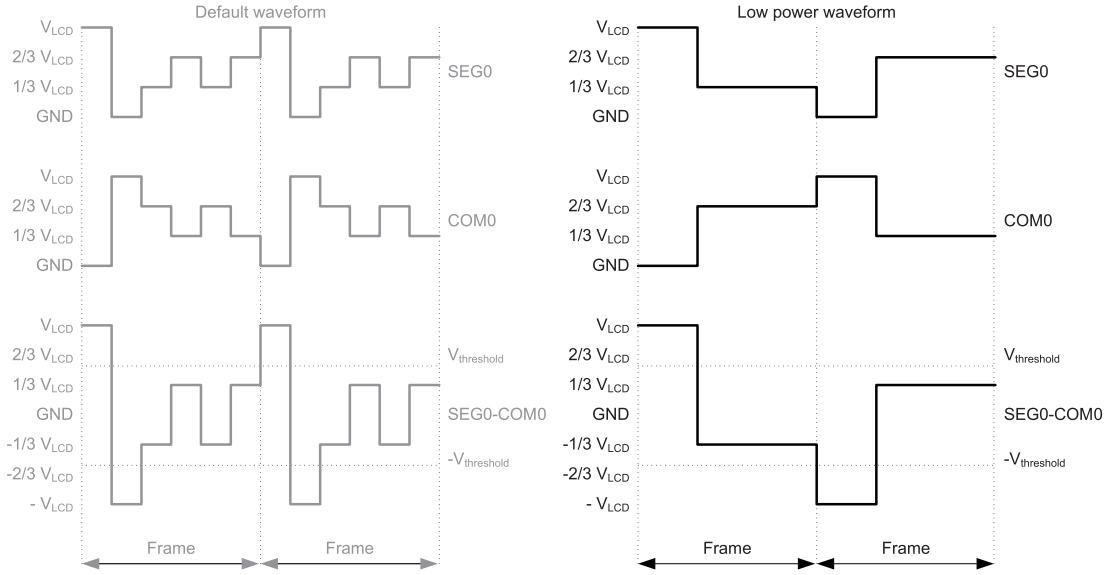
Figure 25-6. Driving an LCD With Four Common Terminals



25.4.5 Low Power Waveform

To reduce toggle activity and hence power consumption, a low power waveform (LPWAV=1) can be selected. The low power waveform requires two subsequent frames with the same display data to obtain zero DC voltage. Consequently, the interrupt flag is only set every two frames. Default and the low power waveform is shown in Figure 25-7 on page 304 for 1/3 duty and 1/3 bias. For other selections of duty and bias, the effect is similar.

Figure 25-7. Low Power Waveform With Three Common Terminals



25.4.6 Operation in Sleep Modes

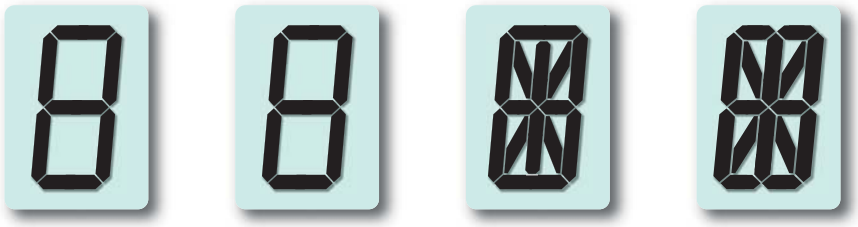
The LCD will continue to operate in Idle mode, in Power-save mode and in Extended Standby mode (blinking included).

25.4.7 ASCII Character Mapping

The LCD controller can automatically handle ASCII characters. Instead of setting and clearing segments of the digit, the user enters the ASCII code and the Digit Decoder updates itself the corresponding segment values in the Display Memory.

Up to 4 types of character mapping are supported.

Figure 25-8. ASCII Character Mapping



7-Segment
- 3 COM term.
- 3 SEG term.

7-Segment
- 4 COM term.
- 2 SEG term.

14-Segment
- 4 COM term.
- 4 SEG term.

16-Segment
- 3 COM term.
- 6 SEG term.

Character mapping saves execution time and allows a fast return to Power-save or Extended Standby mode after display updates.

25.4.8 Display Blanking

When BLANK is written to one, the LCD is blanked after the completion of the current frame. All segment and common pins are driven to GND, discharging the LCD. Data in the Display Memory is preserved. Display blanking should be used before disabling the LCD to avoid DC voltage across the segments, and a slowly fading image.

This mode differs from the one enabled by SEGON = 0 (in CTRLA register) where the segment and common pins are always driven by the programmed waveform and where all the segments are “OFF”.

25.4.9 Display Blinking

There are two ways to blink the display, controlled from software and controlled automatically by hardware.

25.4.9.1 Software Blinking

Setting / clearing segment(s) in the Display Memory allows software blinking. To blink simultaneously all enabled segments, SEGON bit in CRTLA register can be used. The blink rate is software dependant.

25.4.9.2 Hardware Blinking

Up to eight segments (pixels) can be configured to automatically blink. These segments must be connected to the segment terminal SEG1 and/or SEG0. This mode is enabled by setting the BLINKEN bit in the CTRLD register and defining the associated common terminal(s) in the CTRLE register.The blink rate frequency is configured by using the BLINKRATE bit-field in the CTRLD register. A segment will blink if its corresponding bit is set in the Display Memory, otherwise it will remain “OFF”.
If all bits in the CTRLE register are set to zero, then blinking is applied to all enabled segments.

The BLINK command will come into operation at the beginning of the next LCD frame.

Table 25-2. Blinking modes.

SEGON[1]	BLINKEN	BPS1[3:0] | BPS0[3:0]	Comment
0	x	0_b xxxx xxxx	All segments are “OFF”
1	0	0_b xxxx xxxx	All segments are driven by the corresponding data registers
1	1	0_b 0000 0000	All segments are blinking at the blink frequency
		Not equal to zero	Blinking only the selected segment(s) at the blink frequency

Notes: 1. SEGON bit in CTRLA register.

25.4.10 Extended Interrupt Mode

In standard interrupt mode (XIME[4:0]=0), the LCD controller can provide an interrupt every frames. When the extended interrupt mode is enabled, the LCD controller can provide the interrupt every XIME[4:0]+1 frames.

This mode provides an embedded time base for user. This time base can be used by the software in charge of display updates (i.e. scrolling text, progress bar, ...).

The extended interrupt mode saves real time resources and allows the application to stay longer in Power-save or Extended Standby mode.

25.4.11 LCD Power Supply

The LCD power supply manages all voltages for LCD buffers. The XBIAS bit in the CTRLA register defines the source of V_{LCD}. If XBIAS is cleared, V_{LCD} sources voltages from the Bandgap Reference. Otherwise, V_{LCD} must be powered externally.

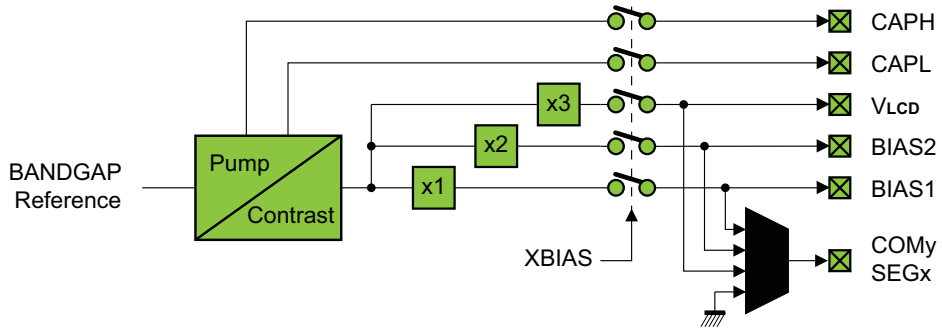
Note that when using external V_{LCD}, the fine contrast controlled by FCONT[5:0] bits of the CRTLG register is inoperative.

Table 25-3. LCD power supply pins behavior.

ENABLE[1]	XBIAS[1]	VLCD (pin)	BIAS2	BIAS1	CAPH / CAPL
0	x	H.Z.	H.Z.	H.Z.	H.Z.
1	0	V_{LCD}	$^2/_3$ V_{LCD} (also in static mode)	$^1/_3$ V_{LCD} (also in static mode)	Pump voltage
	1	Input for V_{LCD}	- Input for BIAS2 - H.Z. if static bias	- Input for BIAS1 - H.Z. if static bias	H.Z.

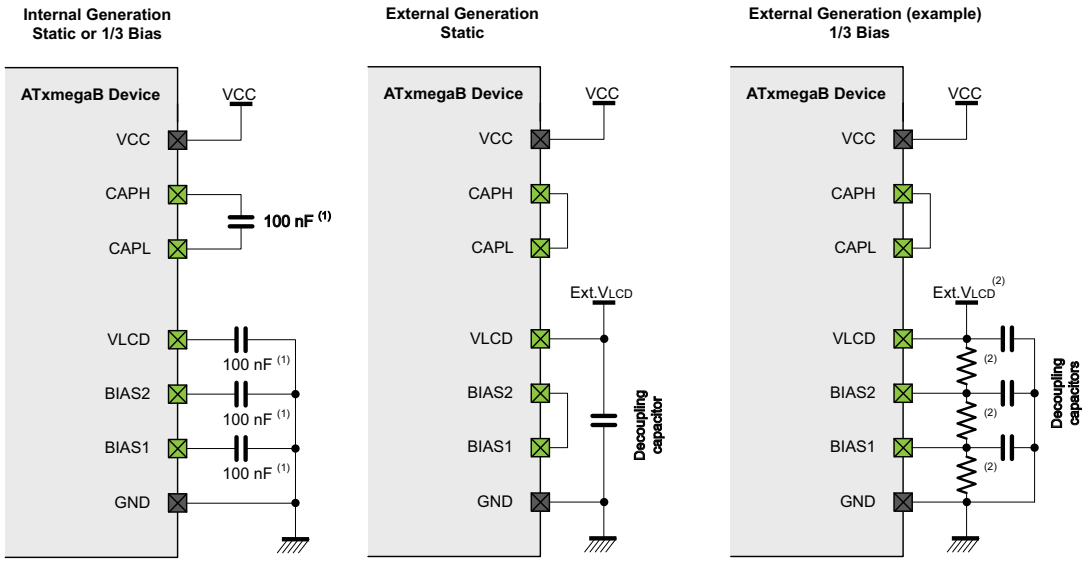
Notes: 1. ENABLE and XBIAS bits of the CTRLA register.

Figure 25-9. LCD Power Supply Block Diagram



Different application schemes for bias generation are shown in Figure 25-10 on page 306.

Figure 25-10.Analog Connections vs. Internal or External Bias Generation



Notes :
1: These values are provided for design guidance only. They should be optimized for the application by the designer based on actual LCD specifications.
2: Bias generation can be provided by other sources of voltage than a division resistor.

25.4.12 Segment and Common Buses Swapping[1]

Segment and/or common buses can be swapped (mirrored) to give more flexibility for LCD interconnects. The first segment (or common) terminal pin becomes the last one, and so on.
It is very useful in Chip on Glass (CoG), Chip on Film (CoF) or Chip on Board (CoB) technologies.

SEGSWP bit and COMSWP bit in the CRTLA register control the order of the respective terminal buses.

Note: 1. Refer to specific device datasheet for availability of this feature.

25.4.13 Port Mask

For LCD panels that do not use all the available segment terminals of the device, it is possible to mask some of the unused pins. PMSK bit-field in the CTRLC register defines the number of segment terminals used in the application. Up to 16 unused segment terminal pins can be used as standard GPIO pins. They are always placed at the end of the segment terminal bus. The 8 last pins of this bus will become PG[0:7] - Port G - and the following 8 pins will become PM[0:7] - Port M.

The GPIO functions on LCD pins are enabled if the corresponding segment terminal is masked or if the LCD controller is disabled. A pure segment terminal - not shared with GPIO - is grounded via a pull-down resistor if it is masked or if the LCD controller is disabled.

Note: SEGSWP bit, which reverses the segment terminal indexing, will be active even if the LCD controller is disabled (ENABLE bit in the CRTLA register) and will thus also modify the GPIO pin mapping.

Examples of a 40-segment LCD controller:

- If 30 segments are used:

Segment terminals [39:32]	=	PG[0:7]	, Port G	(GPIO functions)
Segment terminals [31:30]	=	PM[0:1]	, Port M	(GPIO functions)
Segment terminals [29:0]	=	SEG[29:0]	, LCD	(LCD functions)

- If 20 segments are used:

Segment terminals [39:32]	=	PG[0:7]	, Port G	(GPIO functions)
Segment terminals [31:24]	=	PM[0:7]	, Port M	(GPIO functions)
Segment terminals [23:20]	=	GND (pull down)		
Segment terminals [19:0]	=	SEG[19:0]	, LCD	(LCD functions)

25.5 Register Description – LCD

25.5.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	XBIAS	DATLCK	COMSWP	SEGSWP	CLRDT	SEGON	BLANK
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: LCD Enable**

Writing this bit to one enables the LCD. By writing it to zero, the LCD is turned “OFF” immediately. Turning the LCD “OFF” while driving a display, drives the output to ground to discharge the display (apart from segment terminals which will be controlled by GPIO settings).

- **Bit 6 – XBIAS: External Bias Generation**

When this bit is set, the LCD buffers which drive the intermediate voltage levels are turned “OFF”. When XBIAS is “OFF”, an external source for V_{LCD} is necessary.

- **Bit 5 – DATLCK: Data Register Lock**

Writing this bit to one freezes the Shadow Display Memory. If the Display Memory is modified, the Shadow Display Memory is locked and the display remains unchanged. When the bit is cleared, the Shadow Display Memory is updated when a new frame starts (see Figure 25-2 on page 301).

- **Bit 4 – COMSWP: Common Terminal Bus Swap**(1)

Writing this bit to one inverts the order of the common terminal bus (COM[3:0]). The common terminals disabled by DUTY[1:0] are also affected (see Table 25-4).

Table 25-4. Common terminal bus reverse.

DUTY[1:0]	Number of COM	COMSWP = 0	COMSWP = 1
00	4	COM3,COM2,COM1,COM0	COM0,COM1,COM2,COM3
01	1	–, –, –, COM0	COM0, –, –, –
10	2	–, –, COM1, COM0	COM0, COM1, –, –
11	3	–, COM2, COM1, COM0	COM0, COM1, COM2, –

Note: 1. Refer to specific device datasheet for availability of this feature.

- **Bit 3 – SEGSWP: Segment Terminal Bus Swap**(1)

Writing this bit to one inverts completely the order of the segment terminal bus (SEG[39:0]). The segment terminals un-selected by PMSK[5:0] are also affected (see Table 25-5 on page 309).

Table 25-5. Segment terminal bus reverse (examples)

PMSK[5:0]	Number of SEG	SEGSWP = 0	SEGSWP = 1
000100	4	(*SEG [39:4] unused*), SEG[3:0]	SEG[0:3], (*SEG[4:39] unused*)
001000	8	(*SEG[39:8] unused*), SEG[7:0]	SEG[0:7], (*SEG[8:39] unused*)
010000	16	(*SEG[39:16] unused*), SEG[15:0]	SEG[0:15], (*SEG[16:39] unused*)
101000	40	SEG[39:0]	SEG[0:39]

Note: 1. Refer to specific device datasheet for availability of this feature.

- **Bit 2 – CLRDT: Clear Data Register**

Writing this bit to one clears immediately the Display Memory (but not the control registers). The display will be blanked after completion of a frame. This bit is automatically reset once the Display Memory is cleared.

- **Bit 1 – SEGON: Segments "ON".**

Writing this bit to one enables all segments and the contents of the Display Memory is output on the LCD. Writing it to zero, turns "OFF" all LCD segments.

This bit can be used to flash the LCD, leaving the LCD timing generator enabled.

- **Bit 0 – BLANK: Blanking Display Mode**

When this bit is written to one, the display will be blanked after completion of a frame. All segment and common terminals will be driven to ground. (For more details see "Display Blanking" on page 304). This function does not modify the Display Memory.

25.5.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	PRESC	CLKDIV[2:0]			LPWAV	–	DUTY[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 – PRESC: LCD Prescaler Select**

The PRESC bit selects a tap point from a ripple counter. The ripple counter output can be further divided by setting the Clock Divider (CLKDIV[2:0]). The different selections are shown in Table 25-6. Together they define the prescaler LCD clock (clk_{LCD_PS}), which is clocking the LCD controller.

Table 25-6. LCD prescaler selection.

PRESC	Output From Ripple Counter clk_{LCD} / N	Frame Rates (CLKDIV[2:0] = 0, DUTY = $^1/_4$)	
		$F(clk_{LCD})$ = 32kHz	$F(clk_{LCD})$ = 32768Hz
0	clk_{LCD} / 8	500 Hz	512 Hz
1	clk_{LCD} / 16	250 Hz	256 Hz

- **Bit 6:4 – CLKDIV[2:0]: LCD Clock Division**

The CLKDIV bit-field defines the division ratio in the clock divider. The various selections are shown in Table 25-7. This Clock Divider gives extra flexibility in frame rate setting.

Frame rate equation:

$$FrameRate = \frac{F(clk_{LCD})}{(K \times N \times (1 + CLKDIV))}$$

Where:

N = prescaler divider (8 or 16).
K = 8 for 1/4, 1/2 and static duty.
K = 6 for 1/3 duty.

Table 25-7. LCD clock divider (1/4 dyty).

CLKDIV[2:0]	Divided by	Frame Rate (1/4 Duty)			
		$F(clk_{LCD})$ = 32 kHz		$F(clk_{LCD})$ = 32768 Hz	
		N=8	N=16	N=8	N=16
000	1	500 Hz	250 Hz	512 Hz	256 Hz
001	2	250 Hz	125 Hz	256 Hz	128 Hz
010	3	166.667 Hz	83.333 Hz	170.667 Hz	85.333 Hz
011	4	125 Hz	62.5 Hz	128 Hz	64 Hz
100	5	100 Hz	50 Hz	102.4 Hz	51.2 Hz
101	6	83.333 Hz	41.667 Hz	85.333 Hz	42.667 Hz
110	7	71.429 Hz	35.714 Hz	73.143 Hz	36.671 Hz
111	8	62.5 Hz	31.25 Hz	64 Hz	32 Hz

Note that when using 1/3 duty, the frame rate is increased by 33% compared to the values listed above.

Table 25-8. Example of frame rate calculation.

clk_{LCD}	Duty	K	PRESC	N	CLKDIV[2:0]	Frame rate
32.768kHz	Static	8	1	16	4	32768 / (8 x 16 x (1 + 4)) = 51.2Hz
32.768kHz	1/2	8	1	16	4	32768 / (8 x 16 x (1 + 4)) = 51.2Hz
32.768kHz	1/3	6	1	16	4	32768 / (6 x 16 x (1 + 4)) = 68.267Hz
32.768kHz	1/4	8	1	16	4	32768 / (8 x 16 x (1 + 4)) = 51.2Hz

- **Bit 3 – LPWAV: Low Power Waveform**

When LPWAV is written to one, the low power waveform is outputted on LCD pins, otherwise the standard waveform is outputted. If this bit is modified during display operation the change takes place at the beginning of the next frame. (For more details see “Low Power Waveform” on page 303).

- **Bit 2 – 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 1:0 – DUTY[1:0]: Duty Select**[1]

The DUTY bit-field defines the duty cycle. Common pins that are not used will be driven to ground. The different duty selections are shown in Table 25-9.

Table 25-9. Duty cycle.

DUTY[1:0]	Duty	Bias	COM pins Used
0 0	1/4	1/3	COM[0:3]
0 1	Static	Static	COM0
1 0	1/2	1/3	COM[0:1]
1 1	1/3	1/3	COM[0:2]

Note: 1. Refer to specific device datasheet for duty cycles availability (linked to the number of available common terminals).

25.5.3 CTRLC – Control register C

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	PMSK[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 unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bits 5:0 – PMSK[5:0]: LCD Port Mask**

The PMSK bit-field defines the number of port pins to be used as segment drivers. The unused pins will be driven to ground except the 16 highest pins which become GPIO's.

25.5.4 INTCTRL – Interrupt Control register

Bit	7	6	5	4	3	2	1	0
+0x03	XIME[4:0]					–	FCINTLVL[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

- **Bits 7:3 – XIME[4:0]: eXtended Interrupt Mode Enable**

XIME bit-field defines the number of frames to be completed for one interrupt period.

$$\text{Interrupt Period} = ((\text{XIME}[4:0] + 1) \times 2^{\text{LPWAV}}) \text{ frames}$$

- For default waveforms, the FCIF flag is generated every XIME[4:0] + 1 frames. The range is 1 up to 32 frames.
- For low power waveforms requiring 2 subsequent frames, the FCIF flag is generated every 2 x (XIME[4:0] + 1) frames. The range is 2 up to 64 frames.

Note: This extended interrupt mode generates a stable time base from the frame rate.

- **Bit 2 – 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 1:0 – FCINTLVL[1:0]: Interrupt Level**

This bit-field enables the LCD frame completed interrupt and selects the interrupt level as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 121. The enabled interrupt will be triggered when the FCIF flag in the INTFLAGS register is set.

25.5.5 INTFLAGS – Interrupt Flag register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	–	–	**FCIF**
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 unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 – FCIF: LCD Frames Completed Interrupt Flag**

The generation of this flag depends on the XIME value in the INTCTRL register.
This bit is set by hardware at the beginning of a frame. FCIF is cleared by hardware when executing the corresponding interrupt handling routine. Alternatively, writing a logical one to the flag clears FCIF.

25.5.6 CTRLD – Control register D

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	–	–	**BLINKEN**	–	**BLINKRATE[1:0]**	
Read/Write	R	R	R	R	R/W	R	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 – BLINKEN: Blink Enable**

Writing this bit to one, the blink mode starts at the frequency specified by LCD blink rate (BLINKRATE). By writing it to zero, the LCD blink module stops. This BLINKEN bit takes effect at the beginning of the next LCD frame. (For more details see “Display Blinking” on page 305).

- **Bit 2 – 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 1:0 – BLINKRATE[1:0]: LCD Blink Rate**

The BLINKRATE bit-field defines the frequency of the hardware Display Blinking when the BLINKEN bit is set. Blink frequencies are shown in Table 25-10.

Table 25-10. Blink frequencies.

BLINKRATE[1:0]	Blink frequency
00	4Hz
01	2Hz
10	1Hz
11	0.5Hz

25.5.7 CTRLE – Control register E

Bit	7	6	5	4	3	2	1	0
+0x06	BPS1[3:0]				BPS0[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 – BPS1[3:0]: Blink Segment Selection 1**

This bit-field defines the segment which is connected on SEG1 for blinking. Each bit of BPS1[3:0] corresponds to one of the common terminals.

- **Bits 3:0 – BPS0[3:0]: Blink Segment Selection 0**

This bit-field defines the segment which is connected on SEG0 for blinking. Each bit of BPS0[3:0] corresponds to one of the common terminals.

Note: If no segment to blink is selected (BPS1[3:0] = BPS1[3:0] = 0) and if the BLINKEN bit is set, then the full display is blinking.

25.5.8 CTRLF – Control register F

Bit	7	6	5	4	3	2	1	0
+0x07	–	–	FCONT[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 unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bits 5:0 – FCONT[5:0]: Fine Contrast**

FCONT bit-field defines the maximum voltage clk_{LCD} on segment and common pins. FCONT is a signed number (two's complement). New values take effect at the beginning of each frame.

$$V_{LCD} = 3.0\ V + (\ FCONT[5:0] \times 0.016\ V\ )$$

25.5.9 CTRLG – Control register G

Bit	7	6	5	4	3	2	1	0
+0x08	TDG[1:0]		STSEG[5:0]					
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 7:6 – TDG[1:0]: Type of Digit**(1)

This bit-field specifies the number of segments and segment/common connections used to display a digit. See Table 25-11 and Figure 25-11 on page 314.

Table 25-11. Type of digits.

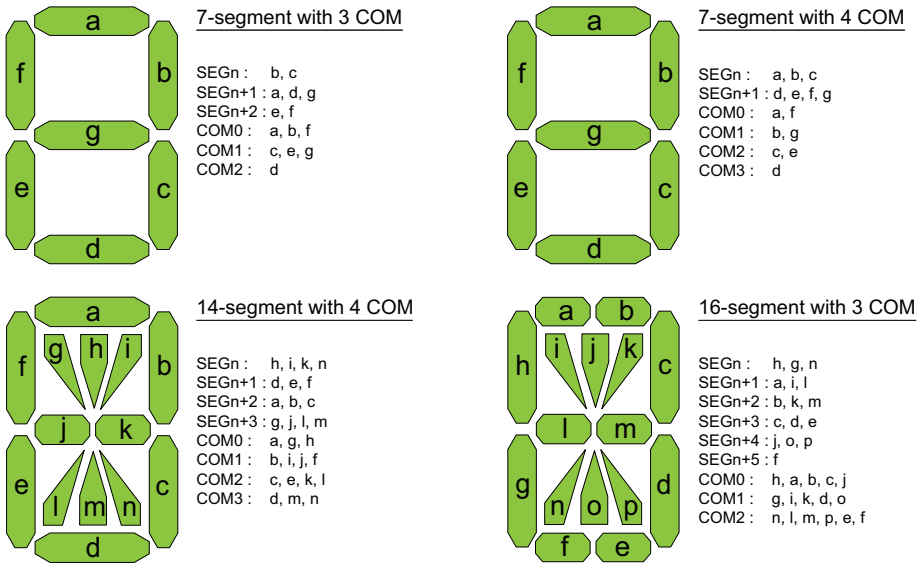
TDG[1:0]	Digit Type
00	7-segment with 3 common terminals, COM[2:0]
01	7-segment with 4 common terminals, COM[3:0]
10	14-segment with 4 common terminals, COM[3:0]
11	16-segment with 3 common terminals, COM[2:0]

Note: 1. Refer to specific device datasheet for “Type of Digit” availability.

- **Bits 5:0 – STSEG[5:0]: Start Segment**

STSEG bit-field defines the first segment terminal used to write the decoded display. This bit-field is automatically incremented or decremented (according to the DEC value of CTRLH register) by the number of segment terminals used in the digit.

Figure 25-11.Segment and Common Terminal Connections for Digit



25.5.10 CTRLH – Control register H

Bit	7	6	5	4	3	2	1	0
+0x09	**DEC**	**DCODE[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 – DEC: Decrement of Start Segment**

Writing this bit to one automatically decrements the STSEG bit-field of CTRLG register by the number of segment terminals used by the digit. If this bit is written to zero, the STSEG bit-field is incremented by the number of segment terminals used by the digit. This action takes place once the digit decoding is finished and prepares the next call to the Digit Decoder.

- **Bits 6:0 – DCODE[6:0]: Display Code**

DCODE bit-field will be computed by the Digit Decoder, and converted to display codes, and then automatically written into the Display Memory according to the STSEG value. This Digit Decoder can be used when the LCD panel is defined with one or more of the configurations above in Figure 25-11 on page 314.

The Table 25-12 on page 315, Table 25-13 on page 316 and Table 25-14 on page 317 show the DCODE[6:0] and display pattern.

The table entry code, DCODE [6:0], is the 7-bit ASCII code of the digit.

Table 25-12. 7-segments Character Table.

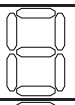
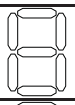
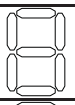
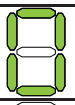
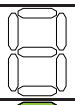
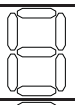
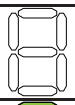
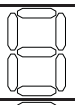
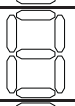
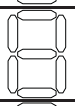
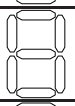
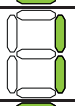
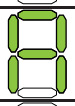
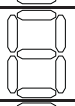
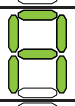
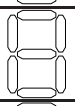
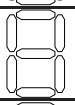
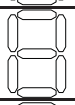
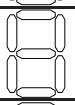
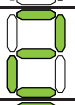
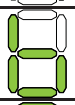
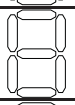
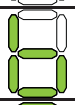
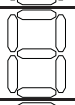
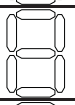
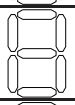
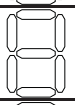
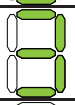
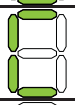
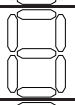
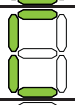
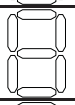
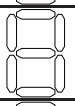
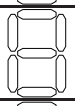
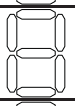
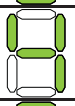
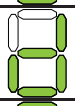
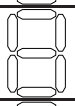
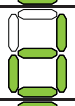
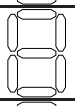
	x000	x001	x010	x011	x100	x101	x110	x111
0000								
0001								
0010								
0011								
0100								
0101								
0110								
0111								
1000								
1001								
1010								
1011								
1100								
1101								
1110								
1111								

Table 25-13. 14-segments Character Table.

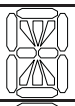
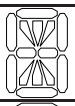
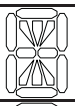
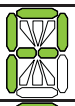
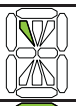
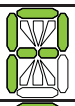
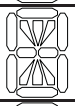
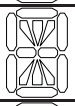
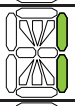
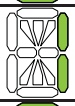
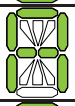
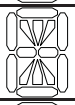
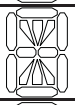
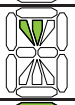
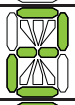
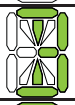
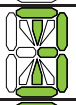
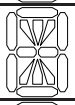
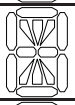
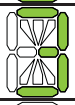
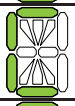
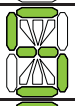
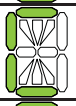
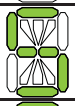
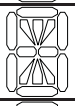
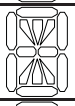
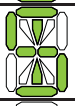
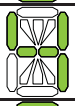
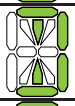
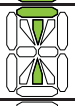
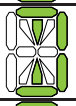
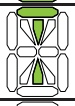
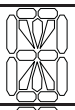
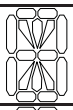
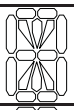
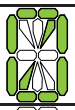
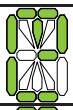
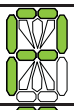
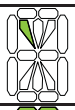
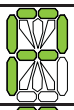
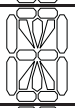
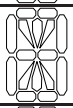
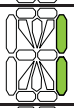
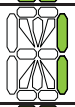
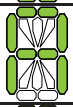
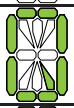
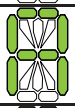
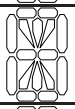
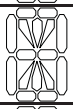
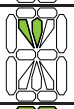
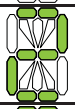
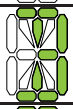
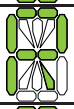
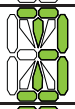
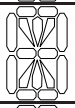
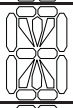
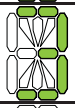
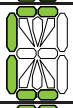
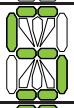
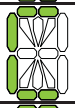
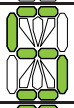
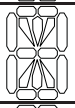
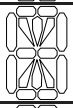
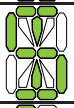
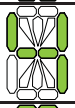
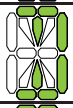
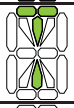
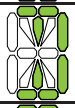
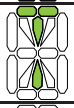
	x000	x001	x010	x011	x100	x101	x110	x111
0000								
0001								
0010								
0011								
0100								
0101								
0110								
0111								
1000								
1001								
1010								
1011								
1100								
1101								
1110								
1111								

Table 25-14. 16-segments Character Table.

	x000	x001	x010	x011	x100	x101	x110	x111
0000								
0001								
0010								
0011								
0100								
0101								
0110								
0111								
1000								
1001								
1010								
1011								
1100								
1101								
1110								
1111								

25.5.11 DATA – LCD Data Memory Mapping

The Display Memory provides access to control the “ON/OFF” state for segments.

Bit	7	6	5	4	3	2	1	0	
+0x23	PIX159	PIX158	PIX157	PIX156	PIX155	PIX154	PIX153	PIX152	DATA19
+0x22	PIX[151:144]								DATA18
+0x21	PIX[143:136]								DATA17
+0x20	PIX[135:128]								DATA16
+0x1F	PIX[127:120]								DATA15
+0x1E	PIX[119:112]								DATA14
+0x1D	PIX[111:104]								DATA13
+0x1C	PIX[103:96]								DATA12
+0x1B	PIX[95:88]								DATA11
+0x1A	PIX[87:80]								DATA10
+0x19	PIX[79:72]								DATA9
+0x18	PIX[71:64]								DATA8
+0x17	PIX[63:56]								DATA7
+0x16	PIX[55:48]								DATA6
+0x15	PIX[47:40]								DATA5
+0x14	PIX[39:32]								DATA4
+0x13	PIX[31:24]								DATA3
+0x12	PIX[23:16]								DATA2
+0x11	PIX[15:8]								DATA1
+0x10	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0	DATA0
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	

Data Memory register offset versus segment (pixel) coordinates (pixel_COM, pixel_SEG):

- LCD_offset = 0x10 + (pixel_COM x $\lfloor$ (Max_SEG + 7) / 8 $\rfloor$) + $\lfloor$ pixel_SEG / 8 $\rfloor$

  Where: ▪ 0x10 is the hexadecimal offset of DATA0 register,
  ▪ Max_SEG is the maximal number of SEG terminals of the device,
  ▪ $\lfloor$ xxx $\rfloor$ means the integer part of xxx.

Bit position of the segment (pixel) in the Data Memory register (between 0 and 7):

- bit_position = pixel_SEG % 8

  Where: ▪ % is the modulo operation.

25.6 Register Summary – LCD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x24 to	Reserved	–	–	–	–	–	–	–	–	
+0x23	DATA19	PIX[159:152]								318
+0x22	DATA18	PIX[151:144]								318
+0x21	DATA17	PIX[143:136]								318
+0x20	DATA16	PIX[135:128]								318
+0x1F	DATA15	PIX[127:120]								318
+0x1E	DATA14	PIX[119:112]								318
+0x1D	DATA13	PIX[111:104]								318
+0x1C	DATA12	PIX[103:96]								318
+0x1B	DATA11	PIX[95:88]								318
+0x1A	DATA10	PIX[87:80]								318
+0x19	DATA9	PIX[79:72]								318
+0x18	DATA8	PIX[71:64]								318
+0x17	DATA7	PIX[63:56]								318
+0x16	DATA6	PIX[55:48]								318
+0x15	DATA5	PIX[47:40]								318
+0x14	DATA4	PIX[39:32]								318
+0x13	DATA3	PIX[31:24]								318
+0x12	DATA2	PIX[23:16]								318
+0x11	DATA1	PIX[15:8]								318
+0x10	DATA0	PIX[7:0]								318
+0x0A to	Reserved	–	–	–	–	–	–	–	–	
+0x09	CTRLH	DEC	DCODE[6:0]							314
+0x08	CTRLG	TDG[1:0]		STSEG[5:0]						313
+0x07	CTRLF	–	–	FCONT[5:0]						313
+0x06	CTRLE	BPS1[3:0]				BPS0[3:0]				313
+0x05	CTRLD	–	–	–	–	BLINKEN	–	BLINKRATE[1:0]		312
+0x04	INTFLAGS	–	–	–	–	–	–	–	FCIF	312
+0x03	INTCTRL	XIME[4:0]					–	FCINTLVL[1:0]		311
+0x02	CTRLC	–	–	PMSK[5:0]						311
+0x01	CTRLB	PRESC	CLKDIV[2:0]			LPWAV	–	DUTY[1:0]		309
+0x00	CTRLA	ENABLE	XBIAS	DATLCK	COMSWP	SEGSWP	CLRDT	SEGON	BLANK	308

25.7 Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	LCD_vect	LCD Interrupt vector

26. ADC – Analog-to-Digital Converter

26.1 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
  - Up to 16x4 differential inputs without gain
  - 8x4 differential input with gain
- Built-in differential gain stage
  - 1/2x, 1x, 2x, 4x, 8x, 16x, 32x, and 64x gain options
- Single, continuous and scan conversion options
- Three internal inputs
  - Internal temperature sensor
  - AV_{CC} voltage divided by 10
  - 1.1V bandgap voltage
- Internal and external reference options
- Compare function for accurate monitoring of user defined thresholds
- Optional DMA transfer of conversion results
- Optional event triggered conversion for accurate timing
- Optional interrupt/event on compare result

26.2 Overview

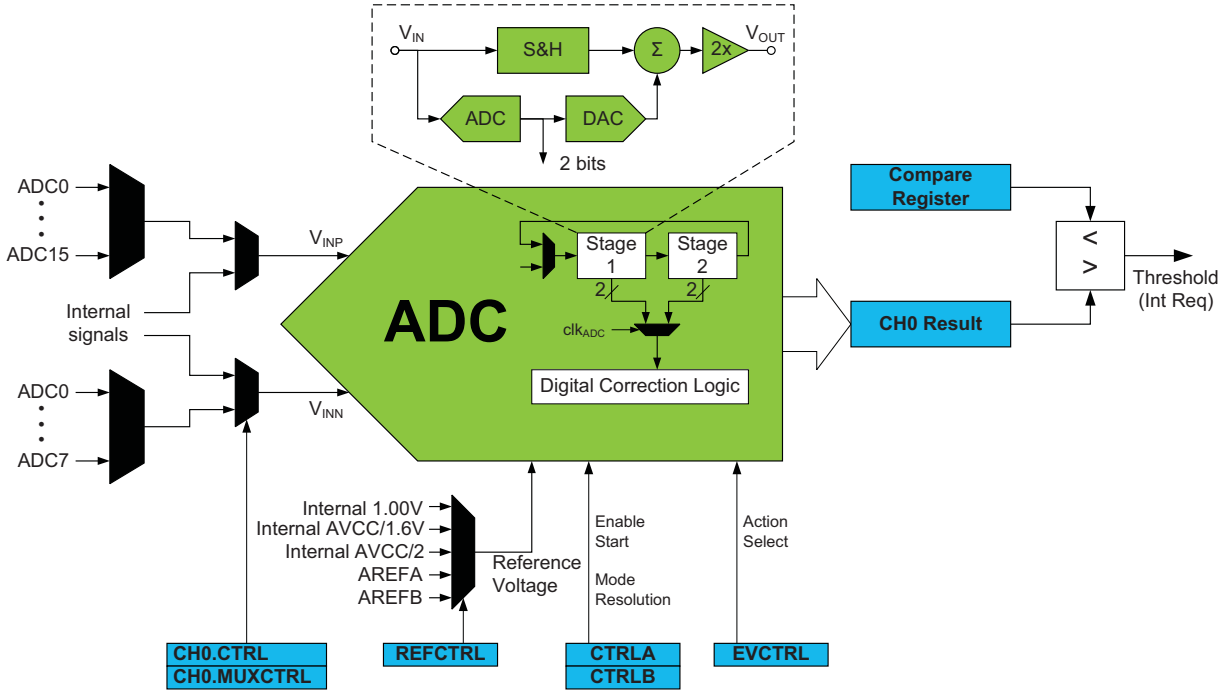
The ADC converts analog signals to digital values. The ADC has 12-bit resolution and is capable of converting up to 300 thousand samples per second (ksps). The input selection is flexible, and both single-ended and differential measurements can be done. For differential measurements, an optional gain 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 DMA 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 $AV_{CC}/10$ 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.

Figure 26-1. ADC overview.



26.3 Input Sources

Input sources are the voltage inputs that the ADC can measure and convert. Four types of measurements can be selected:

- Differential input
- Differential input with gain
- Single-ended input
- Internal input

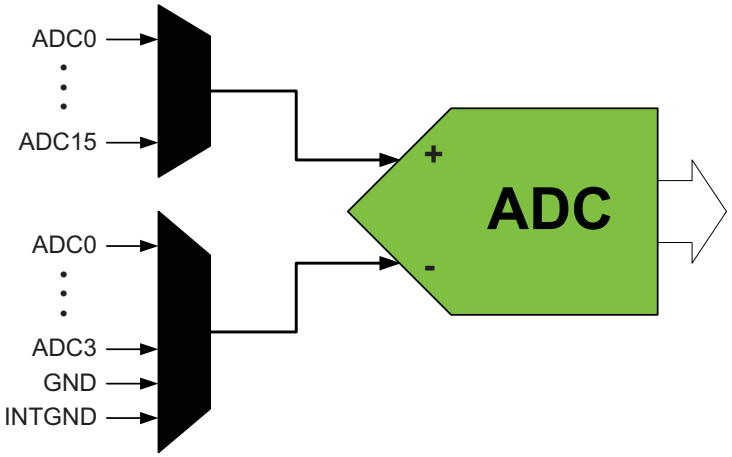
The input pins are used for single-ended and differential input, while the internal inputs are directly available inside the device. In devices with two ADCs, PORTA pins can be input to ADCA and PORTB pins can be input to ADCB. For the devices with only one ADC, input pins may be available for ADCA on both PORTA and PORTB.

The ADC is differential, and so for single-ended measurements the negative input is connected to a fixed internal value. The four types of measurements and their corresponding input options are shown in Figure 26-2 on page 322 to Figure 26-6 on page 324.

26.3.1 Differential Input

When differential input is enabled, all input pins can be selected as positive input, and input pins 0 to 3 can be selected as negative input. The ADC must be in signed mode when differential input is used.

Figure 26-2. Differential measurement without gain.

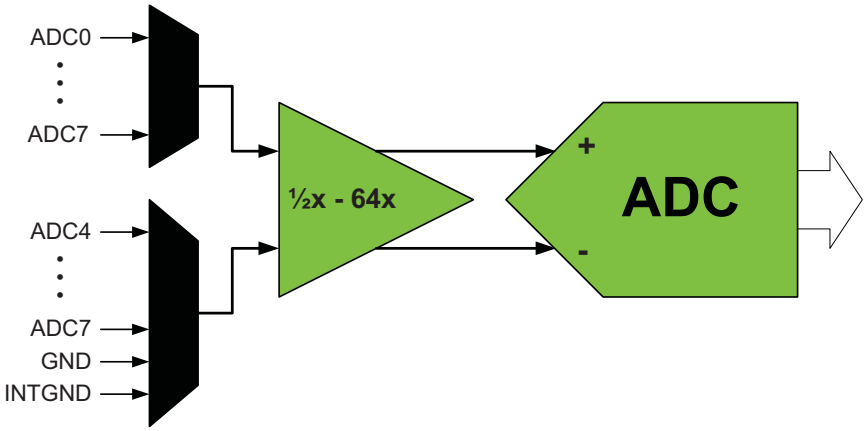


26.3.2 Differential Input with Gain

When differential input with gain is enabled, all input pins can be selected as positive input, and input pins 4 to 7 can be selected as negative input. When gain is enabled, the differential input is first sampled and amplified by the gain stage before the result is converted. The ADC must be in signed mode when differential input with gain is used.

The gain is selectable to 1/2x, 1x, 2x, 4x, 8x, 16x, 32x, and 64x gain.

Figure 26-3. Differential measurement with gain.

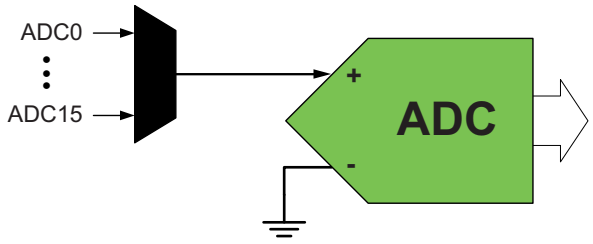


26.3.3 Single-ended Input

For single-ended measurements, all input pins can be used as inputs. Single-ended measurements can be done in both signed and unsigned mode.

The negative input is connected to internal ground in signed mode.

Figure 26-4. Single-ended measurement in signed mode.

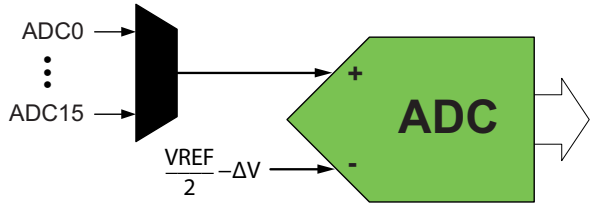


In unsigned mode, the negative input is connected to half of the voltage reference (VREF) voltage minus a fixed offset. The nominal value for the offset is:

$$\Delta V = VREF \times 0.05$$

Since the ADC is differential, the input range is VREF to zero for the positive single-ended input. The offset enables the ADC to measure zero crossing in unsigned mode, and allows for calibration of any positive offset when the internal ground in the device is higher than the external ground. See Figure 26-11 on page 326 for details.

Figure 26-5. Single-ended measurement in unsigned mode.



26.3.4 Internal Inputs

These internal signals can be measured or used by the ADC.

- Temperature sensor
- Bandgap voltage
- AV_{CC} scaled
- Pad 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.

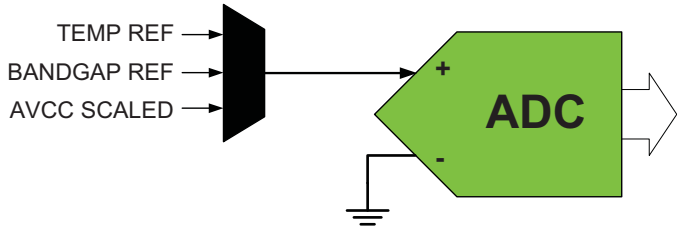
V_{CC} can be measured directly by scaling it down by a factor of 10 before the ADC input. Thus, a V_{CC} of 1.8V will be measured as 0.18V, and V_{CC} of 3.6V will be measured as 0.36V. This enables easy measurement of the V_{CC} voltage.

The internal signals need to be enabled before they can be measured. Refer to their manual sections for Bandgap 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 Gnd can be selected as negative input. Pad Gnd is the gnd level on the pin and identical or very close to the external gnd. Internal Gnd is the internal device gnd level.

Internal Gnd is used as the negative input when other internal signals are measured in single-ended signed mode.

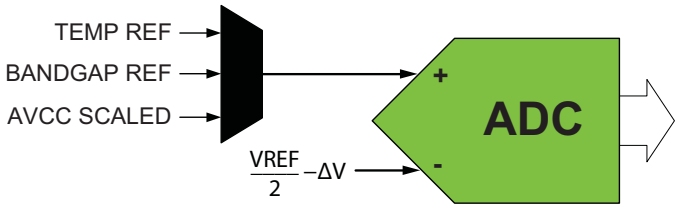
Figure 26-6. Internal measurements in single-ended signed mode.



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 (VREF) minus a fixed offset, as it is for single-ended unsigned input. Refer to Figure 26-11 on page 326 for details.

VINN = VREF/2 - ΔV

Figure 26-7. Internal measurements in unsigned mode.



26.4 Sampling Time Control

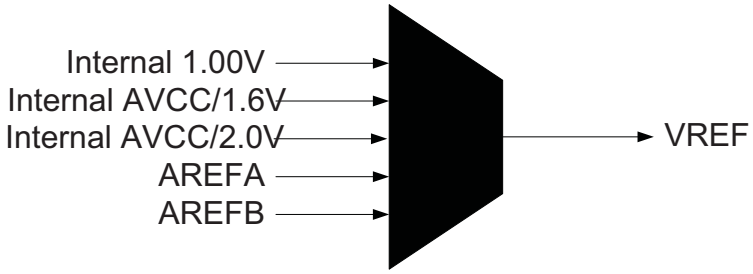
To support applications with high source output resistance, the sampling time can be increased by steps of one half ADC clock cycle up to 64 ADC clock cycles.

26.5 Voltage Reference Selection

The following voltages can be used as the reference voltage (VREF) for the ADC:

- Accurate internal 1.00V voltage generated from the bandgap
- Internal AV_{CC}/1.6V voltage
- Internal AV_{CC}/2V voltage
- External voltage applied to AREF pin on PORTA
- External voltage applied to AREF pin on PORTB

Figure 26-8. ADC voltage reference selection



26.6 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:

$$RES = \frac{VINP - VINN}{VREF} \cdot GAIN \cdot (TOP + 1)$$

VINP and VINN are the positive and negative inputs to the ADC.

For differential measurements, GAIN is 1/2 to 64. For single-ended and internal measurements, GAIN is always 1 and VINP 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:

$$RES = \frac{VINP - (-\Delta V)}{VREF} \cdot (TOP + 1)$$

VINP is the single-ended or internal input.

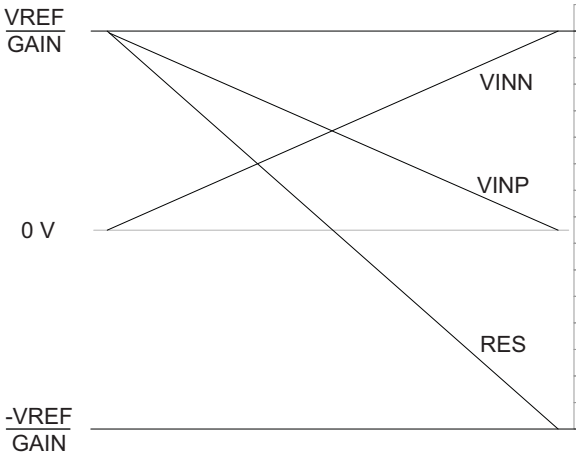
The ADC can be configured to generate either an 8-bit or a 12-bit result. A result with lower resolution will be available faster. See the “ADC Clock and Conversion Timing” on page 326 for a description on the propagation delay.

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.

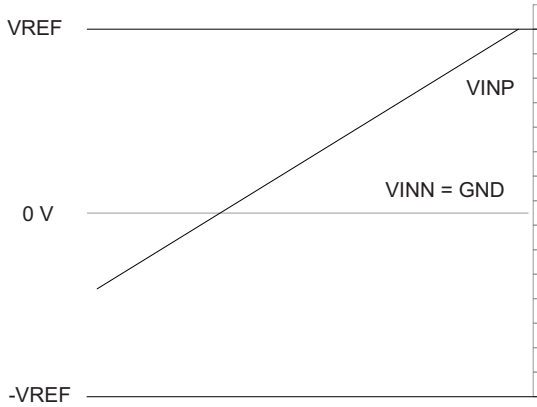
Figure 26-9 on page 325 to Figure 26-11 on page 326 show the different input options, the signal input range, and the result representation with 12-bit right adjusted mode.

Figure 26-9. Signed differential input (with gain), input range, and result representation.



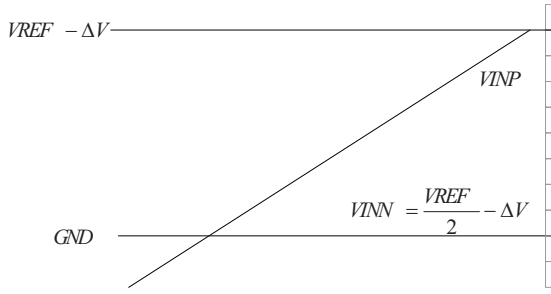
Dec	Hex	Binary	16-bit result register
2047	7FF	0111 1111 1111	0000 0111 1111 1111
2046	7FE	0111 1111 1110	0000 0111 1111 1110
2045	7FD	0111 1111 1101	0000 0111 1111 1101
...	...	...	...
3	3	0000 0000 0011	0000 0000 0000 0011
2	2	0000 0000 0010	0000 0000 0000 0010
1	1	0000 0000 0001	0000 0000 0000 0001
0	0	0000 0000 0000	0000 0000 0000 0000
-1	FFF	1111 1111 1111	1111 1111 1111 1111
-2	FFE	1111 1111 1110	1111 1111 1111 1110
...	...	...	...
-2045	803	1000 0000 0011	1111 1000 0000 0011
-2046	802	1000 0000 0010	1111 1000 0000 0010
-2047	801	1000 0000 0001	1111 1000 0000 0001
-2048	800	1000 0000 0000	1111 1000 0000 0000

Figure 26-10.Signed single-ended and internal input, input range, and result representation.



Dec	Hex	Binary	16-bit result register
2047	7FF	0111 1111 1111	0000 0111 1111 1111
2046	7FE	0111 1111 1110	0000 0111 1111 1110
2045	7FD	0111 1111 1101	0000 0111 1111 1101
...	...	...	...
3	3	0000 0000 0011	0000 0000 0000 0011
2	2	0000 0000 0010	0000 0000 0000 0010
1	1	0000 0000 0001	0000 0000 0000 0001
0	0	0000 0000 0000	0000 0000 0000 0000
-1	FFF	1111 1111 1111	1111 1111 1111 1111
-2	FFE	1111 1111 1110	1111 1111 1111 1110
...	...	...	...
-2045	803	1000 0000 0011	1111 1000 0000 0011
-2046	802	1000 0000 0010	1111 1000 0000 0010
-2047	801	1000 0000 0001	1111 1000 0000 0001
-2048	800	1000 0000 0000	1111 1000 0000 0000

Figure 26-11.Unsigned single-ended and internal input, input range, and result representation.



Dec	Hex	Binary	16-bit result register
4095	FFF	1111 1111 1111	0000 1111 1111 1111
4094	FFE	1111 1111 1110	0000 1111 1111 1110
4093	FFD	1111 1111 1101	0000 1111 1111 1101
...	...	...	...
203	0CB	0000 1100 1011	0000 0000 1100 1011
202	0CA	0000 1100 1010	0000 0000 1100 1010
201	0C9	0000 1100 1001	0000 0000 1100 1001
200	0C8	0000 1100 1000	0000 0000 1100 1000
...			
0	0	0000 0000 0000	0000 0000 0000 0000

26.7 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.

26.8 Starting a Conversion

Before a conversion is started, the input source must be selected. An ADC conversion can be started either by the application software writing to the start conversion bit or from any events in the event system.

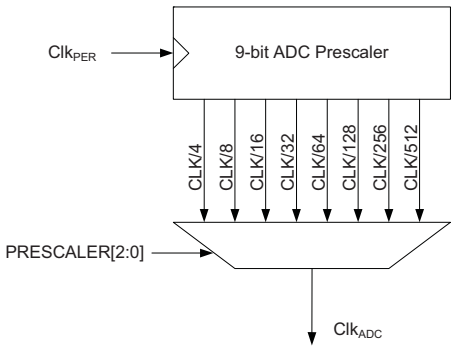
26.8.1 Input Source Scan

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.

26.9 ADC Clock and Conversion Timing

The ADC is clocked from the peripheral clock. 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 26-12.ADC prescaler.



The propagation delay of an ADC measurement is given by:

$$\text{Propagation Delay} = \frac{1 + \frac{RESOLUTION + 1}{2} + GAIN}{f_{ADC}}$$

RESOLUTION is the resolution, 8 or 12 bits. The propagation delay will increase by extra ADC clock cycles if the gain stage (GAIN) is used. A new ADC conversion can start as soon as the previous is completed.

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.

26.9.1 Single Conversion without Gain

Figure 26-13 on page 327 shows the ADC timing for a single conversion without gain. 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 26-13.ADC timing for one single conversion without gain.

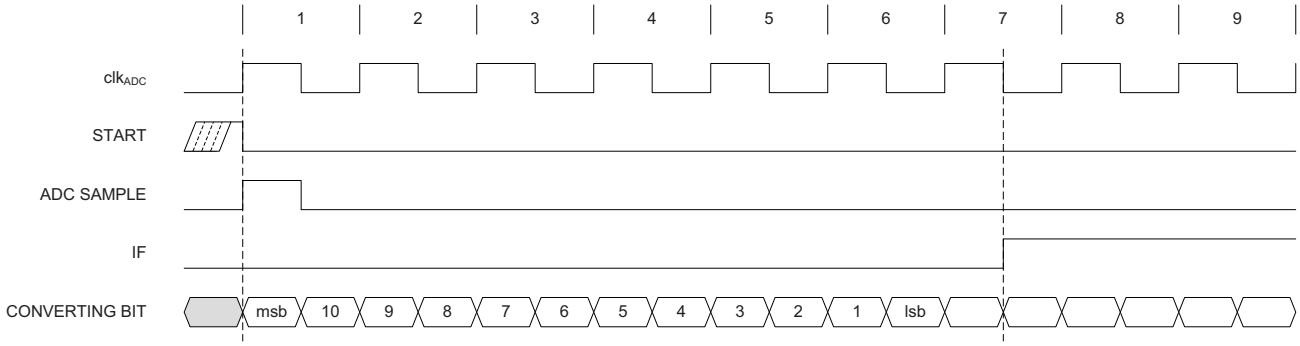
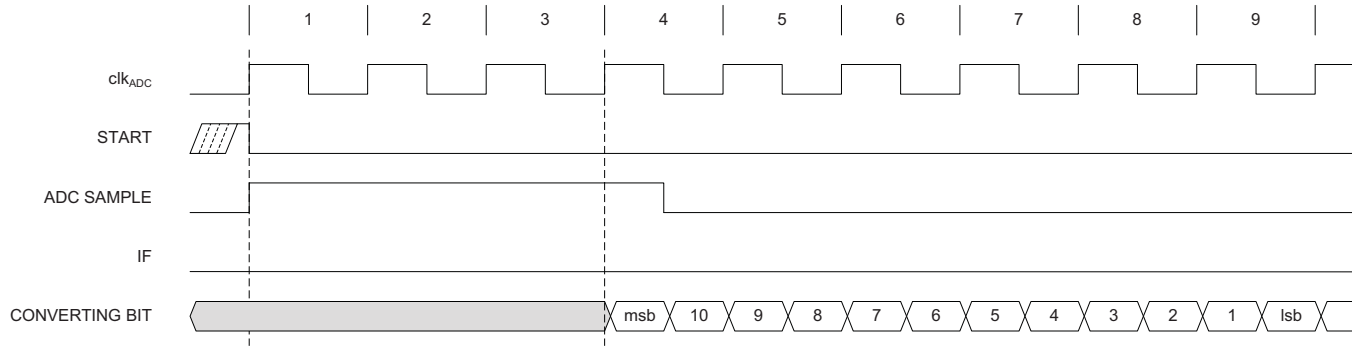


Figure 26-14.ADC timing for one single conversion with increased sampling time (SAMPVAL = 6).



26.9.2 Single Conversion with Gain

Figure 26-15 on page 328 to Figure 26-17 on page 329 show the ADC timing for one single conversion with various gain settings. As seen in the "Overview" on page 320, 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 propagation delay.

Figure 26-15.ADC timing for one single conversion with 2x gain.

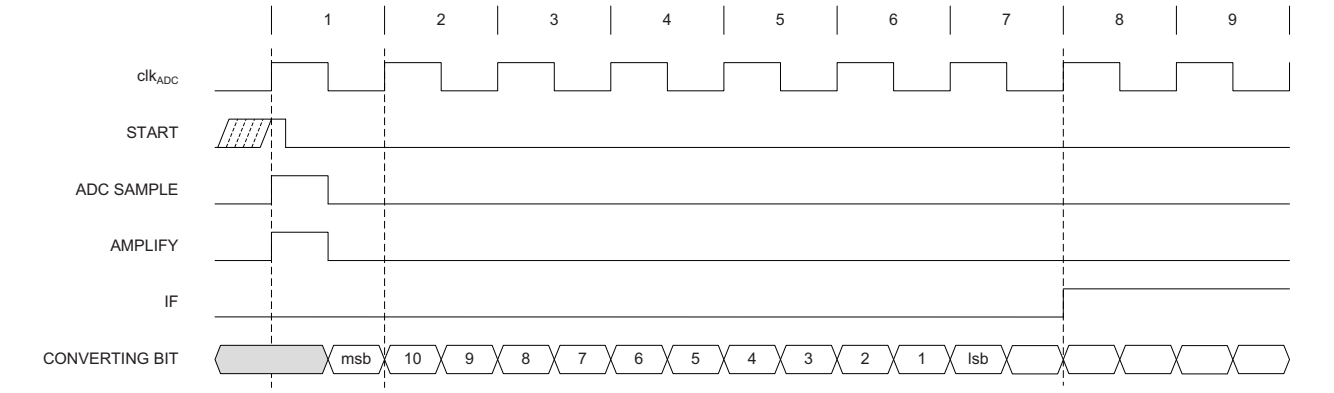


Figure 26-16.ADC timing for one single conversion with 8x gain.

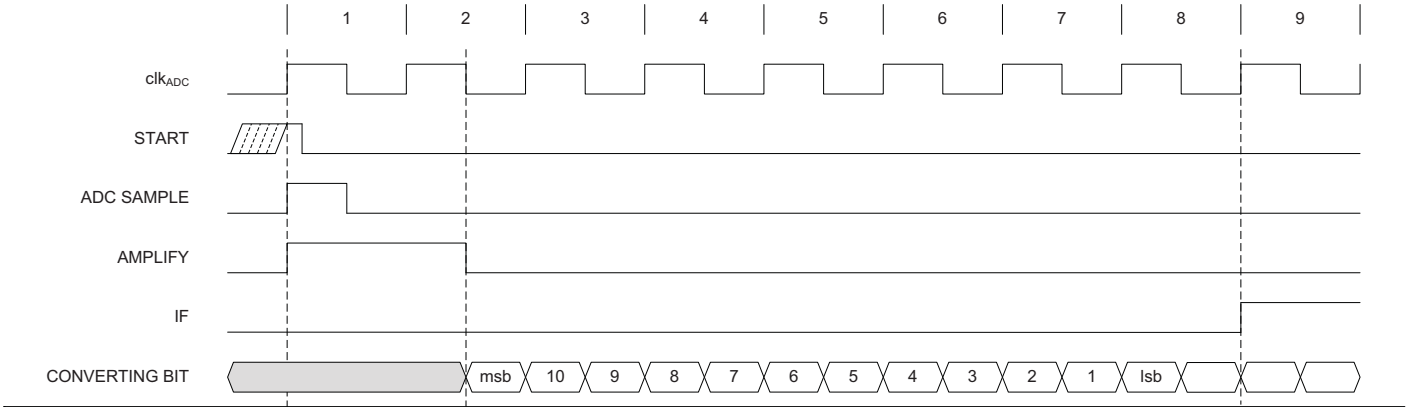
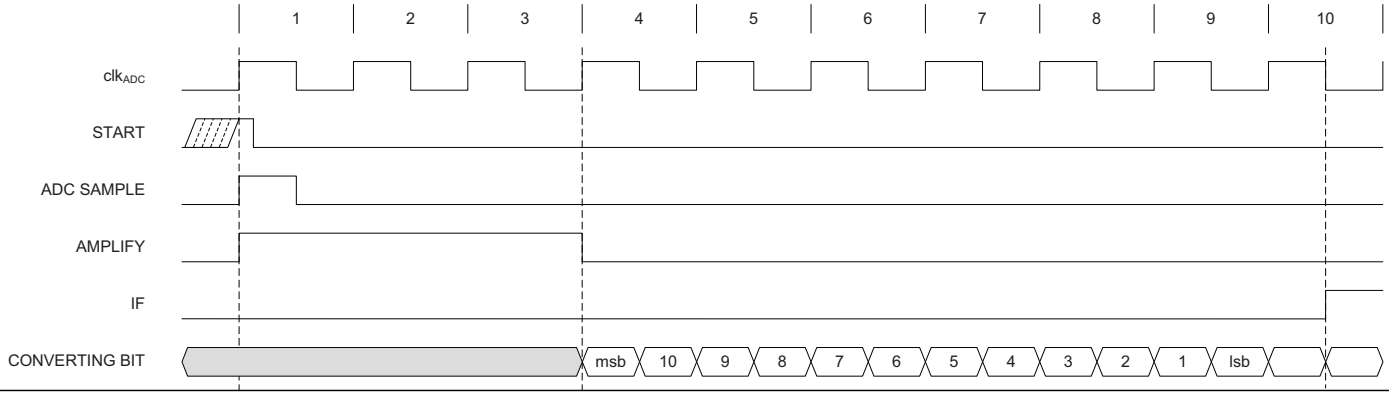


Figure 26-17.ADC timing for one single conversion with 64x gain.



26.10 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 26-18 on page 329 and Figure 26-19 on page 329 show the ADC input channel.

Figure 26-18.ADC input for single-ended measurements.

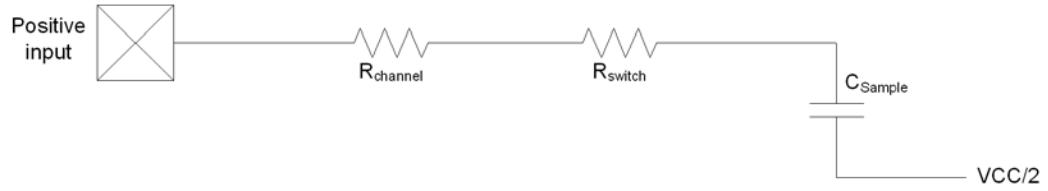
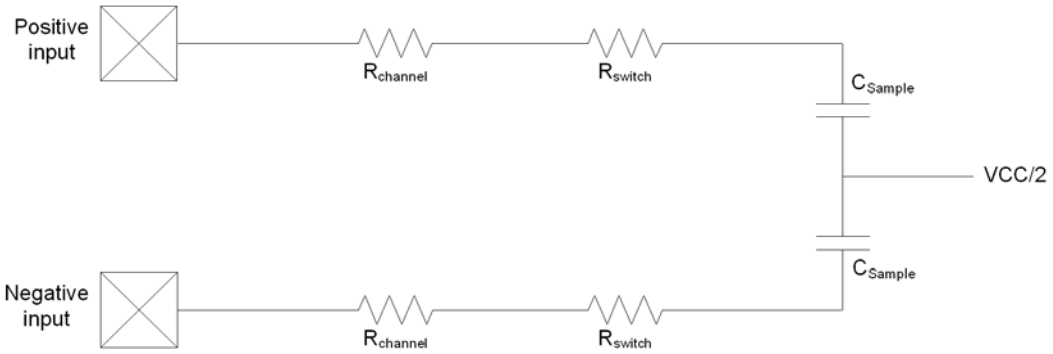


Figure 26-19.ADC input for differential measurements and differential measurements with gain.



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:

$$R_{source} \leq \frac{T_s}{C_{sample} \cdot \ln(2^{n+1})} - R_{channel} - R_{switch}$$

where the ADC sample time, T_S is one-half the ADC clock cycle given by:

$$T_s \leq \frac{1}{2 \cdot f_{ADC}}$$

For details on $R_{channel}$, R_{switch}, and C_{sample}, refer to the ADC electrical characteristic in the device datasheet.

26.11 DMA Transfer

The DMA controller can be used to transfer ADC conversion results to memory or other peripherals. A new conversion result can trigger a DMA transaction. Refer to “DMAC - Direct Memory Access Controller” on page 47 for more details on DMA transfers.

26.12 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.

26.13 Calibration

The ADC has built-in linearity calibration. The value from the production test calibration must be loaded from the signature row and into the ADC calibration register from software to achieve specified accuracy. User calibration of the linearity is not needed, hence not possible. Offset and gain calibration must be done in software.

26.14 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.

26.15 Register Description – ADC

26.15.1 CTRLA – Control register A TBD TPUBSXMEGA-116

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	–	–	CH0START	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

- **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 – CH0START: Channel Start Single Conversion**

Setting this bit will start an ADC conversion. Bit is cleared by hardware when the conversion has started. Writing this bit is equivalent to writing the START bits inside the ADC channel register.

- **Bit 1 – FLUSH: Pipeline Flush**

Setting this bit will flush the ADC. When this is done, the ADC clock is 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 any conversions were pending, these will enter the ADC and complete.

- **Bit 0 – ENABLE: Enable**

Setting this bit enables the ADC.

26.15.2 CTRLB – ADC 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.

- **Bit 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 26-1. The indicated current limitations are nominal values. Refer to the device datasheet for actual current limitation for each setting.

Table 26-1. 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 Running Mode**

This bit controls the free running mode for the ADC. Once a conversion is finished, the next input will be sampled and converted.

- **Bit 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 26-2 for possible settings.

Table 26-2. ADC conversion result resolution.

RESOLUTION[1:0]	Group Configuration	Description
00	12BIT	12-bit result, right adjusted
01	–	Reserved
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.

26.15.3 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 selects the reference for the ADC according to Table 26-3.

Table 26-3. ADC reference selection.

REFSEL[2:0]	Group Configuration	Description
000	INT1V	10/11 of bandgap (1.0V)
001	INTVCC	V_{CC}/1.6
010	AREFA	External reference from AREF pin on PORT A
011	AREFB	External reference from AREF pin on PORT B
100	INTVCC2	V_{CC}/2
101 - 111	–	Reserved

- **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 – 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.

26.15.4 EVCTRL – Event Control register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	EVSEL[1:0]		EVACT[2:0]		
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 4:3 – EVSEL[1:0]: Event Channel Input Select**

These bits select which event channel will trigger the ADC channel. Each setting defines a group of event channels, where the event channel with the lowest number will trigger ADC channel 0, the next event channel will trigger ADC channel 1, and so on. See Table 26-4.

Table 26-4. ADC event channel select.

EVSEL[1:0]	Group Configuration	Selected Event Lines
00	0	Event channel 0 selected inputs
01	1	Event channel 1 selected inputs
10	2	Event channel 2 selected inputs
11	3	Event channel 3 selected inputs

- **Bit 2:0 – EVACT[2:0]: Event Mode**

These bits select and limit how many of the selected event input channel are used, and also further limit the ADC channels triggers. They also define more special event triggers as defined in Table 26-5.

Table 26-5. 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

EVACT[2:0]	Group Configuration	Event Input Operation Mode
101	–	Reserved
110	SYNCSWEEP	The ADC is flushed and restarted for accurate timing
111	–	Reserved

26.15.5 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	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 – PRESCALER[2:0]: Prescaler Configuration**

These bits define the ADC clock relative to the peripheral clock according to Table 26-6.

Table 26-6. 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

26.15.6 INTFLAGS – Interrupt Flag 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

- **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 – CH0IF: Interrupt Flags**

This flag is set when the ADC conversion is complete. If the ADC is configured for compare mode, the interrupt 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.

26.15.7 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

- **Bit 7:0 – TEMP[7:0]: 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.

26.15.8 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

- **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 – SAMPVAL[5:0]: sampling time control register**

These bits control the ADC sampling time in number of half ADC prescaled clock cycles (depends of ADC_PRESCALER value), thus controlling the ADC input impedance. Sampling time is set according to the formula:

Sampling time = (SAMPVAL + 1)(Clk_{ADC} /2)*

26.15.9 CALL – Calibration Value register Low

The CALL and CALH register pair hold the 12-bit calibration value. 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.

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

- **Bit 7:0 – CAL[7:0]: ADC Calibration value**

These are the eight lsbs of the 12-bit CAL value.

26.15.10CALH – Calibration Value 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

- **Bit 3:0 – CAL[11:8]: Calibration value**

These are the four msbs of the 12-bit CAL value.

26.15.11CH0RESH – Channel 0 Result register High

The CH0RESL and CH0RESH register pair represents the 16-bit value, CH0RES. For details on reading 16-bit registers, refer to “Accessing 16-bit Registers” on page 13.

	Bit	7	6	5	4	3	2	1	0
12-bit, left		CHRES[11:4]							
12-bit, right		–	–	–	–	CHRES[11:8]			
8-bit		–	–	–	–	–	–	–	–
	Read/Write	R	R	R	R	R	R	R	R
	Initial Value	0	0	0	0	0	0	0	0

26.15.11.1 12-bit Mode, Left Adjusted

- **Bit 7:0 – CHRES[11:4]: Channel Result High byte**

These are the eight msbs of the 12-bit ADC result.

26.15.11.2 12-bit Mode, Right Adjusted

- **Bit 7:4 – Reserved**

These bits will in practice be the extension of the sign bit, CHRES11, when the ADC works in differential mode, and set to zero when the ADC works in signed mode.

- **Bit 3:0 – CHRES[11:8]: Channel Result High byte**

These are the four msbs of the 12-bit ADC result.

26.15.11.3 8-bit Mode

- **Bit 7:0 – Reserved**

These bits will in practice be the extension of the sign bit, CHRES7, when the ADC works in signed mode, and set to zero when the ADC works in single-ended mode.

26.15.12 CH0RESL – Channel 0 Result register Low

	Bit	7	6	5	4	3	2	1	0
12-/8-bit, right		CHRES[7:0]							
12-bit, left		CHRES[3:0]				–	–	–	–
	Read/Write	R	R	R	R	R	R	R	R
	Initial Value	0	0	0	0	0	0	0	0

26.15.12.1 12-/8-bit Mode

- **Bit 7:0 – CHRES[7:0]: Channel Result Low byte**

These are the eight lsbs of the ADC result.

26.15.12.2 12-bit Mode, Left Adjusted

- **Bit 7:4 – CHRES[3:0]: Channel Result Low byte**

These are the four lsbs of the 12-bit ADC result.

- **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.

26.15.13 CMPH – Compare register High

The CMPH and CMPL register pair represents the 16-bit value, CMP. For 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
+0x19	CMP[15:0]							
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 – CMP[15:0]: Compare Value High byte**

These are the eight msbs of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement, and the msb is the sign bit.

26.15.14 CMPL – Compare register Low

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

- **Bit 7:0 – CMP[7:0]: Compare Value Low byte**

These are the eight lsbs of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement.

26.16 Register Description - ADC Channel

26.16.1 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 on Channel**

Setting this bit will start a conversion on the channel. The bit is cleared by hardware when the conversion has started. Setting this bit when it already is set will have no effect. Writing or reading this bit is equivalent to writing the CH[3:0]START bits in “CTRLA – Control register A TBD TPUBSXMEGA-116” on page 331.

- **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 4:2 – GAIN[2:0]: Gain Factor**

These bits define the gain factor for the ADC gain stage.

See Table 26-7 on page 338. Gain is valid only with certain MUX settings. See “MUXCTRL – MUX Control registers” on page 338.

Table 26-7. ADC gain factor

GAIN[2:0]	Group Configuration	Gain Factor
000	1X	1x
001	2X	2x
010	4X	4x
011	8X	8x
100	16X	16x
101	32X	32x
110	64X	64x
111	DIV2	½x

- **Bit 1:0 – INPUTMODE[1:0]: Channel Input Mode**

These bits define the channel mode.

Table 26-8. 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 26-9. Channel input modes, CONVMODE=1 (singed mode).

INPUTMODE[1:0]	Group Configuration	Description
00	INTERNAL	Internal positive input signal
01	SINGLEENDED	Single-ended positive input signal
10	DIFF	Differential input signal
11	DIFFWGAIN	Differential input signal with gain

26.16.2 MUXCTRL – MUX Control registers

The MUXCTRL register defines the input source for the channel.

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	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 6:3 – MUXPOS[3:0]: MUX Selection on Positive ADC Input**

These bits define the MUX selection for the positive ADC input. Table 26-10 and Table 26-11 show the possible input selection for the different input modes.

Table 26-10. ADC MUXPOS configuration when INPUTMODE[1:0] = 00 (internal) is used.

MUXPOS[3:0]	Group Configuration	Description
0000	TEMP	Temperature reference
0001	BANDGAP	Bandgap voltage
0010	SCALEDVCC	1/10 scaled V_{CC}
0011	–	Reserved
0100-1111	–	Reserved

Table 26-11. ADC MUXPOS configuration when INPUTMODE[1:0] = 01 (single-ended) or INPUTMODE[1:0] = 10 (differential) is used.

MUXPOS[3:0]	Group Configuration	Description
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

Table 26-12. ADC MUXPOS configuration when INPUTMODE[1:0] = 11 (differential with gain) is used.

MUXPOS[3:0]	Group Configuration	Description
0000	PIN0	ADC0 pin
0001	PIN1	ADC1 pin
0010	PIN2	ADC2 pin

MUXPOS[3:0]	Group Configuration	Description
0011	PIN3	ADC3 pin
0100	PIN4	ADC4 pin
0101	PIN5	ADC5 pin
0110	PIN6	ADC6 pin
0111	PIN7	ADC7 pin
1XXX	–	Reserved

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.

- **Bit 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 used.

Table 26-13 and Table 26-14 show the possible input sections.

Table 26-13. ADC MUXNEG configuration, INPUTMODE[1:0] = 10, differential without gain.

MUXNEG[2:0]	Group Configuration	Analog Input
000	PIN0	ADC0 pin
001	PIN1	ADC1 pin
010	PIN2	ADC2 pin
011	PIN3	ADC3 pin
100	–	Reserved
101	GND	PAD ground
110	–	Reserved
111	INTGND	Internal ground

Table 26-14. ADC MUXNEG configuration, INPUTMODE[1:0] = 11, differential with gain.

MUXNEG[2:0]	Group Configuration	Analog Input
000	PIN4	ADC4 pin
001	PIN5	ADC5 pin
010	PIN6	ADC6 pin
011	PIN7	ADC7 pin
100	INTGND	Internal ground
101	–	Reserved
110	–	Reserved
111	GND	PAD ground

26.16.3 INTCTRL – Interrupt Control registers

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 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 – INTMODE: Interrupt Mode**

These bits select the interrupt mode for the channel according to Table 26-5.

Table 26-15. ADC interrupt mode.

INTMODE[1:0]	Group Configuration	Interrupt Mode
00	COMPLETE	Conversion complete
01	BELOW	Compare result below threshold
10	–	Reserved
11	ABOVE	Compare result 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will be triggered for conditions when the IF bit in the INTFLAGS register is set.

26.16.4 INTFLAGS – Interrupt Flag registers

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

- **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 – 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.

26.16.5 RESH – Result register High

For all result registers and with any ADC result resolution, a signed number is represented in 2's complement form, and the msb represents the sign bit.

The RESL and RESH register pair represents the 16-bit value, ADCRESULT. Reading and writing 16-bit values require special attention. Refer to "Accessing 16-bit Registers" on page 13 for details.

	Bit	7	6	5	4	3	2	1	0
12-bit, left.		RES[11:4]							
12-bit, right	+0x05	–	–	–	–	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

26.16.5.1 12-bit Mode, Left Adjusted

- **Bit 7:0 – RES[11:4]: Channel Result High byte**

These are the eight msbs of the 12-bit ADC result.

26.16.5.2 12-bit Mode, Right Adjusted

- **Bit 7:4 – Reserved**

These bits will in practice be the extension of the sign bit, CHRES11, when the ADC works in differential mode, and set to zero when the ADC works in signed mode.

- **Bits 3:0 – RES[11:8]: Channel Result High bits**

These are the four msbs of the 12-bit ADC result.

26.16.5.3 8-bit Mode

- **Bit 7:0 – Reserved**

These bits will in practice be the extension of the sign bit, CHRES7, when the ADC works in signed mode, and set to zero when the ADC works in single-ended mode.

26.16.6 RESL – Result register Low

	Bit	7	6	5	4	3	2	1	0
12-/8-bit, right	+0x04	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

26.16.6.1 12-/8-bit Mode

- **Bit 7:0 – RES[7:0]: Result Low byte**

These are the eight lsbs of the ADC result.

26.16.6.2 12-bit Mode, Left Adjusted

- **Bit 7:4 – RES[3:0]: Result Low bits**

These are the four lsbs of the 12-bit ADC result.

- **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.

26.16.7 SCAN – Input Channel Scan register

Scan is enabled when COUNT is set differently than 0.

Bit	7	6	5	4	3	2	1	0
+0x06	OFFSET[3:0]				COUNT[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 – OFFSET[3:0]: Positive MUX Setting Offset**

The channel scan is enabled when COUNT != 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 + OFFSET. The value is incremented after each conversion until it reaches the maximum value given by COUNT. When OFFSET is equal to COUNT, OFFSET will be cleared on the next conversion.

- **Bit 3:0 – COUNT[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 COUNT + 1. The input channels included are the range from MUXPOS to MUXPOS + COUNT.

26.17 Register Summary – ADC

This is the register summary when the ADC is configured to give standard 12-bit results. The register summaries for 8-bit and 12-bit left adjusted will be similar, but with some changes in the result registers, CH0RESH and CH0RESL.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	–	CH0STAR	FLUSH	ENABLE	331
+0x01	CTRLB	–	CURRLIMIT[1:0]		CONVMO	FREERUN	RESOLUTION[1:0]		–	331
+0x02	REFCTRL	–	REFSEL[2:0]			–	–	BANDGAP	TEMPREF	332
+0x03	EVCTRL	–	–	–	EVSEL[1:0]		EVACT[2:0]			333
+0x04	PRESCALER	–	–	–	–	–	PRESCALER[2:0]			334
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	INTFLAGS	–	–	–	–	–	–	–	CH0IF	334
+0x07	TEMP	TEMP[7:0]								335
+0x08	SAMPCTRL	–	–	SAMPVAL[5:0]						335
+0x09	Reserved	–	–	–	–	–	–	–	–	
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	CALL	CAL[7:0]								335
+0x0D	CALH	–	–	–	–	CAL[11:8]				
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10	CH0RESL	CH0RES[7:0]								336
+0x11	CH0RESH	CH0RES[15:8]								336
+0x12	Reserved	–	–	–	–	–	–	–	–	
+0x13	Reserved	–	–	–	–	–	–	–	–	
+0x14	Reserved	–	–	–	–	–	–	–	–	
+0x15	Reserved	–	–	–	–	–	–	–	–	
+0x16	Reserved	–	–	–	–	–	–	–	–	
+0x17	Reserved	–	–	–	–	–	–	–	–	
+0x18	CMPL	CMP[7:0]								337
+0x19	CMPH	CMP[15:8]								337
+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	–	–	–	–	–	–	–	–	

26.18 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]		337
+0x01	MUXCTRL	–	MUXPOS[3:0]				MUXNEG[2:0]			338
+0x02	INTCTRL	–	–	–	–	INTMODE[1:0]		INTLVL[1:0]		341
+0x03	INTFLAGS	–	–	–	–	–	–	–	IF	341
+0x04	RESL	RES[7:0]								342
+0x05	RESH	RES[15:8]								342
+0x06	SCAN	OFFSET				COUNT				342
+0x07	Reserved	–	–	–	–	–	–	–	–	

26.19 Interrupt vector Summary

Offset	Source	Interrupt Description
0x00	CH0	Analog-to-digital converter channel 0 interrupt vector

27. AC – Analog Comparator

27.1 Features

- Selectable hysteresis
  - None
  - Small
  - Large
- Analog comparator output available on pin
- Flexible input selection
  - All pins on the port
  - Bandgap reference voltage
  - A 64-level programmable voltage scaler of the internal AV_{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

27.2 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 events upon several different combinations of input change.

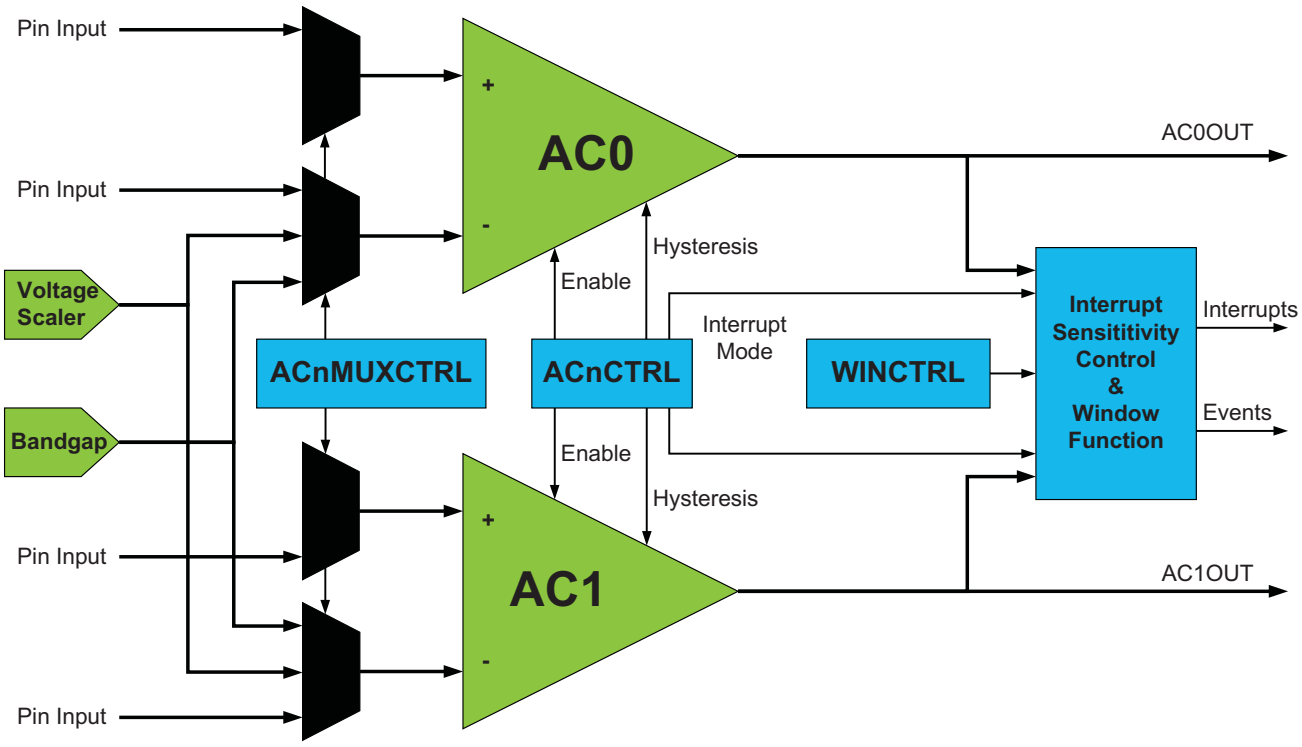
The analog comparator hysteresis can be adjusted in order to achieve the optimal operation for each application.

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 27-1. Analog comparator overview.



27.3 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 AV_{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.

27.3.1 Pin Inputs

Any of analog input pins on the port can be selected as input to the analog comparator.

27.3.2 Internal Inputs

Two internal inputs are available for the analog comparator:

- Bandgap reference voltage
- Voltage scaler, which provides a 64-level scaling of the internal AV_{CC} voltage

27.4 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.

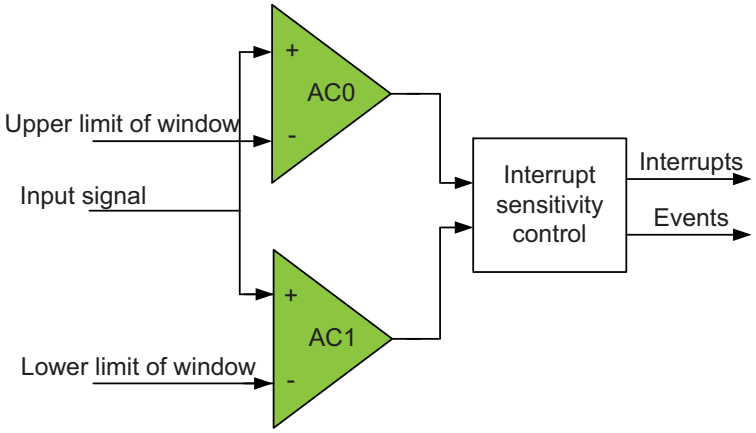
27.5 Interrupts and Events

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). Events are generated at all times for the same condition as the interrupt, regardless of whether the interrupt is enabled or not.

27.6 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 27-2. The Analog comparators in window mode.



27.7 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.

27.8 Register Description

27.8.1 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 27-1.

Table 27-1. 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. 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 27-2. For details on actual hysteresis levels, refer to the device datasheet.

Table 27-2. 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.

27.8.2 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 27-3.

Table 27-3. 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	–	Reserved

- **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 27-4.

Table 27-4. 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	–	Reserved
110	BANDGAP	Internal bandgap voltage
111	SCALER	AV_{CC} voltage scaler

27.8.3 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	–	AC1OUT	AC0OUT
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 – AC1OUT: Analog Comparator 1 Output**

Setting this bit makes the output of AC1 available on pin 6 of the port.

- **Bit 0 – AC0OUT: Analog Comparator 0 Output**

Setting this bit makes the output of AC0 available on pin 7 of the port.

27.8.4 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	SCALEFAC[5:0]					
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 – 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 AVcc voltage scaler. The input to the analog comparator, V_{SCALE}, is:

$$V_{SCALE} = \frac{V_{CC} \cdot (SCALEFAC + 1)}{64}$$

27.8.5 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 27-5.

Table 27-5. 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115. The enabled interrupt will trigger according to the WINTMODE setting.

27.8.6 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 27-6.

Table 27-6. Window mode current state.

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 fromAC0.

- **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 350.

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 348.

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 348.

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.

27.8.7 CURRCTRL – Current Source Control register

Bit	7	6	5	4	3	2	1	0
+0x08	CURRENT	–	–	–	–	–	AC1CURR	AC0CURR
Read/Write	R/W	R	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 6:2 – 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.

27.8.8 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.

27.9 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	348
+0x01	AC1CTRL	INTMODE[1:0]		INTLVL[1:0]		–	HYSMODE[1:0]		ENABLE	348
+0x02	AC0MUXCTR	–	–	MUXPOS[2:0]			MUXNEG[2:0]			349
+0x03	AC1MUXCTR	–	–	MUXPOS[2:0]			MUXNEG[2:0]			349
+0x04	CTRLA	–	–	–	–	–	–	AC1OUT	ACOOUT	350
+0x05	CTRLB	–	–	SCALEFAC5:0]						350
+0x06	WINCTRL	–	–	–	WEN	WINTMODE[1:0]		WINTLVL[1:0]		350
+0x07	STATUS	WSTATE[1:0]		AC1STATE	AC0STATE	–	WIF	AC1IF	AC0IF	351
+0x08	CURRCTRL	CURRENT	–	–	–	–	–	AC1CURR	AC0CURR	352
+0x09	CURRCALIB	–	–	–	–	CALIB[3:0]				352

27.10 Interrupt vector Summary

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

28. IEEE 1149.1 JTAG Boundary Scan Interface

28.1 Features

- **JTAG (IEEE Std. 1149.1-2001 compliant) interface**
- **Boundary scan capabilities according to the JTAG standard**
- **Full scan of all I/O pins**
- **Supports the mandatory SAMPLE, IDCODE, PRELOAD, EXTEST, and BYPASS instructions**
- **Supports the optional HIGHZ and CLAMP instructions**
- **Supports the AVR-specific PDICOM instruction for accessing the PDI**

28.2 Overview

The JTAG interface is mainly intended for testing PCBs by using the JTAG boundary scan capability. Secondarily, the JTAG interface is used to access the Program and Debug Interface (PDI) in its optional JTAG mode.

The boundary scan chain has the capability of driving and observing the logic levels on I/O pins. At the system level, all microcontroller or board components having JTAG capabilities are connected serially by the TDI/TDO signals to form a long shift register. An external controller sets up the devices to drive values at their output pins, and observes the input values received from other devices. The controller compares the received data with the expected result. In this way, boundary scan method provides a mechanism for testing the interconnections and integrity of components on printed circuit boards by using only the four test access port (TAP) signals.

The IEEE Std. 1149.1-2001 defined mandatory JTAG instructions, IDCODE, BYPASS, SAMPLE/ PRELOAD, and EXTEST, together with the optional CLAMP and HIGHZ instructions can be used for testing the printed circuit board. Alternatively, the HIGHZ instruction can be used to place all I/O pins in an inactive drive state, while bypassing the boundary scan register chain of the chip.

The AVR-specific PDICOM instruction makes it possible to use the PDI data register as an interface for accessing the PDI for programming and debugging. This provides an alternative way to access internal programming and debugging resources by using the JTAG interface. For more details on PDI, programming, and on-chip debugging, refer to “Program and Debug Interface” on page 393.

The JTAGEN fuse must be programmed and the JTAGD bit in the MCUCR register must be cleared to enable the JTAG interface and TAP. See “FUSEBYTE4 – Fuse Byte4” on page 31, and “MCUCR – Control register” on page 45 for more details.

When using the JTAG interface for boundary scan, the JTAG TCK clock frequency can be higher than the internal device frequency. A system clock in the device is not required for boundary scan.

28.3 TAP - Test Access Port

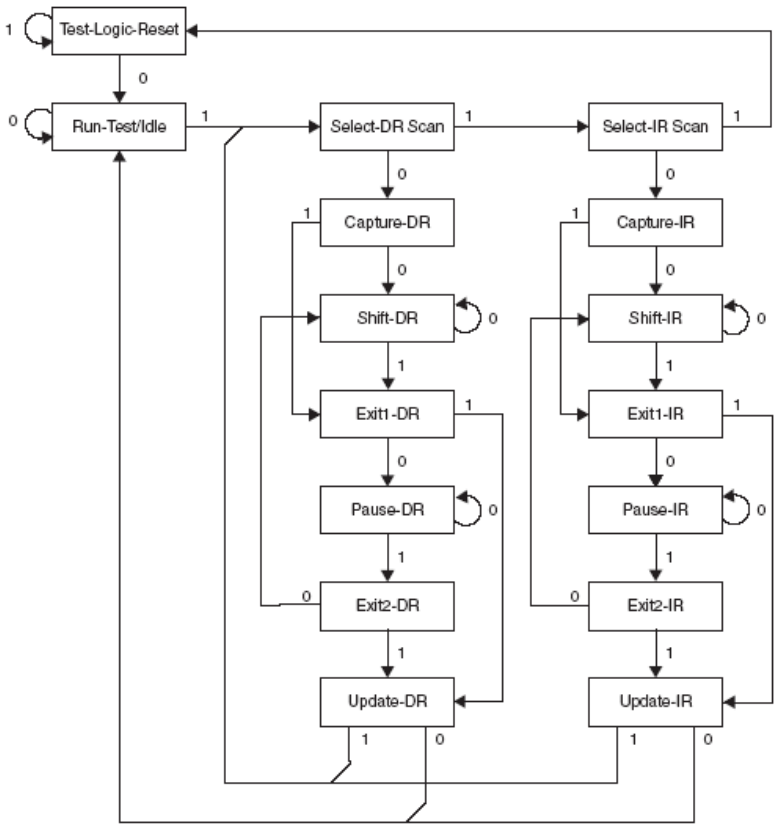
The JTAG interface requires and uses four device I/O pins. In JTAG terminology, these pins constitute the test access port, or TAP. These pins are:

- TMS: Test mode select. The pin is used for navigating through the TAP-controller state machine
- TCK: Test clock. This is the JTAG clock signal, and all operation is synchronous to TCK
- TDI: Test data in. Serial input data to be shifted in to the instruction register or data register (scan chains)
- TDO: Test data out. Serial output data from the instruction register or data register

The IEEE Std. 1149.1-2001 also specifies an optional test reset signal, TRST. This signal is not available.

When the JTAGEN fuse is unprogrammed or the JTAG disable bit is set, the JTAG interface is disabled. The four TAP pins are normal port pins, and the TAP controller is in reset. When enabled, the input TAP signals are internally pulled high and JTAG is enabled for boundary scan operations.

Figure 28-1. TAP controller state diagram.



The TAP controller is a 16-state, finite state machine that controls the operation of the boundary scan circuitry. The state transitions shown in Figure 28-1 depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge on TCK. The initial state after a power-on reset is the test logic reset state.

Assuming the present state is run test/idle, a typical scenario for using the JTAG interface is:

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the shift instruction register, or shift IR, state. While in this state, shift the four bits of the JTAG instruction into the JTAG instruction register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 lsbs in order to remain in the shift IR state. The msb of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR state, 0x01, is shifted out on the TDO pin. The JTAG instruction selects a particular data register as the path between TDI and TDO and controls the circuitry surrounding the selected data register
- Apply the TMS sequence 1, 1, 0 to reenter the run test/idle state. The instruction is latched onto the parallel output from the shift register path in the update IR state. The exit IR, pause IR, and exit2 IR states are used only for navigating the state machine
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the shift data register, or shift DR, state. While in this state, upload the selected data register (selected by the present JTAG instruction in the JTAG instruction register) from the TDI input at the rising edge of TCK. In order to remain in the shift DR state, the TMS input must be held low during the input of all bits except the msb. The msb of the data is shifted in when this state is left by setting TMS high. While the data register is shifted in from the TDI pin, the parallel inputs to the data register captured in the capture DR state are shifted out on the TDO pin
- Apply the TMS sequence 1, 1, 0 to reenter the run test/idle state. If the selected data register has a latched parallel output, the latching takes place in the update DR state. The exit DR, pause DR, and exit2 DR states are used only for navigating the state machine.

As shown in the state diagram, the run test/idle state need not be entered between selecting JTAG instructions and using data registers.

Note: Independently of the initial state of the TAP controller, the test logic reset state can always be entered by holding TMS high for five TCK clock periods.

28.4 JTAG Instructions

The instruction register is four bits wide. Listed below are the JTAG instructions for boundary scan operation and the PDICOM instruction used for accessing the PDI in JTAG mode.

The lsb is shifted in and out first for all shift registers.

The opcode for each instruction is shown beside the instruction name in hex format. The text describes which data register is selected as the path between TDI and TDO for each instruction.

28.4.1 EXTEST; 0x1

EXTEST is the instruction for selecting the boundary scan chain as the data register for testing circuitry external to the AVR XMEGA device package. The instruction is used for sampling external pins and loading output pins with data. For the I/O port pins, both output control (DIR) and output data (OUT) are controllable via the scan chain, while the output control and actual pin value are observable. The contents of the latched outputs of the boundary scan chain are driven out as soon as the JTAG instruction register is loaded with the EXTEST instruction.

The active states are:

- Capture DR: Data on the external pins are sampled into the boundary scan chain
- Shift DR: Data in the Boundary-scan Chain are shifted by the TCK input
- Update DR: Data from the scan chain are applied to output pins

28.4.2 IDCODE; 0x3

IDCODE is the instruction for selecting the 32-bit ID register as the data register. The ID register consists of a version number, a device number, and the manufacturer code chosen by the Joint Electron Devices Engineering Council (JEDEC). This is the default instruction after power up.

The active states are:

- Capture DR: Data in the IDCODE register are sampled into the device identification register
- Shift DR: The IDCODE scan chain is shifted by the TCK input

28.4.3 SAMPLE/PRELOAD; 0x2

SAMPLE/PRELOAD is the instruction for preloading the output latches and taking a snapshot of the input/output pins without affecting system operation. However, the output latches are not connected to the pins. The boundary scan chain is selected as the data register. Since each of the SAMPLE and PRELOAD instructions implements the functionality of the other, they share a common binary value, and can be treated as a single, merged instruction.

The active states are:

- Capture DR: Data on the external pins are sampled into the boundary scan chain
- Shift DR: The boundary scan chain is shifted by the TCK input
- Update DR: Data from the boundary scan chain are applied to the output latches, but the output latches are not connected to the pins

28.4.4 BYPASS; 0xf

BYPASS is the instruction for selecting the bypass register for the data register. This instruction can be issued to make the shortest possible scan chain through the device.

The active states are:

- Capture DR: Loads a zero into the bypass register
- Shift DR: The bypass register cell between TDI and TDO is shifted

28.4.5 CLAMP; 0x4

CLAMP is an optional instruction that allows the state of the input/output pins to be determined from the preloaded output latches. The instruction allows static pin values to be applied via the boundary scan registers while bypassing these registers in the scan path, efficiently shortening the total length of the serial test path. The bypass register is selected as the data register.

The active states are:

- Capture DR: Loads a zero into the bypass register
- Shift DR: The bypass register cell between TDI and TDO is shifted

28.4.6 HIGHZ; 0x5

HIGHZ is an optional instruction for putting all outputs in an inactive drive state (e.g., high impedance). The bypass register is selected as the data register.

The active states are:

- Capture DR: Loads a zero into the bypass register
- Shift DR: The bypass register cell between TDI and TDO is shifted

28.4.7 PDICOM; 0x7

PDICOM is an AVR XMEGA specific instruction for using the JTAG TAP as an alternative interface to the PDI.

The active states are:

- Capture DR: Parallel data from the PDI are sampled into the PDICOM data register
- Shift DR: The PDICOM data register is shifted by the TCK input
- Update DR: Commands or operands are parallel-latched from the PDICOM data register into the PDI

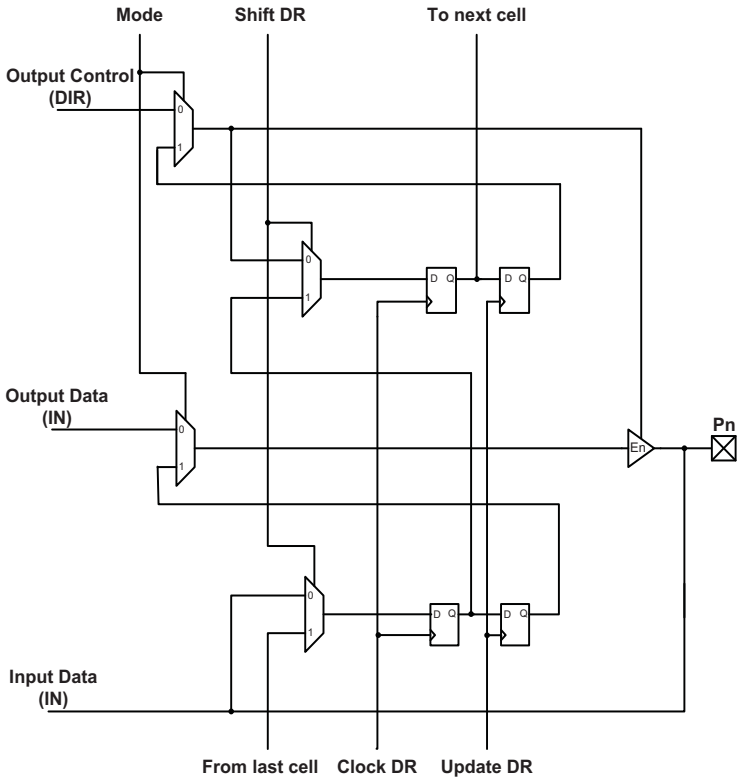
28.5 Boundary Scan Chain

The boundary scan chain has the capability of driving and observing the logic levels on the I/O pins. To ensure a predictable device behavior during and after the EXTEST, CLAMP, and HIGHZ instructions, the device is automatically put in reset. During active reset, the external oscillators, analog modules, and non-default port pin settings (like pull-up/down, bus-keeper, wired-AND/OR) are disabled. It should be noted that the current device and port pin state are unaffected by the SAMPLE and PRELOAD instructions.

28.5.1 Scanning the Port Pins

Figure 28-2 on page 358 shows the boundary scan cell used for all the bidirectional port pins. This cell is able to control and observe both pin direction and pin value via a two-stage shift register. When no alternate port function is present, output control corresponds to the DIR register value, output data corresponds to the OUT register value, and input data corresponds to the IN register value (tapped before the input inverter and input synchronizer). Mode represents either an active CLAMP or EXTEST instruction, while shift DR is set when the TAP controller is in its shift DR state.

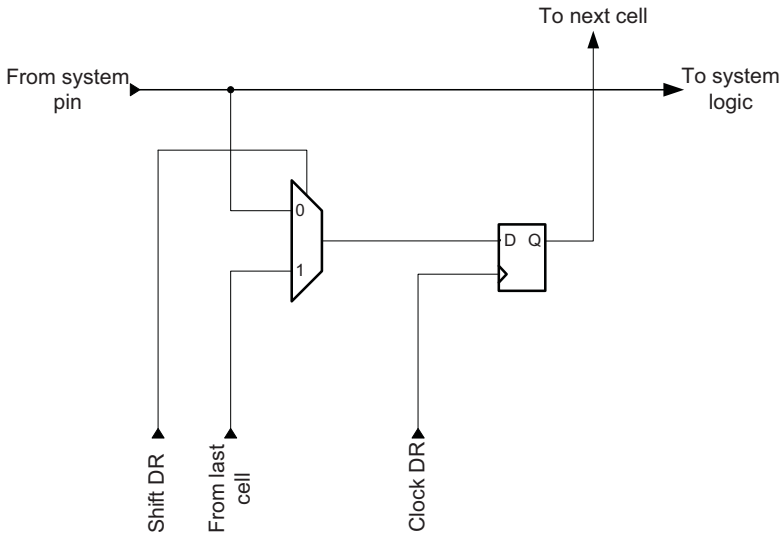
Figure 28-2. Boundary scan cell for bi-directional port pin.



28.5.2 Scanning the PDI Pins

Two observe-only cells are inserted to make the combined RESET and PDI_CLK pin and the PDI_DATA pin observable. Even though the PDI_DATA pin is bidirectional, it is only made observable in order to avoid any extra logic on the PDI_DATA output path.

Figure 28-3. An observe-only input cell.

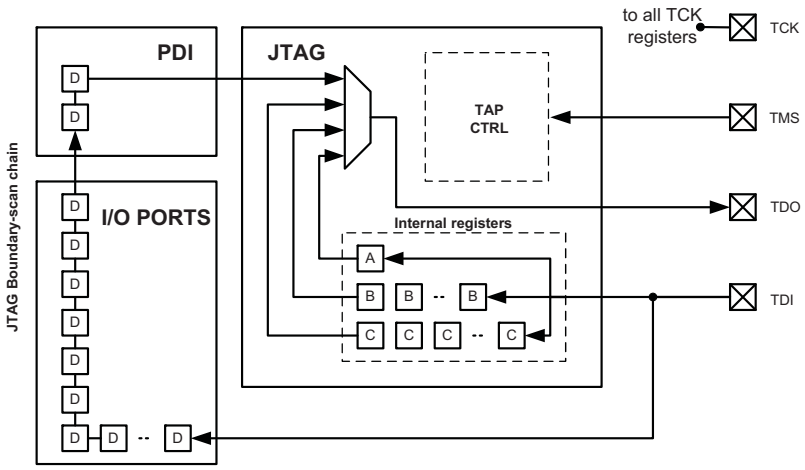


28.6 Data Registers

The supported data registers that can be connected between TDI and TDO are:

- Bypass register (Ref: register A in Figure 28-4 on page 359).
- Device identification register (Ref: register C in Figure 28-4 on page 359).
- Boundary scan chain (Ref: register D in Figure 28-4 on page 359).
- PDICOM data register (Ref: register B in Figure 28-4 on page 359)

Figure 28-4. JTAG data register overview.



28.6.1 Bypass Register

The bypass register consists of a single shift register stage. When the bypass register is selected as the path between TDI and TDO, the register is reset to 0 when leaving the capture DR controller state. The bypass register can be used to shorten the scan chain on a system when the other devices are to be tested.

28.6.2 Device Identification Register

	MSB						LSB
Bit	31	28	27	12	11	1	0
Device ID	Version		Part Number		Manufacturer ID		1
	4 bits		16 bits		11 bits		1 bit

28.6.2.1 Version

Version is a 4-bit number identifying the revision of the device. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1, and so on.

28.6.2.2 Part Number

The part number is a 16-bit code identifying the device. Refer to the device data sheets to find the correct number.

28.6.2.3 Manufacturer ID

The manufacturer ID is an 11-bit code identifying the manufacturer. For Atmel, this code is 0x01F.

28.6.3 Boundary Scan Chain

The boundary scan chain has the capability of driving and observing the logic levels on all I/O pins. Refer to “Boundary Scan Chain” on page 357 for a complete description.

28.6.4 PDICOM Data Register

The PDICOM data register is a 9-bit wide register used for serial-to-parallel and parallel-to-serial conversions of data between the JTAG TAP and the PDI. For details, refer to “Program and Debug Interface” on page 393.

29. Program and Debug Interface

29.1 Features

- Programming
  - External programming through PDI or JTAG interfaces
    - 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
  - Nonintrusive, 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
- JTAG interface
  - Four-pin, IEEE Std. 1149.1 compliant interface for programming and debugging
  - Boundary scan capabilities according to IEEE Std. 1149.1 (JTAG)

29.2 Overview

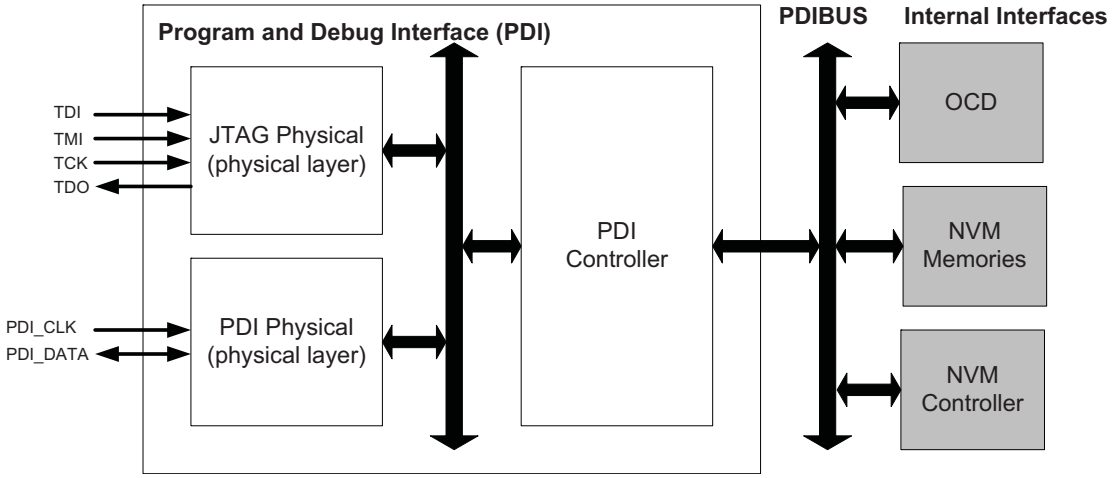
The Program and Debug Interface (PDI) is an Atmel proprietary interface for external programming and on-chip debugging of a device.

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 407.

Debug is supported through an on-chip debug system that offers nonintrusive, 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). A JTAG interface is also available on most devices, and this can be used for programming and debugging through the four-pin JTAG interface. The JTAG interface is IEEE Std. 1149.1 compliant, and supports boundary scan. Any external programmer or on-chip debugger/emulator can be directly connected to either of these interfaces. Unless otherwise stated, all references to the PDI assume access through the PDI physical layer.

Figure 29-1. The PDI with JTAG and PDI physical layers and closely related modules (grey).

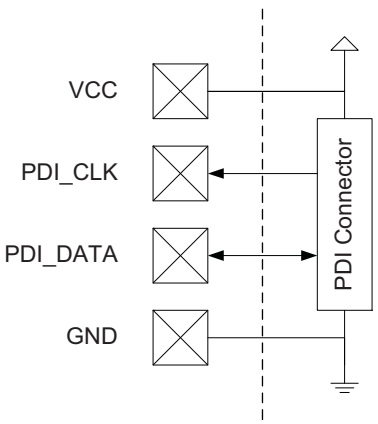


29.3 PDI Physical

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 29-2 on page 362 shows a typical connection.

Figure 29-2. PDI connection.



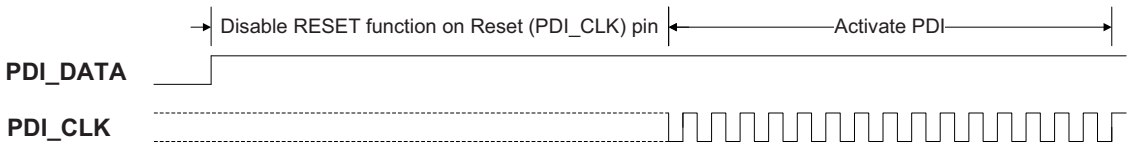
The remainder of this section is intended for use only by third parties developing programmers or programming support for Atmel AVR XMEGA devices.

29.3.1 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 29-3 on page 363.

Figure 29-3. 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.

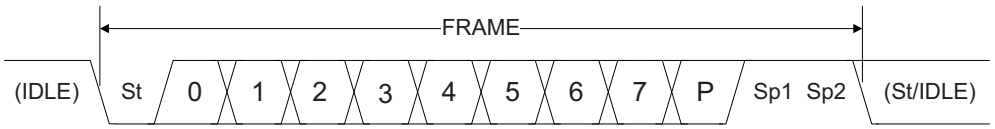
29.3.2 Disabling

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.

29.3.3 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.

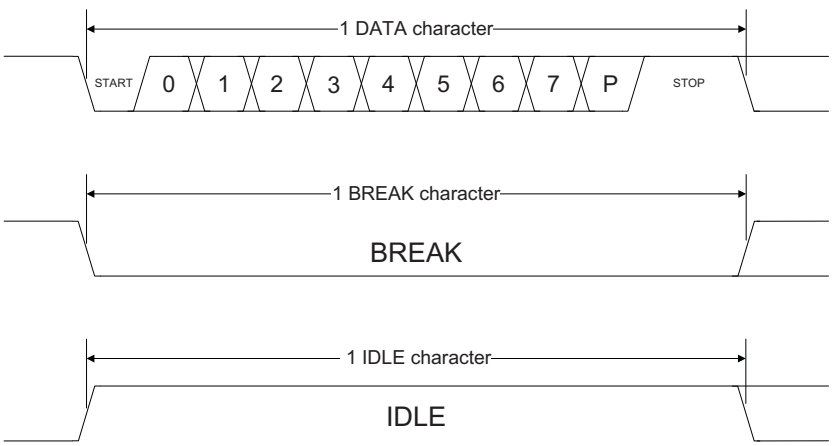
Figure 29-4. PDI serial frame format.



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 29-5. Characters and timing for the PDI physical layer.

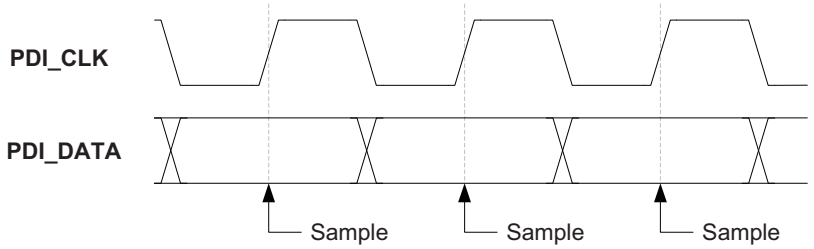


29.3.4 Serial Transmission and Reception

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 29-6 on page 364, 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 29-6. Changing and sampling of data.



29.3.5 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.

29.3.6 Serial Reception

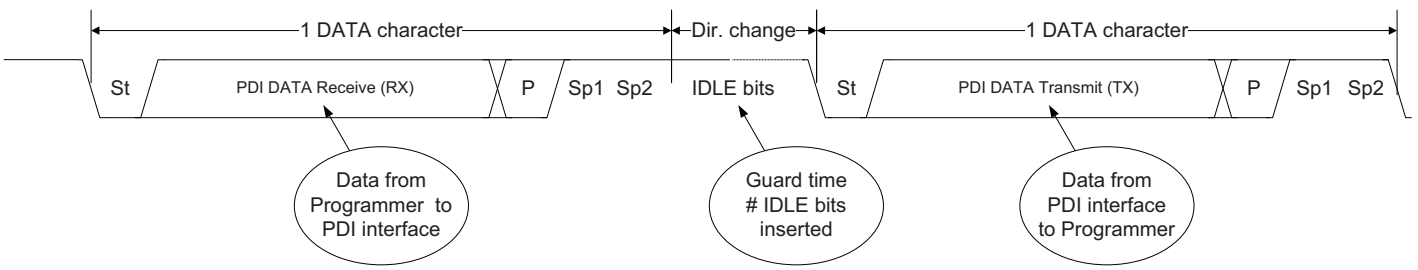
When a start bit is detected, the receiver starts to collect the eight data bits. If the parity bit does not correspond to the parity of the data bits, a parity error has occurred. If one or both of the stop bits are low, a frame error has occurred. If the parity bit is correct, and no frame error is 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.

29.3.7 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 29-7. PDI direction change by inserting IDLE bits.

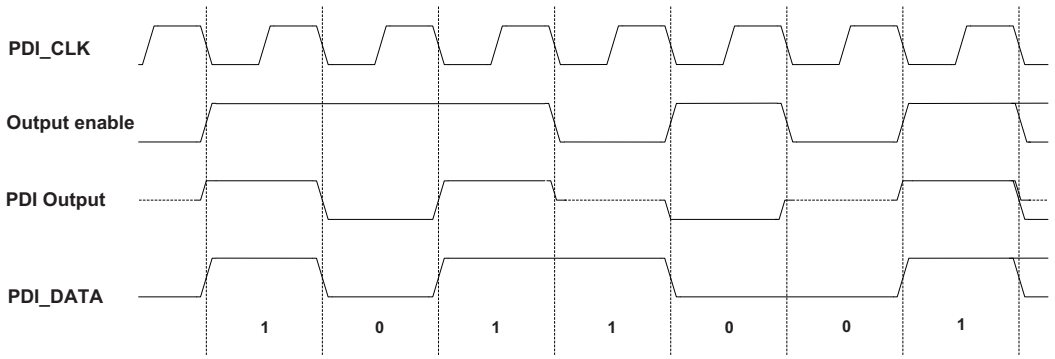


The external programmer will loose control of the PDI_DATA line at the point where the PDI changes from RX to TX mode. The guard time relaxes this critical phase of the communication. When the programmer changes from RX mode to TX mode, a single IDLE bit, at minimum, should be inserted before the start bit is transmitted.

29.3.8 Drive Contention and Collision Detection

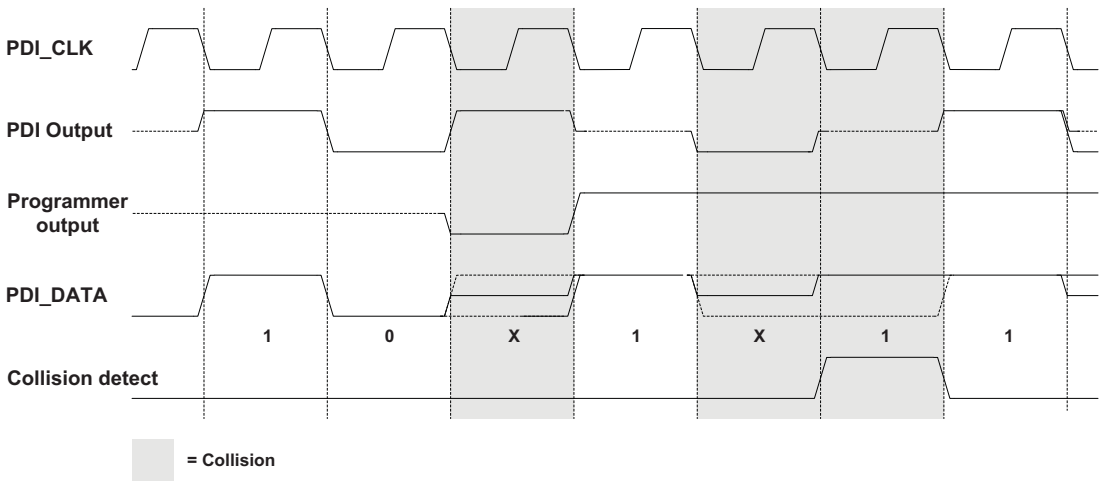
In order to reduce the effect of drive contention (the PDI and the programmer driving the PDI_DATA line at the same time), a mechanism for collision detection is used. The mechanism is based on the way the PDI drives data out on the PDI_DATA line. As shown in Figure 29-8 on page 365, the PDI output driver 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 reenabled due to a change in the bit value.

Figure 29-8. 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 29-9 on page 365. 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 29-9. 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.

29.4 JTAG Physical

The JTAG physical layer handles the basic low-level serial communication over four I/O lines, TMS, TCK, TDI, and TDO. The JTAG physical layer includes BREAK detection, parity error detection, and parity generation. For all generic JTAG details, refer to “IEEE 1149.1 JTAG Boundary Scan Interface” on page 386.

29.4.1 Enabling

The JTAGEN fuse must be programmed and the JTAG disable bit in the MCU control register must be cleared to enable the JTAG interface. This is done by default. When the JTAG PDICOM instruction is shifted into the JTAG instruction register, the JTAG interface can be used to access the PDI for external programming and on-chip debugging.

29.4.2 Disabling

The JTAG interface can be disabled by unprogramming the JTAGEN fuse or by setting the JTAG disable bit in the MCU control register from the application code.

29.4.3 JTAG Instruction Set

The Atmel XMEGA specific JTAG instruction set consist of eight instructions related to boundary scan and PDI access for programming. For more details on JTAG and the general JTAG instruction set, refer to “JTAG Instructions” on page 388.

29.4.3.1 The PDICOM Instruction

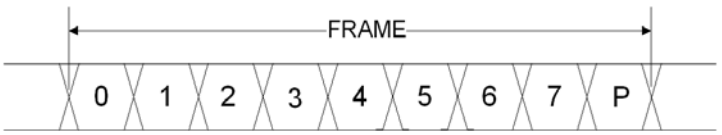
When the PDICOM instruction is shifted into the JTAG instruction register, the 9-bit PDI communication register is selected as the data register. Commands are shifted into the register as results from previous commands are shifted out from the register. The active TAP controller states are (see “TAP - Test Access Port” on page 386):

- Capture DR: Parallel data from the PDI controller is sampled into the PDI communication register
- Shift DR: The PDI communication register is shifted by the TCK input
- Update DR: Commands or operands are parallel-latched into registers in the PDI controller

29.4.4 Frame Format and Characters

The JTAG physical layer supports a fixed frame format. A serial frame is defined to be one character of eight data bits followed by one parity bit.

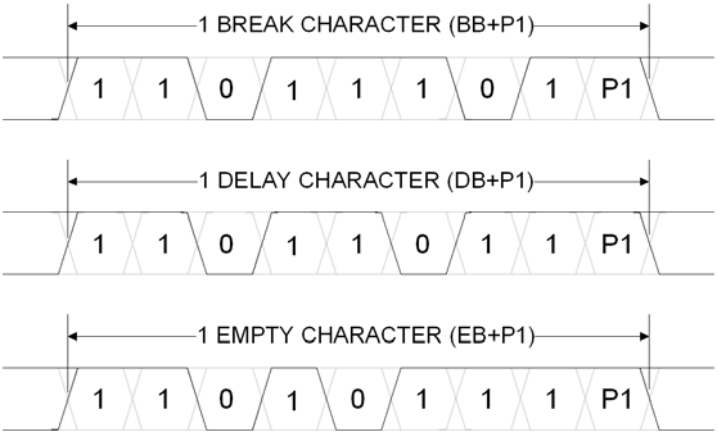
Figure 29-10. JTAG serial frame format



(0-7)	Data/command bits, least-significant bit sent first (0 to 7)
P	Parity bit, even parity used

Three special data characters are used. Common among these is that the parity bit is inverted in order to force a parity error upon reception. The BREAK character (0xBB+P1) is used by the external programmer to force the PDI to abort any ongoing operation and bring the PDI controller into a known state. The DELAY character (0xDB+P1) is used by the PDI to tell the programmer that it has no data ready. The EMPTY character (0xEB+P1) is used by the PDI to tell the programmer that it has no transmission pending (i.e., the PDI is in RX-mode).

Figure 29-11. Special data characters.

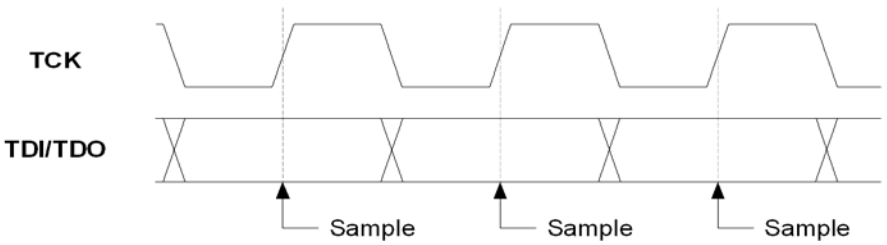


29.4.5 Serial transmission and reception

The JTAG interface supports full-duplex communication. At the same time as input data is shifted in on the TDI pin, output data is shifted out on the TDO pin. However, PDI communication relies on half-duplex data transfer. Due to this, the JTAG physical layer operates only in either transmit (TX) or receive (RX) mode. The available JTAG bit channel is used for control and status signalling.

The programmer and the JTAG interface operate synchronously on the TCK clock provided by the programmer. The dependency between the clock edges and data sampling or data change is fixed. As illustrated in Figure 29-12 on page 367, TDI and TDO is always set up (change) on the falling edge of TCK, while data always should be sampled on the rising edge of TCK.

Figure 29-12. Changing and sampling data.



29.4.6 Serial Transmission

When data transmission is initiated, a data byte is loaded into the shift register and then out on TDO. The parity bit is generated and appended to the data byte during transmission. The transmission speed is given by the TCK signal.

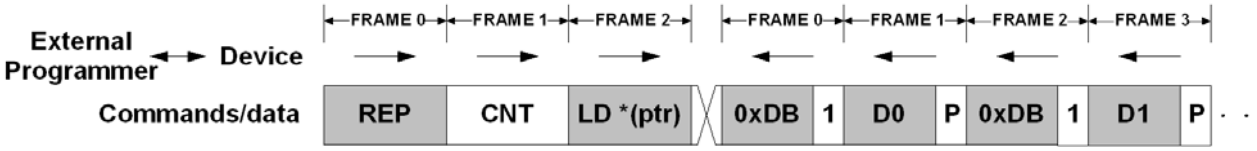
If the PDI is in TX mode (as a response to an LD instruction), and a transmission request from the PDI controller is pending when the TAP controller enters the capture DR state, valid data will be parallel-loaded into the shift register, and a correct parity bit will be generated and transmitted along with the data byte in the shift DR state.

If the PDI is in RX mode when the TAP controller enters the capture DR state, an EMPTY byte will be loaded into the shift register, and the parity bit will be set (forcing a parity error) when data is shifted out in the shift DR state. This situation occurs during normal PDI command and operand reception.

If the PDI is in TX- mode (as a response to an LD instruction), but no transmission request from the PDI controller is pending when the TAP controller enters the capture DR state, a DELAY byte (0xDB) will be loaded into the shift register, and the parity bit will be set (forcing a parity error) when data is shifted out in the shift DR state. This situation occurs during data transmission if the data to be transmitted is not yet available.

Figure 29-13 on page 368 shows an uninterrupted flow of data frames from the PDI as a response to the repeated indirect LD instruction. In this example, the device is not able to return data bytes faster than one valid byte per two transmitted frames. Thus, intermediate DELAY characters are inserted.

Figure 29-13. Data not ready marking.



If a DELAY data frame is transmitted as a response to an LD instruction, the programmer should interpret this as if the JTAG interface had no data ready for transmission in the previous capture DR state. The programmer must initiate repeated transfers until a valid data byte is received. The LD instruction is defined to return a specified number of valid frames, not just a number of frames. Hence, if the programmer detects a DELAY character after transmitting an LD instruction, the LD instruction should not be retransmitted, because the first LD response would still be pending.

29.4.7 Serial Reception

During reception, the PDI collects the eight data bits and the parity bit from TDI and shifts them into the shift register. Every time a valid frame is received, the data is latched in to the update DR state.

The parity checker calculates the parity (even mode) of the data bits in incoming frames and compares the result with the parity bit from the serial frame. In case of a parity error, the PDI controller is signaled.

The parity checker is active in both TX and RX modes. If a parity error is detected, the received data byte is evaluated and compared with the BREAK character (which will always generate a parity error). In case the BREAK character is recognized, the PDI controller is signaled.

29.5 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).

29.5.1 Switching between PDI and JTAG modes

The PDI controller uses either the JTAG or PDI physical layer for establishing a connection to the programmer. Based on this, the PDI is in either JTAG or PDI mode. When one of the modes is entered, the PDI controller registers will be initialized, and the correct clock source will be selected. The PDI mode has higher priority than the JTAG mode. Hence, if the PDI mode is enabled while the PDI controller is already in JTAG mode, the access layer will automatically switch over to PDI mode. If switching physical layer without powering on/off the device, the active layer should be disabled before the alternative physical layer is enabled.

29.5.2 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 Table 33-3 on page 421. 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.

29.5.3 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

29.5.4 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 and whether PDI or JTAG mode is used.

While the PDI is in RX mode, the exceptions are:

- PDI:
  - 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)
- JTAG:
  - The physical layer detects a parity error
  - The physical layer recognizes a BREAK character (also detected as a parity error)

While the PDI is in TX mode, the exceptions are:

- PDI:
  - The physical layer detects a data collision
- JTAG:
  - The physical layer detects a parity error (on the dummy data shifted in on TDI)
  - The physical layer recognizes a BREAK character

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.

29.5.5 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.

29.5.6 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.

29.5.6.1 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.

29.5.6.2 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.

29.5.6.3 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.

29.5.6.4 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.

29.5.6.5 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.

29.5.6.6 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.

29.5.6.7 KEY - Set Activation Key

The KEY instruction is used to communicate the activation key bytes required for activating the NVM interfaces.

29.5.6.8 REPEAT - Set Instruction Repeat Counter

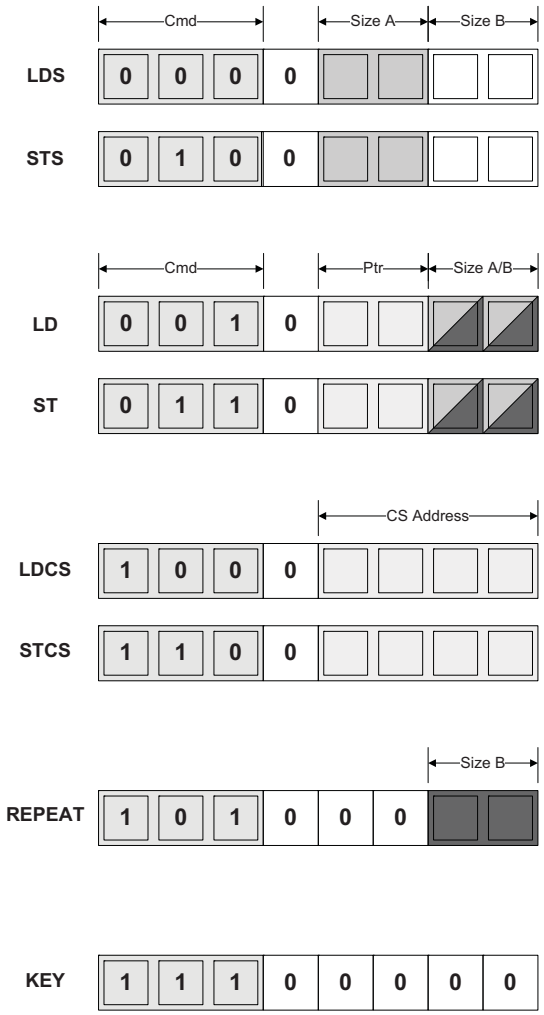
The REPEAT instruction is used to store count values that are serially shifted into the physical layer shift register to the repeat counter register. The instruction that is loaded directly after the REPEAT instruction operand(s) will be repeated a number of times according to the specified repeat counter 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.

29.5.7 Instruction Set Summary

The PDI instruction set summary is shown in Figure 29-14 on page 371.

Figure 29-14. 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

29.6 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.

29.6.1 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.

29.6.2 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).

29.6.3 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.

29.6.4 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.

29.7 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.

29.7.1 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.

29.7.2 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.

29.7.3 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 29-1 on page 374. 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 29-1. Guard time settings.

GUARDTIME	Number of IDLE Bits
000	128
001	64
010	32
011	16
100	8
101	4
110	2
111	2

29.8 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	–	–	–	–	–	–	NVMEN	–	373
+0x01	RESET	RESET[7:0]								373
+0x02	CTRL	–	–	–	–	–	GUARDTIME[2:0]			373
+0x03	Reserved	–	–	–	–	–	–	–	–	

30. Memory Programming

30.1 Features

- Read and write access to all memory spaces from
  - External programmers
  - Application software self-programming
- Self-programming and boot loader support
  - Read-while-write self-programming
  - CPU can run and execute code while flash is being programmed
  - Any communication interface can be used for program upload/download
- External programming
  - Support for in-system and production programming
  - Programming through serial PDI or JTAG 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

30.2 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 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 or JTAG physical interfaces. “External Programming” on page 388 describes PDI and JTAG 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. The flash allows read-while-write self-programming, meaning that the CPU can continue to operate and execute code while the flash is being programmed. “Self-programming and Boot Loader Support” on page 379 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.

30.3 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 393.

30.4 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.

30.4.1 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.

30.4.2 NVM Read-triggered Commands

NVM read-triggered commands are triggered when the NVM is read, and this is typically used for NVM read operations.

30.4.3 NVM Write-triggered Commands

NVM write-triggered commands are triggered when the NVM is written, and this is typically used for NVM write operations.

30.4.4 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.

30.5 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 “Interrupts and Programmable Multilevel Interrupt Controller” on page 115.

30.6 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 datasheet.

30.6.1 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

30.6.2 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 targed locations will be written or erased. Locations that are not targed 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

30.7 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.

30.7.1 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. When using alternative 1 or 2 above for self-programming, the boot loader provides an effective read-modify-write feature, which allows the software to first read the page, do the necessary changes, and then write back the modified data. 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.

30.7.2 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 unerased 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

30.8 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.

30.9 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.

30.10 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 293 for more details.

30.11 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.

30.11.1 Flash Programming

The boot loader support provides a real read-while-write self-programming mechanism for uploading new program code by the device itself. This feature allows flexible application software updates controlled by the device using a boot loader application that reside in the boot loader section in the flash. 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.

30.11.1.1 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, and thus the CPU can run and execute code 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 30-1. Summary of self-programming functionality.

Section being Addressed during Programming	Section that can be Read during Programming	CPU halted?
Application section	Boot loader section	No
Boot loader section	None	Yes
User signature row section	None	Yes

30.11.1.2 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.

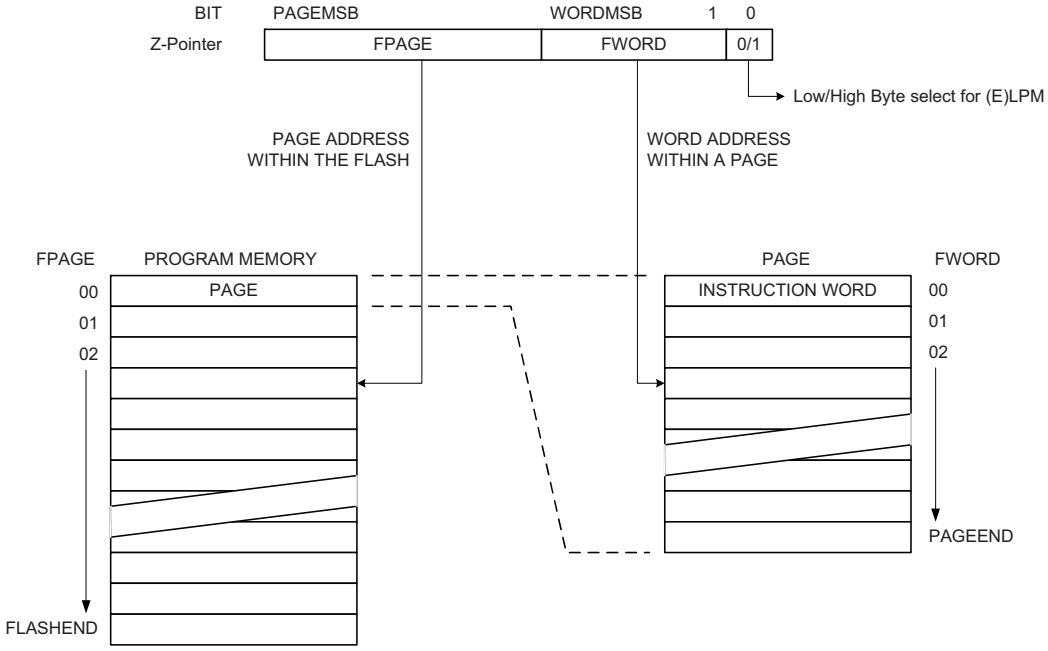
Since the flash is word accessed and organized in pages, the Z-pointer can be treated as having two sections. The least-significant bits address the words within a page, while the most-significant bits address the page within the flash. This is shown in Figure 30-1 on page 380. The word address in the 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 30-1. Flash addressing for self-programming.



30.11.2 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 30-2.

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 "Configuration Change Protection" on page 13. CCP is not required for external programming. The two last columns show the address pointer used for addressing and the source/destination data register.

Section 30.11.1.1 on page 379 through Section 30.11.2.14 on page 384 explain in detail the algorithm for each NVM operation.

Table 30-2. Flash self-programming commands.

CMD[6:0]	Group Configuration	Description	Trigger	CPU Halted	NVM Busy	Change Protected	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								
0x2A	ERASE_BOOT_PAGE	Erase boot loader section page	SPM	Y	Y	Y	Z-pointer	-
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.

30.11.2.1 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.

30.11.2.2 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.

30.11.2.3 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.

30.11.2.4 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.

30.11.2.5 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.

30.11.2.6 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.

30.11.2.7 Erase Application Section

The erase application command is used to erase the complete application section.

1. Load the Z-pointer to point anywhere in the application section.
2. Load the NVM CMD register with the erase application section command

3. Execute the SPM instruction. This requires the timed CCP sequence during self-programming.

The BUSY flag in the STATUS register will be set until the operation is finished. The CPU will be halted during the complete execution of the command.

30.11.2.8 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.

30.11.2.9 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.

30.11.2.10 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.

30.11.2.11 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.

30.11.2.12 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.

30.11.2.13 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.

30.11.2.14 Read User Signature Row / Production Signature Row

The read user signature row and read 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.

30.11.3 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 30-3.

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 30.11.3.1 on page 385 through Section 30.11.3.2 on page 385 explain in detail the algorithm for each NVM operation.

Table 30-3. 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	-

30.11.3.1 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.

30.11.3.2 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.

30.11.4 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.

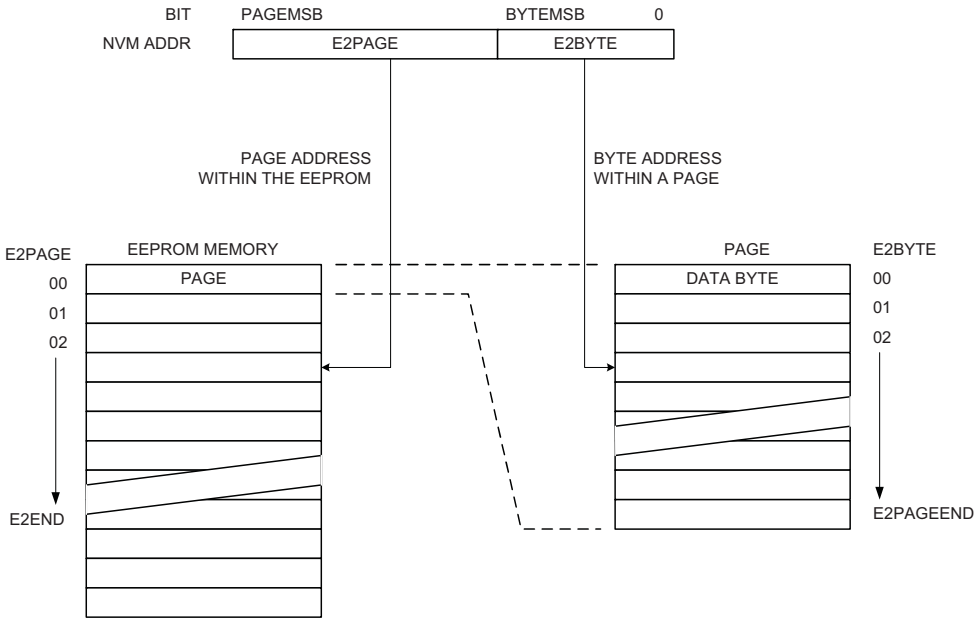
30.11.4.1 Addressing the EEPROM

The EEPROM can be accessed through the NVM controller (I/O mapped), similar to accessing the flash program memory, or it can be memory mapped into the data memory space to be accessed similar to SRAM.

When accessing the EEPROM through the NVM controller, the NVM address (ADDR) register is used to address the EEPROM, while the NVM data (DATA) register is used to store or load EEPROM data.

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 30-2 on page 386. 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 30-2. I/O mapped EEPROM addressing.



When EEPROM memory mapping is enabled, 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 a memory mapped EEPROM page buffer load operation is performed, the CPU is halted for two cycles before the next instruction is executed.

When the EEPROM is memory mapped, the EEPROM page buffer load and EEPROM read functionality from the NVM controller are disabled.

30.11.5 NVM EEPROM Commands

The NVM flash commands that can be used for accessing the EEPROM through the NVM controller are listed in Table 30-4.

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 30.11.5.1 on page 387 through Section 30.11.5.7 on page 388 explain in detail the algorithm for each EEPROM operation.

Table 30-4. 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								
0x33	LOAD_EEPROM_BUFFER	Load EEPROM page buffer	DATA0	N	Y	N	ADDR	DATA0
0x36	ERASE_EEPROM _BUFFER	Erase EEPROM page buffer	CMDEX	N	Y	Y	-	-

CMD[6:0]	Group Configuration	Description	Trigger	CPU Halted	Change Protected	NVM Busy	Address Pointer	Data Register
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	-	-
0x06	READ_EEPROM	Read EEPROM	CMDEX	N	Y	N	ADDR	DATA0

30.11.5.1 Load EEPROM Page Buffer

The load EEPROM page buffer command is used to load one byte into the EEPROM page buffer.

1. Load the NVM CMD register with the load EEPROM page buffer command.
2. Load the NVM ADDR0 register with the address to write.
3. Load the NVM DATA0 register with the data to write. This will trigger the command.

Repeat steps 2-3 until the arbitrary number of bytes are loaded into the page buffer.

30.11.5.2 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.

30.11.5.3 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.

30.11.5.4 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.

30.11.5.5 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.

30.11.5.6 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.

30.11.5.7 Read EEPROM

The read EEPROM command is used to read one byte from the EEPROM.

1. Load the NVM CMD register with the read EEPROM command.
2. Load the NVM ADDR register with the address to read.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

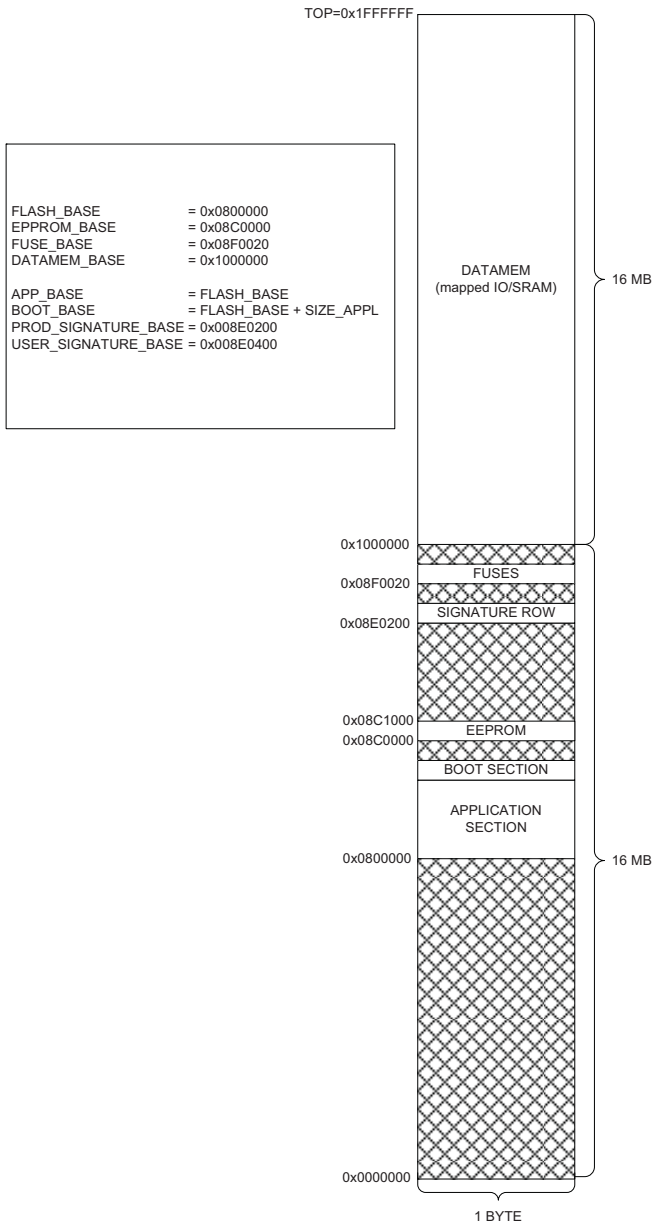
The data byte read will be available in the NVM DATA0 register.

30.12 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 either the JTAG or PDI physical connection. For details on PDI and JTAG and how to enable and use the physical interface, refer to “Program and Debug Interface” on page 361. 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 30-3 on page 389 shows the PDI memory space and the base address for each memory space in the device.

Figure 30-3. Memory map for PDI accessing the data and program memories.



30.12.1 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.

30.12.2 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 30-3 on page 389.

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.

30.12.3 NVM Commands

The NVM commands that can be used for accessing the NVM memories from external programming are listed in Table 30-5 on page 390. This is a superset 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).

" Chip Erase" on page 391 through " Write Fuse/ Lock Bit" on page 393 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 30-5. 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
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

CMD[6:0]	Commands / Operation	Trigger	Change Protected	NVM Busy
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.

30.12.3.1 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.

30.12.3.2 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.

30.12.3.3 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.

30.12.3.4 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.

30.12.3.5 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.

30.12.3.6 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.

30.12.3.7 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.

30.12.3.8 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.

30.12.3.9 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.

30.12.3.10 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.

30.12.3.11 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.

30.13 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 373 for a complete register description of the PDI.

30.14 Register Summary

Refer to “Register Description – NVM Controller” on page 26 for a complete register summary of the NVM controller.

Refer to “Register Summary” on page 374 for a complete register summary of the PDI.

31. 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.

Base Address	Name	Description	Page
0x0000	GPIO	General Purpose IO Registers	page 42
0x0010	VPORT0	Virtual Port 0	page 132
0x0014	VPORT1	Virtual Port 1	
0x0018	VPORT2	Virtual Port 2	
0x001C	VPORT3	Virtual Port 3	
0x0030	CPU	CPU	page 19
0x0040	CLK	Clock Control	page 96
0x0048	SLEEP	Sleep Controller	page 101
0x0050	OSC	Oscillator Control	page 96
0x0060	DFLLRC32M	DFLL for the 32 MHz Internal RC Oscillator	page 96
0x0068	DFLLRC2M	DFLL for the 2 MHz RC Oscillator	
0x0070	PR	Power Reduction	page 98
0x0078	RST	Reset Controller	page 109
0x0080	WDT	Watch-Dog Timer	page 114
0x0090	MCU	MCU Control	page 42
0x00A0	PMIC	Programmable Multilevel Interrupt Controller	page 122
0x00B0	PORTCFG	Port Configuration	page 146
0x00C0	AES	AES Module	page 292
0x0100	DMA	DMA Controller	page 62
0x0180	EVSYS	Event System	page 74
0x01C0	NVM	Non Volatile Memory (NVM) Controller	page 46
0x0200	ADCA	Analog to Digital Converter on port A	page 344
0x0240	ADCB	Analog to Digital Converter on port B	
0x0380	ACA	Analog Comparator pair on port A	page 353
0x0390	ACB	Analog Comparator pair on port B	
0x0400	RTC	Real Time Counter	page 205
0x0480	TWIC	Two Wire Interface on port C	page 254
0x04C0	USB	Universal Serial Bus Interface	page 281
0x0600	PORTA	Port A	page 146
0x0620	PORTB	Port B	
0x0640	PORTC	Port C	
0x0660	PORTD	Port D	
0x0680	PORTE	Port E	
0x06C0	PORTG	Port G	
0x0760	PORTM	Port M	
0x07E0	PORTR	Port R	
0x0800	TCC0	Timer/Counter 0 on port C	page 172
0x0840	TCC1	Timer/Counter 1 on port C	
0x0880	AWEXC	Advanced Waveform Extension on port C	page 195
0x0890	HIRESC	High Resolution Extension on port C	page 197
0x08A0	USARTC0	USART 0 on port C	page 281
0x08C0	SPIC	Serial Peripheral Interface on port C	page 260
0x08F0	IRCOM	Infrared Communication Module	page 285
0x0A00	TCE0	Timer/Counter 0 on port E	page 172
0x0AA0	USARTE0	USART 0 on port E	page 281
0x0D00	LCD	LCD - Liquid Crystal Display	page 319

32. 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
DES	K	Data Encryption	if (H = 0) then R15:R0 ← Encrypt(R15:R0, K) else if (H = 1) then R15:R0 ← Decrypt(R15:R0, K)		1/2
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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	2 / 3[1]
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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
LDI	Rd, K	Load Immediate	Rd ← K	None	1
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.

33. Datasheet 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.

33.1 8291C – 06/2014

1.	Replaced RCOSC48M with USBRCOSC in Section 4.15.19 on page 40 and in Section 4.20 on page 45.
2.	Changed V_{CC} to AV_{CC} in the section "AC – Analog Comparator" on page 345 and onwards, and in "Voltage Reference Selection" on page 324.
3.	Updated last page and footers from template of May 5 2014.

33.2 8291B – 01/2013

1.	Added XMEGA B feature overview inTable 2-1 on page 5.
2.	References to Calibration Row updated to Production Signature Row for consistency.
3.	Added reference to "NVM Flash Commands" on page 380 in "Production Signature Row" on page 22.
4.	Updated "LOCKBITS – Lock Bits register" on page 30. Description of Bit[1:0] updated and added a table note.
5.	Title of Table 4-12 on page 35 changed to "Lock bit protection mode."
6.	Updated "TRIGSRC – Trigger Source" on page 57. The description Bit[7:0] updated.
7	Updated description of "CHnCTRL – Event Channel n Control register" on page 72.
8.	Updated the formula of COMP register in "DFLL 2MHz and DFLL 32MHz" on page 82.
9.	Updated Table 9-2 on page 105, the "Programmable BODLEVEL setting."
10.	Table note added to the Table 10-1 on page 112.
11.	Table note added to the Table 10-2 on page 113.
12.	Updated "Port Interrupt" on page 129.
13.	Updated Table 12-3 on page 130. "Both edge" replaced by "Any edge".
14.	Updated "Port Event" on page 130.
15.	Updated Table 12-10 on page 142, and Table 12-11 on page 143.
16.	Updated "Event Action Controlled Operation" on page 153.
17.	Updated Figure 13-10 on page 155. CH7MUX changed to CHnMUX.
18	Updated Table 13-2 in "DMA Support" on page 160.
19.	Updated Table 14-3 on page 179. CMD changed to BYTEM[1:0]
20.	Updated "Clock Domains" on page 199.
21.	Updated description in "For Output Endpoints" on page 213.
22.	Updated both formula of "BAUD – Baud Rate register" on page 247
23.	Updated "DATA – Data register" on page 248. Added the description of ADDR[7:1] and ADDR[0]

24.	Updated the formula in “Fractional Baud Rate Generation” on page 271.
25.	Updated Figure 21-9 on page 272, the “Fractional baud rate example.”
26.	Added Table 21-5 on page 272, the “USART baud rate.”
27.	Updated “ADC Input Model” on page 329.
26.	Updated “Synchronous Sampling” on page 330.
27	Updated description of “Bit 3:0 – COUNT[3:0]: Number of Input Channels Included in Scan” in “SCAN – Input Channel Scan register” on page 343
28.	Updated Analog Comparator overview block diagram in Figure 27-1 on page 346.

33.3 8291A – 07/2011

1.	Initial revision edited from XMEGA AU Manual rev A 07/11

Table of Contents

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**