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

- **AVR CPU**
- **Memories**
- **DMA - Direct Memory Access Controller**
- **Event System**
- **System Clock and Clock options**
- **Power Management and Sleep Modes**
- **System Control and Reset**
- **Battery Backup System**
- **WDT - Watchdog Timer**
- **Interrupts and Programmable Multi-level Interrupt Controller**
- **PORT - I/O Ports**
- **TC - 16-bit Timer/Counter**
- **AWeX - Advanced Waveform Extension**
- **Hi-Res - High Resolution Extension**
- **RTC - Real Time Counter**
- **RTC32 - 32-bit Real Time Counter**
- **TWI - Two Wire Serial Interface**
- **SPI - Serial Peripheral Interface**
- **USART - Universal Synchronous and Asynchronous Serial Receiver and Transmitter**
- **IRCOM - IR Communication Module**
- **AES and DES Crypto Engine**
- **EBI - External Bus Interface**
- **ADC - Analog to Digital Converter**
- **DAC - Digital to Analog 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**

8-bit AVR® XMEGA A Microcontroller

XMEGA A MANUAL

Preliminary

8077H- AVR-12/09

1. About the Manual

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

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

For more details on applied use and code examples for all peripherals and modules, refer to the XMEGA A 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 of the most important features and a short overview describing the module. The remaining section describes the features and functions in more details.

The register description sections list all registers, and describe each bit/flag and its 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 refer to the defined configuration name used in the XMEGA A and 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 on http://www.atmel.com/avr.

1.3 Recommended Reading

- XMEGA A Device Datasheets
- XMEGA Application Notes

This manual only contains general modules and peripheral descriptions. The XMEGA A device datasheet contains device specific information. The XMEGA application notes 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 XMEGA, and AVR1900 - Getting started with ATxmega128A1 application notes.

2. Overview

The XMEGA A is a family of low power, high performance and peripheral rich CMOS 8/16-bit microcontrollers based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the XMEGA A achieves throughputs approaching 1 Million Instructions Per Second (MIPS) per MHz 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 the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction, executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs many times faster than conventional single-accumulator or CISC based microcontrollers.

The XMEGA A devices provide the following features: In-System Programmable Flash with Read-While-Write capabilities, Internal EEPROM and SRAM, four-channel DMA Controller, eight-channel Event System and Programmable Multi-level Interrupt Controller, up to 78 general purpose I/O lines, 16- or 32-bit Real Time Counter (RTC), up to eight flexible 16-bit Timer/Counters with compare modes and PWM, up to eight USARTs, up to four I^2C and SMBUS compatible Two Wire Serial Interfaces (TWIs), up to four Serial Peripheral Interfaces (SPIs), AES and DES crypto engine, up to two 8-channel, 12-bit ADCs with optional differential input with programmable gain, up to two 2-channel, 12-bit DACs, 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 2-pin interface for programming and debugging, is available. The devices also have an IEEE std. 1149.1 compliant JTAG test interface, and this can also be used for On-chip Debug and programming.

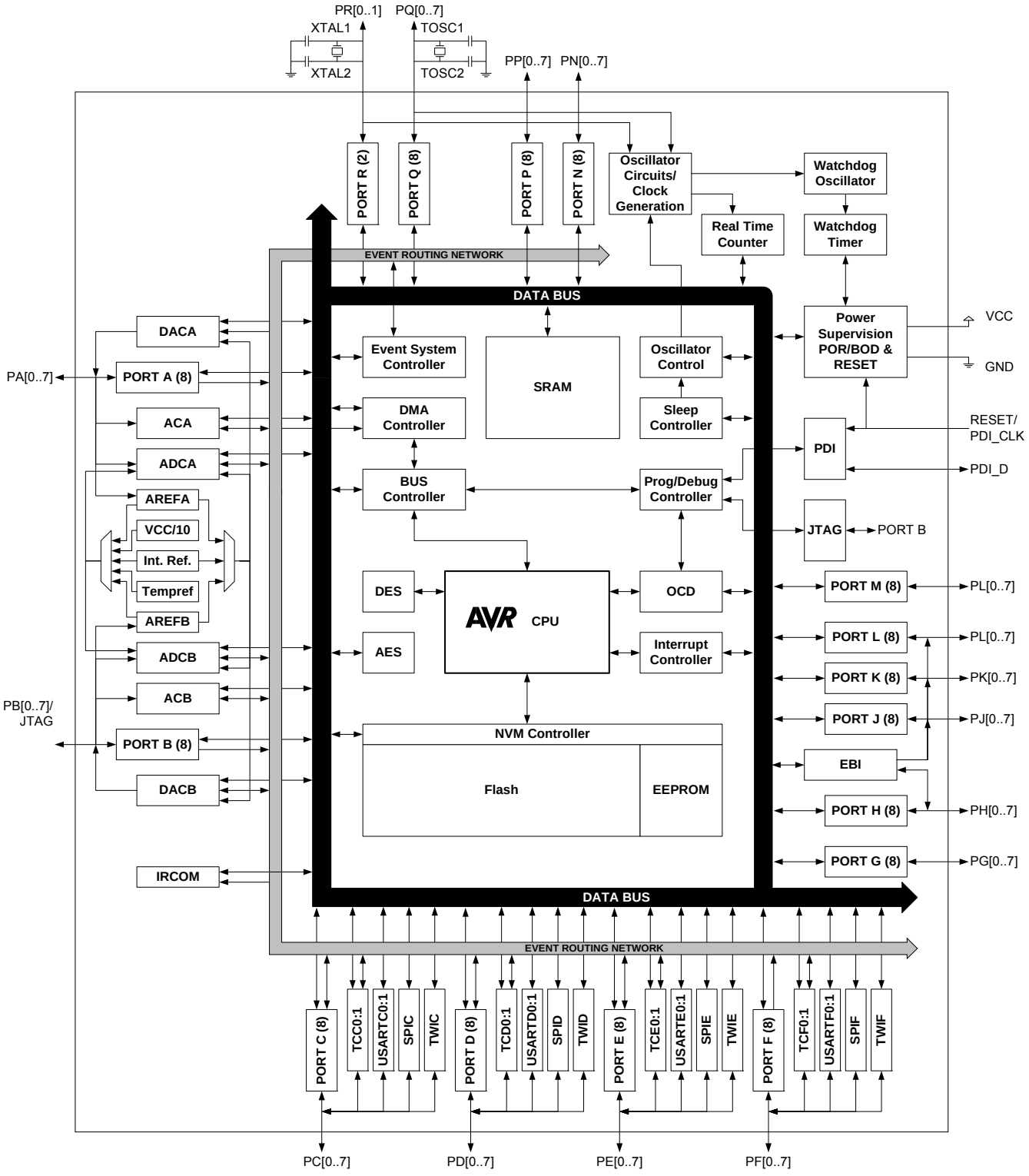
The XMEGA A devices have five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, DMA Controller, Event System, Interrupt Controller and all peripherals to continue functioning. The Power-down mode saves the SRAM and register contents but stops the oscillators, disabling all other functions until the next TWI or pin-change interrupt, or Reset. In Power-save mode, the asynchronous Real Time Counter continues to run, allowing the application to maintain a timer base while the rest of the device is sleeping. In Standby mode, the Crystal/Resonator Oscillator is kept running while the rest of the device is sleeping. This allows very fast start-up from external crystal combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run. To further reduce power consumption, the peripheral clock to each individual peripheral can optionally be stopped in Active mode and Idle sleep mode.

The devices are manufactured using Atmel's high-density nonvolatile memory technology. The program Flash memory can be reprogrammed in-system through the PDI or JTAG. A Bootloader running in the device can use any interface to download the application program to the Flash memory. The Bootloader 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 XMEGA A is a powerful microcontroller family that provides a highly flexible and cost effective solution for many embedded applications.

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

2.1 Block Diagram

Figure 2-1. XMEGA A Block Diagram



3. AVR CPU

3.1 Features

- 8/16-bit high performance AVR RISC CPU
  - 138 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 16M bytes of program memory and 16M bytes of data memory
- True 16/24-bit access to 16/24-bit I/O registers
- Efficient support for both 8-, 16- and 32-bit Arithmetic
- Configuration Change Protection of system critical features

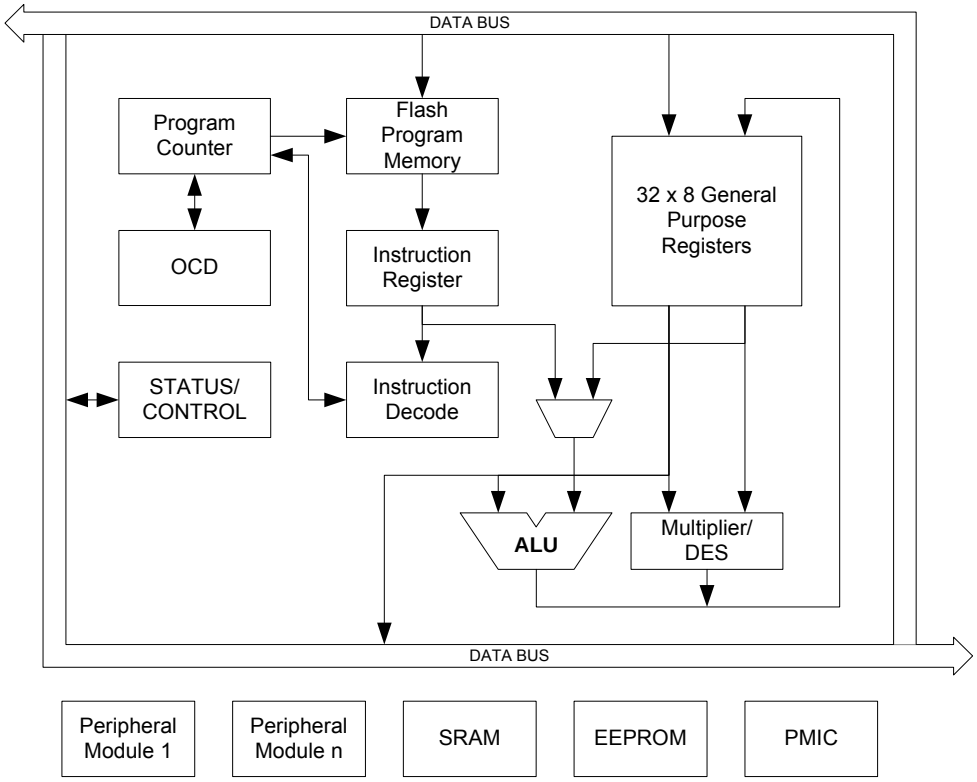
3.2 Overview

XMEGA uses the 8/16-bit AVR CPU. The main function of the CPU is to ensure correct program execution. The CPU is able to access memories, perform calculations and control peripherals. Interrupt handling is described in a separate section, refer to ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123 for more details on this.

3.3 Architectural Overview

In order to maximize performance and parallelism, the AVR uses a Harvard architecture with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the Program Memory. This concept enables instructions to be executed in every clock cycle. For the summary of all AVR instructions refer to ”Instruction Set Summary” on page 388. For details of all AVR instructions refer to http://www.atmel.com/avr.

Figure 3-1. Block Diagram of the AVR 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 (ALU) 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 all linear and regular memory maps. The Data Memory space and the Program Memory space are two different memory spaces.

The Data Memory space is divided into I/O registers and SRAM. In addition the EEPROM can be memory mapped in the Data Memory.

All I/O status and control registers reside in the lowest 4K bytes 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 - 0x3F. The rest is the Extended I/O Memory space, ranging from 0x40 to 0x1FFF. I/O registers here must be access as data space locations using load (LD/LDS/LDD) and store (ST/STS/STD) instructions.

The SRAM holds data, and code cannot be executed from here. It can easily be accessed through the five different addressing modes supported in the AVR architecture. The first SRAM address is 0x2000.

Data address 0x1000 to 0x1FFF is 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. A third section exists inside the Application section. This section, the Application Table section, has separate Lock bits for write and read/write protection. The Application Table section can be used for storing non-volatile data or application software.

3.4 ALU - Arithmetic Logic Unit

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. The ALU operates in direct connection with all the 32 general purpose registers. In a typical single cycle ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File.

In order to operate on data in the Data Memory, these must first be loaded into the Register File. After the operation, the data can be stored from the Register File and back to the Data Memory.

The ALU operations are divided into three main categories - arithmetic, logical, and bit-functions. After an arithmetic or logic operation, the Status Register is updated to reflect information about the result of the operation.

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 and with an unsigned.

A multiplication takes two CPU clock cycles.

3.5 Program Flow

After reset, the program will start to execute from program address 0. Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction. The Program Counter (PC) addresses the location from where instructions are fetched. During interrupts and subroutine calls, the return address PC is stored on the Stack.

When an enabled interrupt occurs, the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine. Hardware clears the corresponding interrupt flag automatically.

A flexible interrupt controller has dedicated control registers with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate interrupt vector, starting from the Reset Vector at address 0 in the Program Memory. All interrupts have a programmable interrupt level. Within each level they have priority in accordance with their interrupt vector position where the lower interrupt vector address has the higher priority.

3.6 Instruction Execution Timing

The AVR CPU is driven by the CPU clock clk_{CPU}. No internal clock division is used. Figure 3-2 on page 8 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 to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 3-2. The Parallel Instruction Fetches and Instruction Executions

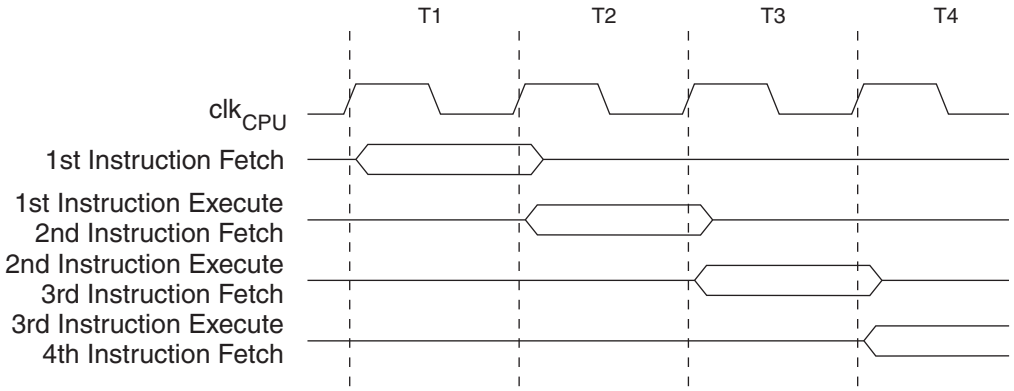
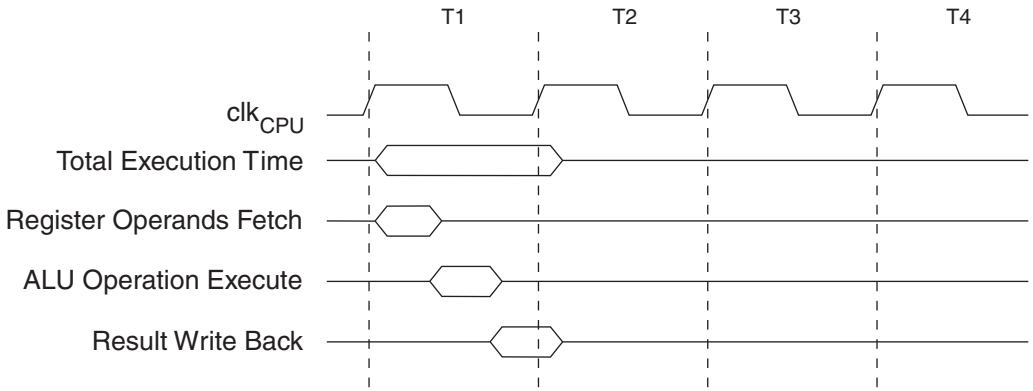


Figure 3-3 on page 8 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

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 is accessible in the I/O Memory space. Data is pushed and popped from the Stack using the PUSH and POP instructions. The Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a pushing data on 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 of the memory size of the device. For devices with 128K bytes or less of program memory the return address is two bytes, hence the Stack Pointer is decremented/incremented by two. For devices with more than 128K bytes of program memory, the return address is three bytes, hence the SP is decremented/incremented by three. The return address is popped of the Stack when returning from interrupts using the RETI instruction, and subroutine calls using the RET instruction.

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

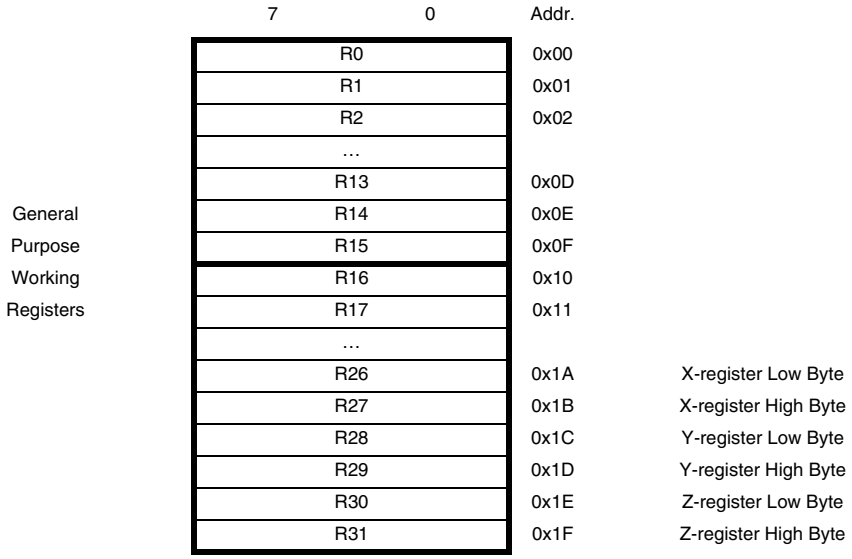
To prevent corruption when updating the Stack Pointer from software, a write to SPL will automatically disable interrupts for up to 4 instructions or until the next I/O memory write.

3.9 Register File

The Register File consists of 32 x 8-bit general purpose registers. In order to achieve the required performance and flexibility, 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

Figure 3-4. AVR CPU General Purpose Working Registers



Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

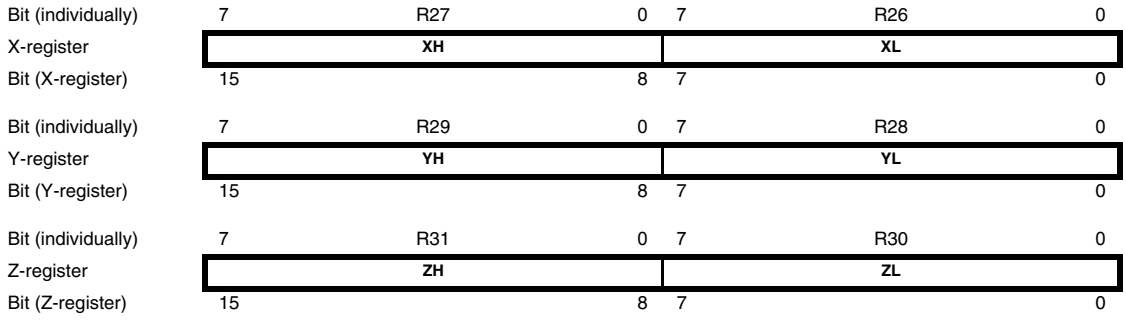
The Register File is located in a separate address space, so the registers are not accessible as data memory.

3.9.1 The X-, Y- and Z- Registers

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

These registers can form 16-bit address pointers for addressing of the Data Memory. The three address registers is called the X-, Y-, 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). In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

3.10 RAMP and Extended Indirect Registers

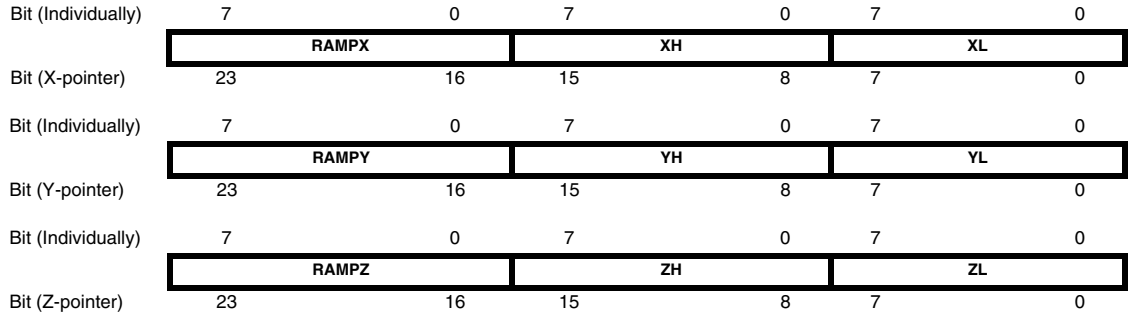
In order to access program memory or data memory above 64K bytes, the address or address pointer must be more than 16-bits. This is done by concatenating one register to one of the X-, Y- or Z-registers, and this register then holds the most significant byte (MSB) in a 24-bit address or address pointer.

These registers are only available on devices with external bus interface and/or more than 64K bytes 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 64K bytes and up to 16M bytes.

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

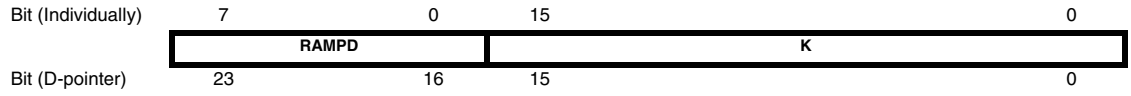


When reading (ELPM) and writing (SPM) program memory locations above the first 128K bytes of the program memory, RAMPZ is concatenated with the Z-register to form the 24-bit address. LPM is not affected by the RAMPZ setting.

3.10.2 RAMPD Register

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

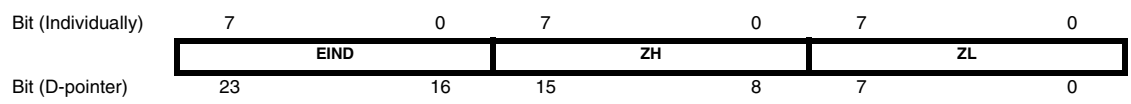
Figure 3-7. The combined RAMPD + K register



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 128K bytes (64K words) of the program memory.

Figure 3-8. The combined EIND + Z register



3.11 Accessing 16-bits Registers

The AVR data bus is 8-bit so accessing 16-bit registers requires atomic operations. These registers must be byte-accessed using two read or write operations. When reading the high byte is buffered and when writing the low byte will be buffered. A 16-bit register is connected to the 8-bit bus and a temporary register using a 16-bit bus. This ensures that the low- and high-byte of 16-bit registers is always accessed simultaneously when reading or writing the register.

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.

Interrupts can corrupt the timed sequence if the interrupt is triggered and try to access the same 16-bit register during an atomic 16-bit read/write operations. 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 register 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 bit, or execution of the protected instructions are only possible after the CPU writes a signature to the CCP register. The different signatures is described the register description.

There are 2 mode of operation, one for protected I/O registers and one for protected SPM/LPM.

3.12.1 Sequence for write operation to protected I/O registers

1. The application code writes the signature for change enable of protected I/O registers to the CCP register.
2. Within 4 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 is written. The protected change is immediately disabled if the CPU performs write operations to the I/O register or data memory, or if the instruction SPM, LPM or SLEEP is executed.

3.12.2 Sequence for execution of protected SPM/LPM

1. The application code writes the signature for execution of protected SPM/LPM to the CCP register.
2. Within 4 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 SLEEP is executed.

Once the correct signature is written by the CPU, interrupts will be ignored for the configuration change enable period. Any interrupt request (including Non-Maskable Interrupts) during the CPP period will set the corresponding interrupt flag as normal and the request is kept pending. After the CPP period 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 the DMA is ignored.

3.13 Fuse Lock

For some system critical features it is possible to program a fuse to disable all changes in 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 data sheet module where this feature is available.

3.14 Register Description

3.14.1 CCP - Configuration Change Protection Register

Bit	7	6	5	4	3	2	1	0	
+0x04	CCP[7:0]								CCP
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 of 4 CPU instruction cycles. All interrupts are ignored during these cycles. After these cycles interrupts automatically 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 be read as zero. Table 3-1 on page 13 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 64K bytes of data memory. When accessing

data addresses below 64K bytes, this register is not in use. This register is not available if the data memory including external memory is less than 64K bytes.

Bit	7	6	5	4	3	2	1	0	
+0x08	RAMPD[7:0]								RAMPD
Read/Write	R/W	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 holds the 8 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 64K bytes of data memory. When accessing data addresses below 64K bytes, this register is not in use. This register is not available if the data memory including external memory is less than 64K bytes.

Bit	7	6	5	4	3	2	1	0	
+0x09	RAMPX[7:0]								RAMPX
Read/Write	R/W	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 holds the 8 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 64K bytes of data memory. When accessing data addresses below 64K bytes, this register is not in use. This register is not available if the data memory including external memory is less than 64K bytes.

Bit	7	6	5	4	3	2	1	0	
+0x0A	RAMPY[7:0]								RAMPY
Read/Write	R/W	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 8 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 64K bytes of data memory. RAMPZ is concatenated with the Z-register when reading (ELPM) program memory locations above the

first 64K bytes, and writing (SPM) program memory locations above the first 128K bytes of the program memory.

When accessing data addressees below 64K bytes, reading program memory locations below 64K bytes and writing program memory locations below 128K bytes, this register is not in use. This register is not available if the data memory including external memory and program memory in the device is less than 64K bytes.

Bit	7	6	5	4	3	2	1	0	
+0x0B	RAMPZ[7:0]								RAMPZ
Read/Write	R/W	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 holds the 8 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 (ECALL) to the whole program memory space devices with more than 128K bytes of program memory. For jump or call to addressees below 128K bytes, this register is not in use. This register is not available if the program memory in the device is less than 128K bytes.

Bit	7	6	5	4	3	2	1	0	
+0x0C	EIND[7:0]								EIND
Read/Write	R/W	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 holds the 8 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 value SP. The SP holds the Stack Pointer that point to the top of the Stack. After reset, the Stack Pointer points to the highest internal SRAM address.

Only the number of bits required to address the available data memory including external memory, up to 64K bytes 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]								SPL
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 byte**

These bits hold the 8 LSB of the 16-bits Stack Pointer (SP).

3.14.8 SPH - Stack Pointer Register High

Bit	7	6	5	4	3	2	1	0	
+0x0E	SP[15:8]								SPH
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.

- **Bits 7:0 - SP[15:8]: Stack Pointer Register High byte**

These bits hold the 8 MSB of the 16-bits 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	SREG
Read/Write	R/W	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. The I-bit is not cleared by hardware after an interrupt has occurred. The I-bit can be set and cleared by the application with the SEI and CLI instructions, as described in the “Instruction Set Description”.

- **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 T by the BST instruction, and a bit in T 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 the “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 the “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 arithmetics. See the “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 the “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 the “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 the “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]								13
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	Reserved	-	-	-	-	-	-	-	-	
+0x07	Reserved	-	-	-	-	-	-	-	-	
+0x08	RAMPD	RAMPD[7:0]								13
+0x09	RAMPX	RAMPX[7:0]								14
+0x0A	RAMPY	RAMPY[7:0]								14
+0x0B	RAMPZ	RAMPZ[7:0]								14
+0x0C	EIND	EIND[7:0]								15
+0x0D	SPL	SPL[7:0]								15
+0x0E	SPH	SPH[7:0]								16
+0x0F	SREG	I	T	H	S	V	N	Z	C	16

4. Memories

4.1 Features

- Flash Program Memory
  - One linear address space
  - In-System Programmable
  - Self-Programming and Bootloader support
  - Application Section for application code
  - Application Table Section for application code or data storage
  - Boot Section for application code or bootloader code
  - Separate lock bits and protection 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
    - 16 bit-accessible General Purpose Register for global variables or flags
  - External Memory support
    - SRAM
    - SDRAM
    - Memory mapped external hardware
  - Bus arbitration
    - Safe and deterministic handling of CPU and DMA Controller priority
  - 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
  - Device ID for each microcontroller device type
  - Serial number for each device
  - Oscillator calibration bytes
  - ADC, DAC and temperature sensor calibration data
- 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 memories in XMEGA. The AVR architecture has two main memory spaces, the Program Memory and the Data Memory. Executable code can only reside in the Program Memory, while data can be stored both in the Program Memory and the Data Memory. The Data Memory includes both SRAM, and an EEPROM Memory for non-volatile data storage. All memory spaces are linear and require no paging. Non-Volatile 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 setting important system functions, and write access is only possible from an external programmer.

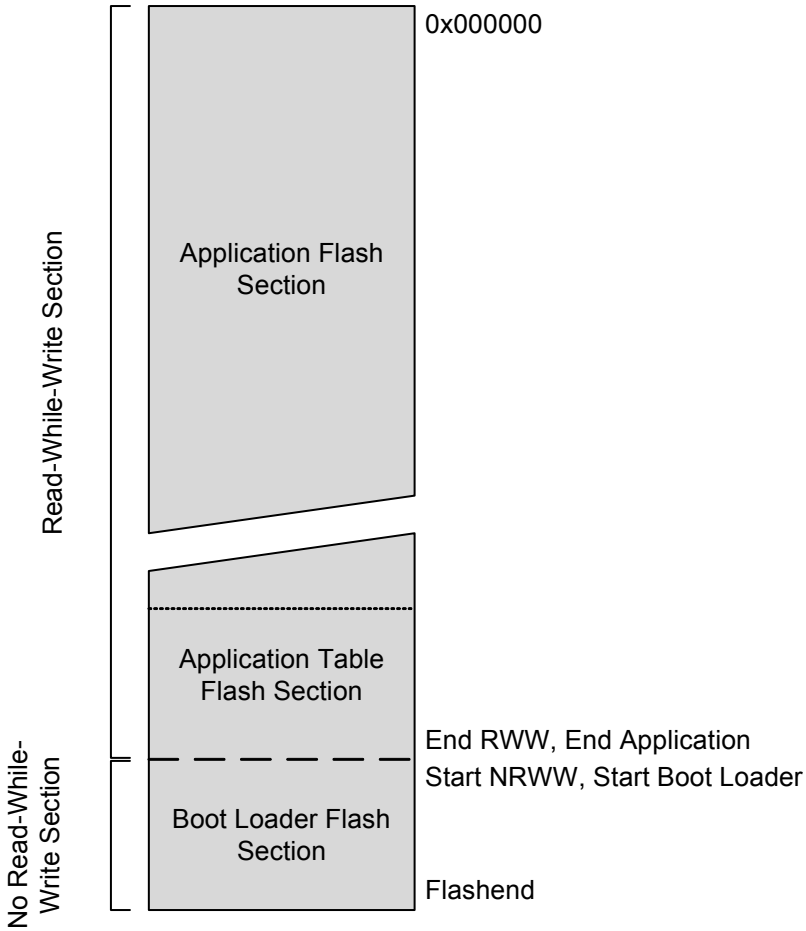
4.3 Flash Program Memory

The XMEGA contains On-chip In-System Reprogrammable Flash memory for program storage. The Flash memory can be accessed for read and write both from an external programmer through the PDI, or from application software running in the CPU.

All AVR instructions are 16 or 32 bits wide, and each Flash location is 16 bits wide. The Flash memory in XMEGA is organized in two main sections, the Application Section and the Boot Loader section, as shown in Figure 4-1 on page 19. The sizes of the different sections are fixed, but device dependent. These two sections have separate lock bits and can have different level of protection. The Store Program Memory (SPM) instruction used to write to the Flash from the application software, will only operate when executed from the Boot Loader Section.

The Application Section contains an Application Table Section with separate lock settings. This can be used for safe storage of Non-volatile 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 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 be 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 since the SPM instruction only can initiate programming when executing from the 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 the calibration conditions such as temperature, voltage references etc. refer to device data sheet.

The production signature row also contains a device ID that identify each microcontroller device type, and a serial number that is unique for each manufactured device. The serial number consist of the production LOT number, wafer number, and wafer coordinates for the device.

The production signature row can not be written or erased, but it can be read from both application software and external programming.

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 programming. The user signature row is one flash page in size, and is meant for static user parameter storage, such as calibration data, custom serial numbers or 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 session and on-chip debug sessions.

4.4 Fuses and Lockbits

The Fuses are used to set important system function and can only be written from an external programming interface. The application software can read the fuses. The fuses are used to configure reset sources such as Brown-out Detector and Watchdog, Start-up configuration, JTAG enable and JTAG user ID.

The Lock bits are used to set protection level on the different flash sections. They are used to block read and/or write access of the code. Lock bits can be written from en external programmer and from the application software to set a more strict protection level, but not to set a less strict protection level. Chip erase is the only way to erase the lock bits. The lock bits are erased after the rest of the flash memory is erased.

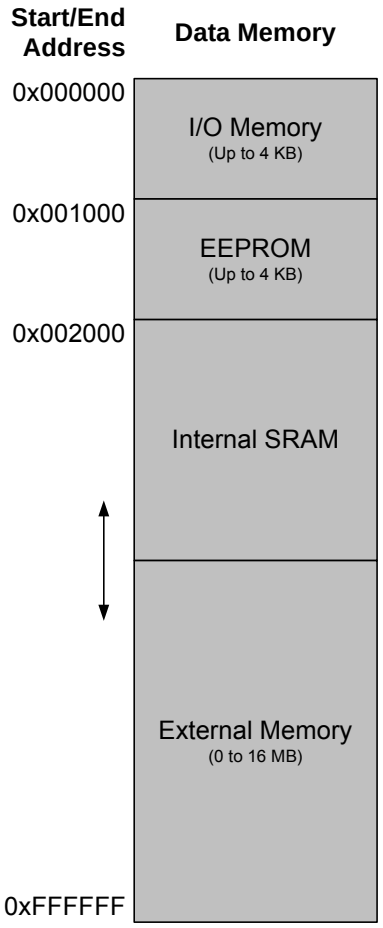
An unprogrammed fuse or lock bit will have the value one, while a programmed flash or lock bit will have the value zero.

Both fuses and lock bits are reprogrammable like the Flash Program memory.

4.5 Data Memory

The Data memory contains the I/O Memory, internal SRAM, optionally memory mapped EEPROM and external memory if available. The data memory is organized as one continuous memory section, as shown in Figure 4-2 on page 22.

Figure 4-2. Data Memory Map



I/O Memory, EEPROM and SRAM will always have the same start addresses for all XMEGA devices. External Memory (if exist) will always start at the end of Internal SRAM and end at address 0xFFFFFF.

4.6 Internal SRAM

The internal SRAM is mapped in the Data Memory space, always starting at hexadecimal address location 0x2000. SRAM is accessed from the CPU by using the load (LD/LDS/LDD) and store (ST/STS/STD) instructions.

4.7 EEPROM

XMEGA has EEPROM memory for non-volatile data storage. It is addressable either in as a separate data space (default), or it can be memory mapped and accessed in normal data space. The EEPROM memory supports both byte and page access.

4.7.1 Data Memory Mapped EEPROM Access

The EEPROM address space can optionally be mapped into the Data Memory space to allow 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 location 0x1000.

4.8 I/O Memory

The status and configuration registers for all peripherals and modules, including the CPU, are addressable through I/O memory locations in the data memory space. All I/O locations can be accessed by the load (LD/LDS/LDD) and store (ST/STS/STD) instructions, transferring data between the 32 general purpose registers in the Register File and the I/O memory. The IN and OUT instructions can address I/O memory locations in the range 0x00 - 0x3F directly. In the address range 0x00 - 0x1F, specific bit manipulating and checking instructions are available. The I/O memory definition for an XMEGA device is shown in "Register Summary" in the device data sheet.

4.8.1 General Purpose I/O Registers

The lowest 16 I/O Memory addresses is reserved for General Purpose I/O Registers. These registers can be used for storing information, and they are particularly useful for storing global variables and flags, as they are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

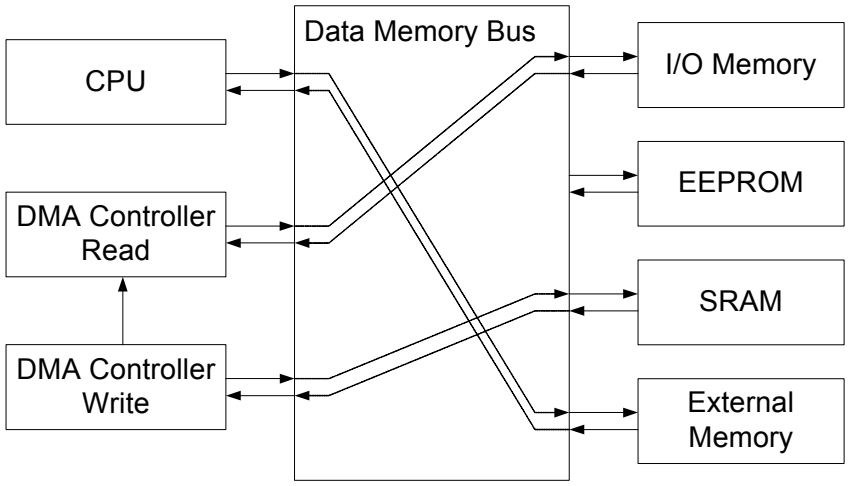
4.9 External Memory

XMEGA has up to 4 ports dedicated to External Memory, supporting external SRAM, SDRAM, and memory mapped peripherals such as LCD displays or other memory mapped devices. For details refer to the External Bus interface (EBI) description. The External Memory address space will always start at the end of Internal SRAM.

4.10 Data Memory and Bus Arbitration

As the Data Memory organized as four separate sets of memories, the different bus masters (CPU, DMA Controller read and DMA Controller write) can access different memories at the same time. As Figure 4-3 on page 24 shows, the CPU can access the External Memory while the DMA (DMA) Controller is transferring data from Internal SRAM to I/O Memory.

Figure 4-3. Bus Access



4.10.1 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
   a. Alternating DMA Controller Read and DMA Controller Write when the they access the same Data Memory section.
3. Bus Master requesting burst access
   a. CPU has priority
4. Bus Master requesting bus access
   a. CPU has priority

4.11 Memory Timing

Read and write access to the I/O Memory takes one CPU clock cycle. Write to SRAM takes one cycle and read from SRAM takes two cycles. For burst read (DMA), new data is available every cycle. EEPROM page load (write) takes one cycle and three cycles are required for read. For burst read, new data is available every second cycle. External memory has multi-cycle read and write. The number of cycles depends on type of memory and configuration of the External Bus Interface. Refer to the instruction summary for more details on instructions and instruction timing.

4.12 Device ID

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

4.13 JTAG Disable

It is possible to disable the JTAG interface from the application software. This will prevent all external JTAG access to the memory, until the next device reset or if JTAG is enabled again

from the application software. As long as JTAG is disabled the I/O pins required for JTAG can be used as normal I/O pins.

4.14 IO Memory Protection

Some features in the device is regarded to be critical for safety in some applications. Due to this, it is possible to lock the IO register related to the Event System and the Advanced Waveform Extensions. As long as the lock is enabled, all related IO 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 12.

4.15 Register Description - NVM Controller

4.15.1 ADDR2 - Non-Volatile Memory Address Register 2

The ADDR2, ADDR1 and ADDR0 registers represents the 24-bit value ADDR.

Bit	7	6	5	4	3	2	1	0	
+0x02	ADDR[23:16]								ADDR2
Read/Write	R/W	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]: NVM Address Register Byte 2**

This register gives the address extended byte when accessing application and boot section.

4.15.2 ADDR1 - Non-Volatile Memory Address Register 1

Bit	7	6	5	4	3	2	1	0	
+0x01	ADDR[15:8]								ADDR1
Read/Write	R/W	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]: NVM Address Register Byte 1**

This register gives the address high byte when accessing either of the memory locations.

4.15.3 ADDR0 - Non-Volatile Memory Address Register 0

Bit	7	6	5	4	3	2	1	0	
+0x00	ADDR[7:0]								ADDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:0 - ADDR[7:0]: NVM Address Register Byte 0**

This register gives the address low byte when accessing either of the memory locations.

4.15.4 DATA2 - Non-Volatile Memory Data Register Byte 2

The DATA2, DATA1 and ADDR0 registers represents the 24-bit value DATA.

Bit	7	6	5	4	3	2	1	0	
+0x06	DATA[23:16]								DATA2
Read/Write	R/W	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]: NVM Data Register 2**

This register gives the data value byte 2 when running CRC check on application section, boot section or combined.

4.15.5 DATA1 - Non-Volatile Memory Data Register 1

Bit	7	6	5	4	3	2	1	0	
+0x05	DATA[15:8]								DATA1
Read/Write	R/W	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]: NVM Data Register Byte 1**

This register gives the data value byte 1 when accessing application and boot section.

4.15.6 DATA0 - Non-Volatile Memory Data Register 0

Bit	7	6	5	4	3	2	1	0	
+0x04	DATA[7:0]								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	

- **Bit 7:0 - DATA[7:0]: NVM Data Register Byte 0**

This register gives the data value byte 0 when accessing either of the memory locations.

4.15.7 CMD - Non-Volatile Memory Command Register

Bit	7	6	5	4	3	2	1	0	
+0x0A	-	CMD[6:0]							CMD
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 - 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]: NVM Command**

These bits define the programming commands for the flash. Bit six is set for external programming commands. See "Memory Programming data sheet" for programming commands.

4.15.8 CTRLA - Non-Volatile Memory Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x0B	-	-	-	-	-	-	-	CMDEX	CTRLA
Read/Write	R	R	R	R	R	R	R	S	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:1 - Reserved Bits**

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: Non-Volatile Memory 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 Section 3.12 ”Configuration Change Protection” on page 12 for details on the CCP.

4.15.9 CTRLB - Non-Volatile Memory Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x0C	-	-	-	-	EEMAPEN	FPRM	EPRM	SPMLOCK	CTRLB
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 will enable 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 will enable power saving for the flash memory. The section not being accessed will be turned off like in sleep mode. If code is running from 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 equally long to the start-up time from the Idle sleep mode.

- **Bit 1 - EPRM: EEPROM Power Reduction Mode**

Setting this bit will enable power saving for the EEPROM memory. The EEPROM will then be powered down equal to entering sleep mode. If access is required, the bus master will be halted equally long as the start-up time from Idle sleep mode.

- **Bit 0 - SPMLOCK: SPM Locked**

The SPM Locked 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 Section 3.12 ”Configuration Change Protection” on page 12 for details on the CCP.

4.15.10 INTCTRL - Non-Volatile Memory Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
+0x0D	-	-	-	-	SPMLVL[1:0]		EELVL[1:0]		INTCTRL
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 Multi-level Interrupt Controller” on page 123. The interrupt is a level interrupt, which will be triggered when the BUSY flag in the STATUS is set to logical 0. Since the interrupt is a level interrupt note the following.

The interrupt should not be enabled before triggering a NVM command, as the BUSY flag wont be set before the NVM command is triggered. Since the interrupt trigger is a level interrupt, 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 Multi-level Interrupt Controller” on page 123. The interrupt is a level interrupt, which will be triggered when the BUSY flag in the STATUS is set to logical 0. Since the interrupt is a level interrupt note the following.

The interrupt should not be enabled before triggering a NVM command, as the BUSY flag wont be set before the NVM command is triggered. Since the interrupt trigger is a level interrupt, the interrupt should be disabled in the interrupt handler.

4.15.11 STATUS - Non-Volatile Memory Status Register

Bit	7	6	5	4	3	2	1	0	
+0x04	BUSY	FBUSY	-	-	-	-	EELOAD	FLOAD	STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - NVMBUSY: Non-Volatile Memory Busy**

The NVMBSY flag indicates whether the NVM memory (FLASH, EEPROM, Lock-bits) is busy being programmed. Once a program operation is started, this flag will be set and it remains set until the program operation is completed. he NVMBSY flag will automatically be cleared when the operation is finished.

- **Bit 6 - FBUSY: Flash Section Busy**

The FBUSY flag indicate whether a Flash operation (Page Erase or Page Write) is initiated. Once a operation is started the FBUSY flag is set, and the Application Section cannot be accessed. The FBUSY bit will automatically be cleared when the operation is finished.

- **Bit 5:2 - Reserved**

These bits are reserved and will always be read as zero. 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 status flag indicates that the temporary EEPROM page buffer has been loaded with one or more data bytes. Immediately after an EEPROM load command is issued and byte is written to NVMDR, or a memory mapped EEPROM buffer load operation is performed, the EELOAD flag is set, and it remains set until an EEPROM page write- or a page buffer flush operation is executed.

- **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. Immediately after a Flash load command has been issues and byte is written to NVMDR, the FLOAD flag is set, and it remains set until an Application- or Boot page write- or a page buffer flush operation is executed.

4.15.12 LOCKBITS - Non-Volatile Memory 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]		LOCKBITS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	1	1	1	1	1	1	1	1	

This register is a direct mapping of the NVM Lockbits into the IO Memory Space, in order to enable direct read access from the application software. Refer to ”LOCKBITS - Non-Volatile Memory Lock Bit Register” on page 34 for description of the Lock Bits.

4.16 Register Description – Fuses and Lockbits

4.16.1 FUSEBYTE0 - Non-Volatile Memory Fuse Byte 0 - JTAG User ID

Bit	7	6	5	4	3	2	1	0	
+0x00	JTAGUID[7:0]								FUSEBYTE0
Read/Write	R/W	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 - JTAGUID[7:0]: JTAG USER ID**

These fuses can be used to set the default JTAG USER ID for the device. During reset, the JTAGUID fuse bits will be loaded into the MCU JTAG User ID Register.

4.16.2 FUSEBYTE1 - Non-Volatile Memory Fuse Byte1 - Watchdog Configuration

Bit	7	6	5	4	3	2	1	0	
+0x01	WDWPER[3:0]				WDPER[3:0]				FUSEBYTE1
Read/Write	R/W	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**

The WDWPER 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 Section 11.7.2 ”WINCTRL – Window Mode Control Register” on page 120 for details.

- **BIT 3:0 - WDPER[3:0]: Watchdog Timeout Period**

The WDPER fuse bits are used to set 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 Section 11.7.1 ”CTRL – Watchdog Timer Control Register” on page 119 for details.

4.16.3 FUSEBYTE2 - Non-Volatile Memory Fuse Byte2 - Reset Configuration

Bit	7	6	5	4	3	2	1	0	
+0x02	–	BOOTRST	–	–	–	–	BODPD[1:0]		FUSEBYTE2
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 fuse bit is reserved. For compatibility with future devices, always write this bit to one when this register is written.

- **Bit 6 - BOOTRST: Boot Loader Section Reset Vector**

The BOOTRST fuse can be programmed so the Reset Vector is pointing to the first address in the Boot Loader Flash Section. In this case, the device will start executing from the from Boot Loader Flash Section after reset.

Table 4-1. Boot Reset Fuse

BOOTRST	Reset Address
0	Reset Vector = Boot Loader Reset
1	Reset Vector = Application Reset (address 0x0000)

- **Bit 5:2 - Reserved**

These fuse bits are reserved. 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**

The BODPD 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 ”Brown-Out Detection” on page 106.

Table 4-2. BOD operation modes in sleep modes

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

4.16.4 FUSEBYTE4 - Non-Volatile Memory fuse Byte4 - Start-up Configuration

Bit	7	6	5	4	3	2	1	0	
+0x04	–	–	–	RSTDISBL	STARTUPTIME[1:0]		WDLOCK	JTAGEN	FUSEBYTE4
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	0	

- **Bit 7:5 - Reserved**

These fuse bits are reserved. 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 this pin low will not cause an external reset.

- **Bit 3:2 - STARTUPTIME[1:0]: Start-up time**

The STARTUPTIME fuse bits can be used to set at a programmable timeout period from all reset sources are released and until the internal reset is released from the delay counter.

The delay is timed from the 1kHz output of the ULP oscillator, refer to Section 9.3 ”Reset Sequence” on page 104 for details.

Table 4-3. Start-up Time

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

- **Bit 1 - WDLOCK: Watchdog Timer lock**

The WDLOCK fuse can be programmed to lock the Watchdog Timer configuration. When this fuse is programmed the Watchdog Timer configuration cannot be changed, and the Watchdog Timer cannot be disabled from the application software. When this fuse is programmed the ENABLE bit in the watchdog CTRL register is automatically set at reset. The WEN bit in the watchdog WINCTRL register is not set automatically and needs to be enabled from software.

Table 4-4. Watchdog Timer locking

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

- **Bit 0 - JTAGEN: JTAG enabled**

The JTAGEN fuse decides whether or not the JTAG interface is enabled.

When the JTAG interface is disabled all access through JTAG is prohibited, and the device can only be accessed using the Program and Debug Interface (PDI).

Table 4-5. JTAG Enable

JTAGEN	Description
0	JTAG enabled
1	JTAG disabled

The JTAGEN fuse is only available on devices with JTAG interface.

4.16.5 FUSEBYTE5 - Non-Volatile Memory Fuse Byte 5

Bit	7	6	5	4	3	2	1	0	
+0x05	-	-	BODACT[1:0]		EESAVE	BODLEVEL[2:0]			FUSEBYTE5
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	1	1	-	-	-	-	-	-	

- **Bit 7:6 - Reserved**

These bits are reserved. 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**

The BODACT fuse bits set the BOD operation mode when the device is in active and idle mode of operation. For details on the BOD and BOD operation modes refer to Section 9.4.2 ”Brown-Out Detection” on page 106.

Table 4-6. BOD operation modes in Active and Idle mode

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

- **Bit 3 - EESAVE: EEPROM memory is preserved through the Chip Erase**

A chip erase command will normally erase the Flash, EEPROM and internal SRAM. If the EESAVE fuse is programmed, the EEPROM is not erased during chip erase. In case EEPROM is used to store data independent of software revision, the EEPROM can be preserved through chip erase.

Table 4-7. EEPROM memory through Chip Erase

EESAVE	Description
0	EEPROM is preserved during chip erase
1	EEPROM is not preserved during chip erase

Changing of the EESAVE fuse bit takes effect immediately after the write time-out elapses. Hence, it is possible to update EESAVE and perform a chip erase according to the new setting of EESAVE without leaving and re-entering programming mode

- **Bit 2:0 - BODLEVEL[2:0] - Brown out detection voltage level**

The BODLEVEL fuse bits sets the nominal BOD level value. During power-on the device is kept in reset until the V_{CC} level has reached the programmed BOD level. Due to this always ensure that the BOD level is set lower than the V_{CC} level, also if the BOD is not enabled and used during normal operation, refer to Section 9.4 ”Reset Sources” on page 104 for details. For BOD level nominal values, see Table 9-2 on page 106.

4.16.6 LOCKBITS - Non-Volatile Memory 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]**		**LOCKBITS**
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 bits indicate the locking mode for the Boot Loader Section. Even though the BLBB bits are writable, they can only be written to a stricter locking. Resetting the BLBB bits is only 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 bits indicate the locking mode for the Application Section. Even though the BLBA bits are writable, they can only be written to a stricter locking. Resetting the BLBA bits is only possible 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 them 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 bits indicate the locking mode for the Application Table Section. Even though the BLBAT bits are writable, they can only be written to a stricter locking. Resetting the BLBAT bits is only possible 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**

These bits indicate the locking mode for the Flash and EEPROM in Programming Mode. These bits are writable only through an external programming interface. Resetting the Lock Bits is only possible by executing a Chip Erase Command.

Table 4-11. Boot Lock Bit for The Boot Section

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 is disabled for the programming interface. The lock bits and fuses are locked for read and write from the programming interface.

4.17 Register Description - Production Signature Row

4.17.1 RCOSC2M - Internal 2 MHz Oscillator Calibration Register

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

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

4.17.2 RCOSC32K - Internal 32.768 kHz Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
+0x01	RCOSC32K[7:0]								RCOSC32K
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 kHz Oscillator Calibration Value**

This byte contains the oscillator calibration value for the internal 32.768 kHz 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 32.768 kHz oscillator, refer to ”RC32KCAL - 32 KHz Oscillator Calibration Register” on page 90 for more details.

4.17.3 RCOSC32M - Internal 32 MHz RC Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
+0x02	RCOSC32M[7:0]								RCOSC32M
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 32 MHz Oscillator Calibration Value**

This byte contains the oscillator calibration value for the internal 32 MHz oscillator. Calibration of the oscillator is performed during production test of the device. During reset this value is automatically loaded into the Calibration Register B for the 32 MHz DFLL, refer to ”CALB - Calibration Register B” on page 92 for more details.

4.17.4 LOTNUM0 - Lot Number Register 0

LOTNUM0, LOTNUM1, LOTNUM2, LOTNUM3, LOTNUM4 and LOTNUM5 contains the LOT number for each device. Together with the wafer number and wafer coordinates this gives an unique identifier or serial number for the device.

Bit	7	6	5	4	3	2	1	0	
+0x07	LOTNUM0[7:0]								LOTNUM0
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.

4.17.5 LOTNUM1 - Lot Number Register 1

Bit	7	6	5	4	3	2	1	0	
+0x07	LOTNUM1[7:0]								LOTNUM1
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.

4.17.6 LOTNUM2 - Lot Number Register 2

Bit	7	6	5	4	3	2	1	0	
+0x08	LOTNUM2[7:0]								LOTNUM2
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.

4.17.7 LOTNUM3- Lot Number Register 3

Bit	7	6	5	4	3	2	1	0	
+0x09	LOTNUM3[7:0]								LOTNUM3
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.

4.17.8 LOTNUM4 - Lot Number Register 4

Bit	7	6	5	4	3	2	1	0	
+0x0A	LOTNUM4[7:0]								LOTNUM4
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.

4.17.9 LOTNUM5 - Lot Number Register 5

Bit	7	6	5	4	3	2	1	0	
+0x0B	LOTNUM5[7:0]								LOTNUM5
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 5of the LOT number for the device.

4.17.10 WAFNUM - Wafer Number Register

Bit	7	6	5	4	3	2	1	0	
+0x10	WAFNUM[7:0]								WAFNUM
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 an unique identifier or serial number for the device.

4.17.11 COORDX0 - Wafer Coordinate X Register 0

COORDX0, COORDX1, COORDY0 and COORDY1 contains the wafer X and Y coordinates for each device. Together with the LOT number and wafer number this gives an unique identifier er or serial number for each devicei

Bit	7	6	5	4	3	2	1	0	
+0x12	COORDX0[7:0]								COORDX0
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.

4.17.12 COORDX1 - Wafer Coordinate X Register 1

Bit	7	6	5	4	3	2	1	0	
+0x13	COORDX1[7:0]								COORDX1
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.

4.17.13 COORDY0 - Wafer Coordinate Y Register 0

Bit	7	6	5	4	3	2	1	0	
+0x14	COORDY0[7:0]								COORDY0
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.

4.17.14 COORDY1 - Wafer Coordinate Y Register 1

Bit	7	6	5	4	3	2	1	0	
+0x15	COORDY1[7:0]								COORDY1
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

4.17.15 ADCACAL0 - ADCA Calibration Register 0

ADCACAL0 and ADCACAL1 contains the calibration value for the Analog to Digital Converter A (ADCA). Calibration of the Analog to Digital Converters are done during production test of the device. The calibration bytes are not loaded automatically into the ADC calibration registers, and this must be done from software..

Bit	7	6	5	4	3	2	1	0	
+0x20	ADCACAL0[7:0]								ADCACAL0
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.

4.17.16 ADCACAL1 - ADCA Calibration Register 1

Bit	7	6	5	4	3	2	1	0	
+0x21	ADCACAL1[7:0]								ADCACAL1
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.

4.17.17 ADCBCAL0 - ADCB Calibration Register 0

ADCBCAL0 and ADCBCAL1 contains the calibration value for the Analog to Digital Converter B(ADCB). Calibration of the Analog to Digital Converters are done during production test of the device. The calibration bytes not loaded automatically into the ADC calibration registers, and this must be done from software.

Bit	7	6	5	4	3	2	1	0	
+0x20	ADCBCAL0[7:0]								ADCBCAL0
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	x	x	x	x	x	x	x	x	

- **Bit 7:0 - ADCBCAL0[7:0] - ADCB Calibration Byte 0**

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

4.17.18 ADCBCAL1 - ADCB Calibration Register 1

Bit	7	6	5	4	3	2	1	0	
+0x21	ADCBCAL1[7:0]								ADCBCAL1
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	x	x	x	x	x	x	x	x	

- **Bit 7:0 - ADCBCAL0[7:0] - ADCB Calibration Byte 1**

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

4.17.19 TEMPSENSE0 - Temperature Sensor Calibration Register 0

TEMPSENSE0 and TEMPSENSE1 contains the 12-bit ADCA value from a temperature measurements done with the internal temperature sensor. The measurements is done in production test at 85C 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]								TEMPSENSE0
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 (8 LSB) of the temperature measurement.

4.17.20 TEMPSENSE1 - Temperature Sensor Calibration Register 1

Bit	7	6	5	4	3	2	1	0	
+0x2F	TEMPSENSE1[7:0]								TEMPSENSE1
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.

4.17.21 DACAGAINCAL - DACA Gain Calibration Register

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

- **Bit 7:0 - DACAGAINCAL[7:0] - DACA Gain Calibration Byte**

This byte contains the gain calibration value for the Digital to Analog Converter A (DACA). Calibration of the Digital to Analog Converters are done during production test of the device. The calibration byte is not loaded automatically into the DAC Gain Calibration register, and this must be done from software.

4.17.22 DACAOFFCAL - DACA Offset Calibration Register

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

- **Bit 7:0 - DACAOFFCAL[7:0] - DACA Offset Calibration Byte**

This byte contains the offset calibration value for the Digital to Analog Converter A (DACA). Calibration of the Digital to Analog Converters are done during production test of the device. The calibration byte is not loaded automatically into the DAC Offset Calibration register, and this must be done from software.

4.17.23 DACBGAINCAL - DACB Gain Calibration Register

Bit	7	6	5	4	3	2	1	0	
+0x32	DACBGAINCAL[7:0]								DACBGAINCAL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	x	x	x	x	

- **Bit 7:0 - DACAGAINCAL[7:0] - DACB Gain Calibration Byte**

This byte contains the gain calibration value for the Digital to Analog Converter B (DACB). Calibration of the Digital to Analog Converters are done during production test of the device. The calibration byte is not loaded automatically into the DAC Gain Calibration register, and this must be done from software.

4.17.24 DACBOFFCAL - DACB Offset Calibration Register

Bit	7	6	5	4	3	2	1	0	
+0x33	DACBOFFCAL[7:0]								DACAOFFCAL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	x	x	x	x	

- **Bit 7:0 - DACBOFFCAL[7:0] - DACB Offset Calibration Byte**

This byte contains the offset calibration value for the Digital to Analog Converter B (DACB). Calibration of the Digital to Analog Converters are done during production test of the device. The calibration byte is not loaded automatically into the DAC Offset Calibration register, and this must be done from software.

4.18 Register Description – General Purpose I/O Memory

4.18.1 GPIORn – General Purpose I/O Register n

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

This is a general purpose register that can be used to store data such as global variables in the bit accessible I/O memory space.

4.19 Register Description – External Memory

Refer to ”EBI - External Bus Interface” on page 268.

4.20 Register Description – MCU Control

4.20.1 DEVID0 - MCU Device ID Register 0

The DEVID0, DEVID1 and DEVID2 contains the 3-byte identification that identify each microcontroller device type. For details on the actual ID refer to the device data sheet.

Bit	7	6	5	4	3	2	1	0	
+0x00	DEVID0[7:0]								DEVID0
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 - DEVID0[7:0]: MCU Device ID Byte 1**

This byte will always be read as 0x1E. This indicates that the device is manufactured by Atmel

4.20.2 DEVID1 - MCU Device ID Register 1

Bit	7	6	5	4	3	2	1	0	
+0x01	DEVID1[7:0]								DEVID1
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]: MCU Device ID Byte 1**

Byte 1 of the device ID indicates the flash size of the device.

4.20.3 DEVID2 - MCU Device ID Register 2

Bit	7	6	5	4	3	2	1	0	
+0x02	DEVID2[7:0]								DEVID2
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]: MCU Device ID Byte 2**

Byte 0 of the device ID indicates the device number.

4.20.4 REVID - MCU Revision ID

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	REVID[3:0]				REVID
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 7:4 - Reserved**

These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 3:0 - REVID[3:0]: MCU Revision ID**

These bits contains the device revision. 0=A, 1=B and so on.

4.20.5 JTAGUID – JTAG User ID Register

Bit	7	6	5	4	3	2	1	0	
+0x04	JTAGUID[7:0]								JTAGUID
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 - JTAGUID[7:0]: JTAG User ID**

The JTAGUID can be used to identify two devices with identical Device ID in a JTAG scan chain. The JTAGUID will during reset automatically be loaded from flash and placed in these registers.

4.20.6 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
+0x06	-	-	-	-	-	-	-	JTAGD	MCUCR
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 - JTAGD: JTAG Disable**

Setting this bit will disable the JTAG interface. This bit is protected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

4.20.7 EVSYSLOCK – Event System Lock Register

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

- **Bit 7:5 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 4 - EVSYS1LOCK:**

Setting this bit will lock all registers in the Event System related to event channels 4 to 7 for further modifications. The following registers in the Event System are locked: CH4MUX, CH4CTRL, CH5MUX, CH5CTRL, CH6MUX, CH6CTRL, CH7MUX, CH7CTRL. This bit is protected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

- **Bit 3:1 - Reserved**

These bits are reserved and will always be read as zero. 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 for further modifications. The following registers in the Event System are locked: CH0MUX, CH0CTRL, CH1MUX, CH1CTRL, CH2MUX, CH2CTRL, CH3MUX, CH3CTRL. This bit is pro-

tected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

4.20.8 AWEXLOCK – Advanced Waveform Extension Lock Register

Bit	7	6	5	4	3	2	1	0	
+0x09	—	—	—	—	—	AWEXELOCK	—	AWEXCLOCK	AWEX_LOCK
Read/Write	R	R	R	R	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 2 - AWEXELOCK: Advanced Waveform Extension Lock for TCE0**

Setting this bit will lock all registers in the AWEXE module for Timer/Counter E0 for further modifications. This bit is protected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

- **Bit 1 - 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 0 - AWEXCLOCK: Advanced Waveform Extension Lock for TCC0**

Setting this bit will lock all registers in the AWEXC module for Timer/Counter C0 for further modifications. This bit is protected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

4.21 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	NVM Address Byte 0								25
+0x01	ADDR1	NVM Address Byte 1								25
+0x02	ADDR2	NVM Address Byte 2								25
+0x03	Reserved									
+0x04	DATA0	NVM Data Byte 0								26
+0x05	DATA1	NVM Data Byte 1								26
+0x06	DATA2	NVM 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

4.22 Register Summary - Fuses and Lockbits

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	FUSEBYTE0	JTAGUID								30
+0x01	FUSEBYTE1	WDWPER3:0]				WDPER[3:0]				30
+0x02	FUSEBYTE2		BOOTRST					BODPD[1:0]		30
+0x03	Reserved									
+0x04	FUSEBYTE4				RSTDISBL	STARTUPTIME[1:0]		WDLOCK	JTAGEN	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

4.23 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		Reserved									
+0x02	YES	RCOSC32K	RCOSC32K[7:0]								36
+0x03	YES	RCOSC32M	RCOSC32M[7:0]								37
+0x04		Reserved									
+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		Reserved									
+0x1B		Reserved									
+0x1C		Reserved									
+0x1D		Reserved									
+0x0E		Reserved									
+0x1E		Reserved									
+0x20	NO	ADCACAL0	ADCACAL0[7:0]								39
+0x21	NO	ADCACAL1	ADCACAL1{7:0]								40
+0x22		Reserved									
+0x23		Reserved									
+0x24	NO	ADCBCAL0	ADCBCAL0[7:0]								40
+0x25	NO	ADCBCAL1	ADCBCAL1[7:0]								40
+0x26		Reserved									
+0x27		Reserved									
+0x28		Reserved									
+0x29		Reserved									
+0x2A		Reserved									
+0x2B		Reserved									
+0x2C		Reserved									
+0x2D		Reserved									
+0x2E	NO	TEMPSENSE0	TEMPSENSE[7:0]								41
+0x2F	NO	TEMPSENSE1					TEMPSENE[11:8]				41
+0x30	NO	DACAOFFCAL	DACAOFFCAL[7:0]								41
+0x31	NO	DACACAINCAL	DACACAINCAL[7:0]								41
+0x32	NO	DACBOFFCAL	DACBOFFCAL[7:0]								42
+0x33	NO	DACBCAINCAL	DACBCAINCAL[7:0]								42
+0x34		Reserved									
+0x35		Reserved									
+0x36		Reserved									
+0x37		Reserved									
+0x38		Reserved									
+0x39		Reserved									
0x3A		Reserved									
+0x3B		Reserved									
+0x3C		Reserved									
+0x3D		Reserved									
+0x3E		Reserved									

4.24 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
+0x04	GPIOR4	GPIOR[7:0]								42
+0x05	GPIOR5	GPIOR[7:0]								42
+0x06	GPIOR6	GPIOR[7:0]								42
+0x07	GPIOR7	GPIOR[7:0]								42
+0x08	GPIOR8	GPIOR[7:0]								42
+0x09	GPIOR9	GPIOR[7:0]								42
+0x0A	GPIOR10	GPIOR[7:0]								42
+0x0B	GPIOR11	GPIOR[7:0]								42
+0x0C	GPIOR12	GPIOR[7:0]								42
+0x0D	GPIOR13	GPIOR[7:0]								42
+0x0E	GPIOR14	GPIOR[7:0]								42
+0x0F	GPIOR15	GPIOR[7:0]								42

4.25 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]								43
+0x01	DEVID1	DEVID1[7:0]								43
+0x02	DEVID2	DEVID2[7:0]								43
+0x03	REVID	–	–	–	–	REVID[3:0]				43
+0x04	JTAGUID	JTAGUID[7:0]								44
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	MCUCR	–	–	–	–	–	–	–	JTAGD	48
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	EVSYSLOCK	–	–	–	EVSYS1LOC	–	–	–	EVSYS0LOCK	
+0x09	AWEXLOCK	–	–	–	–	–	AWEXELOCK	–	AWEXCLOCK	
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	

4.26 Interrupt Vector Summary - NVM Controller

Table 4-12. NVM Interrupt vectors and their word offset address from the NVM Controller interrupt base

Offset	Source	Interrupt Description
0x02	SPM_vect	Non-Volatile Memory SPM Interrupt vector
0x00	EE_vect	Non-Volatile Memory EEPROM Interrupt vector

5. DMA - Direct Memory Access Controller

5.1 Features

- The DMA Controller allows high-speed transfers with minimal CPU intervention
  - from one memory area to another
  - from memory area to peripheral
  - from peripheral to memory area
  - from peripheral to another peripheral
- Four DMA Channels with separate
  - transfer triggers
  - interrupt vectors
  - addressing modes
- From 1 byte to 16M bytes data transfer in a single transaction
- Up to 64 KByte block transfers with repeat
- 1, 2, 4, or 8 byte burst transfers
- Internal and external transfer triggers
- Multiple addressing modes
  - Static
  - Increment
  - Decrement
- Optional reload of source and destination address at the end of each
  - Burst
  - Block
  - Transaction
- Optional Interrupt on end of transaction
- Programmable channel priority

5.2 Overview

The XMEGA Direct Memory Access (DMA) Controller is a highly flexible DMA Controller capable of transferring data between memories and peripherals with minimal CPU intervention. The DMA controller has flexible channel priority selection, several addressing modes, double buffering capabilities and large block sizes.

The DMA Controller can move data between memories and peripherals, between memories and between peripheral registers directly.

There are four DMA channels that have individual source, destination, triggers and block sizes. The different channels also have individual control settings and individual interrupt settings and interrupt vectors. Interrupt requests may be generated both when a transaction is complete or if the DMA Controller detects an error on a DMA channel. When a DMA channel requests a data transfer, the bus arbiter will wait until the AVR CPU is not using the data bus and permit the DMA Controller to transfer data. Transfers are done in bursts of 1, 2, 4 or 8 bytes. Addressing can be static, incremental or decremental. Automatic reload of source and/or destination address can be done after each burst transfer, block transfer, when transmission is complete, or disabled. Both application software, peripherals and Events can trigger DMA transfers.

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 64 KBytes.

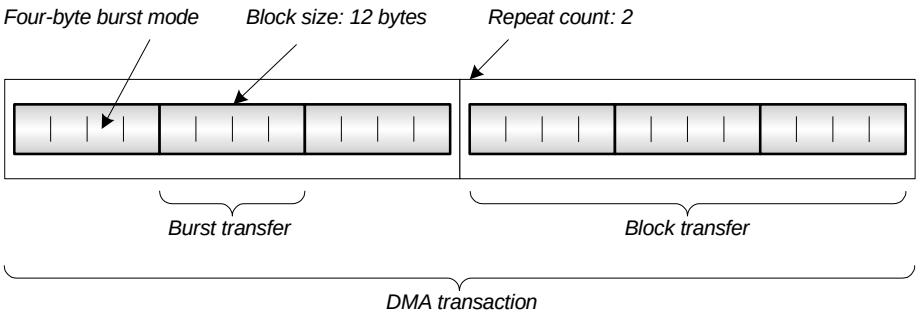
A repeat counter can be enabled to set a number of repeated block transfers before a transaction is complete. The repeat is from 1 to 255 and unlimited repeat count can be achieved by setting the repeat count to zero.

5.3.2 Burst Transfer

As 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 a data bus and a transfer request is pending it will occupy the bus until all bytes in the burst transfer is transferred.

A bus arbiter controls when the DMA controller and the AVR CPU can use the bus. The CPU always has priority, so as long as the CPU request access to the bus, any pending burst transfer must wait. The CPU requests bus access when it executes an instruction that write or read data to SRAM, I/O memory, EEPROM and the External Bus Interface. For more details on memory access bus arbitration, refer to ”Data Memory” on page 22.

Figure 5-1. DMA transaction.



5.4 Transfer Triggers

DMA transfers can only be started 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 - DMA Channel Trigger Source” on page 58.

By default, a trigger starts a block transfer operation. The transfer continues until one block is transferred. When the block is transferred, the channel will wait for the next trigger to arrive before it start transferring the next block. It is possible to select the trigger to start a burst transfer instead of a block transfer. This is called a single shot transfer. A new trigger will then start a new burst transfer. When repeat mode is enabled, the start of transfer of the next block 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, 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, incremental or decremental 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 address is stored by the DMA controller, 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 reload

5.6 Priority Between Channels

If several channels request 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 is called double buffering. When a transmission is completed for the first channel, the second channel is enabled. When a request is detected on the second channel, the transfer starts and when this is completed the first channel is enabled again.

5.8 Transfer Buffers

Each DMA channel has an internal transfer buffer that is used for 2, 4 and 8 byte burst transfers. When a transfer is triggered, a DMA channel will wait until the transfer buffer contains two bytes before the transfer starts. For 4 or 8 byte transfer, any remaining bytes is transferred as soon as they are ready for a DMA channel. The buffer is used to reduce the time the DMA controller occupy the bus. When the DMA controller or a DMA channel is disabled from software, any remaining bytes in the buffer will be transferred before the DMA controller or DMA channel is disabled. This ensures that the source and destination address registers are kept synchronized.

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 memory locations.
- Reading EEPROM memory 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 is cleared. A software reset can only be done 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 only be done when the DMA channel is disabled.

5.11 Protection

In order to insure 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 only modify these registers and bits:

- CTRL register
- INTFLAGS register
- TEMP registers
- CHEN, CHRST, TRFREQ, 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 - DMA Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	ENABLE	RESET	-	-	DBUFMODE[1:0]		PRIMODE[1:0]		CTRL
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 - ENABLE: DMA 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: DMA Software Reset**

Setting this bit enables the software reset. This bit is automatically cleared when reset is completed. This bit can only be set when the DMA Controller is disabled (ENABLE = 0).

- **Bit 5:4 - 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 - DBUFMODE[1:0]: DMA Double Buffer Mode**

These bits enables the double buffer on the different channels according to Table 5-1.

Table 5-1. DMA Double Buffer settings

DBUFMODE[1:0]	Group Configuration	Description
00	DISABLED	No double buffer enabled
01	CH01	Double buffer enabled on channel0/1
10	CH23	Double buffer enabled on channel2/3
11	CH01CH23	Double buffer enabled on channel0/1 and channel2/3

- **Bit 1:0 - PRIMODE[1:0]: DMA Channel Priority Mode**

These bits determine the internal channel priority according to Table 5-2.

Table 5-2. DMA Channel Priority settings

PRIMODE[1:0]	Group Configuration	Description
00	RR0123	Round Robin
01	CH0RR123	Channel0 > Round Robin (Channel 1, 2 and 3)
10	CH01RR23	Channel0 > Channel1 > Round Robin (Channel 2 and 3)
11	CH0123	Channel0 > Channel1 > Channel2 > Channel3

5.13.2 INTFLAGS - DMA Interrupt Status Register

Bit	7	6	5	4	3	2	1	0	
+0x04	CH3ERRIF	CH2ERRIF	CH1ERRIF	CH0ERRIF	CH3TRNFIF	CH2TRNFIF	CH1TRNFIF	CH0TRNFIF	INTFLAGS
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:4 - CHnERRIF[3:0]: DMA 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 3:0 - CHnTRNFIF[3:0]: DMA Channel n Transaction Complete Interrupt Flag**

When a transaction on channel n has been completed, the CHnTRFIF flag will 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 - DMA Status Register

Bit	7	6	5	4	3	2	1	0	
+0x05	CH3BUSY	CH2BUSY	CH1BUSY	CH0BUSY	CH3PEND	CH2PEND	CH1PEND	CH0PEND	STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:4 - CHnBUSY[3:0]: DMA 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 3:0 - CHnPEND[3:0]: DMA 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 TEMPH - DMA Temporary Register High

Bit	7	6	5	4	3	2	1	0	
+0x07	DMTEMP[15:8]								TEMPH
Read/Write	R/W	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]: DMA 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 register requires special attention, for details refer to Section 3.11 ”Accessing 16-bits Registers” on page 12.

5.13.5 TEMPL - DMA Temporary Register Low

Bit	7	6	5	4	3	2	1	0	
+0x06	DMTEMP[7:0]								TEMPL
Read/Write	R/W	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]: DMA Temporary Register 0**

This register is used when reading 24- and 16-bit registers in the DMA controller. Byte 1 of the 16/24-bit registers is stored here when it is written by the CPU. Byte 1 of the 16/24-bit registers are 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. Please see detailed description in the AVR Core manual.

5.14 Register Description – DMA Channel

5.14.1 CTRLA - DMA Channel Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	CHEN	CHRST	REPEAT	TRFREQ	-	SINGLE	BURSTLEN[1:0]		CTRLA
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- CHEN: DMA 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 before the internal transfer buffer is empty and the DMA transfer is aborted.

- **Bit 6 - CHRST: DMA Channel Software Reset**

Setting this bit enables the channel reset. This bit is automatically cleared when reset is completed. This bit can only be set when the DMA channel is disabled (CHEN = 0).

- **Bit 5 - REPEAT: DMA Channel Repeat Mode**

Setting this bit enables the repeat mode. In repeat mode, this bit is cleared by hardware in the beginning of the last block transfer. The REPCNT register should be configured before setting the REPEAT bit.

- **Bit 4 - TRFREQ: DMA Channel 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: DMA Channel 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. This bit can not be changed if the channel is busy.

- **Bit 1:0 - BURSTLEN[1:0]: DMA Channel Burst Mode**

These bits decide the DMA channel burst mode according to Table 5-3 on page 56. These bits can not be changed if the channel is busy.

Table 5-3. 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

5.14.2 CTRLB - DMA Channel Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x04	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]		CTRLB
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 - DMA 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 if the Channel Error Interrupt flag is set.

- **Bit 6 - CHPEND - DMA 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 - DMA Channel Error Interrupt Flag**

If an error condition is detected the DMA channel the ERRIF flag will be set, and the optional interrupt is generated. Since the DMA Channel Error Interrupt share interrupt address with DMA Channel Transaction Complete, the ERRIF will not be cleared when the interrupt vector is executed. This flag is cleared by writing a one to the bit location.

- **Bit 4 - TRNIF - DMA Channel n Transaction Complete Interrupt Flag**

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

Since the DMA Channel Transaction Complete Channel Error Interrupt share interrupt address with DMA Channel Error Interrupt, the TRNIF will not be cleared when the interrupt vector is executed. This flag is cleared by writing a one to the bit location.

- **Bit [3:2] - ERRINTLVL[1:0]: DMA Channel Error Interrupt Level**

These bits enable the interrupt for DMA channel transfer error select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger for the conditions when the ERRIF is set.

- **Bit [1:0] - TRNINTLVL[1:0]: DMA Channel Transaction Complete Interrupt Level**

These bits enable the interrupt for DMA channel transaction complete and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger for the conditions when the TRNIF is set.

5.14.3 ADDRCTRL - DMA Channel 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]		ADDRCTRL
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 - SRCRELOAD[1:0]: DMA Channel Source Address Reload**

These bits decide the DMA channel source address reload according to Table 5-4. These bits can not be changed if 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]: DMA Channel Source Address Mode**

These bits decide the DMA channel source address mode according to Table 5-5. These bits can not 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]: DMA Channel Destination Address Reload**

These bits decide the DMA channel destination address reload according to Table 5-6 on page 58. These bits can not 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.

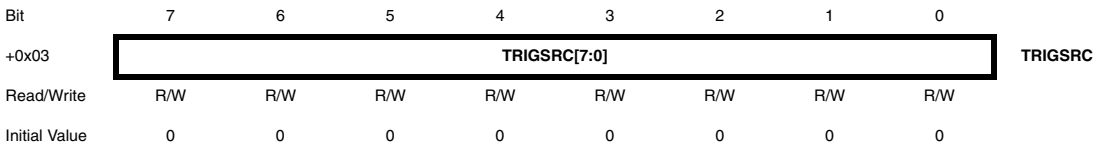
- **Bit 1:0 - DESTDIR[1:0]: DMA Channel Destination Address Mode**

These bits decide the DMA channel destination address mode according to Table 5-7 on page 58. These bits can not 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 - DMA Channel Trigger Source



- **Bit 7:0 - TRIGSRC[7:0]: DMA 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 or peripheral's base value, and the offset value for the trigger source in the module or peripherals. Table 5-8 on page 59 shows the base value for all module and peripherals. Table 5-9 on page 59 to Table 5-12 on page 60 shows the offset value for the trigger sources in the different modules and peripheral types. For modules or peripherals which does not exist for a device, the transfer trigger does not exist. Refer to the device data sheet for the list of peripherals available.

Table 5-8. DMA Trigger Sources, base value for all modules and peripherals

TRIGSRC base value	Group Configuration	Description
0x00	OFF	Software triggers only
0x01	SYS	System DMA triggers base value
0x10	ADCA	ADCA DMA triggers base value
0x15	DACA	DACA DMA trigger bas
0x20	ADCB	ADCB DMA triggers base value
0x25	DACB	DACB DMA triggers base value
0x40	TCC0	Timer/Counter C0 DMA triggers base value
0x46	TCC1	Timer/Counter C1 triggers base value
0x4A	SPIC	SPI C DMA triggers value
0x4B	USARTC0	USART C0 DMA triggers base value
0x4E	USARTC1	USART C1 DMA triggers base value
0x60	TCD0	Timer/Counter D0 DMA triggers base value
0x66	TCD1	Timer/Counter D1 triggers base value
0x6A	SPID	SPI D DMA triggers value
0x6B	USARTD0	USART D0 DMA triggers base value
0x6E	USARTD1	USART D1 DMA triggers base value
0x80	TCE0	Timer/Counter E0 DMA triggers base value
0x86	TCE1	Timer/Counter E1 triggers base value
0x8A	SPIE	SPI E DMA triggers value
0x8B	USARTE0	USART E0 DMA triggers base value
0x8E	USARTE1	USART E1 DMA triggers base value
0xA0	TCF0	Timer/Counter F0 DMA triggers base value
0xA6	TCF1	Timer/Counter F1 triggers base value
0xAA	SPIF	SPI F DMA trigger value
0xAB	USARTF0	USART F0 DMA triggers base value
0xAE	USARTF1	USART F1 DMA triggers base value

Table 5-9. DMA Trigger sources, 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 sources, offset values for DAC and ADC triggers

TRGSRC offset value	Group Configuration	Description
+0x00	CH0	ADC/DAC Channel 0
+0x01	CH1	ADC/DAC Channel 1
+0x02	CH2(1)	ADC Channel 2
+0x03	CH3	ADC Channel 3
+0x04	CH4(2)	ADC Channel 0, 1, 2, 3

Notes: 1. For DAC only Channel 0 and 1 exists and can be used as triggers
2. Channel 4 equals ADC Channel 0 to 3 OR'ed together.

Table 5-11. DMA Trigger sources, 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	Compare or Capture Channel D

Note: 1. CC Channel C and D triggers are only available for Timer/Counter 0.

Table 5-12. DMA Trigger sources, 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 TRFCNTH - DMA Channel Block Transfer Count Register H

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.

Reading and writing 16-bit values requires special attention, for details refer to Section 3.11 ”Accessing 16-bits Registers” on page 12.

Bit	7	6	5	4	3	2	1	0	
+0x05	TRFCNT[15:8]								TRFCNTH
Read/Write	R/W	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]: DMA Channel n Block Transfer Count Register High byte**

These bits hold the 8 MSB of the 16-bits block transfer count.

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

5.14.6 TRFCNTL - DMA Channel Block Transfer Count Register L

Bit	7	6	5	4	3	2	1	0	
+0x04	TRFCNT[7:0]								TRFCNTL
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 - TRFCNT[7:0]: DMA Channel n Block Transfer Count Register Low byte**

These bits hold the 8 LSB of the 16-bits block transfer count.

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

5.14.7 REPCNT - DMA Channel Repeat Counter Register

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

REPCNTcounts 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 - DMA Channel Address Control Register” on page 57), 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 SRCADDR2 - DMA Channel Source Address 2

SRCADDR0, SRCADDR1 and SRCADDR2 represents 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 - DMA Channel Address Control Register” on page 57.

Reading and writing 24-bit values require special attention, for details refer to Section 3.11.1 ”Accessing 24- and 32-bit Registers” on page 12.

Bit	7	6	5	4	3	2	1	0	
+0x0A	SRCADDR[23:16]								SRCADDR2
Read/Write	R/W	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]: DMA Channel Source Address 2**

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

5.14.9 SRCADDR1 - DMA Channel Source Address 1

Bit	7	6	5	4	3	2	1	0	
+0x09	SRCADDR[15:8]								SRCADDR1
Read/Write	R/W	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]: DMA Channel Source Address 1**

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

5.14.10 SRCADDR0 - DMA Channel Source Address 0

Bit	7	6	5	4	3	2	1	0	
+0x08	SRCADDR[7:0]								SRCADDR0
Read/Write	R/W	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]: DMA Channel Source Address 0**

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

5.14.11 DESTADDR2 - DMA Channel Destination Address 2

DESTADDR0, DESTADDR1 and DESTADDR2 represents 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 - DMA Channel Address Control Register” on page 57.

Reading and writing 24-bit values require special attention, for details refer to Section 3.11.1 ”Accessing 24- and 32-bit Registers” on page 12.

Bit	7	6	5	4	3	2	1	0	
+0x0E	DESTADDR[23:16]								DESTADDR2
Read/Write	R/W	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]: DMA Channel Destination Address 2**

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

5.14.12 DESTADDR1 - DMA Channel Destination Address 1

Bit	7	6	5	4	3	2	1	0	
+0x0D	DESTADDR[15:8]								DESTADDR1
Read/Write	R/W	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]: DMA Channel Destination Address 1**

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

5.14.13 DESTADDR0 - DMA Channel Destination Address 0

Bit	7	6	5	4	3	2	1	0	
+0x0C	DESTADDR[7:0]								DESTADDR0
Read/Write	R/W	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]: DMA Channel Destination Address 0**

These bits hold byte 0 of the 24-bits 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[1:0]		PRIMODE[1:0]		53
+0x01	Reserved	-	-	-	-	-	-	-	-	
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	INTFLAGS	CH3ERRIF	CH2ERRIF	CH1ERRIF	CH0ERRIF	CH3TRNFIF	CH2TRNFIF	CH1TRNFIF	CH0TRNFIF	54
+0x04	STATUS	CH3BUSY	CH2BUSY	CH1BUSY	CH0BUSY	CH3PEND	CH2PEND	CH1PEND	CH0PEND	54
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	TEMPL	TEMP[7:0]								55
+0x07	TEMPH	TEMP[15:8]								54
+0x10	CH0 Offset	Offset address for DMA Channel 0								
+0x20	CH1 Offset	Offset address for DMA Channel 0								
+0x30	CH2 Offset	Offset address for DMA Channel 0								
+0x40	CH3 Offset	Offset address for DMA Channel 0								

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	CHEN	CHRST	REPEAT	TRFREQ	-	SINGLE	BURSTLEN		55
+0x01	CTRLB	CHBUSY	CHPEND	ERRIF	TRNIF	ERRINTLVL[1:0]		TRNINTLVL[1:0]		56
+0x02	ADDCTRL	SRCRELOAD[1:0]		SRCDIR[1:0]		DESTRELOAD[1:0]		DESTDIR[1:0]		57
+0x03	TRIGSRC	TRIGSRC[7:0]								58
+0x04	TRFCNTL	TRFCNT[7:0]								61
+0x05	TRFCNTH	TRFCNT[15:8]								60
+0x06	REPCNT	REPCNT[7:0]								61
+0x07	Reserved	-	-	-	-	-	-	-	-	
+0x08	SRCADDR0	SRCADDR[7:0]								62
+0x09	SRCADDR1	SRCADDR[15:8]								62
+0x0A	SRCADDR2	SRCADDR[23:16]								61
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	DESTADDR0	DESTADDR[7:0]								63
+0x0D	DESTADDR1	DESTADDR[15:8]								62
+0x0E	DESTADDR2	DESTADDR[23:16]								62
+0x0F	Reserved	-	-	-	-	-	-	-	-	

5.17 DMA Interrupt Vector Summary

Table 5-13. DMA Interrupt vectors and their word offset addresses from the DMA Controller interrupt base

Offset	Source	Interrupt Description
0x00	CH0_vect	DMA Controller Channel 0 Interrupt vector
0x02	CH1_vect	DMA Controller Channel 1 Interrupt vector
0x04	CH2_vect	DMA Controller Channel 2 Interrupt vector
0x06	CH3_vect	DMA Controller Channel 3 Interrupt vector

6. Event System

6.1 Features

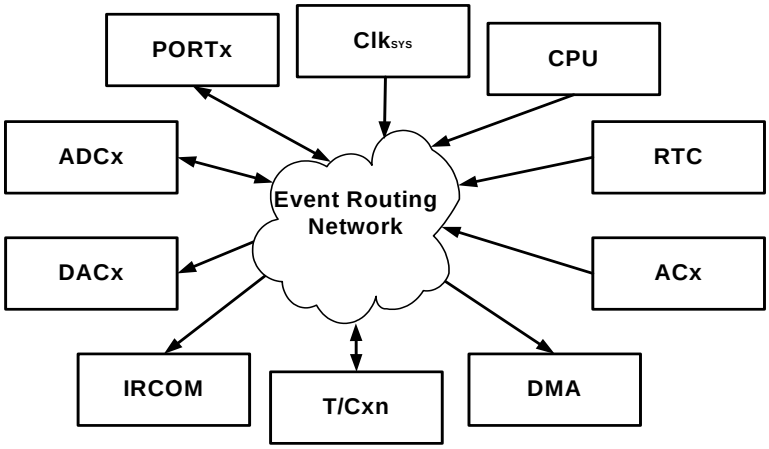
- Inter peripheral communication and signalling
- CPU and DMA independent operation
- 8 Event Channels allows for up to 8 signals to be routed at the same time
- 100% predictable timing between peripherals
- Events can be generated by
  - Timer/Counters (TCxn)
  - Real Time Counter (RTC)
  - Analog to Digital Converters (ADCx)
  - Analog Comparators (ACx)
  - Ports (PORTx)
  - System Clock (Clk_{SYS})
- Events can be used by
  - Timer/Counters (TCxn)
  - Analog to Digital Converters (ADCx)
  - Digital to Analog Converters (DACx)
  - DMA (DMA) Controller
  - Ports
- Advanced Features
  - Manual Event Generation from software (CPU)
  - Quadrature Decoding
  - Digital Filtering
- Operative in Active and Idle mode

6.2 Overview

The Event System is a set of features for inter peripheral communication. It enables the possibility for a change of state in one peripheral to automatically trigger actions in other peripherals. The change of state in a peripheral that will trigger actions in other peripherals is configurable in software. It is a simple, but powerful system as it allows for autonomous control of peripherals without any use of interrupt, CPU or DMA resources.

The indication of a change of state in a peripheral is referred to as an event. The events are passed between peripherals using a dedicated routing network called the Event Routing Network. Figure 6-1 on page 66 shows a basic block diagram of the Event System with the Event Routing Network and the peripherals that are connected.

Figure 6-1. Event System Overview and Connected Peripherals



The CPU is not part of the Event System, but it indicates that it is possible to manually generate events from software or by using the on-chip debug system.

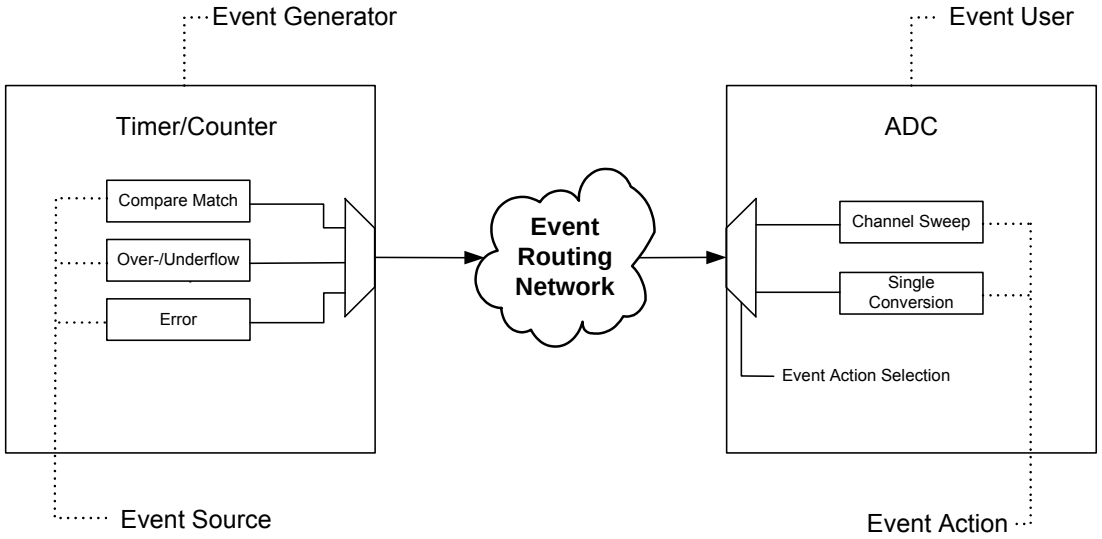
The Event System works in active and idle mode.

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 on the event.

The peripheral from where the event origin 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 be manually generated by writing to the STROBE and DATA registers.

6.3.1 Signaling Events

Signaling events are the most basic type of events. 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' is to be understood as a signaling event.

6.3.2 Data Events

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

The Event Routing Network can route all events to all event users. Event users that are only meant for using signaling events have limited decode capabilities and cannot fully utilize data. How event users decode data events is shown in Table 6-1 on page 67.

Event users that can utilize Data Events can also use Signaling Events. This is configurable, and is described in the data sheet module for each peripheral.

6.3.3 Manually Generating Events

Events can be generated manually by writing the DATA and STROBE register. This can be done from software, and by accessing the registers directly during on-chip debugging. 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.

Manually generated events last for one clock cycle and will overwrite events from other event during that clock cycle. When manually generating events, event channels where no events are entered will let other events through.

Table 6-1 on page 67 shows the different events, how they can be manually generated and how they are decoded.

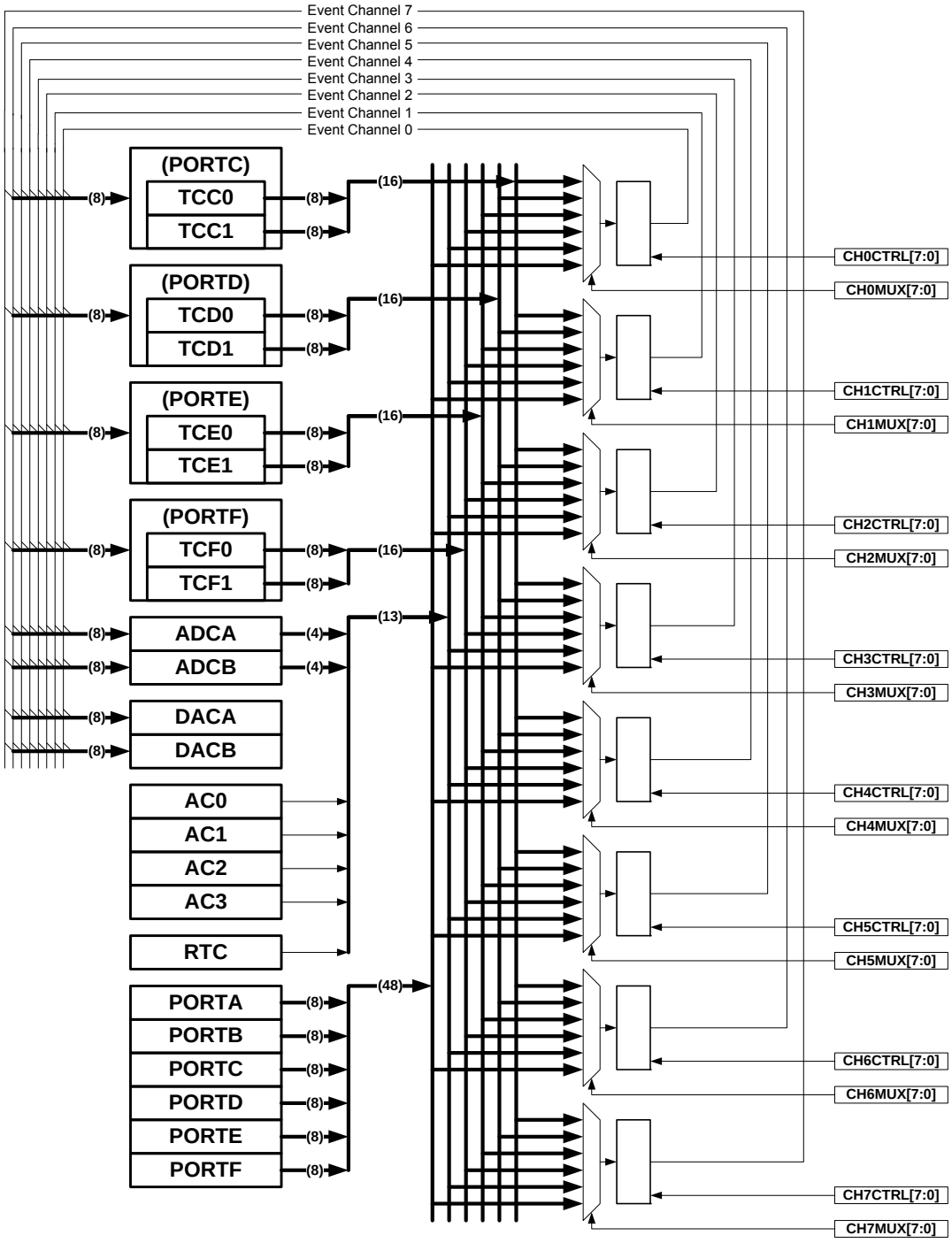
Table 6-1. Manually Generated Events and decoding of events

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 events between peripherals. It consists of eight multiplexers (CHnMUX), where events from all event sources are routed into all multiplexers. The multiplexers select which event is routed back as input to all peripherals. 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 these are described in the data sheet for each peripheral. The Event Routing Network is shown on Figure 6-3 on page 68.

Figure 6-3. Event Routing Network



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

Not all XMEGA parts contain all peripherals. This only means that 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 low level on an I/O pin, will generate events continuously. Details on this are described in the datasheet for each peripheral, but unless stated, an event lasts for one peripheral clock cycle only.

It takes maximum two clock cycles from an event is generated until the event actions in other peripherals is triggered. It takes one clock cycle from the event happens until it is registered by the event routing network on the first positive clock edge. It takes an additional clock cycle to route the event through the event channel to the event user.

6.6 Filtering

Each event channel includes a digital filter. When this is enabled for an event channel, an event must be sampled with the same value for 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 three Quadrature Decoders (QDECs). This enables the Event System to decode quadrature input on I/O pins, and send data events that a Timer/Counter can decode to trigger the appropriate event action: count up, count down or index/reset. Table 6-2 on page 69 summarizes which quadrature decoder data events are available, how they are decoded, and how they can be generated. The QDECs and related features, control and status register are available for event channel 0, 2 and 4.

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 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

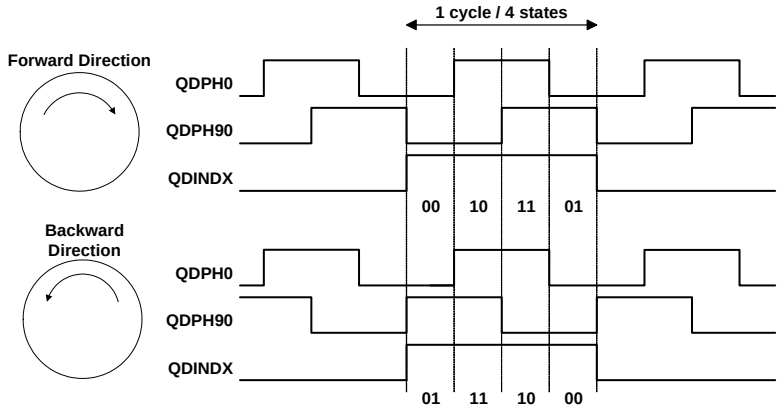


Figure 6-4 shows typical quadrature signals from a rotary encoder. The signals QDPH0 and QDPH90 are the two quadrature signals. When QDPH90 leads QDPH0, the rotation is defined as positive or forward. When QDPH0 leads QDPH90, the rotation is defined as negative, or reverse. The concatenation of the two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

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

6.7.2 QDEC Setup

For a full QDEC setup the following is required:

- I/O port pins - quadrature signal input
- The Event System - quadrature decoding
- A Timer/Counter - up, down and optional index count

The following procedure should be used for QDEC setup:

- Choose two successive pins on a port as QDEC phase inputs.
- Set pin direction for QDPH0 and QDPH90 as input.
- Set pin configuration for QDPH0 and QDPH90 to low level sense.
- Select QDPH0 pin as multiplexer input for an event channel, n.
- Enable quadrature decoding and digital filtering in the Event Channel.
- Optional:
  a. Setup QDEC index (QINDX).
  b. Select a third pin for QINDX input.
  c. Set pin direction for QINDX as input.
  d. Set pin configuration for QINDX to sense both edges.
  e. Select QINDX as multiplexer input for Event Channel n+1
  f. Set the Quadrature Index Enable bit in Event Channel n+1.
  g. Select the Index Recognition mode for Event Channel n+1.
- Set quadrature decoding as event action for a Timer/Counter.
- Select Event Channel n as event source 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 by setting CLKSEL to a CLKSEL_DIV1.

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]								CHnMUX
Read/Write	R/W	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 if 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 the 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 March
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	ACB_CH0	ACB Channel 0
0001	0	1	0	0	ACB_CH1	ACB Channel 1
0001	0	1	0	1	ACB_WIN	ACB Window
0001	0	1	1	X		(Reserved)

Table 6-3. CHnMUX[7:0] Bit Settings (Continued)

CHnMUX[7:4]	CHnMUX[3:0]				Group Configuration	Event Source
0001	1	X	X	X		(Reserved)
0010	0	0	n		ADCA_CHn	ADCA Channel n (n =0, 1, 2 or 3)
0010	0	1	n		ADCB_CHn	ADCB Channel n (n=0, 1, 2 or 3)
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 M (M=1 to 32768)
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
1101	1	E			See Table 6-4	Timer/Counter D1 event type E
1110	0	E			See Table 6-4	Timer/Counter E0 event type E
1110	1	E			See Table 6-4	Timer/Counter E1 event type E
1111	0	E			See Table 6-4	Timer/Counter F0 event type E
1111	1	E			See Table 6-4	Timer/Counter F1 event type E

Note: 1. The description of how PORTS generate events are described in "Port Event" on page 136.

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_CCA	Capture or Compare B (x = C, D, E or F) (n= 0 or 1)
1	1	0	TCxn_CCA	Capture or Compare C (x = C, D, E or F) (n= 0 or 1)
1	1	1	TCxn_CCA	Capture or Compare D (x = C, D, E or F) (n= 0 or 1)

6.8.2 CHnCTRL – Event Channel n Control Register

.

Bit	7	6	5	4	3	2	1	0	
	-	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			CHnCTRL
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 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 on page 73. These bits is only needed to set when a quadrature encoed with a connected index signal is used.

These bits are only available for CH0CTRL, CH2CTRL and CH4CTRL

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 QDEC index source, and the index data event will be enabled.

These bit is only available for CH0CTRL, CH2CTRL and CH4CTRL.

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

Setting this bit enables QDEC operation.

These bits is only available for CH0CTRL, CH2CTRL and CH4CTRL.

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

These bits define the length of digital filtering used. Events will be passed through to the event channel only when the event source has been active and sampled with the same level for a a number of peripheral clock for the number of cycles as defined by DIGFILT.

Table 6-6. Digital Filter Coefficient values

DIGFILT[2:0]	Group Configuration	Description
000	1SAMPLE	1 sample
001	2SAMPLES	2 samples
010	3SAMPLES	3 samples
011	4SAMPLES	4 samples
100	5SAMPLES	5 samples
101	6SAMPLES	6 samples
110	7SAMPLES	7 samples
111	8SAMPLES	8 samples

6.8.3 STROBE – Event Strobe Register

If the STROBE register location is written, each event channel will be set according to the STROBE[n] and corresponding DATA[n] bit setting 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]								STROBE
Read/Write	R/W	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 – Event Data Register

This register contains the data value when manually generating a data event. This register must be written before the STROBE register, for details See ”STROBE – Event Strobe Register” on page 74.

Bit	7	6	5	4	3	2	1	0	
+0x11	DATA[7:0]								DATA
Read/Write	R/W	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]								71
+0x01	CH1MUX	CH1MUX[7:0]								71
+0x02	CH2MUX	CH2MUX[7:0]								71
+0x03	CH3MUX	CH3MUX[7:0]								71
+0x04	CH4MUX	CH4MUX[7:0]								71
+0x05	CH5MUX	CH5MUX[7:0]								71
+0x06	CH6MUX	CH6MUX[7:0]								71
+0x07	CH7MUX	CH7MUX[7:0]								71
+0x08	CH0CTRL	-	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			73
+0x09	CH1CTRL	-	-	-	-	-	DIGFILT[2:0]			73
+0x0A	CH2CTRL	-	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			73
+0x0B	CH3CTRL	-	-	-	-	-	DIGFILT[2:0]			73
+0x0C	CH4CTRL	-	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			73
+0x0D	CH5CTRL	-	-	-	-	-	DIGFILT[2:0]			73
+0x0E	CH6CTRL	-	-	-	-	-	DIGFILT[2:0]			73
+0x0F	CH7CTRL	-	-	-	-	-	DIGFILT[2:0]			73
+0x10	STROBE	STROBE[7:0]								74
+0x11	DATA	DATA[7:0]								74

7. System Clock and Clock options

7.1 Features

- **Fast start-up time**
- **Safe run-time clock switching**
- **Internal Oscillators:**
  - **32 MHz run-time calibrated RC oscillator**
  - **2 MHz run-time calibrated RC oscillator**
  - **32.768 kHz calibrated RC oscillator**
  - **32 kHz Ultra Low Power (ULP) oscillator with 1 kHz output**
- **External clock options**
  - **0.4 - 16 MHz Crystal Oscillator**
  - **32.768 kHz Crystal Oscillator**
  - **External clock**
- **PLL with internal and external clock options with 1 to 31x multiplication**
- **Clock Prescalers with 1 to 2048x division**
- **Fast peripheral clock running at 2 and 4 times the CPU clock speed**
- **Automatic Run-Time Calibration of internal oscillators**
- **Crystal Oscillator failure detection**

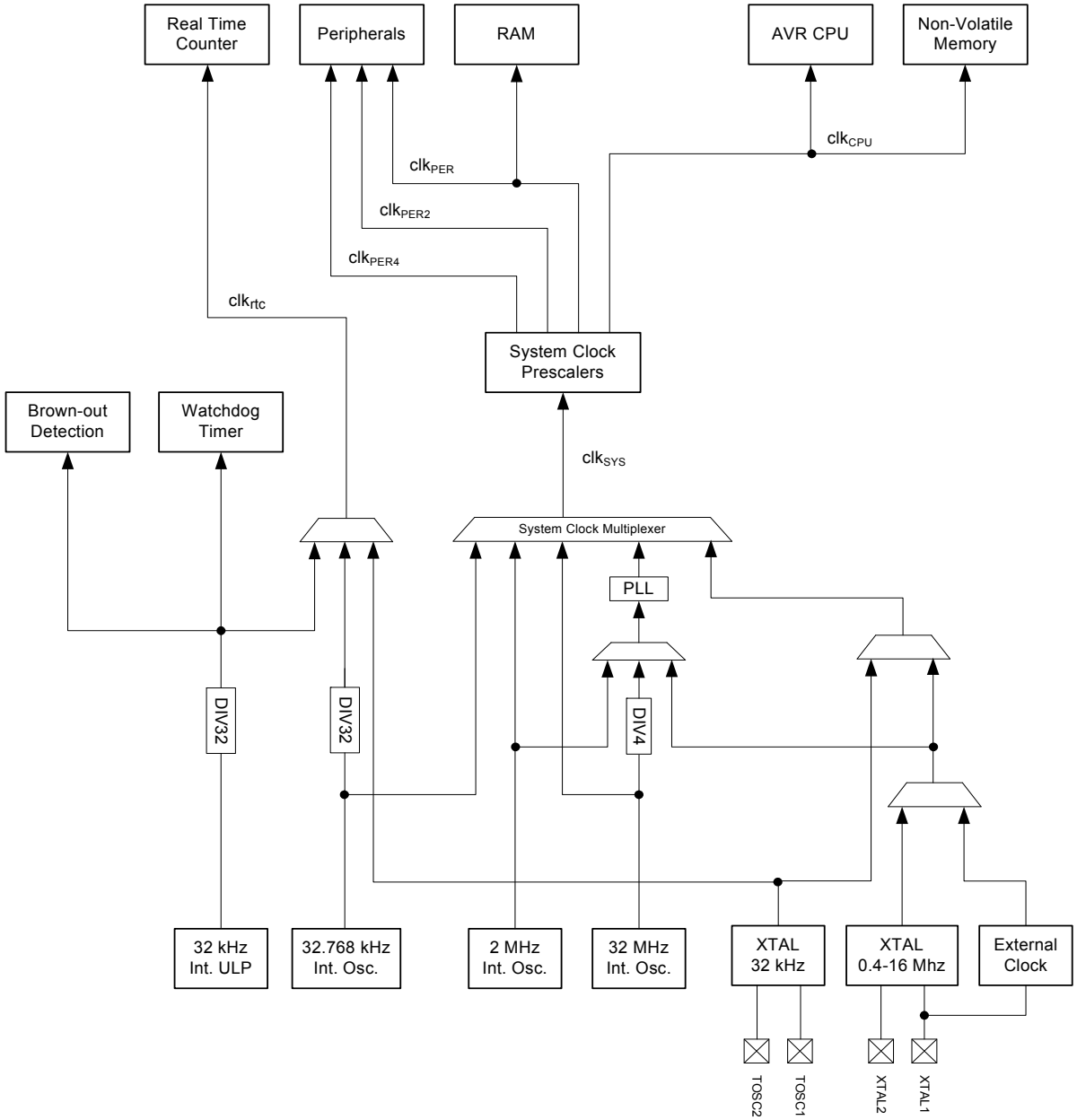
7.2 Overview

XMEGA has a flexible clock system, supporting a large number of clock sources. It incorporates both accurate integrated oscillators, and external crystal oscillators and resonators. 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. A Crystal Oscillator Failure Monitor can be enabled to issue a Non-Maskable Interrupt and switch to internal oscillator if the external oscillator fails.

After reset, the device will always start up running from the 2 MHz internal oscillator. During normal operation, the System Clock source and prescalers can be changed from software at any time.

Figure 7-1 on page 77 presents the principal clock system in the XMEGA. All of the clocks do not need to be active at a given time. The clocks to the CPU and peripherals can be stopped using sleep modes and power reduction registers as described in ”Power Management and Sleep” on page 95.

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



7.3 Clock Distribution

Figure 7-1 on page 77 presents the principal clock distribution in the XMEGA.

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 Clock.

7.3.2 CPU Clock - clk_{CPU}

The CPU Clock is routed to the CPU and Non-Volatile 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 if 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_{ASY}

The Asynchronous Clock allows the Real Time Counter (RTC) to be clocked directly from an external 32.768 kHz crystal oscillator, or the 32 times prescaled output from the internal 32.768 kHz oscillator or ULP oscillator. The dedicated clock domain allows operation of this peripheral, even if the device is in sleep mode where the rest of the clocks are stopped.

7.4 Clock Sources

The clock sources are divided in two main groups: internal oscillators and external clock sources. Most of the clock sources can be directly enabled and disabled from software, while others are automatically enabled or disabled dependent on current peripheral settings. After reset the device starts up running from the 2 MHz internal oscillator. The 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 data sheet.

7.4.1.1 32 kHz Ultra Low Power Oscillator

This oscillator provides an approximate 32 kHz clock. The 32 kHz 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 providing both a 32 kHz output and a 1 kHz output. The oscillator is automatically enabled/disabled when used as clock source for any part of the device. This oscillator can be selected as clock source for the RTC.

7.4.1.2 32.768 kHz Calibrated Internal Oscillator

This RC oscillator provides an approximate 32.768 kHz clock. A factory-calibrated value is written to the 32.768 kHz oscillator calibration register during reset to ensure that the oscillator is running within its specification. The calibration register can also be written from software for run-time calibration of the oscillator frequency. The oscillator employs a built in prescaler providing both a 32.768 kHz output and a 1.024 kHz output.

7.4.1.3 32 MHz Run-time Calibrated Internal Oscillator

This RC oscillator provides an approximate 32 MHz clock. The oscillator employs a Digital Frequency Locked Loop (DFLL) that can be enabled for automatic run-time calibration of the oscillator. A factory-calibrated value is written to the 32 MHz DFLL Calibration Register during reset to ensure that the oscillator is running within its specification. The calibration register can also be written from software for manual run-time calibration of the oscillator.

7.4.1.4 *2 MHz Run-time Calibrated Internal Oscillator*

This RC oscillator provides an approximate 2 MHz clock. The oscillator employs a Digital Frequency Looked Loop (DFLL) that can be enabled for automatic run-time calibration of the oscillator. A factory-calibrated value is written to the 2 MHz DFLL Calibration Register during reset to ensure that the oscillator is running within its specification. The calibration register can also be written from software for manual run-time calibration of the oscillator.

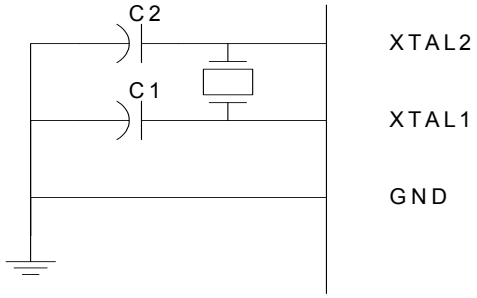
7.4.2 External Clock Sources

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 for driving a 32.768 kHz crystal oscillator.

7.4.2.1 *0.4 - 16 MHz Crystal Oscillator*

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

Figure 7-2. Crystal Oscillator Connection

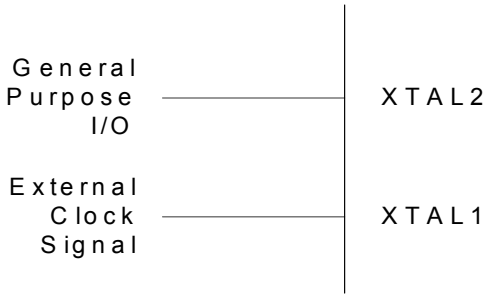


Two capacitors, C1 and C2, may be added to match the required load capacitance for the connected crystal.

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 79. In this mode, XTAL2 can be used as a general I/O pin.

Figure 7-3. External Clock Drive Configuration

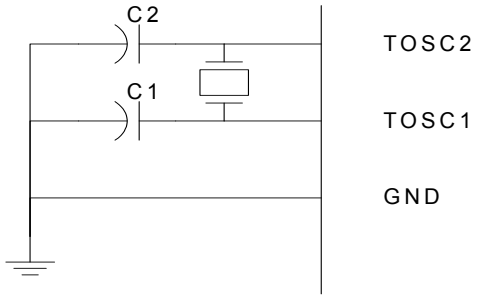


7.4.2.3 *32.768 kHz Crystal Oscillator*

A 32.768 kHz crystal oscillator can be connected between TOSC1 and TOSC2 by enabling a dedicated Low Frequency Oscillator input circuit. A typical connection is shown in Figure Figure

7-4 on page 80. A low power mode with reduced voltage swing on TOSC2 is available. This oscillator can be used as clock source for the System Clock, RTC and as the DFLL reference.

Figure 7-4. 32.768 kHz crystal oscillator connection



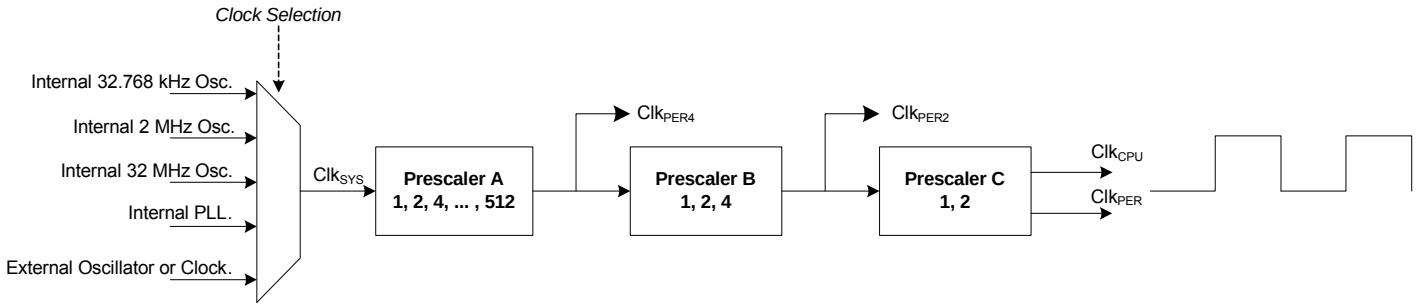
Two capacitors, C1 and C2, may be added to match the required load capacitance for the connected crystal.

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 clock source, or to disable the oscillator currently used as 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 1 to 512. Then prescaler B and C can be individually configured to either pass the clock through or divide it by a factor of 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 always updated in accordance to 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 the clk_{PER4}. Prescaler B and prescaler C can be enabled to divide the clock speed further and enable peripheral modules to run at twice or four times the CPU Clock frequency. If Prescaler B and C are not used all the clocks will run at the same frequency as 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 12.

7.6 PLL with 1-31x Multiplication Factor

A built-in Phase Locked Loop (PLL) can be used to generate a high frequency system clock. The PLL has a user selectable multiplication factor from 1 to 31. The output frequency, f_{OUT} is given by the input frequency, f_{IN} multiplied with the multiplication factor, PLL_FAC. The PLL must have a minimum output frequency of 10 MHz.

$$f_{OUT} = f_{IN} \cdot \text{PLL_FAC}$$

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

- 2 MHz internal oscillator
- 32 MHz internal oscillator divided by 4
- 0.4 - 16 MHz Crystal Oscillator
- External clock

To enable the PLL the following procedure must be followed:

1.Enable clock reference 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 stabile and the PLL has locked.

If using PLL and DFLL the active reference cannot be disabled.

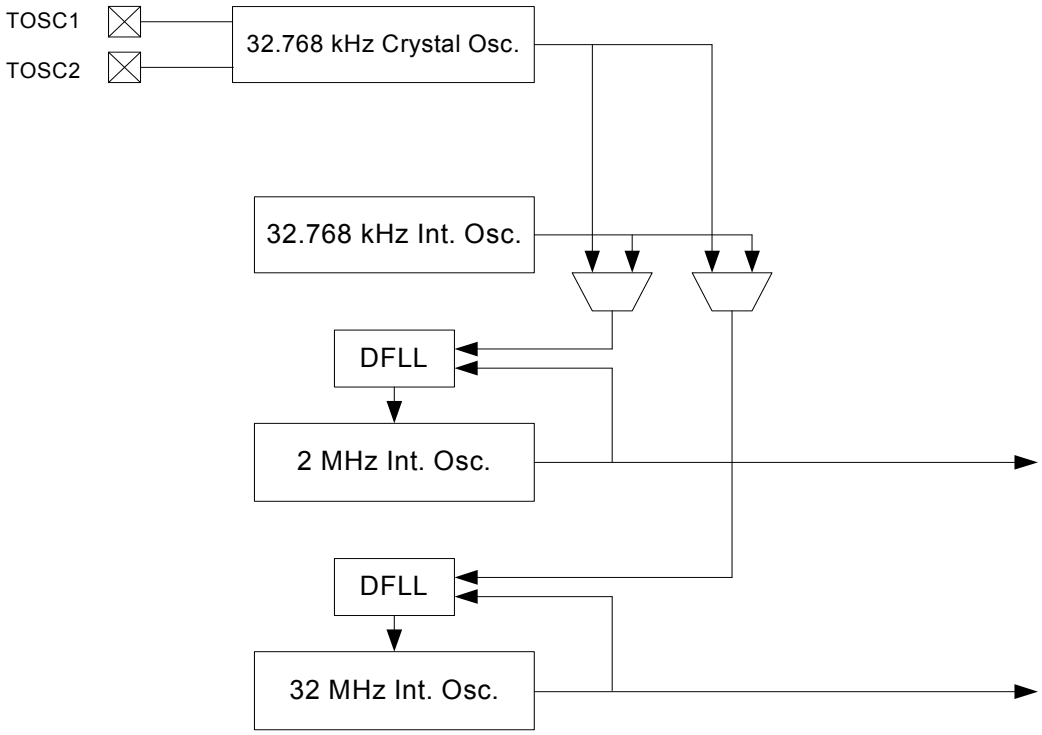
7.7 DFLL 2 MHz and DFLL 32 MHz

Two built-in Digital Frequency Locked Loops (DFLLs) can be used to improve the accuracy of the 2 MHz and 32 MHz internal oscillators. The DFLL compares the oscillator frequency with a more accurate reference clock to do automatic run-time calibration of the oscillator. The choices for the reference clock sources are:

- 32.768 kHz Calibrated Internal Oscillator
- 32.768 kHz Crystal Oscillator connected to the TOSC pins

The DFLLs divide the reference clock by 32 to use a 1.024 kHz reference. The reference clock is individually selected for each DFLL as shown on Figure 7-6 on page 82.

Figure 7-6. Figure 5-5. DFLL reference clock selection



When the DFLL is enabled it will count each oscillator clock cycle, and for each reference clock edge, the counter value is compared to the fixed ideal relationship between the reference clock and the 1kHz reference frequency. If the internal oscillator runs too fast or too slow, the DFLL will decrement or increment the corresponding DFLL Calibration Register value by one to adjust the oscillator frequency slightly. When the DFLL is enabled the DFLL Calibration Register cannot be written from software.

The ideal counter value representing the number of oscillator clock cycles for each reference clock cycle is loaded to the DFLL Oscillator Compare Register during reset. The register can also be written from software to change the frequency the internal oscillator is calibrated to.

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. For the DFLL Calibration Register to be reloaded with the default value it has after reset, the DFLL must disabled before entering sleep and enabled the again after leaving sleep.

The active reference cannot be disabled when the DFLL is enabled.

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

For details on internal oscillator accuracy when the DFLL is enabled, refer to the device data sheet.

7.8 External Clock Source Failure Monitor

To handle external clock source failures, there is a built-in monitor circuit monitoring the oscillator or clock used to derive the XOSC clock. The External Clock Source Failure Monitor is disabled by default, and it must be enabled from software before it can be used. If an external

clock or oscillator is used to derive the System Clock (i.e clock reference for the PLL when this is used as the active system clock) and an clock or oscillator fails (stops), the device will:

- Switch to the 2 MHz internal oscillator, independently of any clock system lock setting.
- Reset the Oscillator Control Register and System Clock Selection Register to their default values.
- Set the External Clock Source Failure Detection Interrupt Flag.
- Issue a non-maskable interrupt (NMI).

If the external oscillator fails when it is not used as the System Clock source, the external oscillator is automatically disabled while the system clock will continue to operate normally.

If the external clock is below 32 kHz then the failure monitor mechanism should not be enabled in order to avoid unintentional fail detection.

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

The failure monitor is automatically disabled in all sleep modes where the external clock or oscillator is stopped. During wake-up from sleep it is automatically enabled again.

The External Clock Source Failure Monitor setting is 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 12.

7.9 Register Description - Clock

7.9.1 CTRL - System Clock Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	-	-	SCLKSEL[2:0]			CTRL
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**

SCLKSEL is used to select the source for the System Clock. See Table 7-1 for the different selections. Changing the system clock source will take 2 clock cycles on the old clock source and 2 clock cycles on the new clock source. These bits are protected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

SCLKSEL cannot be changed if the new source is not stable.

Table 7-1. System Clock Selection

SCLKSEL[2:0]	Group Configuration	Description
000	RC2MHz	2 MHz Internal RC Oscillator
001	RC32MHz	32 MHz Internal RC Oscillator
010	RC32KHz	32 kHz Internal RC Oscillator
011	XOSC	External Oscillator or Clock
100	PLL	Phase Locked Loop
101	-	Reserved
110	-	Reserved
111	-	Reserved

7.9.2 PSCTRL - System Clock Prescaler Register

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

- **Bit 7 - Reserved**

This bit is reserved and will always be read as zero. 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 run-time to change the clock 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 Factor**

These bits define the division ratio of the clock prescaler 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}. 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 80 fore more details.

Table 7-3. Prescaler B and C division factor

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 - Clock System Lock Register

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	-	-	-	LOCK	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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 - LOCK: Clock System Lock**

When the LOCK bit is written to one the CTRL and PSCTRL registers cannot be changed, and the system clock selection and prescaler settings is protected against all further updates until after the next reset. This bits are protected by the Configuration Change Protection mechanism, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

The LOCK bit will only be cleared by a system reset.

7.9.4 RTCCTRL - RTC Control Register

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	RTCSRC[2:0]			RTCEN	RTCCTRL
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]: Clock Source**

These bits select the clock source for the Real Time Counter according to Table 7-4..

Table 7-4. RTC Clock Source

RTCSRC[2:0]	Group Configuration	Description
000	ULP	1 kHz from internal 32 kHz ULP
001	TOSC	1.024 kHz from 32.768 kHz Crystal Oscillator on TOSC
010	RCOSC	1.024 kHz from internal 32.768 kHz RC Oscillator
011	-	Reserved
100	-	Reserved
101	TOSC32	32.768 kHz from 32.768 kHz Crystal Oscillator on TOSC
110	-	Reserved
111	-	Reserved

- **Bit 0 - RTCEN: RTC Clock Source Enable**

Setting the RTCEN bit enables the selected clock source for the Real Time Counter.

7.10 Register Description - Oscillator

7.10.1 CTRL - Oscillator Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	PLLEN	XOSCEN	RC32KEN	RC32MEN	RC2MEN	CTRL
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 should be configured with the desired multiplication factor and input source. See ”STATUS - Oscillator Status Register” on page 88..

- **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 and enable an external clock source. The external clock source should be allowed time to become stable before it is selected as source for the System Clock. See ”STATUS - Oscillator Status Register” on page 88.

- **Bit 2 - RC32KEN: 32.768 kHz Internal RC Oscillator Enable**

Setting this bit enables the 32.768 kHz internal RC oscillator. The oscillator must be stable before it is selected as source for the System Clock. See ”STATUS - Oscillator Status Register” on page 88.

- **Bit 1- RC32MEN: 32 MHz Internal RC Oscillator Enable**

Setting this bit will enable the 32 MHz internal RC oscillator. The oscillators should be allowed time to become stable before wither is selected as source for the System Clock. See ”STATUS - Oscillator Status Register” on page 88.

- **Bit 0 - RC2MEN: 2 MHz Internal RC Oscillator enable**

Setting this bit enables the 2MHz internal RC oscillator. The oscillator should be allowed time to become stable before wither is selected as source for the System Clock. See ”STATUS - Oscillator Status Register” on page 88.

By default the 2 Mhz Internal RC 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	R32MRDY	RC2MRDY	STATUS
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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 4 - PLLRDY: PLL Ready**

The PLLRDY flag is set when the PLL has locked on selected frequency and ready to be used as he System Clock source.

- **Bit 3 - XOSCRDY: External clock source Ready**

The XOSCRDY flag is set when the external clock source is stable and ready to be used as the System Clock source.

- **Bit 2 - RC32KRDY: 32.768 kHz Internal RC Oscillator Ready**

The RC32KRDY flag is set when the 32.768 kHz internal RC oscillator is stable and ready to be used as the System Clock source.

- **Bit 1 - RC32MRDY: 32 MHz Internal RC Oscillator Ready**

The R32MRFY flag is set when the 32 MHz internal RC oscillator is stable and ready to be used as the System Clock source.

- **Bit 0 - RC2MRDY: 2 MHz Internal RC Oscillator Ready**

The RC2MRDY flag is set when the 2 MHz internal RC 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	-	XOSCSEL[3:0]				XOSCCTRL
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 - FRQRANGE[1:0]: Crystal Oscillator Frequency Range Select**

These bits select the frequency range for the connected crystal oscillator according to Table 7-5 on page 89.

Table 7-5. Oscillator frequency range selection

FRQRANGE[1:0]	Group Configuration	Frequency range	Recommended range for capacitors C1 and C2 (pF)
00	04TO2	0.4 MHz - 2 MHz	100
01	2TO9	2 MHz - 9 MHz	15
10	9TO12	9 MHz - 12 MHz	15
11	12TO16	12 MHz - 16 MHz	10

- **Bit 5 - X32KLPM: Crystal Oscillator 32.768 kHz Low Power Mode**

Setting this bit enables low power mode for the 32.768 kHz Crystal Oscillator. This will reduce the swing on the TOSC2 pin.

- **Bit 4 - 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 3:0 - XOSCSEL[3:0]: Crystal Oscillator Selection**

These bits select the type and start-up time for the crystal or resonator that is connected to the XTAL or TOSC pins. See Table 7-6 for crystal selections. If external clock or external oscillator is selected as source for System Clock, see ”CTRL - Oscillator Control Register” on page 87, this configuration can not be changed.

Table 7-6. External Oscillator selection and Startup Time

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

Notes:
1. This option should only be used when frequency stability at start-up is not important for the application. The option is not suitable for crystals.
2. This option is intended for use with ceramic resonators and will ensure frequency stability at start-up. It can also be used when the frequency stability at start-up is not important for the application.

7.10.4 XOSCFAIL - XOSC Failure Detection Register

Bit	7	6	5	4	3	2	1	0	
+0x03	–	–	–	–	–	–	XOSCFDIF	XOSCFDEN	XOSCFAIL
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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 - XOSCFDIF: Failure Detection Interrupt Flag**

If the external clock source oscillator failure monitor is enabled, the XOSCFDIF is set when a failure is detected. Writing logic one to this location will clear XOSCFDIF. Note that having this flag set will not stop the fail monitor circuit to request a new interrupt if the external clock sources are re-enabled and a new failure occurs.

- **Bit 0 - XOSCFDEN: Failure Detection Enable**

Setting this bit will enable the failure detection monitor, and a Non-Maskable Interrupt will be issued when the XOSCFDIF is set.

This bit is protected by the Configuration Change Protection mechanism, refer to Section 3.12 ”Configuration Change Protection” on page 12 for details. Once enabled, the failure detection will only be disabled by a reset.

7.10.5 RC32KCAL - 32 KHz Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
+0x04	RC32KCAL[7:0]								RC32KCAL
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]: 32.768 KHz Internal Oscillator Calibration Register**

This register is used to calibrate the Internal 32.768 kHz 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 32.768 kHz. 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]		-	PLLFAC[4:0]					PLLCTRL
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 - PLLSRC[1:0]: Clock Source**

The PLLSRC bits select the input source for the PLL according to Table 7-7 on page 91.

Table 7-7. PLL Clock Source

CLKSRC[1:0]	Group Configuration	PLL input source
00	RC2M	2 MHz Internal RC Oscillator
01	-	Reserved
10	RC32M	32 MHz Internal RC Oscillator
11	XOSC	External Clock Source(1)

Notes: 1. 32 kHz TOSC cannot be selected as source for the PLL. An external clock must be minimum 0.4 MHz to be used as source clock.

- **Bit 5 - 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 4:0 - PLLFAC[4:0]: Multiplication Factor**

The PLLFAC bits set the multiplication factor for the PLL. The multiplication factor can be in the range from 1x to 31x. The output frequency from the PLL should not exceed 200 MHz. The PLL must have a minimum output frequency of 10 MHz.

7.10.7 DFLLCTRL - DFLL Control Register

Bit	7	6	5	4	3	2	1	0	
+0x06	–	–	–	–	–	–	R32MCREF	RC2MCREF	DFLLCTRL
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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 - RC32MCREF: 32 MHz Calibration Reference**

This bit is used to select the calibration source for the 32 MHz DFLL. By default this bit is zero and the 32.768 kHz internal RC oscillator is selected. If this bit is set to one the 32.768 kHz Crystal Oscillator connected to TOSC selected as reference. The XOSCEN bit in the CTRL register must be set to enable the external oscillator, and the XOSCLSEL bits in the XOSCCTRL register must be set to 32.768 kHz TOSC when this clock source is selected as the the 32 MHz DFLL reference.

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

This bit is used to select the calibration source for the 2 MHz DFLL. By default this bit is zero and the 32.768 kHz internal RC oscillator is selected. If this bit is set to one the 32.768 kHz Crystal Oscillator on TOSC is selected as reference. The XOSCEN bit in the CTRL register must be set to enable the external oscillator, and the XOSCLSEL bits in the XOSCCTRL register must be set to 32.768 kHz TOSC when this clock source is selected as the the 2 MHz DFLL reference.

7.11 Register Description - DFLL32M/DFLL2M

7.11.1 CTRL - DFLL Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	–	–	–	–	–	–	–	ENABLE	CTRL
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 reserved and will always be read as zero. 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

7.11.2 CALA - Calibration Register A

CALA and CALB register holds the 13 bit DFLL calibration value that is used for automatic run-time calibration 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 be calibrated according to the calibration value in these registers also when the DFLL is disabled.

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

- **Bit 7:0 - CALL[7:0]: DFLL Calibration bits**

These bits hold the 7 Least Significant Bits (LSB) of the calibration value for the oscillator. After reset CALL is set to its middle value, and during automatic runtime calibration of the oscillator these bits are use to change the oscillator frequency. The bits are controlled by the DFLL when the DFLL is enabled.

7.11.3 CALB - Calibration Register B

Bit	7	6	5	4	3	2	1	0	
+0x03	–	–	–	CALH[12:8]					CALB
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	x	x	x	x	x	

- **Bit 7:4 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 4:0 - CALH[12:8]: DFLL Calibration bits**

These bits hold the 6 Most Significant Bits (MSB) of the calibration value for the 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 approximate to the nominal frequency for the oscillator. These bits are not changed during automatic runtime calibration of the oscillator.

7.11.4 COMP0 - Oscillator Compare Register 0

COMP0, COMP1 and COMP2 represent the register value COMP that hold the oscillator compare value. During reset COMP is loaded with the default value representing the ideal relationship between oscillator frequency and the 1.024 kHz reference clock. It is possible to write these bits from software, and then enable the oscillator to tune to a frequency different than its nominal frequency. These bits can only be written when the DFLL is disabled.

Bit	7	6	5	4	3	2	1	0	
+0x04	COMP[7:0]								COMP0
Read/Write	R/W	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]**

These bits are the low byte of the COMP register.

7.11.5 COMP1 - Oscillator Compare Register 1

Bit	7	6	5	4	3	2	1	0	
+0x05	COMP[15:8]								COMP1
Read/Write	R/W	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]**

These bits are the middle byte of the COMP register.

7.11.6 COMP2 - Oscillator Compare Register 2

7.11.6.1 COMP2 - Oscillator Compare Register 2, 2 MHz DFLL /32 MHz DFLL

Bit	7	6	5	4	3	2	1	0	
+0x06	COMP[23:16]								COMP2
Read/Write	R/W	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[23:16]**

These bits are the highest bits of the COMP register.

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]			84
+0x01	PSCTRL	–	PSADIV[4:0]					PSBCDIV[1:0]		84
+0x02	LOCK	–	–	–	–	–	–	–	LOCK	86
+0x03	RTCCTRL	–	–	–	–	RTCSRC[2:0]			RTCEN	86
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	

7.13 Register Summary - Oscillator

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	PLLEN	XOSCEN	RC32KEN	R32MEN	RC2MEN	87
+0x01	STATUS	–	–	–	PLLRDY	XOSCRDY	RC32KRDY	R32MRDY	RC2MRDY	87
+0x02	XOSCCTRL	FRQRANGE[1:0]		X32KLPM	–	XOSCSEL[3:0]				88
+0x03	XOSCFAIL	–	–	–	–	–	–	–	–	89
+0x04	RC32KCAL	RC32KCAL[7:0]								90
+0x05	PLLCTRL	PLLSRC[1:0]		-	PLLFAC[4:0]					90
+0x06	DFLLCTRL	–	–	–	–	–	–	–	–	91
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	Reserved	–	–	–	–	–	–	–	–	
+0x09	Reserved	–	–	–	–	–	–	–	–	
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	Reserved	–	–	–	–	–	–	–	–	
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	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	92
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	CALA	CALL[7:0]								92
+0x03	CALB	–	–	–	CALH[11:8]					92
+0x04	COMP0	COMP[7:0]								92
+0x05	COMP1	COMP[15:8]								93
+0x06	COMP2	COMP[23:16]								93
+0x07	COMP2	COMP[7:0]								93

7.15 Crystal Oscillator Failure Interrupt Vector Summary

Table 7-8. Crystal Oscillator Failure Interrupt vector and its word offset address Crystal Oscillator Failure interrupt base

Offset	Source	Interrupt Description
0x00	OSCF_vect	Crystal Oscillator Failure Interrupt Vector (NMI)

8. Power Management and Sleep

8.1 Features

- **5 sleep modes**
  - **Idle**
  - **Power-down**
  - **Power-save**
  - **Standby**
  - **Extended standby**
- **Power Reduction register to disable clock to unused peripherals**

8.2 Overview

XMEGA provides various sleep modes and software controlled clock gating in order to tailor power consumption to the application's requirement. Sleep modes enables the microcontroller to shut down unused modules to save power. When the device enters sleep mode, program execution is stopped and interrupts or reset is used to wake the device again. The individual clock to unused peripherals can be stopped during normal operation or in sleep, enabling a 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 has five different sleep modes. A dedicated Sleep instruction (SLEEP) is available to enter sleep. Before executing SLEEP, the selected sleep mode to enter must be configured. The available interrupt wake-up sources is dependent on the selected 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 clock**	**RTC clock**	**System clock source**	**RTC clock source**	**Asynchronous Port Interrupts**	**TWI Address match interrupts**	**Real Time Clock Interrupts**	**All interrupts**
Idle		X	X	X	X	X	X	X	X
Power-down						X	X		
Power-save			X		X	X	X	X	
Standby				X		X	X		
Extended Standby			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 start-up time for the system clock source must be added to the wake-up time for sleep modes where the clock source is stopped. For details on the start-up time for the different oscillators options refer to "System Clock and Clock options" on page 76.

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 Non-Volatile Memory are stopped, (note that any active programming will be completed) but all peripherals including the Interrupt Controller, Event System and DMA Controller are kept running. Any interrupt request interrupts will wake the device.

8.3.2 Power-down Mode

In Power-down mode all system clock sources, including the Real Time Counter clock source are stopped. This allows operation of asynchronous modules only. The only interrupts that can wake up the MCU are the Two Wire Interface address match interrupts, and asynchronous port interrupts.

8.3.3 Power-save Mode

Power-save mode is identical to Power-down, with one exception:

If the Real Time Counter (RTC) is enabled, it will keep running during sleep and the device can also wake up from either RTC Overflow or Compare Match 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 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 provides 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 in most cases 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 mode to reduce the overall power consumption significantly. 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 Register Description – Sleep

8.5.1 CTRL- Sleep Control Register

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

- **Bit 7:4 - Reserved**

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

- **Bit 3:1 - SMODE[2:0]: Sleep Mode Selection**

These bits select sleep modes according to Table 8-2 on page 97.

Table 8-2. Sleep mode

SMODE[2:0]	SEN	Group Configuration	Description
XXX	0	OFF	No sleep mode enabled
000	1	IDLE	Idle Mode
001	1	-	Reserved
010	1	PDOWN	Power-down Mode
011	1	PSAVE	Power-save Mode
100	1	-	Reserved

Table 8-2. Sleep mode

SMODE[2:0]	SEN	Group Configuration	Description
101	1	-	Reserved
110	1	STDBY	Standby Mode
111	1	ESTDBY	Extended Standby Mode

- **Bit 1 - 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 clearing it immediately after waking up.

8.6 Register Description – Power Reduction

8.6.1 PRGEN - General Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	AES	EBI	RTC	EVSYS	DMA	PRGEN
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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 4 - AES: AES Module**

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

- **Bit 3 - EBI: External Bus Interface**

Setting this bit stops the clock to the External Bus Interface. When the bit is cleared the peripheral should be reinitialized to ensure proper operation. Note that the EBI is not present for all devices.

- **Bit 2 - RTC: Real-Time Counter**

Setting this stops the clock to the Real Time Counter. When the 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 the bit is cleared the module will continue like before the shutdown.

- **Bit 0 - DMA: DMA Controller**

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

8.6.2 PRPA/B - Power Reduction Port A/B Register

Bit	7	6	5	4	3	2	1	0	
+0x01/+0x02	-	-	-	-	-	DAC	ADC	AC	PRPA/B
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

- **Bit 7:3 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 2 - DAC: Power Reduction DAC**

Setting this bit stops the clock to the DAC. The DAC should be disabled before shut down.

- **Bit 1 - ADC: Power Reduction ADC**

Setting this bit stops the clock to the ADC. The ADC should be disabled before shut down.

- **Bit 0 - AC: Power Reduction Analog Comparator**

Setting this bit stops the clock to the Analog Comparator. The AC should be disabled before shut down.

8.6.3 PRPC/D/E/F - Power Reduction Port C/D/E/F Register

Bit	7	6	5	4	3	2	1	0	
	-	TWI	USART1	USART0	SPI	HIRES	TC1	TC0	PRPC/D/E/F
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - Reserved**

This bit is reserved and will always be read as zero. 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 the bit is cleared the peripheral should be reinitialized to ensure proper operation.

- **Bit 5 - USART1**

Setting this bit stops the clock to the USART1. When the bit is cleared the peripheral should be reinitialized to ensure proper operation.

- **Bit 4 - USART0**

Setting this bit stops the clock to the USART0. When the 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 the bit is cleared the peripheral should be reinitialized to ensure proper operation.

- **Bit 2 - HIRES: Hi-Resolution Extension**

Setting this bit stops the clock to the Hi-Resolution Extension for the Timer/Counters. When the 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 the Timer/Counter 1. When the 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 the Timer/Counter 0. When the bit is cleared the peripheral will continue like before the shut down.

8.7 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	97
+0x01	Reserved	-	-	-	-	-	-	-	-	
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	Reserved	-	-	-	-	-	-	-	-	
+0x04	Reserved	-	-	-	-	-	-	-	-	
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	Reserved	-	-	-	-	-	-	-	-	
+0x07	Reserved	-	-	-	-	-	-	-	-	

8.8 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	-	-	-	AES	EBI	RTC	EVSYS	DMA	98
+0x01	PRPA	-	-	-	-	-	DAC	ADC	AC	99
+0x02	PRPB	-	-	-	-	-	DAC	ADC	AC	99
+0x03	PRPC	-	TWI	USART1	USART0	SPI	HIRES	TC1	TC0	99
+0x04	PRPD	-	TWI	USART1	USART0	SPI	HIRES	TC1	TC0	99
+0x05	PRPE	-	TWI	USART1	USART0	SPI	HIRES	TC1	TC0	99
+0x06	PRPF	-	TWI	USART1	USART0	SPI	HIRES	TC1	TC0	99
+0x07	Reserved	-	-	-	-	-	-	-	-	

9. Reset System

9.1 Features

- **Power-on reset source**
- **Brown-out reset source**
- **Software reset source**
- **External reset source**
- **Watchdog reset source**
- **Program and Debug Interface reset source**

9.2 Overview

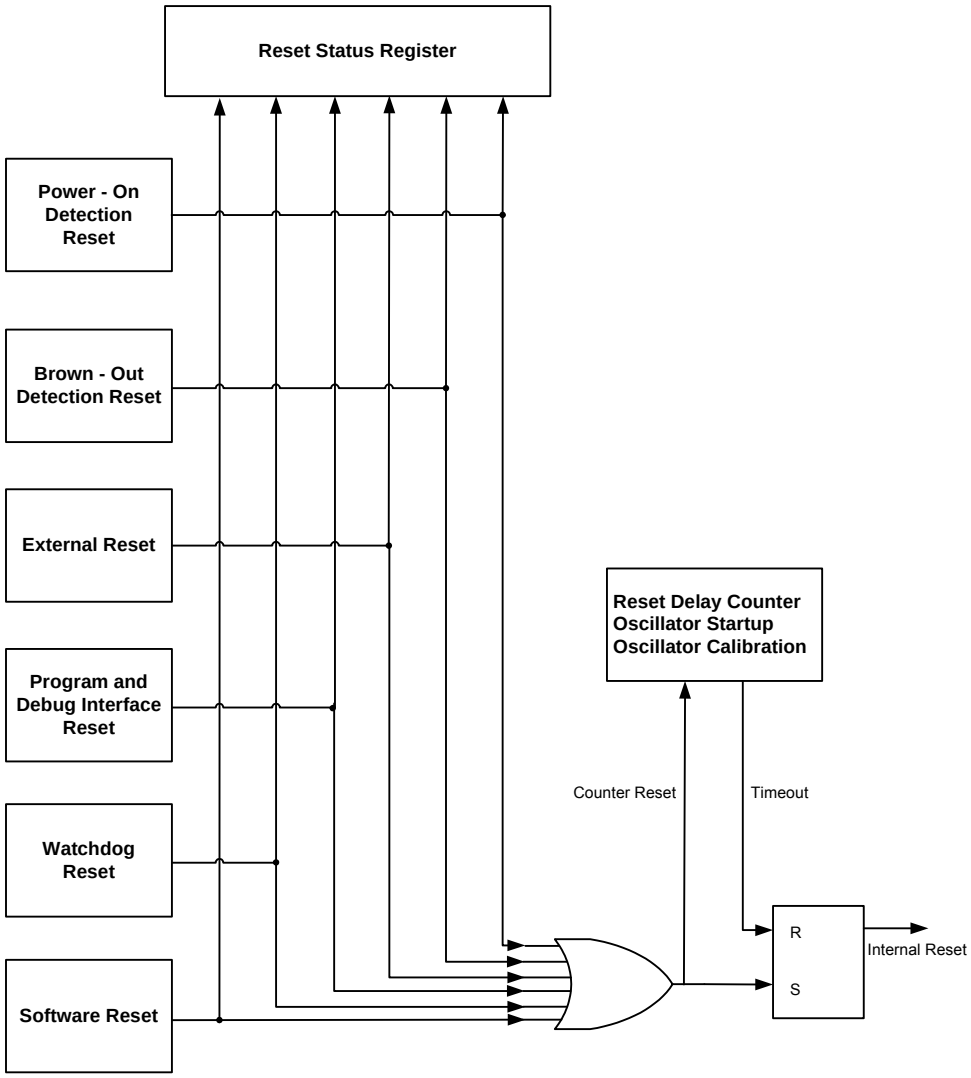
The Reset System will issue a system reset and set the device to its initial state if a reset source goes active. All IO registers will be set to their initial value, and the program counter is reset to the Reset Vector location. The reset controller is asynchronous, hence no running clock is required to reset the device.

XMEGA has seven different reset sources. If more than one reset source is active, the device will be kept in reset until all reset sources have released their reset. After reset is released from all reset sources, the default oscillator is started and calibrated before the internal reset is released and the device starts running.

The reset system has a status register with individual flags for each reset source. The Status register is cleared at Power-on Reset, hence this register will show which source(s) that has issued a reset since the last power-on. A software reset feature makes it possible to issue a system reset from the user software.

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

Reset request from any reset source 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 internal reset is released and the device starts running.

- Reset Counter Delay
- Oscillator startup
- Oscillator calibration

If one of the reset request occur during this, the reset sequence will start over again.

9.3.1 Reset Counter Delay

The Reset Counter Delay is the programmable period from all reset requests are released and until the reset counter times out and releases reset. The Reset Counter Delay is timed from the 1 kHz output of the Ultra Low Power (ULP) Internal Oscillator, and the number of cycles before the timeout is set by the STARTUPTIME fuse bits. The selectable delays are shown in Table 9-1.

Table 9-1. Reset Counter Delay

SUT[1:0]	Number of 1 kHz ULP oscillator clock cycles
00	64
01	4
10	Reserved
11	0

9.3.2 Oscillator Startup

After the Reset Counter Delay, the default clock is started. This is the 2 MHz internal RC oscillator, and this uses 6 clock cycles to startup and stabilize.

9.3.3 Oscillator Calibration

After the default oscillator has stabilized, oscillator calibration values are loaded from Non-Volatile Memory into the Oscillator Calibration registers. Loading the calibration values takes 24 clock cycles on the internal 2 MHz oscillator. The 2 MHz, 32 MHz and 32 kHz internal RC oscillators are calibrated. When this is done the device will enter active mode and program execution will begin.

9.4 Reset Sources

9.4.1 Power-On Reset

A Power-on detection circuit will give a Power-On Reset (POR) when supply voltage (V_{CC}) is applied to the device and the V_{CC} slope is increasing in the Power-on Slope Range (V_{POSR}). Power-on reset is released when the V_{CC} stops rising or when the V_{CC} level has reached the Power-on Threshold Voltage (V_{POT}) level.

When V_{CC} is falling, the POR will issue a reset when the Vpot level is reached. The Vpot level is lower than the minimum operating voltage for the device, and is only used for power-off function-

ality and not to ensure safe operation. The Brown-out detection (BOD) must be enabled to ensure safe operation and detect if V_{CC} voltage drops below minimum operating voltage.

Only the Power-on reset Flag will be set after Power-on reset. The Brown-out Reset Flag is not set even though the BOD circuit is used.

Figure 9-2. Power-On Reset (POR).

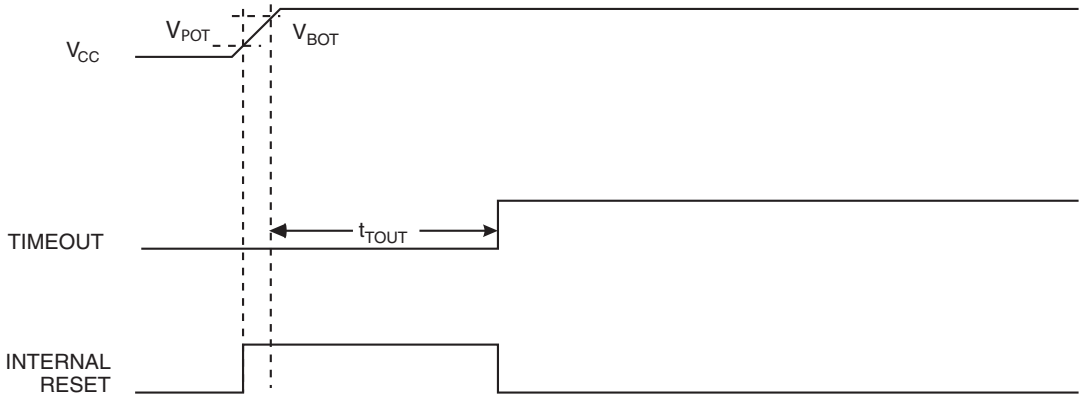
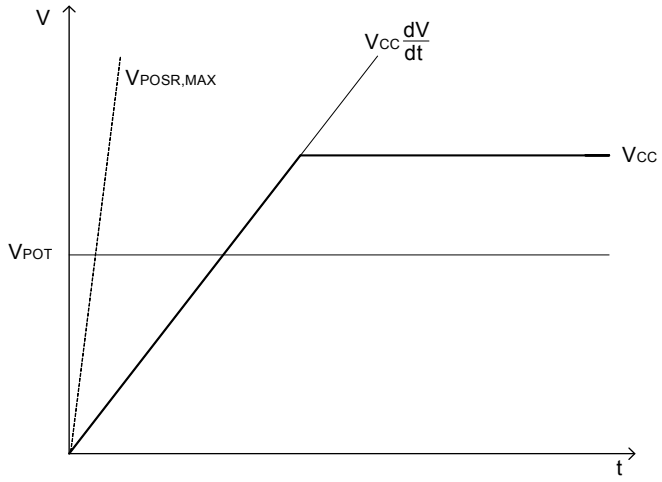


Figure 9-3. Increasing V_{CC} slope in the Power-on slope range.



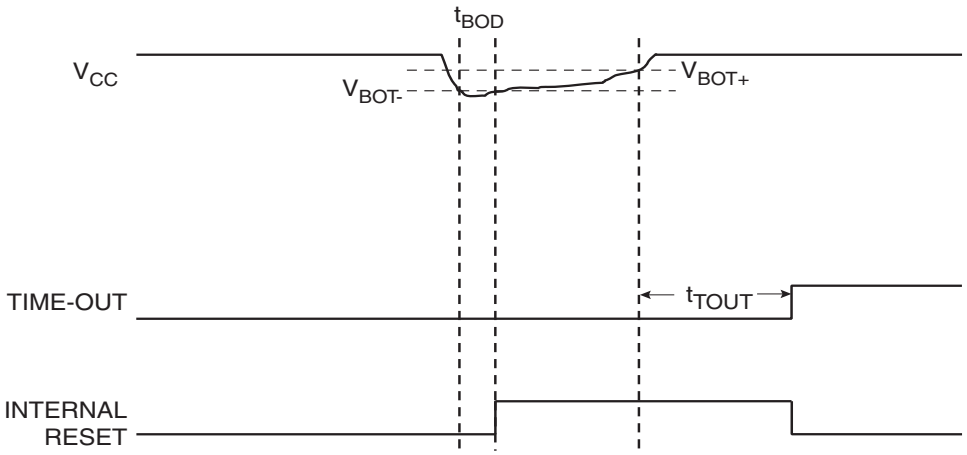
For characterization data on the V_{POT} level for rising and falling V_{CC}, and V_{POSR} slope consult the device data sheet.

Note that the Power-on detection circuit is not designed to detect drops in the V_{CC} voltage. Brown-out detection must be enabled to detect falling V_{CC} voltage, even if the V_{CC} level falls below the V_{POT} level.

9.4.2 Brown-Out Detection

The Brown-Out Detection (BOD) circuit monitors that the V_{CC} level is kept above a configurable trigger level, V_{BOT}. When the BOD is enabled, a BOD reset will be given if the V_{CC} level falls below the trigger level for a minimum time, t_{BOD}. The reset is kept active until the V_{CC} level rises above the trigger level again.

Figure 9-4. Brown-out Detection reset.



For characterization data on t_{BOD} consult the device data sheet. The trigger level is determined by a programmable BODLEVEL setting, see Table 9-2.

Table 9-2. Programmable BODLEVEL setting

BODLEVEL[2:0] (1)	V_{BOT}	UNIT
111	1.6	V
110	1.8	
101	2.0	
100	2.2	
011	2.4	
010	2.7	
001	2.9	
000	3.2	

Notes: 1. The values here are nominal values only. For typical, maximum and minimum numbers consult the device data sheet.

2. Changing these fuse bits will have no effect until leaving programming mode.

The BOD circuit has 3 modes of operation:

- **Disabled:** In this mode there is no monitoring of the V_{CC} level, and hence it is only recommended for applications where the power supply is stable.
- **Enabled:** In this mode the V_{CC} level is continuously monitored, and a drop in V_{CC} below V_{BOT} for at least t_{BOD} will give a brown-out reset.

- **Sampled:** In this mode the BOD circuit will sample the VCC level with a period identical to the 1 kHz output from the Ultra Low Power (ULP) 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 2 positive edges of the 1 kHz ULP output will not be detected. If a brown-out 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 brown-out detection setting for all sleep modes except idle mode.

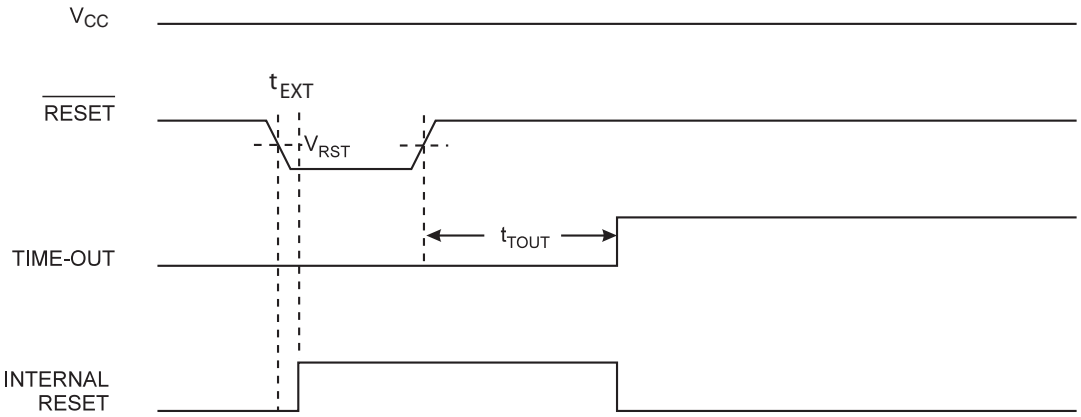
Table 9-3. BOD setting Fuse Decoding

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 reset pin includes an internal pull-up resistor.

Figure 9-5. External reset characteristics.

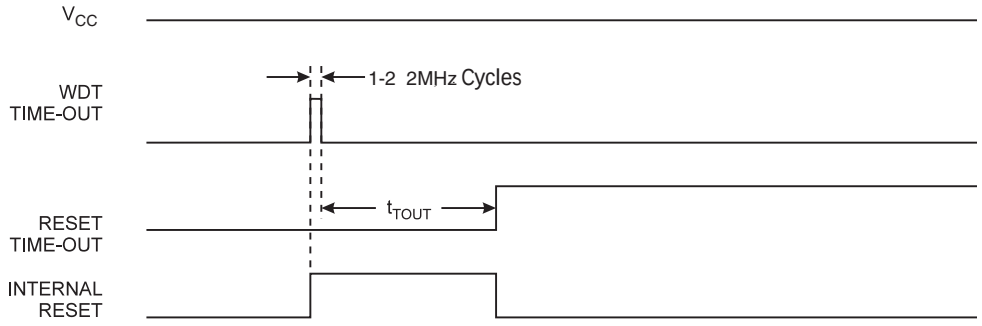


For characterization data on V_{RST} and t_{EXT} and pull-up resistor values consult the device data sheet.

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 1-2 clock cycles on the 2 MHz internal RC oscillator.

Figure 9-6. Watchdog reset.

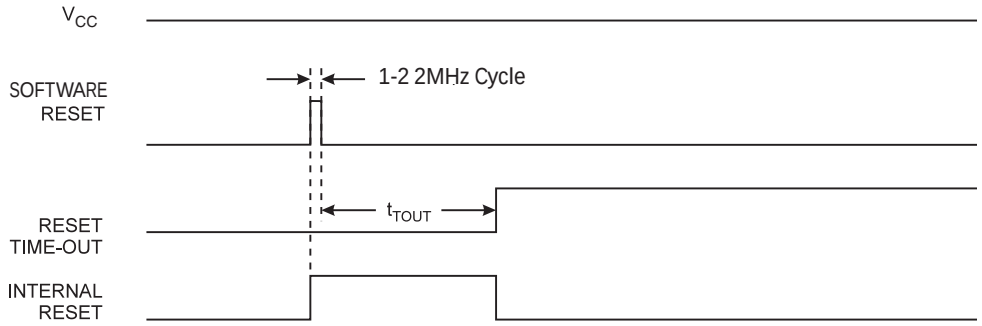


For information on configuration and use of the WDT, refer to the Section 11. ”WDT – Watchdog Timer” on page 117.

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 1-2 system clock CPU cycles after writing the bit. It is not possible to execute any instruction from a software reset is requested and 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 only accessible from debuggers and programmers.

9.5 Register Description

9.5.1 STATUS - Reset Status Register

Bit	7	6	5	4	3	2	1	0	
+0x00	–	–	SRF	PDIRF	WDRF	BORF	EXTRF	PORF	STATUS
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 reserved and will always be read as zero. 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: Brown Out Reset Flag**

This flag is set if a Brown out 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 - Reset Control Register

Bit	7	6	5	4	3	2	1	0	
+0x01	–	–	–	–	–	–	–	SWRST	CTRL
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, for details refer to Section 3.12 ”Configuration Change Protection” on page 12.

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	109
+0x01	CTRL	–	–	–	–	–	–	–	SWRST	109

10. Battery Backup System

10.1 Features

- **Battery Backup voltage supply from dedicated V_{BAT} power pin for:**
  - **One Ultra Low-power 32-bit Real Time Counter**
  - **One 32.768 kHz crystal oscillator with failure detection monitor**
  - **Two Battery Backup Registers**
- **Automatic switching from main power to battery backup power at:**
  - **Brown-Out Detection (BOD)**
- **Automatic switching from battery backup power to main power:**
  - **Device reset after Brown-Out Reset (BOR) is released**
  - **Device reset after Power-On Reset (POR) and BOR is released**

10.2 Overview

The Battery Backup System includes functionality that enables automatic power switching between main power and a battery backup power. Figure 10-1 on page 112 shows an overview of the system.

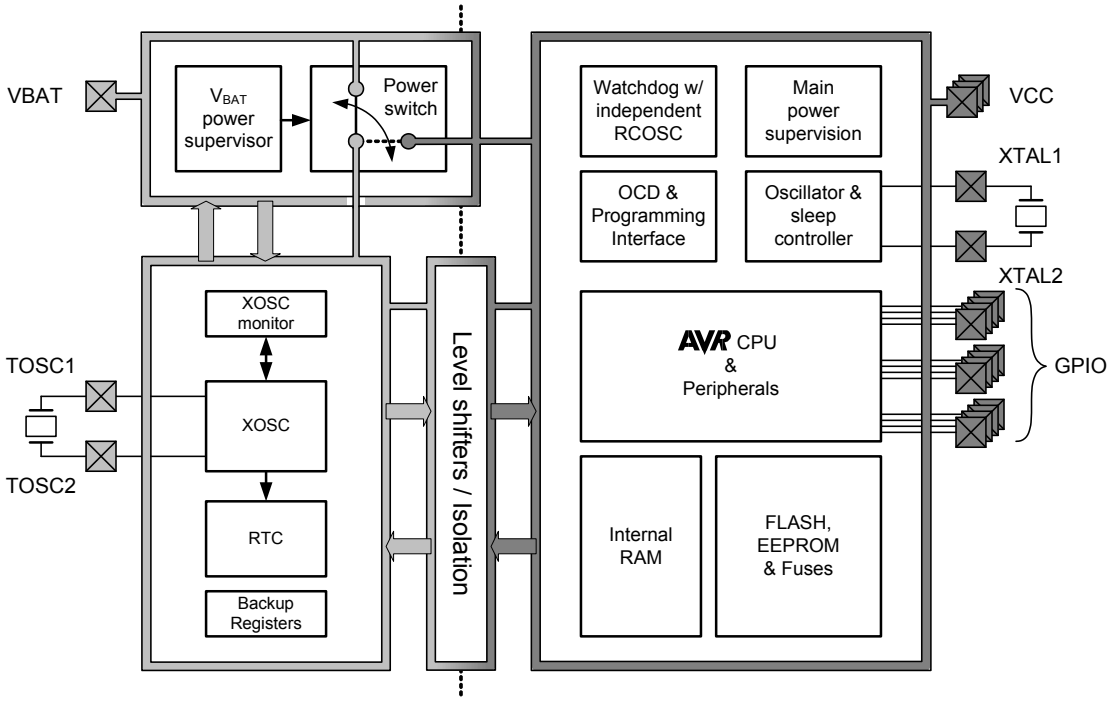
The Battery Backup Module supports connection of a backup battery to the dedicated V_{BAT} power pin. This will ensure power to the 32-bit Real Time Counter, a 32.768 kHz crystal oscillator with failure detection monitor and two battery backup registers when the main power source is unavailable.

Upon main power loss the device will detect this and automatically switch the Battery Backup Module to be powered from the V_{BAT} pin. After main power has been restored and both main POR and BOR are released, the Battery Backup Module will automatically switch back to be powered from main power again. The main BOD is used to detect the Vcc voltage levels, and this must be enabled for the power switching to work.

The 32-bit Real Time Counter (RTC) must be clocked from the 1 Hz or 1.024 kHz output of a 32.768 kHz crystal oscillator connected between the TOSC1 and TOSC2 pins when running from V_{BAT}. For more details on the 32-bit RTC refer to the “32-bit Real Time Counter” section in the XMEGA A Manual.

10.3 Battery Backup Module

Figure 10-1. Battery Backup Module and its power domain implementation



The Battery Backup Module consists of the following:

- A V_{BAT} power supervisor that ensures correct operation and detection of falling power in the backup battery power domain. It includes
  - A Battery Backup Power-On Detector (BBPOD).
  - A sampled Battery Backup Brown-Out Detector (BBBOD).
- A Power-switch for switching power to the battery backup domain between main power and the V_{BAT} pin.
- A 32.768 kHz Crystal Oscillator (XOSC) with 32.768 kHz, 1.024 kHz and 1 Hz output.
- A XOSC Failure Monitor for detecting failure on the Crystal Oscillator.
- A 32-bit Real Time Counter.
- Two Battery Backup registers.

The main BOD is used to detect Brown-Out on the main power so this must be enabled for the power switching to work.

When main BOD is detected, the Battery Backup Module will switch to be powered from the V_{BAT} pin. When main power is restored and/or the device is reset, software must enable access to the Battery Backup Module by writing a dedicated Access Enable bit in the module.

10.3.1 Battery Backup Power-On Detection

The Battery Backup Power-On Detection circuit will set the POD Flag (BBPODF) when power is connected to the V_{BAT} pin.

The the 32-kHz crystal oscillator, oscillator failure detection and Real Time Counter must be enabled from software before they can be used.

10.3.2 Battery Backup Brown-Out Detection

The Battery Backup Brown-Out Detector (BBBOD) ensures detection of falling power in the Battery Backup Module before the oscillator stops. When the V_{BAT} pin voltage drops below the BBBOD threshold voltage, the BOD Flag (BBBODF) will be set.

The BBBOD is sampled from the 1 Hz output of an ultra low power oscillator, and only designed for detecting slow changes in the V_{BAT} pin voltage level.

The BBBOD is turned off when the device runs from the main power. It is enabled when the Battery Backup Module is switched to be powered from the V_{BAT} pin after a main power BOD.

10.4 Main Power Loss

When the main power drops below the programmed BOD threshold voltage, the device will:

1. Switch the Battery Backup Module to be powered from the V_{BAT} pin. The BBBOD is enabled.
2. Ignore any input signals to the Battery Backup Module.
3. The part of the device that is only powered from V_{CC} will be reset.

10.5 Main Reset and Start-up Sequence

The Battery Backup System adds some considerations regarding main reset and start-up sequence. At every start-up after main reset the software should:

1. Check for sufficient power on V_{BAT} pin by reading the BBPWR flag.
2. Check the Battery Backup Module status by reading the BBPODF and BBBODF flags.
3. Enable access to the Battery Backup Module, by setting the Access Enable (ACCEN) bit.

Based on the status in Battery Backup Module, two different things should be done after the initial sequence. Each is described in the two sections below.

10.5.1 Battery Backup enabled

If BBPODF and/or BBBODF is not set, it indicates that the Battery Backup Modules has not had any power loss. Then the device should:

1. Set the Access Enable (ACCEN) bit
2. Check for 32-Khz Crystal Oscillator failure by reading the XOSC Failure flag.

If the XOSC Failure flag is cleared no further actions is required.

If the XOSC Failure flag is set, this indicates a failure on the external oscillator in the Battery Backup Module. The software should assume that the RTC counter value is invalid and take appropriate action.

10.5.2 Battery Backup failed and not enabled

If BBPODF and/or BBBODF is set this indicate that voltage on the V_{BAT} pin has had a drop sometime during the period when the rest of the device was unpowererd. The following procedure should be followed:

1. Set the ACCEN bit and apply a RESET
1. Enable the Crystal oscillator.
2. Wait until Crystal Oscillator ready flag is set.
3. Enable XOSC Fault Detection.
4. Configure and enable the RTC32.

10.6 Register Description

10.6.1 CTRL: Battery Backup Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	XOSCSEL	XOSCEN	XOSCFDEN	ACCEN	RESET	CTRL
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Reset 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- XOSCSEL: 32-kHz Crystal Oscillator Output Selection**

This bit selects which clock output from the 32.768 kHz crystal oscillator that will be used as clock input for the 32-bit Real Time Counter (RTC).

By default this bit is zero, and the 1 Hz clock output is used as input for the RTC. Setting this bit will select the 1.024 kHz clock output as input for the RTC. The Crystal Oscillator must be enabled before the output is available. This bit cannot be changed when XOSCEN is set.

- **Bit 3- XOSCEN: Crystal Oscillator Enable**

Setting this bit will enable the 32.768 kHz crystal oscillator, and thSetting this bite default 1 Hz output from the oscillator. Writing the bit to zero will have no effect. The oscillator will remain enabled until a Battery Backup RESET is issued.

- **Bit 2- XOSCFDEN: Crystal Oscillator Failure DSetting this bitetection Enable**

Setting this bit will enable the 32.768 kHz crystal oscillator monitor. Writing the bit to zero will have no effect. When enabled, the Crystal Oscillator Monitor will remain enabled until a Battery Backup RESET is issued.

- **Bit 1- ACCEN: Battery Backup Module Access Enable**

Setting this bit will enable access to the Battery Backup Module. After main reset this bit must be set in order to access (read from and write to) the battery backup domain functions and register settings, except for the BBPODF, the BBBODF and the BBPWR flag.

- **Bit 0- RESET: Battery Backup Reset**

Setting this bit will force a reset of the Battery Backup Module. Writing the bit to zero will have no effect. Writing a one to the XOSCEN or the XOSCFDEN bit the same time will block writing to this bit. When this bit is set XOSCEN, XOSCFDEN, XOSCSEL, and XOSCRDY in STATUS will be cleared.

These bits are protected by the Configuration Change Protection mechanism, for detailed description refer to ”Configuration Change Protection” on page 12.

10.6.2 STATUS: Battery Backup Status Register

Bit	7	6	5	4	3	2	1	0	
+0x01	BBPWR	-	-	-	XOSCRDY	XOSCFAIL	BBBODF	BBPODF	STATUS
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	x	x	0	0	

- **Bit 7 - BBPWR: Battery Backup Power**

The Power on the V_{BAT} pin is automatically checked each time the main domain is powered. The BBPWR flag will be set to one when no V_{BAT} pin power is detected. The flag can be cleared by writing a one to its bit location.

- **Bit [6:4] - Reserved**

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

- **Bit 3- XOSCRDY: Crystal Oscillator Ready**

When the 32 kHz crystal oscillator is stable and ready to be used, this flag is set. This flag can be cleared by writing a one to its bit location.

- **Bit 2 - XOSCFAIL: Crystal Oscillator Failure**

If the 32 kHz crystal oscillator has had an failure, the flag will be read as one. This flag can be cleared by applying a RESET to the battery backup module.

- **Bit 1 - BBBODF: Battery Backup Brown-Out Detection Flag**

This flag is read one if a Battery Backup BOD is detected when the Battery Backup Module is powered from the V_{BAT} pin. This flag can be cleared by writing a one to its bit location. This flag is not valid when BBPWR is set.

- **Bit 0 - BBPODF: Battery Backup Power-On Detection Flag**

This flag is set if a Battery Backup Power-On is detected, i.e when power is connected to V_{BAT}. This flag can be cleared by writing a one to its bit location. This flag is not valid when BBPWR is set.

10.6.3 BACKUP0: Battery Backup Register 0

Bit	7	6	5	4	3	2	1	0	
+0x02	BACKUP0[7:0]								BACKUP0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	x	x	x	x	x	x	x	x	

- **Bit [7:0] - BACKUP0: Battery Backup Register 0**

This register can be used to store data in the Battery Backup Module before the main power is lost or removed.

10.6.4 BACKUP1: Battery Backup Register 1

Bit	7	6	5	4	3	2	1	0	
+0x03	BACKUP1[7:0]								BACKUP1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	x	x	x	x	x	x	x	x	

- **Bit [7:0] - BACKUP0: Battery Backup Register 1**

This register can be used to store data in the Battery Backup Module before the main power is lost or removed.

10.7 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL				XOSCSEL	XOSCEN	XOSCFDEN	ACCEN	RESET	114
+0x01	STATUS	BBPWR				XOSCRDY	OSCFAIL	BBBODF	BBPODF	115
+0x02	BACKUP0	BACKUP0[7:0]								115
+0x03	BACKUP1	BACKUP1[7:0]								116
+0x04										
+0x05										
+0x06										
+0x07										

11. WDT – Watchdog Timer

11.1 Features

- **11 selectable timeout period, from 8 ms to 8s**
- **Two operation modes**
  - **Standard mode**
  - **Window mode**
- **Runs from 1 kHz Ultra Low Power clock reference**
- **Configuration lock**

11.2 Overview

The Watchdog Timer (WDT) is a system function for monitoring correct program operation, making it possible to recover from error situations, for instance run-away 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 system reset. The WDT is reset by executing the WDR (Watchdog Timer Reset) instruction from the application code.

The WDT also has a window mode that enables the user to define a time slot where WDT must be reset within. If the WDT is reset too early or too late, a system reset will be issued.

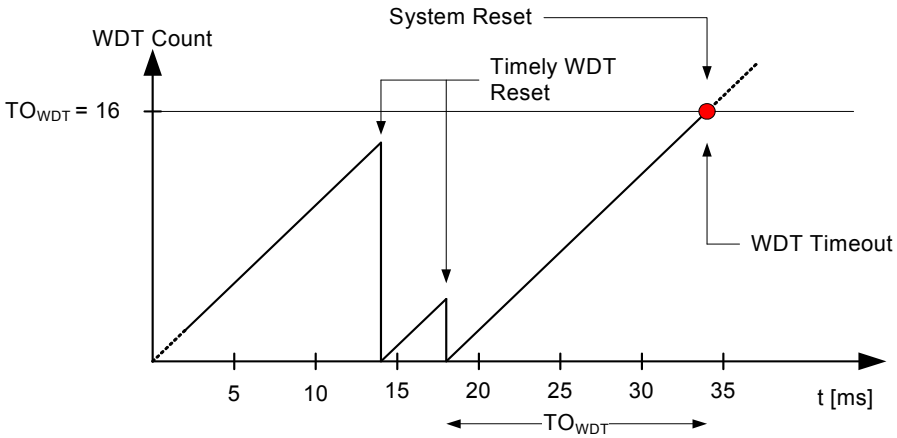
The WDT will run in all power modes if enabled. It runs from a CPU independent clock source, and will continue to operate to issue a system reset even if the main clocks fail.

The Configuration Change Protection mechanism ensures that the WDT settings cannot be changed by accident. In addition the settings can be locked by a fuse.

11.3 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 the WDT will issue a system reset. There are 11 possible WDT timeout (TO_{WDT}) periods selectable from 8 ms to 8s, and the WDT can be reset at any time during the period. After each reset, a new timeout period is started. The default timeout period is controlled by fuses. Normal mode operation is illustrated in Figure 11-1.

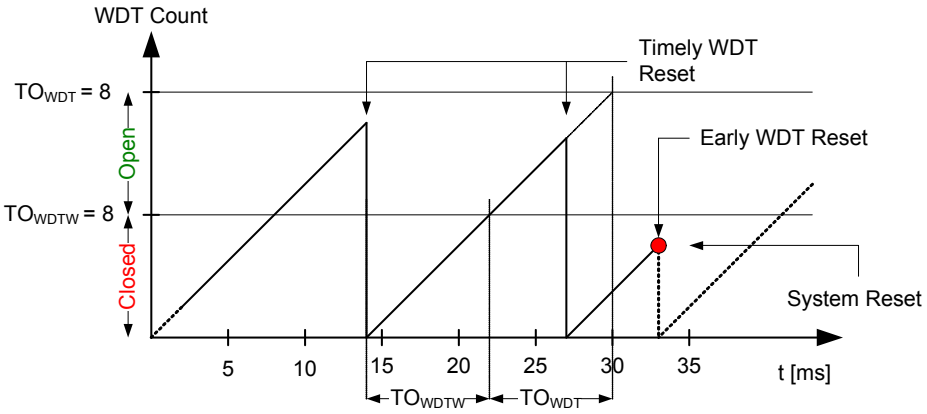
Figure 11-1. Normal mode operation.



11.4 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 from 8 ms to 8s where the WDT cannot be reset: if the WDT is reset in this period the WDT will issue a system reset. The normal WDT timeout period, which is also 8 ms to 8s, defines the duration of the "open" period, in which the WDT can (and should) be reset. The open period will always follow the closed period, 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. The window mode operation is illustrated in Figure 11-2.

Figure 11-2. Window mode operation.



11.5 Watchdog Timer clock

The WDT is clocked from the 1 kHz output from the internal 32 kHz Ultra Low Power (ULP) oscillator. Due to the ultra low power design, the oscillator is not very accurate 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 the ULP oscillator accuracy, consult the device data sheet.

11.6 Configuration Protection and Lock

The WDT is designed with two security mechanisms to avoid unintentional changes of 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 is to lock the configuration by setting the WDT lock fuse. When this fuse is set, the Watchdog Time Control Register can not be changed, hence the WDT can not be disabled from software. After system reset the WDT will resume at 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.

11.7 Registers Description

11.7.1 CTRL – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	PER[3:0]				ENABLE	CEN	CTRL
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]: Watchdog Timeout Period**

These bits determine the Watchdog timeout period as a number of 1 kHz ULP oscillator cycles. In window mode operation, these bits define the open window period. The different typical time-out periods are found in Table 11-1. The initial values of these bits are set by the Watchdog Timeout Period (WDP) fuses, and will be 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 detailed description refer to Section 3.12 ”Configuration Change Protection” on page 12.

Table 11-1. Watchdog timeout periods

PER[3:0]	Group Configuration	Typical timeout periods
0000	8CLK	8 ms
0001	16CLK	16 ms
0010	32CLK	32 ms
0011	64CLK	64 ms
0100	125CLK	0.125 s
0101	250CLK	0.25 s
0110	500CLK	0.5 s
0111	1KCLK	1.0 s
1000	2KCLK	2.0 s
1001	4KCLK	4.0 s
1010	8KCLK	8.0 s
1011		Reserved
1100		Reserved
1101		Reserved
1110		Reserved
1111		Reserved

- **Bit 1 - ENABLE: Watchdog Enable**

This bit enables the WDT.

In order to change this bit the CEN bit in ”CTRL – Watchdog Timer Control Register” on page 119 must be written to one at the same time. This bit is protected by the Configuration Change Protection mechanism, for detailed description refer to Section 3.12 ”Configuration Change Protection” on page 12.

- **Bit 0 - CEN: Watchdog Change Enable**

This bit enables the possibility to change the configuration of the ”CTRL – Watchdog Timer Control Register” on page 119. 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 detailed description refer to Section 3.12 ”Configuration Change Protection” on page 12.

11.7.2 WINCTRL – Window Mode Control Register

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	WPER[3:0]				WEN	WCEN	WINCTRL
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	

- **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 - WPER[3:0]: Watchdog Window Mode Timeout Period**

These bits determine the closed window period as a number of 1 kHz ULP oscillator cycles in window mode operation. The typical different closed window periods are found in Table 11-2. The initial values of these bits are set by the Watchdog Window Timeout Period (WDWP) fuses, and will be 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 detailed description refer to Section 3.12 ”Configuration Change Protection” on page 12.

Table 11-2. Watchdog closed window periods

WPER[3:0]	Group Configuration	Typical closed window periods
0000	8CLK	8 ms
0001	16CLK	16 ms
0010	32CLK	32 ms
0011	64CLK	64 ms
0100	125CLK	0.125 s
0101	250CLK	0.25 s

Table 11-2. Watchdog closed window periods (Continued)

WPER[3:0]	Group Configuration	Typical closed window periods
0110	500CLK	0.5 s
0111	1KCLK	1.0 s
1000	2KCLK	2.0 s
1001	4KCLK	4.0 s
1010	8KCLK	8.0 s
1011		Reserved
1100		Reserved
1101		Reserved
1110		Reserved
1111		Reserved

- **Bit 1 - WEN: Watchdog Window Mode Enable**

This bit enables the Watchdog Window Mode. In order to change this bit the WCEN bit in ”WINCTRL – Window Mode Control Register” on page 120 must be written to one at the same time. This bit is protected by the Configuration Change Protection mechanism, for detailed description refer to Section 3.12 ”Configuration Change Protection” on page 12.

- **Bit 0 - WCEN: Watchdog Window Mode Change Enable**

This bit enables the possibility to change the configuration of the ”WINCTRL – Window Mode Control Register” on page 120. 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.

11.7.3 STATUS – Watchdog Status Register

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	-	-	-	SYNCBUSY	STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:1 - Res:Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 - SYNCBUSY**

When writing to the CTRL or WINCTRL registers, the WDT needs to be synchronized to the other clock domains. During synchronization the SYNCBUSY bit will be read as one. This bit is automatically cleared after the synchronization is finished. Synchronization will only take place when the ENABLE bit for the Watchdog Timer is set.

11.8 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	119
+0x01	WINCTRL	-	-	WPER[3:0]				WEN	WCEN	120
+0x02	STATUS	-	-	-	-	-	-	-	SYNCBUSY	121

12. Interrupts and Programmable Multi-level Interrupt Controller

12.1 Features

- **Separate interrupt vector for each interrupt**
- **Short, predictable interrupt response time**
- **Programmable Multi-level Interrupt Controller**
  - **3 programmable interrupt levels**
  - **Selectable priority scheme within low level interrupts (round-robin or fixed)**
  - **Non-Maskable Interrupts (NMI)**
- **Interrupt vectors can be moved to the start of the Boot Section.**

12.2 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. When the interrupt is enabled and the interrupt condition is present this will generate a corresponding interrupt request. All interrupts have a separate interrupt vector address.

The Programmable Multi-level Interrupt Controller (PMIC) controls the handling of interrupt requests, and prioritizing between the different interrupt levels and interrupt priorities. 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 or high. 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.

12.3 Operation

Interrupts must be globally enabled for any interrupts to be generated. This is done by setting the global interrupt enable bit (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 right state.

12.4 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 to 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 for details on lock bit settings.

Interrupts are automatically disabled for up to 4 CPU clock cycles when the Configuration Change Protection register is written with the correct signature, refer to Section 3.12 ”Configuration Change Protection” on page 12 for more details.

12.4.1 NMI – Non-Maskable Interrupts

Non-Maskable Interrupts (NMI) are hardwired. It is not selectable which interrupts represent NMI and which represent regular interrupts. Non-Maskable Interrupts must be enabled before they can be used. Refer to the device data sheet for NMI present on each the device.

A 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 interrupt handler. If more than one NMI is requested at the same time, priority is static according to interrupt vector address where lowest address has highest priority.

12.4.2 Interrupt Response Time

The interrupt response time for all the enabled interrupts is five CPU clock cycles minimum. During these five clock cycles the program counter is pushed on the stack. After five clock cycles, 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 multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the device is in sleep mode, the inter-

rupt 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 five clock cycles. During these five clock cycles, the program counter is popped from the stack and the stack pointer is incremented.

12.5 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 12-1.

Table 12-1. Interrupt level

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 on 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.

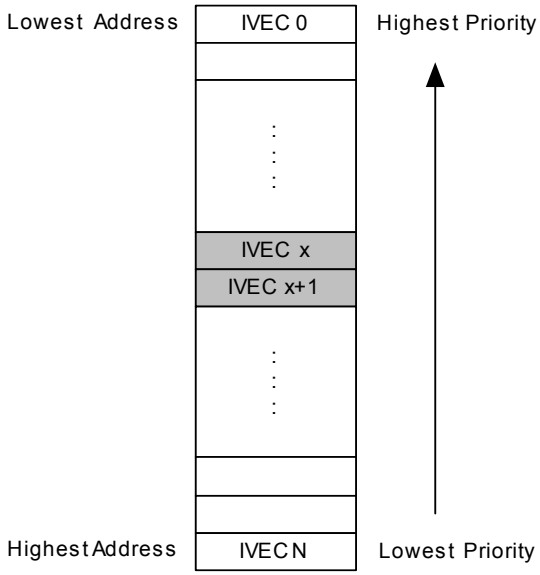
12.6 Interrupt priority

Within each interrupt level, all interrupts have a priority. When several interrupt requests are pending, the order of 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.

12.6.1 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 data sheet for interrupt vector table with the base address for all modules and peripherals with interrupt. 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.

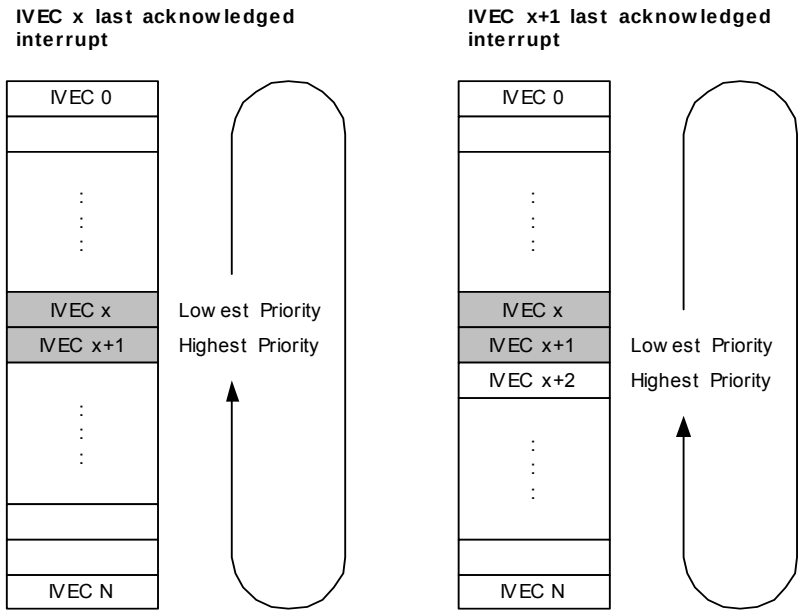
Figure 12-1. Static priority.



12.6.2 Round-robin scheduling

To avoid the possible starvation problem for low level interrupts with static priority, the PMIC gives the possibility for 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 next time one or more interrupts from the low level is requested.

Figure 12-2. Round-robin scheduling.



12.7 Moving Interrupts Between Application and Boot Section

The interrupt vectors can be moved from the default location in the Application Section in Flash to the start of the Boot Section.

12.8 Register Description

12.8.1 STATUS - PMIC Status Register

Bit	7	6	5	4	3	2	1	0	
+0x00	NMIEX	-	-	-	-	HILVLEX	MEDLVLEX	LOLVLEX	STATUS
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 reserved and will always be read as zero. 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 if a high level interrupt is executing or 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 if a medium level interrupt is executing or 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 if a low level interrupt is executing or 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.

12.8.2 INTPRI - PMIC Priority Register

Bit	7	6	5	4	3	2	1	0	
+0x01	INTPRI[7:0]								INTPRI
Read/Write	R/W	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 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, so if default static priority is needed the register must be written to zero.

12.8.3 CTRL - PMIC Control Register

Bit	7	6	5	4	3	2	1	0	
+0x02	RREN	IVSEL	-	-	-	HILVLEN	MEDLVLEN	LOLVLEN	CTRL
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 moved to the beginning of the Boot section of the Flash. Refer to the device data sheet for the absolute address.

This bit is protected by the Configuration Change Protection mechanism, refer to Section 3.12 ”Configuration Change Protection” on page 12 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**

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**

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**

When this bit is set all low level interrupts are enabled. If this bit is cleared, low level interrupt requests will be ignored.

12.9 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	127
+0x01	INTPRI	INTPRI[7:0]								127
+0x02	CTRL	RREN	IVSEL	-	-	-	HILVLEN	MEDLVLEN	LOLVLEN	128

13. I/O Ports

13.1 Features

- Selectable input and output configuration for each pin individually
- Flexible pin configuration through dedicated Pin Configuration Register
- Synchronous and/or asynchronous input sensing with port interrupts and events
- Asynchronous wake-up signalling
- Highly configurable output driver and pull settings:
  - Totem-pole
  - Pull-up/-down
  - Wired-AND
  - Wired-OR
  - Bus keeper
  - Inverted I/O
- Slew rate control
- Flexible pin masking
- Configuration of multiple pins in a single operation
- Read-Modify-Write (RMW) support
- Toggle/clear/set registers for OUT and DIR registers
- Clock output on port pin
- Event Channel 0 output on port pin 7
- Mapping of port registers (virtual ports) into bit accessible I/O memory space

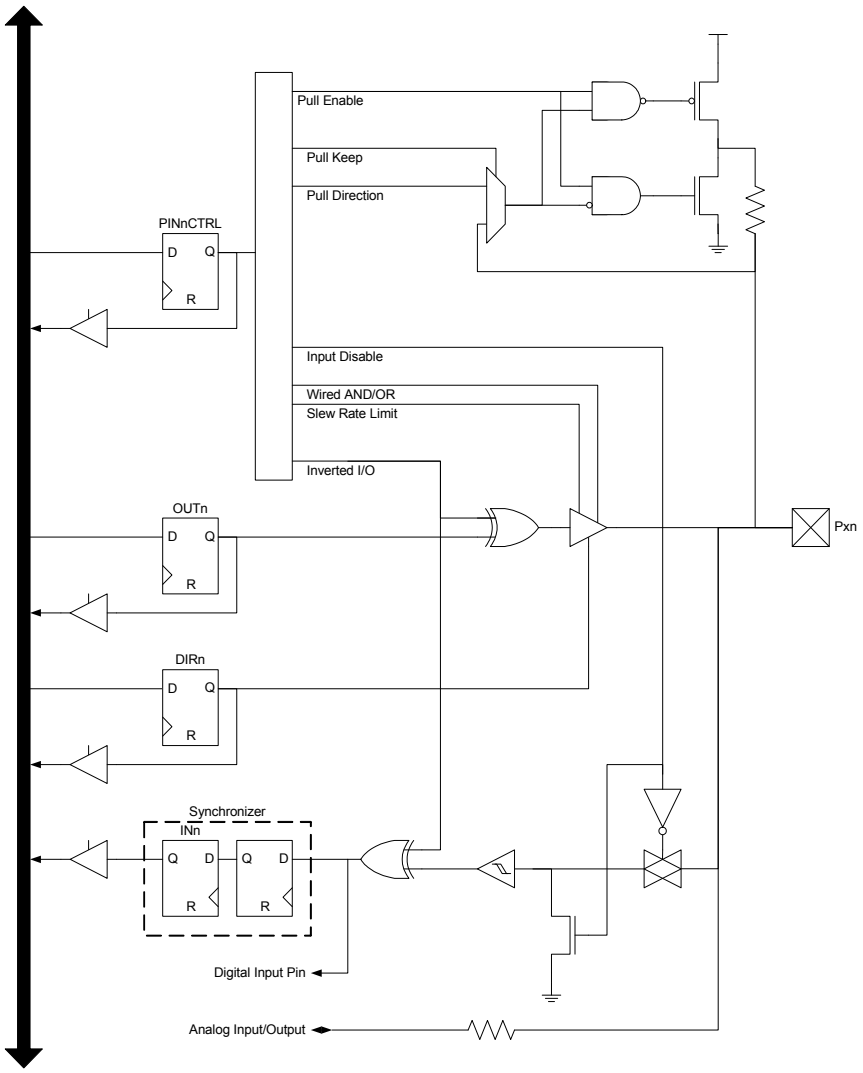
13.2 Overview

XMEGA has flexible General Purpose I/O (GPIO) Ports. A port consists of up to 8 pins ranging from pin 0 to 7, where each pin can be configured as input or output with highly configurable driver and pull settings. The ports also implement several functions including interrupts, synchronous/asynchronous input sensing and asynchronous wake-up signalling.

All functions are individual per pin, but several pins may be configured in a single operation. All ports have true Read-Modify-Write (RMW) functionality when used as general purpose I/O ports. The direction of one port pin can be changed without unintentionally changing the direction of any other pin. The same applies when changing drive value when configured as output, or enabling/disabling of pull-up or pull-down resistors when configured as input.

Figure 13-1 on page 130 shows the I/O pin functionality, and the registers that is available for controlling a pin.

Figure 13-1. General I/O pin functionality.



13.3 Using the I/O Pin

Use of an I/O pin is controlled from the user software. Each port has one Data Direction (DIR), Data Output Value (OUT) that is 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 the pin value. The pin value can always be read regardless of the pin being configured as input or output, except if digital input is disabled.

I/O pins are tri-stated when reset condition becomes active, even if no clocks are running.

13.4 I/O Pin Configuration

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 the pin.

For totem-pole output there are 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 only have active resistors 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 direction.

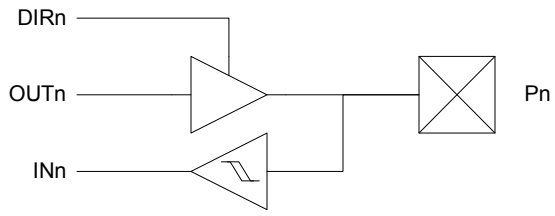
Since pull configuration is configured through the pin configuration register, all intermediate port states during switching of pin direction and pin values are avoided.

The I/O pin configurations are summarized with simplified schematics from Figure 13-2 on page 131 to Figure 13-7 on page 133.

13.4.1 Totem-pole

When configured for Totem-pole (push-pull), the pin is driven low or high according to the corresponding bit setting in OUT. 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 is connected.

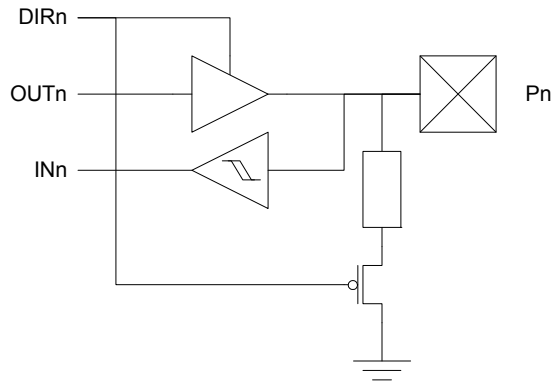
Figure 13-2. I/O pin configuration - Totem-pole (push-pull)



13.4.1.1 Totem-pole with Pull-down

In this mode, the configuration is as for Totem-pole, expect the pin is configured with internal pull-down when set as input.

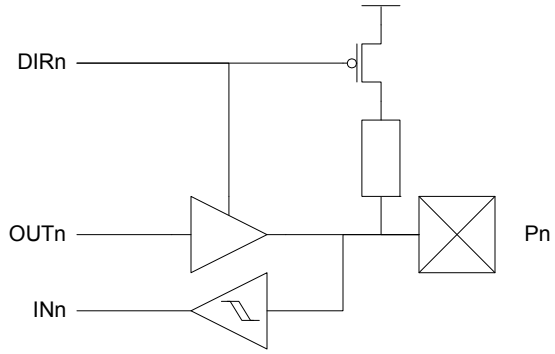
Figure 13-3. I/O pin configuration - Totem-pole with pull-down (on input)



13.4.1.2 *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 13-4. I/O pin configuration - Totem-pole with pull-up (on input).

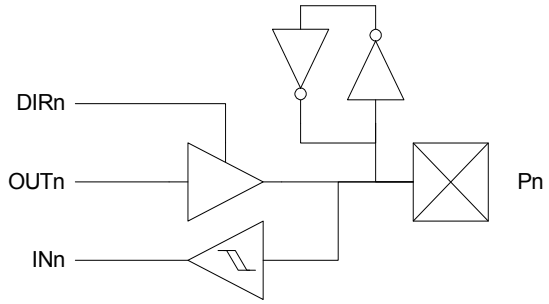


13.4.2 Bus-keeper

When the pin is configured for bus-keeper, it provides a weak bus-keeper that will keep the pin at its logic level when the pin is no longer driven to any logic state. If last level on the pin/bus was 1, the bus-keeper will use the internal pull resistor to keep the bus high. If last logic level on the pin/bus was 0, the bus-keeper will use the internal pull resistor to keep the bus high.

The bus-keeper's week output produces the same logical level as the last output level. It acts as a pull-up if the last level was '1', and pull-down if the last level was '0'.

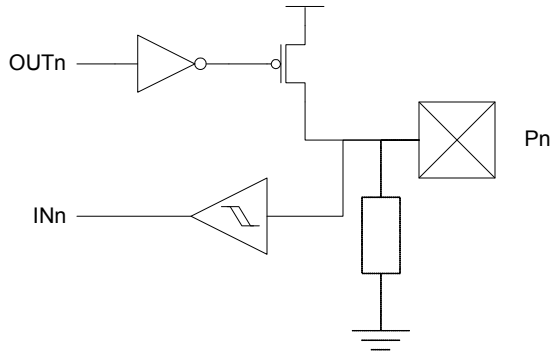
Figure 13-5. I/O pin configuration - Totem-pole with bus-keeper



13.4.3 Wired-OR

With Wired-OR configuration, the pin will be driven high when the corresponding bit in OUT is written to one. When OUT 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.

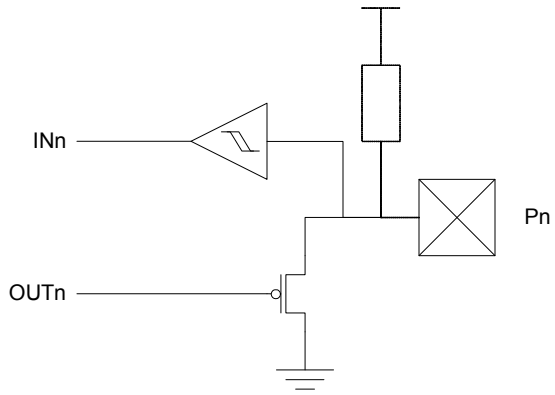
Figure 13-6. Output configuration - Wired-OR with optional pull-down.



13.4.4 Wired-AND

With Wired-AND configuration, the pin will be driven low when the corresponding bit in OUT is written to zero. When OUT is set to one, the pin is released allowing the pin to be pulled low 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.

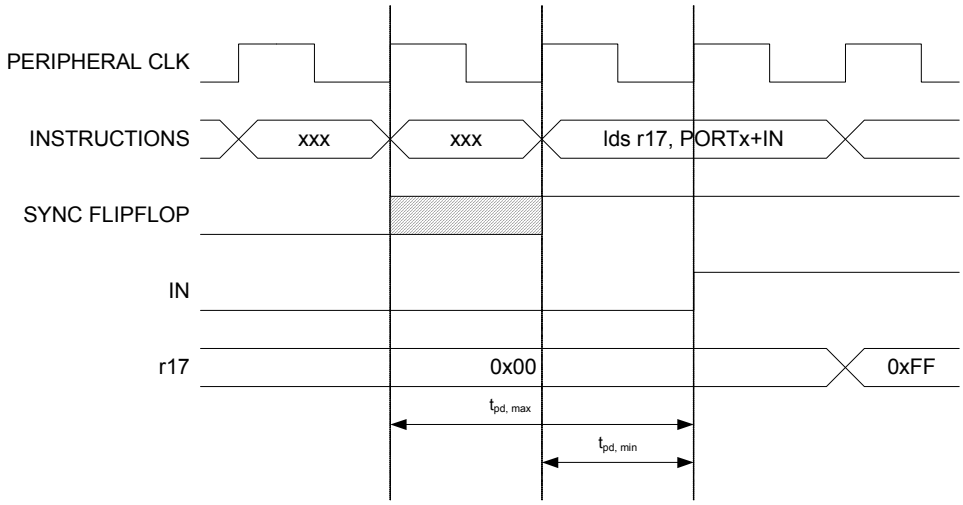
Figure 13-7. Output configuration - Wired-AND with optional pull-up.



13.5 Reading the Pin value

Independent of the pin data direction, the pin value can be read from the IN register as shown in Figure 13-1 on page 130. 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 is needed to avoid metastability if the physical pin changes value near the edge of the internal clock. The Synchronizer introduces a delay on the internal signal line. Figure 13-8 on page 134 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 13-8. Synchronization when reading an externally applied pin value.

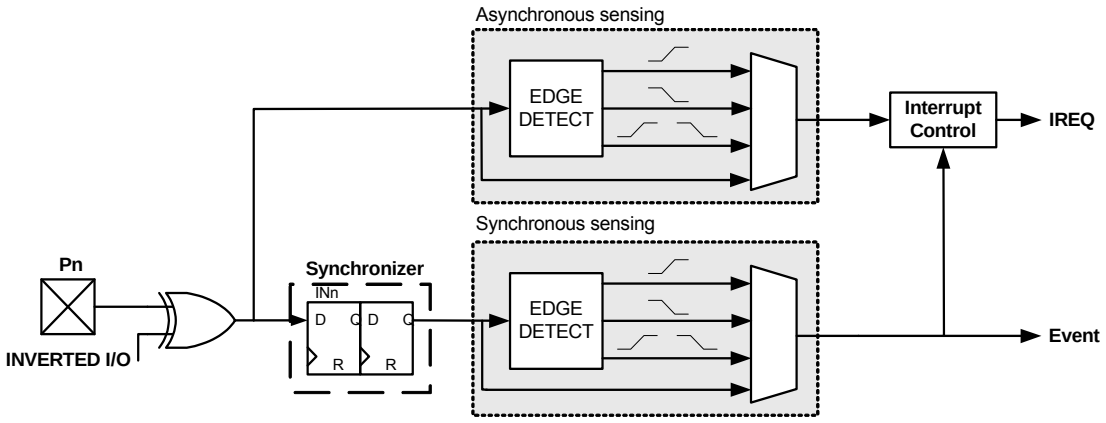


13.6 Input Sense Configuration

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 rising edge, falling edge or both edges, or detection of low level. High level can be detected by using inverted input. Input sensing can be used to trigger interrupt requests (IREQ) or s when there is a change on the pin.

The I/O pins support synchronous and asynchronous input sensing. Synchronous sensing requires presence of the peripheral clock, while asynchronous sensing does not require any clock.

Figure 13-9. Input sensing.



13.7 Port Interrupt

Each port has two interrupt vectors, and it is configurable which pins on the port that can be used to trigger each interrupt request. Port interrupts must be enabled before they can be used. Which sense configurations that can be used to generate interrupts is dependent on whether synchronous or asynchronous input sensing is used.

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 start-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 kept until the completion of the currently executing instructions for an interrupt to be generated. In all sleep modes the low level must be kept until the end of the device start-up time for an interrupt to be generated. If the low level disappears before the end of the start-up time, the device will still wake up, but no interrupt will be generated.

Table 13-1, Table 13-2, and Table 13-3 on page 136 summarizes when interrupts can be triggered for the various input sense configurations.

Table 13-1. Synchronous 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.

Table 13-2. 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.

Table 13-3. Limited asynchronous sense support

Sense settings	Supported	Interrupt description
Rising edge	No	-
Falling edge	No	-
Both edges	Yes	Pin value must be kept unchanged.
Low level	Yes	Pin-level must be kept unchanged.

13.8 Port Event

Port pins can generate an event when there is a change on the pin. The sense configurations decide when each pin will generate events. Event generation requires the presence of a peripheral clock, hence 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.

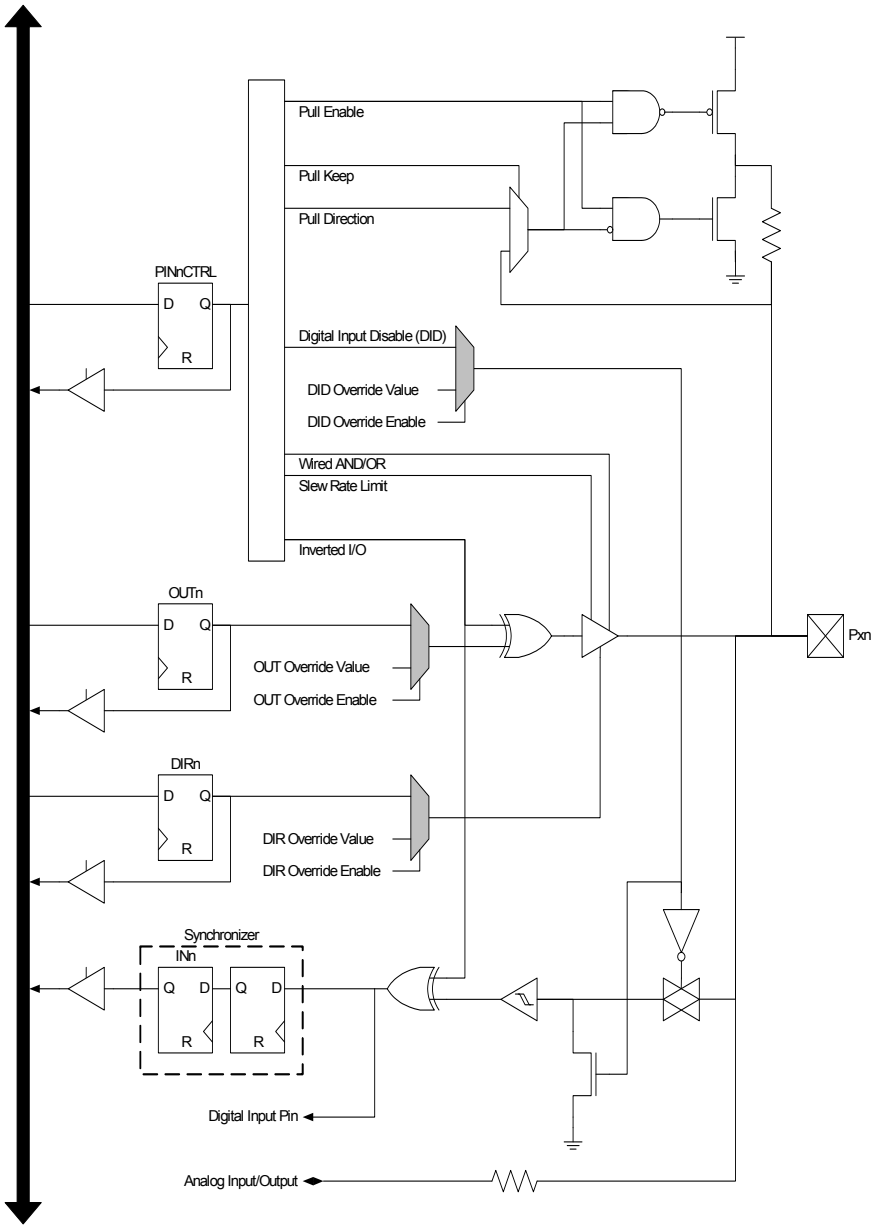
A level sensing, a low level pin value will not generate events, and a high pin value will continuously generate events. For events to be generated on low level, the pin configuration mst be set to inverted I/O.

13.9 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 this 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 section for that peripheral.

The port override signals and related logic (grey) is shown in Figure 13-10 on page 137. These signals are not accessible from software, but are internal signals between the overriding peripheral and the port pin.

Figure 13-10. Port override signals and related logic



13.10 Slew-rate Control

Slew-rate control can be enabled for all I/O pins individually. Enabling the slew rate limiter will typically increase the rise/fall time by 50-150% depending on voltage, temperature and load. For information about the characteristics of the slew-rate limiter, please refer to the device data sheet.

13.11 Clock and Event Output

It is possible to output both the Peripheral Clock and the signaling event from Event Channel 0 to pin. Output port pin is selected from software. If an event occur on Event Channel 0, this will

be visible on the port pin as long as the event last. Normally this is one peripheral clock cycle only.

13.12 Multi-configuration

MPCMASK can be used to set a bit mask for the pin configuration registers. When setting bit n in MPCMASK, PINnCTRL is added to the pin configuration mask. During the next write to any of the port's pin configuration registers, the same value will be written to all the port's pin configuration registers set by the mask. The MPCMASK register is cleared automatically after the write operation to the pin configuration registers is finished.

13.13 Virtual Port Registers

Virtual port registers allow for port registers in the extended I/O memory space to be mapped virtually in the I/O memory space. When mapping a port, writing to the virtual port register will be the same as writing to the real port register. This enables use of I/O memory specific instructions for bit-manipulation, and the I/O memory specific instructions IN and OUT on port register that normally resides in the extended I/O memory space. There are four virtual ports, so up to four ports can be mapped virtually at the same time. The mapped registers are IN, OUT, DIR and INTFLAGS.

13.14 Register Description – Ports

13.14.1 DIR - Data Direction Register

Bit	7	6	5	4	3	2	1	0	
+0x00	DIR[7:0]								DIR
Read/Write	R/W	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. 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.

13.14.2 DIRSET - Data Direction Set Register

Bit	7	6	5	4	3	2	1	0	
+0x01	DIRSET[7:0]								DIRSET
Read/Write	R/W	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.

13.14.3 DIRCLR - Data Direction Clear Register

Bit	7	6	5	4	3	2	1	0	
+0x02	DIRCLR[7:0]								DIRCLR
Read/Write	R/W	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.

13.14.4 DIRTGL - Data Direction Toggle Register

Bit	7	6	5	4	3	2	1	0	
+0x03	DIRTGL[7:0]								DIRTGL
Read/Write	R/W	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 on 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.

13.14.5 OUT - Data Output Value

Bit	7	6	5	4	3	2	1	0	
+0x04	OUT[7:0]								OUT
Read/Write	R/W	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 in 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.

13.14.6 OUTSET - Data Output Value Set Register

Bit	7	6	5	4	3	2	1	0	
+0x05	OUTSET[7:0]								OUTSET
Read/Write	R/W	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 on 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.

13.14.7 OUTCLR - Data Output Value Clear Register

Bit	7	6	5	4	3	2	1	0	
+0x06	OUTCLR[7:0]								OUTCLR
Read/Write	R/W	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 on 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.

13.14.8 OUTTGL - Data Output Value Toggle Register

Bit	7	6	5	4	3	2	1	0	
+0x07	OUTTGL[7:0]								OUTTGL
Read/Write	R/W	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 on 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.

13.14.9 IN - Data Input Value Register

Bit	7	6	5	4	3	2	1	0	
+0x08	IN[7:0]								IN
Read/Write	R/W	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 driver is enabled. INn shows the value of pin n on the port.

13.14.10 INTCTRL - Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
+0x09	-	-	-	-	INT1LVL[1:0]		INT0LVL[1:0]		INTCTRL
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 interrupt request for port interrupt n and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123.

13.14.11 INT0MASK - Interrupt 0 Mask Register

Bit	7	6	5	4	3	2	1	0	
+0x0A	INT0MSK[7:0]								INT0MASK
Read/Write	R/W	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 Register**

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.

13.14.12 INT1MASK - Interrupt 1 Mask Register

Bit	7	6	5	4	3	2	1	0	
+0x0B	INT1MSK[7:0]								INT1MASK
Read/Write	R/W	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 Register**

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.

13.14.13 INTFLAGS - Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
+0x0C	-	-	-	-	-	-	INT1IF	INT0IF	INTFLAGS
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 reserved and will always be read as zero. 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 according to the pin's input sense configuration occurs, 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.

13.14.14 PINnCTRL - Pin n Configuration Register

Bit	7	6	5	4	3	2	1	0	
	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			PINnCTRL
Read/Write	R/W	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 - SRLEN: Slew Rate Limit Enable**

Setting this bit will enable slew-rate limiting on pin n.

- **Bit 6 - INVEN: Inverted I/O Enable**

Setting this bit will enable inverting 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 13-4.

Table 13-4. Output/Pull Configuration

		Description	
OPC[2:0]	**Group Configuration**	**Output configuration**	**Pull configuration**
000	TOTEM	Totempole	(N/A)
001	BUSKEEPER	Totempole	Bus keeper
010	PULLDOWN	Totempole	Pull-down (on input)
011	PULLUP	Totempole	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 13-5. The sense configuration decides how the pin can trigger port interrupts and events. When the input buffer is not disabled, the schmitt triggered input is sampled (synchronized) and can be read in the IN register.

Table 13-5. 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)
100		Reserved
101		Reserved
110		Reserved
111	INTPUT_DISABLE	Input buffer disabled(2)

Note: 1. A low pin value will not generate events, and a high pin value will continuously generate events.
2. Only Port A - F supports the input buffer disable option.

13.15 Register Description – Multiport Configuration

13.15.1 MPCMASK - Multi-pin Configuration Mask Register

Bit	7	6	5	4	3	2	1	0	
+0x00	MPCMASK[7:0]								MPCMASK
Read/Write	R/W	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 several pins in a port to be configured at the same time. Writing a one to bit n allows that pin to be part of the multi-pin configuration. When a pin configuration is written to one of the PINnCTRL registers of the port, that value is written to all the PINnCTRL registers of the pins matching the bit pattern in the MPCMASK register for that port. It is not necessary to write to one of the registers that is set by the MPCMASK register. The MPCMASK register is automatically cleared after any PINnCTRL registers is written.

13.15.2 VPCTRLA - Virtual Port-map Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x02	VP1MAP[3:0]				VP0MAP[3:0]				VPCTRLA
Read/Write	R/W	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 13-6 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 13-6 for configuration.

13.15.3 VPCTRLB - Virtual Port-map Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x03	VP3MAP[3:0]				VP2MAP[3:0]				VPCTRLB
Read/Write	R/W	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 13-6 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 13-6 for configuration.

Table 13-6. 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

Table 13-6. Virtual Port mapping. (Continued)

VPnMAP[3:0]	Group Configuration	Description
1101	PORTP	PORTP mapped to virtual Port n
1110	PORTQ	PORTQ mapped to virtual Port n
1111	PORTR	PORTR mapped to virtual Port n

13.15.4 CLKEVOUT - Clock and Event Out Register

Bit	7	6	5	4	3	2	1	0	
+0x04	-	-	EVOUT[1:0]		-	-	CLKOUT[1:0]		CLKEVOUT
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 reserved and will always be read as one. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 5:4 - EVOUT[1:0] - Event Output Port**

These bits decide which port the Event Channel 0 from the Event System should be output to. Pin 7 on the selected port is always used, and the CLKOUT bits must be set different from EVOUT. The pin must be configured as an output pin for the Signaling Event to be available on the pin.

Table 13-7 on page 145 shows the possible configurations.

Table 13-7. Event Channel 0 output configurations

EVOUT[1:0]	Group Configuration	Description
00	OFF	Event out disabled
01	PC7	Event Channel 0 output on Port C pin 7
10	PD7	Event Channel 0 output on Port D pin 7
11	PE7	Event Channel 0 output on Port E pin 7

- **Bits 3:2 - Reserved**

These bits are reserved and will always be read as one. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1:0 - CLKOUT[1:0] - Clock Output Port**

These bits decide which port the Peripheral Clock should be output to. Pin 7 on the selected port is always used. The Clock output setting, will override the Event output setting, thus if both are enabled on the same port pin, the Peripheral Clock will be visible. The pin must be configured as an output pin for the Clock to be available on the pin.

Table 13-8 on page 146 shows the possible configurations.

Table 13-8. Clock output configurations

CLKOUT[1:0]	Group Configuration	Description
00	OFF	Clock out disabled
01	PC7	Clock output on Port C pin 7
10	PD7	Clock output on Port D pin 7
11	PE7	Clock output on Port E pin 7

13.16 Register Description – Virtual Port

13.16.1 DIR - Data Direction

Bit	7	6	5	4	3	2	1	0	
+0x00	DIR[7:0]								DIR
Read/Write	R/W	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 Register**

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.

13.16.2 OUT - Data Output Value

Bit	7	6	5	4	3	2	1	0	
+0x01	OUT[7:0]								OUT
Read/Write	R/W	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.

13.16.3 IN - Data Input Value

Bit	7	6	5	4	3	2	1	0	
+0x02	IN[7:0]								IN
Read/Write	R/W	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 B" 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.

13.16.4 INTFLAGS - Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	-	-	INT1IF	INT0IF	INTFLAGS
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 reserved and will always be read as zero. 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 according to the pin's input sense configuration occurs, 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 the flags mapped. When a port is mapped as virtual, accessing this register is identical to accessing the actual INTFLAGS register for the port.

13.17 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]								138
+0x01	DIRSET	DIRSET[7:0]								138
+0x02	DIRCLR	DIRCLR[7:0]								139
+0x03	DIRTGL	DIRTGL[7:0]								139
+0x04	OUT	OUT[7:0]								139
+0x05	OUTSET	OUTSET[7:0]								139
+0x06	OUTCLR	OUTCLR[7:0]								140
+0x07	OUTTGL	OUTTGL[7:0]								140
+0x08	IN	IN[7:0]								140
+0x09	INTCTRL	-	-	-	-	INT1LVL[1:0]		INT0LVL[1:0]		141
+0x0A	INT0MASK	INT0MSK[7:0]								141
+0x0B	INT1MASK	INT1MSK[7:0]								141
+0x0C	INTFLAGS	-	-	-	-	-	-	INT1IF	INT0IF	142
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	
+0x10	PIN0CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x11	PIN1CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x12	PIN2CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x13	PIN3CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x14	PIN4CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x15	PIN5CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x16	PIN6CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x17	PIN7CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]			142
+0x18	Reserved	-	-	-	-	-	-	-	-	
+0x19	Reserved	-	-	-	-	-	-	-	-	
+0x1A	Reserved	-	-	-	-	-	-	-	-	
+0x1B	Reserved	-	-	-	-	-	-	-	-	
+0x1C	Reserved	-	-	-	-	-	-	-	-	
+0x1D	Reserved	-	-	-	-	-	-	-	-	
+0x1E	Reserved	-	-	-	-	-	-	-	-	
+0x1F	Reserved	-	-	-	-	-	-	-	-	

13.18 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]								143
+0x01	Reserved	-	-	-	-	-	-	-	-	
+0x02	VPCTRLA	VP1MAP[3:0]				VP0MAP[3:0]				143
+0x03	VPCTRLB	VP3MAP[3:0]				VP2MAP[3:0]				144
+0x04	CLKEVOUT			EVOUT[1:0]				CLKOUT[1:0]		145

13.19 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]								146
+0x01	OUT	OUT[7:0]								146
+0x02	IN	IN[7:0]								146
+0x03	INTFLAGS	-	-	-	-	-	-	INT1IF	INT0IF	147

13.20 Interrupt vector Summary - Ports

Table 13-9. Ports Interrupt vectors and their word offset address

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

14. TC - 16-bit Timer/Counter

14.1 Features

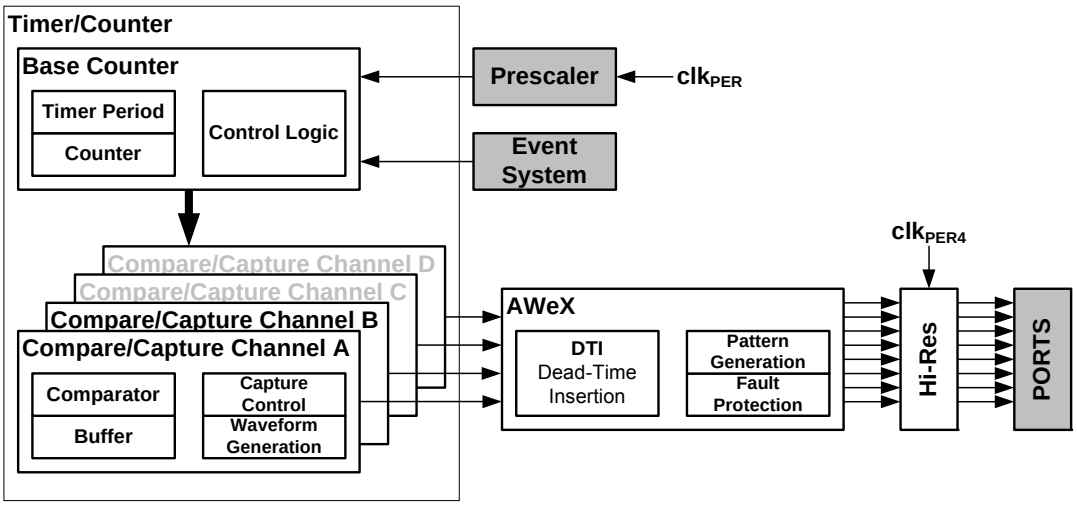
- 16-bit Timer/Counter
- Double Buffered Timer Period Setting
- Up to 4 Combined Compare or Capture (CC) Channels (A, B, C, and D)
- All Compare or Capture Channels are Double Buffered
- Waveform Generation:
  - Single Slope Pulse Width Modulation
  - Dual Slope Pulse Width Modulation
  - Frequency Generation
- Input Capture:
  - Input Capture with Noise Cancelling
  - Frequency capture
  - Pulse width capture
- 32-bit input capture Direction Control
- Timer Overflow and Timer Error Interrupts / Events
- One Compare Match or Capture Interrupt / Event per CC Channel
- Supports DMA Operation
- Hi-Res- Hi-Resolution Extension
  - Increases PWM/FRQ Resolution by 2-bits (4x)
- AWeX - Advanced Waveform Extension
  - 4 Dead-Time Insertion (DT) Units with separate high- and low-side settings
  - Event controlled fault protection
  - Single channel multiple output operation
  - Pattern Generation

14.2 Overview

XMEGA has a set of high-end and very flexible 16-bit Timer/Counters (TC). Their basic capabilities include accurate program execution timing, frequency and waveform generation, event management, and time measurement of digital signals. The Hi-Resolution Extension (Hi-Res) and Advanced Waveform Extension (AWeX) can be used together with a Timer/Counter to ease implementation of more advanced and specialized frequency and waveform generation features.

A block diagram of the 16-bit Timer/Counter with extensions and closely related peripheral modules (in grey) is shown in Figure 14-1 on page 150.

Figure 14-1. 16-bit Timer/Counter and Closely Related Peripheral



The 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, waveform generation (FRQ or PWM) or various input capture operations.

Compare and capture cannot be done at the same time, i.e. a single Timer/Counter cannot simultaneously perform both waveform generation and capture operation. When used for compare operations, the CC channels is referred to as compare channels. When used for capture operations, the CC channels are referred to as capture channels.

The Timer/Counter comes in two versions: Timer/Counter 0 that has four CC channels, and Timer/Counter 1 that has two CC channels. Hence, all registers and register bits that are related to CC channel 3 and CC channel 4 will only exist in Timer/Counter 0.

All Timer/Counter units are connected to the common peripheral clock prescaler, the Event System, and their corresponding general purpose I/O port.

Some of the Timer/Counters will have Extensions. The function of the Timer/Counter Extensions can only be performed by these Timers. The Advanced Waveform Extension (AWeX) can be used for Dead Time Insertion, Pattern Generation and Fault Protection. The AWeX Extension is only available for Timer/Counter 0.

Waveform outputs from a Timer/Counter can optionally be passed through to a Hi-Resolution (Hi-Res) Extension before forwarded to the port. This extension, running at up to four times the Peripheral Clock frequency, to enhance the resolution by four times. All Timer/Counters will have the Hi-Res Extention.

14.2.1 Definitions

The following definitions are used extensively throughout the Timer/Counter documentation:

Table 14-1. Timer/Counter definitions

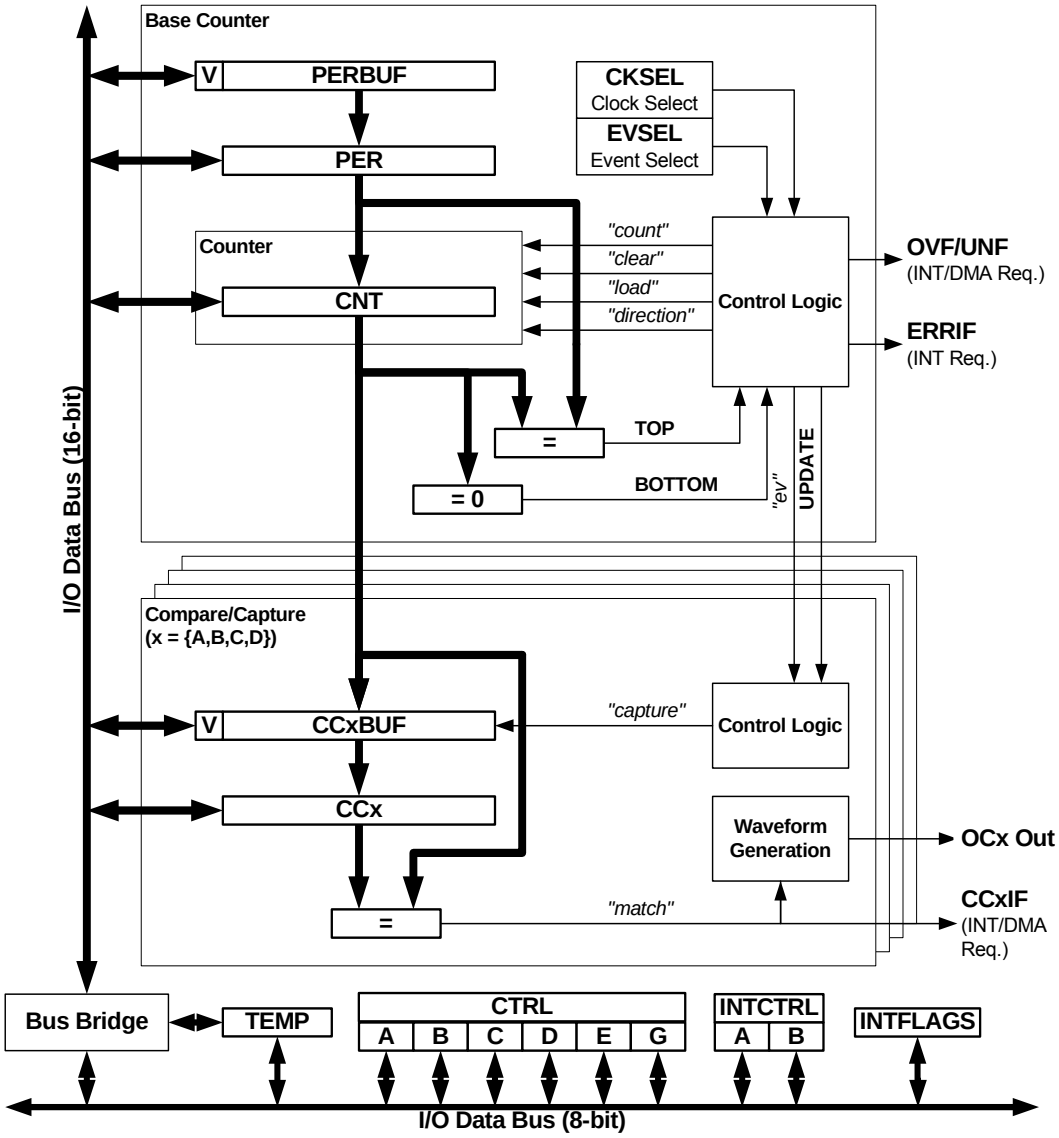
Name	Description
BOTTOM	The Counter reaches the BOTTOM when it becomes zero.
MAX	The Counter reaches MAXimum when it becomes all ones.
TOP	The Counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be equal to the period (PER) or the Compare Channel A (CCA) register setting. This is selected by the Waveform Generator Mode.
UPDATE	The Timer/Counter signalizes an update when it reaches BOTTOM or TOP dependent of 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 if the clock is given externally (from an event).

14.3 Block Diagram

Figure 14-2 on page 152 shows a detailed block diagram of the Timer/Counter without the extensions.

Figure 14-2. Timer/Counter Block Diagram



The Counter Register (CNT), the Period Registers w/buffer (PER and PERBUF), and the compare and Capture registers w/buffers (CCx and CCxBUF) are 16-bit registers.

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.

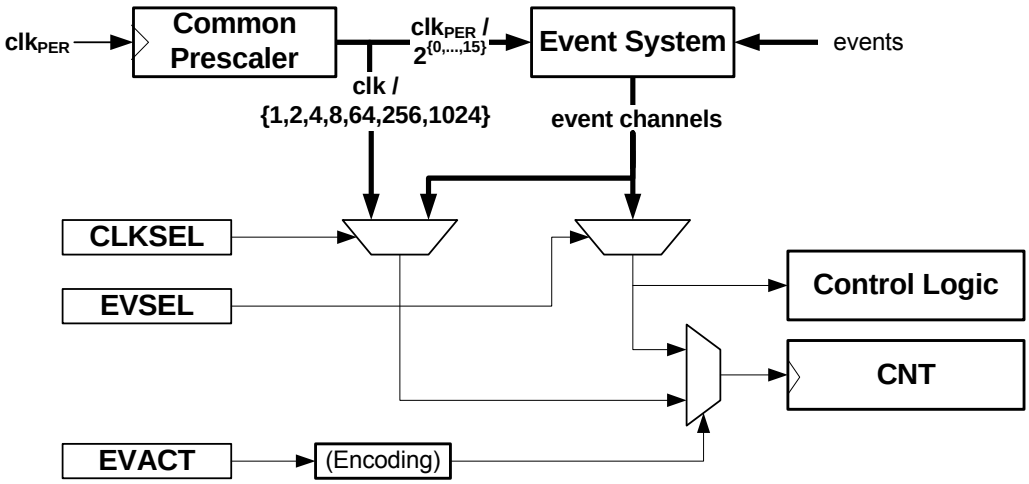
The counter value is also compared to the CCx registers. These comparators can be used to generate interrupt requests or request DMA transactions. They also generate events for the Event System. The waveform generator modes use the comparators to set the waveform period or pulse width.

A prescaled peripheral clock and events from the Event System can be used for controlling the counter. The Event System is also used as source to the input capture. Combined with the Quadrature Decoding functionality in the Event System QDEC, the Timer/Counter can be used for high speed Quadrature Decoding.

14.4 Clock and Event Sources

The Timer/Counter can be clocked from the Peripheral Clock (clk_{PER}) and from the Event System, and Figure 14-3 shows the clock and event selection logic.

Figure 14-3. Clock and Event Selection



The Peripheral Clock is fed into the Common Prescaler (common for all Timer/Counters in a device). A selection of the prescaler outputs is directly available for the Timer/Counter. In addition the whole range from 1 to 2^{15} times prescaling is available through the Event System.

Each Timer/Counter has separate clock selection (CLKSEL), to select one of the prescaler outputs directly or an event channel as the Counter (CNT) input. This is referred to as Normal Operation for the Counter, for details refer to "Normal Operation" on page 155. By using the Event System, any event source such as an external clock signal on any I/O pin can be used as clock input.

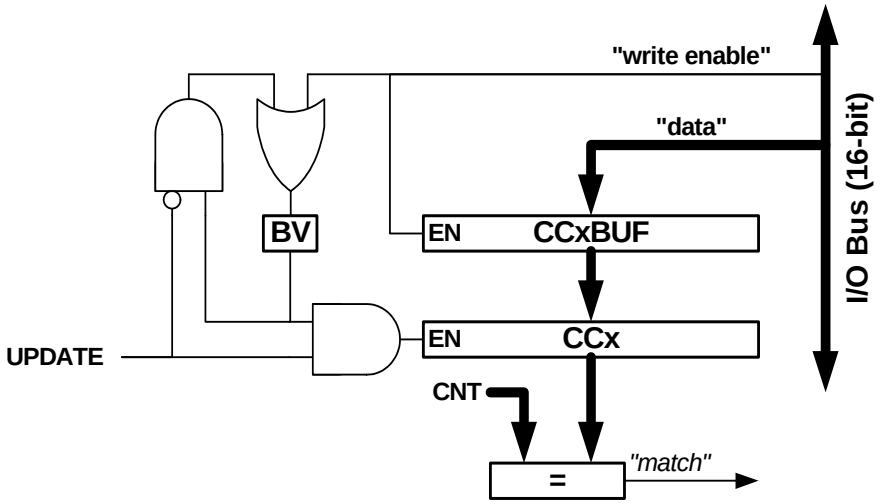
In addition the Timer/Counter can be controlled via the Event System. The Event Selection (EVSEL) and Event Action (EVACT) settings can be used to trigger an event action from one or more events. This is referred to as Event Action Controlled Operation for the Counter, for details refer to "Event Action Controlled Operation" on page 155. When Event Action Controlled Operation is used, the clock selection must be set to us an event channel as the Counter input.

By default no clock input is selected and the Timer/Counter is not running (OFF state).

14.5 Double Buffering

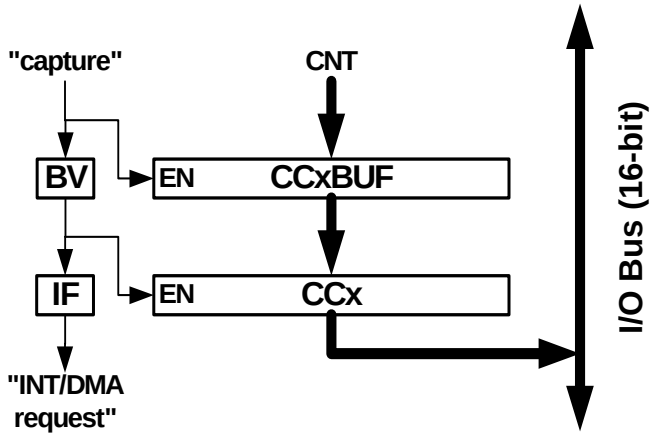
The Period Register and the CC registers are all double buffered. Each buffer registers have an associated Buffer Valid (BV) flag, which indicate that the buffer contains a valid, i.e. a new value that is to be copied into the belonging period or compare register. For the Period register and for the CC channels when used for compare operation, the Buffer Valid flag is set when data is written to the buffer register and cleared on UPDATE condition. This is shown for a compare register in Figure 14-4 on page 154.

Figure 14-4. Period and Compare Double Buffering



When the CC channels is used for capture operation a similar Double buffering mechanism is used, but the Buffer Valid flag is set on the capture event as shown in Figure 14-5. For capture the buffer and the corresponding CCx register acts like a FIFO. When the CC register is empty or read, any contents in the buffer is passed to the CC register. The Buffer valid flag is passed to the CCx Interrupt Flag (IF) which is them set and the optional interrupt is generated.

Figure 14-5. Capture Double Buffering



Both the CCx and CCxBUF registers are available in the I/O register address map. This allows initialization and bypassing of the buffer register, and the double buffering feature.

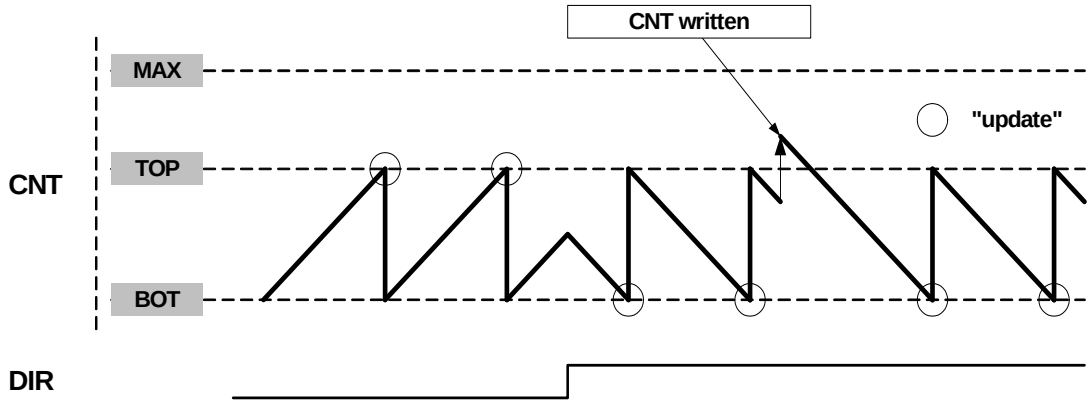
14.6 Counter Operation

Dependent of the mode of operation, the Counter is cleared, reloaded, incremented, or decremented at each Timer/Counter clock input.

14.6.1 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 TOP is reached when up-counting, the counter will be set to zero when the next clock is given. When down-counting the Counter is reloaded with Period Register value when BOTTOM is reached.

Figure 14-6. Normal Mode of Operation



As shown in Figure 14-6 changing the counter value while the counter is running is possible. 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.

14.6.2 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 event actions can be selected to:

- Event system controlled Up/Down counting.
- Event system controlled Quadrature Decode counting.

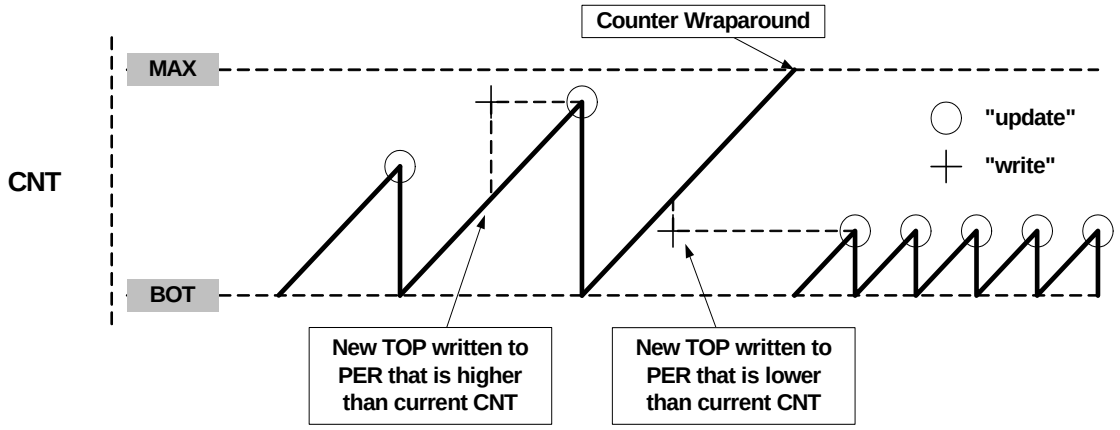
14.6.3 32-bit Operation

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 clock input for another Timer/Counter (most significant timer).

14.6.4 Changing the Period

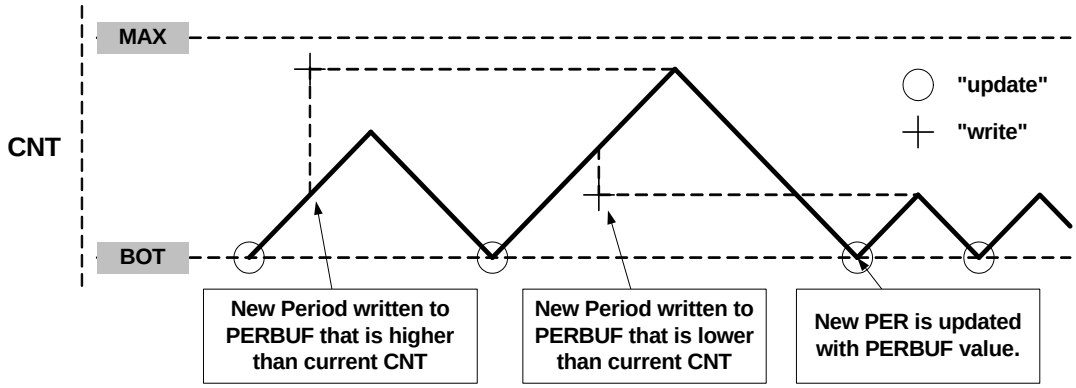
The Counter period is changed by writing a new TOP value to the Period Register. If double buffering is not used, any period update is immediate as shown in Figure 14-7 on page 156.

Figure 14-7. Changing The Period without Buffering



When double buffering is used, the buffer can be written at any time, but the Period Register is always updated on the “update” condition as shown in Figure 14-8. This prevents wraparound and generation of odd waveforms.

Figure 14-8. Changing Period using Buffering

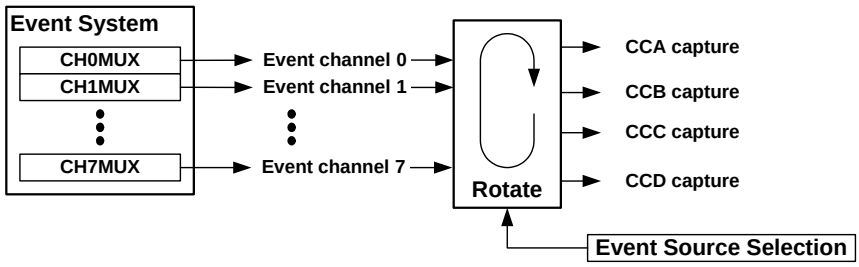


14.7 Capture Channel

The CC channels can be used as capture channel to capture external events and give them a time-stamp indicating time of occurrence. To use capture the Counter must be set in 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 the event channel that will trigger CC channel A, and the following event channels will then trigger events on the following CC channels if configured. For instance setting the Event Source Select to event channel 2 will result in CC channel A being connected to Event Channel 2, CC channel B to event channel 3 and so on.

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 channel to use 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 value in the Count register into the enabled CC channel register.

When an I/O pin is used as 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 134. If the Period register value is set lower than 0x8000, the polarity of the I/O pin edge will be stored in the Most Significant Bit (MSB) of the Capture register after a Capture. 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.

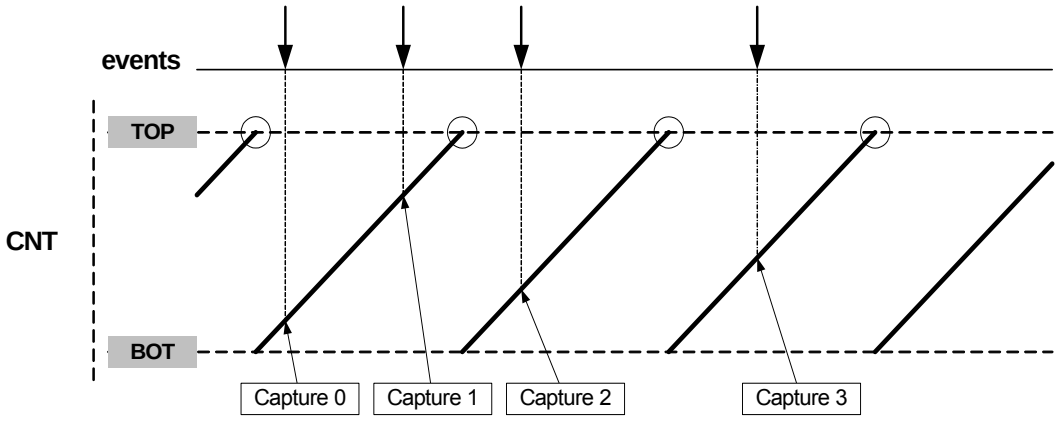
Three different types of capture are available.

14.7.1 Input Capture

Selecting the input capture event action, makes the enabled capture channel perform an input capture on any event. The interrupt flags will be set and indicate that there is a valid capture result in the corresponding CC register. Equally the buffer valid flags indicates valid data in the buffer registers. Refer to ”Double Buffering” on page 153 for more details on capture double buffering.

The counter will continuously count for BOTTOM to TOP, then restart on BOTTOM as shown in Figure 14-9. The figure also shows four capture events for one capture channel.

Figure 14-9. Input capture timing



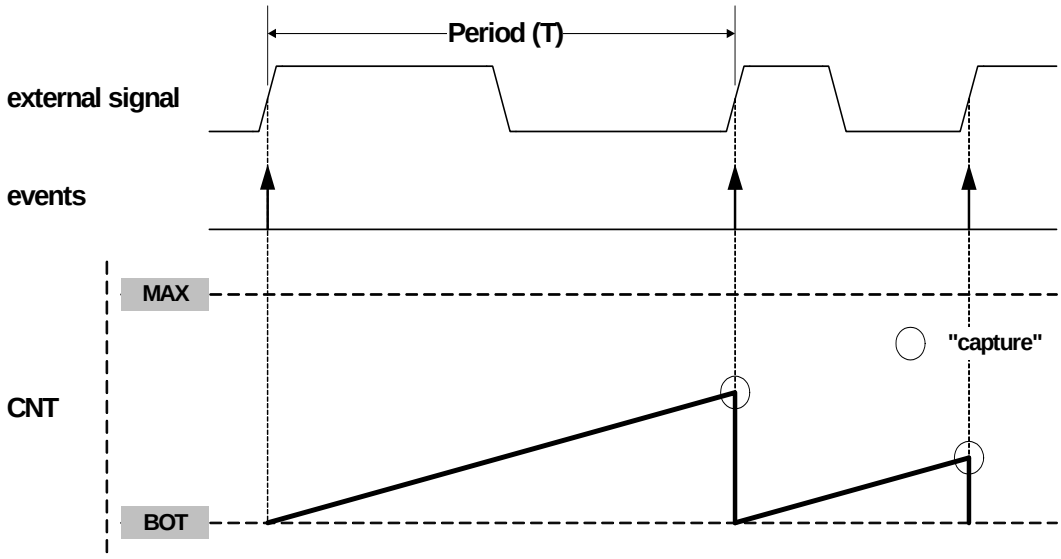
14.7.2 Frequency Capture

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

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

Figure 14-10 on page 158 shows an example where the Period is measured twice for an external signal.

Figure 14-10. Frequency capture of an external signal

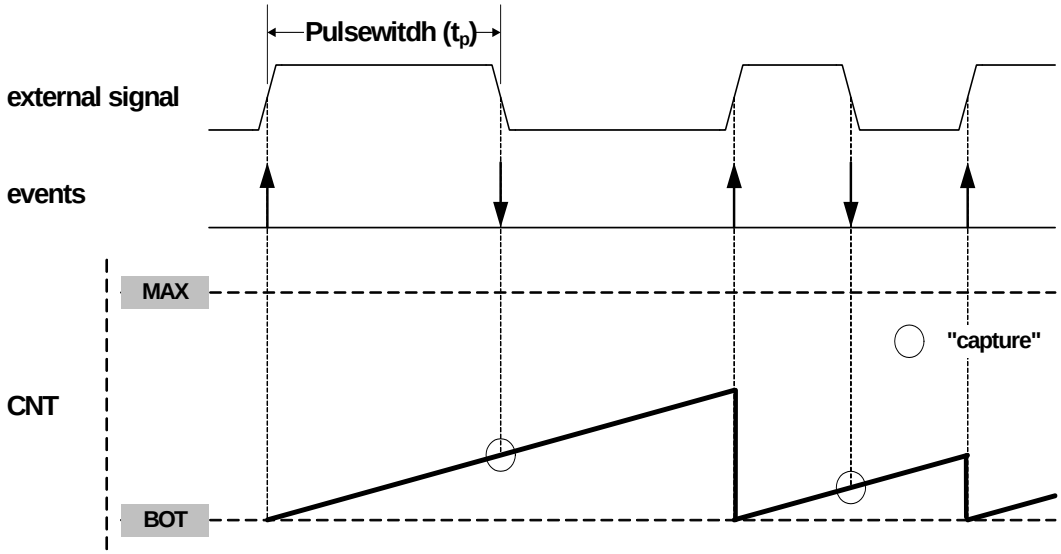


Since all capture channels uses the same Counter (CNT), only one capture channels must be enabled at a time. If two capture channels are used with different source, the Counter will be restarted on positive edge events from both input sources, and the result from the input capture will have no meaning.

14.7.3 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 start at zero at every start of a pulse and the input capture will be performed at the end of the pulse. The event source must be an I/O pin and the sense configuration for the pin must be set up to generate an event on both edges. Figure 14-11 on page 159 shows and example where the pulse width is measured twice for an external signal.

Figure 14-11. Pulse-width capture of external signal.



14.7.4 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 clock input for the most significant timer.

Since all events are pipelined, the most significant timer will be updated one peripheral clock period after an overflow occurs for the least significant timer. To compensate for this delay the capture event for the most significant timer must be equally delayed by setting the Event Delay bit for this timer.

14.7.5 Capture Overflow

The Timer/Counter can detect buffer overflow on any of the Input Capture Channels. In the case where both the Buffer Valid flag and Capture Interrupt Flag are set, and a new capture event is detected there is nowhere to store the new time-stamp. If a buffer overflow is detected the new value is rejected, the Error Interrupt Flag is set and the optional interrupt is generated.

14.8 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 is generated.

The compare buffer register provides double buffer capability equivalent to 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 signal from the Timer/Counter control logic. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM/FRQ pulses, thereby making the output glitch-free.

14.8.1 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:

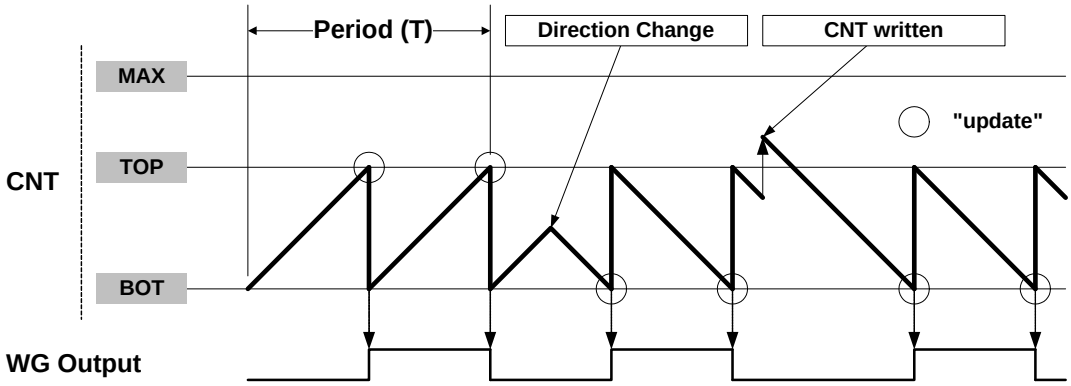
1. A waveform generation mode must be selected.
2. Event actions must be disabled.
3. The CC channels to be used must be enabled. This will override the corresponding port pin output register.
4. The direction for the associated port pin must be set to output.

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

14.8.2 Frequency (FRQ) Waveform Generation

For frequency generation the period time (T) is controlled by the CCA register instead of PER, which in this case is not in use. The Waveform Generation (WG) output is toggled on each compare match between the CNT and CCA registers as shown in Figure 14-12 on page 160.

Figure 14-12. Frequency Waveform Generation



The waveform generated will have a maximum frequency of half of the Peripheral clock frequency (f_{PER}) when CCA is set to zero (0x0000). This also applies when using the Hi-Res Extension since this only increase the resolution and not the frequency. The waveform frequency (f_{FRQ})is defined by the following equation:

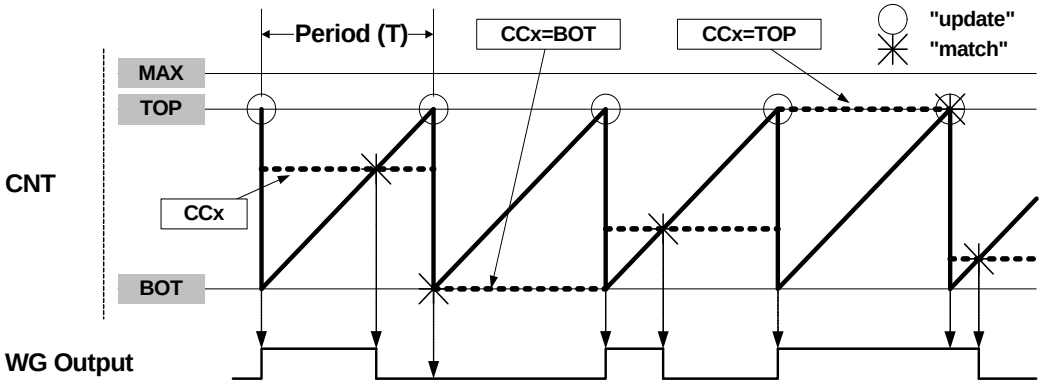
$$f_{FRQ} = \frac{f_{PER}}{2N(\mathrm{CCA}+1)}$$

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

14.8.3 Single Slope PWM Generation

For single slope PWM generation, the Period (T) is controlled by the PER, while CCx registers control the duty cycle of the WG output. Figure 14-13 shows how the counter counts from BOTTOM to TOP 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 14-13. Single slope Pulse Width Modulation



The PER register defines the PWM resolution. The minimum resolution is 2-bit (PER=0x0003), and maximum resolution is 16-bit (PER=MAX).

The following equation can be used for calculate the exact resolution for single-slope PWM (R_{PWM_SS}):

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

The single slow PWM frequency (f_{PWM_SS}) depends on the period setting (PER) and the Peripheral clock frequency (f_{PER}), and can be calculated by the following equation:

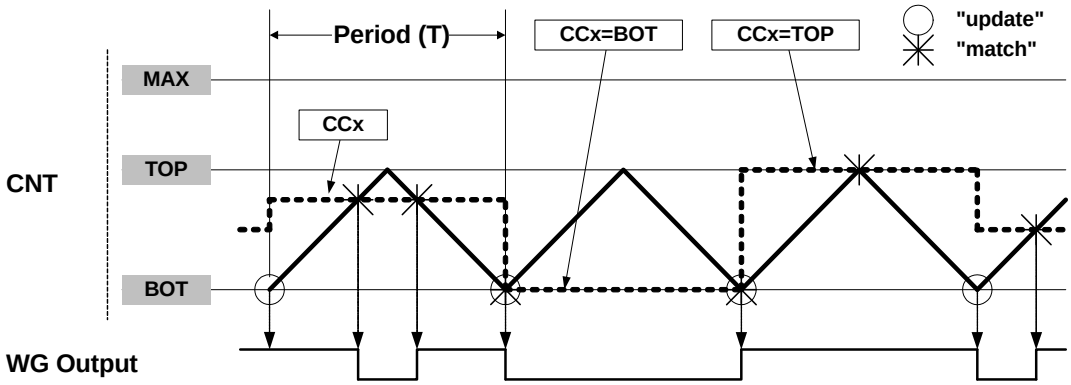
$$f_{PWM_SS} = \frac{f_{PER}}{N(PER + 1)}$$

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

14.8.4 Dual Slope PWM

For dual slope PWM generation, the Period (T) is controlled by the PER, while CCx registers control the duty cycle of the WG output. Figure 14-14 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 upcounting and set on compare match when down counting.

Figure 14-14. Dual-slope Pulse Width Modulation



Using dual-slope PWM result 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-bit (PER=0x0003), and maximum resolution is 16-bit (PER=MAX).

The following equation can be used for calculate the exact resolution for dual-slope PWM (R_{PWM_DS}):

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

The PWM frequency depends on the period setting (PER) and the Peripheral Clock frequency (f_{PER}), and can be calculated by the following equation:

$$f_{PWM_DS} = \frac{f_{PER}}{2NPER}$$

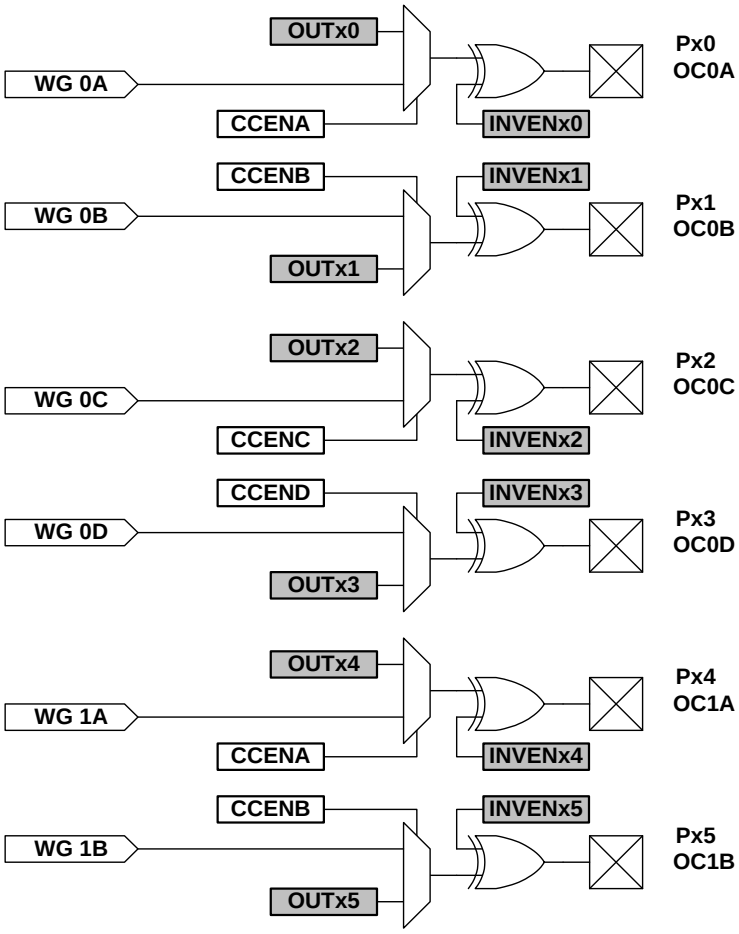
N represents the prescaler divider used (1, 2, 4, 8, 64, 256, 1024, or event channel n).

14.8.5 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 14-15 on page 163 shows the port override for Timer/Counter 0 and 1. For Timer/Counter 1, CC channel A to D will override port pin 0 to 3 output value (OUTxn) on the corresponding port pin (Pxn). For Timer/Counter 1, CC channel A and B will override port pin 4 and 5. Enabling inverted I/O on the port pin (INVENxn) inverts the corresponding WG output.

Figure 14-15. Port override for Timer/Counter 0 and 1



14.9 Interrupts and events

The T/C 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 the T/C can generate an error interrupt if any of the CC channels is used for capture and a buffer overflow condition occurs on a capture channel.

Event will be generated for all conditions that can generate interrupts. For details on event generation and available events refer to ”Event System” on page 65.

14.10 DMA Support.

The interrupt flags can be used to trigger DMA transactions. Table 14-2 on page 164 lists the transfer triggers available from the T/C, and the DMA action that will clear the transfer trigger. For more details on using DMA refer to ”DMA - Direct Memory Access Controller” on page 49.

Table 14-2. DMA Request Sources

Request	Acknowledge	Comment
OV/UNFIF	DMA Controller writes to CNT DMA Controller writes to PER DMA Controller writes to PERBUF	
ERRIF	N/A	
CCxIF	DMA Controller access of CCx DMA Controller access of CCxBUF	Input Capture operation Output Compare operation

14.11 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 only be given when the Timer/Counter is not running (OFF).

14.12 Register Description

14.12.1 CTRLA - Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	-	CLKSEL[3:0]				CTRLA
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 bits**

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 14-3.

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

Table 14-3. Clock Select

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
1xxx	EVCHn	Event channel n, n= [0,...,7]

14.12.2 CTRLB - Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x01	CCDEN	CCCEN	CCBEN	CCAEN	-	WGMODE[2:0]			CTRLB
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 FRQ or PWM waveform generation mode of operation will override of the port output register for the corresponding OCn output pin.

When input capture operation is selected the CCxEN bits enables 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, the TOP value, the UPDATE condition, the Interrupt/event condition, and type of waveform that is generated, according to Table 14-4 on page 166.

No waveform generation is performed in 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 14-4. Timer Waveform Generation Mode

WGMODE[2:0]	Group Configuration	Mode of operation	Top	Update	OVFIF/Event
000	NORMAL	Normal	PER	TOP	TOP
001	FRQ	FRQ	CCA	TOP	TOP
010		Reserved	-	-	-
011	SS	Single Slope PWM	PER	BOTTOM	BOTTOM
100		Reserved	-	-	-
101	DS_T	Dual Slope PWM	PER	BOTTOM	TOP
110	DS_TB	Dual Slope PWM	PER	BOTTOM	TOP and BOTTOM
111	DS_B	Dual Slope PWM	PER	BOTTOM	BOTTOM

14.12.3 CTRLC - Control Register C

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	CMPD	CMPC	CMPB	CMPA	CTRLC
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 n**

These bits allow direct access to the Waveform Generator's output compare value when the Timer/Counter is set in “OFF” state. This is used to set or clear the WG output value when the Timer/Counter is not running.

14.12.4 CTRLD - Control Register D

Bit	7	6	5	4	3	2	1	0	
+0x03	EVACT[2:0]			EVDLY	EVSEL[3:0]				CTRLD
Read/Write	R/W	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 14-5 on page 167.

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

Table 14-5. 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 the any of the capture event action changes the behavior of the CCx registers and related status and control bits to be used as for capture. The error status flag (ERRIF) will in this configuration indicate a buffer overflow.

- **Bit 4 – EVDLY: Timer Delay Event**

When this bit is set, the selected event source is delayed by one peripheral clock cycle. This feature is intended for 32-bit input capture operation. Adding the event delay is necessary for compensating for the carry propagation delay that is inserted 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 14-6. When the Event Action is set to 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 14-6. Timer Event Source Selection

EVSEL[3:0]	Group Configuration	Event Source
0000	OFF	None
0001		Reserved
0010		Reserved
0011		Reserved
0100		Reserved
0101		Reserved
0110		Reserved
0111		Reserved
1xxx	CHn	Event channel n, n={0,...,7}

14.12.5 CTRLE - Control Register E

Bit	7	6	5	4	3	2	1	0	
+0x04	-	-	-	-	-	-	-	BYTEM	CTRLE
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 - BYTEM: Byte Mode:**

Enabling the Byte Mode, sets the Timer/Counter in Byte (8-bit) Mode. Setting this bit will disable the update of the temporary register (TEMP) when any of the 16-bit Timer/Counter registers are accessed. In addition the upper byte of the counter (CNT) register will be set to zero after each counter clock.

14.12.6 INTCTRLA - Interrupt Enable Register A

Bit	7	6	5	4	3	2	1	0	
+0x06	-	-	-	-	ERRINTLVL[1:0]		OVFINTLVL[1:0]		INTCTRLA
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 Multi-level Interrupt Controller” on page 123.

- **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 Multi-level Interrupt Controller” on page 123.

14.12.7 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]		INTCTRLB
Read/Write	R/W	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[1:0] - Compare or Capture x Interrupt Level:**

These bits enable the Timer Compare or Capture Interrupt and select the interrupt level as described in ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123.

14.12.8 CTRLFCLR/CTRLFSET - Control Register F Clear/Set

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

Bit	7	6	5	4	3	2	1	0	
+0x09	-	-	-	-	CMD[1:0]		LUPD	DIR	CTRLFSET
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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 yield 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 each bit to be set or cleared without using of a Read-Modify-Write operation on a single register.

- **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 can be used for software control of update, restart, and reset of the Timer/Counter. The command bits are always read as zero.

Table 14-7. 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 “OFF“state)

- **Bit 1 - LUPD: Lock Update:**

When this bit is set no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffers, including DTI buffers, are valid before an update is performed.

This bit has no effect when input capture operation is enabled.

- **Bit 0 - DIR: Counter Direction:**

When zero, this bit indicates that the counter is counting up (incrementing). A one indicates that the counter is in 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.

14.12.9 CTRLGCLR/CTRLGSET - Control Register G Clear/Set

Bit	7	6	5	4	3	2	1	0	
+0x0A/ +0x0B	-	-	-	CCDBV	CCCBV	CCBBV	CCABV	PERBV	CTRLGCLR/SET
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 section ”CTRLFCLR/CTRLFSET - Control Register F Clear/Set” on page 169 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 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.

14.12.10 INTFLAGS - Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
+0x0C	CCDIF	CCCIF	CCBIF	CCAIF	-	-	ERRIF	OVFIF	INTFLAGS
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 will in this mode of operation 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 releases the request.

- **Bit 3:2 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 - ERRIF: Error Interrupt Flag**

The ERRIF is set on multiple occasions depending on mode of operation.

In FRQ or PWM waveform generation mode of operation the 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 the ERRIF is set if a buffer overflow occurs on any of the CC channels.

For event controlled QDEC operation the ERRIF is set when an incorrect index signal is given.

The ERRIF 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 - OVFIF: Overflow/Underflow Interrupt Flag**

The OVFIF is set either on a TOP (overflow) or BOTTOM (underflow) condition depending on the WGMODE setting. The 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.

The 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.

14.12.11 TEMP - Temporary Register for 16-bit Access

The TEMP register is used for single cycle 16-bit access to the 16-bit Timer/Counter registers from 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 Section 3.11 ”Accessing 16-bits Registers” on page 12.

Bit	7	6	5	4	3	2	1	0	
+0x0F	TEMP[7:0]								TEMP
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.12.12 CNTH - Counter Register H

The CNTH and CNTL register pair represents the 16-bit value CNT. CNT contains the 16-bit counter value in the Timer/Counter. The CPU and DMA write access has priority over count, clear, or reload of the counter.

For more details on reading and writing 16-bit register refer to ”Accessing 16-bits Registers” on page 12.

Bit	7	6	5	4	3	2	1	0	
+0x21	CNT[15:8]								CNTH
Read/Write	R/W	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]**

These bits holds the 8 MSB of the 16-bit Counter register.

14.12.13 CNTL - Counter Register L

Bit	7	6	5	4	3	2	1	0	
+0x20	CNT[7:0]								CNTL
Read/Write	R/W	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]**

These bits holds the 8 LSB of the 16-bit Counter register.

14.12.14 PERH - Period Register H

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	
+0x27	PER[15:8]								PERH
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]**

These bits holds the 8 MSB of the 16-bit Period register.

14.12.15 PERL - Period Register L

Bit	7	6	5	4	3	2	1	0	
+0x26	PER[7:0]								PERL
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]**

These bits holds the 8 LSB of the 16-bit Period register.

14.12.16 CCxH - Compare or Capture Register n H

The CCxH and CCxL register pair represents the 16-bit value CCx.

These 16-bit registers have two functions dependent of 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 all continuously compared to the counter value. Normally the outputs form the comparators are then used for generating waveforms.

CCx are updated with the buffer value from the corresponding CCxBUF register when an UPDATE condition occurs.

Bit	7	6	5	4	3	2	1	0	
	CCx[15:8]								CCxH
Read/Write	R/W	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]**

These bits holds the 8 MSB of the 16-bit Compare or Capture register.

14.12.17 CCxL - Compare or Capture Register n L

Bit	7	6	5	4	3	2	1	0	
	CCx[7:0]								CCxL
Read/Write	R/W	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]**

These bits holds the 8 LSB of the 16-bit Compare or Capture register.

14.12.18 PERBUFH - Timer/Counter Period Buffer H

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 CPU or DMA will affect the PERBUFV flag.

Bit	7	6	5	4	3	2	1	0	
+0x37	PERBUF[15:8]								PERBUFH
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]**

These bits holds the 8 MSB of the 16-bit Period Buffer register.

14.12.19 PERBUFL - Timer/Counter Period Buffer L

Bit	7	6	5	4	3	2	1	0	
+0x36	PERBUF[7:0]								PERBUFL
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]**

These bits holds the 8 LSB of the 16-bit Period Buffer register.

14.12.20 CCxBUFH - Compare or Capture x Buffer Register H

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 register using CPU or DMA will affect the corresponding CCxBV status bit.

Bit	7	6	5	4	3	2	1	0	
	CCxBUF[15:8]								CCxBUFH
Read/Write	R/W	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]**

These bits holds the 8 MSB of the 16-bit Compare or Capture Buffer register.

14.12.21 CCxBUFL - Compare or Capture x Buffer Register L

Bit	7	6	5	4	3	2	1	0	
	CCxBUFx[7:0]								CCxBUFL
Read/Write	R/W	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]**

These bits holds the 8 LSB of the 16-bit Compare or Capture Buffer register.

14.13 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]				165
+0x01	CTRLB	CCDEN	CCCEN	CCBEN	CCAEN	-	WGMODE[2:0]			165
+0x02	CTRLC	-	-	-	-	CPMD	CPMC	CPMB	CPMA	166
+0x03	CTRLD	EVACT[2:0]			EVDLY	EVSEL[3:0]				167
+0x04	CTRLE	-	-	-	-	-	-	-	BYTEM	168
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	INTCTRLA	-	-	-	-	ERRINTLVL[1:0]		OVINTLVL[1:0]		168
+0x07	INTCTRLB	CCCINTLVL[1:0]		CCCINTLVL[1:0]		CCBINTLVL[1:0]		CCAINTLVL[1:0]		168
+0x08	CTRLFCLR	-	-	-	-	CMD[1:0]		LUPD	DIR	169
+0x09	CTRLFSET	-	-	-	-	CMD[1:0]		LUPD	DIR	170
+0x0A	CTRLGCLR	-	-	-	CCDBV	CCCBV	CCBBV	CCABV	PERBV	170
+0x0B	CTRLGSET	-	-	-	CCDBV	CCCBV	CCBBV	CCABV	PERBV	170
+0x0C	INTFLAGS	CCDIF	CCCIF	CCBIF	CCAIF	-	-	ERRIF	OVFIF	171
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	TEMP	TEMP[7:0]								171
+0x10 to +0x1F	Reserved	-	-	-	-	-	-	-	-	
+0x20	CNTL	CNT[7:0]								172
+0x21	CNTH	CNT[15:8]								172
+0x22 to +0x25	Reserved	-	-	-	-	-	-	-	-	
+0x26	PERL	PER[7:0]								173
+0x27	PERH	PER[8:15]								172
+0x28	CCAL	CCA[7:0]								173
+0x29	CCAH	CCA[15:8]								173
+0x2A	CCBL	CCB[7:0]								173
+0x2B	CCBH	CCB[15:8]								173
+0x2C	CCCL	CCC[7:0]								173
+0x02D	CCCH	CCC[15:8]								173
+0x2E	CCDL	CCD[7:0]								173
+0x2F	CCDH	CCD[15:8]								173
+0x30 to +0x35	Reserved	-	-	-	-	-	-	-	-	
+0x36	PERBUFL	PERBUF[7:0]								174
+0x37	PERBUFH	PERBUF[15:8]								173
+0x38	CCABUFL	CCABUF[7:0]								174
+0x39	CCABUFH	CCABUF[15:8]								174
+0x3A	CCBBUFL	CCBBUF[7:0]								174
+0x3B	CCBBUFH	CCBBUF[15:8]								174
+0x3C	CCCBUFL	CCCBUF[7:0]								174
+0x3D	CCCBUFH	CCCBUF[15:8]								174
+0x3E	CCDBUFL	CCDBUF[7:0]								174
+0x3F	CCDBUFH	CCDBUF[15:8]								174

14.14 Interrupt Vector Summary

Table 14-8. Timer/Counter Interrupt vectors and their word offset address

Offset	Source	Interrupt Description
0x00	OVF_vect	Timer/Counter Overflow/Underflow Interrupt vector offset
0x02	ERR_vect	Timer/Counter Error Interrupt vector offset
0x4	CCA_vect	Timer/Counter Compare or Capture Channel A Interrupt vector offset
0x6	CCB_vect	Timer/Counter Compare or Capture Channel B Interrupt vector offset
0x8	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. Only available on Timer/Counter with 4 Compare or Capture channels 16-bit.

15. AWeX – Advanced Waveform Extension

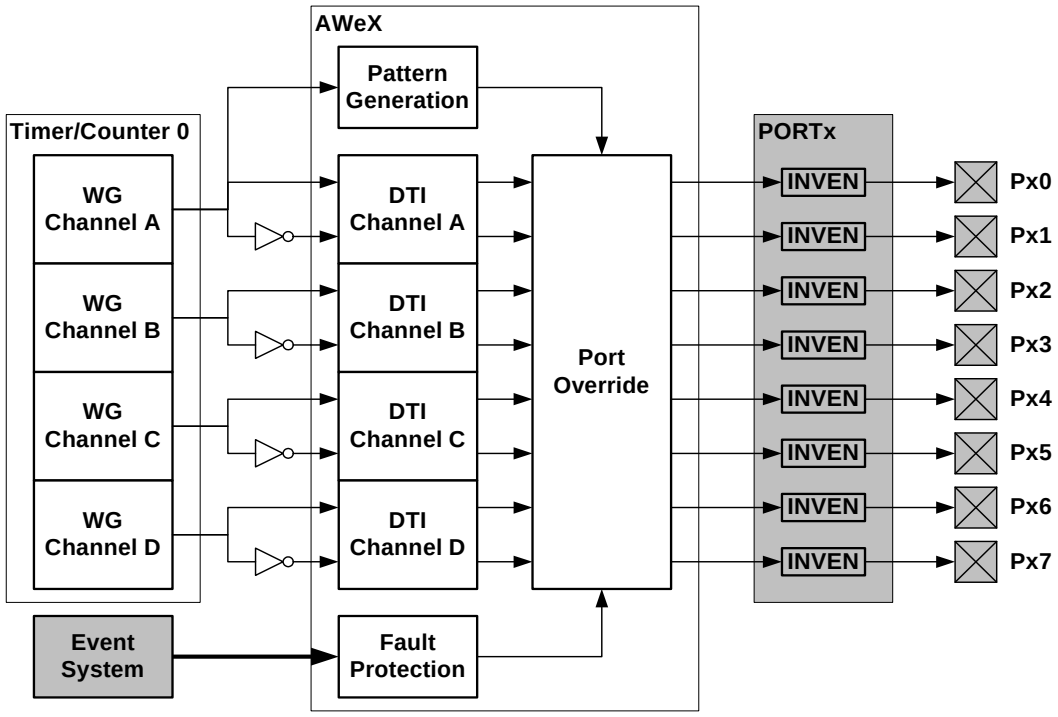
15.1 Features

- 4 Dead-Time Insertion (DTI) Units (8-pin)
  - 8-bit Resolution
  - Separate High and Low Side Dead-Time Setting
  - Double Buffered Dead-Time
  - Halts Timer During Dead-Time (Optional)
- Event Controlled Fault Protection
- Single Channel Multiple Output Operation (for BLDC control)
- Double Buffered Pattern Generation
- The Hi-Resolution Timer Extension Increases PWM/FRQ Resolution by 2-bits (4x)

15.2 Overview

The Advanced Waveform Extention (AWeX) provides extra features to the Timer/Counter in Waveform Generation (WG) modes. The AWeX enables easy and robust implementation of advanced motor control (AC, BLDC, SR, and Stepper) and power control applications.

Figure 15-1. Advanced Waveform eXtention and closely related peripherals (grey)



As shown in Figure 15-1 on page 176 each of the waveform generator outputs from the Timer/Counter 0 are split into a complimentary pair of outputs when any AWeX features is enabled. These output pairs go through a Dead-Time Insertion (DTI) unit that enables generation of 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. Optionally the final output can be inverted by using the invert I/O (INVEN) bit setting for the port pin (Pxn).

The Pattern Generation unit can be used to generate a synchronized bit pattern on the port it is connected to. In addition, the waveform generator output from the 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.

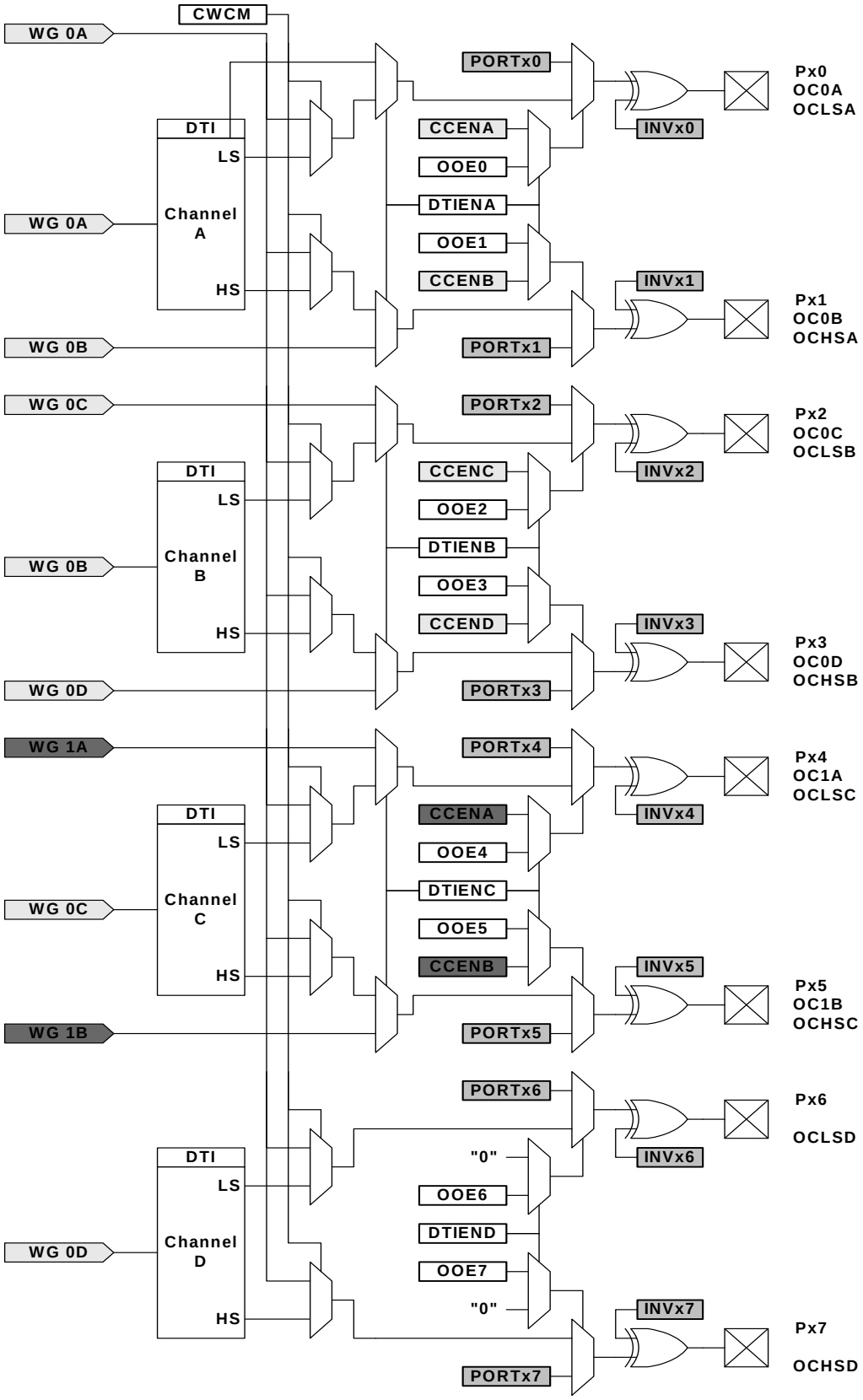
The Fault Protection unit is connected to the Event System, enabling any event to trigger a fault condition that will disable the AWeX output.

15.3 Port Override

Common for all the timer/counter extensions is the port override logic. Figure 15-2 on page 178 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 takes control over the CCxEN.

Note that timer/counter 1 (TCx1) can still be used even when DTI channels A, B, and D are enabled.

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

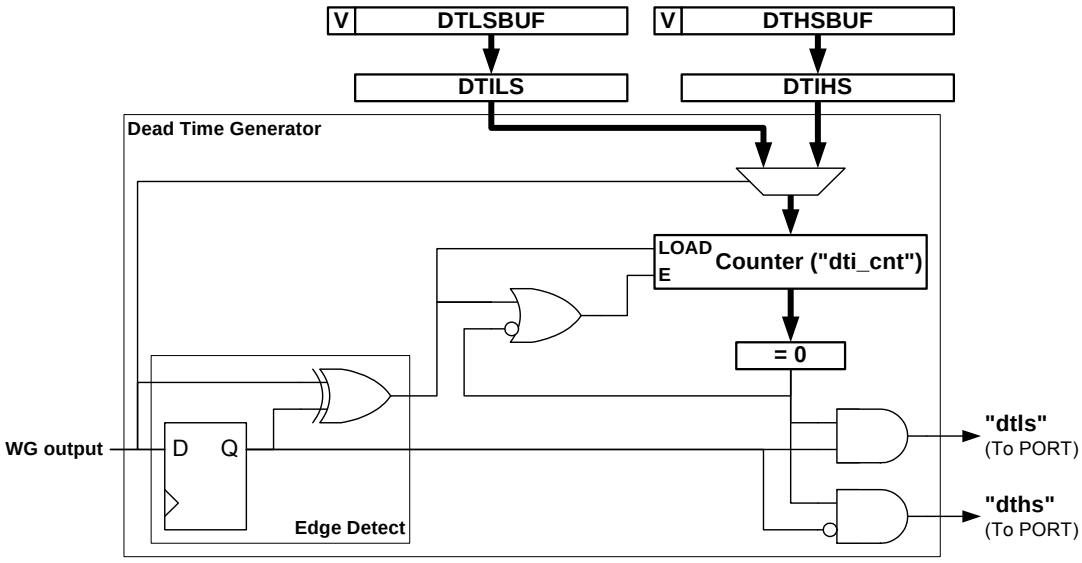


15.4 Dead Time Insertion

The Dead Time Insertion (DTI) unit enables generation of “off” time where both the non-inverted Low Side (LS) and inverted High Side (HS) of the WG output is low. This “off” time is called dead-time, and dead-time insertion ensure that the LS and HS does not switch simultaneously.

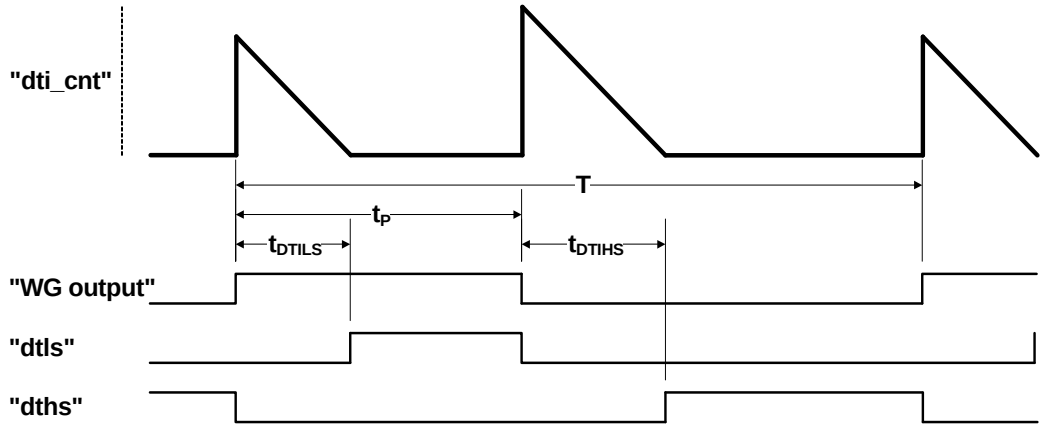
The DTI unit consists of four equal dead time generators, one for each of the capture or compare channel in Timer/Counter 0. Figure 15-3 on page 179 shows the block diagram of one dead time generator. The dead time registers that define the number of peripheral clock cycles the dead time is going to last, are common for all four channels. The High Side and Low Side can 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 180, the 8-bit Dead Time Counter (dti_cnt) is decremented by one for each peripheral clock cycle until it reaches zero. A non-zero 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 with the DTx register value according to the edge of the input. Positive edge initiates a counter reload of the DTLS Register and a negative edge a reload of DTHS Register.

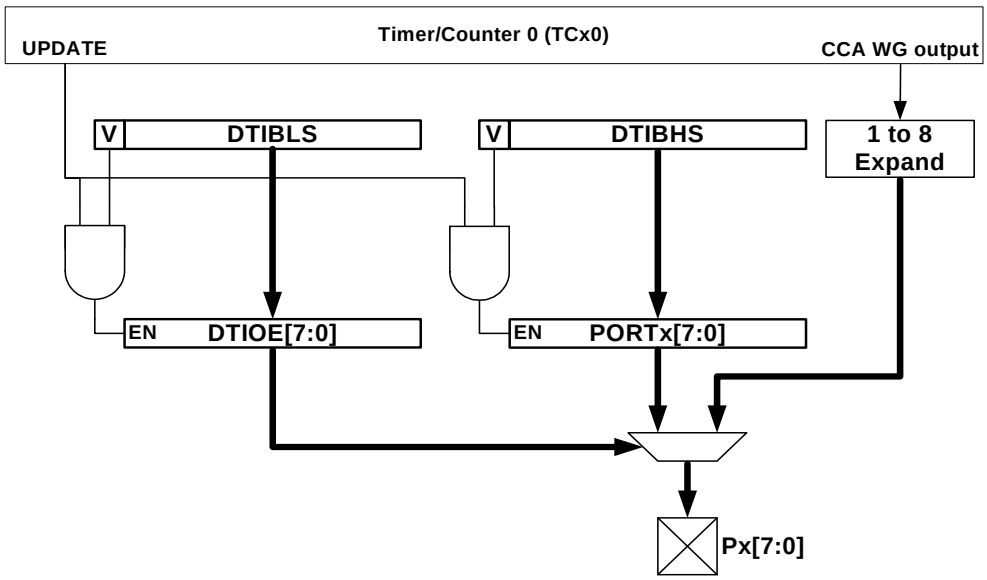
Figure 15-4. Dead Time Generator timing diagram



15.5 Pattern Generation

The pattern generator extension reuses the DTI registers to produce a synchronized bit pattern on the port it is connected to. In addition, the waveform generator output from CC channel A (CCA)) can be distributed to and override all the port pins. These features are primarily intended for handling the commutation sequence in BLDC and Stepper Motor Applications.

Figure 15-5. Pattern Generator block diagram



A block diagram of the pattern generator is shown in Figure 15-5 on page 180. For each port pin where the corresponding OOE bit is set the multiplexer will output the waveform from CCA.

As for all other types of the 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 pins direction 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.

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

15.6.1 Fault Actions

Two different even actions can be selected:

- The Clear Override Enable action will clear the Output Override Enable register (OUTOVEN) and disable the output override on all Timer/Counter outputs. The result is that the in the output will be as set by the port pin configuration.
- The Direction Clear action will clear the Direction (DIR) register in the associated port, setting all port pins as tri-stated inputs.

When a fault is detected the Fault Detection Flag is set, and the Timer/Counter's Error Interrupt Flag is set and the optional interrupt is generated.

From the event occurs in one peripherals until the Fault Protection triggers the event action, there is maximum two peripheral clock cycles. The Fault Protection is fully independent of the CPU and DMA, but it requires the Peripheral Clock to run.

15.6.2 Fault Restore Modes

After a fault, that is when the fault condition is no longer active, it is selectable how the AWeX and Timer/Counter can return from fault state and restore with normal operation. Two different modes are available:

- 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 entering fault state and the Clear Override Enable action is selected, the OUTOVEN[7:0] bits are reassigned a value on the next UPDATE condition. In pattern generation mode the register is restored with the value in the DTLSBUF register. Otherwise the register bits are restored according to the enabled DTI channels.

When entering fault state and Direction Clear action is select is set, corresponding DIR[7:0] bits is restored with the value in the DTLSBUF register in pattern generation mode and for the pin pairs corresponding to enabled DTI channels otherwise.

The UPDATE condition used to restore the normal operation is the same update as 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 Advanced Waveform Extension

Lock Register. For more details refer to ”IO Memory Protection” on page 25 and ”AWEXLOCK – Advanced Waveform Extension Lock Register” on page 45.

When the lock bit is set, the Control Register A, the Output Override Enable Register and the Fault Dedec.tion 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 ”IO Memory Protection” on page 25 and ”EVSYSLOCK – Event System Lock Register” on page 44.

15.6.4 On-Chip Debug

When fault detection is enabled an OCD system receives a break request from the debugger, this will by default function as a fault source. When an OCD break request is received, the AWeX and corresponding Timer/Counter will enter fault state and the specified fault action(s) 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, and 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 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	CTRL
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 - RES - 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 if set. This will override the DTI if enabled, 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 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]								FDEMASK
Read/Write	R/W	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 enables the corresponding event channel as fault condition input source. Event 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]		FDCTRL
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 - RES - 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 the 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 - RES - 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 use, and when this 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 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.

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

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

Table 15-1. Fault actions

FDACT[1:0]	Group Configuration	Description
00	NONE	None (Fault protection disabled)
01	CLEAROE	Clear all override enable (OUTOVEN) bits, i.e. disable the output override.
11	CLEARDIR	Clear all Direction (DIR) bits, which correspond to 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	STATUS
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 reserved and will always be read as zero. 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 the 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 UPDATE condition. If this bit is zero no action will be taken. The connected Timer/Counter'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 UPDATE condition. If this bit is zero no action will be taken. Note that 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]								DTBOTH
Read/Write	R/W	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 both 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

Bit	7	6	5	4	3	2	1	0	
+0x07	DTBOTHBUF[7:0]								DTBOTHBUF
Read/Write	R/W	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 both 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]								DTLS
Read/Write	R/W	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]								DTHS
Read/Write	R/W	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]								DTLSBUF
Read/Write	R/W	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 contents 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]								DTHSBUF
Read/Write	R/W	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 contents 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]								OUTOVEN
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 only be written if the fault detect flag (FDF) is zero.

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

These bits enable override of 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	Page
+0x00	CTRL	-	-	PGM	CWCM	DTICDAEN	DTICCCEN	DTICCBEN	DTICCAEN	182
+0x01	Reserved	-	-	-	-	-	-	-	-	
+0x02	FDEMASK	FDEVMASK[7:0]								183
+0x03	FDCTRL	-	-	-	FDDBD	-	FDMODE	FDACT[1:0]		183
+0x04	STATUS	-	-	-	-	-	FDF	DTBHSV	DTBLSV	184
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	DTBOTH	DTBOTH[7:0]								184
+0x07	DTBOTHBUF	DTBOTHBUF[7:0]								185
+0x08	DTLS	DTLS[7:0]								185
+0x09	DTHS	DTHS[7:0]								185
+0x0A	DTLSBUF	DTLSBUF[7:0]								185
+0x0B	DTHSBUF	DTHSBUF[7:0]								185
+0x0C	OUTOVEN	OUTOVEN[7:0]								186

16. Hi-Res - High Resolution Extension

16.1 Features

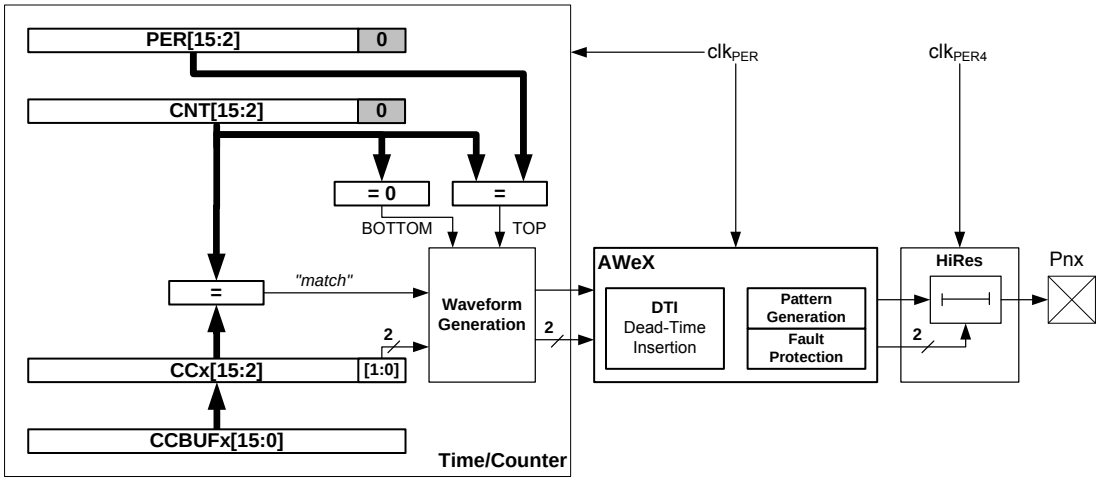
- **Increases Waveform Generator Resolution by 4x (2 bits)**
- **Supports Frequency generation, and single and dual-slope PWM operation**
- **Supports Dead-Time Insertion (AWeX)**
- **Supports Pattern Generation (AWeX)**

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 (two bits). It can be used during Frequency and PWM generation, and also in combination with the corresponding AWeX.

The Hi-Res Extension uses the Peripheral 4x Clock. The System Clock prescalers must be set up so the Peripheral 4x Clock frequency is four times higher than the Peripheral and CPU clock frequency (see ”System Clock Selection and Prescalers” on page 80) when the Hi-Res Extension is enabled.

Figure 16-1. Timer/Counter operation with Hi-Res Extension enabled



The Hi-Res Extension is implemented by letting the Timer/Counter run at 4x its normal speed. When Hi-Res Extension is enabled, the counter will ignore its two lowest significant bits (LSB) and count by four for each Peripheral clock cycle. Overflow/Underflow and Compare match of the 14 most significant bits (LSB) is done in the Timer/Counter. Count and Compare of the two LSB is then handled and compared in the Hi-Res Extension running from the Peripheral 4x clock.

The two LSB of the Period register must always be set to zero to ensure correct operation. If the Count register is read, the two LSB will always be read as zero since the Timer/Counter run from the Peripheral clock.

The Hi-Res Extension has narrow pulse deletion preventing output of any pulse shorter than one Peripheral clock cycle, e.g. a compare value lower than foure will have no visible output.

16.3 Register Description

16.3.1 CTRLA - Hi-Res Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	–	–	–	–	–	–	HREN[1:0]		CTRLA
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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1:0 - HREN[1:0]: Hi-Resolution Enable**

Enables Hi-Resolution mode for a Timer/Counter according to Table 16-1.

Setting one or both HREN bits will enable Hi-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. Hi-Resolution Enable

HREN[1:0]	Hi-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							HREN[1:0]		189

17. RTC - Real Time Counter

17.1 Features

- 16-bit resolution
- Selectable clock reference
  - 32.768 kHz
  - 1.024 kHz
- Programmable prescaler
- 1 Compare register
- 1 Period register
- Clear Timer on overflow
- Optional Interrupt/ Event on overflow and compare match

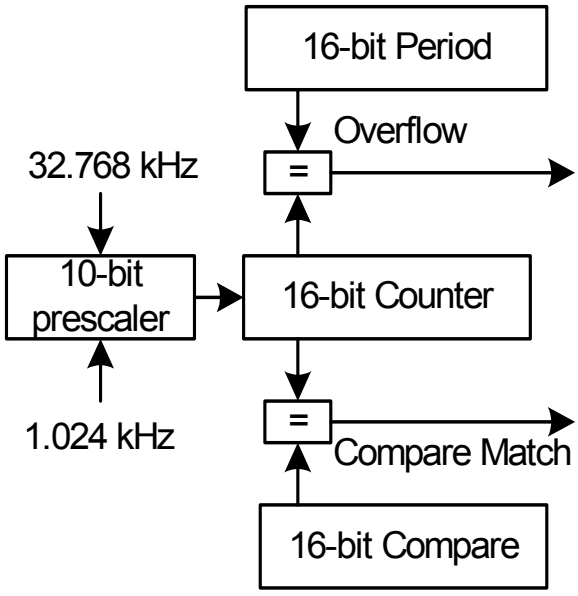
17.2 Overview

The Real Time Counter (RTC) is a 16-bit counter, counting reference clock cycles and giving an event and/or an interrupt request when it reaches a configurable compare and/or top value. The reference clock is typically generated from a high accuracy crystal of 32.768 kHz, and the design is optimized for low power consumption. The RTC typically operate in low power sleep modes, keeping track of time and waking up the device at regular intervals.

The RTC reference clock may be taken from an 32.768 kHz or 1.024 kHz input. Both an external 32.768 kHz crystal oscillator or the 32 kHz internal RC oscillator can be selected as clock source. For details on reference clock selection to the RTC refer to ”RTCCTRL - RTC Control Register” on page 86 in the Clock System section. The RTC has a programmable prescaler to scale down the reference clock before it reaches the Counter.

The RTC can generate both compare and overflow interrupt request and/or events.

Figure 17-1. Real Time Counter overview.



17.2.1 Clock domains

The RTC is asynchronous, meaning it operates from a different clock source and 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 or until a configuration change has effect on the RTC. This synchronization time is described for each register.

17.2.2 Interrupts and events

The RTC can generate both interrupts and events. The RTC will give a compare interrupt request and/or event when the counter value equals the Compare register value. The RTC will give an overflow interrupt request and/or event when the counter value equals the Period register value. The overflow will also reset the counter value to zero.

Due to the asynchronous clock domains event will only will only be generated for every third overflow or compare if the period register is zero. If the period register is one, events will only be generated for every second overflow or compare. When the period register is equal to or above two, events will trigger at every overflow or compare just as the interrupt request.

17.3 Register Description

17.3.1 CTRL - Real Time Counter Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	-	-	PRESCALER[2:0]			CTRL
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 - Reserved**

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

- **Bits 2:0 - PRESCALER[2:0]: RTC Clock Prescaling factor**

These bits define the prescaling factor for the RTC clock before the counter according to Table 17-1 on page 191.

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 - Real Time Counter Status Register

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	-	-	-	-	SYNCBUSY	STATUS
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 - SYNCBUSY: RTC Synchronization Busy Flag**

This bit is set when the CNT, CTRL, PER or COMP register is busy synchronizing between the RTC clock and system clock domains.

17.3.3 INTCTRL - Real Time Counter Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	COMPINTLVL[1:0]		OVFINTLVL[1:0]		INTCTRL
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.

- **Bits 3:2 - COMPINTLVL[1:0]: RTC Compare Match Interrupt Enable**

These bits enable the RTC Compare Match Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger when the COMPIF in the INTFLAGS register is set.

- **Bits 1:0 - OVFINTLVL[1:0]: RTC Overflow Interrupt Enable**

These bits enable the RTC Overflow Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger when the OVFIF in the INTFLAGS register is set.

17.3.4 INTFLAGS - RTC Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	-	-	COMPIF	OVFIF	INTFLAGS
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 - COMPIF: RTC Compare Match Interrupt Flag**

This flag is set on the next count after a Compare Match condition occurs. The flag is cleared automatically when RTC compare match interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **Bit 0 - OVFIF: RTC Overflow Interrupt Flag**

This flag is set on the next count after an Overflow condition occurs. The flag is cleared automatically when RTC overflow interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

17.3.5 TEMP - RTC Temporary Register

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

- **Bits 7:0 - TEMP[7:0]: Real Time Counter Temporary Register**

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 low byte is read by the CPU. For more details refer to ”Accessing 16-bits Registers” on page 12.

17.3.6 CNTH - Real Time Counter Register H

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 require special attention, refer to ”Accessing 16-bits Registers” on page 12 for details.

Due to synchronization between RTC clock and the 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 - Real Time Counter Status Register” on page 192 is cleared before writing to this register.

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

- **Bits 7:0 - CNT[15:8]: Real Time Counter value High byte**

These bits hold the 8 MSB of the 16-bit Real Time Counter value.

17.3.7 CNTL - Real Time Counter Register L

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

- **Bits 7:0 - CNT[7:0]: Real Time Counter value Low byte**

These bits hold the 8 LSB of the 16-bit Real Time Counter value.

17.3.8 PERH - Real Time Counter Period Register High

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 the OVFIF in the INTFLAGS register and clear CNT. Reading and writing 16-bit values require special attention, refer to ”Accessing 16-bits Registers” on page 12 for details.

Due to synchronization between RTC clock and the 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 - Real Time Counter Status Register” on page 192 is cleared before writing to this register.

Bit	7	6	5	4	3	2	1	0	
+0x0B	PER[15:8]								PERH
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]: Real Time Counter Period High byte**

These bits hold the 8 MSB of the 16-bit RTC top value.

17.3.9 PERL - Real Time Counter Period Register L

Bit	7	6	5	4	3	2	1	0	
+0x0A	PER[7:0]								PERL
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[7:0]: Real Time Counter Period Low byte**

These bits hold the 8 LSB of the 16-bit RTC top value.

17.3.10 COMPH - Real Time Counter Compare Register H

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 the COMPIF in the INTFLAGS register. Reading and writing 16-bit values require special attention, refer to ”Configuration Change Protection” on page 12 for details.

Due to synchronization between RTC clock and the system clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. Application SW needs to check that the SYNCBUSY flag in the ”STATUS - Real Time Counter Status Register” on page 192 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	
+0x0D	COMP[15:8]								COMPH
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 - COMP[15:8]: Real Time Counter Compare Register High byte**

These bits hold the 8 MSB of the 16-bit RTC compare value.

17.3.11 COMPL - Real Time Counter Compare Register L

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

- **Bits 7:0 - COMP[7:0]: Real Time Counter Compare Register Low byte**

These bits hold the 8 LSB 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]			191
+0x01	STATUS	-	-	-	-	-	-	-	SYNCBUSY	192
+0x02	INTCTRL	-	-	-	-	COMPINTLVL[1:0]		OVFINTLVL[1:0]		192
+0x03	INTFLAGS	-	-	-	-	-	-	COMPIF	OVFIF	193
+0x04	TEMP	TEMP[7:0]								193
+0x08	CNTL	CNT[7:0]								193
+0x09	CNTH	CNT[15:8]								194
+0x0A	PERL	PER[7:0]								194
+0x0B	PERH	PER[15:8]								194
+0x0C	COMPL	COMP[7:0]								195
+0x0D	COMPH	COMP[15:8]								195

17.5 Interrupt Vector Summary

Table 17-2. RTC Interrupt vectors and their word offset address

Offset	Source	Interrupt Description
0x00	OVF_vect	Real Time Counter Overflow Interrupt vector
0x02	COMP_vect	Real Time Counter Compare Match Interrupt vector

18. RTC32 - 32-bit Real Time Counter

18.1 Features

- **32-bit resolution**
- **Selectable clock reference**
  - **1.024 kHz**
  - **1 Hz**
- **One Compare register**
- **One Period register**
- **Clear Timer on overflow**
- **Optional Interrupt/ Event on overflow and compare match**
- **Isolated V_{BAT} power domain with dynamic switch over from/to V_{CC} power domain**

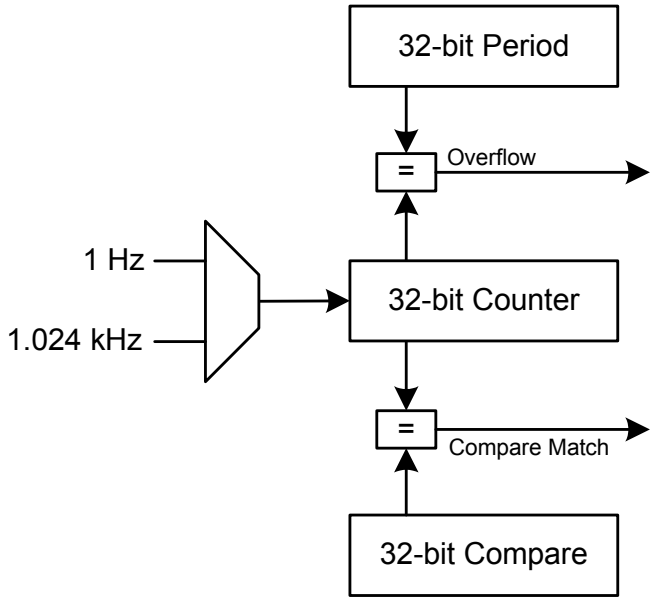
18.2 Overview

The 32-bit Real Time Counter is a 32-bit counter, counting reference clock cycles and giving an event and/or an interrupt request when it reaches a configurable compare and/or top value. The reference clock is generated from a high accuracy 32.768 kHz crystal, and the design is optimized for low power consumption. The Real Time Counter (RTC) typically operates in low power sleep modes, keeping track of time and waking up the device at regular intervals.

The RTC input clock can be taken from a 1.024 kHz or 1 Hz prescaled output from the 32.768 kHz reference clock. The RTC will give a compare interrupt request and/or event when the counter value equals the Compare register value. The RTC will give an overflow interrupt request and/or event when the counter value equals the Period register value. Counter overflow will also reset the counter value to zero.

The RTC can generate both compare and overflow interrupt request and/or events.

Figure 18-1. 32-bit Real Time Counter Overview



18.2.1 Clock selection

An external 32.768 kHz crystal oscillator must be used as clock source. Two different frequency outputs are available form this, and the RTC clock input can be 1.024 kHz or 1 Hz.

18.2.2 Clock domains

The RTC is asynchronous, meaning it operates from a different clock source and 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 or until a configuration change has effect on the RTC. This synchronization time is described for each register.

The Peripheral clock must be eight times faster than the RTC clock (1.024 kHz or 1 Hz) when any the Control and Count register is accessed (read or written), expect 12 times faster when the Count register is written.

18.2.3 Power domains

For devices where the RTC is located in the V_{BAT} power domain this enables the RTC to function correctly with no V_{CC} available. When running from the V_{BAT} power domain, only the 1 Hz clock source is available. A dynamic power-switch is used to automatically switch from V_{CC} domain to the V_{BAT} domain if V_{CC} falls below the operating voltage level for the device. When the V_{CC} voltage is restored, the power is automatically switched back to V_{CC}.

18.2.4 Interrupts and events

The RTC can generate both interrupts and events. The RTC will give a compare interrupt request and/or event when the counter value equals the Compare register value. The RTC will give an overflow interrupt request and/or event when the counter value equals the Period register value. The overflow will also reset the counter value to zero.

Due to the asynchronous clock domains event will only will only be generated for every third overflow or compare if the period register is zero. If the period register is one, events will only be generated for every second overflow or compare. When the period register is equal to or above two, events will trigger at every overflow or compare just as the interrupt request.

18.3 Register Description

18.3.1 CTRL - Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00						-	-	ENABLE	CTRL
Read/Write	R	R	R	R	R	R	R	R/W	
Reset 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 - ENABLE: RTC Enable**

Setting this bit enables the RTC. The synchronization time between the RTC and the System Clock domains is one half RTC clock cycle from writing the register and until this has effect in RTC clock domain, i.e until the RTC starts.

For the RTC to start running the PER Register must also be set to a different value that zero.

18.3.2 SYNCCTRL - Synchronisation Control/Status Register

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	SYNCCNT	-	-	-	SYNCBUSY	SYNCCTRL
Read/Write	R	R	R	R/W	R	R	R	R/W	
Reset Value	0	0	0	0	0	0	0	0	

- **Bits 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- SYNCCNT: Enable Synchronization of the CNT register**

Setting this bit will start synchronization of CNT register from the RTC clock to the System Clock domain. The bit is automatically cleared when synchronization is done.

- **Bits 3:1 - Reserved**

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

- **Bit 0 - SYNCBUSY: RTC Synchronization Busy Flag**

This flag is set when the CTRL or CNT registers are busy synchronizing from the System Clock to the RTC clock domain. The CTRL register synchronization is triggered when it is written. The CNT register are synchronized when the most significant byte of the register is written.

18.3.3 INTCTRL - Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	COMPINTLVL[1:0]		OCINTLVL[1:0]		INTCTRL
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Reset 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.

- **Bits 3:2 - COMPINTLVL[1:0]: RTC Compare Match Interrupt Enable**

These bits enable the RTC Compare Match Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger when the COMPIF in the INTFLAGS register is set.

- **Bits 1:0 - OVFINTLVL[1:0]: RTC Overflow Interrupt Enable**

These bits enable the RTC Overflow Interrupt and select the interrupt level as described in Section 12. "Interrupts and Programmable Multi-level Interrupt Controller" on page 123. The enabled interrupt will trigger when the OVFIF in the INTFLAGS register is set

18.3.4 INTFLAGS - RTC Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	-	-	COMPIF	OVFIF	INTFLAGS
Read/Write	R	R	R	R	R	R	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 - Reserved**

These bits are reserved and will always be read as zero.

- **Bit 1 - COMPIF: RTC Compare Match Interrupt Flag**

This flag is set on the next count after a Compare Match condition occurs. The flag is cleared automatically when RTC compare match interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **Bit 0 - OVFIF: RTC Overflow Interrupt Flag**

This flag is set on the next count after an Overflow condition occurs. The flag is cleared automatically when RTC overflow interrupt vector is executed. The flag can also be cleared by writing a one to its bit location

18.3.5 CNT3 - Counter Register 3

CNT3, CNT2, CNT1 and CNT0 registers represent the 32-bit value CNT. CNT counts positive clock edges on the RTC clock.

Synchronization of a new CNT value to the RTC domain is triggered by writing CNT3. The synchronization time is up to 12 Peripheral clock cycles from updating the register until this has an effect in RTC domain. Write operations to CNT register will be blocked if the SYNCBUSY flag is set.

The Synchronization of CNT value from RTC domain to System Clock domain can be done by writing one to the SYNCCNT bit in the CTRL register. The updated and synchronized CNT register value is available after eight Peripheral Clock cycles.

After writing to the high byte of the CNT register, the condition for setting OVFIF and COMPIF, as well as the Overflow and Compare Match Wakeup condition, will be disabled for the following two RTC clock cycles.

Bit	7	6	5	4	3	2	1	0	
+0x07	CNT[31:24]								CNT3
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.6 CNT2 - Counter Register 2

Bit	7	6	5	4	3	2	1	0	
+0x06	CNT[23:16]								CNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.7 CNT1 - Counter Register 1

Bit	7	6	5	4	3	2	1	0	
+0x05	CNT[15:8]								CNT1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.8 CNT0 - Counter Register 0

Bit	7	6	5	4	3	2	1	0	
+0x04	CNT[7:0]								CNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.9 PER3 - Period Register 3

PER3, PER2, PER1 and PER0 registers represent the 32-bit value PER. PER is constantly compared with the counter value (CNT). A compare match will set the OVFIF in the INTFLAGS register, and CNT will be set to zero in the next RTC clock cycle.

The PER register can only be written if the RTC is disabled and not currently synchronizing, i.e when both ENABLE and SYNCBUSY are zero.

After writing a byte in the PER register the condition for setting OVFIF and the Overflow Wakeup condition is disabled for the following two RTC clock cycles.

After reset this register is 0x0000, and it must be set to a value different that zero before the enabled RTC starts counting.

Bit	7	6	5	4	3	2	1	0	
+0x0B	PER[31:24]								PER3
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.10 PER2 - Period Register 2

Bit	7	6	5	4	3	2	1	0	
+0x0A	PER[23:16]								PER2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.11 PER1 - Period Register 1

Bit	7	6	5	4	3	2	1	0	
+0x09	PER[15:8]								PER1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.12 PER0 - Period Register 0

Bit	7	6	5	4	3	2	1	0	
+0x08	PER[7:0]								PER0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.13 COMP3 - Compare Register 3

The COMP0, COMP1, COMP2 and COMP3 registers represents the 32-bit value COMP. COMP is constantly compared with the counter value (CNT). A compare match will set the COMPIF in the INTFLAGS register, and the optional interrupt is generated.

If the COMP value is higher than the PER value, no RTC Compare Match interrupt requests or events will be generated

After writing the high byte of the COMP register, the condition for setting OVFIF and COMPIF, as well as the Overflow and Compare Match Wakeup condition, will be disabled for the following two RTC clock cycles.

Bit	7	6	5	4	3	2	1	0	
+0x0F	COMP[31:24]								COMP3
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.14 COMP2 - Compare Register 2

Bit	7	6	5	4	3	2	1	0	
+0x0E	COMP[23:16]								COMP2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.15 COMP1 - Compare Register 1

Bit	7	6	5	4	3	2	1	0	
+0x0D	COMP[15:8]								COMP1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.3.16 COMP0 - Compare Register 0

Bit	7	6	5	4	3	2	1	0	
+0x0C	COMP[7:0]								COMP0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset Value	0	0	0	0	0	0	0	0	

18.4 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	-	-	-	-	-	-	-	ENABLE	198
+0x01	SYNCCTRL	-	-	-	SYNCCNT	-	-	-	SYNCBUSY	199
+0x02	INTCTRL	-	-	-	-	COMPINTLVL[1:0]		OVFINTLVL[1:0]		199
+0x03	INTFLAGS	-	-	-	-	-		COMPIF	OVFIF	200
+0x04	CNT0	CNT[7:0]								201
+0x05	CNT1	CNT[15:8]								201
+0x06	CNT2	CNT[23:16]								201
+0x07	CNT3	CNT[31:24]								200
+0x08	PER0	PER[7:0]								202
+0x09	PER1	PER[15:8]								202
+0x0A	PER2	PER[23:16]								202
+0x0B	PER3	PER[31:24]								201
+0x0C	COMP0	COMP[7:0]								203
+0x0D	COMP1	COMP[15:8]								203
+0x0E	COMP2	COMP[23:16]								202
+0x0F	COMP3	COMP[31:24]								202

18.5 Interrupt Vector Summary

Table 18-1. RTC32 Interrupt vectors and their word offset address

Offset	Source	Interrupt Description
0x00	OVF_vect	Real Time Counter Overflow Interrupt vector
0x02	COMP_vect	Real Time Counter Compare Match Interrupt vector

19. TWI – Two Wire Interface

19.1 Features

- **Fully Independent Master and Slave Operation**
- **Multi-Master, Single Master, or Slave Only Operation**
- **Phillips I^2C compatible**
- **SMBus compatible**
- **100 kHz and 400 kHz support at low system clock frequencies**
- **Slew-Rate Limited Output Drivers**
- **Input Filter provides noise suppression**
- **7-bit, and General Call Address Recognition in Hardware**
- **Address mask register for address masking or dual address match**
- **10-bit addressing supported**
- **Optional Software Address Recognition Provides Unlimited Number of Slave Addresses**
- **Slave can operate in all sleep modes, including Power Down**
- **Support for 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 bi-directional 2-wire bus communication, which is I^2C and SMBus compatible.

A device connected to the bus must act as a master or 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 several masters, and an arbitration process handles priority if two or more masters try to transmit at the same time.

The TWI module in XMEGA implements both master and slave functionality. The master and slave functionality are separated from each other and can be enabled separately. They have separate control and status register, and separate interrupt vectors. Arbitration lost, errors, collision and clock hold on the bus will be detected in hardware and indicated in separate status flags available in both master and slave mode.

The master module contains a baud rate generator for flexible clock generation. Both 100 kHz and 400 kHz bus frequency at low system clock speed is supported. Quick Command and Smart Mode can be enabled to auto trigger operations and reduce software complexity.

For the slave, 7-bit and general address call recognition is implemented in hardware. 10-bit addressing is also supported. A dedicated address mask register can act as a second address match register or as a mask register for the slave address to match on a range of addresses. The slave logic continues to operate in all sleep modes, including Power down. This enables the slave to wake up from sleep on TWI address match. It is possible to disable the address matching and let this be handled in software instead. This allows the slave to detect and respond to several addresses. Smart Mode can be enabled to auto trigger operations and reduce software complexity.

The TWI module includes bus state logic that collects information to detect START and STOP conditions, bus collision and bus errors. This is used to determine the bus state (idle, owner, busy or unknown) in master mode. The bus state logic continues to operate in all sleep modes including Power down.

It is possible to disable the internal TWI drivers in the device, and enabling a 4-wire interface for connecting external bus drivers.

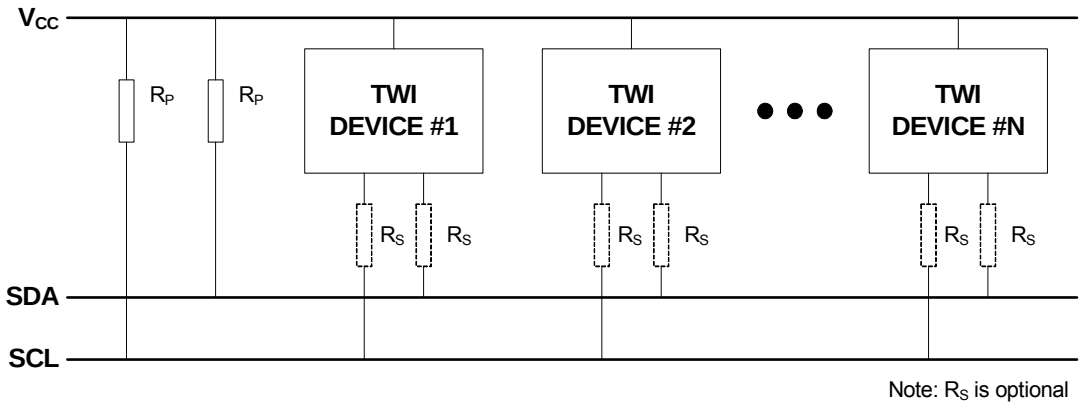
19.3 General TWI Bus Concepts

The Two-Wire Interface (TWI) provides a simple two-wire bi-directional 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 will provide a high level on the lines when none of the connected devices are driving the bus. A constant current source can be used as an alternative to the pull-up resistors.

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 illustrates the TWI bus topology.

Figure 19-1. TWI Bus Topology



Note: R_S is optional

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. 7-bit or 10-bit addressing can be used.

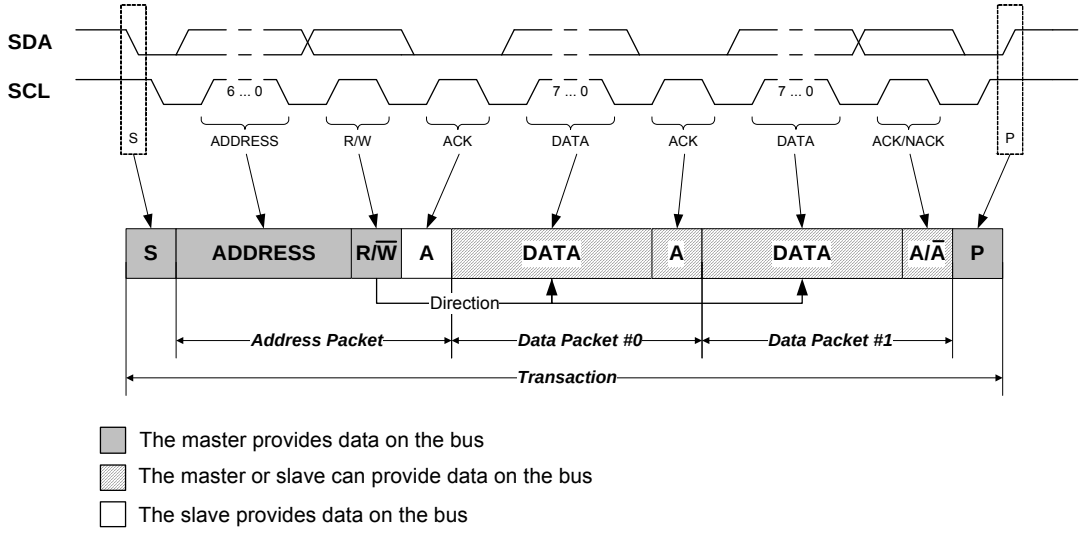
Several masters can be connected to the same bus, and this is called a multi-master environment. An arbitration mechanism is provided for resolving bus ownership between masters since only one master device may own the bus at any given time.

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 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/$\overline{W}$), is 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 shows a TWI transaction.

Figure 19-2. Basic TWI Transaction Diagram Topology



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.

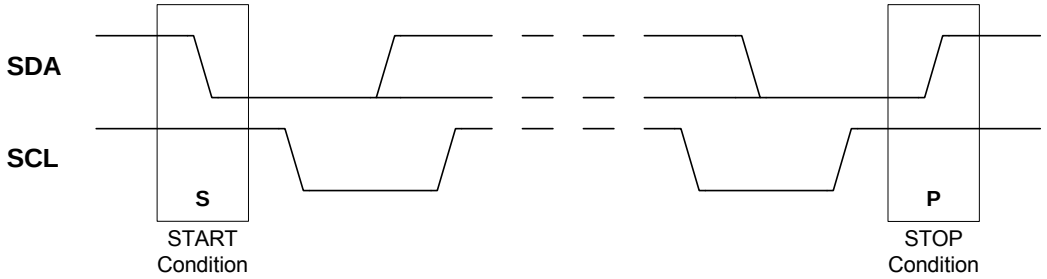
19.3.1 Electrical Characteristics

The TWI in XMEGA follows the electrical specifications and timing of I^2C and SMBus. These specifications are not 100% compliant so to ensure correct behavior the inactive bus timeout period should be set in TWI master mode.

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

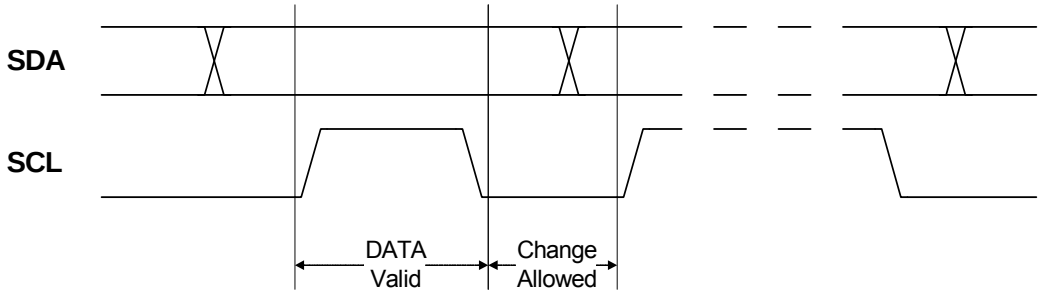


Multiple START conditions can be issued during a single transaction. A START condition not directly following a STOP condition, are named a Repeated START condition (Sr).

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 8 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, and NACK by leaving the line SCL high during the ninth clock cycle.

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 the next SCL cycle, while all other slaves should keep the TWI lines released, and wait for the next START and address. The 7-bit address, the R/$\overline{W}$ bit and the acknowledge bit combined is the address packet. Only one address packet for each START condition is given, also when 10-bit addressing is used.

The R/$\overline{W}$ 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. Opposite, for a Master Read operation the slave will start to transmit data after acknowledging its address.

19.3.5 Data Packet

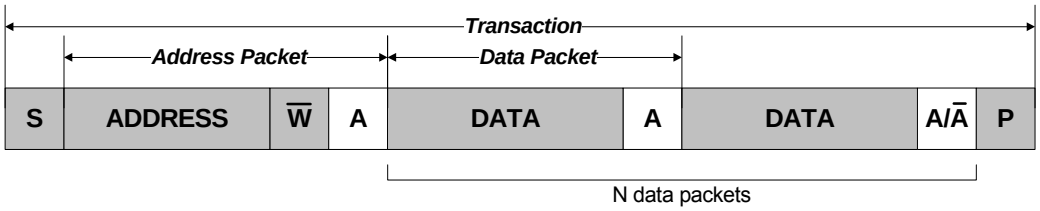
Data packets succeed an address packet or another data packet. All data packets are nine bits long, consisting of one data byte and an acknowledge bit. The direction bit in the previous address packet determines the direction in which the data is transferred.

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 combined transaction.

Figure 19-5 illustrates the Master Write transaction. The master initiates the transaction by issuing a START condition (S) followed by an address packet with direction bit set to zero (ADDRESS+$\overline{W}$).

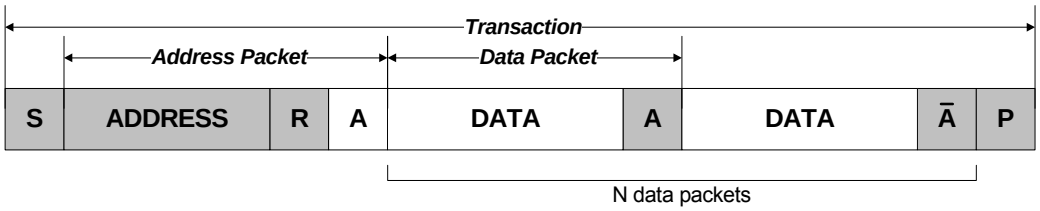
Figure 19-5. Master Write Transaction



Given that 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 signal a NACK to the data, the master must assume that the slave cannot receive any more data and terminate the transaction.

Figure 19-6 illustrates the Master Read transaction. The master initiates the transaction by issuing a START condition followed by an address packet with direction bit set to one (ADRESS+R). The addressed slave must acknowledge the address for the master to be allowed to continue the transaction.

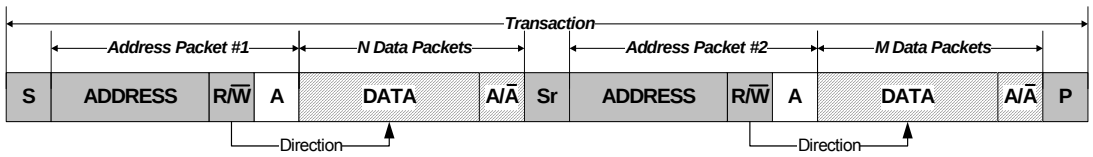
Figure 19-6. Master Read Transaction



Given that 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 a Repeated START conditions (Sr).

Figure 19-7. Combined Transaction

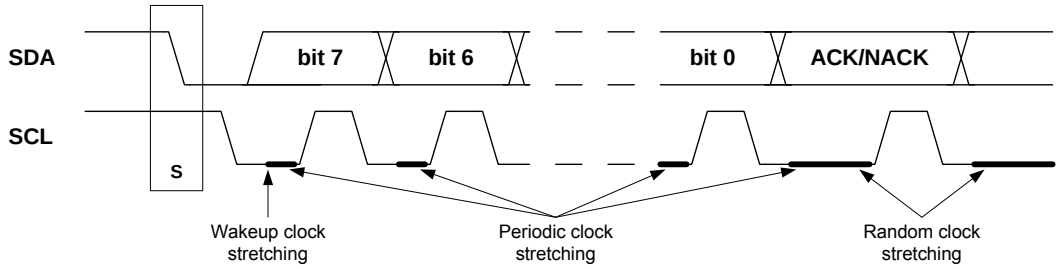


19.3.7 Clock and Clock Stretching

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



If the device is in a sleep mode and a START condition is detected the clock is stretched during the wake-up period for the device.

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 performing 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 only start a bus transaction if it has detected that the bus is idle. As the TWI bus is a multi master bus, it is possible that two devices 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

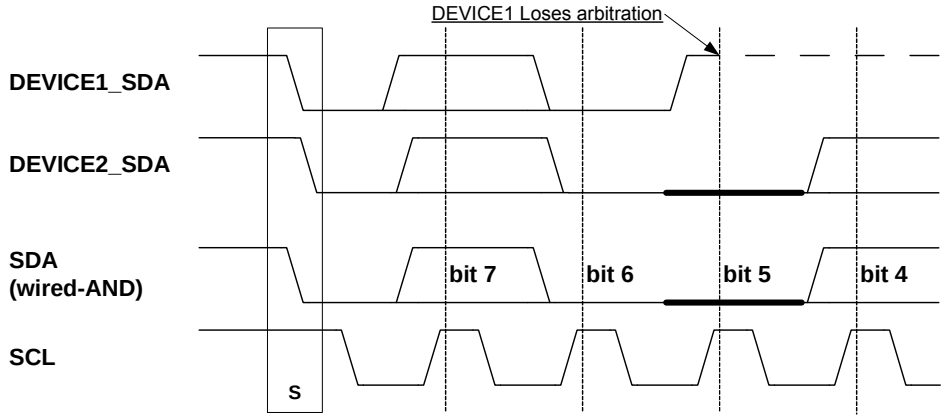


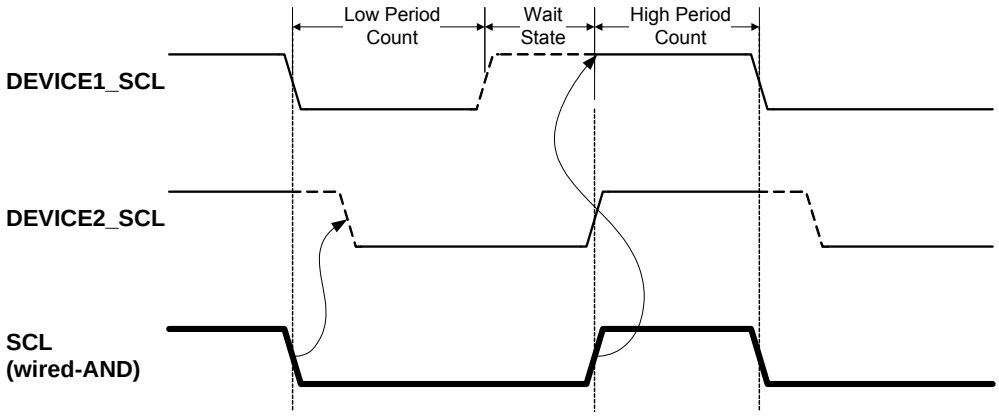
Figure 19-9 shows an example where two TWI masters are contending for bus ownership. Both devices are able to issue a START condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

Arbitration between a repeated START condition and a data bit, a STOP condition and a data bit, or a repeated START condition and 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 clock stretching previously described. Figure 19-10 shows an example where two masters are competing for the 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 start timing their low clock period. The timing length of the low clock period can vary between 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 before 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 become high. The

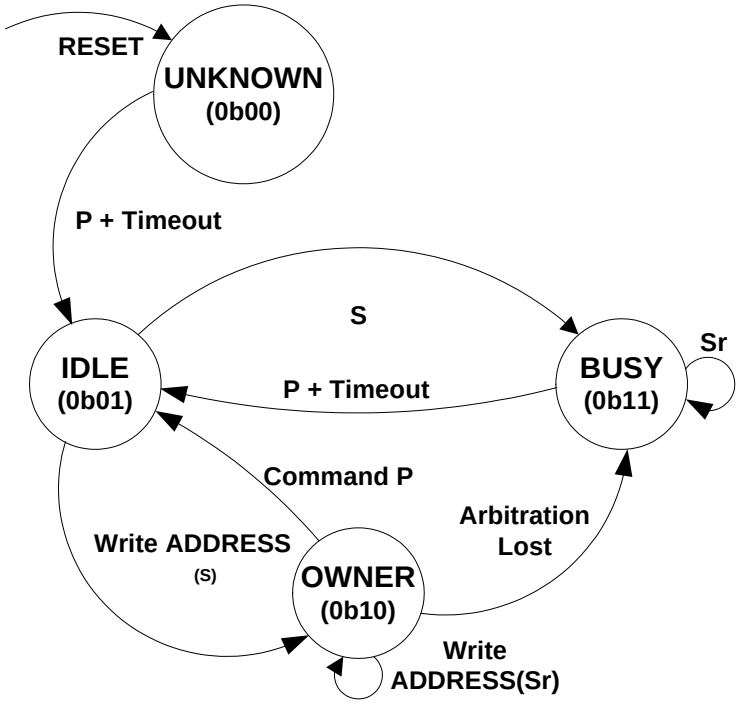
device which first completes its high period (DEVICE1) forces the clock line low and the procedure are then repeated. The result of this is that the device with the shortest clock period determines the high period while the low period of the clock is determined by the longest clock period.

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 bit counter. This is 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 value of the Bus State bits according to state is shown in binary in the figure.

Figure 19-11. Bus State, State Diagram



After a system reset, the bus state is unknown. From this 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 a 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 the bus state will not re-enter the unknown state from any of the other states. Only a system reset or disabling 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 bus state will change from busy to idle on the occurrence of a timeout.

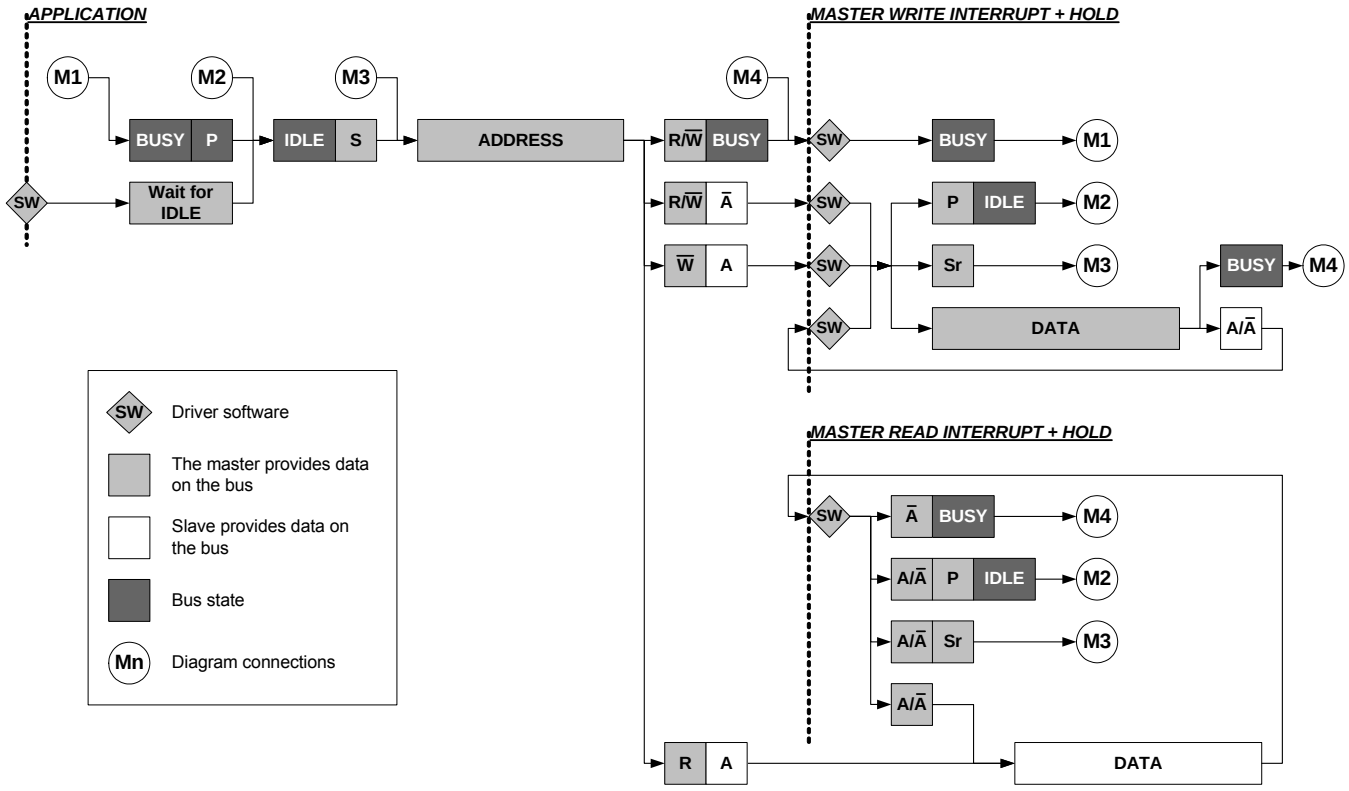
If a START condition is generated internally while in idle state the owner state is entered. If the complete transaction was performed without interference, i.e. no collisions are detected, the master will issue a STOP condition and the bus state changes back to idle. If a collision is detected the arbitration 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.

19.5 TWI Master Operation

The TWI master is byte-oriented with 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 at 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 slave address and direction bit. If the bus is busy the TWI master will wait until the bus becomes idle. When the bus is idle the master will issue a START condition on the bus before the address byte is transmitted.

Depending on arbitration and the R/$\overline{W}$ direction bit one of four distinct cases (1 to 4) 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 Write Interrupt Flag and Arbitration Lost flag.

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 is set and the master Received Acknowledge flag is 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 receiving 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 3 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 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 looses arbitration during transfer, the Arbitration Lost flag is set.

19.5.3 Receiving Data Packets

Assuming case 4 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

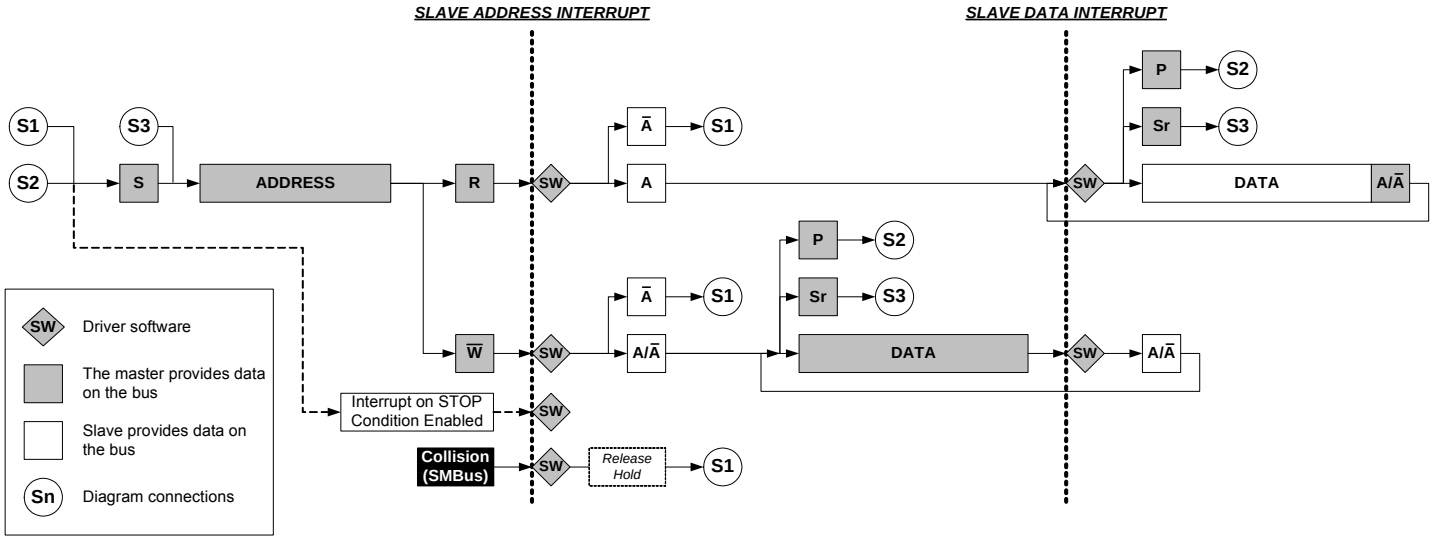
since arbitration can be lost during the transmission. If a collision is detected the master looses arbitration and the Arbitration Lost flag is set.

19.6 TWI Slave Operation

The TWI slave is byte-oriented with optional interrupts after each byte. There are separate slave Data Interrupt and Address/Stop Interrupt. 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 any 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 at 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 the correct address. If the received address is not a match, the slave will not acknowledge the address and wait for a new START condition.

The slave Address/Stop Interrupt Flag is set when a START condition succeeded by a valid address packet 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/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/W direction bit and bus condition one of four distinct cases (1 to 4) arises following the address packet. The different cases must be handled in software.

19.6.1.1 Case 1: 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, 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. If NACK is sent by the slave, the slave will wait for a new condition and address match.

19.6.1.2 Case 2: 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 indicated the slave will wait for a new START condition and address match.

19.6.1.3 Case 3: 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 4: STOP condition received.

Operation is the same as case 1 or 2 above with one exception. 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 drivers 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 pin 0 (Pn0) and 1 (Pn1) is used for SDA and SCL. The external driver interface uses port pin 0 to 3 for the signals SDA_IN, SCL_IN, SDA_OUT and SCL_OUT.

19.8 Register Description - TWI

19.8.1 CTRL– TWI Common Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	-	-	-	SDAHOLD	EDIEN	CTRL
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 - SDAHOLD: SDA Hold Time Enable.**

Setting this bit to one enables an internal hold time on SDA with respect to the negative edge of SCL.

- **Bit 0 - EDIEN: External Driver Interface Enable**

Setting this bit enables the use of the external driver interface, clearing this bit enables normal two wire mode. See Table 19-1 for details.

Table 19-1. 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, no input filter.

19.9 Register Description - TWI Master

19.9.1 CTRLA - TWI Master Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	INTLVL[1:0]		RIEN	WIEN	ENABLE	-	-	-	CTRLA
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**

The Interrupt Level (INTLVL) bit select the interrupt level for the TWI master interrupts.

- **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 unequal zero 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 unequal zero 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 - TWI Master Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	-	TIMEOUT[1:0]		QCEN	SMEN	CTRLB
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 Bits**

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 unequal zero will enable the inactive bus time-out supervisor. If the bus is inactive for longer than the TIMEOUT settings, the bus state logic will enter the idle state.

Figure 19-2 lists the timeout settings.

Table 19-2. TWI master inactive bus timeout settings

TIMEOUT[1:0]	Group Configuration	Description
00	DISABLED	Disabled, normally used for I^2C
01	50US	50 µs, normally used for SMBus at 100 kHz
10	100US	100 µs
11	200US	200 µs

- **Bit 1 - QCEN: Quick Command Enable**

Setting the Quick Command Enable (QCEN) bit enables Quick Command. When Quick Command is enabled, a STOP condition is sent immediate after the slave acknowledges the address.

- **Bit 0 - SMEN: Smart Mode Enable**

Setting the Smart Mode Enable (SMEN) bit enables Smart Mode. When Smart mode is enabled, the Acknowledge Action, as set by the ACKACT bit in Control Register C, is sent immediately after reading the DATA register.

19.9.3 CTRLC - TWI Master Control Register C

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	-	ACKACT	CMD[1:0]		CTRLC
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**

The Acknowledge Action (ACKACT) 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 Control Register B is set, the Acknowledge Action is performed when the DATA register is read.

Table 19-3 lists the acknowledge actions.

Table 19-3. 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 19-4. 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 19-4. CMD Bit Description

CMD[1:0]	MODE	Operation
00	X	Reserved
01	X	Execute Acknowledge Action succeeded by repeated START condition
10	$\overline{W}$	No operation
	R	Execute Acknowledge Action succeeded by a byte receive
11	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 - Master Status Register

Bit	7	6	5	4	3	2	1	0	
+0x03	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]		STATUS
Read/Write	R/W	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 - RIF: Read Interrupt Flag**

This Read Interrupt Flag (RIF) is set when a byte is successfully received in Master Read mode, i.e. no arbitration lost or bus error occurred during the operation. Writing a one to this bit location will clear the 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 automatically cleared 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**

The Write Interrupt Flag (WIF) 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. The WIF is also set if arbitration is lost during sending of 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 the 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 automatically cleared for the same conditions as RIF.

- **Bit 5 - CLKHOLD: Clock Hold**

The master Clock Hold (CLKHOLD) flag is set when the master is holding the SCL line low. This is a status flag, and a read only bit that is set when the RIF and WIF is set. Clearing the interrupt flags and releasing the SCL line, will indirectly clear this flag.

The flag is also automatically cleared for the same conditions as RIF.

- **Bit 4 - RXACK: Received Acknowledge**

The Received Acknowledge (RXACK) flag contains the most recently received acknowledge bit from 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**

The Arbitration Lost (ARBLOST) flag is set if arbitration is lost while transmitting a high data bit, a NACK bit, or while issuing a START or Repeated START condition on the bus. Writing a one to this bit location will clear the ARBLOST flag.

Writing the ADDR register will automatically clear the ARBLOST flag.

- **Bit 2 - BUSERR: Bus Error**

The Bus Error (BUSERR) flag is set if an illegal bus condition has occurred. An illegal bus condition occurs if a Repeated START or 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 the BUSERR flag.

Writing the ADDR register will automatically clear the BUSERR flag.

- **Bit 1:0 - BUSSTATE[1:0]: Bus State**

The Bus State (BUSSTATE) bits indicate the current TWI bus state as defined in Table 19-5. The change of bus state is dependent on bus activity. Refer to the Section 19.4 ”TWI Bus State Logic” on page 212.

Table 19-5. TWI master Bus State

BUSSTATE[1:0]	Group Configuration	Description
00	UNKNOWN	Unknown Bus State
01	IDLE	Idle
10	OWNER	Owner
11	BUSY	Busy

Writing 01 to the BUSSTATE bits forces the bus state logic into 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 - TWI Baud Rate Register

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

$$f_{TWI} = \frac{f_{sys}}{2(5 + TWMBR)}\text{[Hz]} \quad [1]$$

The BAUD register must be set to a value that results in a TWI bus clock frequency (f_{TWI}) equal or less 100 kHz or 400 kHz dependent on standard used by the application. The following equation [2] expresses equation [1] with respect to the BAUD value:

$$TWMBR = \frac{f_{sys}}{2f_{TWI}} - 5 \quad [2]$$

The BAUD register should be written while the master is disabled.

19.9.6 ADDR - TWI Master Address Register

Bit	7	6	5	4	3	2	1	0	
+0x05	ADDR[7:0]								ADDR
Read/Write	R/W	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. The WIF is set.

If the Bus State is unknown when ADDR is written. The WIF is set, and the BUSERR flag 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 -TWI Master Data Register

Bit	7	6	5	4	3	2	1	0	
+0x06	DATA[7:0]								DATA
Read/Write	R/W	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 is 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 protected in 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. The WIF and the CLKHOLD flag are set.

In Master Read mode the RIF and the CLKHOLD flag 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 the WIF and BUSERR flag are set instead of the RIF.

Accessing the DATA register will clear the master interrupt flags and the CLKHOLD flag.

19.10 Register Description - TWI Slave

19.10.1 CTRLA - TWI Slave Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	INTLVL[1:0]		DIEN	APIEN	ENABLE	PIEN	PMEN	SMEN	CTRLA
Read/Write	R/W	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]: TWI Slave Interrupt Level**

The Slave Interrupt Level (INTLVL) bits select the interrupt level for the TWI slave interrupts.

- **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 unequal zero 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 unequal zero for interrupt to be generated.

- **Bit 3 - ENABLE: Enable TWI Slave**

Setting the Enable TWI Slave (ENABLE) bit enables the TWI slave.

- **Bit 2 - PIEN: Stop Interrupt Enable**

Setting the Stop Interrupt Enable (PIEN) bit will set the APIF in the STATUS register when a STOP condition is detected.

- **Bit 1 - PMEN: Promiscuous Mode Enable**

By setting the Promiscuous Mode Enable (PMEN) 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**

Setting the Smart Mode Enable (SMEN) 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 - TWI Slave Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	-	-	ACKACT	CMD[1:0]		CTRLB
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**

The Acknowledge Action (ACKACT) 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 19-6 lists the acknowledge actions.

Table 19-6. TWI slave acknowledge action

ACKACT	Action
0	Send ACK
1	Send NACK

- **Bit 1:0 - CMD[1:0]: Command**

Writing the Command (CMD) bits triggers the slave operation as defined by Table 19-7. 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 19-7. TWI slave command

CMD[1:0]	DIR	Operation
00	X	No action
01	X	Reserved
10	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	**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 the 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, the CLKHOLD flag 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– TWI Slave Status Register

Bit	7	6	5	4	3	2	1	0	
+0x02	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP	STATUS
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 - DIF: Data Interrupt Flag**

The Data Interrupt Flag (DIF) 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 the 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 automatically cleared when writing a valid command to the CMD bits in the CTRLB register

- **Bit 6 - APIF: Address/Stop Interrupt Flag**

The Address/Stop Interrupt Flag (APIF) 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 the APIF. 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.

The flag is also automatically cleared for the same condition as DIF.

- **Bit 5 - CLKHOLD: Clock Hold**

The slave Clock Hold (CLKHOLD) 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 the DIF or APIF is set. Clearing the interrupt flags and releasing the SCL line, will indirectly clear this flag.

- **Bit 4 - RXACK: Received Acknowledge**

The Received Acknowledge (RXACK) 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**

The slave Collision (COLL) flag is set when slave is 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 the COLL flag.

The flag is also automatically cleared when a START or Repeated START condition is detected.

- **Bit 2 - BUSERR: TWI Slave Bus Error**

The slave Buss Error (BUSERR) flag is set when an illegal bus condition has occurs during a transfer. An illegal bus condition occurs if a Repeated START or 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 the BUSERR flag.

For bus errors to be detected, the bus state logic must be enabled. This is done by enable TWI master.

- **Bit 1 - DIR: Read/Write Direction**

The Read/Write 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**

The Slave Address or Stop (AP) flag indicates whether a valid address or a STOP condition caused the last setting of the APIF in the STATUS register.

Table 19-8. 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 - TWI Slave Address Register

Bit	7	6	5	4	3	2	1	0	
+0x03	ADDR[7:0]								ADDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the TWI slave address used by the slave address match logic to determine if a master has addressed the slave. The 7 most significant bits (ADDR[7:1]) represents the slave address and the least significant bit (ADDR[0]) is used for general call address recognition.

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.

When using 10-bit addressing the address match logic only support hardware address recognition of the first byte of a 10-bit address. By setting ADDR[7:1] = "0b11110nn", 'nn' represents bit 9 and 8 for the slave address. The next byte received is bit 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, the 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.

19.10.5 DATA - TWI Slave Data Register

Bit	7	6	5	4	3	2	1	0	
+0x04	DATA[7:0]								DATA
Read/Write	R/W	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 is 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 protected in hardware. The Data register can only be accessed 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 start to clock the data byte from the slave, followed by the slave receiving the acknowledge bit from the master. The DIF and the CLKHOLD flag are set.

When a master write data to the slave the DIF and the CLKHOLD flag 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.

Accessing the DATA register will clear the slave interrupt flags and the CLKHOLD flag.

19.10.6 ADDRMASK - TWI Slave Address Mask Register

Bit	7	6	5	4	3	2	1	0	
+0x05	ADDRMASK[7:1]							ADDREN	ADDRMASK
Read/Write	R/W	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]: Read/Write Direction**

These bits in the ADDRMASK register can act as a second address match register, or 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 2 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 2 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	EDIEN	217
+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	-	-	-	217
+0x01	CTRLB	-	-	-	-	TIMEOUT[1:0]		QCEN	SMEN	218
+0x02	CTRLC	-	-	-	-	-	ACKACT	CMD[1:0]		219
+0x03	STATUS	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]		220
+0x04	BAUD	BAUD[7:0]								221
+0x05	ADDR	ADDR[7:0]								222
+0x06	DATA	DATA[7:0]								222

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	223
+0x01	CTRLB	-	-	-	-	-	ACKACT	CMD[1:0]		223
+0x02	STATUS	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP	224
+0x03	ADDR	ADDR[7:0]								226
+0x04	DATA	DATA[7:0]								226
+0x05	ADDRMASK	ADDRMASK[7:1]							ADDREN	227

19.14 Interrupt Vector Summary

Table 19-9. TWI Interrupt vectors and their word offset addresses

Offset	Source	Interrupt Description
0x00	SLAVE_vect	TWI Slave Interrupt vector
0x02	MASTER_vect	TWI Master Interrupt vector

20. SPI – Serial Peripheral Interface

20.1 Features

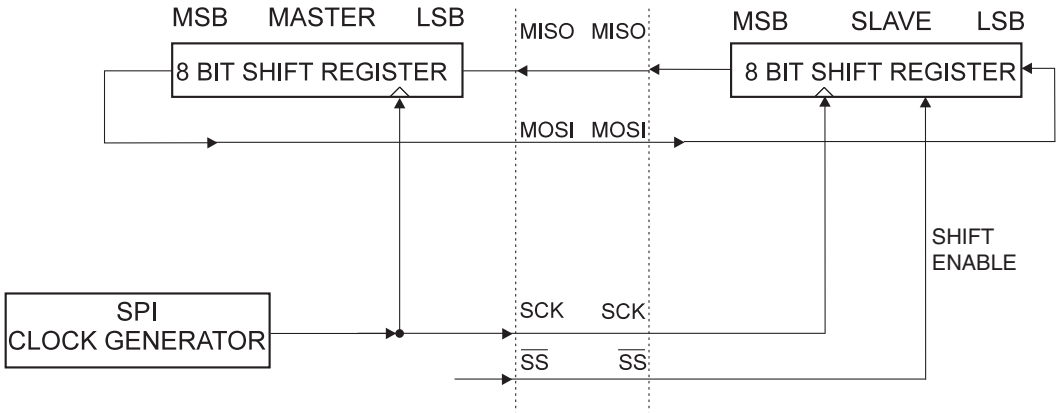
- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Eight Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI 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 AVR devices. 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 CPUs with SPI is shown in Figure 20-1 on page 229. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select ($\overline{SS}$) pin of 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 is always shifted from Master to Slave on the Master Out - Slave In (MOSI) line, and from Slave to Master on the Master In - Slave Out (MISO) line. After each data packet, the Master can synchronize the Slave by pulling high the $\overline{SS}$ line.

Figure 20-1. SPI Master-slave Interconnection



The XMEGA SPI module is single buffered in the transmit direction and double 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 is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of this clock signal, the minimum low and high periods must be:

Low period: longer than 2 CPU clock cycles.

High period: longer than 2 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 overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

20.3 Master Mode

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. The $\overline{SS}$ pin 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 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 signal the end of 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 configured as an input, it must be held high to ensure Master operation. If the $\overline{SS}$ pin is input and being driven low by external circuitry, 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

When configured as a Slave, the SPI interface will remain sleeping with MISO 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 and assuming the MISO pin is configured as output, the Slave will start to shift out data on the first SCK clock pulse. As one byte has been completely shifted, the SPI Interrupt Flag is set. The Slave may continue to place 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 data. Any partially received packet in the shift register will be dropped.

As the $\overline{SS}$ pin is used to signal start and end of 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

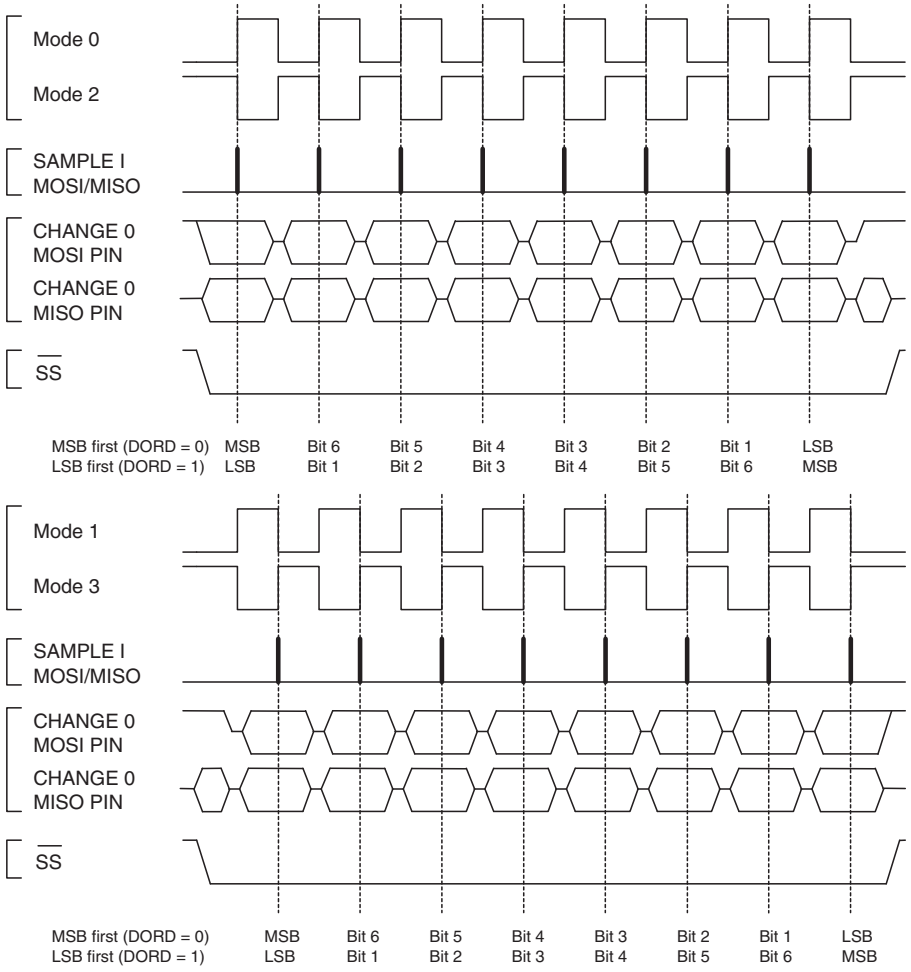
There are four combinations of SCK phase and polarity with respect to serial data. The SPI data transfer formats are shown in Figure 20-2. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize.

Table 20-2. SPI Modes

Mode	Leading Edge	Trailing Edge
0	Rising, Sample	Falling, Setup
1	Rising, Setup	Falling, Sample
2	Falling,Sample	Rising, Setup
3	Falling, Setup	Rising, Sample

Leading edge is the first clock edge in a clock cycle. Trailing edge is the last clock edge in a clock cycle.

Figure 20-2. SPI Transfer modes



20.6 DMA Support

DMA support on the SPI module is only available in Slave mode. The SPI Slave can trigger a DMA transfer as one byte has been shifted into the Data Register. It is possible to set up the XMEGA USART in SPI mode to have DMA support for master mode, for details refer to Section 21.10 ”USART in Master SPI Mode” on page 247.

20.7 Register Description

20.7.1 CTRL - SPI Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	CLK2X	ENABLE	DORD	MASTER	MODE[1:0]		PRESCALER[1:0]		CTRL
Read/Write	R/W	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: SPI Clock Double**

When this bit is set the SPI speed (SCK Frequency) will be doubled in Master mode (see Table 20-4 on page 233).

- **Bit 6 - ENABLE: SPI Enable**

Setting this bit enables the SPI modules. This bit must be set to enable any SPI operations.

- **Bit 5 - DORD: Data Order**

DORD decide the data order when a byte is shifted out from the Data register. When DORD is written to one, the LSB of the data byte is transmitted first, and when DORD is written to zero, the MSB of the data byte is transmitted first.

- **Bit 4 - MASTER: Master/Slave 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 is driven low while MASTER is set, MASTER will be cleared.

- **Bit 3:2 - MODE[1:0]: SPI Mode**

These bits select the transfer mode. The four combinations of SCK phase and polarity with respect to serial data is shown in Figure 20-3 on page 232. This decide whether the first edge in a clock cycles (leading edge) is rising or falling, and if data setup and sample is on lading or trailing edge.

When the leading edge is rising the bit SCK is low when idle, and when the leading edge is falling the SCK is high when idle.

Table 20-3. SPI transfer modes

MODE[1:0]	Group Configuration	Leading Edge	Trailing Edge
00	0	Rising, Sample	Falling, Setup
01	1	Rising, Setup	Falling, Sample
10	2	Falling,Sample	Rising, Setup
11	3	Falling, Setup	Rising, Sample

- **Bits 1:0 - PRESCALER[1:0]: SPI Clock Prescaler**

These two bits control the SCK rate of the device configured in a 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 20-4 on page 233.

Table 20-4. Relationship Between SCK and the Peripheral Clock (clk_{PER}) frequency

CLK2X	PRESCALER[1:0]	SCK Frequency
0	00	$clk_{PER}/4$
0	01	$clk_{PER}/16$
0	10	$clk_{PER}/64$
0	11	$clk_{PER}/128$
1	00	$clk_{PER}/2$
1	01	$clk_{PER}/8$
1	10	$clk_{PER}/32$
1	11	$clk_{PER}/64$

20.7.2 INTCTRL - SPI Interrupt Control Register

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

- **Bits 7:2 - Reserved**

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

- **Bits 1:0 - INTLVL[1:0]: SPI Interrupt Level**

These bits enable the SPI Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will be triggered when the IF in the STATUS register is set.

20.7.3 STATUS - SPI Status Register

Bit	7	6	5	4	3	2	1	0	
+0x02	SPIF	WCOL	-	-	-	-	-	-	STATUS
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - IF: SPI Interrupt Flag**

When a serial transfer is complete and one byte is completely shifted in/out of the DATA register, the IF bit is set. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the IF bit. The IF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit can be cleared by first reading the STATUS register with IF set, and then access the DATA register.

- **Bit 6 - WRCOL: Write Collision Flag**

The WRCOL bit is set if the DATA register is written during a data transfer. The WRCOL bit is cleared by first reading the STATUS register with WRCOL set, and then accessing the DATA register.

- **Bit 5:0 - Reserved**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

20.7.4 DATA - SPI Data Register

Bit	7	6	5	4	3	2	1	0	
+0x03	DATA[7:0]								DATA
Read/Write	R/W	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 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, and return the last bytes 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]		232
+0x01	INTCTRL	-	-	-	-	-	-	INTLVL[1:0]		233
+0x02	STATUS	IF	WRCOL	-	-	-	-	-	-	233
+0x03	DATA	DATA[7:0]								234

20.9 SPI Interrupt vectors

Table 20-5. SPI Interrupt vector and its offset word address

Offset	Source	Interrupt Description
0x00	SPI_vect	SPI Interrupt vector

21. USART

21.1 Features

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- Enhanced Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun and Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode
- Master SPI mode, Three-wire Synchronous Data Transfer
  - Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)
  - LSB First or MSB First Data Transfer (Configurable Data Order)
  - Queued Operation (Double Buffered)
  - High Speed Operation ($f_{XCK,max} = f_{PER}/2$)
- IRCOM Module for IrDA compliant pulse modulation/demodulation

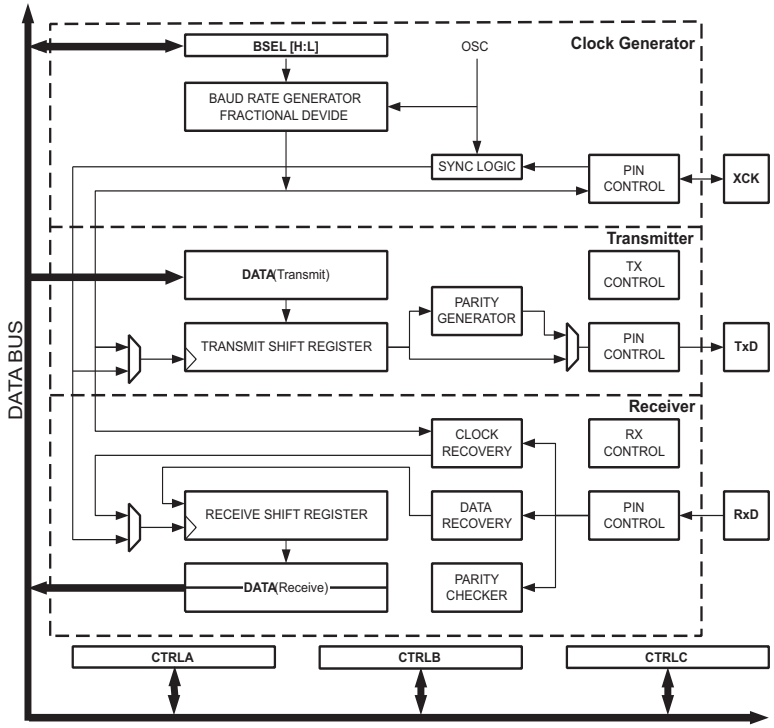
21.2 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication module. The USART supports full duplex communication, and both asynchronous and clocked synchronous operation. The USART can be set in Master SPI compliant mode and be 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 direction, enabling continued data transmission without any delay between frames. There are separate interrupt vectors for receive and transmit complete, enabling 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 236. The main parts are the Clock Generator, the Transmitter and the Receiver, indicated in dashed boxes.

Figure 21-1. USART Block Diagram



The Clock Generation logic has a fractional baud rate generator that is able to generate a wide range of USART baud rates. It also includes synchronization logic for external clock input in synchronous slave operation.

The Transmitter consists of a single write buffer (DATA), a shift register, Parity Generator and control logic for handling different frame formats. The write buffer allows continuous data transmission without any delay between frames.

The Receiver consists of a two level FIFO 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.

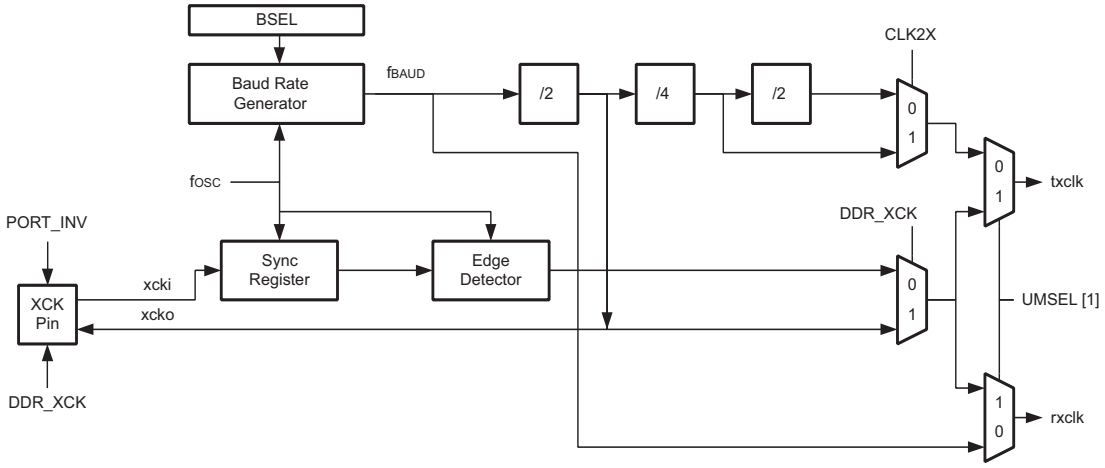
When the USART is set in Master SPI compliant mode, all USART specific logic is disabled, leaving the transmit and receive buffers, shift registers, and Baud Rate Generator enabled. Pin control and interrupt generation is identical in both modes. The registers are used in both modes, but the functionality 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 115.2 kbps. For details refer to Section 22. ”IRCOM - IR Communication Module” on page 256 for details.

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.I



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 SPI master mode operation. The generated output frequency (f_{BAUD}) is given by the period setting (BSEL), an optional scale setting (BSACLE) and the Peripheral Clock frequency (f_{PER}). Table 21-1 on page 238 contains equations for calculating the baud rate (in bits per second) and for calculating the BSEL value for each mode of operation. BSEL can be set to any value between 0 and 4095. It also show the maximum baud rate versus peripheral clock speed.

Fractional baud rate generation can be used in asynchronous mode of operation to increase the average resolution. A scale factor (BSCALE) allows the baud rate to be optionally left or right scaled. Choosing a positive scale value will results in right scaling, which increase the period and consequently reduce the frequency of the produced baud rate, without changing the resolution. If the scale value is negative the divider uses fractional arithmetic counting to increase the resolution by distributing the fractional divide value over time. BSCALE can be set to any value from -7 to +7, where 0 implies no scaling. There is a limit to how high the scale factor can be and the value 2^{BSCALE} must be at least half of the minimum number of clock cycles a frame takes, see Section 21.9 on page 246 for more details.

Table 21-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Conditions	Equation for Calculation Baud Rate[1]	Equation for Calculation BSEL Value
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 SPI Master 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 bit per second (bps)

21.3.2 External Clock

External clock is used in synchronous slave mode operation. The XCK clock input is sampled on the Peripheral Clock frequency (f_{PER}) by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register is then passed through an edge detector. This process introduces a delay of two peripheral clock periods, and therefore the maximum external XCK clock frequency (f_{XCK})is limited by the following equation:

$$f_{XCK} < \frac{f_{PER}}{4}$$

Each high and low period the XCK clock cycles must be sampled twice by the Peripheral Clock. If the XCK clock has jitter, or the high/low period duty cycle is not 50/50, the maximum XCK clock speed must be reduced accordingly.

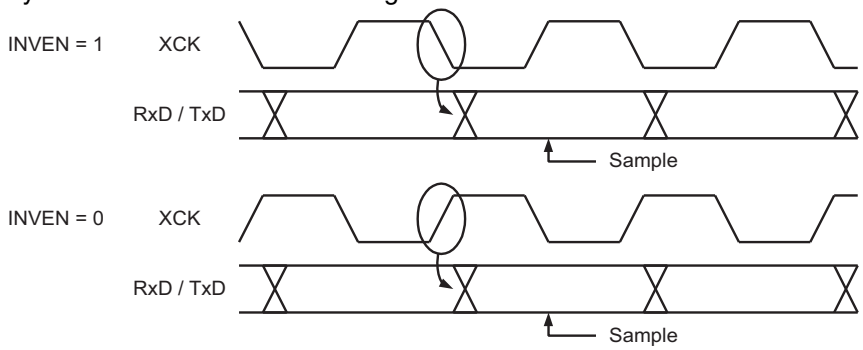
21.3.3 Double Speed Operation (CLK2X)

Double Speed operation can be enabled to allow for higher baud rates on lower peripheral clock frequencies under asynchronous operation. When Double Speed operation is enabled the baud rate for a given asynchronous baud rate setting as shown in Table 21-1 on page 238 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 more accurate baud rate setting and peripheral clock are required. See Section 21.8 ”Asynchronous Data Reception” on page 243 for more details on accuracy.

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 and 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 opposite XCK clock edge of the edge where data output (TxD) is changed.

Figure 21-3. Synchronous Mode XCKn Timing.



Using the Inverted I/O (INVEN) setting in the Pin Configuration Register for the corresponding XCK port pin, it is selectable which XCK clock edge is used for data sampling and which is used for data change. If inverted I/O is disabled (INVEN=0) data will be changed at rising XCK clock edge and sampled at falling XCK clock edge. If inverted I/O is enabled (INVEN=1) data will be changed at falling XCK clock edge and sampled at rising XCK clock edge. For more details, see in “I/O Ports” on page 106.

21.3.5 SPI Clock Generation

For SPI operation only master mode with internal clock generation is supported. This is identical to the USART synchronous master mode and the baud rate or BSEL setting are calculated by using the same equations, see Table 21-1 on page 238.

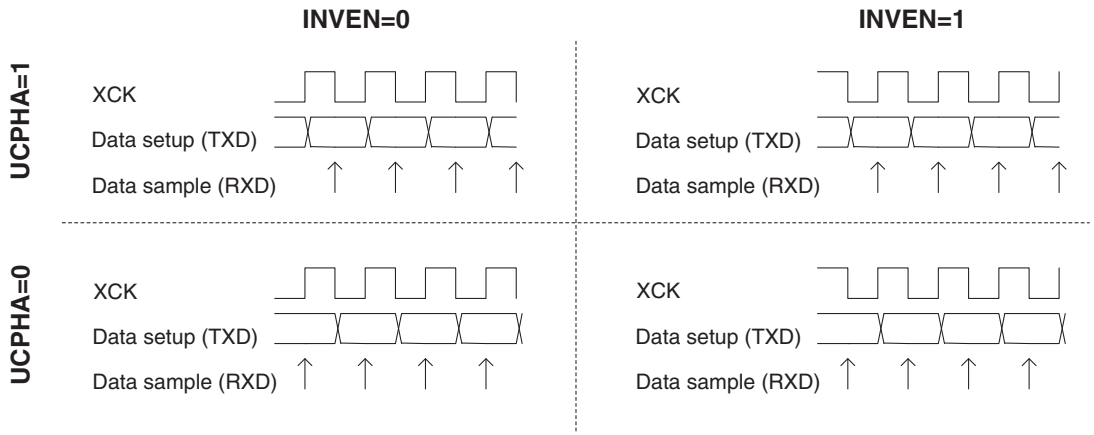
There are four combinations of the XCK (SCK) clock phase and polarity with respect to serial data, and these are determined by the Clock Phase (UCPHA) control bit and the Inverted I/O pin (INVEN) setting. The data transfer timing diagrams are shown in Figure 21-4 on page 240. Data bits are shifted out 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 on page 239. Changing the setting of any of these bits during transmission will corrupt for 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

Leading edge is the first clock edge in a clock cycle. Trailing edge is the last clock edge in 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 SPI operation (See Section 21.4.2 ”SPI Frame Formats” on page 241). The USART accepts all 30 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 the least significant data bit and all data bits ending with the most significant bit. 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 idle (high) state. Figure 21-5 on page 240 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 21-5. Frame Formats

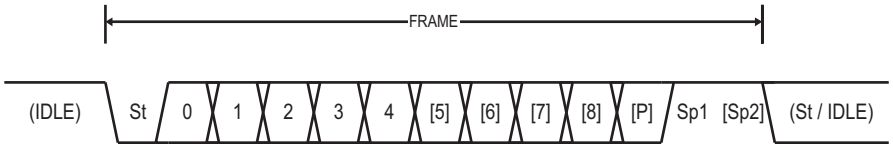


Table 1.

St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can 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 data bits that is one 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 data bits that is one 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 8 data bits. The USART in Master SPI mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

When a complete frame of 8 bits is transmitted, a new frame can directly follow it, or the communication line returns to idle (high) state.

21.5 USART Initialization

USART initialization should use the following sequence:

1. Set the TxD pin value high, and optionally 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 mode of operation (enables the 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 changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The transit and receive complete interrupt flags can be used to check that the Transmitter has completed all transfers, and that there are no unread data in the receive buffer.

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 in the corresponding port. For details on port pin control refer to ”I/O Ports” on page 129.

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 is 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 less than eight bits, the most significant bits written to the 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 Transmitter is disabled it will no longer override the TxDn pin and the pin direction is set as input.

21.7 Data Reception - The USART Receiver

When the Receiver is enabled, the RxD pin is given the function 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 less 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 Flags is cleared.

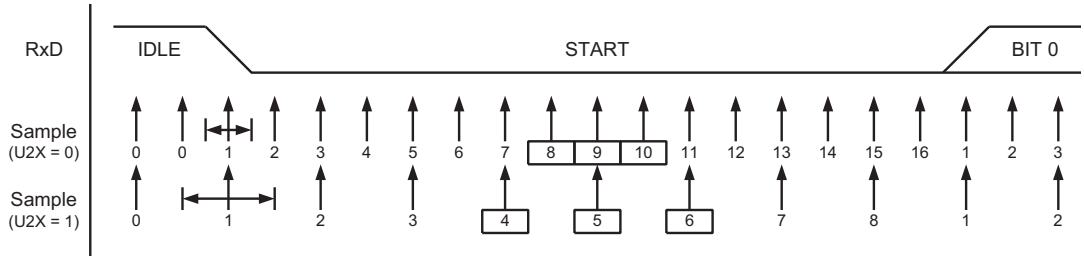
21.8 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the incoming asynchronous serial frames at the RxD pin to the internally generated baud rate clock. The data recovery logic 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 logic synchronizes internal clock to the incoming serial frames. Figure 21-6 on page 243 illustrates the sampling process of 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 zero are samples done when the RxD line is idle, i.e. no communication activity.

Figure 21-6. Start Bit Sampling

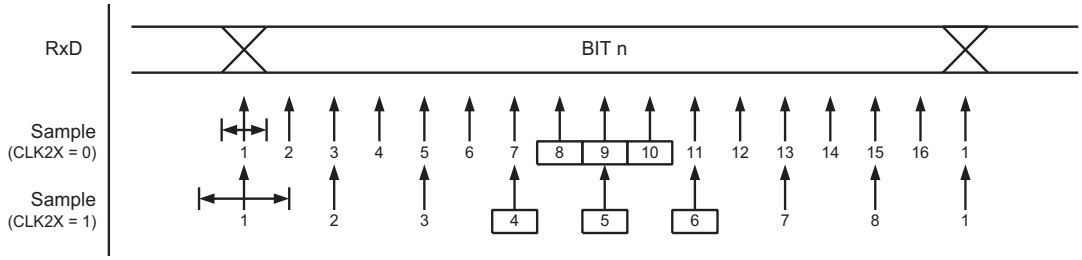


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Sample 1 denotes the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure) to decide if a valid start bit is received. If two or more of these three samples have a low level (the majority wins), the start bit is accepted. The clock recovery logic is synchronized and the data recovery can begin. If two or more of the three samples have a high level the start bit is rejected as a noise spike and the Receiver starts looking for the next high to low-transition. The synchronization 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 244 shows the sampling process of data and parity bits.

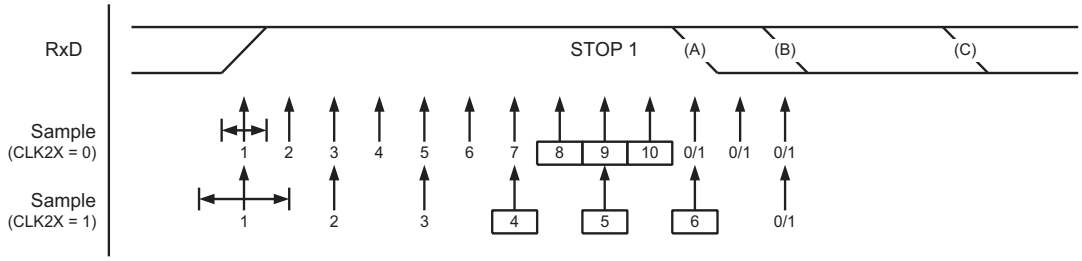
Figure 21-7. Sampling of Data and Parity Bit



As for start bit detection, identical majority voting technique is used on the three center samples (indicated with sample numbers inside boxes) for deciding of the logic level of the received bit. This majority voting process acts as a low pass filter for the received signal on the RxD pin. The process is repeated for each bit until a complete frame is received. Including the first, but excluding additional stop bits. If the stop bit sampled has a logic 0 value, the Frame Error (FERR) Flag will be set.

Figure 21-8 on page 244 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 Sampling 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 point marked (A) in Stop Bit Sampling and Next Start Bit Sampling. For Double Speed mode the first low level must be delayed to (B). (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 on bit rates that are too fast or too slow, or 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.

Table 1.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

Table 1.

D	Sum of character size and parity size (D = 5 to 10 bit).
S	Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
S_F	First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for Double Speed mode.
S_M	Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for Double Speed mode.
R_{slow}	Is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate.
R_{fast}	Is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 21-3 and Table 21-4 on page 245 list the maximum receiver baud rate error that can be tolerated. Normal Speed mode has higher toleration of baud rate variations.

Table 21-3. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (CLK2X = 0)

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 (CLK2X = 1)

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

Table 21-4. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (CLK2X = 1) (Continued)

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 of the maximum receiver baud rate error was made under the assumption that the Receiver and Transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The Receiver's system clock will always have some minor instability. In addition, the baud rate generator can not always do an exact division of the peripheral clock frequency to get the baud rate wanted. In this case the BSEL and BSCALE value should be selected to give the lowest possible error.

21.9 The Impact of Fractional Baud Rate Generation

Fractional baud rate generation is possible for asynchronous operation due to the relatively high number of clock cycles (i.e. samples) for each frame. Each bit is sampled sixteen times, but only the center samples are of importance. This leaves some slack for each bit. Not only that, but 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 assumes that normal speed mode is used, the total number of samples for a frame is, (1+8+1)*16, or 160. As earlier stated, the UART can tolerate plus minus some samples. The critical factor is the time from the falling edge of the start bit (i.e. the clock synchronization) to the last bit's (i.e. the first stop bit) value is recovered.

Standard baud rate generators have the unwanted property of having large frequency steps between high baud rate settings. Worst case is found between BSEL value 0x000 and 0x001. Going from an BSEL value of 0x000 for which has a 10-bit frame of 160 samples, to an BSEL value 0x001 with 320 samples, shows a 50% change in frequency. However, when increasing the BSEL values the step change will quickly decrease. 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 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 tick each time the counter reaches zero, and a sample of the received signal on RXD is taken for each baud rate clock tick. For the fractional baud rate generator the count sequence can have an uneven period:

2, 1, 0, 3, 2, 1, 0, 2, 1, 0, 3, 2, …

In this example an extra cycle is added every second cycle. This gives a baud rate clock tick jitter, but the average period has been increased by a fraction, more precisely 0.5 clock cycles.

The impact of the 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

samples per 10-bit frame compared to the previous from 160 to 320. Higher negative scale factor gives even finer granularity. There is a limit to how high the scale factor can be. A rule of thumb is that the value 2^{BSCALE} must be at least half of the minimum number of clock cycles a frame takes. 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.

21.10 USART in Master SPI Mode

Using the USART in Master SPI mode (MSPIM) 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 USART a data transfer is initiated by writing to the DATA location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to DATA is 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 in Master SPI mode are identical in function to the normal USART operation. The receiver error status flags are not in use and is always read as zero.

Disabling of the USART transmitter or receiver in Master SPI mode is identical in function to the normal USART operation.

21.11 USART SPI vs. SPI

The USART in Master SPI mode is fully compatible with the SPI regarding:

- Master mode timing diagram.
- The UCPHA bit functionality is identical to the SPI CPHA bit.
- The UDORD bit functionality is identical to the SPI DORD bit.

Since the USART in Master SPI mode reuses the USART resources, the use of the USART in MSPIM is somewhat different compared to the XMEGA SPI module. In addition to differences of the control register bits and no SPI slave support, the following features differ between the two modules:

- The Transmitter USART in Master SPI mode includes buffering. The XMEGA SPI has no transmit buffer.
- The Receiver in USART in Master SPI includes an additional buffer level.
- The SPI WCOL (Write Collision) bit is not included in USART in Master SPI mode.
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting BSEL accordingly.
- Interrupt timing is not compatible.
- Pin control differs due to the master only operation of the USART in Master SPI mode.

A comparison of the USART in Master SPI mode and the SPI pins is shown Table 21-5.

Table 21-5. 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

21.12 Multi-processor Communication Mode

Enabling the Multi-processor Communication Mode (MPCM) effectively reduces the number of incoming frames that has to be handled by the Receiver in a system with multiple MCUs 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.

If the Receiver is set up to receive frames that contain 5 to 8 data bits, the first stop bit is used to indicate the frame type. If the Receiver is set up for frames with 9 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. The Transmitter is unaffected by the MPCM setting, but if 5- to 8-bit character frames are used, the Transmitter must be set to use two stop bit 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 normal, while the other slave MCUs will ignore the received frames until another address frame is received.

21.12.1 Using Multi-processor Communication Mode

For an MCU to act as a master MCU, it should use a 9-bit character frame format. The ninth bit must be set when an address frame is being transmitted and cleared when a data frame is being transmitted. The slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication mode:

1. All Slave MCUs are in Multi-processor 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 new address frame from the Master. The process then repeats from 2.

Using any of the 5- to 8-bit character frame formats is possible, but impractical since the Receiver must change between using n and n+1 character frame formats. This makes full duplex operation difficult since the Transmitter and Receiver uses 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 physical compliant modulation and demodulation for baud rates up to 115.2 Kbps. When IRCOM mode is enabled, Double Transmission Speed cannot be used for the USART.

For devices with more than one USART, IRCOM mode can only be enabled for one USART at a time. For details refer to Section 22. ”IRCOM - IR Communication Module” on page 256.

21.14 DMA Support

DMA support is available on the UART, USRT and SPI Master mode peripherals. For details on different USART DMA transfer triggers refer to Section 5.4 ”Transfer Triggers” on page 50.

21.15 Register Description

21.15.1 DATA - USART I/O Data Register

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

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register (DATA). The Transmit Data Buffer Register (TXB) will be the destination for data written to the DATA Register location. Reading the DATA Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the DREIF Flag in the STATUS Register is set. Data written to DATA when the DREIF Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. The data is then transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO and the corresponding flags in the Status Register (STATUS) will change state whenever the receive buffer is accessed (read). Always read STATUS before DATA in order to get the correct flags.

21.15.2 STATUS - USART Status Register

Bit	7	6	5	4	3	2	1	0	
+0x01	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	-	RXB8	STATUS
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: USART Receive Complete Interrupt Flag**

This flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). When the Receiver is disabled, the receive buffer will be flushed and consequently the RXCIF will become zero.

When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from DATA in order to clear the RXCIF. If not, a new interrupt will occur directly after the return from the current interrupt. This flag can also be cleared by writing a one to its bit location.

- **Bit 6 - TXCIF: USART Transmit Complete Interrupt Flag**

This flag is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data in the transmit buffer (DATA). The TXCIF is automatically cleared when the transmit complete interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

- **Bit 5 - DREIF: USART Data Register Empty Flag**

The DREIF indicates if the transmit buffer (DATA) is ready to receive new data. The flag is one when the transmit buffer is empty, and zero when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. DREIF is set after a reset to indicate that the Transmitter is ready. Always write this bit to zero when writing the STATUS register.

DREIF is cleared by writing DATA. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to DATA in order to clear DREIF or disable the Data Register Empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.

- **Bit 4 - FERR: Frame Error**

The FERR flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The bit is set if the received character had a Frame Error, i.e. when the first stop bit was zero, and cleared when the stop bit of the received data is one. This bit is valid until the receive buffer (DATA) is read. The FERR is not affected by setting the SBMODE bit in CTRLC since the Receiver ignores all, except for the first stop bit. Always write this bit location to zero when writing the STATUS register.

This flag is not used in Master SPI mode of operation.

- **Bit 3 - BUFOVF: Buffer Overflow**

The BUFOVF flag indicates data loss due to a receiver buffer full condition. This flag is set if a Buffer Overflow condition is detected. A Buffer Overflow occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, 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 of operation.

- **Bit 2 - PERR: Parity Error**

If parity checking is enabled and the next character in the receive buffer has a Parity Error this flag is set. If Parity Check is not enabled the PERR will always be read as zero. This bit is valid until the receive buffer (DATA) is read. Always write this bit location to zero when writing the STATUS register. For details on parity calculation refer to ”Parity Bit Calculation” on page 241.

This flag is not used in Master SPI mode of operation.

- **Bit 1 - 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 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 unused in Master SPI mode of operation.

21.15.3 CTRLA – USART Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		CTRLA
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 Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will be triggered when the RXCIF in the STATUS register is set.

- **Bit 3:2 - TXCINTLVL[1:0]: Transmit Complete Interrupt Level**

These bits enable the Transmit Complete Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will be triggered when the TXCIF in the STATUS register is set.

- **Bit 1:0 - DREINTLVL[1:0]: USART Data Register Empty Interrupt Level**

These bits enable the Data Register Empty Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will be triggered when the DREIF in the STATUS register is set.

21.15.4 CTRLB - USART Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x04	-	-	-	RXEN	TXEN	CLK2X	MPCM	TXB8	CTRLB
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 of operation.

- **Bit 1 - MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, the USART Receiver ignores all the incoming frames that do not contain address information. The Transmitter is unaffected by the MPCM setting. For more detailed information see ”Multi-processor Communication Mode” on page 248.

This bit is unused in Master SPI mode of 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 of operation.

21.15.5 CTRLC - USART 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]: USART Communication Mode**

These bits select the mode of operation of the USART as shown in Table 21-6.

Table 21-6. 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 Section 22. ”IRCOM - IR Communication Module” on page 256 for full description on using IRCOM mode.

2. See "USART" on page 235 for full description of the Master SPI Mode (MSPIM) operation.

- **Bits 5:4 - PMODE[1:0]: Parity Mode**

These bits enable and set the type of parity generation according to Table 21-7 on page 253. 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 of operation.

Table 21-7. PMODE Bits 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 21-8 on page 253. The Receiver ignores this setting.

This bit is unused in Master SPI mode of operation.

Table 21-8. SBMODE Bit Settings

SBMODE	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:0 - CHSIZE[2:0]: Character Size**

The CHSIZE[2:0] bits sets the number of data bits in a frame according to Table 21-9 on page 253. The Receiver and Transmitter use the same setting.

Table 21-9. CHSIZE Bits Settings

CHSIZE[2:0]	Group Configuration	Character size
000	5BIT	5-bit
001	6BIT	6-bit
010	7BIT	7-bit
011	8BIT	8-bit
100		Reserved
101		Reserved
110		Reserved
111	9BIT	9-bit

- **Bit 2 - UDORD: Data Order**

This bit sets the frame format. When written to one the LSB of the data word is transmitted first. When written to zero the MSB of the data word is transmitted first. The Receiver and Transmitter use the same setting. Changing the setting of UDORD will corrupt all ongoing communication for both receiver and transmitter.

- **Bit 1 - UCPHA: Clock Phase**

The UCPHA bit setting determine if data is sampled on the leading (first) edge or tailing (last) edge of XCKn. Refer to the ”SPI Clock Generation” on page 239 for details.

21.15.6 BAUDCTRLA - USART Baud Rate Register

Bit	7	6	5	4	3	2	1	0	
+0x06	BSEL[7:0]								BAUDCTRLA
Read/Write	R/W	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]: USART Baud Rate Register**

This is a 12-bit value which contains the USART baud rate setting. The BAUDCTRLB contains the four most significant bits, and the BAUDCTRLA contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing BAUDCTRLA will trigger an immediate update of the baud rate prescaler.

21.15.7 BAUDCTRLB - USART Baud Rate Register

Bit	7	6	5	4	3	2	1	0	
+0x07	BSCALE[3:0]				BSEL[11:8]				BAUDCTRLB
Read/Write	R/W	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]: USART Baud Rate Scale factor**

These bits select the Baud Rate Generator scale factor. The scale factor is given in two's complement form from -7 (0b1001) to 7 (0b0111). The -8 (0b1000) setting is reserved. For positive scale values the Baud Rate Generator is prescaled by 2^{BSCALE}. For negative values the Baud Rate Generator will use fractional counting, which increases the resolution. See equations in Table 21-1 on page 238.

- **Bit 3:0 - BSEL[3:0]: USART Baud Rate Register**

This is a 12-bit value which contains the USART baud rate setting. The BAUDCTRLB contains the four most significant bits, and the BAUDCTRLA contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing BAUDCTRLA will trigger an immediate update of the baud rate prescaler.

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]								249
+0x01	STATUS	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	-	RXB8	249
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	CTRLA	-	-	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		251
+0x04	CTRLB	-	-	-	RXEN	TXEN	CLK2X	MPCM	TXB8	251
+0x05	CTRLC	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]			253
+0x06	BAUDCTRLA	BSEL[7:0]								255
+0x07	BAUDCTRLB	BSCALE[3:0]				BSEL[11:8]				254

21.16.2 Register Description - USART in Master SPI Mode

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DATA	DATA[7:0]								249
+0x01	STATUS	RXCIF	TXCIF	DREIF	-	-	-	-	-	249
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	CTRLA	-	-	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		251
+0x04	CTRLB	-	-	-	RXEN	TXEN	-	-	-	251
+0x05	CTRLC	CMODE[1:0]		-	-	-	UDORD	UCPH	-	252
+0x06	BAUDCTRLA	BSEL[7:0]								254
+0x07	BAUDCTRLB	BSCALE[3:0]				BSEL[11:8]				254

21.17 Interrupt Vector Summary

Table 21-10. USART Interrupt vectors and their word offset address

Offset	Source	Interrupt Description
0x00	RXC_vect	USART Receive Complete Interrupt vector
0x02	DRE_vect	USART Data Register Empty Interrupt vector
0x04	TXC_vect	USART Transmit Complete Interrupt vector

22. IRCOM - IR Communication Module

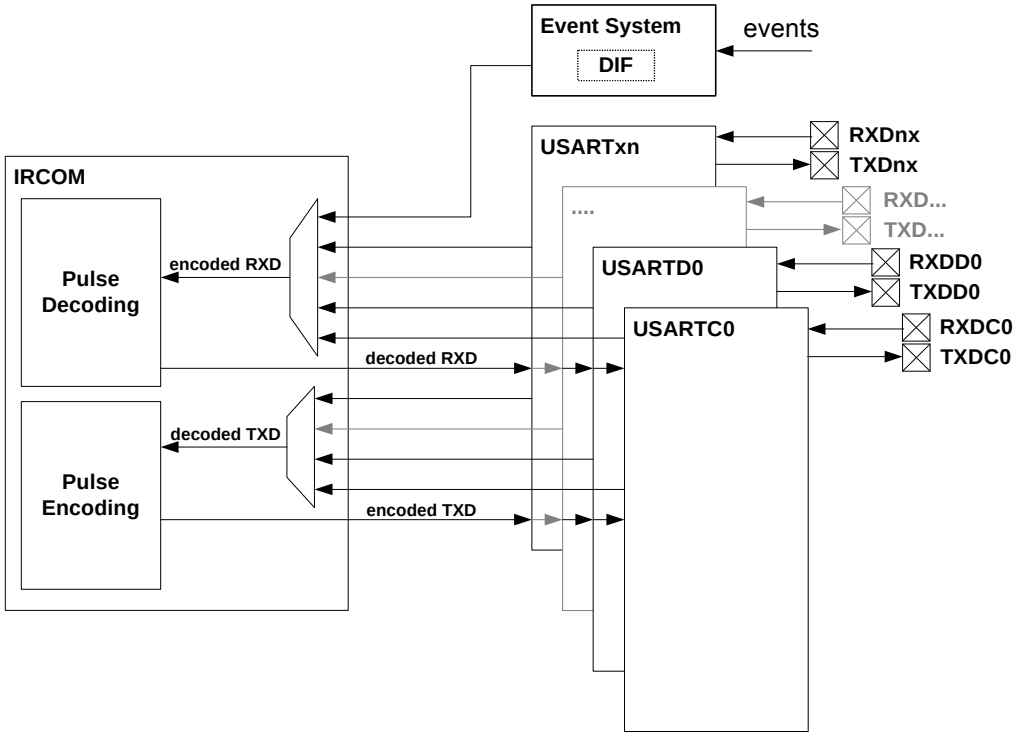
22.1 Features

- **Pulse modulation/demodulation for infrared communication**
- **IrDA 1.4 Compatible for baud rates up to 115.2 kbps**
- **Selectable pulse modulation scheme**
  - **3/16 of 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 contains an Infrared Communication Module (IRCOM) IrDA 1.4 compatible module for baud rates up to 115.2 kbps. This supports three modulation schemes: 3/16 of baud rate period, fixed programmable pulse time based on the Peripheral Clock speed, or pulse modulation disabled. There is one IRCOM available, and this can be connected to any USART to enable infrared pulse coding/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. When this is done signals between the USART and the RX/TX pins are routed through the module as shown in Figure 22-1 on page 256. 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 baud rate period.
- Fixed programmable pulse time based on the Peripheral Clock speed.
- Pulse modulation disabled.

For reception, a minimum high-level pulse width for the pulse to be decoded as a logical 0 can be selected. Shorter pulses will then be discarded and the bit will be decoded to logical 1 as if no pulse where received.

One IRCOM will be available for use with any USART in the device. 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 other I/O pins or sources 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 Digital Input Filter (DIF) on the Event Channels, that can be used for filtering. Refer to Section 6. ”Event System” on page 65” for details on using the Event System.

22.3 Registers Description

22.3.1 TXPLCTRL - IRCOM Transmitter Pulse Length Control Register

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

- **Bits 7:0 - TXPLCTRL[7:0] - Transmitter Pulse Length Control**

The 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 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.

Note: TXPCTRL must be configured before USART transmitter is enabled (TXEN).

22.3.2 RXPLCTRL - IRCOM Receiver Pulse Length Control Register

Bit	7	6	5	4	3	2	1	0	
+0x02	RXPLCTRL[7:0]								RXPLCTRL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 - RXPLCTRL[7:0] - Receiver Pulse Length Control**

The 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 to zero, filtering is disabled. Setting this value between 1 and 255 will enable filtering, where x+1 equal samples is required for the pulse to be accepted.

Note: RXPCTRL must be configured before USART receiver is enabled (RXEN).

22.3.3 CTRL - IRCOM Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	-	EVSEL[3:0]				CTRL
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.

- **Bits 3:0 - EVSEL [3:0]: Event Channel Selection**

These bits select the event channel source for the IRCOM Receiver, according to Table 22-1 on page 259. 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 Select

EVSEL[3:0]	Group Configuration	Event Source
0000		None
0001		(Reserved)
0010		(Reserved)
0011		(Reserved)
0100		(Reserved)
0101		(Reserved)
0110		(Reserved)
0111		(Reserved)
1xxx	CHn	Event System Channelx; x = {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]				258
+0x01	TXPLCTRL	TXPLCTRL[7:0]								258
+0x02	RXPLCTRL	RXPLCTRL[7:0]								258

23. Crypto Engines

23.1 Features

- **Data Encryption Standard (DES) core instruction**
- **Advanced Encryption Standard (AES) crypto module**
- **DES Instruction**
  - **Encryption and Decryption**
  - **DES supported**
  - **Single-cycle DES instruction**
  - **Encryption/Decryption in 16 clock cycles per 8-byte block**
- **AES Crypto Module**
  - **Encryption and Decryption**
  - **Support 128-bit keys**
  - **Support 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 encryption. These are supported through an AES peripheral module and a DES core instruction.

DES is supported by a DES instruction in the AVR XMEGA core. The 8-byte key and 8-byte data blocks must be loaded into the Register file, and then DES must be executed 16 times to encrypt/decrypt the data block.

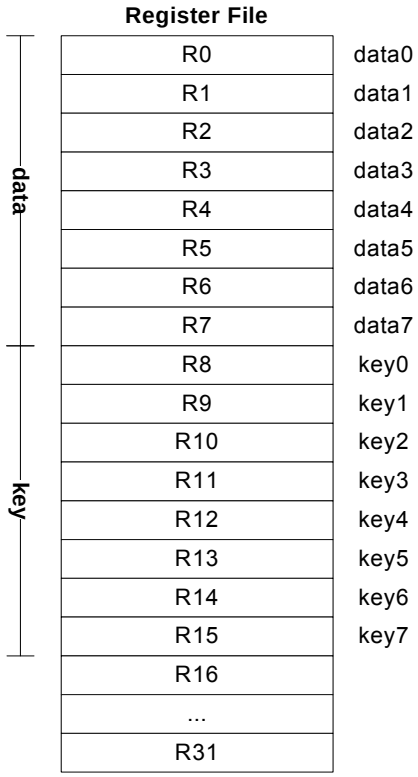
The AES Crypto Module encrypts and decrypts 128-bit data blocks with the use of a 128-bit key. The key and data must be loaded into the module before encryption/decryption is started. It takes 375 peripheral clock cycles before encrypted/decrypted data can be read out.

23.3 DES Instruction

The DES instruction is a single cycle instruction, that needs to be executed 16 times subsequently in order to decrypt or encrypt a 64-bit (8 bytes) data block.

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 LSB of data is placed in LSB of R0 and MSB of data is placed in MSB of R7. The full 64-bit key (including parity bits) is placed in registers R8-R15, with LSB of key in LSB of R8 and MSB of key in MSB of 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 memory in the AES Crypto Module. This is done by writing the AES Key Register and State register sequentially with 16 bytes.

It is selectable from software 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 XOR'ed with the current data in the State memory.

The AES module uses 375 clock cycles before the encrypted/decrypted ciphertext/plaintext is available for readout in the State memory.

The following procedure for setup and use is recommended:

1. Enable AES interrupts (optional)
2. Select the AES direction, 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 the optional interrupt is generated.

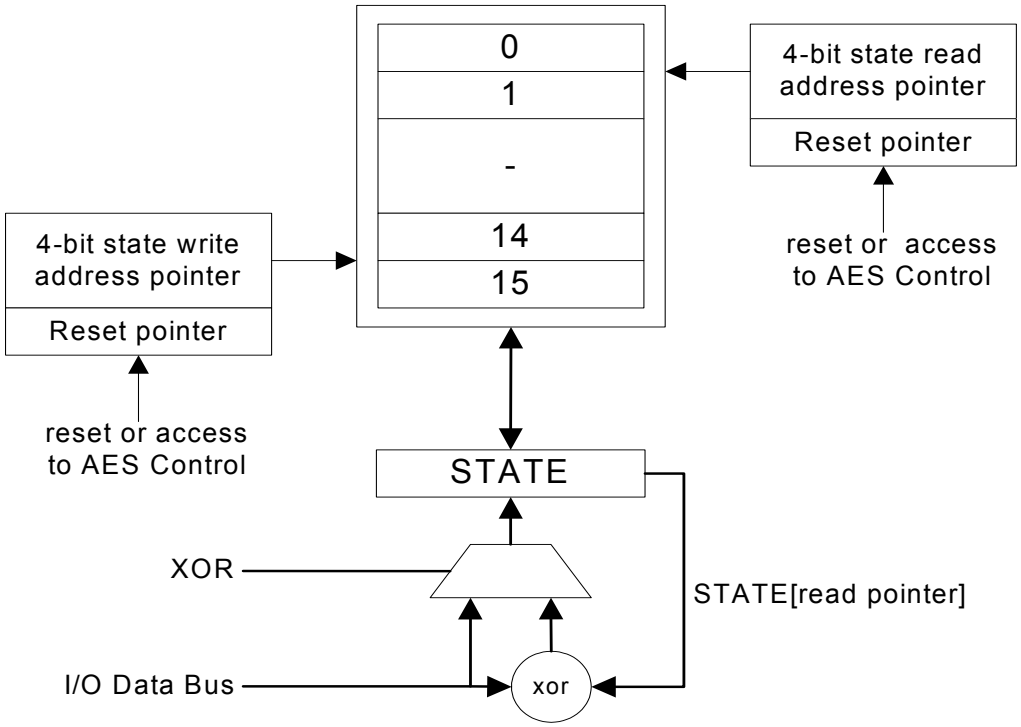
23.4.1 Key and State Memory

The AES Key and State memory are both 16 x 8-bit memories that are accessible through the Key (KEY) and State (STATE) register, 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 are 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 is done more than 16 times) will also set the affected pointer to zero. The address pointers are not accessible from software. Read and write memory pointers are both incremented during write operations in XOR mode.

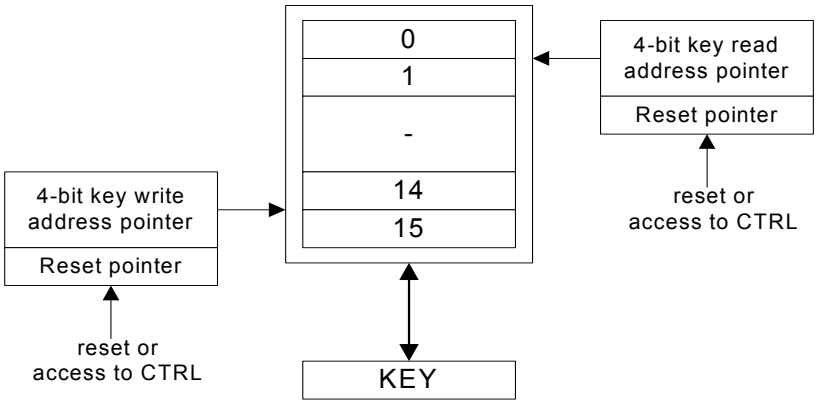
Access to the Key and State registers are only possible when encryption/decryption is not in progress.

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. plain text in the encryption mode, and cipher text 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 by hardware by using the AES Crypto Module and process a dummy data block in encryption mode, using the same Key. After the end of the encryption, reading from the Key memory allows to obtain the last subkey, i.e. get the result of the Key Expansion procedure. Table 23-1 on page 263 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 form the last loaded subkey.

23.4.2 DMA Support

The AES module can trigger a DMA transfer when encryption/decryption procedure is complete. Fore more details on DMA transfer triggers, refer to Section 5.4 ”Transfer Triggers” on page 50.

23.5 Register Description - AES

23.5.1 CTRL - AES Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	START	AUTO	RESET	DECRYPT	-	XOR	-	-	CTRL
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: AES 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 flag in STATUS is set.

- **Bit 6 - AUTO: AES 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 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 not will the encryption/decryption be started with an incorrect Key.

- **Bit 5 - RESET: AES 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: AES 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: AES State XOR Load Enable**

Setting this bit enables XOR data load to the State memory. When this bit is set the data loaded to the State memory is bitwise XOR'ed with current data in the State memory. Writing this bit to zero disables XOR load mode, thus new data written to the State memory overwrites the current data in the State memory.

- **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	STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - ERROR: AES 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 register 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 reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 - SRIF: AES 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								STATE
Read/Write	R/W	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 made after setting the appropriate AES-mode and direction. During encryption/ decryption this register can not be accessed.

23.5.4 KEY - AES Key Register

Bit	7	6	5	4	3	2	1	0	
+0x03	KEY								KEY
Read/Write	R/W	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 made after setting the appropriate AES-mode and direction.

23.5.5 INTCTRL - AES Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
+0x04	-	-	-	-	-	-	INTLVL[1:0]		INTCTRL
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]: AES Interrupt priority and enable**

These bits enable the AES Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. 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	-	-	264
+0x01	STATUS	ERROR	-	-	-	-	-	-	SRIF	265
+0x02	STATE	STATE[7:0]								265
+0x03	KEY	KEY[7:0]								266
+0x04	INTCTRL	-	-	-	-	-	-	INTLVL[1:0]		266
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06										
+0x07										
+0x08										
+0x09										
+0x0A										
+0x0B										
+0x0C										
+0x0D										
+0x0E										
+0x0F										

23.7 AES Interrupt vector

Table 23-2. AES Interrupt vector

Offset	Source	Interrupt Description
0	AES	AES Interrupt vector offset

24. EBI - External Bus Interface

24.1 Features

- Supports SRAM up to
  - 512K Bytes using 2-port EBI
  - 16M Bytes using 3-port EBI
- Supports SDRAM up to
  - 128M bit using 3-port EBI
- Four software configurable Chip Selects
- Software configurable Wait State insertion
- Clocked from the Fast Peripheral Clock at two times the CPU speed

24.2 Overview

The External Bus Interface (EBI) is the interface for connecting external peripheral and memory to access it through the data memory space. When the EBI is enabled, data address space outside the internal SRAM becomes available using dedicated EBI pins.

The EBI can interface external SRAM, SDRAM, and/or peripherals such as LCD displays and other memory mapped devices.

The address space, and the number of pins used for the external memory is selectable from 256 bytes (8-bit) and up to 16M bytes (24-bit). Various multiplexing modes for address and data lines can be selected for optimal use of pins when more or less pins are available for the EBI. The complete memory will be mapped into one linear data address space continuing from the end of the internal SRAM, refer to ”Data Memory” on page 22 for details.

The EBI has four chip selects with separate configuration. Each can be configured for SRAM, SRAM Low Pin Count (LPC) or SDRAM.

The EBI is clocked from the Fast Peripheral clock, running up to two times faster than the CPU and supporting speeds of up to 64 MHz.

For SDRAM both 4-bit and 8-bit SDRAM is supported, and SDRAM configurations such as CAS Latency and Refresh rate is configurable in software.

For more details on SRAM and SDRAM, and how these memory types are organized and work, refer to SRAM and SDRAM specific documentation and data sheets. This section only contains EBI specific details.

24.3 Chip Select

The EBI module has four Chip Select lines (CS0 to CS3) where each can be associated with separate address ranges. The chip selects control which memory or memory mapped external hardware that is accessed when a given memory address is issued on the EBI. Each Chip Select has separate configuration, and can be configured for SRAM or SRAM Low Pin Count (LPC). Chip Select 3 can also be configured for SDRAM.

The data memory address space associated for each chip select is decided by a configurable base address and address size for each Chip Select.

24.3.1 Base Address

The base address is the lowest address in the address space for a chip select. This decides the first location in data memory space where the connected memory hardware can be accessed.

The base address associated with each chip select must be on a 4 Kbyte boundary, i.e to address 0, 4096, 8192 etc.

24.3.2 Address Size

The address size selects how many bits of the address that should be compared when generating a chip select. The address size can be anything from 256 bytes to 16M bytes. If the address space is set to anything larger than 4K bytes, the base address must be on a boundary equal to the address space. With 1M byte address space for a chip select, the base address must be on a 0, 1M byte, 2M byte etc. boundary.

If the EBI is configured so that if the address spaces overlap, the internal memory space have priority, followed by Chip Select 0 (CS0), CS1, CS2 and CS3.

24.3.3 Chip Select as Address Lines

If one or more Chip Select lines are unused, they can in some combinations be used as address lines instead. This can enable larger external memory or external CS generation. Each column in Figure 24-1 on page 269 shows enabled chip select lines (CSn), and the address lines available on unused chip select lines (Ann). Column four shows that all four CS lines are used as address lines when only CS3 is enabled, and this is for SDRAM configuring.

Figure 24-1. Chip Select and address line combinations

CS3	CS3	CS3	A19
CS2	CS2	CS2	A18
CS1	CS1	A17	A17
CS0	A16	A16	A16

24.4 I/O Pin Configuration

When the EBI is enabled it will override the direction and/or value for the I/O pins where the EBI lines are placed. The EBI will override the direction and value for the I/O pins where the EBI data lines are placed. The EBI will only override value, but not direction for the I/O pins where the EBI address and control lines are placed. These I/O pins must be configured to output when the EBI is used. I/O pins for unused EBI address and control lines can be used as normal I/O pins or for other alternate functions on the pins.

For control signals that are active-low, the pin output value should be set to one (high). For control signals that are active-high, pin output value should be set to zero (low). Address lines does not requires specific pin output value configuration. The Chip Select lines should have pull-up resistors to ensure that these are kept high during power-on and reset. If a Chip Select line is active-high, a pull-down should be used instead of a pull-up.

For more details on I/O pin configuration refer to Section 13. on page 129.

24.5 EBI Clock

The EBI is clocked from the Peripheral 2x (Clk_{2PER}) Clock. This clock can run at the CPU Clock frequency, but it can also run at two times the CPU Clock frequency. This can be used to lower the EBI access time. Refer to "System Clock and Clock options" on page 76 for details the Peripheral 2x Clock and how to configure this.

24.6 SRAM Configuration

For use with SRAM the EBI can be configured for various address multiplexing modes by using external address latches, or with no multiplexing. When a limited number of pins on the device are is available for the EBI, Address Latch Enable (ALE) signals are used to control external latches that multiplex address lines from the EBI. The available configurations is shown in Section 24.6.1 on page 270 through Section 24.6.4 on page 271. Table 24-1 on page 270 describe the SRAM interface signals.

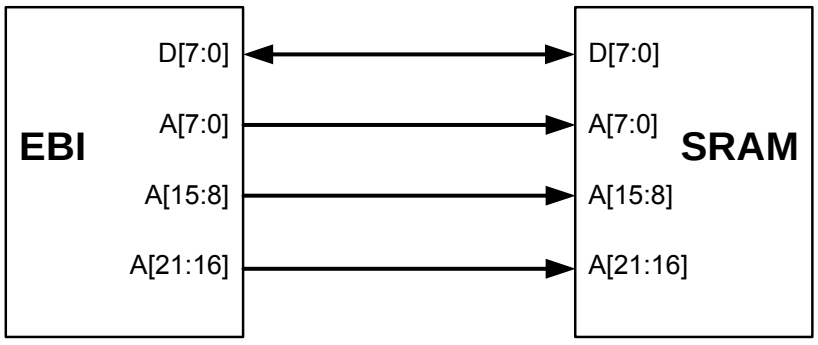
Table 24-1. SRAM Interface signals

Signal	Description
$\overline{CS}$	Chip Select
$\overline{WE}$	Write Enable
$\overline{RE}$	Read Enable
ALE[2:0]	Address Latch Enable
A[23:0]	Address
D[7:0]	Data bus
AD[7:0]	Combined Address and Data

24.6.1 No Multiplexing

When no multiplexing is used, there is a one-to-one connection between the EBI and the SRAM. No external address latches are used.

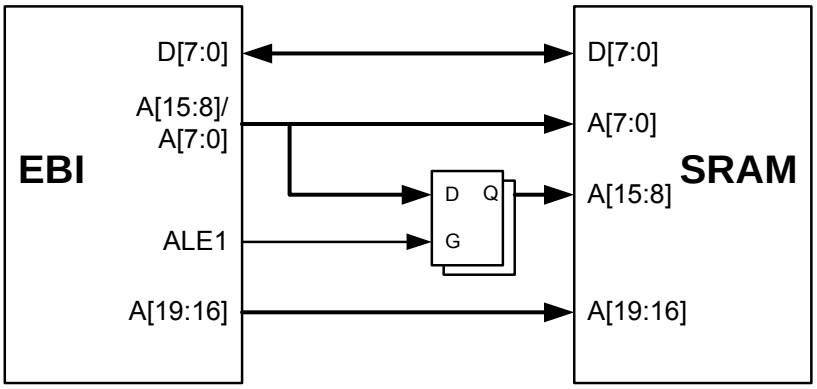
Figure 24-2. Non-multiplexed SRAM connection



24.6.2 Multiplexing address byte 0 and 1

When address byte 0 (A[7:0]) and address byte 1 (A[15:8]) are multiplexed, they are output from the same port, and the ALE1 signal from the device control the address latch.

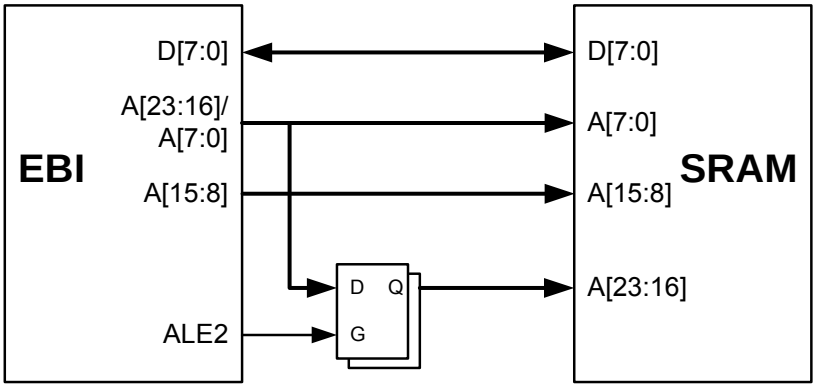
Figure 24-3. Multiplexed SRAM connection using ALE1



24.6.3 Multiplexing address byte 0 and 2

When address byte 0 (A[7:0]) and address byte 2 (A[23:16) are multiplexed, they are output from the same port, and the ALE2 signal from the device control the address latch.

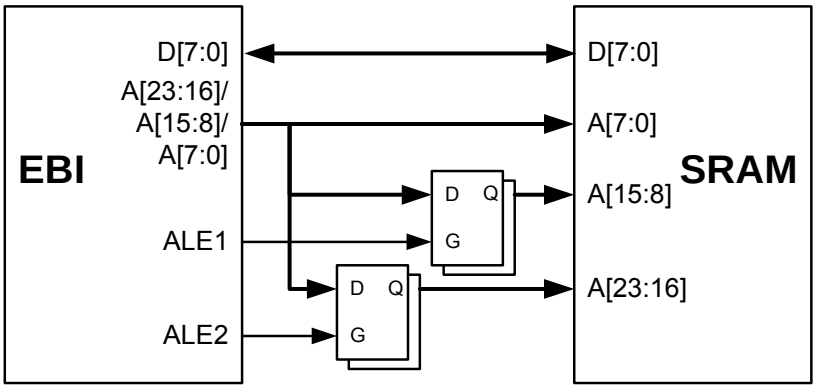
Figure 24-4. Multiplexed SRAM connection using ALE2



24.6.4 Multiplexing address byte 0, 1and 2

When address byte 0 (A[7:0]), address byte 1 (A[15:8]) and address byte 2 (A[23:16] are multiplexed, they are output from the same port, and the ALE1 and ALE2 signal from the device control the external address latches.

Figure 24-5. Multiplexed SRAM connection using ALE1 and ALE2



24.6.5 Address Latch Requirements

The Address Latch timing and parameter requirements are described in ”EBI Timing” on page 277.

24.6.6 Timing

SRAM or external memory devices may have different timing requirements. To meet these varying requirements, each Chip Select can be configured with different wait-states. Timing details is described in ”EBI Timing” on page 277.

24.7 SRAM LPC Configuration

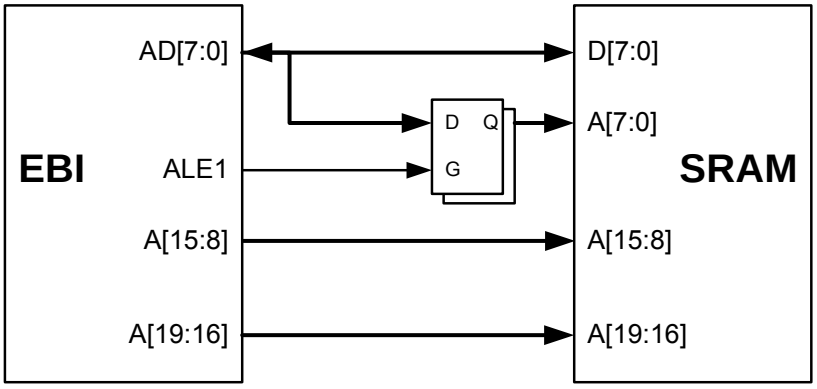
The SRAM Low Pin Count (LPC) configuration enables EBI to be configured for multiplexing modes where the data and address lines are multiplexed. Compared to SRAM configuration, this can further reduce the number of pins required for the EBI. The available configurations is shown in Section 24.7.1 on page 272 through Section 24.7.2 on page 273.

Timing and Address Latch requirements is as for SRAM configuration.

24.7.1 Multiplexing Data with Address Byte 0

When the data byte and address byte 0 (AD[7:0]) are multiplexed, they are output from the same port, and the ALE1 signal from the device controls the address latch.

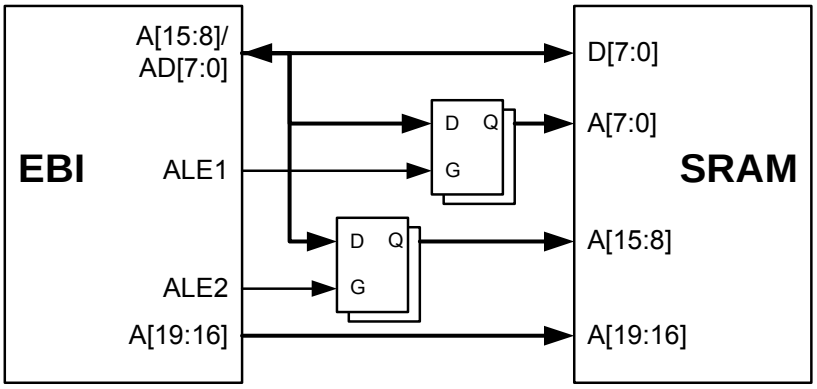
Figure 24-6. Multiplexed SRAM LPC connection using ALE1



24.7.2 Multiplexing Data with Address Byte 0 and 1

When the data byte and address byte 0 (AD[7:0]), and address byte 1 (A[15:8]) are multiplexed, they are output from the same port, and the ALE1 and ALE2 signal from the device control the external address latches.

Figure 24-7. Multiplexed SRAM LPC connection using ALE1 and ALE2



24.8 SDRAM Configuration

Chip Select 3 on the EBI can be configured from SDRAM operation, and the EBI must be configured for 3-Port or 4-Port interface. The SDRAM can be configured for 4-bit or 8-bit data bus, and 4-Port interface must be used for 8-bit data bus. The SDRAM interface signals from the EBI to the SDRAM is listed in Table 24-2 on page 273.

Table 24-2. SDRAM Interface signals

Signal	Description
$\overline{CS}$	Chip Select
$\overline{WE}$	Write Enable
$\overline{RAS}$	Row Address Strobe
$\overline{CAS}$	Column Address Strobe
$\overline{DQM}$	Data Mask Signal/ Output Enable
CKE	Clock Enable
CLK	Clock
BA[1:0]	Bank Address
A[12:0]	Address bus
A[10]	Precharge
D[7:0]	Data bus

24.8.1 Supported Commands

The SDRAM commands that are supported by the EBI is listed in Table 24-3 on page 274.

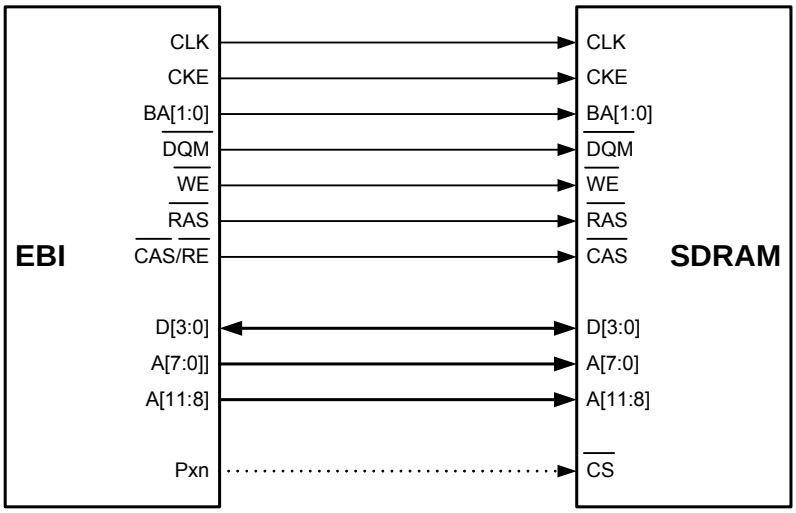
Table 24-3. Supported SDRAM commands

Command	Description
NOP	No Operation
ACTIVE	Activate the selected bank and select the row.
READ	Input the starting column address and begin the burst read operation.
WRITE	Input the starting column address and begin the burst write operation.
PRECHARGE	Deactivate the open row of selected bank or all banks
AUTO REFRESH	Refresh one row of each bank
LOAD MODE	
SELF REFRESH	Activate self refresh mode

24.8.2 3-Port EBI Configuration

When 3 EBI ports are available, SDRAM can be connected with 3-Port EBI configuration. When this is done only 4-bit data bus is available, and any chip select must be controlled from software using a general purpose I/O pin (Pxn).

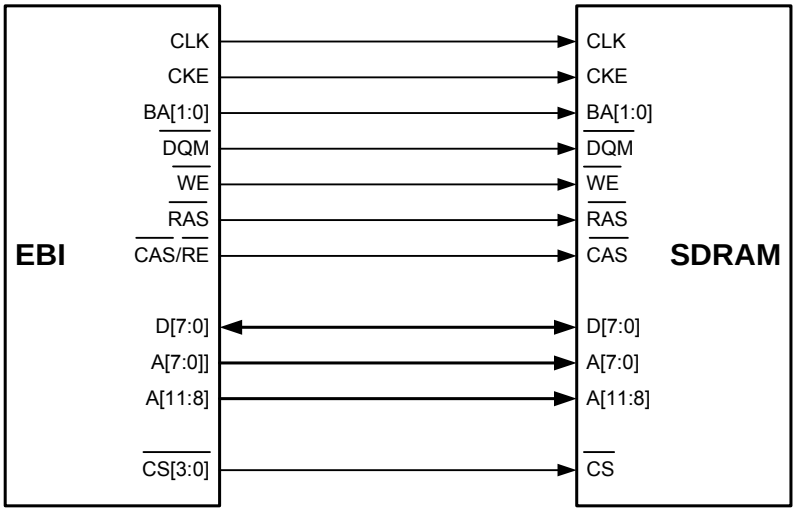
Figure 24-8. 3-Port SDRAM configuration



24.8.3 4-Port EBI Configuration

When 4 EBI ports are available, SDRAM can be connected with 3-Port or 4-Port EBI configuration. When 4-Port configuration is used, 8-bit data bus is available and all four chip selects will be available.

Figure 24-9. 4-Port SDRAM configuration



24.8.4 Timing

The Clock Enable (CKE) signal is required for SDRAM when the EBI is clocked at 2x the CPU clock speed.

24.8.5 Initialization

Configuring Chip Select 3 to SDRAM will enable the initialization of the SDRAM. The “Load Mode Register” command is automatically issued at the end of the initialization. For the correct information to be loaded to the SDRAM, one must do one of the following:

1. Configure SDRAM control registers before enabling Chip Select 3 to SDRAM.
2. Issue a “Load Mode Register” command and perform a dummy-access after SDRAM is initialized.

The SDRAM initialization is non-interruptible by other EBI accesses.

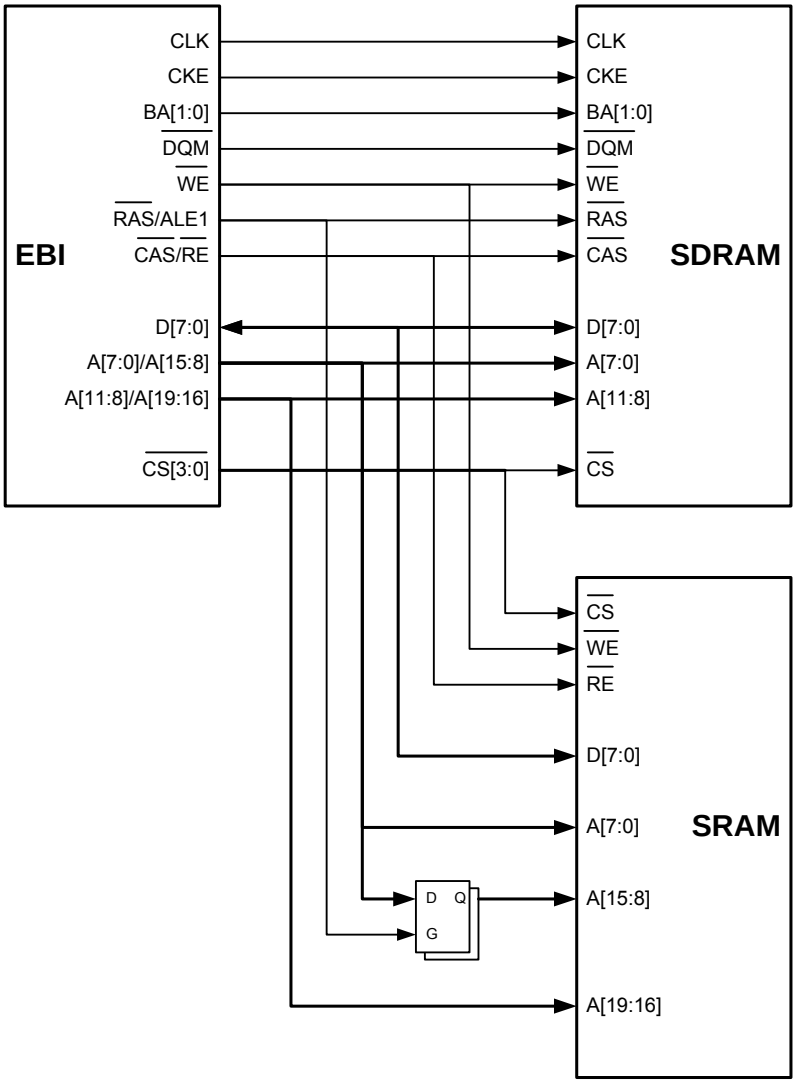
24.8.6 Refresh

The EBI will automatically handle the refresh of the SDRAM as long as the refresh period is configured. Refresh will be done as soon as available after the refresh counter reaches the period. The EBI can collect up to 4 refresh commands in case the interface is busy on another chip select or in the middle of a read/write at a time a refresh should have been performed.

24.9 Combined SRAM & SDRAM Configuration

Combined SRAM and SDRAM configuration enables the EBI to have both SDRAM and SRAM connected at the same time. This only available for devices with 4 port EBI interface. Figure 24-10 on page 276 shows the configuration with all interface signals.

Figure 24-10. Combined SRAM and SDRAM connection



24.10 EBI Timing

(See Section 33. "Appendix A: EBI Timing Diagrams" on page 392)

24.10.1 SRAM

Table 24-4. EBI SRAM Characteristics[1]

Symbol	Parameter	Min	Typical	Max	Unit
$t_{CLKPER2}$	Clock period	2 x Clk_{SYS}			
t_{ALH}	Address Hold after ALE Low		$t_{CLKPER2}$		ns
t_{ALS}	Address Setup before ALE low (read and write)		$t_{CLKPER2}$		ns
t_{ALW}	ALE Width		$t_{CLKPER2}$		ns
t_{ARH}	Address Hold after RE high		$t_{CLKPER2}$		ns
t_{AWH}	Address Hold after WE high		$t_{CLKPER2}$		ns
t_{ARS}	Address Setup before RE high		$t_{CLKPER2}$		ns
t_{AWS}	Address Setup before WE low		$t_{CLKPER2}$		ns
t_{DRH}	Data Hold after RE high	0			ns
t_{DRS}	Data Setup to RE high	TBD			ns
t_{DWH}	Data Hold after WE high		$t_{CLKPER2}$		ns
t_{DWS}	Data Setup to WE high		$t_{CLKPER2}$		ns
t_{RW}	Read Enable Pulse width		$t_{CLKPER2}$ x SRWS		ns
t_{WW}	Write Enable Pulse width		$t_{CLKPER2}$ x SRWS		ns
t_{CH}	Chip Select hold after RE/WE high		$t_{CLKPER2}$		ns
t_{CRS}	Chip Select setup to RE low or ALE high		0		ns
t_{CWS}	Chip Select setup to WE low		$t_{CLKPER2}$		ns

Note: 1. For characterization of Clk_{SYS} and t_{DRS} refer to device data sheet

24.10.2 SDRAM

Table 24-5. EBI SDRAM Characteristics[1]

Symbol	Parameter	Min	Typical	Max	Unit
$t_{CLKPER2}$	Clock period	2 x Clk_{SYS}			ns
t_{AH}	Address Hold time		0.5 x $t_{CLKPER2}$		ns
t_{AS}	Address Setup time		0.5 x $t_{CLKPER2}$		ns
t_{CH}	Clock High-level witdh		0.5 x $t_{CLKPER2}$		ns
t_{CL}	Clock low-level witdh		0.5 x $t_{CLKPER2}$		ns
t_{CKH}	CKE Hold time		0.5 x $t_{CLKPER2}$		ns
t_{CKS}	CKE Setup time		0.5 x $t_{CLKPER2}$		ns
t_{CMH}	CS, RAS, CAS, WE, DQM hold time		0.5 x $t_{CLKPER2}$		ns
t_{CMS}	CS, RAS, CAS, WE, DQM setup time		0.5 x $t_{CLKPER2}$		ns
t_{DRH}	Data in Hold		0		ns
t_{DRS}	Data in Setup	TBD	0.5 x $t_{CLKPER2}$		ns
t_{DWH}	Data out Hold		0.5 x $t_{CLKPER2}$		ns
t_{DWS}	Data out Setup		0.5 x $t_{CLKPER2}$		ns

Note: 1. For characterization of Clk_{SYS} and t_{DRS} refer to device data sheet

24.11 Register Description - EBI

24.11.1 CTRL - EBI Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	SDDATAW[1:0]		LPCMODE[1:0]		SRMODE[1:0]		IFMODE[1:0]		CTRL
Read/Write	R/W	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 - SDDATAW[1:0]: EBI SDRAM Data Width Setting**

These bits select the EBI SDRAM data width configuration according to Table 24-6 on page 279.

Table 24-6. SDRAM Mode

SDDATAW[1:0]	Group Configuration	Description
00	4BIT	4 bit data bus
01	8BIT(1)	8 bit data bus
10	-	Reserved
11	-	Reserved

Note: 1. 8-bit data bus only available for 4-port EBI interface

- **Bit 5:4 - LPCMODE[1:0]: EBI SRAM Low Pin-count Mode**

These bits select the EBI SRAM LPC configuration according to Table 24-7 on page 279

Table 24-7. SRAM LPC Mode

LPCMODE[1:0]	Group Configuration	ALE	Description
00	ALE1	ALE1	Data multiplexed with Address byte 0
01	-	-	Reserved
10	ALE12	ALE1 & 2	Data multiplexed with Address byte 0 and 1
11	-	-	Reserved

- **Bit 3:2 - SRMODE[1:0]: SRAM Mode**

These bits selects the EBI SRAM configuration according to Table 24-8 on page 279.

Table 24-8. SRAM Mode

SRMODE[1:0]	Group Configuration	ALE	Description
00	ALE1	ALE1	Address byte 0 and 1 multiplexed
01	ALE2(1)	ALE2	Address byte 0 and 2 multiplexed
10	ALE12(1)	ALE1 & 2	Address byte 0, 1 and 2 multiplexed
11	NOALE	No ALE	No address multiplexing

Note: 1. ALE2 and NOALE only available with 4-port EBI interface

- **Bit 1:0 - IFMODE[1:0]: EBI Interface Mode**

These bits select EBI interface mode and the number of ports that should be enabled and overridden for EBI, according to Table 24-9 on page 280.

Table 24-9. EBI Mode

IFMODE[1:0]	Group Configuration	Description
00	DISABLED	EBI Disabled
01	3PORT	EBI enabled with 3-port interface
10	4PORT	EBI enabled with 4-port interface
11	2PORT	EBI enabled with 2-port interface

24.11.2 SDRAMCTRLA - SDRAM Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	-	SDCAS	SDROW	SDCOL[1:0]		SDRAMCTRLA
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 reserved and will always be read as zero.

- **Bit 3 - SDCAS: SDRAM CAS Latency**

This bit sets the CAS latency as a number of Peripheral 2x Clock cycles. By default this bit is zero and the CAS latency is two Peripheral 2x Clock cycles. When this bit is set to one the CAS latency is three Peripheral 2x Clock cycles.

- **Bit 2 - SDROW: SDRAM Row Bits**

This bit sets the number of row bit used for the connected SDRAM. By default this bit is zero, and the row bit setting is set to 11 Row Bits. When this bit is set to one the row bit setting is set to 12 Row Bits.

- **Bit 1:0 - SDCOL[1:0]: SDRAM Column Bits**

These bits select the number of column bits that are used for the connected SDRAM according to table.Table 24-10 on page 280.

Table 24-10. SDRAM Column Bits

SDCOL[1:0]	Group Configuration	Description
00	8BIT	8 Column Bits
01	9BIT	9 Column Bits
10	10BIT	10 Column Bits
11	11BIT	11 Column Bits

24.11.3 REFRESH - SDRAM Refresh Period Register

Bit	7	6	5	4	3	2	1	0	
+0x04	REFRESH[7:0]								REFRESHL
+0x05	-	-	-	-	-	-	REFRESH[9:8]		REFRESHH
	15	14	13	12	11	10	9	8	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:10 - Reserved**

These bits are reserved and will always be read as zero.

- **Bit 9:0 - REFRESH[9:0]: SDRAM Refresh Period**

This register sets the refresh period as a number of Peripheral 2x clock (CLK_{PER}) cycles. If the EBI is busy with another external memory access at time of refresh, up to 4 refresh will be remembered and given at the first available time.

24.11.4 INITDLY - SDRAM Initialization Delay Register

Bit	7	6	5	4	3	2	1	0	
+0x06	INITDLY[7:0]								INITDLYL
+0x07	-	-	INITDLY[9:8]						INITDLYH
	15	14	13	12	11	10	9	8	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:14 - Reserved**

These bits are reserved and will always be read as zero.

- **Bit 13:0 - INITDLY[13:0]: SDRAM Initialization Delay**

This register is used to delay the initialisation sequence after the controller is enabled until all voltages are stabilized and the SDRAM clock has been running long enough to take the SDRAM chip through its initialisation sequence. The initialisation sequence includes pre-charge all banks to their idle state issuing an auto-refresh cycle and then loading the mode register. The setting in this register is as a number of Peripheral 2x clock cycles.

24.11.5 SDRAMCTRLB - SDRAM Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x08	MRDLY[1:0]		ROWCYCDLY[2:0]			RPDLY[2:0]			SDRAMCTRLB
Read/Write	R/W	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 - MRDLY[1:0]: SDRAM Mode Register Delay**

These bits select the delay between Mode Register command and an Activate command in number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-11 on page 282.

Table 24-11. SDRAM Column Bits settings

MRDLY[1:0]	Group Configuration	Description
00	0CLK	0 CLK_{PER2} cycles delay
01	1CLK	1 CLK_{PER2} cycles delay
10	2CLK	2 CLK_{PER2} cycles delay
11	3CLK	3 CLK_{PER2} cycles delay

- **Bit 5:3 - ROWCYCDLY[2:0]: SDRAM Row Cycle Delay**

These bits select the delay between a Refresh and an Activate command in number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-12 on page 282..

Table 24-12. SDRAM Row Cycle Delay settings

ROWCYDLY[2:0]	Group Configuration	Description
000	0CLK	0 CLK_{PER2} cycles delay
001	1CLK	1 CLK_{PER2} cycles delay
010	2CLK	2 CLK_{PER2} cycles delay
011	3CLK	3 CLK_{PER2} cycles delay
100	4CLK	4 CLK_{PER2} cycles delay
101	5CLK	5 CLK_{PER2} cycles delay
110	6CLK	6 CLK_{PER2} cycles delay
111	7CLK	7 CLK_{PER2} cycles delay

- **Bit 2:0 - RPDLY[2:0]: SDRAM Row to Pre-charge Delay**

RPDLY defines the delay between a Pre-charge command and another command in number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-13 on page 282.

Table 24-13. SDRAM Row Cycle Delay settings

RPDLY[2:0]	Group Configuration	Description
000	0CLK	0 CLK_{PER2} cycles delay
001	1CLK	1 CLK_{PER2} cycles delay
010	2CLK	2 CLK_{PER2} cycles delay
011	3CLK	3 CLK_{PER2} cycles delay
100	4CLK	4 CLK_{PER2} cycles delay
101	5CLK	5 CLK_{PER2} cycles delay
110	6CLK	6 CLK_{PER2} cycles delay
111	7CLK	7 CLK_{PER2} cycles delay

24.11.6 SDRAMCTRLC - SDRAM Control Register C

Bit	7	6	5	4	3	2	1	0	
+0x09	WRDLY[1:0]		ESRDLY[1:0]			ROWCOLDLY[1:0]			SDRAMCTRLC
Read/Write	R/W	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 - WRDLY[1:0]: SDRAM Write Recovery Delay**

These bits select the Write Recovery time in number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-11 on page 282.

Table 24-14. SDRAM Write Recovery Delay settings

WRDLY[1:0]	Group Configuration	Description
00	0CLK	0 CLK_{PER2} cycles delay
01	1CLK	1 CLK_{PER2} cycles delay
10	2CLK	2 CLK_{PER2} cycles delay
11	3CLK	3 CLK_{PER2} cycles delay

- **Bit 5:3 - ESRDLY[2:0]: SDRAM Exit Self Refresh to Active Delay**

This field defines the delay between CKE set high and an Activate command in a number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-15 on page 283.

Table 24-15. SDRAM Exit Self Refresh Delay settings

ESRDLY[2:0]	Group Configuration	Description
000	0CLK	0 CLK_{PER2} cycles delay
001	1CLK	1 CLK_{PER2} cycles delay
010	2CLK	2 CLK_{PER2} cycles delay
011	3CLK	3 CLK_{PER2} cycles delay
100	4CLK	4 CLK_{PER2} cycles delay
101	5CLK	5 CLK_{PER2} cycles delay
110	6CLK	6 CLK_{PER2} cycles delay
111	7CLK	7 CLK_{PER2} cycles delay

- **Bit 2:0 - ROWCOLDLY[2:0]: SDRAM Row to Column Delay**

This field defines the delay between an Activate command and a Read/Write command as a number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-16 on page 283.

Table 24-16. SDRAM Row Column Delay settings

ROWCOLDLY[2:0]	Group Configuration	Description
000	0CLK	0 CLK_{PER2} cycles delay
001	1CLK	1 CLK_{PER2} cycles delay
010	2CLK	2 CLK_{PER2} cycles delay
011	3CLK	3 CLK_{PER2} cycles delay

Table 24-16. SDRAM Row Column Delay settings (Continued)

ROWCOLDLY[2:0]	Group Configuration	Description
100	4CLK	4 CLK_{PER2} cycles delay
101	5CLK	5 CLK_{PER2} cycles delay
110	6CLK	6 CLK_{PER2} cycles delay
111	7CLK	7 CLK_{PER2} cycles delay

24.12 Register Description - EBI Chip Select

24.12.1 CTRLA - Chip Select Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	-	ASIZE[4:0]					MODE[1:0]		CTRLA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - Reserved**

This bit is reserved and will always be read as zero.

- **Bit 6:2 - ASIZE[4:0]: Chip Select Address Size**

These bits select the address size for the Chip Select. This is the size of the block above the base address.

Table 24-17. Address space encoding

ASIZE[5:0]	Group Configuration	Address Size	Address Lines Compared
00000	256B	256 Bytes	ADDR[23:8]
00001	512B	512 Bytes	ADDR[23:9]
00010	1K	1K Bytes	ADDR[23:10]
00011	2K	2K Bytes	ADDR[23:11]
00100	4K	4K Bytes	ADDR[23:12]
00101	8K	8K Bytes	ADDR[23:13]
00110	16K	16K Bytes	ADDR[23:14
00111	32K	32K Bytes	ADDR[23:15]
01000	64K	64K Bytes	ADDR[23:16]
01001	128K	128K Bytes	ADDR[23:17]
01010	256K	256K Bytes	ADDR[23:18]
01011	512K	512K Bytes	ADDR[23:19]
01100	1M	1M Bytes	ADDR[23:20]
01101	2M	2M Bytes	ADDR[23:21]
01110	4M	4M Bytes	ADDR[23:22]

Table 24-17. Address space encoding (Continued)

ASIZE[5:0]	Group Configuration	Address Size	Address Lines Compared
01111	8M	8M Bytes	ADDR[23]
10000	16M	16M (1)	-
Other		-	Reserved

1. Entire available data space used.

- **Bit 1:0 - MODE[1:0]: Chip Select Mode**

These bits select the Chip Select Mode and decide what type of interface is used for the external memory or peripheral according to Table 24-18 on page 285.

Table 24-18. Chip Select Mode selection

MODE[1:0]	Group Configuration	Description
00	DISABLE	Chip Select Disabled
01	SRAM	Enable Chip Select for SRAM
10	LPC	Enable Chip Select for SRAM LPC
11	SDRAM	Enable Chip Select for SDRAM (1)

1. SDRAM can only be selected for CS3

24.12.2 CTRLB (SRAM) - Chip Select Control Register B

This configuration options in this register depend on the Chip Select Mode configuration. The register description below is valid when the Chip Select Mode is configured for SRAM or SRAM LPC.

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	-	-	SRWS[2:0]			CTRLB
Read/Write	R/W	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 7:3 - Reserved**

These bits are reserved and will always be read as zero.

- **Bit 2:0 - SRWS[2:0]: SRAM Wait State**

These bits select the number of wait states for SRAM and SRAM LPC access as a number of Peripheral 2x clock (CLK_{PER2}) cycles, according to Table 24-19 on page 285.

Table 24-19. Wait State selection

SRWS[2:0]	Group Configuration	Description
000	0CLK	0 CLK_{PER2} cycles wait state
001	1CLK	1 CLK_{PER2} cycles wait state
010	2CLK	2 CLK_{PER2} cycles wait state
011	3CLK	3 CLK_{PER2} cycles wait state
100	4CLK	4 CLK_{PER2} cycles wait state

Table 24-19. Wait State selection (Continued)

SRWS[2:0]	Group Configuration	Description
101	5CLK	5 CLK_{PER2} cycles wait state
110	6CLK	6 CLK_{PER2} cycles wait state
111	7CLK	7 CLK_{PER2} cycles wait state

24.12.3 CTRLB (SDRAM) - Chip Select Control Register B

This configuration options in this register depend on the Chip Select Mode configuration. The register description below is valid when the Chip Select Mode is configured for SDRAM.

Bit	7	6	5	4	3	2	1	0	
+0x01	SDINITDONE	-	-	-	-	SDREN	SDMODE[1:0]		CTRLB
Read/Write	R/W	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 - SDINITDONE: SDRAM Initialization Complete**

This flag is set at the end of the SDRAM initialization sequence. The flag will remain set as long as the EBI is enabled and the Chip Select is configured for SDRAM.

- **Bit 6:3 - Reserved**

These bits are reserved and will always be read as zero.

- **Bit 2 - SDSREN: SDRAM Self-refresh Enable**

When this bit is written to one the EBI controller will send a Self-refresh command to the SDRAM. For leaving the self refresh mode, the bit must be written to zero.

- **Bit 1:0 SDMODE[1:0]: SDRAM Mode**

These bits select mode when accessing the SDRAM according to Table 24-20 on page 286.

Table 24-20. SDRAM Mode

SRMODE[1:0]	Group Configuration	Description
00	NORMAL	Normal Mode. Access to the SDRAM is decoded normally.
01	LOAD	Load Mode. The EBI issues a “Load Mode Register” command when the SDRAM is accessed.
10	-	Reserved
11	-	Reserved

24.12.4 BASEADDR - Chip Select Base Address Register

Bit	7	6	5	4	3	2	1	0	
+0x02	BASEADDR[15:12]				-	-	-	-	BASEADDRL
+0x03	BASEADDR[23:16]								BASEADDRH
	15	14	13	12	11	10	9	8	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:4 - BASEADDR[23:12]: Chip Select Base Address**

The base address is the lowest address in the address space enabled by a chip select. Together with the Chip Select Address Size (ASIZE) setting in “CTRLA - Chip Select Control Register A” this gives the address space for the Chip Select.

- **Bit 3:0 - Reserved**

These bits are reserved and will always be read as zero.

24.13 Register Summary - EBI

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	SDDATAW[1:0]		LPCMODE[1:0]		SRMODE[1:0]		IFMODE[1:0]		279
+0x01	SDRAMCTRLA	-	-	-	-	SDCAS	SDROW	SDCOL[1:0]		280
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	Reserved	-	-	-	-	-	-	-	-	
+0x04	REFRESHL	SDRAM Refresh Period Low Byte								281
+0x05	REFRESHH	-	-	-	-	-	-	SDRAM Refresh Period High		281
+0x06	INITDLYL	SDRAM Initialization Time Low Byte								281
+0x07	INITDLYH	-	-	SDRAM Initialization Time High Byte						281
+0x08	SDRAMCTRLB	MRDLY[1:0]		ROWCYCDLY[[2:0]			RPDLY[2:0]			281
+0x09	SDRAMCTRLC	RWDLY[1:0]		ESRDLY[2:0]			ROWCOLDLY[2:0]			283
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	Reserved	-	-	-	-	-	-	-	-	
+0x0D	Reserved	-	-	-	-	-	-	-	-	
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	
+0x10	CS0	Chip Select 0 Offset Address								
+0x14	CS1	Chip Select 1 Offset Address								
+0x18	CS2	Chip Select 2 Offset Address								
+0x1C	CS3	Chip Select 3 Offset Address								

24.14 Register Summary - EBI Chip Select

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA		-	ASIZE[4:0]					MODE[1:0]		284
+0x01	CTRLB	(SRAM)	-	-	-	-	-	SRWS[2:0]			285
		(SDRAM)	SDINITDONE	-	-	-	-	SDSREN	SDMODE[1:0]		286
+0x02	BASEADDRL		Chip Select Base Address Low Byte				-	-	-	-	286
+0x03	BASEADDRH		Chip Select Base Address High Byte								286

25. ADC - Analog to Digital Converter

25.1 Features

- 12-bit resolution
- Up to 2 Msps sample and conversion rate
- Single-ended or Differential measurements
- Signed and Unsigned mode
- 4 result registers with individual input control
- 8 - 16 single-ended inputs
- 8x4 differential inputs without gain
- 8x4 differential input with gain
- 4 internal inputs
  - Temperature Sensor
  - DAC Output
  - VCC voltage divided by 10
  - Bandgap voltage
- 1x, 2x, 4x, 8x, 16x, 32x or 64x software selectable gain
- 4 inputs can be sampled within 1.5 µs
- 8-, or 12-bit selectable resolution
- Minimum single result propagation delay of 2.5 µs (8-bit resolution)
- Minimum single result propagation delay of 3.5 µs (12-bit resolution)
- Built-in reference
- Optional external reference
- Optional event triggered conversion for accurate timing
- Optional DMA transfer of conversion results
- Optional interrupt/event on compare result

25.2 Overview

The ADC converts analog voltages to digital values. The ADC has 12-bit resolution and is capable of converting up to 2 million samples per second. 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.

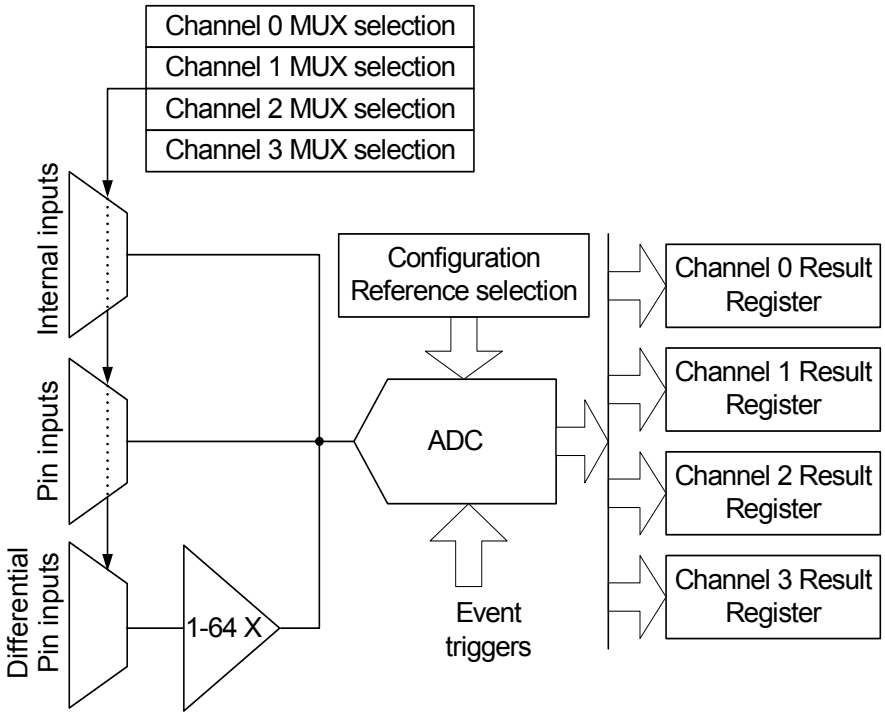
This is a pipeline ADC. A pipeline ADC consists of several consecutive stages, where each stage convert one part of the result. The pipeline design enables high sample rate at low clock speeds, and remove dependencies between sample speed and propagation delay. This also means that a new analog voltage can be sampled and a new ADC measurement started while other ADC measurements are ongoing.

ADC measurements can either be started by application software or an incoming event from another peripheral in the device. Four different result registers with individual input selection (MUX selection) are provided to make it easier for the application to keep track of the data. Each result register and MUX selection pair is referred to as an ADC Channel. It is possible to use DMA to move ADC results directly to memory or peripherals when conversions are done.

Both internal and external analog reference voltages can be used. A very accurate internal 1.00V reference is available.

An integrated temperature sensor is available and the output from this can be measured with the ADC. The output from the DAC, VCC/10 and the Bandgap voltage can also be measured by the ADC.

Figure 25-1. ADC overview



25.3 Input sources

The input sources for the ADC are the analog 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 analog 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 XMEGA devices with only one ADC but with analog input pins on both PORTA and PORTB, both PORTA and PORTB analog pins can then be used as input.

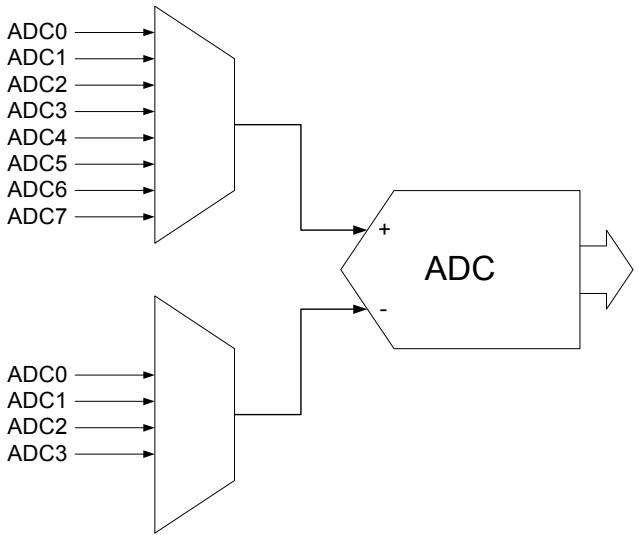
The MUX Control registers select which input that is converted and the type of measurements in one operation. The four types of measurements and their corresponding MUXes are shown in Figure 25-2 on page 291 to Figure 25-6 on page 293.

The ADC itself is always differential, also for single ended inputs where the negative input for the ADC will be connected to an fixed internal value.

25.3.1 Differential input

When differential input is selected all analog input pins can be selected as positive input, and analog input pins 0 to 3 can be selected as negative input. The ADC must be set in signed mode when differential input is used.

Figure 25-2. Differential measurement without gain

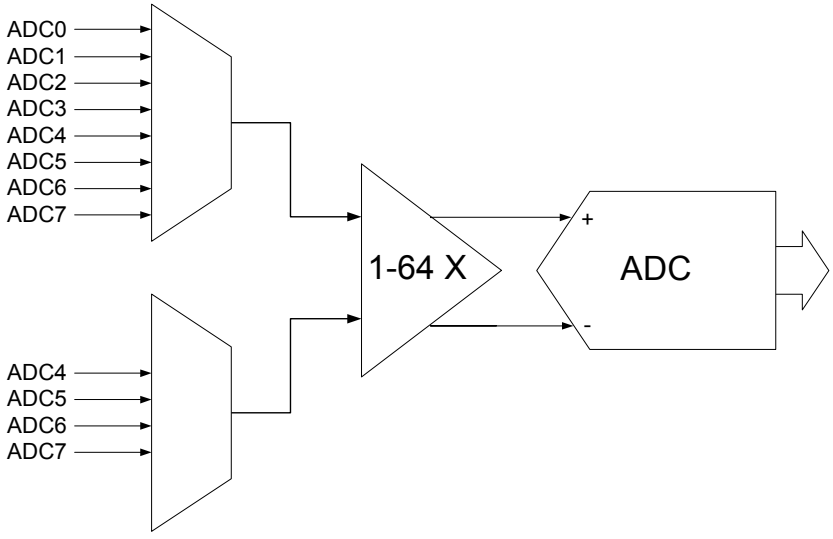


25.3.2 Differential input with gain

When differential input with gain is selected all analog input pins can be selected as positive input, and analog input pins 4 to 7 can be selected as negative input. When the gain stage is used, the differential analog input is first sampled and amplified by the gain stage before the result is fed into the ADC. The ADC must be set in signed mode when differential input with gain is used.

The gain is selectable to 1x, 2x, 4x, 8x, 16x, 32x and 64x gain.

Figure 25-3. Differential measurement with gain

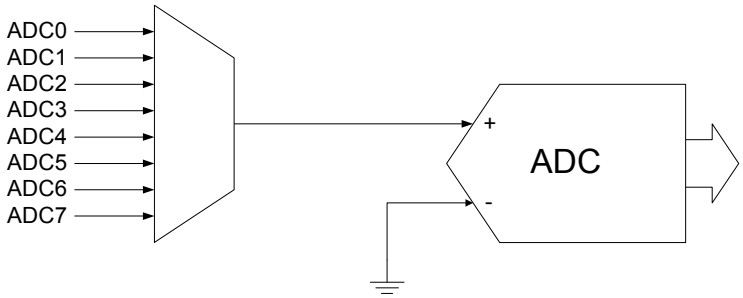


25.3.3 Single ended input

For single ended measurements all analog 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 25-4. Single-ended measurement in signed mode

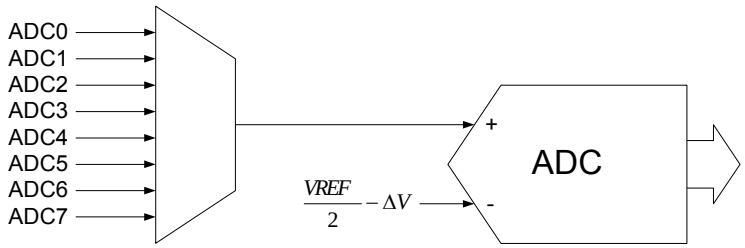


In unsigned mode the negative input is connected to half of the voltage reference (VREF) voltage minus a fixed offset. The nominal value for the offset is:

$$\Delta V = VREF \times 0.05$$

Since the ADC is differential, unsigned mode is achieved by dividing the reference by two internally, resulting in an input range from VREF to zero for the positive single ended input. The offset enables the ADC to measure zero cross detection in unsigned mode, and to calibrate any positive offset where the internal ground in the device is higher than the external ground. See Figure 25-11 on page 296 for details.

Figure 25-5. Single ended measurement in unsigned mode



25.3.4 Internal inputs

Four internal analog signals can be selected as input and measured by the ADC.

- Temperature sensor
- Bandgap voltage
- V_{CC} scaled
- DAC output

The voltage output from an internal temperature reference can be measured with the ADC and the voltage output will give an ADC result representing the current temperature in the microcontroller. During production test, the ADC measures a fixed temperature using the internal

temperature sensor, and the value is store in the production calibration row and can be used for temperature sensor calibration.

The bandgap voltage is an accurate voltage reference inside the microcontroller that is the source for other internal voltage references.

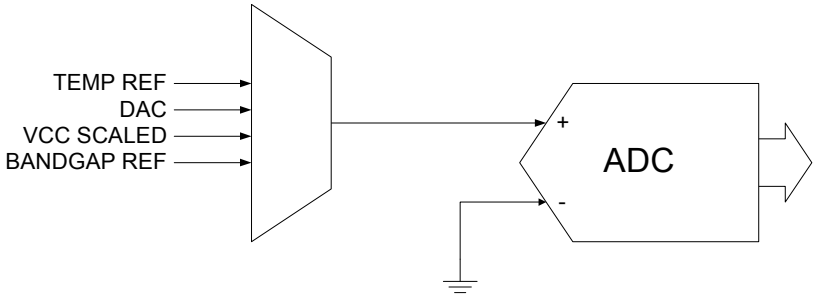
V_{CC} can be measured directly by scaling it down and dividing it by 10 before the ADC input. Thus, V_{CC} of 1.8 V will be measured as 0.18 V and V_{CC} of 3.6 V will be measured as 0.36 V.

The output from the DAC module can also be measured by the ADC. It is the output directly from the DAC, and not the Sample and Hold (S/H) outputs, that is available for the ADC.

Some of the internal signals need to be turned on specifically before they can be used measured. Refer to the manual for these modules for details of how to turn them on. The sample rate for the internal signals is lower than the maximum speed for the ADC, refer to the ADC characteristics in the device data sheets for details on sample rate of the internal signals.

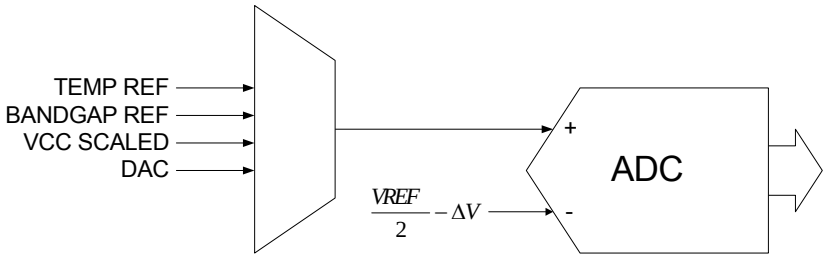
When measuring the internal signals, the negative input is connected to internal ground in signed mode.

Figure 25-6. Internal measurements in signed mode



In unsigned mode the negative input is connected to a fixed value which is half of the voltage reference (VREF) minus a fixed offset as it is for single ended unsigned input. Refer to Figure 25-11 on page 296 for details.

Figure 25-7. Internal measurements in unsigned mode



25.4 ADC Channels

To facilitate the use of the ADC, the design implements four separate MUX control registers with corresponding result registers. The MUX/result register pair is called an ADC channel, and this is shown in Figure 25-1 on page 290. Each ADC channel can be set up individually to measure different input sources, use different start conversion triggers, and they use different events and interrupts. The result is stored in different result registers.

As an example of the ADC channel scheme, one of the MUX/result register pairs can be setup to do single-ended measurements triggered by event signal input, the second MUX/result pair can measure a differential input on another event signal input, and the two last MUX/result pairs can measure two other input sources started by the application software.

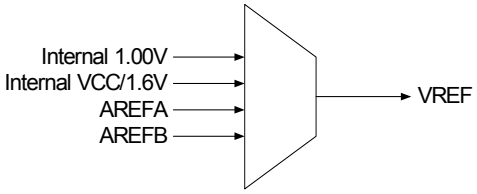
All the ADC channels use the same ADC for the conversions, but due to the pipelined design a new conversion can be started on each ADC clock cycle. This means that multiple ADC conversions can be progressing simultaneously and independently without changing the MUX settings. A conversion result can be kept in one result register, independently of other result registers that are continuously updated with new conversion results. This can help reduce software complexity, and different software modules can start conversions and read conversion results fully independent of each other.

25.5 Voltage reference selection

The following voltages can be used as the voltage reference (VREF) for the ADC:

- Accurate internal 1.00V voltage.
- Internal V_{CC}/1.6V voltage.
- External voltage applied to AREF pin on PORTA.
- External voltage applied to AREF pin on PORTB.

Figure 25-8. ADC voltage reference selection



25.6 Conversion Result

The ADC can be set up to be either in signed or in unsigned mode. This setting is global for the ADC and all ADC channels.

In signed mode, both negative and positive voltages can be measured, both for single ended and differential input. 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). In unsigned mode the TOP value is 4095 and results will be in the range 0 - 4095 (0 - 0x0FFF).

Signed mode must be used when any of the ADC inputs are set up for differential measurements. In unsigned mode only single ended or internal signals can be measured.

The result of the analog to digital conversion is written to one of the result registers, RES.

In signed mode the ADC transfer function can be written as:

$$RES = \frac{VINP - VINN}{VREF} \cdot GAIN \cdot TOP$$

VINP and VINN are the positive and negative inputs to the ADC. GAIN is 1 unless differential channels with gain is used.

In unsigned mode the ADC transfer functions can be written as:

$$RES = \frac{VINP - (-\Delta V)}{VREF} \cdot TOP$$

VINP is the single ended or internal input.

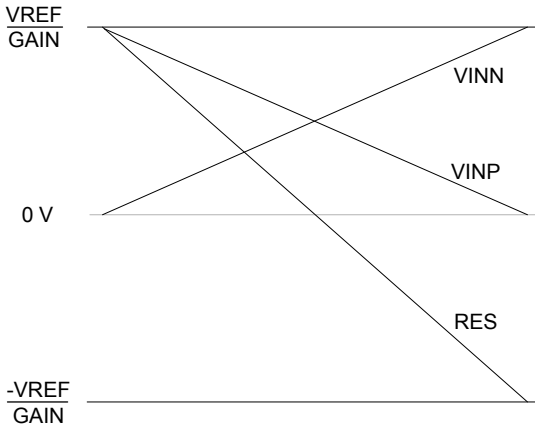
The application software selects if an 8- or 12-bit result should be generated. A result with lower resolution will be available faster. See the ”ADC Clock and Conversion Timing” on page 296 for a description on how to calculate the propagation delay.

The result registers are 16-bit. An 8-bit result is always represented right adjusted in the 16-bit result registers. Right adjusted means that the 8 LSB is found in the low byte. A 12-bit result can be represented both left- or right adjusted. Left adjusted means that the 8 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.

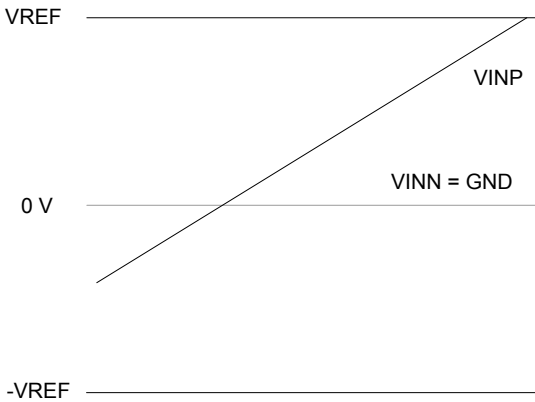
Figure 25-9 on page 295 to Figure 25-11 on page 296 shows the different input options, the signal input range and the result representation with 12-bit right adjusted mode.

Figure 25-9. Signed differential input (with gain), input range, and result representation



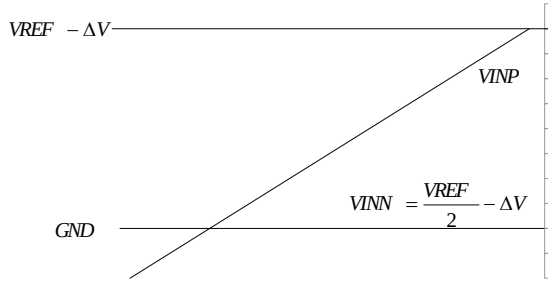
Dec	Hex	Binary	16-bit result register
2047	7FF	0111 1111 1111	0000 0111 1111 1111
2046	7FE	0111 1111 1110	0000 0111 1111 1110
2045	7FD	0111 1111 1101	0000 0111 1111 1101
...	...	...	...
3	3	0000 0000 0011	0000 0000 0000 0011
2	2	0000 0000 0010	0000 0000 0000 0010
1	1	0000 0000 0001	0000 0000 0000 0001
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 25-10. 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 25-11. 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

25.7 Compare function

The ADC has a built in 12-bit compare function. The ADC compare register can hold a 12-bit value that represent an analog threshold voltage. Each ADC Channel can be configured to automatically compare its result with this 12-bit compare value to give an interrupt or event only when the result is above or below the threshold.

All four ADC Channels share the same compare register.

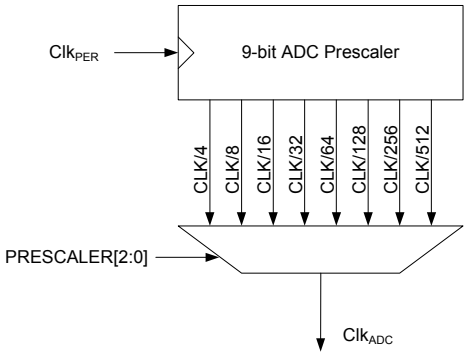
25.8 Starting a conversion

Before a conversion is started, the desired input source must be selected for one or more ADC channels. An ADC conversion for a ADC channel can either be started by the application software writing to the start conversion bit for the ADC channel, or from any of the events in the Event System. It is possible to write the start conversion bit for several ADC channels at the same time, or to use one event to trigger conversions on several ADC channels at the same time. Using this it is possible to scan several or all ADC channels from one event.

25.9 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 is within the minimum and maximum frequency for the ADC.

Figure 25-12. ADC Prescaler



The maximum ADC sample rate is given by the he ADC clock frequency (f_{ADC}). The ADC can sample a new measurement on every ADC clock cycle.

$$\text{Sample Rate} = f_{ADC}$$

The propagation delay of an ADC measurement is given by:

$$\text{Propagation Delay} = \frac{1 + \frac{RES}{2} + GAIN}{f_{ADC}}$$

RES is the resolution, 8- or 12-bit. The propagation delay will increase by one extra ADC clock cycle if the Gain Stage (GAIN) is used.

Even though the propagation delay is longer than one ADC clock cycle, the pipelined design removes any limitations on sample speed versus propagation delay.

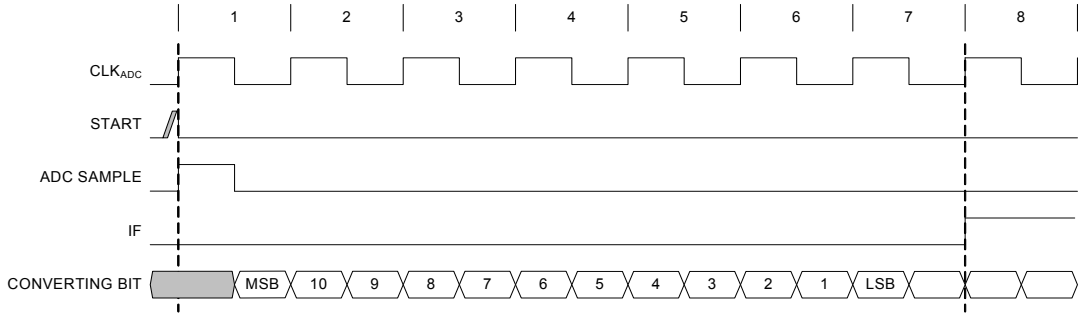
25.9.1 Single conversion without gain

Figure 25-13 on page 297 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 minimum one peripheral clock cycles before the ADC clock cycle where the conversion actually start (indicated with the grey slope of the START trigger).

The analog input source is sampled in the first half of the first cycle, and the sample time is always a half ADC clock period. Using a faster or slower ADC clock and sample rate will affect the sample time.

The Most Significant Bit (MSB) of the result is converted first, and the rest of the bits are converted during the next 3 (for 8-bit results) or 5 (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. The result is available in the Result Register for readout.

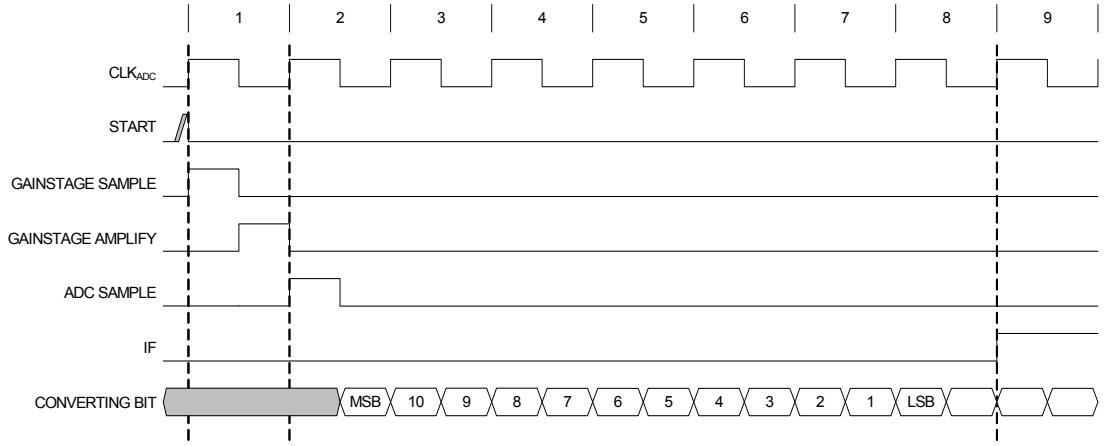
Figure 25-13. ADC timing for one single conversion without gain



25.9.2 Single conversion with gain

Figure 25-14 on page 298 shows the ADC timing for one single conversion with gain. As seen in the "Overview" on page 289 the gain stage is placed prior to the actual ADC. This means that the gainstage will sample and amplify the analog input source before the ADC samples an converts the amplified analog value. Compared to a single conversion without gain this adds one ADC clock cycle (between START and ADC Sample) for the gain stage sample and amplify. The sample time for the gain stage is a half ADC clock cycle.

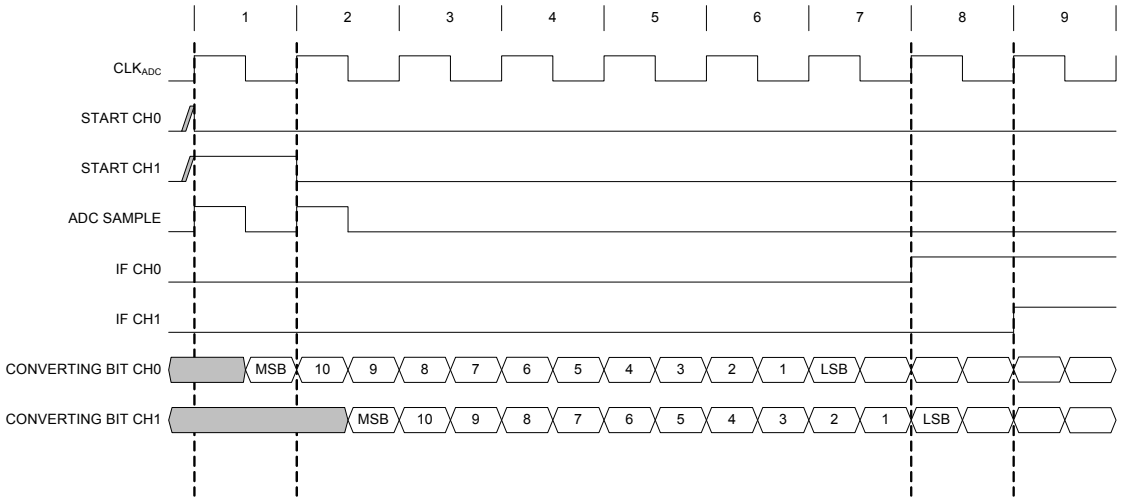
Figure 25-14. ADC timing for one single conversion with gain



25.9.3 Single conversions on two ADC channels

Figure 25-15 on page 298 shows the ADC timing for single conversions on two ADC channels. The pipelined design enables the second conversion to start on the next ADC clock cycle after t the first conversion is started. In this example both conversions is triggered at the same time, but for ADC Channel 1 (CH1) the actual start is not until the ADC sample and conversion of the MSB for ADC Channel 0 (CH0) is done.

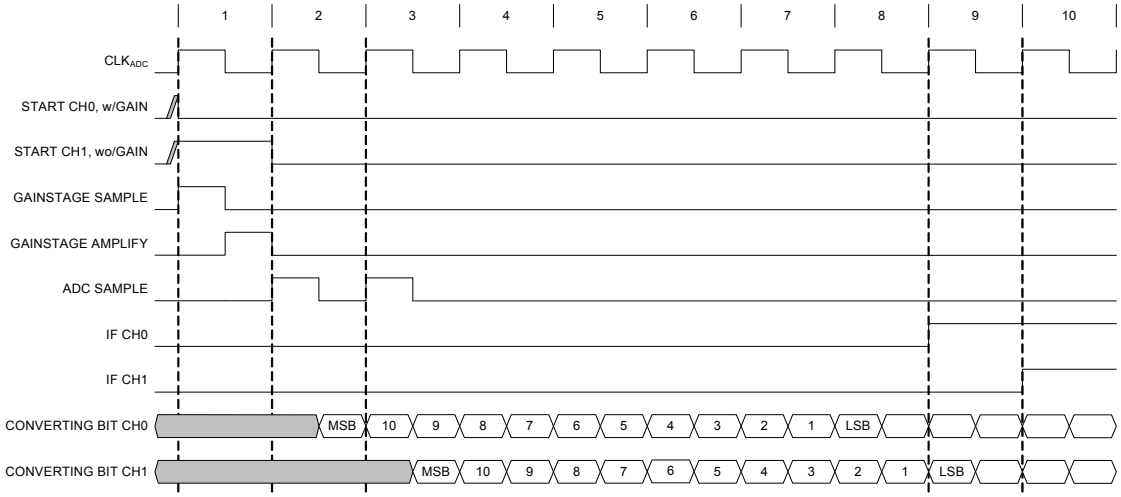
Figure 25-15. ADC timing for single conversions on two ADC channels



25.9.4 Single conversions on two ADC channels, CH0 with gain

Figure 25-16 on page 299 shows the conversion timing for single conversions on two ADC channels where ADC Channel 0 uses the gain stage. As the gain stage introduce one addition cycle for the gain sample and amplify, the sample for ADC Channel 1 is also delayed one ADC clock cycle, until the ADC sample and MSB conversion is done for ADC Channel 0.

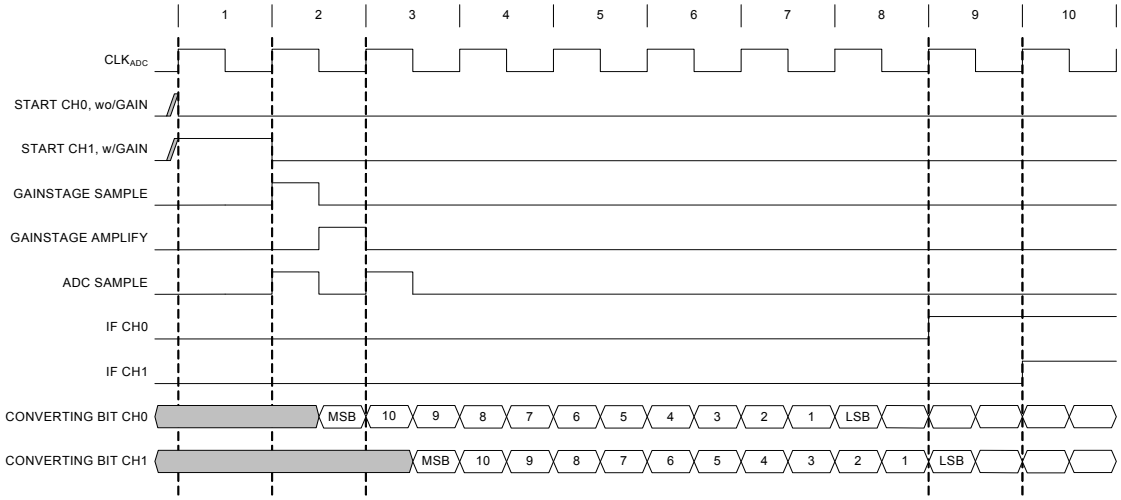
Figure 25-16. ADC timing for single conversion on two ADC channels, CH0 with gain



25.9.5 Single conversions on two ADC channels, CH1 with gain

Figure 25-17 on page 299 shows the conversion timing for single conversions on two ADC channels where ADC Channel 1 uses the gain stage.

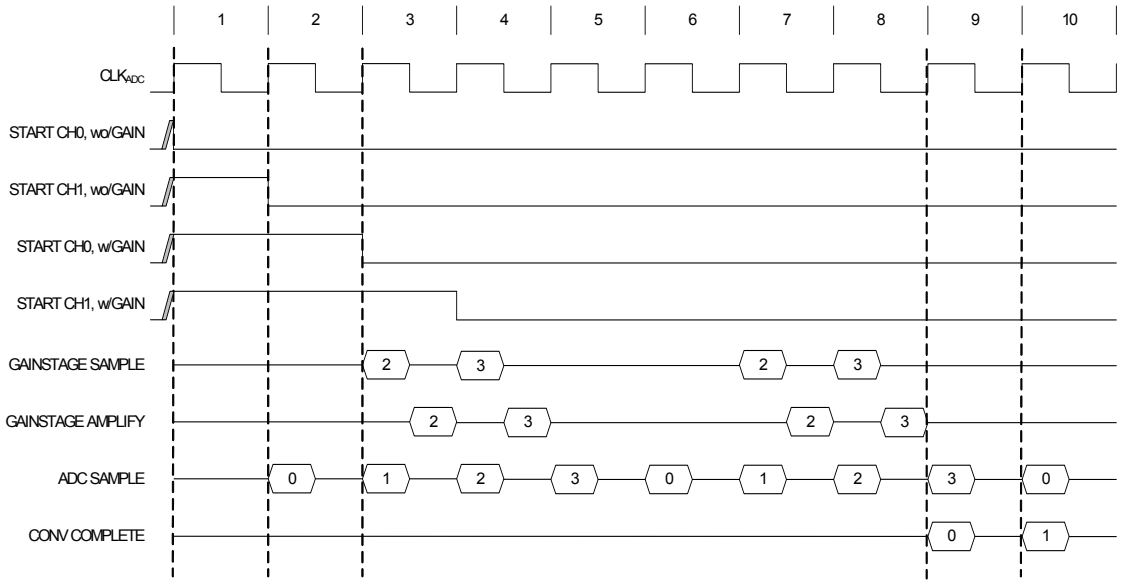
Figure 25-17. ADC timing for single conversion on two ADC channels, CH1 with gain



25.9.6 Free running mode on two ADC channels with gain

Figure 25-18 on page 300 shows the conversion timing for all four ADC channels in free running mode, CH0 and CH1 without gain and CH2 and CH3 with gain. When set up in free running mode a ADC channel will continuously sample and do new conversions. In this example all ADC channels are triggered at the same time, and each ADC channel sample and start converting as soon as the previous ADC channel is done with it sample and MSB conversion. After four ADC clock cycles all ADC channels have done the first sample and started the first conversion, and each ADC channels can then do the sample conversion start for the second conversions. After 8 (for 12-bit mode) ADC clock cycles the first conversion is done for ADC Channel 0, and the results for the rest of the ADC Channels is available in the next ADC clock cycles. After the next clock cycle (in cycle 10) the result from the second ADC Channel is done and available and so on. In this mode up to 8 conversions are ongoing at the same time.

Figure 25-18. ADC timing for free running mode



25.10 ADC Input Model

An analog voltage input must be able to charge the sample and hold (S/H) capacitor in the ADC in order to achieve maximum accuracy. Seen externally the ADC input consists of a input channel ($R_{channel}$) and the switch (R_{switch}) resistance, and the S/H capacitor. Figure 25-19 on page 300 and Figure 25-20 on page 300 shows the ADC input channels.

Figure 25-19. ADC input for single ended measurements

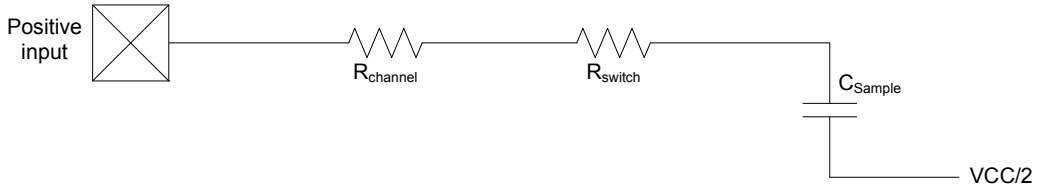
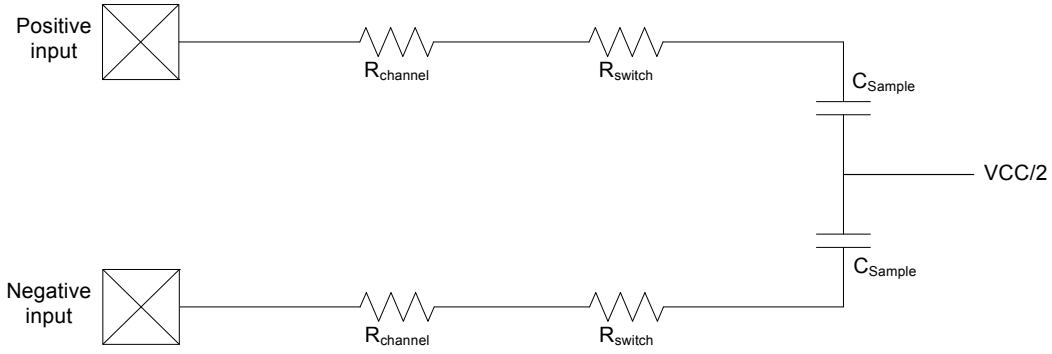


Figure 25-20. ADC input for differential measurements and differential measurements with gain



In order to achieve *n* bit 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 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 and ADC gain stage electrical characteristic in the device data sheet.

25.11 DMA transfer

The DMA Controller can be used to transfer ADC conversion results to memory or peripherals. A new conversion completed in any of the ADC result registers may trigger a DMA transfer request. See the DMA Controller manual for more details on DMA transfers.

25.12 Interrupts and events

The ADC can generate both interrupt requests and events. The ADC channels have individual interrupt settings. Interrupt requests and events can be generated either when an ADC conversion is complete or if an ADC measurement is above or below the ADC Compare register values.

25.13 Calibration

The ADC has a built-in calibration mechanism that calibrated the internal pipeline in the ADC. The calibration value from the production test must be loaded from the signature row and into the ADC calibration register from software to obtain 12 bit accuracy.

25.14 Channel priority

Since the System Clock can be faster than the ADC clock, it is possible to have the start conversion bit set for several ADC channels within the same ADC clock period. Events may also trigger conversions on several ADC channels and give the same scenario. In this case the ADC Channel with the lowest number will be prioritized. This is shown the timing diagrams in ”ADC Clock and Conversion Timing” on page 296.

25.15 Synchronous sampling

Starting an ADC conversion may cause an unknown delay between the software start or event and the actual conversion start since conversion of other higher priority ADC channels may be pending, or since the System clock may be much faster than the ADC Clock. To start an ADC conversion immediately on an incoming event, it is possible to flush the ADC for 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 all ongoing conversions in the ADC pipeline will be lost. The ADC can either be flushed from software, or the incoming event can be set up to do this automatically. If flushing is used it is important that the time between each conversion start trigger is longer than the propagation delay to ensure that one conversion is finished before the ADC pipeline is flushed and the next conversion is started.

In microcontrollers with two ADC peripherals, it is possible to start two ADC samples synchronously in the two ADCs by using the same event channel to trigger both ADCs.

25.16 Register Description - ADC

25.16.1 CTRLA - ADC Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	DMASEL[1:0]		CH[3:0]START				FLUSH	ENABLE	CTRLA
Read/Write	R/W	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 – DMASEL[1:0]: DMA Request Selection**

In addition to giving DMA transfer request for each ADC channel, the ADC can be set up to give a combined request for all channels. The combined request is decided according to the DMASEL bits. See Table 25-1 for details.

Table 25-1. ADC DMA Request Selection

DMASEL[1:0]	Group Configuration	Description
00	OFF	No combined DMA request
01	CH01	ADC Channel 0 or 1
10	CH012	ADC Channel 0 or 1 or 2
11	CH0123	ADC Channel 0 or 1 or 2 or 3

- **Bits 5:2 – CH[3:0]START: ADC Channel Start single conversion**

Setting any of these bits will start a conversion on the corresponding ADC channel. Setting several bits at the same time will start a conversion sweep on the selected ADC channels, starting with the channel with lowest number. These bits are cleared by hardware when the conversion has started.

- **Bit 1 – FLUSH: ADC Pipeline Flush:**

Setting this bit will flush the ADC pipeline. When this is done the ADC Clock will be restarted on the next Peripheral clock edge and all conversions in progress are aborted and lost.

After the flush and the ADC Clock restart, the ADC will resume where it left off. I.e. if a channel sweep was in progress or any conversions was pending, these will enter the ADC pipeline and complete.

- **Bit 0 – ENABLE: ADC Enable**

Setting this bit enables the ADC.

25.16.2 CTRLB - ADC Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x01	-	-	-	CONVMODE	FREERUN	RESOLUTION[1:0]		-	CTRLB
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 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 - CONVMODE: ADC Conversion Mode**

This bit controls whether the ADC should work in signed or unsigned mode. By default this bit is zero and the ADC is then configured for unsigned mode where single ended and internal signals can be measured. When this bit is set to one the ADC is configured for signed mode where also differential input can be used.

- **Bit 3 - FREERUN: ADC Free Running Mode**

This bit controls the free running mode for the ADC. When the bit is set to one the ADC channels defined in the EVCTRL register are swept repeatedly.

- **Bits 2:1 - RESOLUTION[1:0]: ADC Conversion Result Resolution**

These bits define whether the ADC completes the conversion at 12- or 8-bit result. They also define whether the 12-bit result is left or right oriented in the 16-bit result registers. See Table 25-2 on page 303 for possible settings.

Table 25-2. 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.

25.16.3 REFCTRL - ADC Reference Control register[1]

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	REFSEL[1:0]		-	-	BANDGAP	TEMPREF	REFCTRL
Read/Write	R	R	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Note: 1. Refer to datasheet for limitations on reference voltage.

- **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.

- **Bits 6:4 – REFSEL[1:0]: ADC Reference Selection**

These bits selects the reference and conversion range for the ADC according to Table 25-3 on page 304.

Table 25-3. ADC Reference Configuration

REFSEL[1:0]	Group Configuration	Description
00	INT1V	Internal 1.00V
01	INTVCC	Internal V_{CC}/1.6
10[1]	AREFA	External reference from AREF pin on PORT A.
11[2]	AREFB	External reference from AREF pin on PORT B.

Notes: 1. Only available if AREF exist on PORT A.
2. Only available it AREF exist on PORT B.

- **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 to prepare for ADC measurement. Note that if any other functions are using the bandgap already, this bit does not need to be set. This could be when the internal 1.00V reference is used in ADC or DAC, or if the Brown-out Detector is enabled.

- **Bit 0 – TEMPREF: Temperature Reference enable**

Setting this bit enables the temperature reference to prepare for ADC measurement.

25.16.4 EVCTRL - ADC Event Control Register

Bit	7	6	5	4	3	2	1	0	
+0x03	SWEEP[1:0]		EVSEL[2:0]			EVACT[2:0]			EVCTRL
Read/Write	R/W	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 - SWEEP[1:0]: ADC Channel Sweep**

These bits control which ADC channels are included in a channel sweep triggered by the event system or in free running mode. See Table 25-4 on page 304.

Table 25-4. ADC Channel Select

SWEEP[1:0]	Group Configuration	Active ADC channels for channel sweep
00	0	Only ADC channel 0
01	01	ADC channels 0 and 1
10	012	ADC channels 0, 1, and 2
11	0123	ADC channels 0, 1, 2, and 3

- **Bits 5:3 - EVSEL[2:0]: event channel input select**

These bits define which event channel should trigger which ADC channel. Each setting defines a group of event channels, where the event channels with the lowest number will trigger ADC channel 0 and the next event channel will trigger ADC channel 1 and so on. The number of incoming event in use is defined by the EVACT bits. See Table 25-5 on page 305.

Table 25-5. ADC Event Line Select

EVSEL[2:0]	Group Configuration	Selected event lines
000	0123	Event channel 0, 1, 2, 3 as selected inputs
001	1234	Event channel 1, 2, 3, 4 as selected inputs
010	2345	Event channel 2, 3, 4, 5 as selected inputs
011	3456	Event channel 3, 4, 5, 6 as selected inputs
100	4567	Event channel 4, 5, 6, 7 as selected inputs
101	567	Event channel 5, 6, 7 as selected inputs (Max 3 event inputs in use)
110	67	Event channel 6, 7 as selected inputs (Max 2 event inputs in use)
111	7	Event channel 7 as selected input (Max 1 event input in use)

- **Bits 2:0 - EVACT[2:0]: ADC Event Mode**

These bits define how many of the selected event channel that are in use, and also some special event modes as defined in Table 25-5 on page 305. This is for instance a complete channel sweep triggered by a single event, or an event re-synchronized conversion to achieve a very accurate timing for the conversion.

Table 25-6. 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 channel 0
010	CH01	Event channel with the two lowest numbers, defined by EVSEL trigger conversion on channel 0 and 1 respectively
011	CH012	Event channel with the three lowest numbers, defined by EVSEL trigger conversion on channel 0, 1 and 2 respectively
100	CH0123	Event channel defined by EVSEL trigger conversion on channel 0, 1, 2 and 3 respectively
101	SWEEP	One sweep of all active ADC channels defined by SWEEP on incoming event channel with the lowest number, defined by EVSEL
110	SYNCSWEEP	One sweep of all active ADC channels defined by SWEEP on incoming event channel with the lowest number, defined by EVSEL. In addition, the conversion will be synchronized on event to ensure a very accurate timing for the conversion.
111		Reserved

25.16.5 PRESCALER - ADC Clock Prescaler register

Bit	7	6	5	4	3	2	1	0	
+0x04	-	-	-	-	-	PRESCALER[2:0]			PRESCALER
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 reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bits 2:0 - PRESCALER[2:0]: ADC Prescaler configuration**

These bits define the ADC clock relative to the Peripheral clock, according to Table 25-7 on page 306.

Table 25-7. ADC Prescaler settings

PRESCALER[2:0]	Group Configuration	System 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

25.16.6 INTFLAGS - ADC Interrupt Flag register

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

- **Bits 7:4 - Reserved**

These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bits 3:0 - CH[3:0]IF: Interrupt flags**

These flags are set when the ADC conversion is complete for the corresponding ADC channel. If an ADC channel is configured for compare mode, the corresponding flag will be set if the compare condition is met. CHnIF is automatically cleared when the ADC channel n interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

25.16.7 TEMP - ADC Temporary register

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

- **Bits 7:0 - TEMP[7:0]: ADC Temporary Register**

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 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 Section 3.11 ”Accessing 16-bits Registers” on page 12.

25.16.8 CALL - ADC Calibration value registers

The CALL and CALH register pair hold the 12-bit value ADC calibration value CAL. The ADC is calibrated during production programming, 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]								CAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 - CAL[7:0]: ADC Calibration value**

This is the 8 LSB of the 12-bit CAL value.

25.16.9 CALH - ADC Calibration value registers

Bit	7	6	5	4	3	2	1	0	
+0x0D	-	-	-	-	CAL[11:8]				CAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 - CAL[11:8]: ADC Calibration value**

This is the 4 MSB of the 12-bit CAL value.

25.16.10 CHnRESH - ADC Channel n Result register High

The CHnRESL and CHnRESH register pair represents the 16-bit value CHnRES. For details on reading 16-bit register refer to Section 3.11 ”Accessing 16-bits Registers” on page 12.

	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

25.16.10.1 12-bit mode, left adjusted

- **Bits 7:0 - CHRES[11:4]: ADC Channel Result, high byte**

These are the 8 MSB of the 12-bit ADC result.

25.16.10.2 12-bit mode, right adjusted

- **Bits 7:4 - Reserved**

These bits will in practice be the extension of the sign bit CHRES11 when ADC works in differential mode and set to zero when ADC works in signed mode.

- **Bits 3:0 - CHRES[11:8]: ADC Channel Result, high byte**

These are the 4 MSB of the 12-bit ADC result.

25.16.10.3 8-bit mode

- **Bits 7:0 - Reserved**

These bits will in practice be the extension of the sign bit CHRES7 when ADC works in signed mode and set to zero when ADC works in single-ended mode.

25.16.11 CHnRESL - ADC Channel n Result register Low

	Bit	7	6	5	4	3	2	1	0
12-/8-		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

25.16.11.1 12-/8-bit mode

- **Bits 7:0 - CHRES[7:0]: ADC Channel Result, low byte**

These are the 8 LSB of the ADC result.

25.16.11.2 12-bit mode, left adjusted

- **Bits 7:4 - CHRES[3:0]: ADC Channel Result, low byte**

These are the 4 LSB of the 12 bit ADC result.

- **Bits 3:0 - Reserved**

These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

25.16.12 CMPH - ADC Compare register High

The CMPH and CMPL register pair represents the 16-bit value ADC Compare (CMP). For details on reading and writing 16-bit registers refer to Section 3.11 ”Accessing 16-bits Registers” on page 12.

Bit	7	6	5	4	3	2	1	0	
+0x19	CMP[15:0]								CMPH
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 - CMP[15:0]: ADC Compare value high byte**

These are the 8 MSB of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement and the MSB is the sign bit.

25.16.13 CMPL - ADC Compare register Low

Bit	7	6	5	4	3	2	1	0	
+0x18	CMP[7:0]								CMPL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 - CMP[7:0]: ADC compare value high byte**

These are the 8 LSB of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement.

25.17 Register Description - ADC Channel

25.17.1 CTRL - ADC Channel Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00	START	-	-	GAIN[2:0}			INPUTMODE[1:0]		CTRL
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**

Writing this to one 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 these bits is equivalent to writing the CH[3:0]START bits in ”CTRLA - ADC Control Register A” on page 302.

- **Bits 6:5 - Reserved**

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

- **Bits 4:3 - GAIN[2:0]: ADC Gain Factor**

These bits define the gain factor in order to amplify input signals before ADC conversion.

See Table 25-6 on page 305 for different gain factor settings. Gain is only valid with certain MUX settings, see "MUXCTRL - ADC Channel MUX Control registers" on page 310.

Table 25-8. 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		Reserved

- **Bit 1:0 - INPUTMODE[1:0]: Channel Input Mode**

These bits define the channel mode. This setting is independent of the ADC CONVMODE (signed/unsigned mode) setting, but differential input mode can only be done in ADC signed mode. In single ended input mode, the negative input to the ADC will be connected to a fixed value both for ADC signed and unsigned mode.

Table 25-9. 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 25-10. Channel Input Modes, CONVMODE=1 (signed mode)

INPUTMODE[1:0]	Group Configuration	Description
00	INTERNAL	Internal positive input signal
01	SINGLEENDED	Single-ended positive input signal
10	DIFF	Differential input signal
11	DIFFWGAIN	Differential input signal with gain

25.17.2 MUXCTRL - ADC Channel MUX Control registers

The MUX register defines the input source for the channel.

Bit	7	6	5	4	3	2	1	0	
+0x01	-	MUXPOS[3:0]				-	MUXNEG[1:0]		MUXCTRL
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.

- **Bits 6:3 - MUXPOS[3:0]: MUX selection on Positive ADC input**

These bits define the MUX selection for the positive ADC input. Table 25-11 on page 311 and Table 25-12 on page 311 shows the possible input selection for the different input modes.

Table 25-11. ADC MUXPOS Configuration when INPUTMODE[1:0] = 00 (Internal) is used

MUXPOS[2:0]	Group Configuration	Analog input
000	TEMP	Temperature Reference.
001	BANDGAP	Bandgap voltage
010	SCALEDVCC	1/10 scaled V_{CC}
011	DAC	DAC output
100		Reserved
101		Reserved
110		Reserved
111		Reserved

Table 25-12. ADC MUXPOS Configuration when INPUTMODE[1:0] = 01 (Single-ended), INPUTMODE[1:0] = 10 (Differential) or INPUTPMODE[1:0] = 1 (Differential with gain) is used.

MUXPOS[2:0]	Group Configuration	Analog input
000	PIN0	ADC0 pin
001	PIN1	ADC1 pin
010	PIN2	ADC2 pin
011	PIN3	ADC3 pin
100	PIN4	ADC4 pin
101	PIN5	ADC5 pin
110	PIN6	ADC6 pin
111	PIN7	ADC7 pin

For devices with more then 8 inputs into an ADC multiplexer, the MUXPOS3 bit is used to select ADC channels 8 (ADC8) and above.

- **Bits 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 - MUXNEG[1:0]: MUX selection on Negative ADC input**

These bits define the MUX selection for the negative ADC input when differential measurements are done. For internal or single-ended measurements, these bits are not in use.

Table 25-13 on page 312 andTable 25-14 on page 312 shows the possible input sections.

Table 25-13. ADC MUXNEG Configuration, INPUTMODE[1:0] = 10, Differential without gain

MUXNEG[1:0]	Group Configuration	Analog input
00	PIN0	ADC0 pin
01	PIN1	ADC1 pin
10	PIN2	ADC2 pin
11	PIN3	ADC3 pin

Table 25-14. ADC MUXNEG Configuration, INPUTMODE[1:0] = 11, Differential with gain

MUXNEG[1:0]	Group Configuration	Analog input
00	PIN4	ADC4 pin
01	PIN5	ADC5 pin
10	PIN6	ADC6 pin
11	PIN7	ADC7 pin

- **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.

25.17.3 INTCTRL - ADC Channel Interrupt Control registers

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	INTMODE[1:0}		INTLVL[1:0]		INTCTRL
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: ADC Interrupt Mode**

These bits select the interrupt mode for channel n according to Table 25-15.

Table 25-15. 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]: ADC Interrupt Priority Level and Enable**

These bits enable the ADC channel interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will be triggered when the IF in the INTFLAGS register is set.

25.17.4 INTFLAGS - ADC Channel Interrupt Flag registers

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	-	-	-	IF	INTFLAGS
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Reserved**

These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 – IF: ADC Channel 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.

25.17.5 RESH - ADC Channel n 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 Section 3.11 ”Accessing 16-bits Registers” on page 12 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

25.17.5.1 12-bit mode, left adjusted

- **Bits 7:0 - RES[11:4]: ADC Channel Result, high byte**

These are the 8 MSB of the 12-bit ADC result.

25.17.5.2 12-bit mode, right adjusted

- **Bits 7:4 - Reserved**

These bits will in practice be the extension of the sign bit CHRES11 when ADC works in differential mode and set to zero when ADC works in signed mode.

- **Bits 3:0 - RES[11:8]: ADC Channel Result, high byte**

These are the 4 MSB of the 12-bit ADC result.

25.17.5.3 8-bit mode

- **Bits 7:0 - Reserved**

These bits will in practice be the extension of the sign bit CHRES7 when ADC works in signed mode and set to zero when ADC works in single-ended mode.

25.17.6 RESL - ADC Channel n Result register Low

	Bit	7	6	5	4	3	2	1	0
12-/8-	+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

25.17.6.1 12-/8-bit mode

- **Bits 7:0 - RES[7:0]: ADC Channel Result, low byte**

These are the 8 LSB of the ADC result.

25.17.6.2 12-bit mode, left adjusted

- **Bits 7:4 - RES[3:0]: ADC Channel Result, low byte**

These are the 4 LSB of the 12 bit ADC result.

- **Bits 3:0 - Reserved**

These bits are reserved and will always read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

25.18 Register Summary - ADC

This is the register summary when the ADC is configured to give standard 12-bit results. The register summary for 8-bit and 12-bit left adjusted will be similar, but with some changes in the result registers CHnRESH and CHnRESL.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	DMASEL[1:0]		CH[3:0]START				FLUSH	ENABLE	302
+0x01	CTRLB	-	-	-	CONVMODE	FREERUN	RESOLUTION[1:0]		-	302
+0x02	REFCTRL	-	-	REFSEL[1:0]		-	-	BANDGAP	TEMPREF	303
+0x03	EVCTRL	SWEEP[1:0]		EVSEL[2:0]			EVACT[2:0]			304
+0x04	PRESCALER	-	-	-	-	-	PRESCALER[2:0]			306
+0x05	Reserved	-	-	-	-	-	-	-	-	
+0x06	INTFLAGS	-	-	-	-	CH[3:0]IF				306
+0x07	TEMP	TEMP[7:0]								307
+0x08	Reserved	-	-	-	-	-	-	-	-	
+0x09	Reserved	-	-	-	-	-	-	-	-	
+0x0A	Reserved	-	-	-	-	-	-	-	-	
+0x0B	Reserved	-	-	-	-	-	-	-	-	
+0x0C	CALL	CAL[7:0]								307
+0x0D	CALH	-	-	-	-	CAL[11:8]				
+0x0E	Reserved	-	-	-	-	-	-	-	-	
+0x0F	Reserved	-	-	-	-	-	-	-	-	
+0x10	CH0RESL	CH0RES[7:0]								308
+0x11	CH0RESH	CH0RES[15:8]								307
+0x12	CH1RESL	CH1RES[7:0]								308
+0x13	CH1RESH	CH1RES[15:8]								307
+0x14	CH2RESL	CH2RES[7:0]								308
+0x15	CH2RESH	CH2RES[15:8]								307
+0x16	CH3RESL	CH3RES[7:0]								308
+0x17	CH3RESH	CH3RES[15:8]								307
+0x18	CMPL	CMP[7:0]								309
+0x19	CMPH	CMP[15:8]								308
+0x1A	Reserved	-	-	-	-	-	-	-	-	
+0x1B	Reserved	-	-	-	-	-	-	-	-	
+0x1C	Reserved	-	-	-	-	-	-	-	-	
+0x1D	Reserved	-	-	-	-	-	-	-	-	
+0x1E	Reserved	-	-	-	-	-	-	-	-	
+0x1F	Reserved	-	-	-	-	-	-	-	-	
+0x20	CH0 Offset									
+0x28	CH1 Offset									
+0x30	CH2 Offset									
+0x38	CH3 Offset									

25.19 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]		309
+0x01	MUXCTRL	-	MUXPOS[3:0]				-	MUXNEG[1:0]		310
+0x02	INTCTRL	-	-	-	-	INTMODE[1:0]		INTLVL[1:0]		312
+0x03	INTFLAGS	-	-	-	-	-	-	-	IF	313
+0x04	RESL	RES[7:0]								314
+0x05	RESH	RES[15:8]								313
+0x06	Reserved	-	-	-	-	-	-	-	-	
+0x07	Reserved	-	-	-	-	-	-	-	-	

25.20 Interrupt vector Summary

Table 25-16. Analog to Digital Converter Interrupt vectors and their word offset address

Offset	Source	Interrupt Description
0x00	CH0	Analog to Digital Converter Channel 0 Interrupt vector
0x02	CH1	Analog to Digital Converter Channel 1 Interrupt vector
0x04	CH2	Analog to Digital Converter Channel 2 Interrupt vector
0x06	CH3	Analog to Digital Converter Channel 3 Interrupt vector

26. DAC - Digital to Analog Converter

26.1 Features

- **12-bit resolution**
- **Up to 1 Msps conversion rate**
- **Flexible conversion range**
- **Multiple trigger sources**
- **1 continuous time or 2 Sample/Hold (S/H) outputs**
- **Built-in offset and gain calibration**
- **High drive capabilities**
- **Internal/External reference**
- **Possibility to use as input to analog comparator or ADC**
- **Power reduction mode**

26.2 Overview

The DAC converts digital values to analog voltages. The DAC has 12-bit resolution and is capable of converting 1 million samples per second. The output from the DAC can either be continuous to one pin, or fed to two different pins using a sample and hold (S/H) circuitry. A separate low power mode is available, and the DAC can be gain and offset calibrated.

The output signal swing is defined by the reference voltage (VREF). The following sources are available as VREF in the DAC:

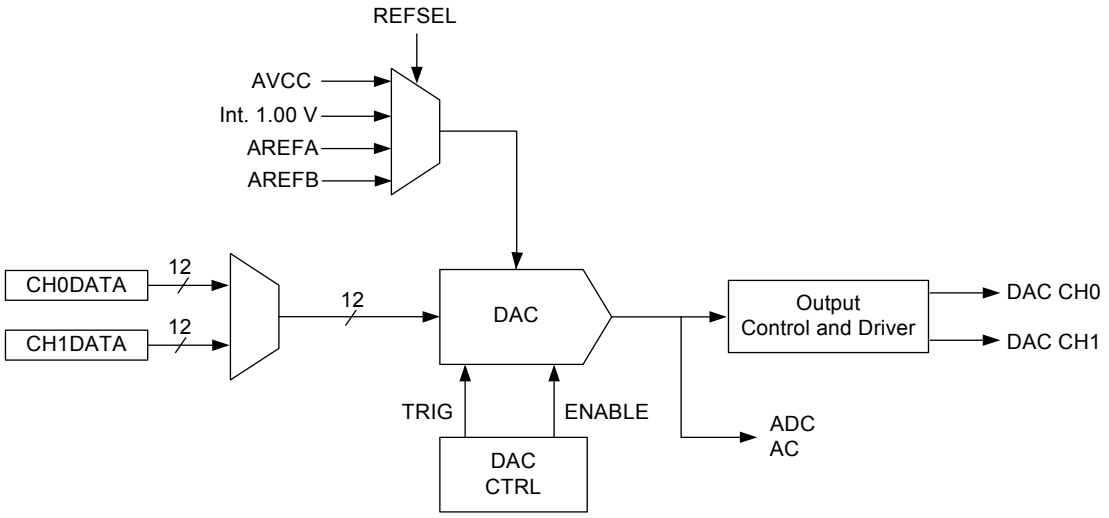
- AV_{CC}
- Internal 1.00V
- External reference applied to the AREF pin on PORTA or PORTB

The output voltage from a DAC channel is given as:

$$V_{DACx} = \frac{CHnDATA}{0xFFF} \cdot VREF$$

Figure 26-1 on page 318 illustrates the basic functionality related to the DAC. Not all functionality is shown.

Figure 26-1. DAC overview



26.3 Starting a conversion

Conversions are either done when data is written to the data registers, or timed by an incoming event.

If auto trigger mode is not selected, a new conversion is automatically started when there is a new value in the DAC data register. When auto trigger mode is selected, the new conversion will be started on an incoming event from the selected event channel if there is new data in the data register which has not been converted.

Both application software and the DMA controller may write to the data registers.

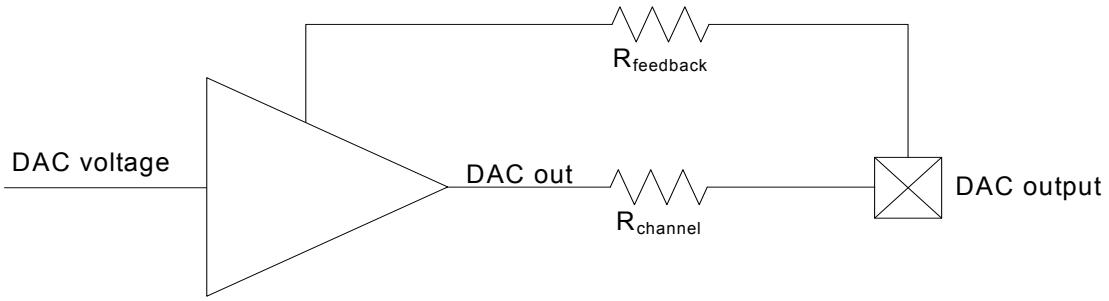
26.4 Output channels

The output from the DAC can either be continuous to one pin (Channel 0), or fed into two different pins using a sample and hold circuitry (S/H). With S/H these two outputs can act independently and create two different analog signals, different in both voltage and frequency. The two S/H outputs have individual data and conversion control registers. The DAC output may be used as input to other peripherals in XMEGA, such as the Analog Comparator or the Analog to Digital Converter. It is the output directly from the DAC, not the S/H outputs, that is available for these peripherals.

26.5 DAC Output model

Each DAC output channel has a driver buffer with feedback ensures that the voltage on the DAC output pin is equal to the DACs internal voltage until the DAC output reaches its saturation voltage. Figure 26-2 on page 319 shows the DAC output model, for details on $R_{channel}$, refer to the DAC characteristics in the device data sheet.

Figure 26-2. DAC output model



Notes: 1. The DAC $R_{channel}$ is ~300 Ω for ATxmegaA3 and ATxmegaA4
2. The DAC $R_{channel}$ is ~850 Ω for ATxmegaA1

26.6 DAC clock

The DAC is clocked from the Peripheral clock (clk_{PER}) directly. The DAC conversion interval and refresh rate in S/H mode is configured relative to the Peripheral Clock.

26.7 Timing constraints

Some timing constraints are given to make sure the DAC operates correctly. The timing constraints are relative to the frequency of the Peripheral clock. Not meeting the timing constraints may reduce the accuracy of DAC conversions.

- The DAC sampling time is the time interval between a completed channel conversion until starting a new conversion. This should not be less than 1 µs for single channel mode and 1.5 µs for dual channel (S/H) mode.
- The DAC refresh time is the time interval between each time a channel is updated in dual channel mode. This should not be more than 30 µs.

26.8 Low Power mode

To reduce the power consumption in DAC conversions, the DAC may be set in a Low Power mode. In Low Power mode, the DAC is turned off between each conversion. Conversion time will be longer if new conversions are started in this mode. To put the DAC into low power mode, you need to set the bit 1 in CTRLA register (called DACCRA).

26.9 Calibration

To achieve optimal accuracy, it is possible to calibrate both gain and offset error in the DAC. There is a 7-bit calibration value for gain adjustment and a 7-bit calibration value for offset adjustment.

To get the best calibration result it is recommended to use the same VREF, output channel selection, sampling time, and refresh interval when calibrating as will be used in normal DAC operation. The theoretical transfer function for the DAC was shown in ”Overview” on page 317. Including errors, the DAC output value can be expressed as:

$$V_{DACxX} = gain \cdot \frac{CHnDATA}{0xFFF} + offset$$

In an ideal DAC, gain is 1 and offset is 0.

The calibration of the DAC adjust the offset and gain. To calibrate offset you can output mid code and adjust the offset calibration until you get ~0 LSB offset The gain is adjusted around mid code so it should not affect the offset calibration if you read the output at mid code and max (or min code) and adjust the calibration values until you get ~0 LSB gain.

26.10 Register Description

26.10.1 CTRLA – DAC Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	IDOEN	CH1EN	CH0EN	.	ENABLE	CTRLA
Read/Write	R	R	R	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 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 - IDOEN: DAC Internal Output Enable**

Setting this bit routs the internal DAC output to the ADC and Analog Comparator MUXes.

- **Bit 3 - CH1EN: DAC Channel 1 Output Enable**

Setting this bit will make channel 1 available on pin while clearing the bit makes channel 1 only available for internal use.

- **Bit 2 - CH0EN: DAC Channel 0 Output Enable**

Setting this bit will make channel 0 available on pin while clearing the bit makes channel 0 only available for internal use.

- **Bit 1 - Res - 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 - ENABLE: DAC Enable**

This bit enables the entire DAC.

26.10.2 CTRLB – DAC Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x01	-	CHSEL[1:0]		-	-	-	CH1TRIG	CH0TRIG	CTRLB
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - Reserved**

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

- **Bits 6:5 - CHSEL[1:0]: DAC Channel Selection**

These bits control wether the DAC should operate with single or dual channel outputs. Table 26-1 shows the available selections.

Table 26-1. DAC channel selection

CHSEL[1:0]	Description
00	Single channel operation (for channel 0 only)
01	Reserved
10	Duel channel operation (S/H for channel 0 and channel 1)
11	Reserved

- **Bits 4:2 - Reserved**

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

- **Bit 1 - CH1TRIG: DAC Auto trigged mode Channel 1**

If this bit is set, the incoming event on the event channel selected in the EVCTRL Register will start the conversion when a new value is written to high byte of the data register CH1DATA.

- **Bit 0 - CH0TRIG: DAC Auto trigged mode Channel 0**

If this bit is set, the incoming event on the event channel selected in the EVCTRL Register will start the conversion when a new value is written to high byte of the data register CH0DATA.

26.10.3 CTRLC – DAC Control Register C

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	REFSEL[1:0]		-	-	LEFTADJ	CTRLC
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	

- **Bits 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.

- **Bits 4:3 - REFSEL[1:0]: DAC Reference Selection**

These bits control the reference and thus the conversion range of the DAC.

Table 26-2 shows the available options.

Table 26-2. DAC Reference selection

REFSEL[1:0]	Group Configuration	Description
00	INT1V	Internal 1.00 V
01	AVCC	AV_{CC}
10	AREFA	AREF on PORTA
11	AREFB	AREF on PORTB

- **Bit 2:1 - Reserved**

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

- **Bit 0 - LEFTADJ: DAC Left-Adjust Value**

If this bit is set, CH0DATA and CH1DATA are left-adjusted.

26.10.4 EVCTRL – DAC Event Control Register

Bit	7	6	5	4	3	2	1	0	
+0x03	-	-	-	-	-	EVSEL[2:0]			EVCTRL
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 - Reserved**

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

- **Bits 2:1 - EVSEL[2:0]: DAC Event Channel Input Selection**

These bits define which channel from the Event System that is used for triggering a DAC conversion.

Table 26-3 shows the available selections.

Table 26-3. DAC Event input Selection

EVSEL[2:0]	Group Configuration	Description
000	0	Event channel 0 as input to DAC
001	1	Event channel 1 as input to DAC
010	2	Event channel 2 as input to DAC
011	3	Event channel 3 as input to DAC
100	4	Event channel 4 as input to DAC
101	5	Event channel 5 as input to DAC
110	6	Event channel 6 as input to DAC
111	7	Event channel 7 as input to DAC

26.10.5 TIMCTRL – DAC Timing Control Register

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

- **Bit 7 - Reserved**

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

- **Bits 6:4 - [2:0]: - CONINTVAL - DAC Conversion Interval**

These bits control the minimum interval between two successive conversions. The interval must be set relative to the Peripheral clock (clk_{PER}) to ensure that a new conversion is not started until the result from the previous conversion has settled. The DAC Conversion Interval should never be set lower than 1 µs during single channel operation, and not lower than 1.5 µs during dual channel (S/H) operation.

Table 26-4 shows the available control settings as a number of peripheral clock cycles. To allow for longer conversion intervals during dual channel operation, a 50% increase in the number peripheral clock cycles is automatically added.

Table 26-4. DAC Conversion Interval

CONINTCAL[2:0]	Group Configuration	clk_{PER} cycles for Single channel operation	clk_{PER} cycles for dual channel (S/H) operation
000	1CLK	1 CLK	1 CLK
001	2CLK	2 CLK	3 CLK
010	4CLK	4 CLK	6 CLK
011	8CLK	8 CLK	12 CLK
100	16CLK	16 CLK	24 CLK
101	32CLK	32 CLK	48 CLK
110	64CLK	64 CLK	96 CLK
111	128CLK	128 CLK	192 CLK

The number of clock cycles selected multiplied with the period of the Peripheral clock gives the minimum DAC conversion internal.

- **Bits 3:0 - REFRESH[3:0]: DAC Channel Refresh Timing Control**

These bits control time interval between each time a channel is refreshed in dual channel (S/H) mode. The interval must be set relative to the Peripheral clock to avoid loosing accuracy of the converted value.

Table 26-5 shows the available timing control settings as a number of peripheral clock cycles.

Table 26-5. DAC Channel refresh control selection

REFRESH[3:0]	Group Configuration	clk_{PER} cycles refresh interval
0000	16CLK	16 CLK
0001	32CLK	32 CLK
0010	64CLK	64 CLK
0011	128CLK	128 CLK
0100	256CLK	256 CLK
0101	512CLK	512 CLK
0110	1024CLK	1024 CLK
0111	2048CLK	2048 CLK
1000	4096CLK	4096 CLK
1001	8192CLK	8192 CLK
1010	16384CLK	16384 CLK
1011	32768CLK	32768 CLK
1100	65536CLK	65536 CLK

Table 26-5. DAC Channel refresh control selection (Continued)

1101		Reserved
1110		Reserved
1111	OFF	Auto refresh off

The number of clock cycles selected multiplied with the period of the Peripheral clock gives the DAC refresh time.

26.10.6 STATUS – DAC Status Register

Bit	7	6	5	4	3	2	1	0	
+0x05	-	-	-	-	-	-	CH1DRE	CH0DRE	STATUS
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 - Reserved**

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

- **Bit 1 - CH1DRE: DAC Channel 1 Data Register Empty**

This bit indicates that the data register for channel 1 is empty, meaning that a new conversion value may be written. If the bit is cleared, writing to the data register may cause losing a conversion value. This bit is directly used for DMA Request.

- **Bit 0 - CH0DRE: DAC Channel 0 Data register Empty**

This bit indicates that the data register for channel 0 is empty, meaning that a new conversion value may be written. If the bit is cleared, writing to the data register may cause losing a conversion value.This bit is directly used for DMA Request.

26.10.7 CH0DATAH – DAC Channel 0 Data Register High

The two registers CHnDATAH and CHnDATAL are the high byte and low byte respectively of the 12-bit value CHnDATA that is converted to an analog voltage on DAC channel n. By default, the 12 bits are distributed with 8 bits in CHnDATAL and 4 bits in 4 LSB position of CHnDATAH (right-adjusted).To select left-adjusted data it is possible by setting the LEFTADJ bit in the CTRLC register.

When this is selected, it is also possible to do 8-bit conversions by writing only CHnDATAH register.

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x19	-	-	-	-	CHDATA[11:8]			
Left-adjust		CHDATA[11:4]							
Right-adjust	Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

26.10.7.1 Right-adjusted

- **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.

- **Bits 3:0 - CHDATA[11:8]: DAC Conversion Data Register Channel 0, 4 MSB**

These bits are the 4 MSB of the 12-bit value to convert to channel 0 in right-adjusted mode.

26.10.7.2 Left-adjusted

- **Bits 7:0 - CHDATA[11:4]: DAC Conversion Data Register Channel 0, 8 MSB**

These bits are the 8 MSB of the 12-bit value to convert to channel 0 in left-adjusted mode.

26.10.8 CH0DATAL – DAC Channel 0 Data Register Low

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x18	CHDATA[7:0]							
Left-adjust		CHDATA[3:0]				-	-	-	-
Right-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

26.10.8.1 Right-adjusted

- **Bits 7:0 - CHDATA[7:0]: DAC Conversion Data Register Channel 0, 8 LSB**

These bits are the 8 LSB of the 12-bit value to convert to channel 0 in right-adjusted mode.

26.10.8.2 Left-adjusted

- **Bits 7:4 - CHDATA[3:0]: DAC Conversion Data Register Channel 0, 4 LSB**

These bits are the 4 LSB of the 12-bit value to convert to channel 0 in left-adjusted mode.

- **Bits 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.

26.10.9 CH1DATAH – DAC Channel 1 Data Register High

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x1B	-	-	-	-	CHDATA[11:8]			
Left-adjust		CHDATA[11:4]							
Right-adjust	Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

26.10.9.1 Right-adjusted

- **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.

- **Bits 3:0 - CHDATA[11:8]: DAC Conversion Data Register Channel 1, 4 MSB**

These bits are the 4 MSB of the 12-bit value to convert to channel 1 in right-adjusted mode.

26.10.9.2 Left-adjusted

- **Bits 7:0 - CHDATA[11:4]: DAC Conversion Data Register Channel 1, 8 MSB**

These bits are the 8 MSB of the 12-bit value to convert to channel 1 in left-adjusted mode.

26.10.10 CH1DATAL – DAC Channel 1 Data Register Low byte

	Bit	7	6	5	4	3	2	1	0
Right-adjust	+0x1A	CHDATA[7:0]							
Left-adjust		CHDATA[3:0]				-	-	-	-
Right-adjust	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Left-adjust	Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Right-adjust	Initial Value	0	0	0	0	0	0	0	0
Left-adjust	Initial Value	0	0	0	0	0	0	0	0

26.10.10.1 Right-adjusted

- **Bits 7:0 - CHDATA[7:0]: DBC Conversion Data Register Channel 1, 8 LSB**

These bits are the 8 lsb of the 12-bit value to convert to channel 1 in right-adjusted mode.

26.10.10.2 Left-adjusted

- **Bits 7:4 - CHDATA[3:0]: DAC Conversion Data Register Channel 1, 4 LSB**

These bits are the 4 lsb of the 12-bit value to convert to channel 1 in left-adjusted mode.

- **Bits 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.

26.10.11 GAINCAL – DAC Gain Calibration Register

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

- **Bit 7 - Reserved**

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

- **Bits 6:0 - GAINCAL[6:0]: DAC Gain Calibration value**

These bits are used to compensate the gain error in the DAC. See ”Calibration” on page 319 for details on how to calibrate gain.

26.10.12 OFFSETCAL – DAC Offset Calibration Register

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

- **Bit 7 - Reserved**

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

- **Bits 6:0 - OFFSETCAL[6:0]: DAC Offset Calibration value**

These bits are used to compensate the offset error in the DAC. See ”Calibration” on page 319 for details on how to calibrate offset.

26.11 Register Summary

This is the I/O summary when the DAC is configured to give standard 12-bit results. I/O summary for 12-bit left adjusted will be similar, but with some changes in the data registers CHnDATAL and CHnDATAH.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	-	-	-	-	CH1EN	CH0EN	LPMODE	ENABLE	320
+0x01	CTRLB	-	CHSEL[1:0]		-	-	-	CH1TRIG	CH0TRIG	320
+0x02	CTRLC	-	-	-	REFSEL[1:0]		-	-	LEFTADJ	321
+0x03	EVCTRL	-	-	-	-	-	EVSEL[2:0]			322
+0x04	TIMCTRL	-	CONINTVAL[2:0]			REFRESH[3:0]				322
+0x05	STATUS	-	-	-	-	-	-	CH1DRE	CH0DRE	324
+0x06	Reserved	-	-	-	-	-	-	-	-	
+0x07	Reserved	-	-	-	-	-	-	-	-	
+0x08	GAINCAL	-	GAINCAL[6:0]							326
+0x09	OFFSETCAL	-	OFFSETCAL[6:0]							327
+0x10	Reserved	-	-	-	-	-	-	-	-	
+0x11	Reserved	-	-	-	-	-	-	-	-	
+0x12	Reserved	-	-	-	-	-	-	-	-	
+0x13	Reserved	-	-	-	-	-	-	-	-	
+0x14	Reserved	-	-	-	-	-	-	-	-	
+0x15	Reserved	-	-	-	-	-	-	-	-	
+0x16	Reserved	-	-	-	-	-	-	-	-	
+0x17	Reserved	-	-	-	-	-	-	-	-	
+0x18	CH0DATAL	CHDATA[7:0]								325
+0x19	CH0DATAH	-	-	-	-	CHDATA[11:8]				324
+0x1A	CH1DATAL	CHDATA[7:0]								326
+0x1B	CH1DATAH	-	-	-	-	CHDATA[11:8]				325

27. AC - Analog Comparator

27.1 Features

- **Flexible input selection**
- **High speed option**
- **Low power option**
- **Selectable input hysteresis**
- **Analog comparator output available on pin**
- **Window mode**

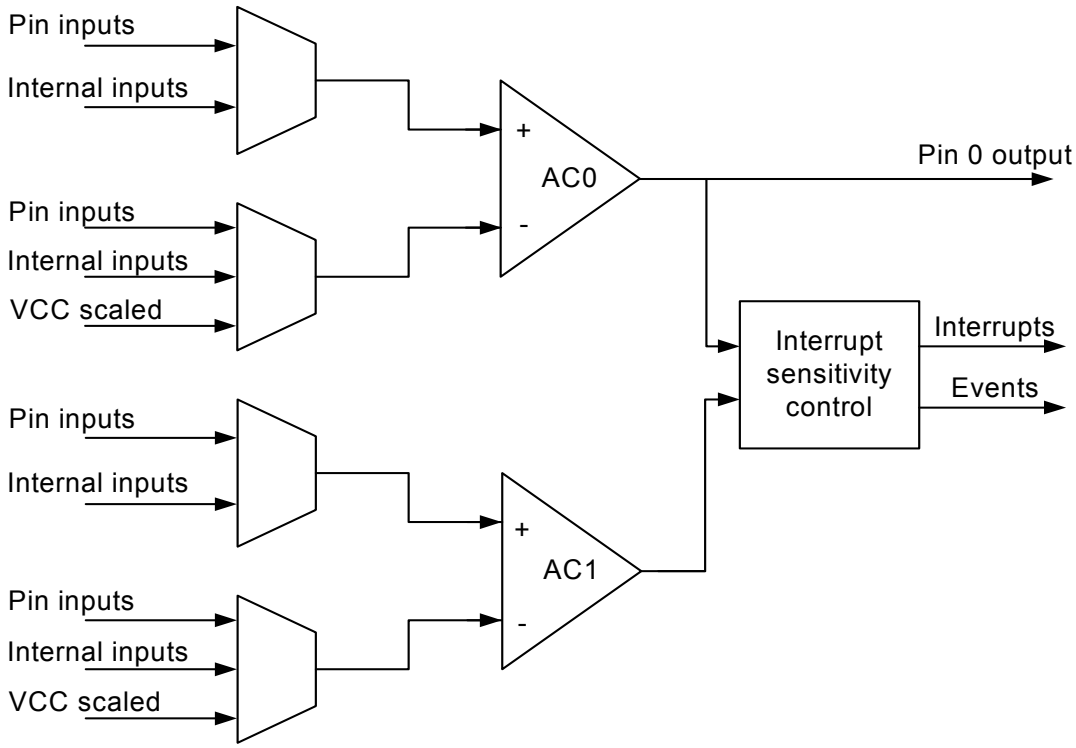
27.2 Overview

The Analog Comparator (AC) compares the voltage level on two inputs and gives a digital output based on this comparison. The Analog Comparator may be configured to give interrupt requests and/or events upon several different combinations of input change.

Two important properties of the Analog Comparator when it comes to the dynamic behavior, are hysteresis and propagation delay. Both these parameters may be adjusted in order to find the optimal operation for each application.

The Analog Comparators are always grouped in pairs (AC0 and AC1) on each analog port. They have identical behavior but separate control registers.

Figure 27-1. Analog Comparator overview.



27.3 Input Channels

Each Analog Comparator has one positive and one negative input. Each input may be chosen among a wide selection of input channels: the analog input pins, internal inputs and a scaled inputs. The digital output from the Analog Comparator is one when the difference between the positive and the negative input is positive, and zero when the difference is negative.

27.3.1 Pin Inputs

The analog input pins on the port can be selected as input to the Analog Comparator.

27.3.2 Internal Inputs

There are three Internal inputs that are directly available for the Analog Comparator:

- Output from the DAC (If available on the specific device).
- Bandgap reference voltage.
- Voltage scaler that can do a 64-level scaling of the internal VCC voltage.

27.4 Start of Signal Compare

In order to start a signal compare, the Analog Comparator must be configured with the preferred properties and inputs, before the module is enabled to start comparing the two selected inputs. The result of the comparison is continuous and available for application software and the Event System.

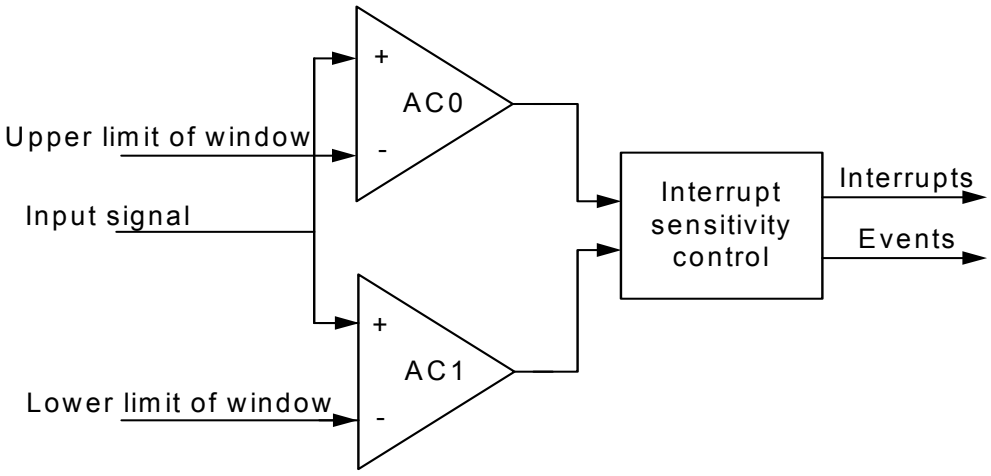
27.5 Generating Interrupts and Events

The Analog Comparator can be configured to generate interrupts when the output toggles, when output changes from zero to one (rising edge) or when the output changes from one to zero (falling edge). Events will be generated for the same condition as the interrupt, and at all times, regardless of the interrupt being enabled or not.

27.6 Window Mode

Two Analog Comparators on the same analog port can be configured to work together in Window Mode. In this mode a voltage range may be defined, and the Analog Comparators may give information about whether an input signal is within this range or not.

Figure 27-2. Analog Comparator Window Mode



27.7 Input hysteresis

Application software can select between no, low, and high hysteresis. Adding hysteresis can avoid constant toggling of the output if the input signals are very close to each other and some noise exists in either of the signals or in the system.

27.8 Power consumption vs. propagation delay

It is possible to enable High-speed mode to get the shortest possible propagation delay. This mode consumes more power than the default Low-power mode that has a longer propagation delay.

27.9 Register Description

27.9.1 ACnCTRL – Analog Comparator n Control Register

Bit	7	6	5	4	3	2	1	0	
+0x00 / +0x01	INTMODE[1:0]		INTLVL[1:0]		HSMODE	HYSMODE[2:0]		ENABLE	ACnCTRL
Read/Write	R/W	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 - INTMODE[1:0]:Analog Comparator Interrupt Modes**

These bits configure the interrupt mode for Analog Comparator n according to Table 27-1.

Table 27-1. Analog Comparator n Interrupt Settings

INTMODE[1:0]	Group Configuration	Description
00	BOTHEDGES	Comparator interrupt 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

- **Bits 5:4 - INTLVL[1:0]: Analog Comparator Interrupt Level**

These bits enable the Analog Comparator n Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger according to the INTMODE setting.

- **Bit 3 - HSMODE: Analog Comparator High-Speed Mode Select**

Setting this bit selects High-speed mode and clearing this bit to select Low-power mode.

- **Bits 2:1 - HYSMODE[1:0]: Analog Comparator Hysteresis Mode Select**

These bits select hysteresis according to Table 27-2. For details on actual hysteresis levels refer to device data sheet.

Table 27-2. Analog Comparator n 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: Analog Comparator Enable**

Settings this bit enables the Analog Comparator n.

27.9.2 ACnMUXCTRL – Analog Comparator Control Register

Bit	7	6	5	4	3	2	1	0	
+0x02 / +0x03	-	-	MUXPOS[2:0]			MUXNEG[2:0]			ACnMUXCTRL
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 - Reserved**

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

- **Bits 5:3 - MUXPOS[2:0]: Analog Comparator Positive Input MUX Selection**

These bits select which input to be connected to the positive input of Analog Comparator n, according to Table 27-3.

Table 27-3. Analog Comparator n 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

Table 27-3. Analog Comparator n Positive Input MUX Selection (Continued)

101	PIN5	Pin 5
110	PIN6	Pin 6
111	DAC	DAC Output

- **Bits 2:0 - MUXNEG[2:0]: Analog Comparator Negative Input MUX Selection**

These bits select which input to be connected to the negative input of Analog Comparator n, according to Table 27-4 on page 333.

Table 27-4. Analog Comparator n Negative Input MUX Selection

MUXNEG[2:0]	Group Configuration	Negative Input MUX Selection
000	PIN0	Pin 0
001	PIN1	Pin 1
010	PIN3	Pin 3
011	PIN5	Pin 5
100	PIN7	Pin 7
101	DAC	DAC Output
110	BANDGAP	Internal Bandgap Voltage
111	SCALER	VCC Voltage Scaler

27.9.3 CTRLA – Control Register A

Bit	7	6	5	4	3	2	1	0	
+0x04	-	-	-	-	-	-	-	AC0OUT	CTRLA
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 – AC0OUT: Analog Comparator Output**

Setting this bit makes the output of Analog Comparator 0 available on pin 7 on the same port.

27.9.4 CTRLB – Control Register B

Bit	7	6	5	4	3	2	1	0	
+0x05	-	-	SCALEFAC[5:0]						CTRLB
Read/Write	R/W	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 - Res - 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 - SCALEFAC[5:0]: Analog Comparator Input Voltage Scaling Factor**

These bits define the scaling factor for the Vcc voltage$_F$. The input to the Analog Comparator, V_{SCALE}, is:

$$V_{SCALE} = \frac{V_{CC} \cdot (SCALEFAC + 1)}{64}$$

27.9.5 WINCTRL – Analog Comparator Window Function Control Register

Bit	7	6	5	4	3	2	1	0	
+0x06	-	-	-	WEN	WINTMODE[1:0]		WINTLVL[1:0]		WINCTRL
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	

- **Bits 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: Analog Comparator Window Enable**

Setting this bit enables Window Mode for the two Analog Comparators on the same port.

- **Bits 3:2 - WINTMODE[1:0]: Analog Comparator Window Interrupt Mode Settings**

These bits configure the interrupt mode for Analog Comparator Window Mode according to Table 27-5.

Table 27-5. Analog Comparator 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]: Analog Comparator Window Interrupt Enable**

These bits enable the Analog Comparator Window Mode Interrupt and select the interrupt level as described in Section 12. ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123. The enabled interrupt will trigger according to the WINTMODE setting.

27.9.6 STATUS – Analog Comparator Common Status Register

Bit	7	6	5	4	3	2	1	0	
+0x07	WSTATE[1:0]		AC1STATE	AC0STATE	-	WIF	AC1IF	AC0IF	STATUS
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]: Analog Comparator Window Mode Current State**

These bits show the current state of the signal if the Window Mode is enabled according to Table 27-6.

Table 27-6. Analog Comparator 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	-	Reserved

- **Bit 5 - AC1STATE: Analog Comparator 1 Current State**

This bit shows the current state of the input signal to Analog Comparator 1.

- **Bit 4 - AC0STATE: Analog Comparator 0 Current State**

This bit shows the current state of the input signal to Analog Comparator 0.

- **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 Section 27.9.5 ”WINCTRL – Analog Comparator Window Function Control Register” on page 334.

The WIF 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 Analog Comparator 1. AC1IF is set according to the INTMODE setting in the corresponding Section 27.9.1 ”ACnCTRL – Analog Comparator n Control Register” on page 331.

The AC1IF 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 Analog Comparator 0. AC0IF is set according to the INTMODE setting in the corresponding Section 27.9.1 ”ACnCTRL – Analog Comparator n Control Register” on page 331.

The AC0IF is automatically cleared when the analog comparator interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

27.10 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]		HSMODE	HYSMODE[1:0]		ENABLE	331
+0x01	AC1CTRL	INTMODE[1:0]		INTLVL[1:0]		HSMODE	HYSMODE[1:0]		ENABLE	331
+0x02	AC0MUXCTRL	-	-	MUXPOS[2:0]			MUXNEG[2:0]			332
+0x03	AC1MUXCTRL	-	-	MUXPOS[2:0]			MUXNEG[2:0]			332
+0x04	CTRLA	-	-	-	-	-	-	-	ACOOUT	333
+0x05	CTRLB	-	-	SCALEFAC5:0]						333
+0x06	WINCTRL	-	-	-	WEN	WINTMODE[1:0]		WINTLVL[1:0]		334
+0x07	STATUS	WSTATE[1:0]		AC1STATE	AC0STATE	-	WIF	AC1IF	AC0IF	334

27.11 Interrupt vector Summary

Table 27-7. Analog Comparator Interrupt vectors

Offset	Source	Interrupt Description
0x00	COMP0_vect	Analog Comparator 0 Interrupt vector
0x02	COMP1_vect	Analog Comparator 1 Interrupt vector
0x04	WINDOW_vect	Analog Comparator Window Interrupt vector

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, PRELOAD, EXTEST, and BYPASS instructions.**
- **Supports the optional IDCODE, HIGHZ, and CLAMP instructions.**
- **Supports the AVR specific PDICOM instruction for accessing the PDI for debugging and programming in its optional JTAG mode.**

28.2 Overview

The JTAG Boundary-scan interface is mainly intended for testing PCBs by using the JTAG Boundary-scan capability. Secondary, the JTAG interface is reused 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 system level, all ICs 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 observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuit Boards by using the four TAP signals only.

The IEEE 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. Initial scanning of the Data Register path will show the ID-Code of the device, since IDCODE is the default JTAG instruction. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-Register. Therefore to avoid damaging the board when issuing the EXTEST instruction for the first time, the merged SAMPLE/PRELOAD should be used for setting initial values to the scan ring. SAMPLE/PRELOAD is also used for taking a non-intrusive snapshot of the external pins during normal operation of the part. The CLAMP 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. 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. Note that the PDICOM instruction has nothing to do with Boundary-scan testing, but represents an alternative way to access internal programming and debugging resources by using the JTAG interface. For more details on PDI, programming and on-chip debug refer to Section 29. ”Program and Debug Interface” on page 344.

The JTAGEN Fuse must be programmed and the JTAGD bit in the MCUCR Register must be cleared to enable the JTAG Interface and Test Access Port.

When using the JTAG interface for Boundary-scan, the JTAG TCK clock frequency can be higher than the internal device frequency. The System Clock in the device is not required for Boundary-scan.

28.3 TAP - Test Access Port

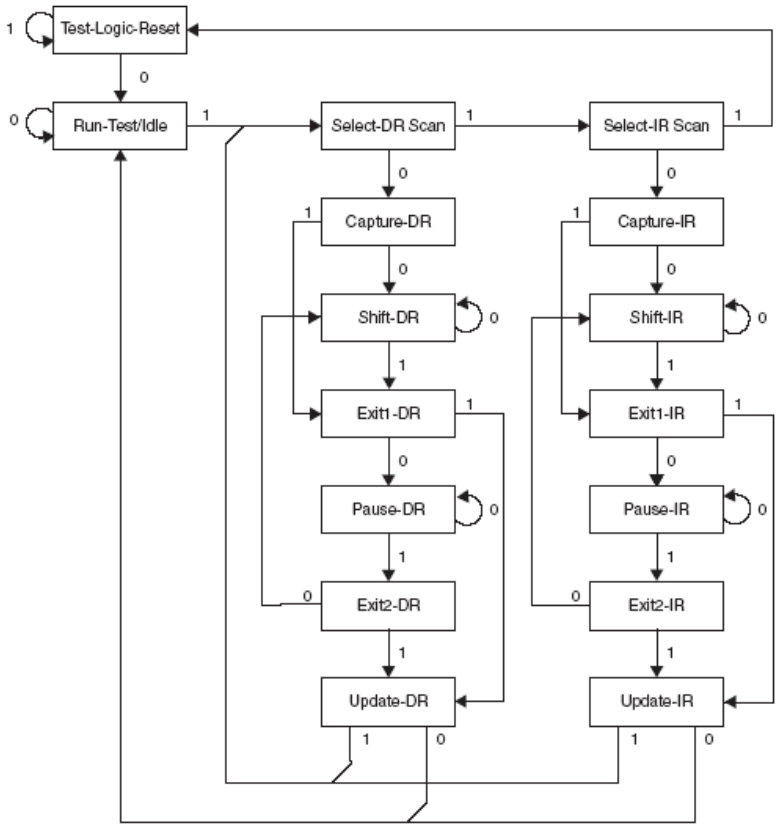
The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port - TAP. These pins are:

- TMS: Test mode select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG 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 Instruction Register or Data Register.

The IEEE std. 1149.1-2001 also specifies an optional TAP signal; TRST - Test ReSeT. This 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 the 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 depicted in Figure 28-1 depend on the signal present on TMS (shown adjacent to each state transition) at a time of the rising edge at TCK. The initial state after a Power-on Reset is Test-Logic-Reset.

Assuming Run-Test/Idle is the present state, 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 - Shift-IR state. While in this state, shift the four bits of the JTAG instructions 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 path between TDI and TDO and controls the circuitry surrounding the selected Data Register.
- Apply the TMS sequence 1, 1, 0 to re-enter 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 only used 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 - 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 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 is shifted out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter 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 only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers.

Note: Independent 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 4-bit wides. 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 behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

28.4.1 EXTEST; 0x1

EXTEST is mandatary and the instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For the I/O port pins, both output control (DIR) and output data (OUT) is controllable via the scan chain, while the output control and actual pin value is observable. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-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 is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

28.4.2 IDCODE; 0x3

IDEOCE is mandatory and the instruction for selecting the 32 bit ID-Register as Data Register. The ID-Register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE Register is 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/RELOAD mandatory and the instruction for pre-loading 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 Data Register. Note that 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 one 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 is applied to the output latches. However, the output latches are not connected to the pins.

28.4.4 BYPASS; 0xf

BYPASS is mandatory and the instruction for selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic "0" into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

28.4.5 CLAMP; 0x4

CLAMP is optional and the instruction for allowing the state of the input/output pins to be determined from the preloaded output latches. The Bypass register is selected as Data Register.

The active states are:

- Capture-DR: Loads a logical "0" into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

28.4.6 HIGHZ; 0x5

HIGHZ is optional and the instruction for putting all outputs in an inactive drive state (e.g. high impedance). The Bypass register is selected as Data Register.

The active states are:

- Capture-DR: Loads a logical "0" into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

28.4.7 PDICOM; 0x7

PDICO is an AVR specific instruction and using the JTAG TAP as an alternative interface towards the PDI (Programming and Debug Interface).

The active states are:

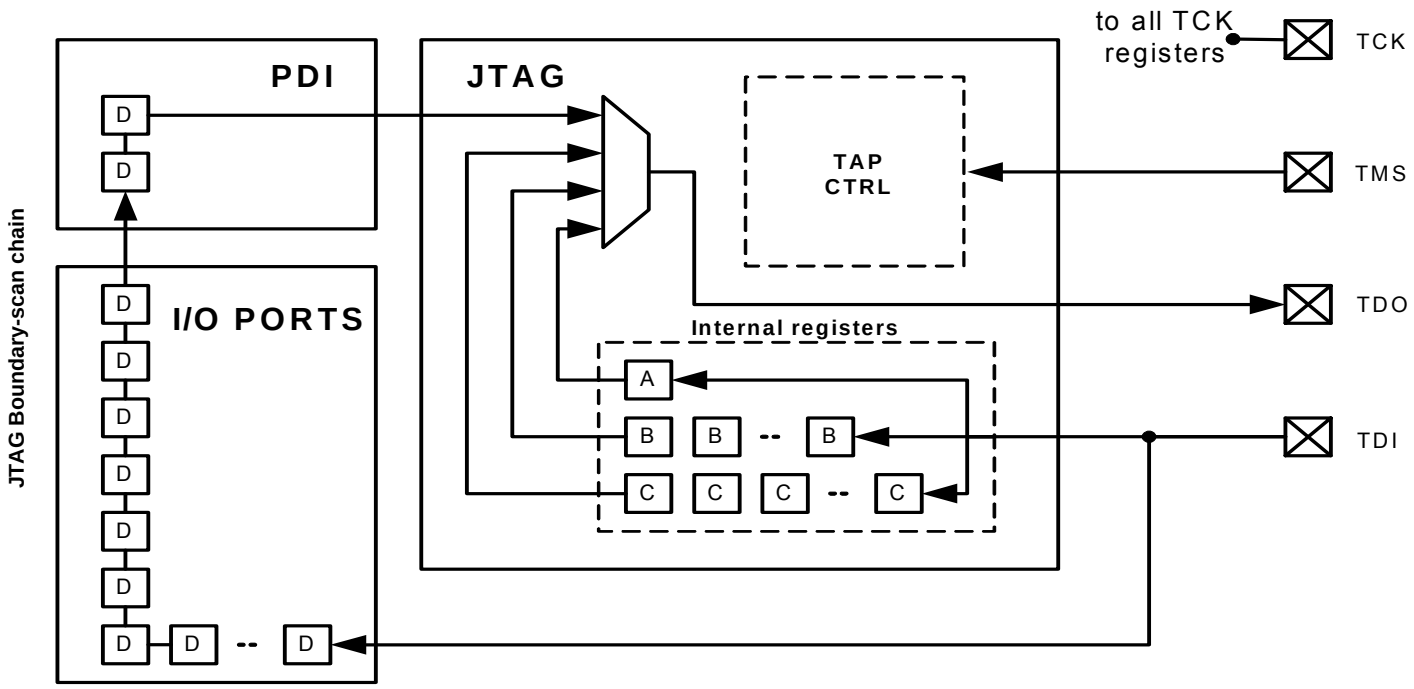
- Capture-DR: Parallel data from the PDI is sampled into the PDICOM data register.
- Shift-DR: The PDICOM data register is shifted by the TCK input.
- Update-DR: Commands or operands are parallel-latched from the PDICOM data register into the PDI.

28.5 Data registers

The supported data registers that can be connected between TDI and TDO are:

- Bypass register (Ref: register A in Figure 28-2 on page 341).
- Device Identification register (Ref: registers C in Figure 28-2 on page 341).
- Boundary-scan chain (Ref: register D in Figure 28-2 on page 341).
- PDICOM data register (Ref: register B in Figure 28-2 on page 341)

Figure 28-2. JTAG data register overview

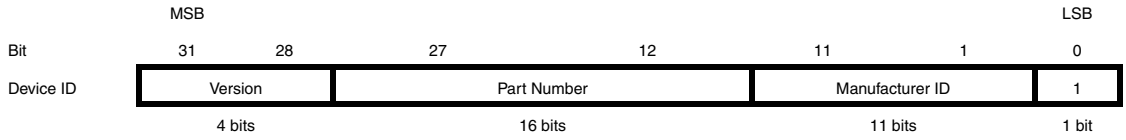


28.5.1 Bypass register

The Bypass Register consists of a single Shift Register stage. When the Bypass Register is selected as 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.

28.5.2 Device Identification register

Figure 28-3. Device Identification register



28.5.2.1 Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1 and so on.

28.5.2.2 Part Number

The part number is a 16-bit code identifying the component. Refer to the device data sheets to find the correct number.

28.5.2.3 Manufacturer ID

The Manufacturer ID is an 11-bit code identifying the manufacturer. For Atmel this code is 11x01F.

28.5.3 Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on all I/O pins. Refer to Section 28.6 ”Boundary-scan chain” on page 342 for a complete description.

28.5.4 PDICOM data register

The PDICOM data register is a 9-bit wide register used for serial-to-parallel - and parallel-to-serial conversion of data between the JTAG TAP and the PDI. For details refer to Section 29. ”Program and Debug Interface” on page 344.

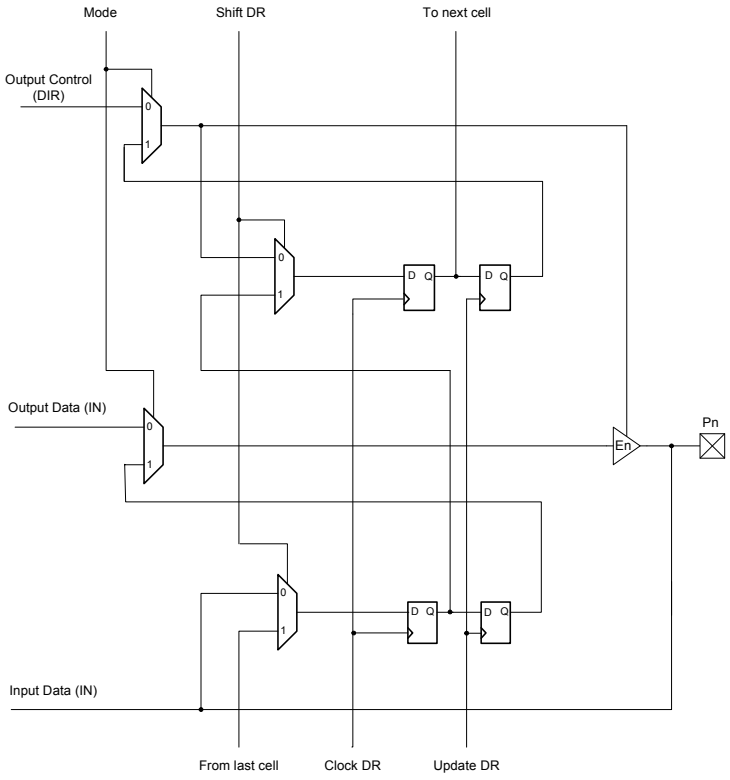
28.6 Boundary-scan chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the I/O pins. To ensure a predictable chip behavior during and after the instructions EXTEST, CLAMP and HIGHZ, the chip is automatically put in reset. During active reset, the external oscillators, analog modules, and non-default port settings (like pull-up/down, bus-keeper, wired-AND/-OR) are disabled. It should be noted that the current chip - and port state is unaffected by the SAMPLE and PRELOAD instructions.

28.6.1 Scanning the Port Pins

Figure 28-4 on page 343 shows the Boundary-scan Cell used for all the bi-directional 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 - synchronizer). Mode represents either an active CLAMP or EXTEST instruction, while ShiftDR is set when the TAP controller is in its Shift-DR state.

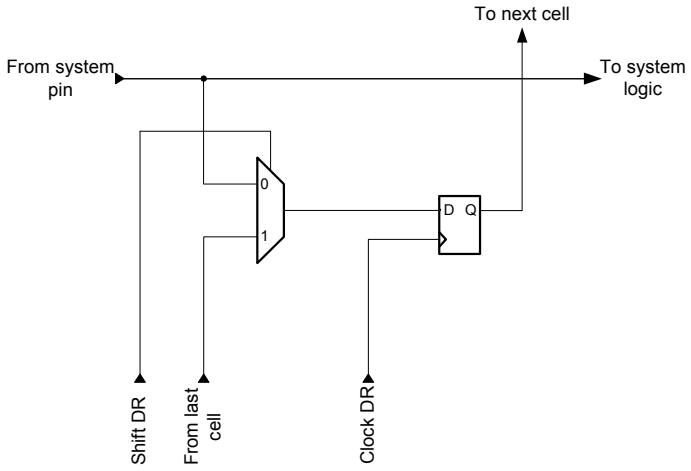
Figure 28-4. Bidirectional boundary scan cell



28.6.2 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 bi-directional, it is only made observable in order to avoid any extra logic on the PDI_DATA output path.

Figure 28-5. An observe-only input cell



29. Program and Debug Interface

29.1 Features

- Program and Debug Interface (PDI)
  - 2-pin interface for external programming and on-chip debugging
  - Uses Reset pin and dedicated Test pin
    - No I/O pins required during programming or debugging
- Programming Features
  - Flexible communication protocol
  - 8 Flexible instructions.
  - Minimal protocol overhead.
  - Fast
    - 10 MHz programming clock at 1.8V V_{CC}
  - Reliable
    - Built in error detection and handling
- Debugging Features
  - Non-Intrusive Operation
    - Uses no hardware or software resource
  - Complete Program Flow Control
    - Symbolic Debugging Support in Hardware
    - Go, Stop, Reset, Step into, Step over, Step out, Run-to-Cursor
  - 1 dedicated program address breakpoint or symbolic breakpoint for AVR studio/emulator
  - 4 Hardware Breakpoints
  - Unlimited Number of User Program Breakpoints
  - Uses CPU for Accessing I/O, Data, and Program
  - High Speed Operation
    - No limitation on system clock frequency
- JTAG Interface
  - JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard
  - Programming features as for PDI
  - On-chip debug features as for PDI

29.2 Overview

The Program and Debug Interface (PDI) is an Atmel proprietary interface for external programming and on-chip debugging of the device.

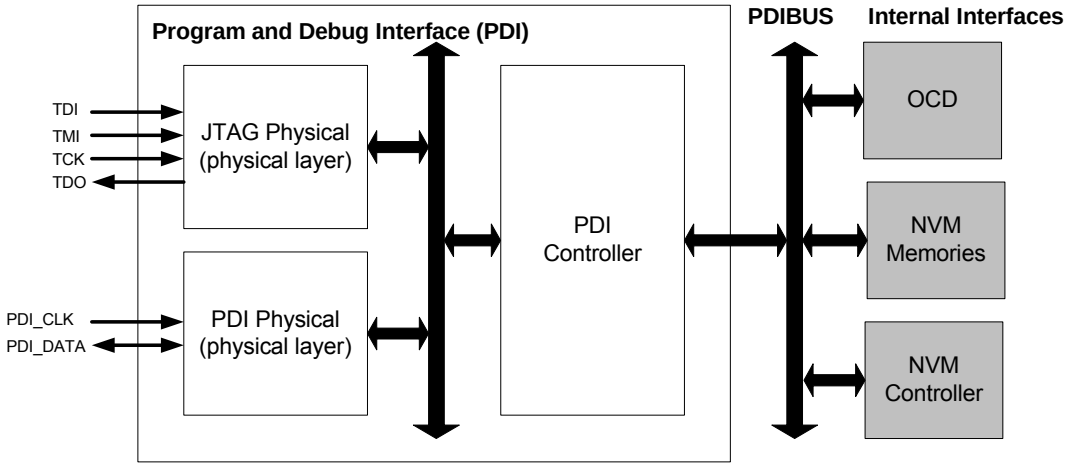
The PDI supports high-speed programming of all Non-Volatile Memory (NVM) spaces; Flash, EEPOM, Fuses, Lockbits and the User Signature Row. This is done by accessing the NVM Controller, and executing NVM Controller commands as described in Memory Programming.

The On-Chip Debug (OCD) system supports fully intrusive operation. During debugging no software or hardware resources in the device is used (except for four I/O pins required if JTAG connection is used). The OCD system has full program flow control, supports unlimited number of program and data breakpoints and has full access (read/write) to all memories.

Both programming and debugging can be done through two physical interfaces. The primary interface is the PDI Physical. This is a 2-pin interface using the Reset pin for the clock input

(PDI_CLK), and the dedicated Test 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 4-pin JTAG interface. The JTAG interface is IEEE std. 1149.1 compliant, and supports boundary scan. Unless otherwise stated, all references to the PDI assumes access through the PDI physical. Any external programmer or on-chip debugger/emulator can be directly connected to these interfaces, and no external components are required.

Figure 29-1. The PDI with JTAG and PDI physical and closely related modules (grey)



29.3 PDI Physical

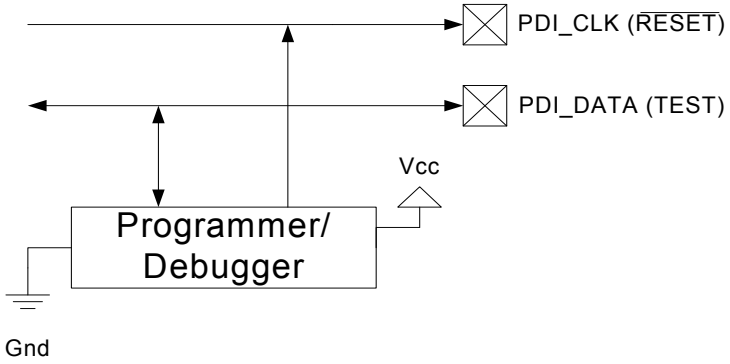
The PDI physical layer handles the basic low-level serial communication. The physical layer uses a bi-directional half-duplex synchronous serial receiver and transmitter (as a USART in USRT mode). The physical layer includes start-of-frame detection, frame error detection, parity generation, parity error detection, and collision detection.

The PDI is accessed through two pins:

- PDI_CLK: PDI clock input (Reset pin).
- PDI_DATA: PDI data input/output (Test pin).

In addition to these two pins, V_{CC} and GND must also be connected between the External Programmer/debugger and the device. Figure 29-2 on page 345 shows a typical connection.

Figure 29-2. PDI connection



The remainder of this section is only intended for third parties developing programming support for XMEGA.

29.3.1 Enabling

The PDI Physical must be enabled before it can be used. 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 data sheet 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.

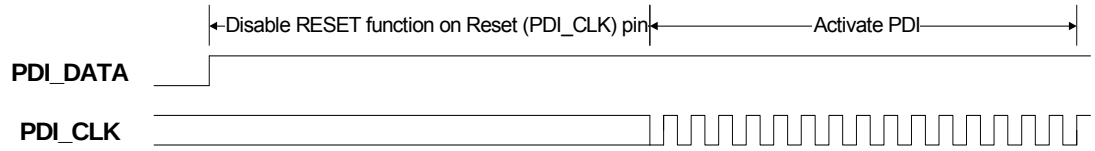
In the next step of the enabling procedure the PDI_DATA line must be kept high for 16 PDI_CLK cycles (16 positive edges detected). The first PDI_CLK cycle must start no later than 100uS after the $\overline{\text{RESET}}$ functionality of the Reset pin was disabled. If this does not occur in time the $\overline{\text{RESET}}$ functionality of the Reset pin is automatically enabled again and the enabling procedure must start over again.

After this the PDI is enabled and ready to receive instructions. The enable sequence is shown in Figure 29-3 on page 346.

The $\overline{\text{RESET}}$ pin is sampled when the PDI interface is enabled. The $\overline{\text{RESET}}$ register is then set according to the state of the $\overline{\text{RESET}}$ pin, preventing the device from running code after the reset-functionality of this pin is disabled.

The PDI_DATA pin has an internal pull-down resistor.

Figure 29-3. Sequence for enabling the PDI.



29.3.2 Disabling

If the clock frequency on the PDI_CLK is lower than approximately 10 kHz, this is regarding as inactivity on the clock line. This will then automatically disable the PDI. If not disabled by fuse, the $\overline{\text{RESET}}$ function on the Reset (PDI_CLK) pin is automatically enabled again. If the time-out occurs during the PDI enabling sequence, the whole sequence must be started from the beginning.

This also means that the minimum programming frequency is approximately 10 kHz.

29.3.3 Frame Format and Characters

The PDI physical layer uses a fixed frame format. A serial frame is defined to be one character of eight data bits with start and stop bits and a parity bit.

Figure 29-4. PDI serial frame format.

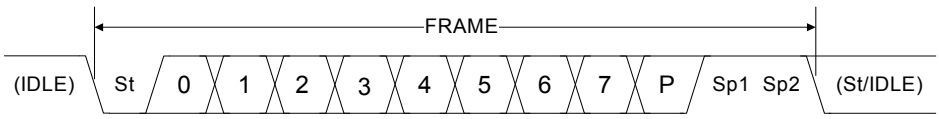


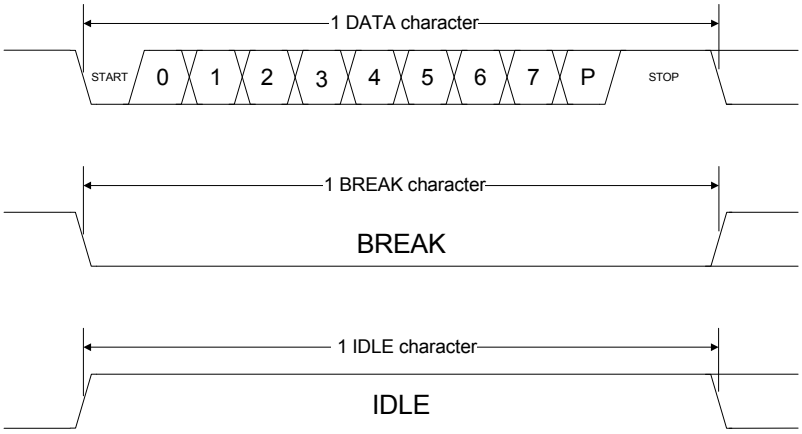
Table 29-1.

St	Start bit, always low.
(0-7)	Data bits (0 to 7)
P	Parity bit, even parity is used
Sp1	Stop bit 1, always high.
Sp2	Stop bit 2, always high.

29.3.3.1 Characters

Three different characters, DATA, BREAK and IDLE, are used. The BREAK character is equal to 12 bit-length of low level. The IDLE character is equal to 12 bit-length of high level. Both the BREAK and the IDLE character can be extended beyond the bit-length of 12.

Figure 29-5. Characters and timing for the PDI Physical.

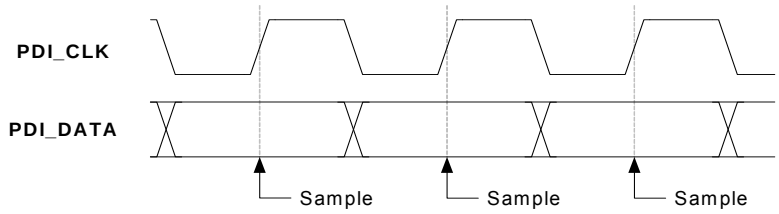


29.3.4 Serial transmission and reception

The PDI physical layer is either in Transmit (TX) or Receive (RX) mode of operation. 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 347, output data (either from the programmer or from the PDI) is always set up (changed) on the falling edge of PDI_CLK, while data is always sampled on the rising edge of PDI_CLK.

Figure 29-6. Changing and sampling of data.



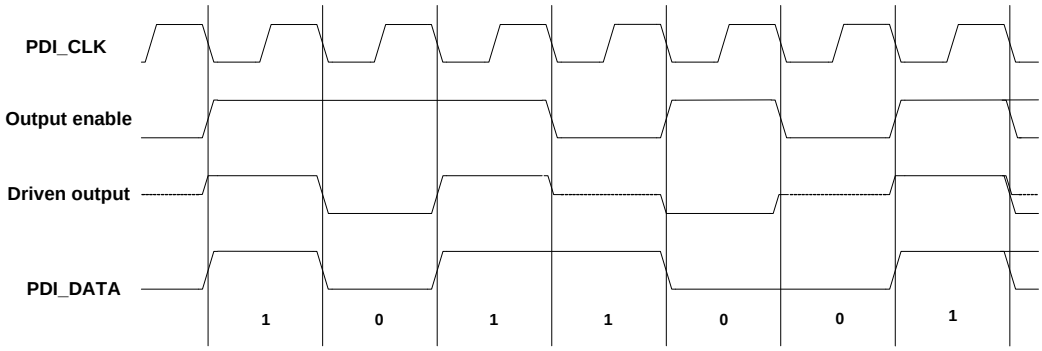
29.3.5 Serial Transmission

When a data transmission is initiated (by the PDI Controller), the transmitter simply shifts the start bit, data bits, the parity bit, and the two stop bits out 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 a RX mode waiting for a BREAK character.

29.3.5.1 Drive contention and collision detection

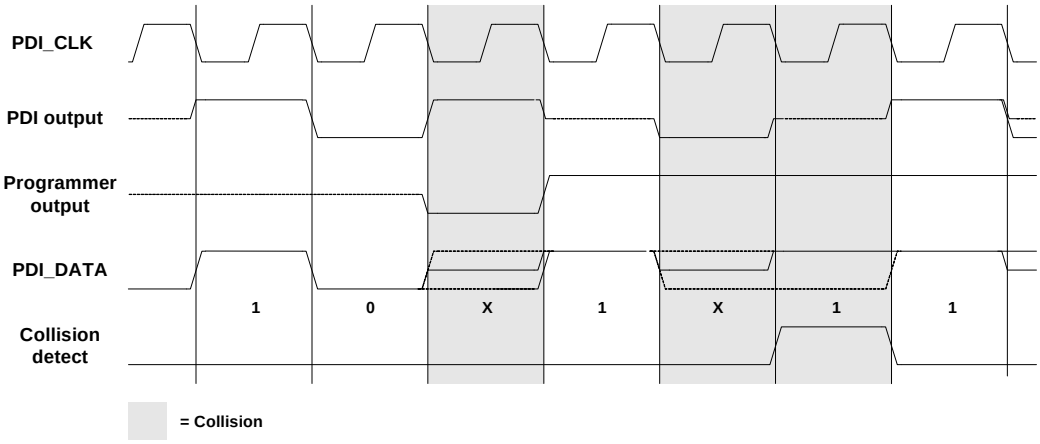
In order to reduce the effect of a drive contention (the PDI and the programmer drives the PDI_DATA line at the same time), a mechanism for collision detection is supported. The mechanism is based on the way the PDI drives data out on the PDI_DATA line. As shown in Figure 7, the output pin driver is only active 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 only actively driven the first clock cycle. After this point the output driver is automatically tri-stated, and the PDI_DATA pin has a bus-keeper responsible for keeping the pin-value unchanged until the output driver is re-enabled due to a bit value change.

Figure 29-7. Driving data out on the PDI_DATA using bus-keeper



If the programmer and the PDI both drives the PDI_DATA line at the same time, the situation of drive contention will occur as illustrated in Figure 29-8 on page 349. 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 than what the PDI expects, a collision is detected.

Figure 29-8. 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 output driver will be active all the time preventing polling of the PDI_DATA line. However, within a single frame the two stop bits should always be transmitted as ones, enabling collision detection at least once per frame.

29.3.6 Serial Reception

When a start bit is detected, the receiver starts to collect the eight data bits and shift them into the shift register. 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 detected, the received data bits are parallelized and made available for the PDI controller.

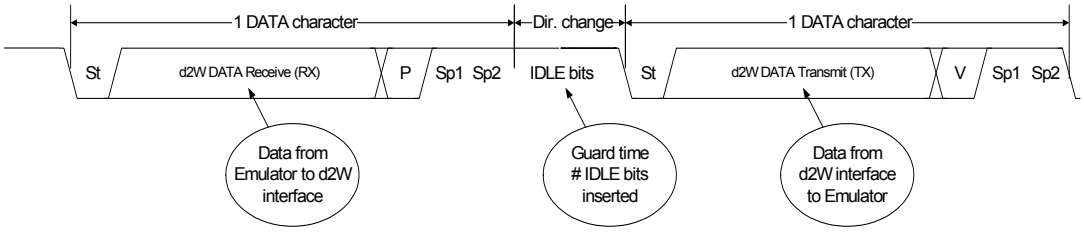
29.3.6.1 BREAK detector

When the PDI is in TX-mode, a BREAK character signalized by the programmer will not be interpreted as a BREAK, but 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 (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.

29.3.7 Direction Change

In order to ensure correct timing of the half-duplex operation, a simple Guard Time mechanism is added to the PDI physical interface during direction change. When the PDI changes from operating in RX-mode to operate in TX-mode, a configurable number of additional 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. Writing the Guard Time bits in the PDI Controller's Control Register specifies the additional Guard Time. The default Guard Time value is +128 bits.

Figure 29-9. PDI direction change by inserting IDLE bits



The programmer will loose control of the PDI_DATA line at the point where the PDI target 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, minimum a single IDLE bit should be inserted before the start bit is transmitted.

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 more details refer to ”IEEE 1149.1 JTAG Boundary Scan interface” on page 337.

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. By default the JTAGEN fuse is programmed, and the JTAG interface is enabled. When the JTAG instruction PDICOM is shifted into the JTAG instruction register, the PDI communication register is chosen as the data register connected between TDI and TDO. In this mode, the JTAG interface can be used to access the PDI for external programming and on-chip debug.

29.4.2 Disabling

The JTAG interface can be disabled by either 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 XMEGA JTAG Instruction set consist of eight instructions related to Boundary Scan and PDI access for NVM programming, for details on the instruction set refer to ”JTAG instructions” on page 339.

29.4.3.1 The PDICOM instruction

The 9-bit PDI communication register is selected as data register. Commands are shifted into the register, while results from previous commands are shifted out from the register. The active TAP-controller states are:

- 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

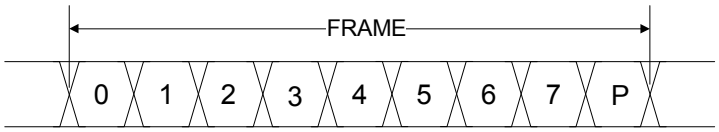


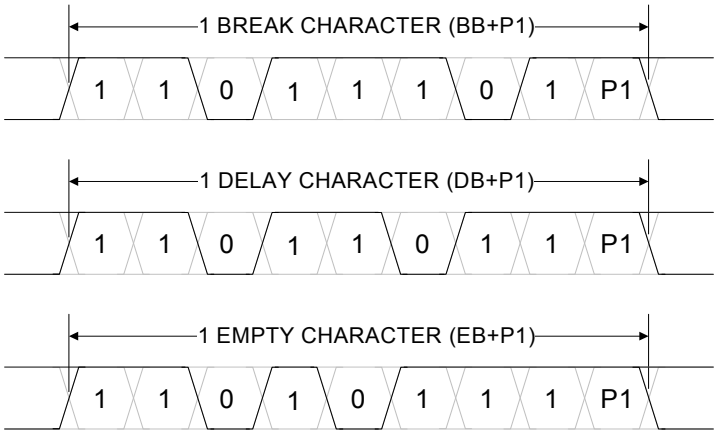
Table 29-2.

(0-7)	Data/command bits, least significant bit sent first (0 to 7)
P	Parity bit, even parity is used

29.4.4.1 Special data characters

Three data characters are given a special meaning. Common for all three characters is that the Parity bit is inverted in order to force parity error upon reception. The BREAK character (0xBB+P1) is used by the External Programmer to force the PDI to abort any on-going 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 programmer that it has no transmission pending (i.e. the PDI is in RX-mode).

Figure 29-11. Special data characters

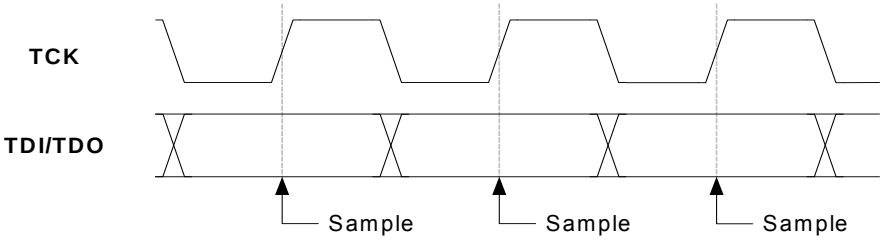


29.4.5 Serial transmission and reception

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. Dictated by the PDI Controller, the JTAG physical layer operates 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 29-10 on page 351, TDI and TDO is always set up (changed) 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 in parallel into the shift register, and then serialized by shifting the byte out on TDO. The Parity bit is generated and stitched to the data byte during transmission. The transmission speed is dictated by the TCK signal.

29.4.6.1 Status signalling

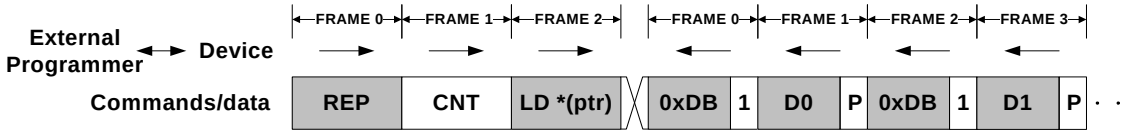
If the PDI is in TX-mode (as a response to an LD-instruction), and a transmission request from the PDI Controller is pending when the TAP-controller enters the Capture-DR state, valid data will be parallel-loaded into the shift-register and a correct Parity bit will be generated and transmitted along 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 (0xEB) will be parallel-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 yet pending when the TAP-controller enters the Capture-DR state, a DELAY byte (0xDB) will be parallel-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 352 shows an uninterrupted flow of data frames from the PDI (Device) as a response to the repeated indirect LD instruction. However, in this example the device is not able to return data bytes faster than one valid byte per two transmitted frames, intermediate DELAY characters are inserted.

Figure 29-13. Date 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 yet ready for transmission in the previous DR-Capture state. The proper reaction from the programmer is to 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 receiver 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 a parallel way in the Update-DR state.

29.4.7.1 Parity checker

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 signalized.

29.4.7.2 BREAK detector

The Parity checker is active in both TX- and RX-mode. If a parity error is detected, the received data byte is evaluated and compared with the BREAK character (which always will generate a parity error). In case the BREAK character is recognized, the PDI Controller is signalized.

29.5 PDI Controller

The PDI Controller includes 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 a 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 in a way 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 target), or it may involve data to be returned back to the programmer (e.g. a data byte is read from a location within the target).

29.5.1 Switching between PDI- and JTAG-mode

The PDI Controller uses either the JTAG - or the PDI physical layer for establishing a connection to the programmer. Based on this, the PDI is said to be in either JTAG or PDI mode. When one of the modes are entered, the PDI Controller registers will be initialized, and the correct clock source is selected by the clock system. It should be noted that 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. Still, if by some reason a user wants to switch physical layer without power on/off the device, the active layer should be disabled (to trigger a reset of the PDI) 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 NVM memories for programming, a unique key must be signalized by using the KEY instruction. The internal interfaces is accessed as one linear address space using a dedicated bus (PDIBUS) between the PDI and the internal interfaces. The PDIBUS address space is the one shown in Figure 30-4 on page 380. The NVM controller must be enabled for the PDI controller to have any access to the NVMs. The PDI controller can only access the NVM and NVM controller in programming mode. The PDI controller does not ever 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 depends 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, these exceptions are defined as:

- 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, these exceptions are defined:

- 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.

All exceptions are signalized to the PDI Controller. All on-going operations are then aborted and the PDI is put in the ERROR state. The PDI will remain in this 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 instructions set that is used for all access to the PDI itself and to the internal interfaces.All instructions are byte instructions. Most of the instructions require a number of byte operands following the instruction. The instructions allow to external programmer to access the PDI Controller, the NVM Controller and the NVM 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 serial 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-bytes address - and data access. Four different

address/data sizes are supported; byte, word, 3 bytes, and long (4 bytes). It should be noted that multiple-bytes access is internally broken down to repeated single-byte accesses. The main advantage with the multiple-bytes access is that it gives a way to reduce the protocol overhead. When using the LDS, 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 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 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 address - and data access. Four different address/data sizes are supported; byte, word, 3 bytes, and long (4 bytes). It should be noted that multiple-bytes access is internally broken down to repeated single-byte accesses. The main advantage with the multiple-bytes access is that it gives a way to reduce the protocol overhead. When using the STS, 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 to 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 into the Pointer register prior to the data access. Indirect addressing can be combined with pointer increment. In addition to read data from the PDIBUS Data Space, the Pointer register can be read by the LD instruction. Even though the protocol is based on byte-wise communication, the LD instruction supports multiple-bytes address - and data access. Four different address/data sizes are supported; byte, word, 3 bytes, and long (4 bytes). It should be noted that multiple-bytes access is internally broken down to repeated single-byte accesses. The main advantage with the multiple-bytes access is that it gives a way to reduce 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 into the Pointer register prior to the data access. Indirect addressing can be combined with pointer increment. In addition to write data to the PDIBUS Data Space, the Pointer register can be written by the ST instruction. 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). It should be noted that multiple-bytes access is internally broken down to repeated single-byte accesses. The main advantage with the multiple-bytes access is that it gives a way to reduce 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 to 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 is 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 that is required for activating the NVM interfaces.

29.5.6.8 *REPEAT - Set Instruction Repeat Counter*

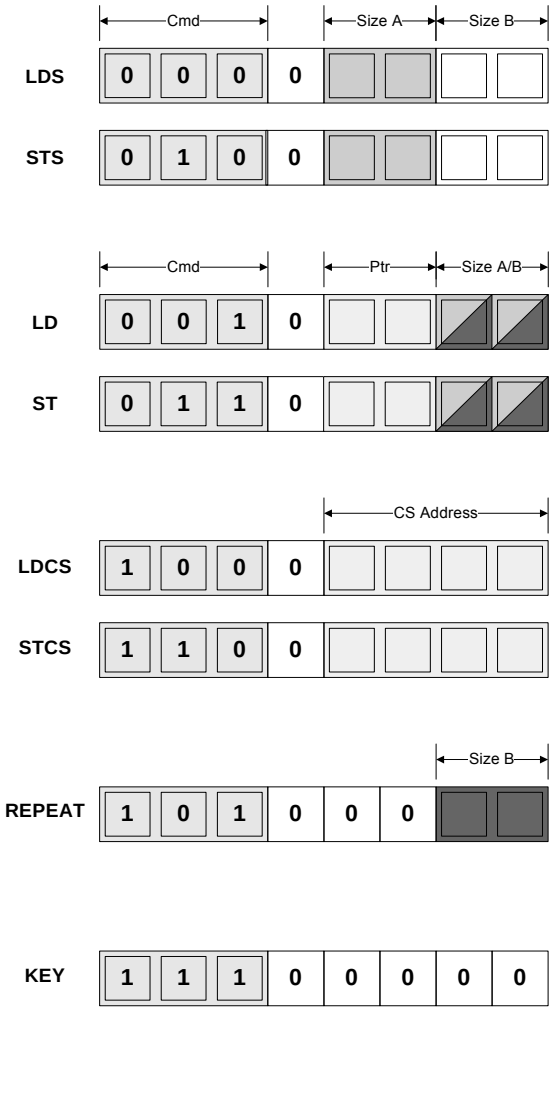
The REPEAT instruction is used to store count values that are serially shifted into the physical layer shift-register to the Repeat Counter Register. The instruction that is loaded directly after the REPEAT instruction operand(s) will be repeated a number of times according to the specified Repeat Counter 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 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 357.

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

These registers are all internal registers that are involved in instruction decoding or PDIBUS addressing. None of these registers are accessible as register 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 specifying 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 the 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 byte while the MSB 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 will always be 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 following immediately after the REPEAT instruction and its operands, the Repeat Counter register is decremented until it reaches zero, indicating that all repetitions are completed. The repeat counter is also involved in key reception.

29.6.4 Operand Count Register

Immediately after and instruction (except the LDCS and the 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 that have been transferred.

29.7 Register Description - PDI Control and Status Register

These register are registers that are accessible in the PDI Control and Status Register Space (CSRS) using the instructions LDCS and STCS. The CSRS is allocated for registers directly involved in configuration and status monitoring of the PDI itself.

29.7.1 STATUS - Program and Debug Interface Status Register

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	-	-	-	NVMEN	-	STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:2 - Reserved Bits**

These bits are reserved and will always be read as zero. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1- NVMEN: Non-Volatile 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 Bit**

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.

29.7.2 RESET - Program and Debug Interface Reset register

Bit	7	6	5	4	3	2	1	0	
+0x01	RESET[7:0]								CTRLB
Read/Write	R/W	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 (0x00 is recommended). Reading the least LSB bit the will return the status of the RESET. The 7 MSB bits will always return the value 0x00 regardless of whether the device is in reset or not.

29.7.3 CTRL - Program and Debug Interface Control Register

Bit	7	6	5	4	3	2	1	0	
+0x02	-	-	-	-	-	GUARDTIME[2:0]			CTRL
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 additional IDLE bits of Guard Time that are inserted in between PDI reception and - transmission direction change. The default Guard Time is 128 IDLE bits, and the available settings is shown in Table 29-3 on page 360. In order to speed up the communication, the Guard Time should be set to the lowest safe configuration accepted. It should be noted that no Guard Time is inserted when switching from TX - to RX mode.

Table 29-3. Guard Time settings.

GUARDTIME	Number of IDLE bits
000	+128
001	+64
010	+32
011	+16
100	+8
101	+4
110	+2
111	+0

29.8 Register Summary

Address	Name	Address	Name	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	-	-	-	-	-	-	NVMEN	-	359
+0x01	RESET	RESET[7:]								359
+0x02	CTRL	-	-	-	-	-	GUARDTIME[2:0]			359
+0x03	Reserved									
+0x04	Reserved									
+0x05	Reserved									
+0x06	Reserved									
+0x07	Reserved									
+0x08	Reserved									
+0x09	Reserved									
+0x0A	Reserved									
+0x0B	Reserved									
+0x0C	Reserved									
+0x0D	Reserved									
+0x0E	Reserved									
+0x0F	Reserved									
+0x10	Reserved									

30. Memory Programming

30.1 Features

- Read and Write access to all memory spaces from
  - External programmers
  - Application Software
- Self-Programming and Boot Loader Support
  - Real Read-While-Write Self-Programming
  - The 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
  - Fast and reliable interfaces.
- 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
- Code Efficient Algorithm
- Efficient Read-Modify-Write Support

30.2 Overview

This section describes how to program the Non Volatile Memory (NVM) in XMEGA, and covers both self-programming and external programming. The NVM consist of the Flash Program Memory, User Signature and Calibration 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 18.

The NVM can be accessed for read and write both from application software through self-programming and from an external programmer. For both external programming and self-programming access to the NVM is done through the common NVM Controller, and the two methods of programming are very similar. Memory access is done by loading address and/or data into the NVM, and a set of commands and triggers that make the NVM Controller perform specific tasks on the NVM.

From external programming all memory spaces can be read and written, expect for the Calibration 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 379 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 Calibration 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 367 describes this in detail.

For both self-programming and external programming it is possible to run an automatic CRC check on the Flash or a section of the Flash to verify its content.

The device can be locked to prevent read and/or write 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

All access to the Non Volatile Memories is done through the NVM Controller. This controls all NVM timing and access privileges, and hold the status of the NVM. This is the common NVM interface for both the external programming and self-programming. For more details on the NVM Controller refer to ”Register Description” on page 385.

30.4 NVM Commands

The NVM Controller has a set of commands that decide the task to perform on the NVM. This is issued to the NVM Controller 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 is setup for the operation, each command has a trigger that will start the operation. Bases on the 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 memory 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 NVN is written, and this is typically used for NVM write operations.

30.4.4 CCP 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 12

30.5 NVM Controller Busy

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 operation finished before a new can start. The External programmer or application software must ensure that the NVM is not addressed while 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 read from external programmers.
- All NVM read from the Application Section.

During Self-Programming interrupts must be disabled, or the Interrupt Vector table should be moved to the Boot Loader Sections as described in ”Interrupts and Programmable Multi-level Interrupt Controller” on page 123.

30.6 Flash and EEPROM Page Buffers

The Flash memory is updated in a page-by-page fashion. The EEPROM can be updated both in a byte-by-byte and page-by-page fashion. 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. For both Flash and EEPROM, page erase and write each requires 4 ms.

The size of the page buffers depend on the Flash and EEPROM size in each device, and details on page size and page number is described in each device data sheet.

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. If an already loaded location is written again, this will corrupt the content of that Flash page buffer location.

Flash page buffer Locations that are not loaded will have the value 0xFFFF, and this value will then be programmed into the flash page locations.

The Flash Page Buffer is automatically erased after:

- A system 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. If an already loaded location is written twice, this will corrupt the content of that EEPROM page buffer location.

EEPROM page buffer locations that are loaded will get tagged by the NVM Controller. During a page write or page erase, only target locations will be written or erased. Locations that are not target, will not be written or erased, and the corresponding EEPROM location will remain unchanged. This means that also before an EEPROM page erase, data must be loaded to the selected page buffer location to tag them. If the data in the page buffer is not going to be written afterword, the actual values in the buffer does 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 Flash and EEPROM page programming, filling the page buffers and writing the page buffer into Flash or EEPROM is two separate operations. The sequence of this is the 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 the content in the flash Page.

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 buffer before a split Page Erase and Page Write:

- Fill the Flash Page Buffer.
- Perform a Flash Page Erase.
- Perform a Flash Page Write.

Alternative 2, fill the page buffer before an atomic Page Erase and Write:

- Fill the Flash Page Buffer.
- Perform a 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 enables 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 selected number of data bytes stored in the EEPROM page buffer, the selected locations in the EEPROM page must be erased. Programming an un-erased EEPROM page will corrupt the content in the EEPROM page. The EEPROM page buffer must be loaded before any Page Erase or Page Write operations:

Alternative 1, fill the page buffer before a Page Erase:

- 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 buffer before a Page Erase and Write:

- Fill the EEPROM page buffer with the selected number of bytes.
- Perform an 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 - Non-Volatile Memory 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 read or write can be corrupt as supply voltage is too low for the CPU and the Flash to operate properly.

30.9.1 Write Corruption

To ensure that the voltage is correct during a complete write sequence to the Flash memory, the BOD is automatically enabled by hardware when the write sequence starts. If a BOD reset occurs, 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.

30.9.2 Read Corruption

The NVM can be read incorrectly if the supply voltage is too low so the CPU execute instructions incorrectly. To ensure that this does not happen the BOD can be enabled.

30.10 CRC Functionality

It is possible to run an automatic Cyclic Redundancy Check (CRC) on the Flash Program Memory. This can be issued from external programming or software to do a CRC on the Application Section, Boot Loader Section or a selected address range of the Flash.

Once the CRC is started, the CPU will be halted until the CRC is done and the checksum is available in the NVM Data Register. The CRC takes one CPU Clock cycle per word that is included in the CRC address range.

The polynomial that is used for CRC is fixed, and this is: $x^{24} + 4x^3 + 3x + 1$.

30.11 Self-Programming and Boot Loader Support

Both the EEPROM and the Flash memory can be read and written from the application software in the device. This 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 both the Application Section can read from the Flash, User Signature Row, Calibration 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 downloading and uploading 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 code if the feature is not needed anymore.

30.11.1.1 Application and Boot Loader sections

The Application and Boot Loader sections are different when it comes to self-programming. The Application Section is Read-While-Write (RWW) while the Boot Loader Section is No Read-While-Write (NRWW). Here “Read-While-Write” refers to the section being programmed (erased or written), not the section being read during a Boot Loader software update. Whether the CPU can continue to run and execute code or is halted to stop program execution during a Boot Loader software update is depending on the Flash address being programmed:

- When erasing or writing a page located inside the Application Section (RWW), the Boot Loader Section (NRWW) can be read during the operation, thus the CPU can run and execute code from the Boot Loader Section (NRWW).
- When erasing or writing a page located inside the Boot Loader Section (NRWW), the CPU is halted during the entire operation and code cannot execute.

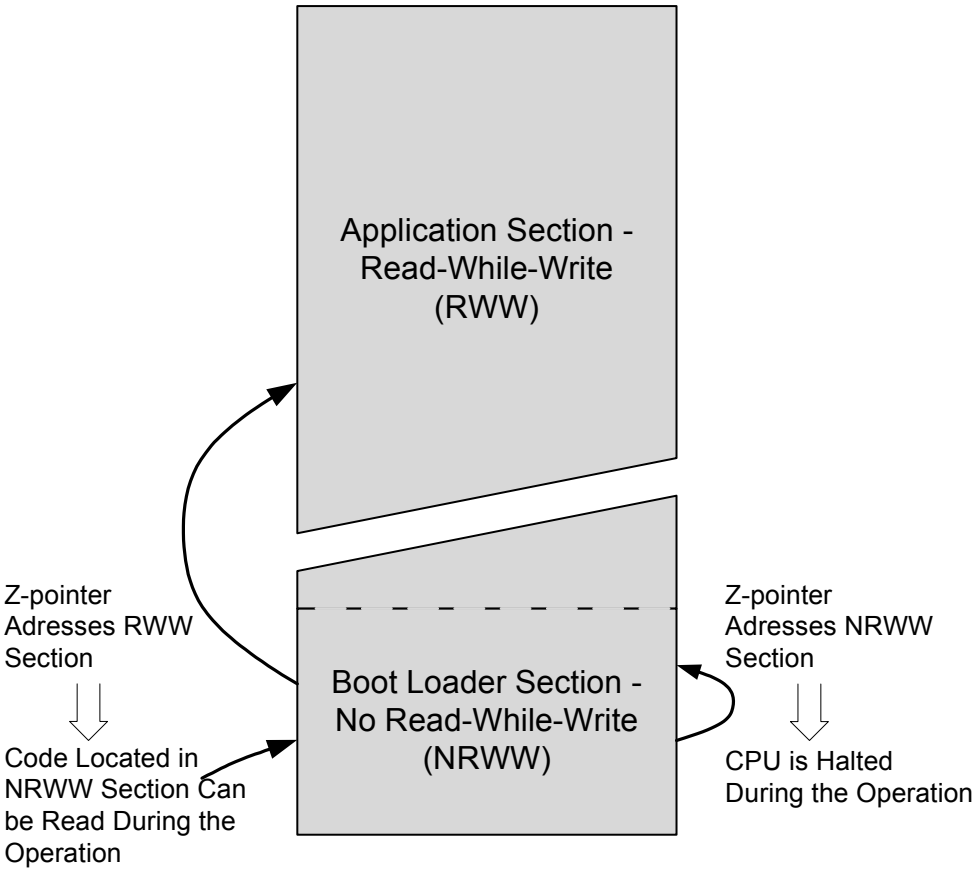
The User Signature Row section is NRWW, hence erasing or writing this section has the same properties as for the Boot Loader Section.

During an on-going programming, the software must ensure that the Application Section is not accessed. Doing this will halt the program execution from the CPU. The user software can not read data located in the Application Section during a Boot Loader software operation.

Table 30-1. Summary of RWW and NRWW functionality

Section being addressed by Z-pointer during the programming?	Section that can be read during programming	CPU Halted	Read-While-Write Supported
Application Section (RWW)	Boot Loader Section (NRWW)	No	Yes
Boot Loader Section (NRWW)	None	Yes	No
User Signature Row section (NRWW)	None	Yes	No

Figure 30-1. Read-While-Write vs. No Read-While-Write



30.11.1.2 Addressing the Flash

The Z-pointer is used to hold the Flash memory address for read and write access. The Z pointer consists of the ZL and ZH registers in the register file, and RAMPZ Register for devices with more than 64K bytes for Flash memory. For more details on the Z-pointer refer to "The X-, Y- and Z- Registers" on page 10.

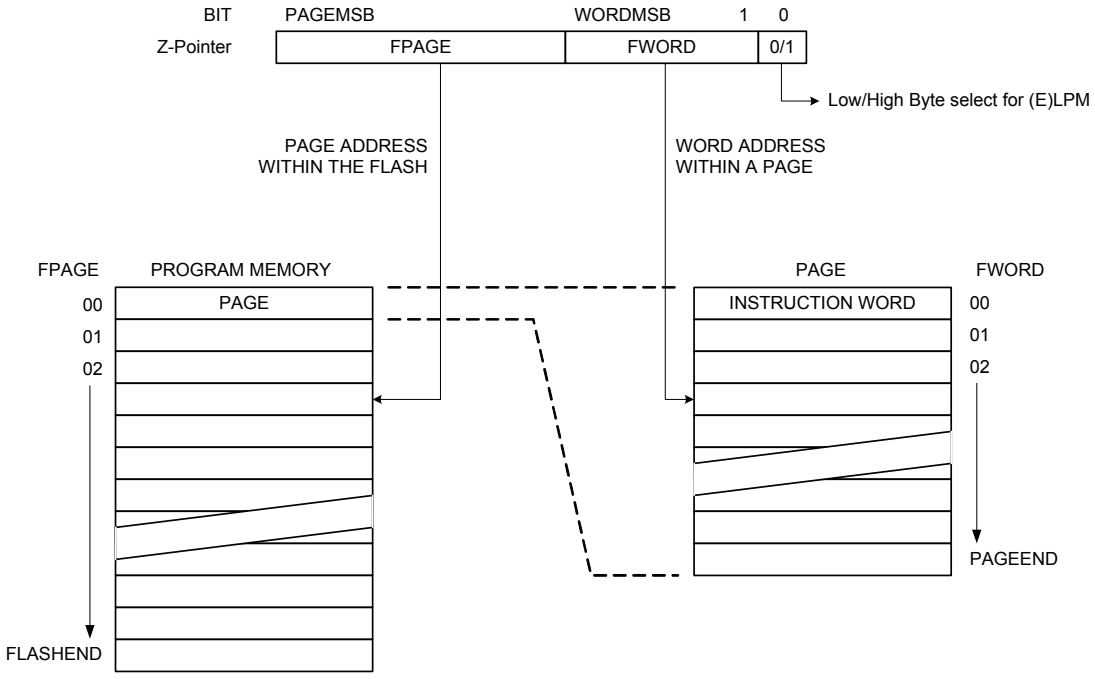
Since the Flash is word accessed and organized in pages, the Z-pointer can be treated as having two sections. The least significant bits address the words within a page, while the most significant bits address the page within the Flash. This is shown in Figure 30-2 on page 369. The word address in the page (FWORD) is held by the bits [WORDMSB:1] in the Z-pointer. The remaining bits [PAGEMSB:WORDMSB+1] in the Z-pointer holds 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 LMP), 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 data sheet for details on this.

Once a programming operation is initiated, the address is latched and the Z-pointer can be updated and used for other operations.

Figure 30-2. 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 Calibration 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 is triggered by a executing the SPM instruction (SPM).

The Change Protected column indicate if the trigger is protected by the Configuration Change Protection (CCP). This is a special sequence to write/execute the trigger during self-programming, for more details refer to ”CCP - Configuration Change Protection Register” on page 13. CCP is not required for external programming. The two last columns shows the address pointer used for addressing, and the source/destination data register.

Section 30.11.1.1 on page 367 through Section 30.11.2.14 on page 374 explain in details 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 & 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 & 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 & 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	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	-	-
Calibration Row								
0x02	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.

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, and this is not a valid AVR CPU opcode/instruction.

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 PCPAGE. 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 PCPAGE. 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 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 PCPAGE. 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 & Write Application Section / Boot Loader Section Page

The Erase & Write Application Section Page and Erase & 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 PCPAGE. Other bits in the Z-pointer will be ignored during this operation.
2. Load the NVM CMD register with the Erase & 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 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 & Write Application Section command requires that the Z-pointer addresses the Application section, and the Erase & 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 / Calibration Row*

The Read User Signature Row and Red Calibration Row commands are used to read one byte from the User Signature Row or 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 / Calibration Row command
3. Execute the LPM instruction.

The destination register will be loaded during the execution of the LPM instruction.

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 is triggered by a executing the SPM instruction (SPM).

The Change Protected column indicate if the trigger is protected by the Configuration Change Protection (CCP) during self-programming. The two last columns shows the address pointer used for addressing, and the source/destination data register.

Section 30.11.3.1 on page 374 through Section 30.11.3.2 on page 374 explain in details 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	Address pointer	Data register	NVM Busy
0x00	NO_OPERATION	No Operation	-	-	-	-	-	-
Fuses and Lock Bits								
0x07	READ_FUSES	Read Fuses	CMDEX	N	N	ADDR	DATA	Y
0x08	WRITE_LOCK_BITS	Write Lock Bits	CMDEX	N	Y	ADDR	-	Y

30.11.3.1 *Write Lock Bits Write*

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 Buffer is 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 registers with the address to 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 read.

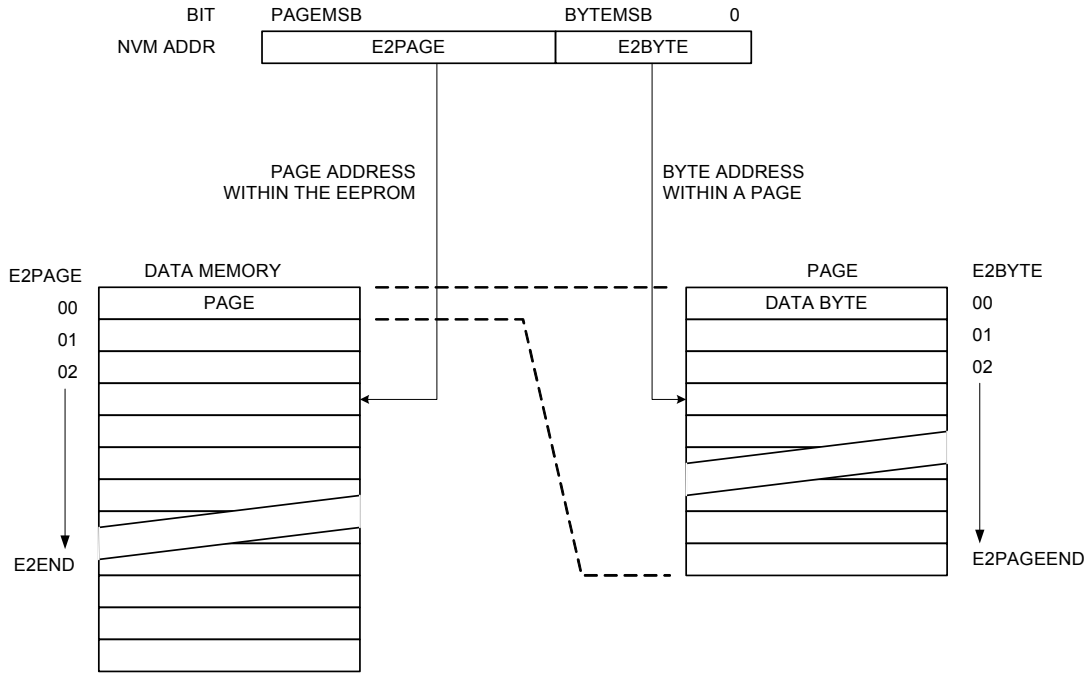
30.11.4.1 Addressing the EEPROM

The EEPROM can be accessed through the NVM controller (I/O mapped), similar to 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 section. 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 30-3 on page 376. The byte address in the page (E2BYTE) is held by the bits [1:BYTEMSB] in the ADDR register. The remaining bits [PAGEMSB:BYTEMSB+1] in the ADDR register holds the EEPROM page address (E2PAGE). Together E2BYTE and E2PAGE holds 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 data sheet for details on this.

Figure 30-3. 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 3 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 is 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 reading the NVM DATA0 register (DATA0).

The Change Protected column indicate if the trigger is protected by the Configuration Change Protection (CCP) during self-programming. CCP is not required for external programming. The two last columns shows the address pointer used for addressing, and the source/destination data register.

Section 30.11.5.1 on page 377 through Section 30.11.5.7 on page 378 explains in details the algorithm for each EEPROM operation.

Table 30-4. EEPROM Self-Programming Commands

CMD[6:0]	Group Configuration	Description	Trigger	CPU Halted	Change Protected	Address pointer	Data register	NVM Busy
0x00	NO_OPERATION	No Operation	-	-	-	-	-	-
EEPROM Page buffer								
0x33	LOAD_EEPROM_BUFFER	Load EEPROM Page Buffer	DATA0	N	Y	ADDR	DATA0	N
0x36	ERASE_EEPROM _BUFFER	Erase EEPROM Page Buffer	CMDEX	N	Y	-	-	Y
EEPROM								Y
0x32	ERASE_EEPROM_PAGE	Erase EEPROM Page	CMDEX	N	Y	ADDR	-	Y
0x34	WRITE_EEPROM_PAGE	Write EEPROM Page	CMDEX	N	Y	ADDR	-	Y
0x35	ERASE_WRITE_EEPROM_PAGE	Erase & Write EEPROM Page	CMDEX	N	Y	ADDR	-	Y
0x30	ERASE_EEPROM	Erase EEPROM	CMDEX	N	Y	-	-	Y
0x06	READ_EEPROM	Read EEPROM	CMDEX	N	Y	ADDR	DATA0	N

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 2-3 until for the arbitrary number of bytes to be loaded into the page buffer.

30.11.5.2 Erase EEPROM Page Buffer

The Erase EEPROM 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 EPPROM Page Erase

The Erase EEPROM Erase command is used to erase one EEPROM page.

1. Set up the NVM CMD register to Erase EEPROM Page command.
2. Load the NVM ADDRESS register with 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 correspond with the loaded and tagged locations in the EEPROM page buffer.

30.11.5.4 Write EEPROM Page

The Write EEPROM Page command is used to write all locations that is 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 for EEPROM page to write.
3. Set the CMDEX bit in 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 & Write EEPROM Page

The Erase & Write EEPROM Page command is used to first erase an EEPROM page and write the EEPROM page buffer into that page in EEPROM, in one atomic operation.

1. Load the NVM CMD register with the Erase & Write EEPROM Page command.
2. Load the NVM ADDR register with the address for EEPROM page to write.
3. Set the CMDEX bit in 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 the locations in all EEPROM pages that corresponds the loaded and tagged locations in the EEPROM page buffer.

1. Set up the NVM CMD register to Erase EPPROM command.
2. Set the CMDEX bit in 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 EPPROM

The Read EEPROM command is used to read one byte from the EEPROM,

1. Load the NVM CMD register with the Read EPPROM command.
2. Load the NVM ADDR register with the address to read.
3. Set the CMDEX bit in NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The data byte read will be available in the NVM DATA0.

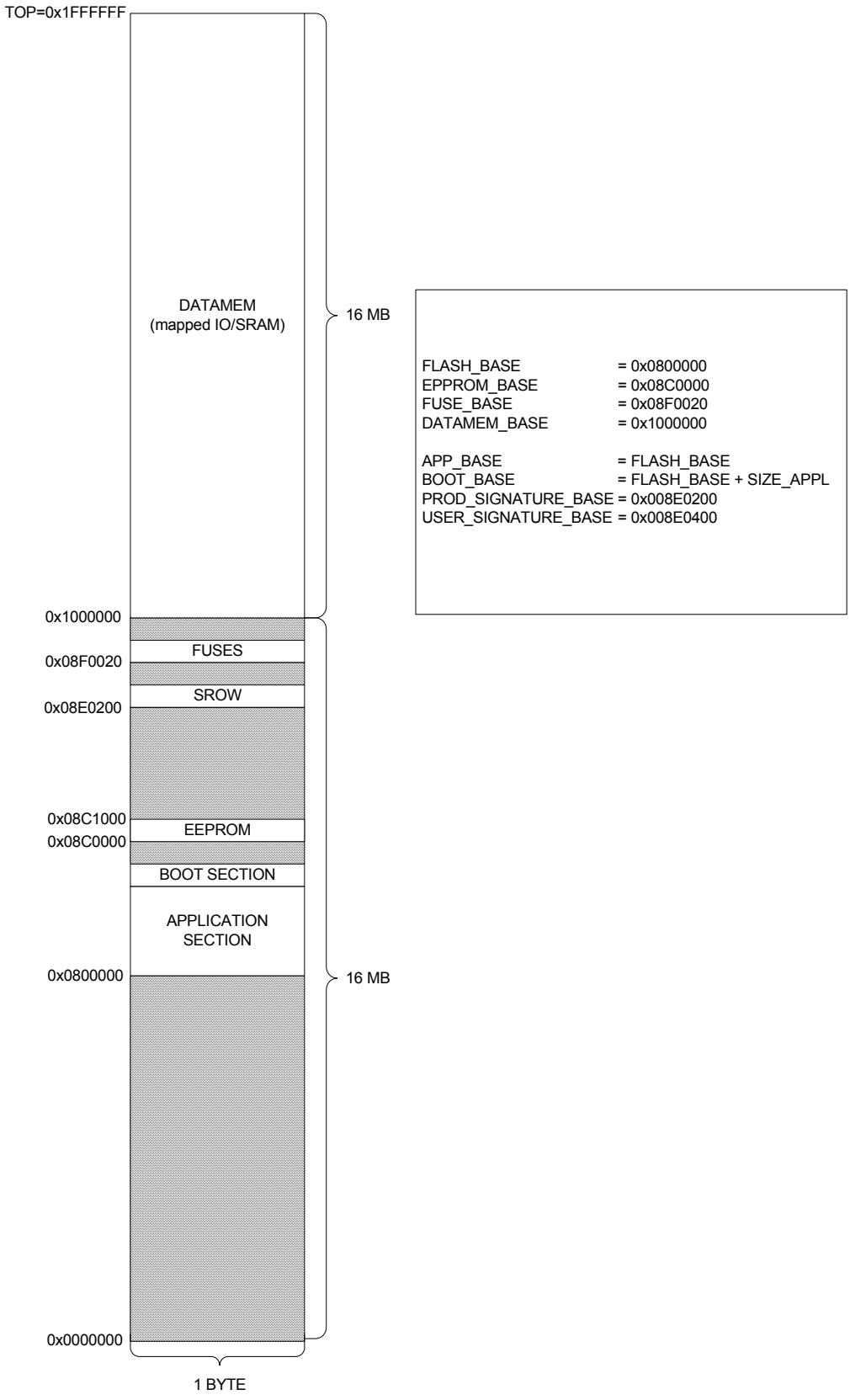
30.12 External Programming

External Programming is the method for programming non volatile code and data into the device from an external programmer or debugger. This can be done both in-system (In-System Programming) or in mass production programming. The only restrictions on clock speed and voltage is the maximum and minimum operating conditions for the device. Refer to the device data sheet for details on this.

For external programming the device is accessed through the PDI and PDI Controller, 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 344. The remainder of this section assumes that the correct physical connection to the PDI is enabled.

Through the PDI, the external programmer access all NVM memories and NVM Controller using the PDI Bus. Doing this all data and program memory spaces are mapped into the linear PDI memory space. Figure 30-4 on page 380 shows the PDI memory space and the base address for each memory space in the device.

Figure 30-4. Memory map for PDI accessing the data and program memories.



30.12.1 Enabling External Programming Interface

NVM programming from the PDI requires enabling, and this is one the following fashion.

1. Load the RESET register in the PDI with 0x59 - the Reset Signature.
2. Load the correct 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 active from the PDI.

30.12.2 NVM Programming

30.12.2.1 Addressing the NVM

When the PDI NVM interface is enabled, all the memories in the device is memory-mapped in the PDI address space. The PDI controller does not need to access the NVM controller's address or data registers, but that 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 trying to load data from the PDIBUS address space). For the reminder of this section all references to reading and writing data or program memory addresses from PDI, refer to the memory map as shown in Figure 30-4 on page 380.

The PDI is always using byte addressing, 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. The user must pay attention to page boundaries for both page buffer loads and page buffer writes.

30.12.2.2 NVM Busy

During programming (page erase and page write) when the NVM is busy, the complete 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. This is a super-set of the commands available for self-programming.

For external programming, the Trigger for Action Triggered Commands is to set the CMDEX bit in the NVM CTRLA register (CMDEX). The Read Triggered Commands are triggered by a direct or indirect Load instruction (LDS or LD) from the PDI (PDI Read). The Write Triggered Commands is triggered by a direct or indirect Store instruction (STS or ST) from the PDI (PDI Write).

Section 30.12.3.1 on page 383 through Section 30.12.3.11 on page 385 explains 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				

CMD[6:0]	Commands / Operation	Trigger	Change Protected	NVM Busy
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
0x02E	Flash Page Write	PDI Write	N	Y
0x2F	Erase & 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 & 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
0x2D	Erase & Write Boot Loader Section Page	PDI Write	N	Y
0x39	Boot Loader Section CRC	NVMAA	Y	Y
Calibration 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 & 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.

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 depend EESAVE fuse setting, refer to ”FUSEBYTE5 - Non-Volatile Memory Fuse Byte 5” on page 32 for details. The User Signature Row, Calibration Row and Fuses are not effected.

1. Load the NVM CMD register with Chip Erase command.
2. Set the CMDEX bit in NVM CTRLA register. This requires the timed CCP sequence during self-programming.

Once this operation starts the PDIBUS 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 PDIBUS 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 Calibration row sections.

1. Load the NVM CMD register with the Read NVM command.
2. Read the selected memory address by doing a PDI Read operation.

Dedicated Read EEPROM, Read Fuse and Read Signature Row and Read Calibration Row commands are also available for the various memory sections. The algorithm for these commands are the same as for the NVM Read 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. This requires the timed CCP sequence during self-programming.

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 accessing and the PDI uses byte addressing, the PDI must write the Flash Page Buffer in 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. 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.

30.12.3.6 *Write Page*

The Write Application Section Page, Write Boot Loader Section Page, Write User Signature Row and Write EEPROM Page is 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 & Write Page*

The Erase & Write Application Section Page, Erase & Write Boot Loader Section Page, and Erase & Write EEPROM Page is 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 & 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 command is used to erase the complete section selected.

1. Load the NVM CMD register with Erase Application/ Boot/ EEPROM Section command
2. Write the selected section by doing a PDI Write. The section is written by addressing any byte location within the section.

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 independent 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. This requires the timed CCP sequence during self-programming.

Once this operation starts the PDIBUS 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 PDIBUS 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 command is 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 Summary - NVM Controller” on page 46 for complete register description on the NVM Controller.

Refer to ”Register Description - PDI Control and Status Register” on page 359 for complete register description on the PDI.

30.14 Register Summary

Refer to ”Register Summary - NVM Controller” on page 46 for complete register summary on the NVM Controller.

Refer to ”Register Summary” on page 361 for complete register summary on 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	48
0x0010	VPORT0	Virtual Port 0	148
0x0014	VPORT1	Virtual Port 1	
0x0018	VPORT2	Virtual Port 2	
0x001C	VPORT3	Virtual Port 3	
0x0030	CPU	CPU	17
0x0040	CLK	Clock Control	94
0x0048	SLEEP	Sleep Controller	101
0x0050	OSC	Oscillator Control	94
0x0060	DFLLRC32M	DFLL for the 32 MHz Internal RC Oscillator	94
0x0068	DFLLRC2M	DFLL for the 2 MHz RC Oscillator	
0x0070	PR	Power Reduction	101
0x0078	RST	Reset Controller	110
0x0080	WDT	Watch-Dog Timer	122
0x0090	MCU	MCU Control	48
0x00A0	PMIC	Programmable Multilevel Interrupt Controller	128
0x00B0	PORTCFG	Port Configuration	148
0x00C0	AES	AES Module	267
0x00F0	VBAT	Battery Backup System	116
0x0100	DMA	DMA Controller	53
0x0180	EVSYS	Event System	75
0x01C0	NVM	Non Volatile Memory (NVM) Controller	46
0x0200	ADCA	Analog to Digital Converter on port A	315
0x0240	ADCB	Analog to Digital Converter on port B	
0x0300	DACA	Digital to Analog Converter on port A	327
0x0320	DACB	Digital to Analog Converter on port B	
0x0380	ACA	Analog Comparator pair on port A	336
0x0390	ACB	Analog Comparator pair on port B	
0x0400	RTC	Real Time Counter	196
0x0420	RTC32	32-bit Real Time Counter	204
0x0440	EBI	External Bus Interface	288
0x0480	TWIC	Two Wire Interface on port C	228
0x0490	TWID	Two Wire Interface on port D	
0x04A0	TWIE	Two Wire Interface on port E	
0x04B0	TWIF	Two Wire Interface on port F	
0x0600	PORTA	Port A	148
0x0620	PORTB	Port B	
0x0640	PORTC	Port C	
0x0660	PORTD	Port D	
0x0680	PORTE	Port E	
0x06A0	PORTF	Port F	
0x06E0	PORTH	Port H	
0x0700	PORTJ	Port J	
0x0720	PORTK	Port K	
0x07C0	PORTQ	Port Q	
0x07E0	PORTR	Port R	
0x0800	TCC0	Timer/Counter 0 on port C	175
0x0840	TCC1	Timer/Counter 1 on port C	
0x0880	AWEXC	Advanced Waveform Extension on port C	187
0x0890	HIRESC	High Resolution Extension on port C	189
0x08A0	USARTC0	USART 0 on port C	255
0x08B0	USARTC1	USART 1 on port C	
0x08C0	SPIC	Serial Peripheral Interface on port C	234
0x08F0	IRCOM	Infrared Communication Module	259
0x0900	TCD0	Timer/Counter 0 on port D	175
0x0940	TCD1	Timer/Counter 1 on port D	
0x0980	AWEXD	Advanced Waveform Extension on port C	187
0x0990	HIRESD	High Resolution Extension on port D	189
0x09A0	USARTD0	USART 0 on port D	255

Base Address	Name	Description	Page
0x09B0	USARTD1	USART 1 on port D	
0x09C0	SPID	Serial Peripheral Interface on port D	234
0x0A00	TCE0	Timer/Counter 0 on port E	175
0x0A40	TCE1	Timer/Counter 1 on port E	
0x0A80	AWEXE	Advanced Waveform Extension on port E	187
0x0A90	HIRESE	High Resolution Extension on port E	189
0x0AA0	USARTE0	USART 0 on port E	255
0x0AB0	USARTE1	USART 1 on port E	
0x0AC0	SPIE	Serial Peripheral Interface on port E	234
0x0B00	TCF0	Timer/Counter 0 on port F	175
0x0B40	TCF1	Timer/Counter 1 on port F	
0x0B80	AWEXF	Advanced Waveform Extension on port C	187
0x0B90	HIRESF	High Resolution Extension on port F	189
0x0BA0	USARTF0	USART 0 on port F	255
0x0BB0	USARTF1	USART 1 on port F	
0x0BC0	SPIF	Serial Peripheral Interface on port F	234

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
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)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CALL	k	call Subroutine	PC ← k	None	3 / 4[1]
RET		Subroutine Return	PC ← STACK	None	4 / 5[1]
RETI		Interrupt Return	PC ← STACK	I	4 / 5[1]
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	2 / 3 / 4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC ← PC + 2 or 3	None	2 / 3 / 4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data Transfer Instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
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]

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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	-
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]
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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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
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. Appendix A: EBI Timing Diagrams

33.1 SRAM 3-Port ALE1 CS

Figure 33-1. Write, no ALE

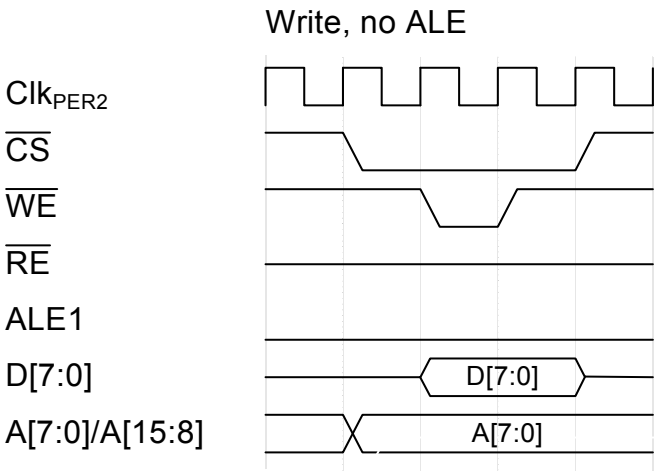


Figure 33-2. Write, ALE

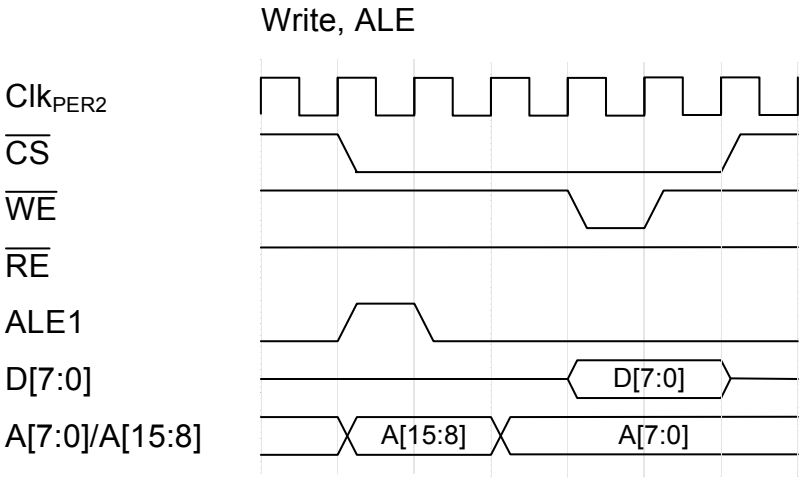


Figure 33-3. Read, no ALE

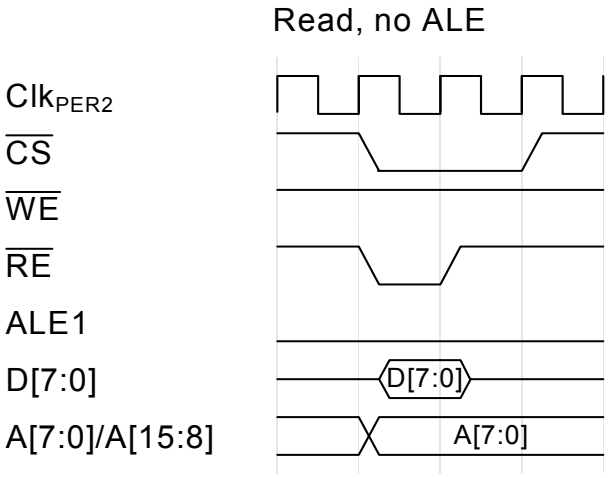
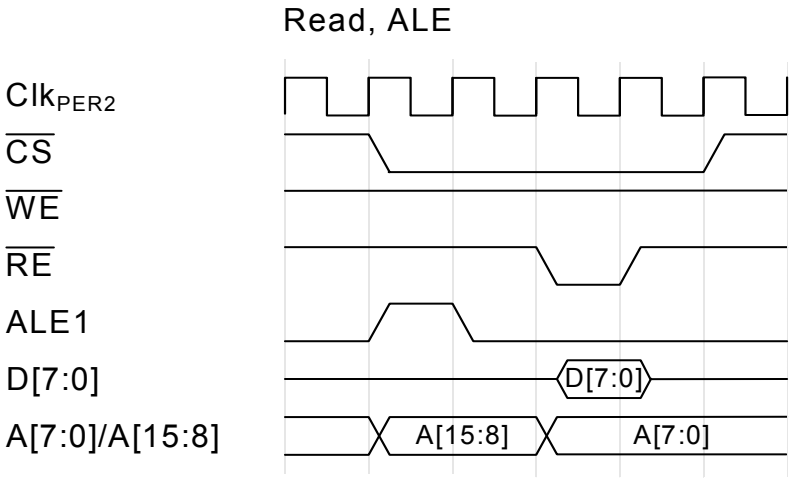


Figure 33-4. Read, ALE



33.2 SRAM 3-Port ALE12 CS

Figure 33-5. Write, no ALE

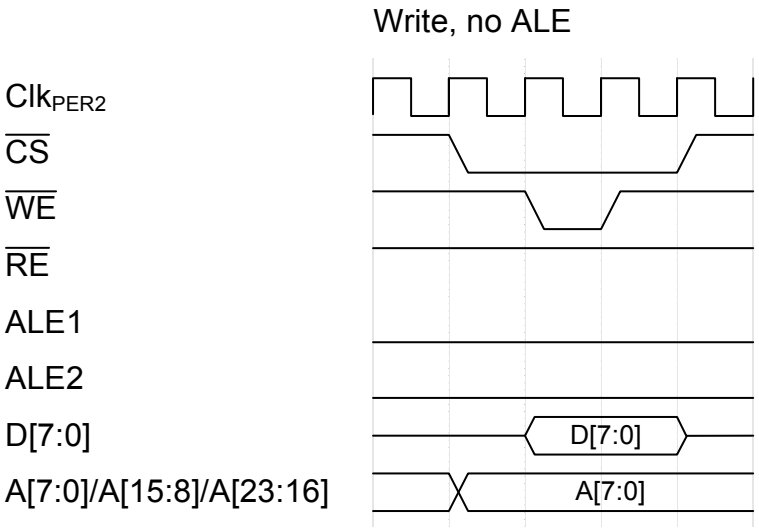


Figure 33-6. Write, ALE1

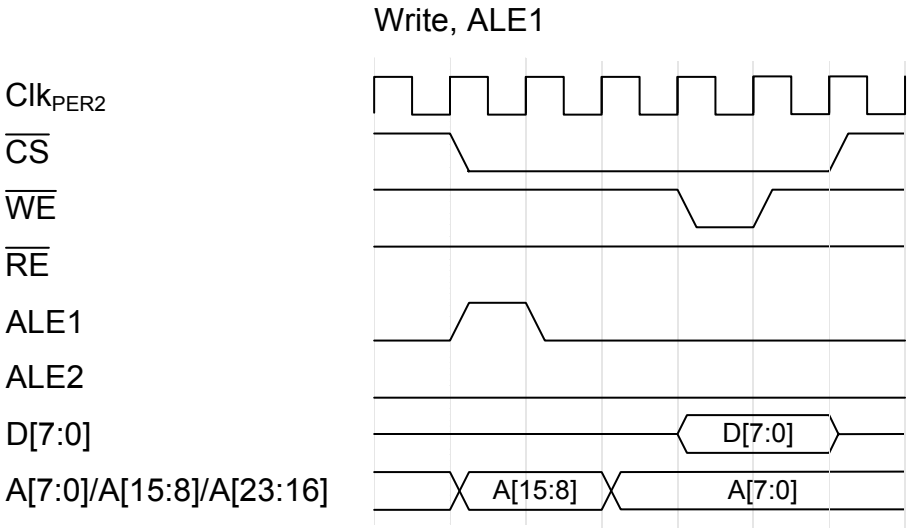


Figure 33-7. Write, ALE1 + ALE2

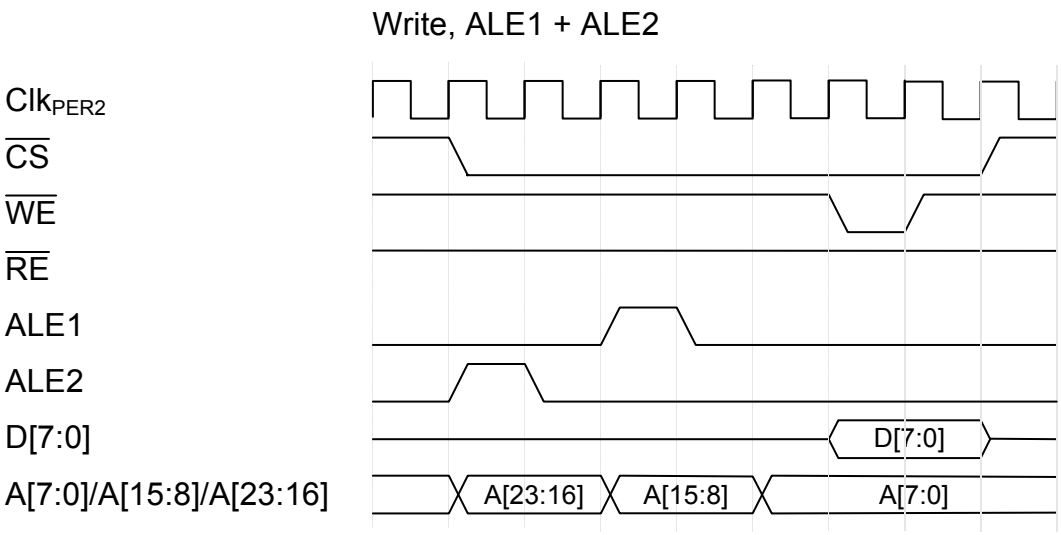


Figure 33-8. Read, no ALE

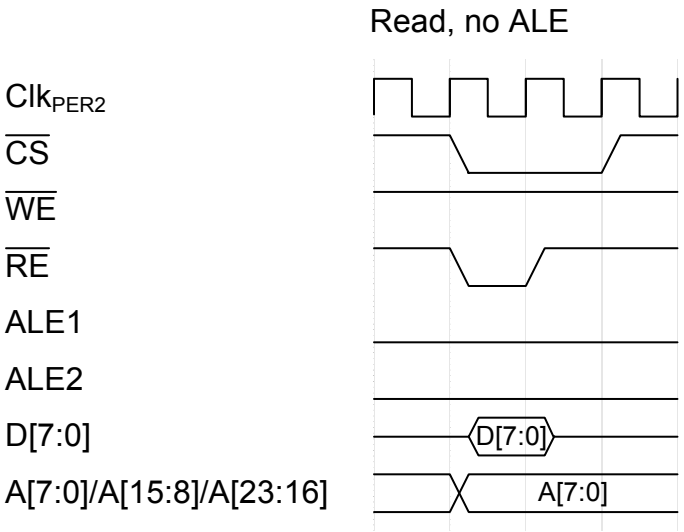


Figure 33-9. Read, ALE1

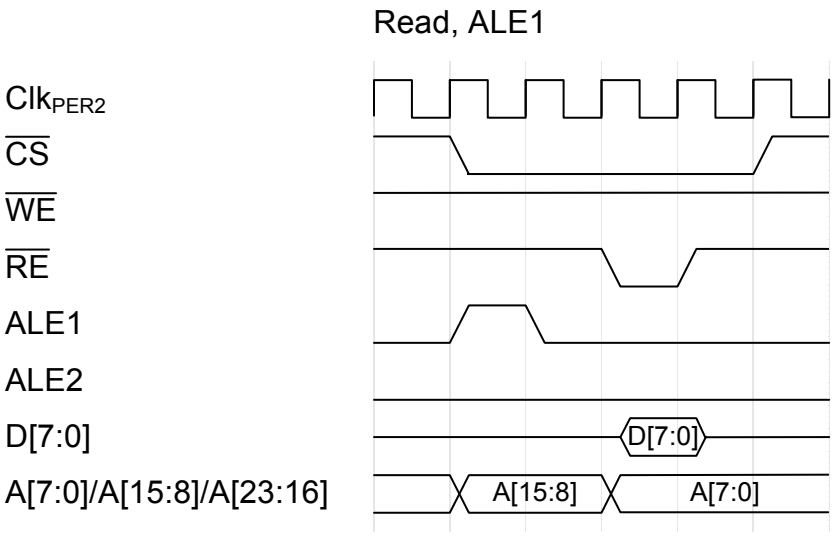
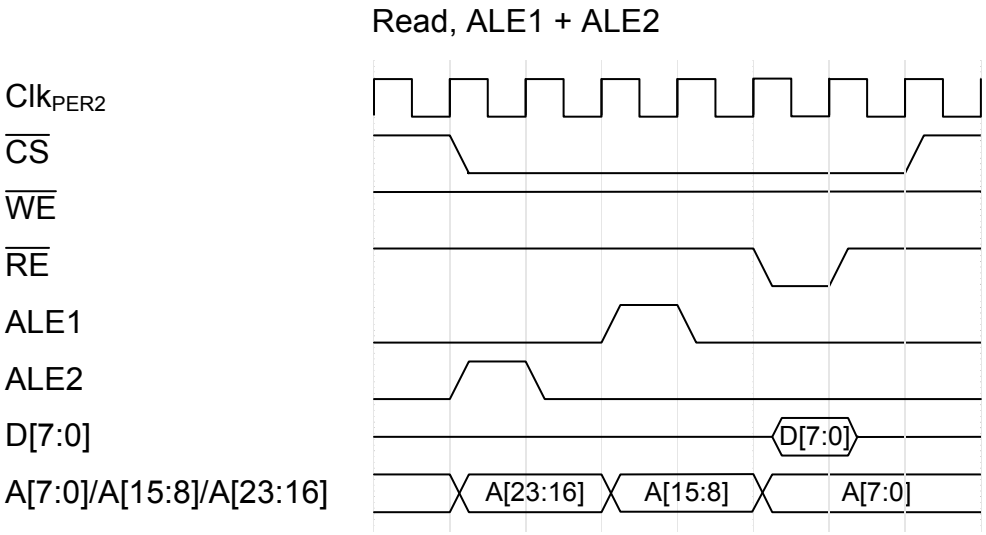


Figure 33-10. Read, ALE1 + ALE2



33.3 SRAM 4 -Port ALE2 CS

Figure 33-11. Write, no ALE

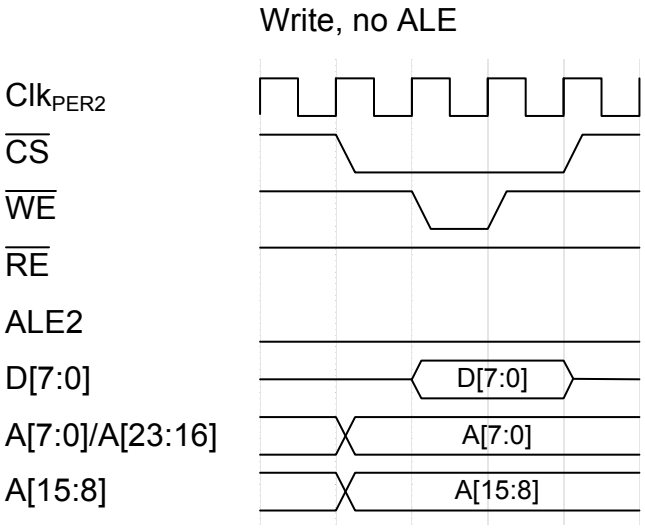


Figure 33-12. Write, ALE

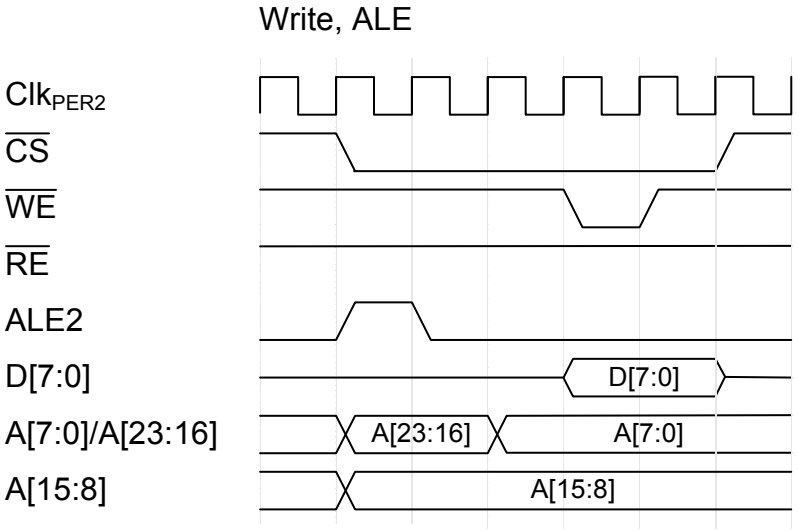


Figure 33-13. Read, no ALE

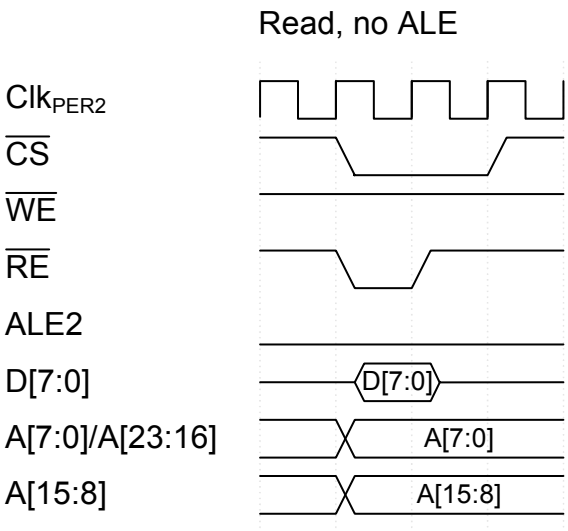
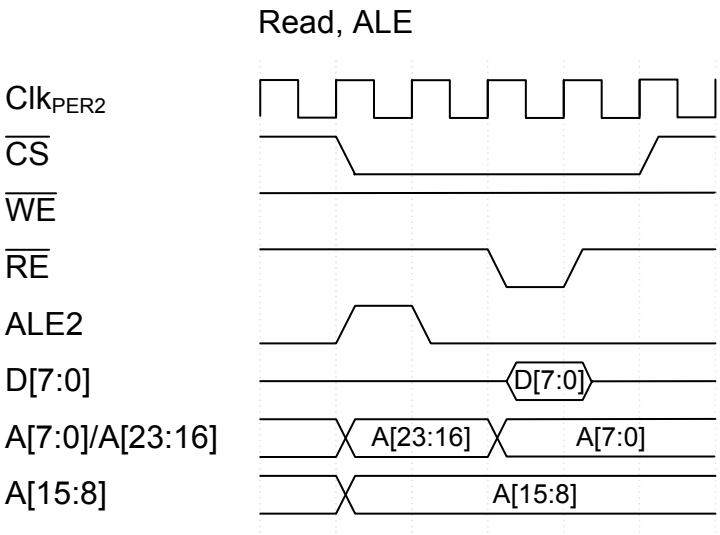


Figure 33-14. Read, ALE



33.4 SRAM 4 - Port NOALE CS

Figure 33-15. Write

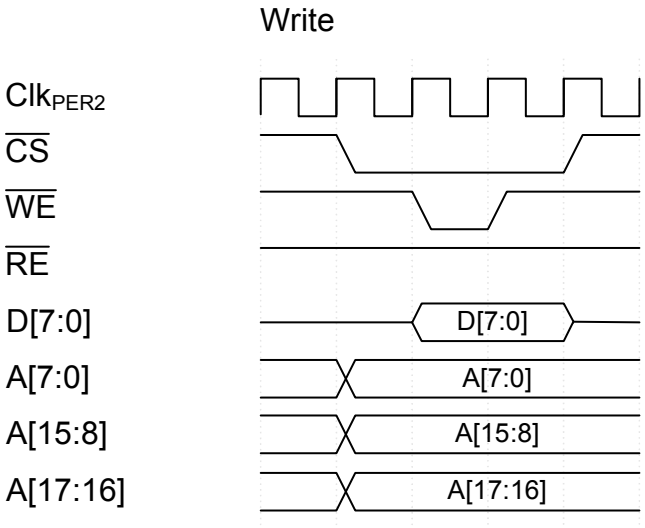
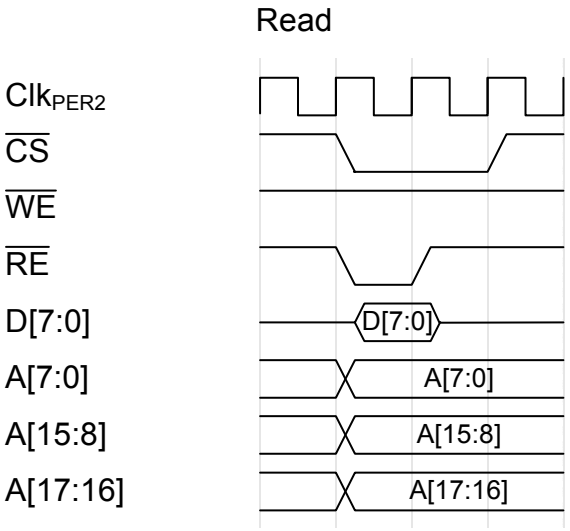


Figure 33-16. Read



33.5 LPC 2 - Port ALE12 CS

Figure 33-17. Write, ALE1

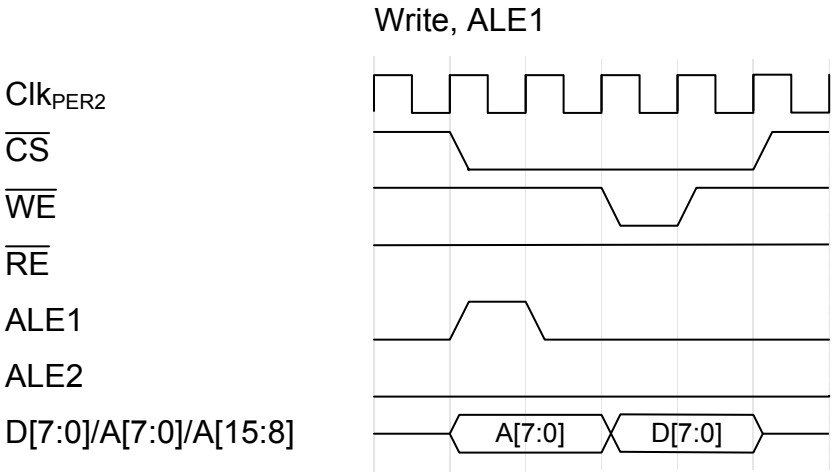


Figure 33-18. Write, ALE1 + ALE2

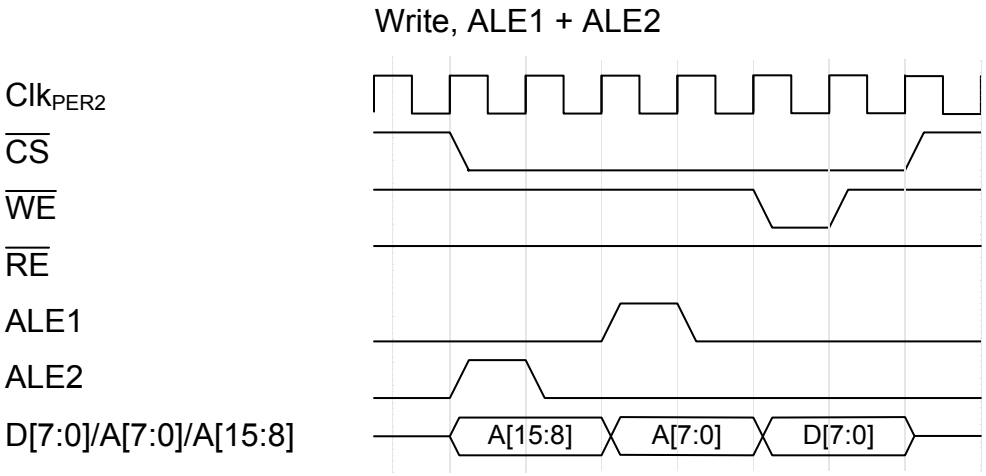


Figure 33-19. Read, ALE1

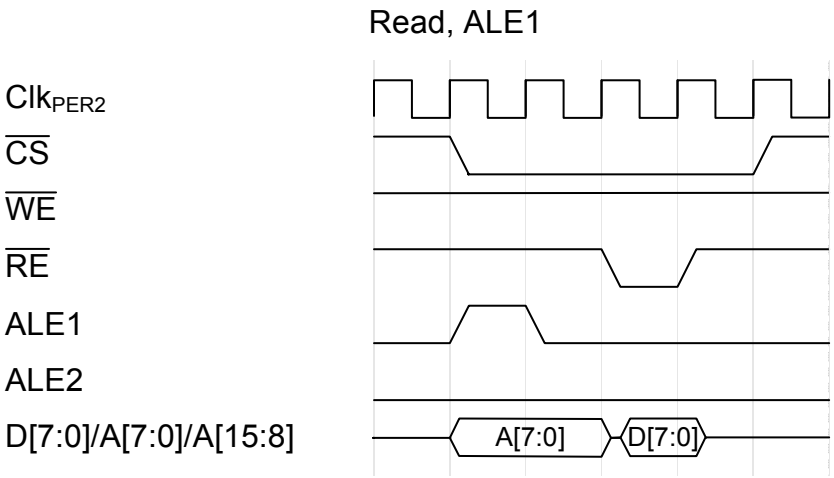
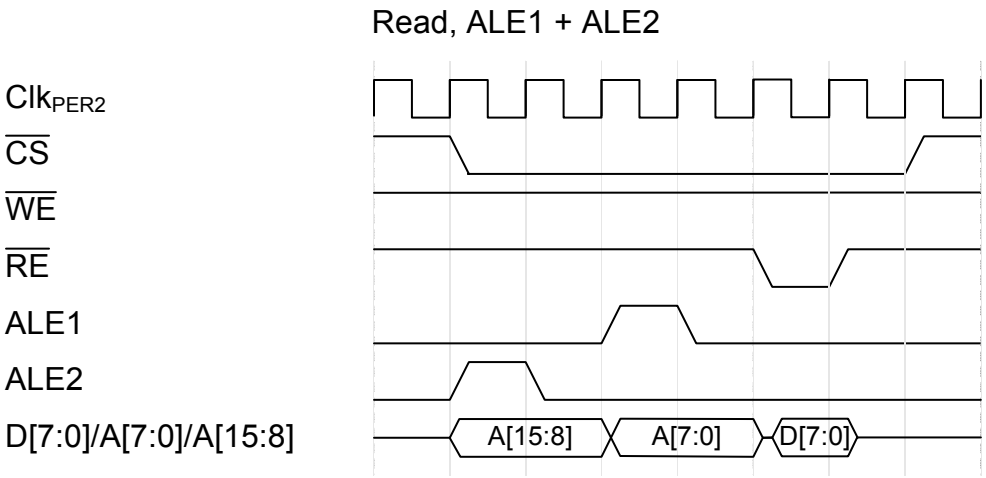


Figure 33-20. Read, ALE1 + ALE2



33.6 LPC 3 - Port ALE1 CS

Figure 33-21. Write

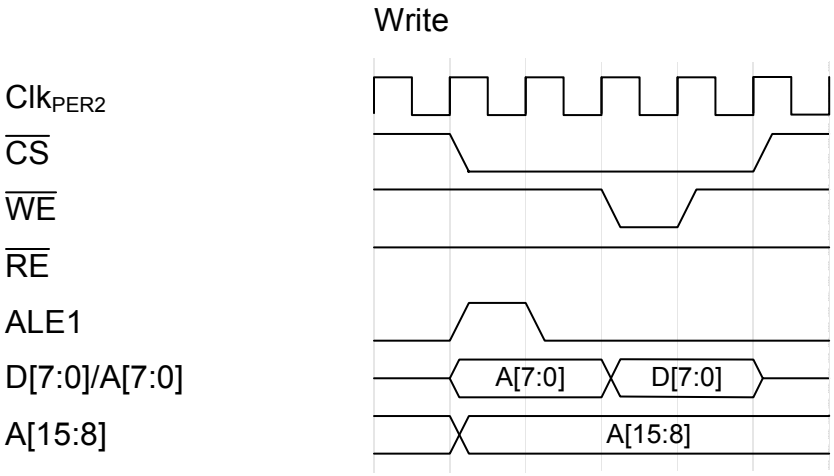
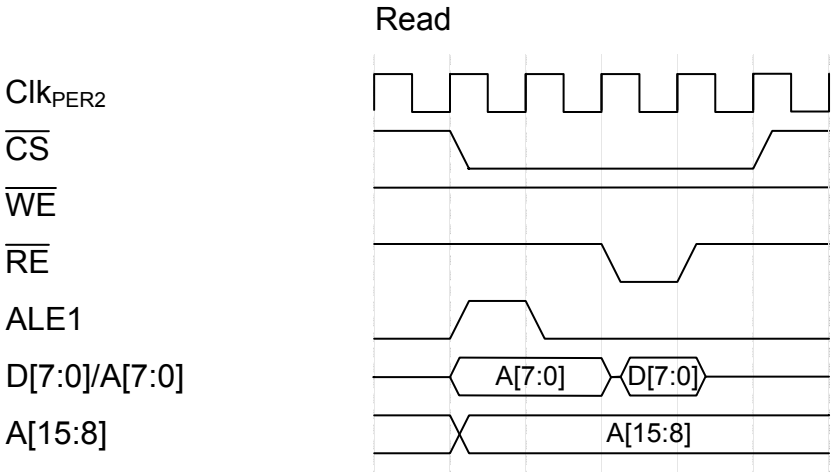


Figure 33-22. Read



33.7 LPC 2 - Port ALE1 CS

Figure 33-23. Write

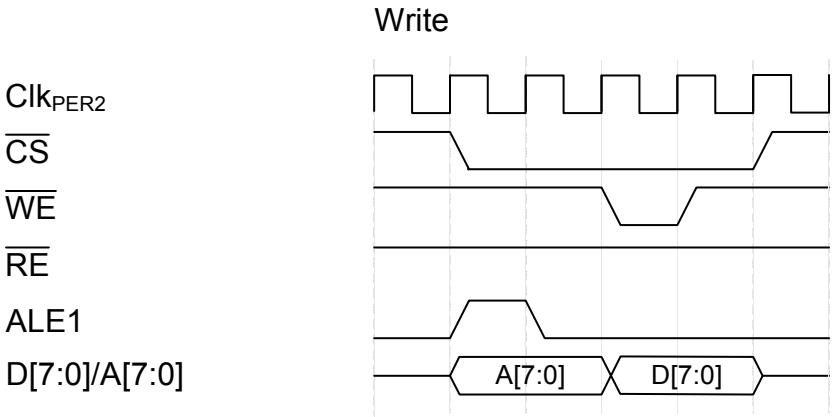
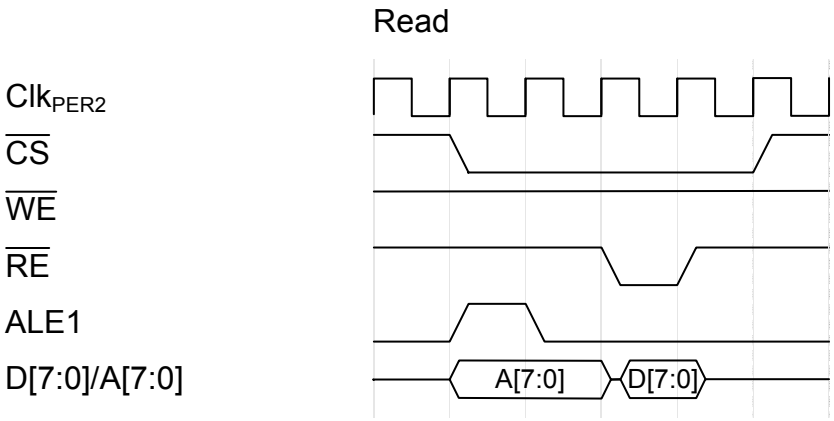


Figure 33-24. Read



33.8 SRAM 3 - Port ALE1 noCS

Figure 33-25. Write, no ALE

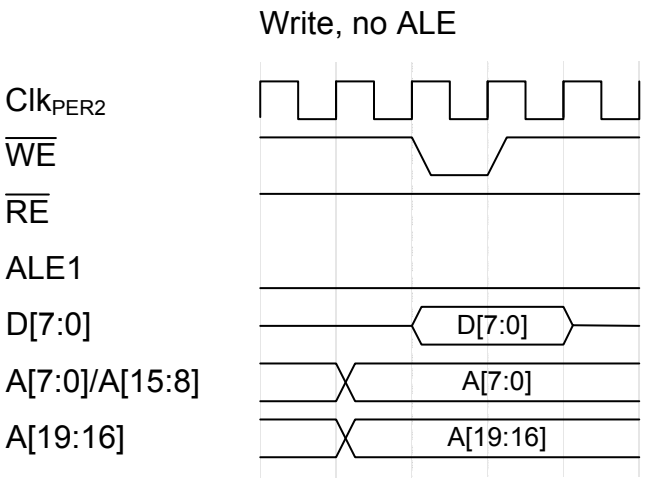


Figure 33-26. Write, ALE

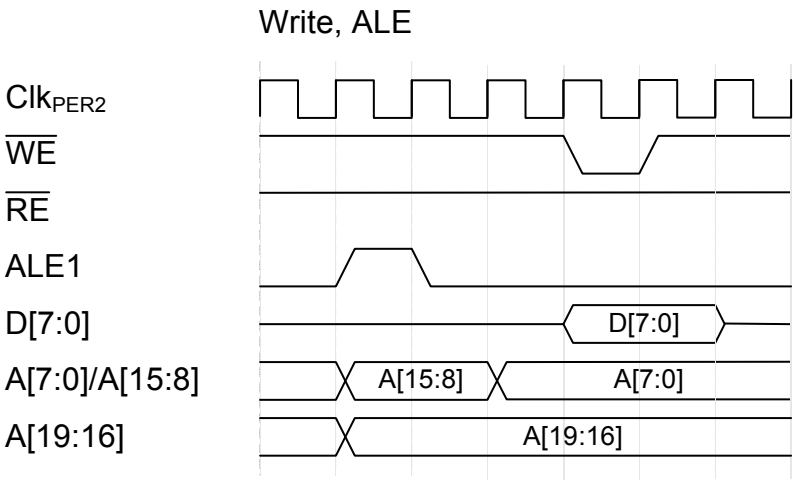


Figure 33-27. Read, no ALE

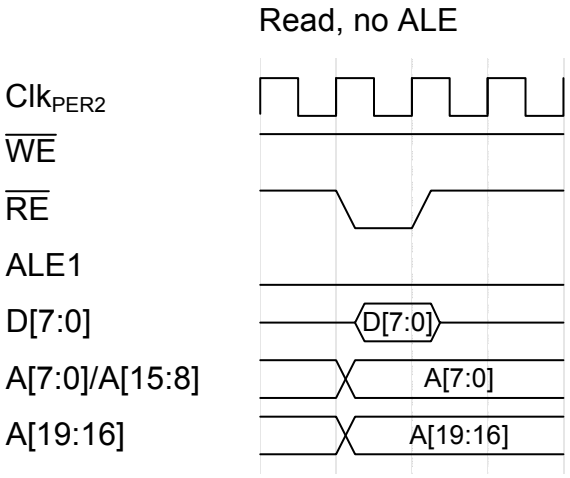
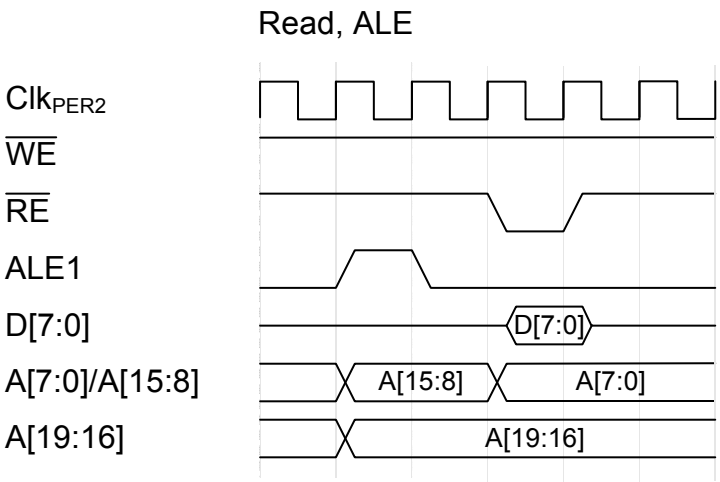


Figure 33-28. Read, ALE



33.9 SRAM 4 - Port NOALE no CS

Figure 33-29. Write

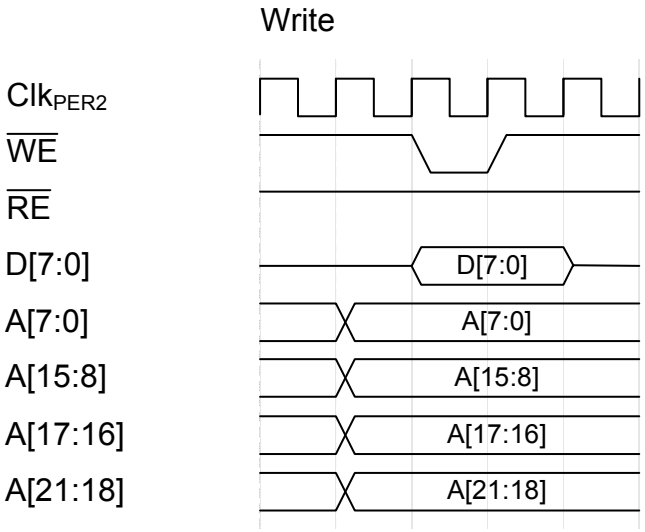
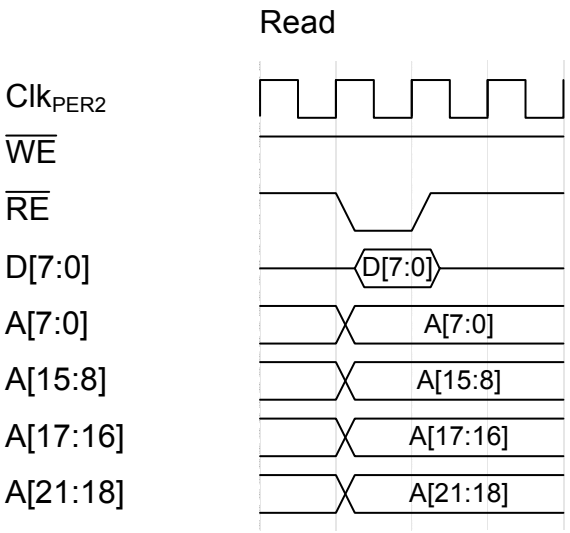


Figure 33-30. Read



33.10 LPC 2 - Port ALE12 no CS

Figure 33-31. Write, ALE1

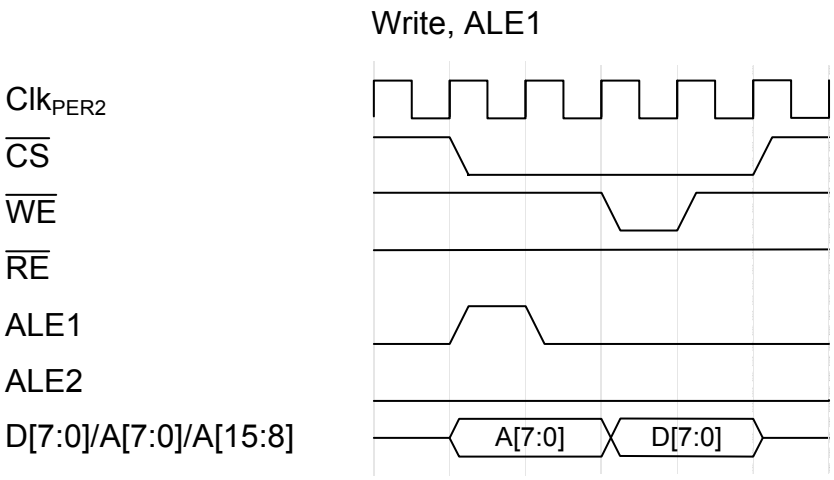


Figure 33-32. Write, ALE1 + ALE2

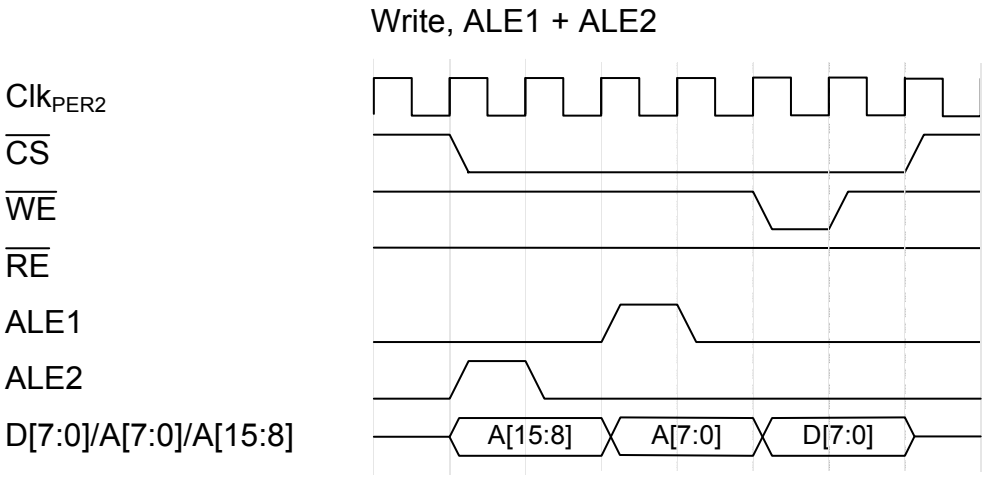


Figure 33-33. Read, ALE1

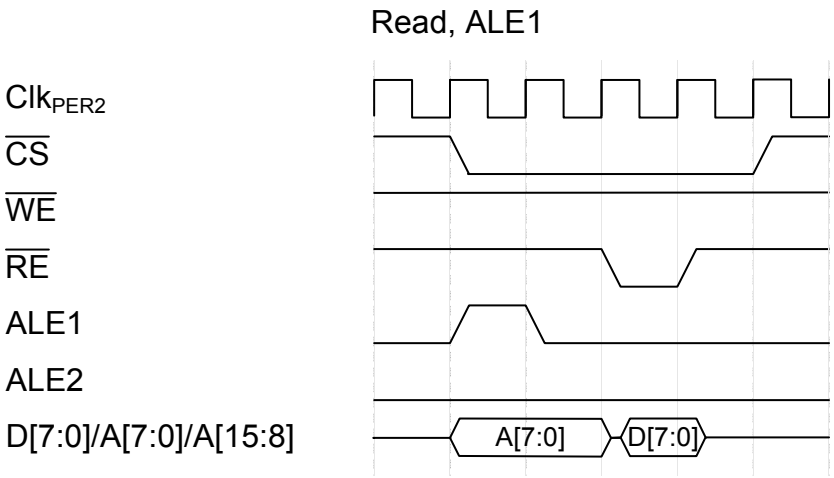
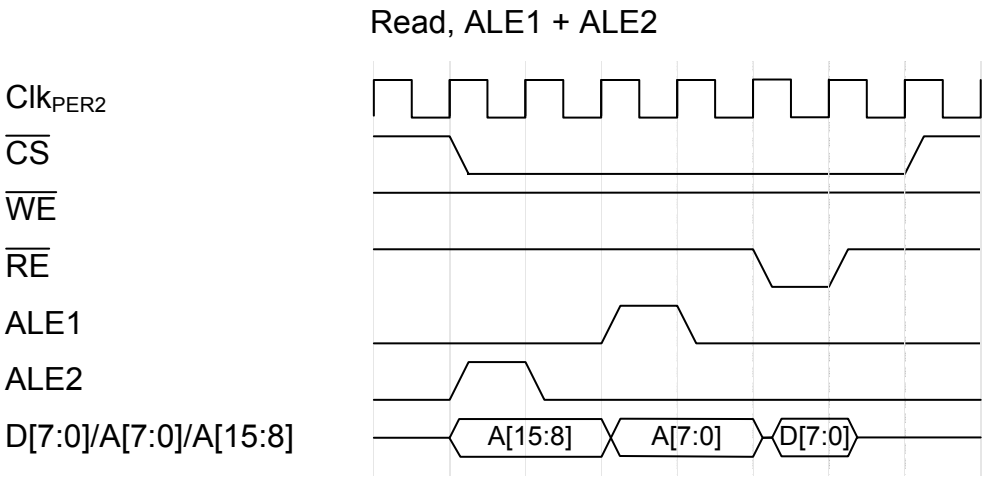
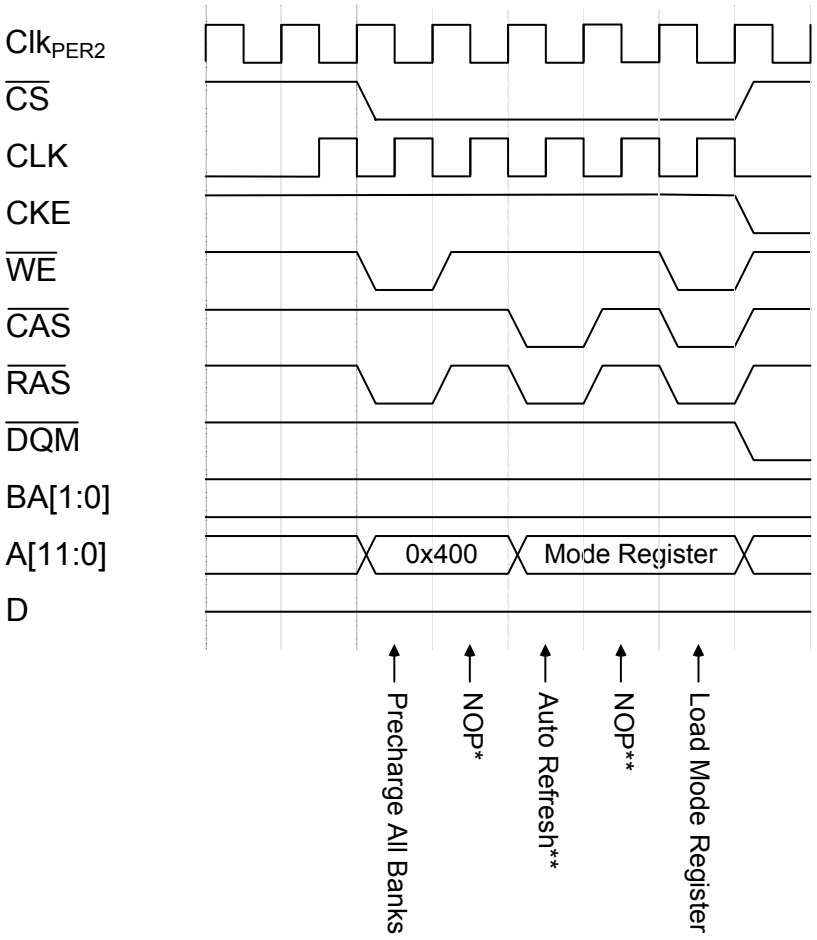


Figure 33-34. Read, ALE1 + ALE2



33.11 SDRAM init

Figure 33-35. SDRAM init

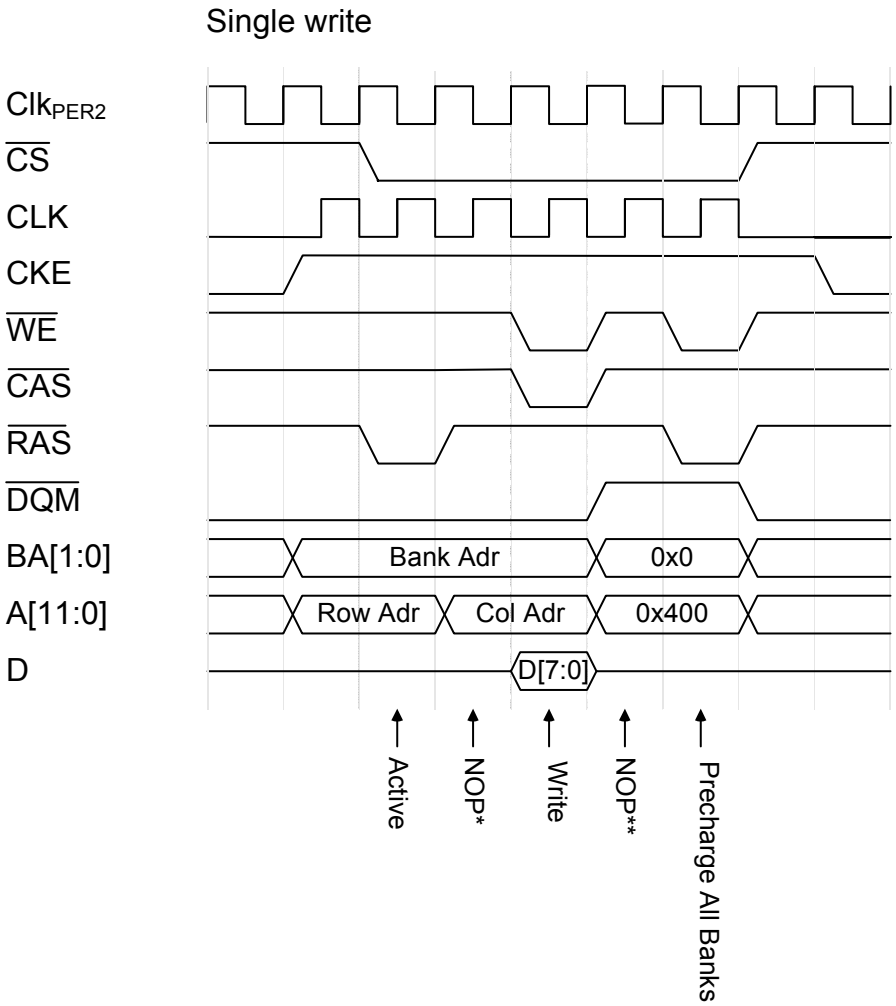


* The number of NOPs is equal to RPDLY[2:0] (RPDLY = 1 is shown)

** The Auto Refresh and following NOPs are repeated 8 times
The number of NOPs is equal to ROWCYCDLY[2:0] (ROWCYCDLY = 1 is shown)

33.12 SDRAM 8-bit rwite

Figure 33-36. Single write

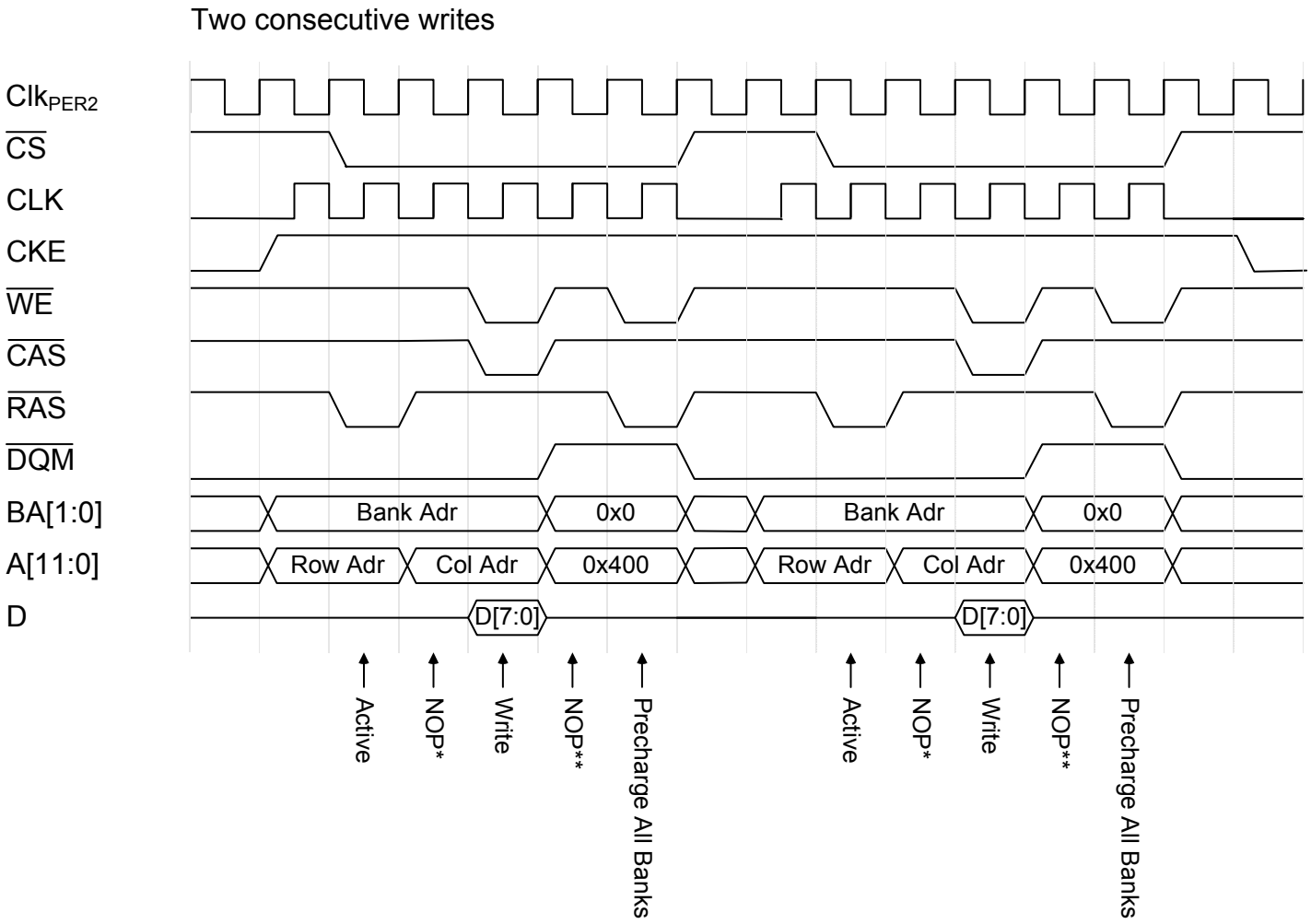


* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-37. Two consecutive writes

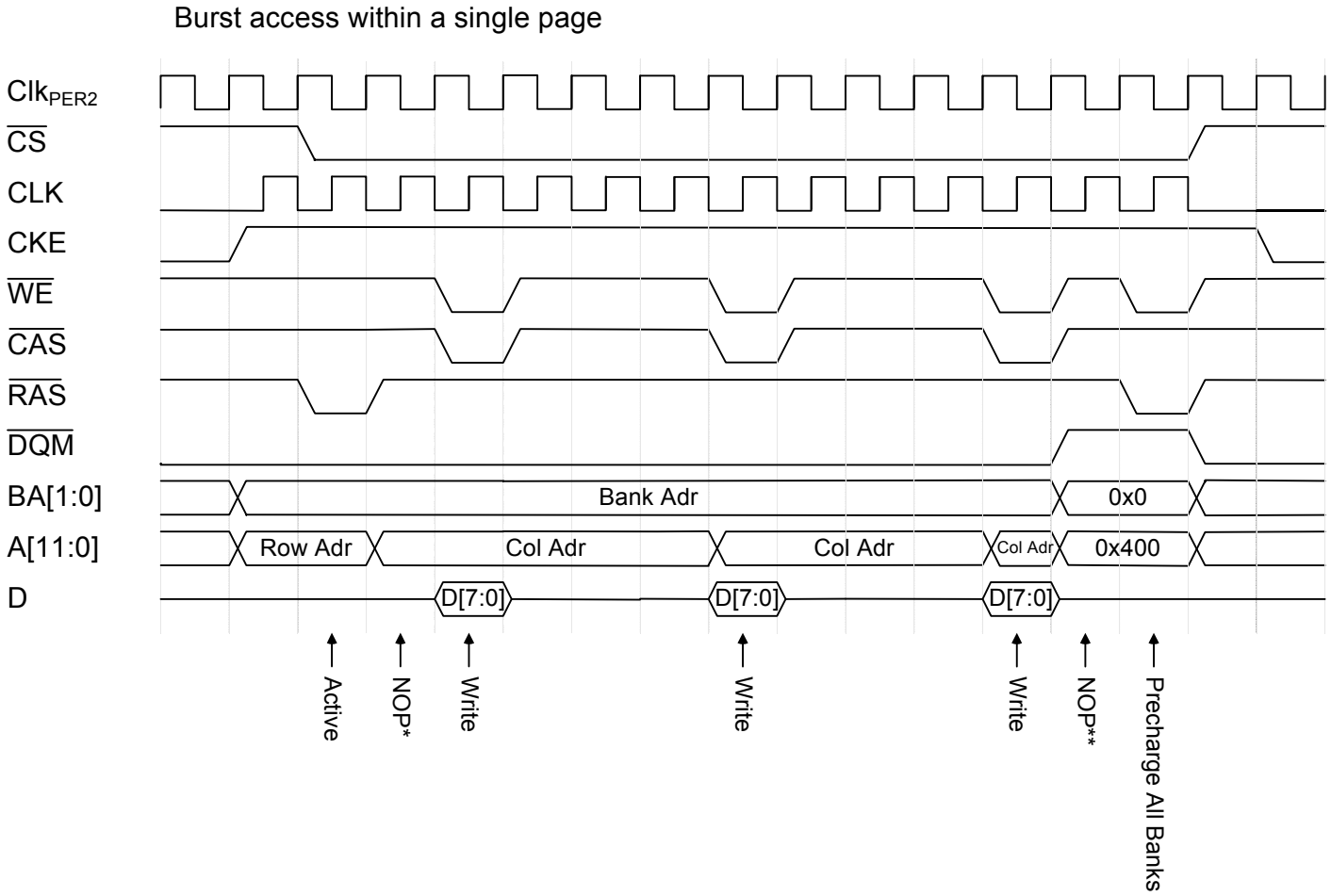


* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-38. Burst access within a single page

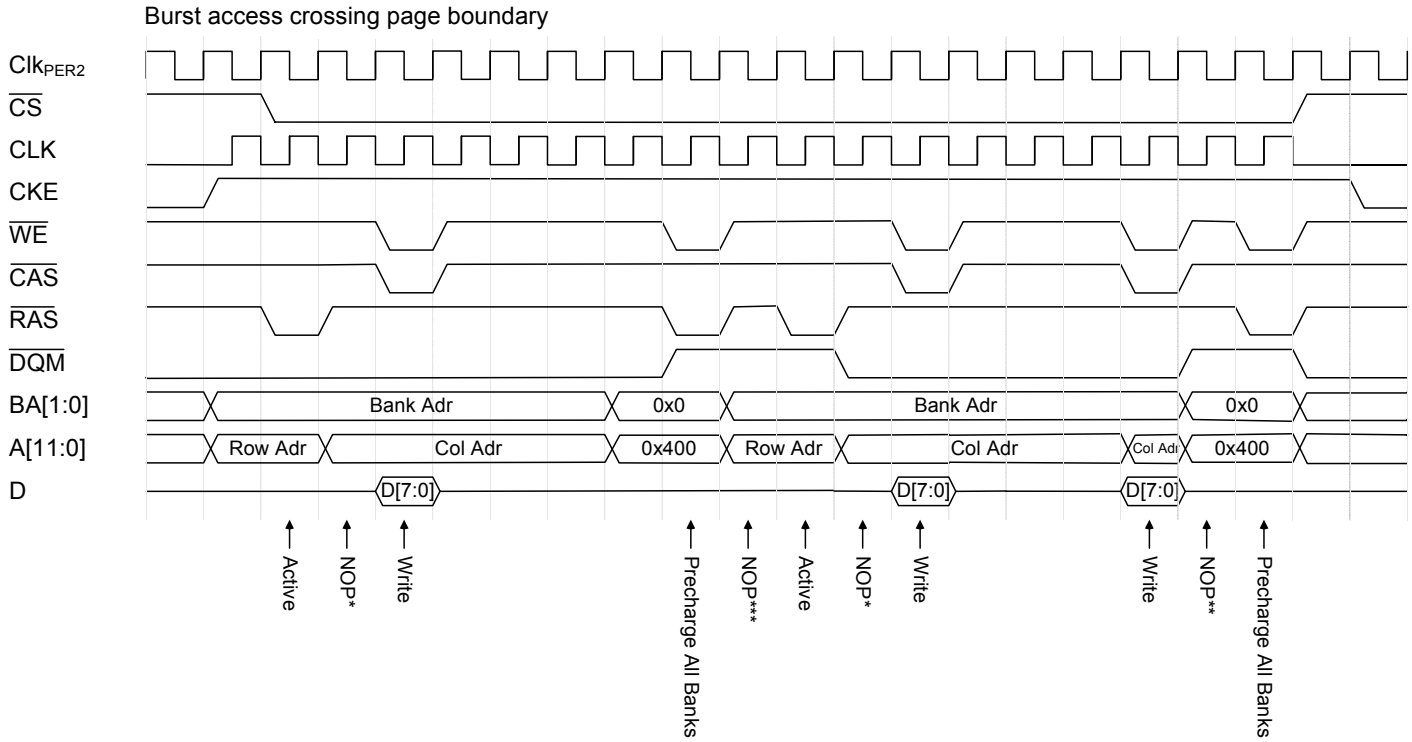


* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-39. Burst access crossing page boundary



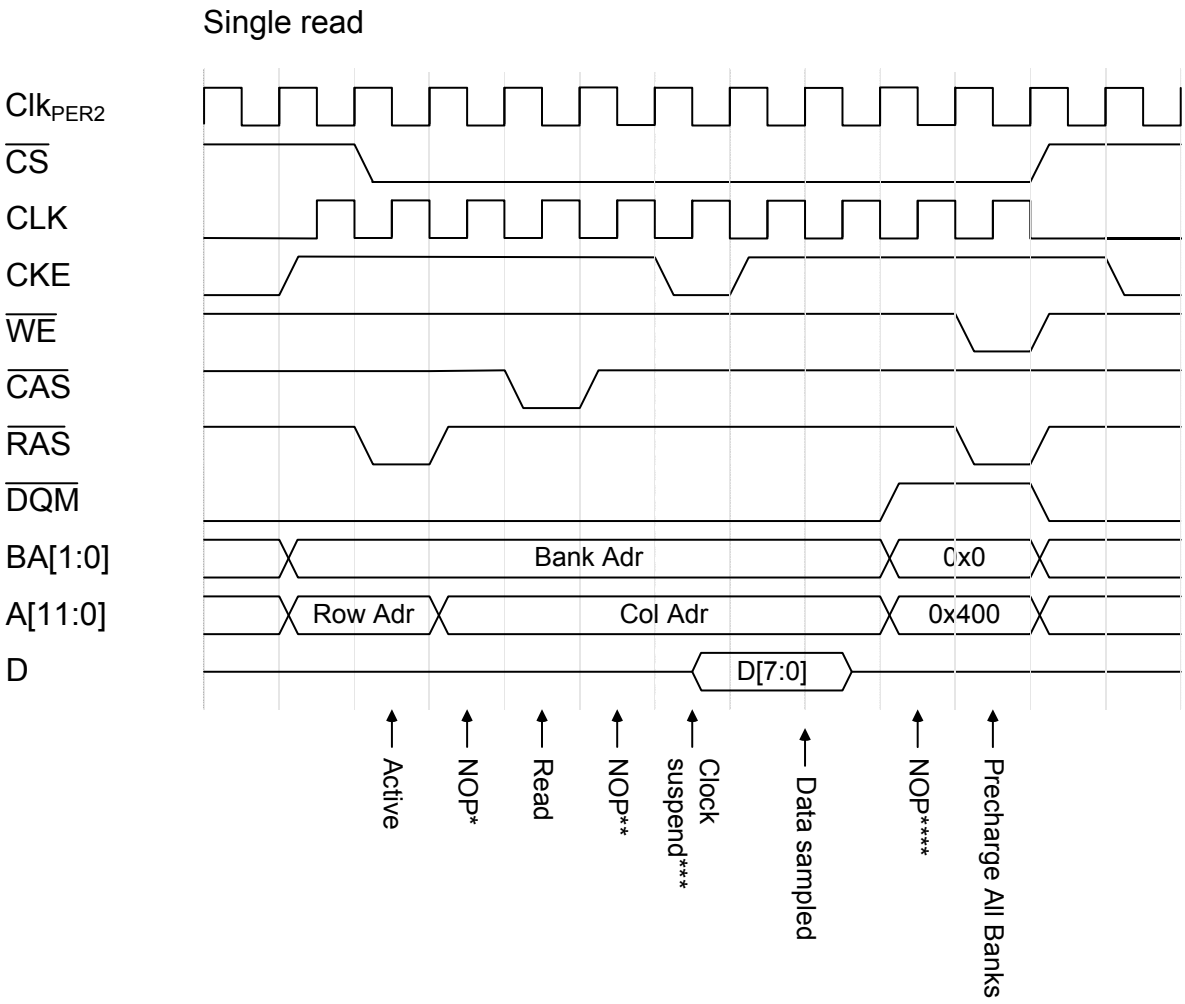
* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

33.13 SDRAM 8-bit read

Figure 33-40. Single read



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

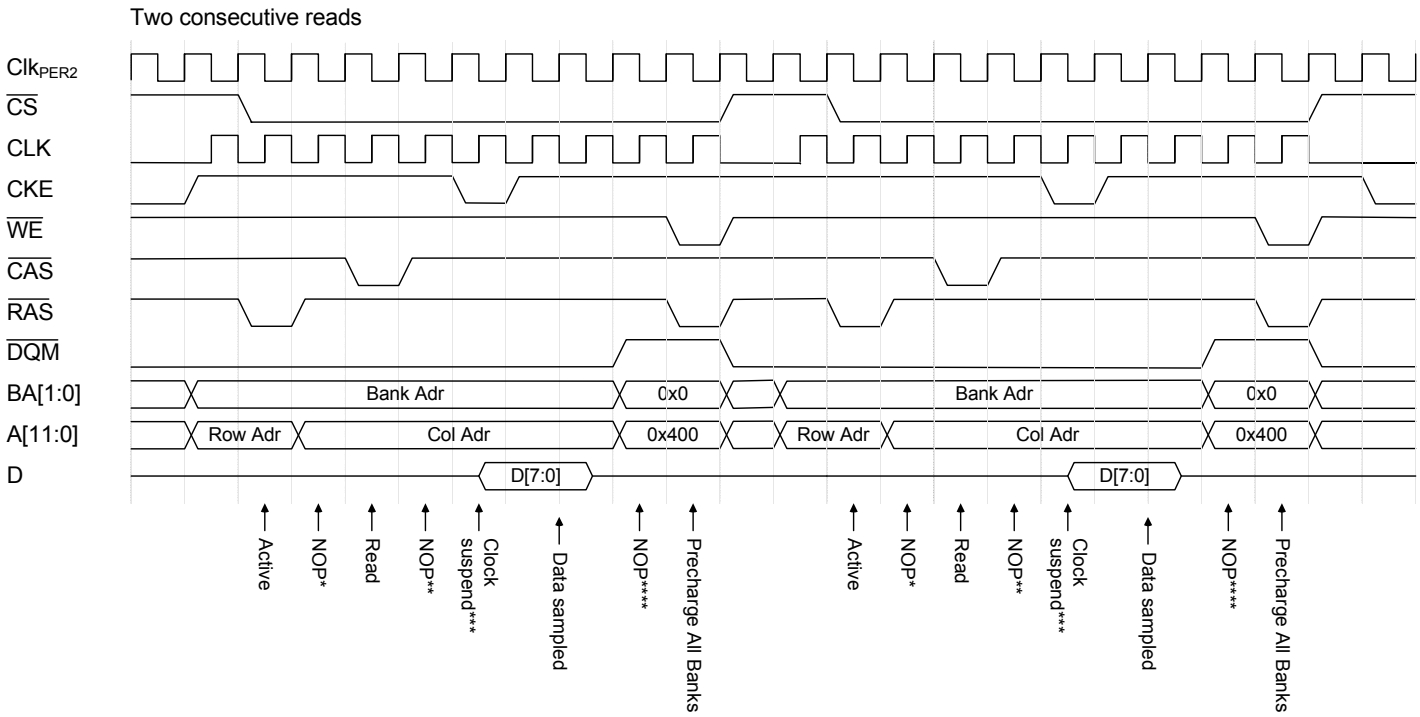
** NOP is only inserted for CAS3

*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-41. Two consecutive reads



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

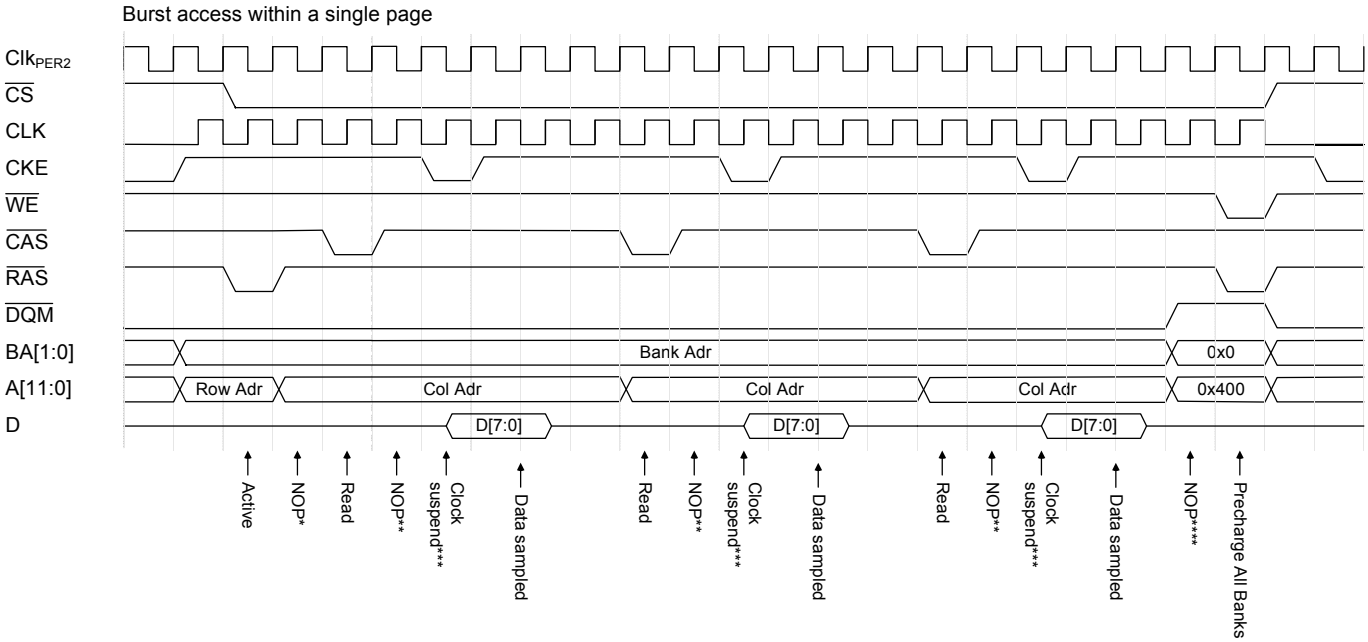
** NOP is only inserted for CAS3

*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-42. Burst access within a single page



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

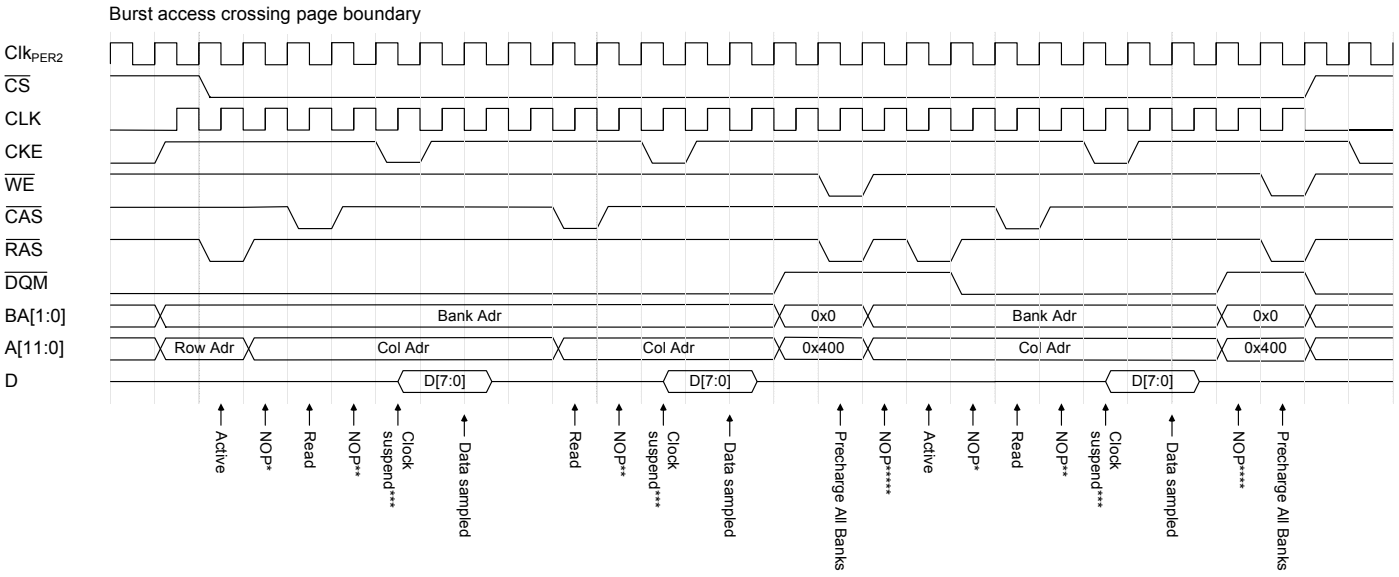
** NOP is only inserted for CAS3

*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-43. Burst access crossing page boundary



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** NOP is only inserted for CAS3

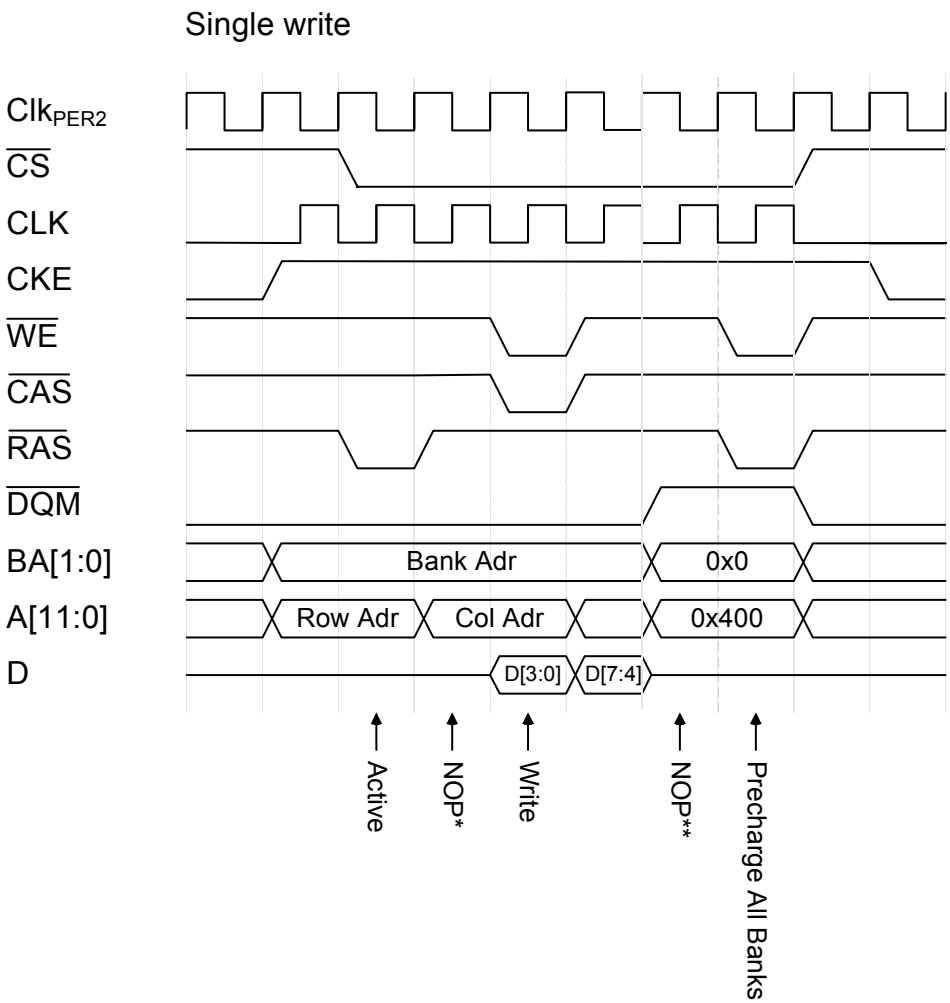
*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

33.14 SDRAM 4-bit rwite

Figure 33-44. Single write

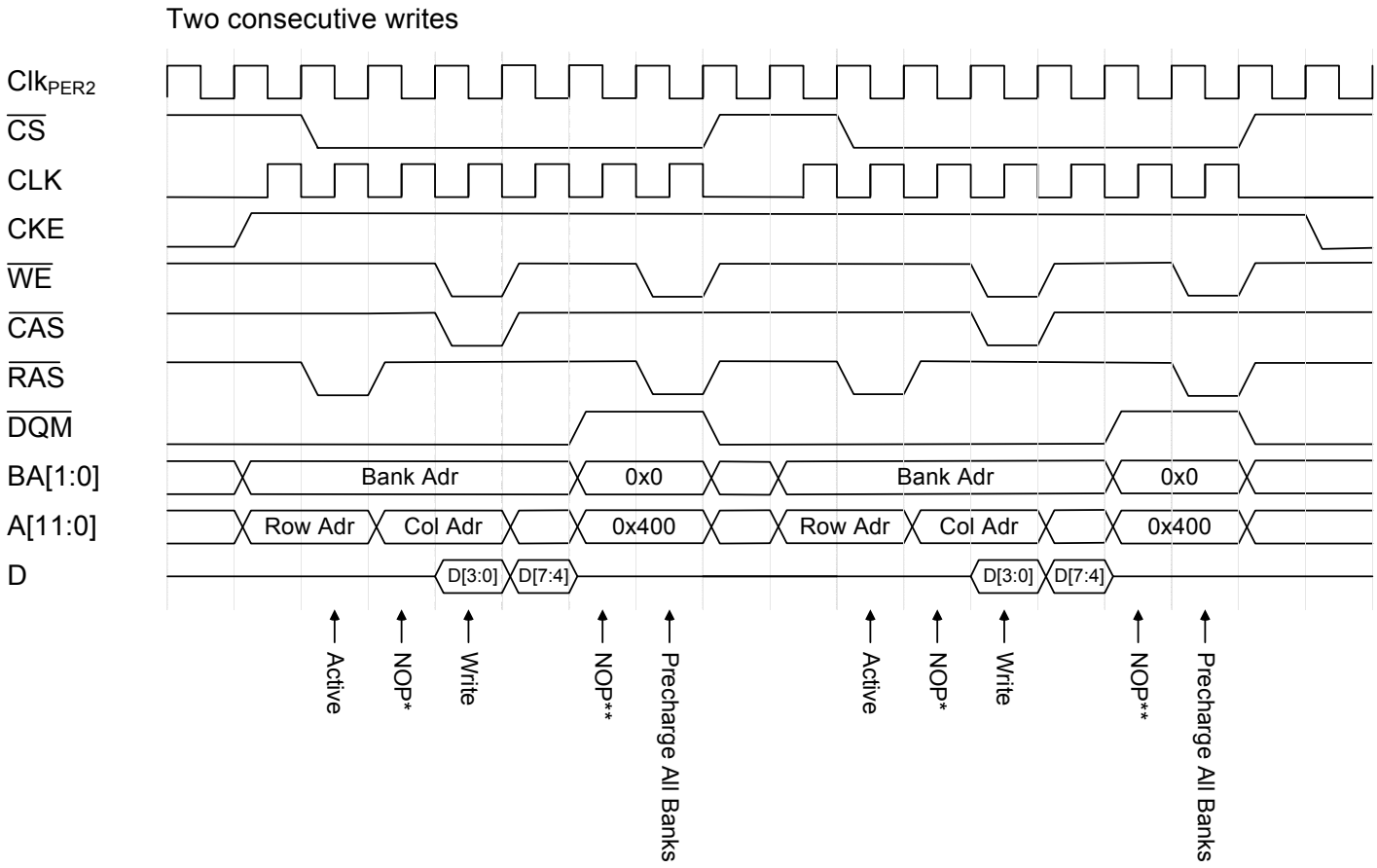


* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-45. Two consecutive writes

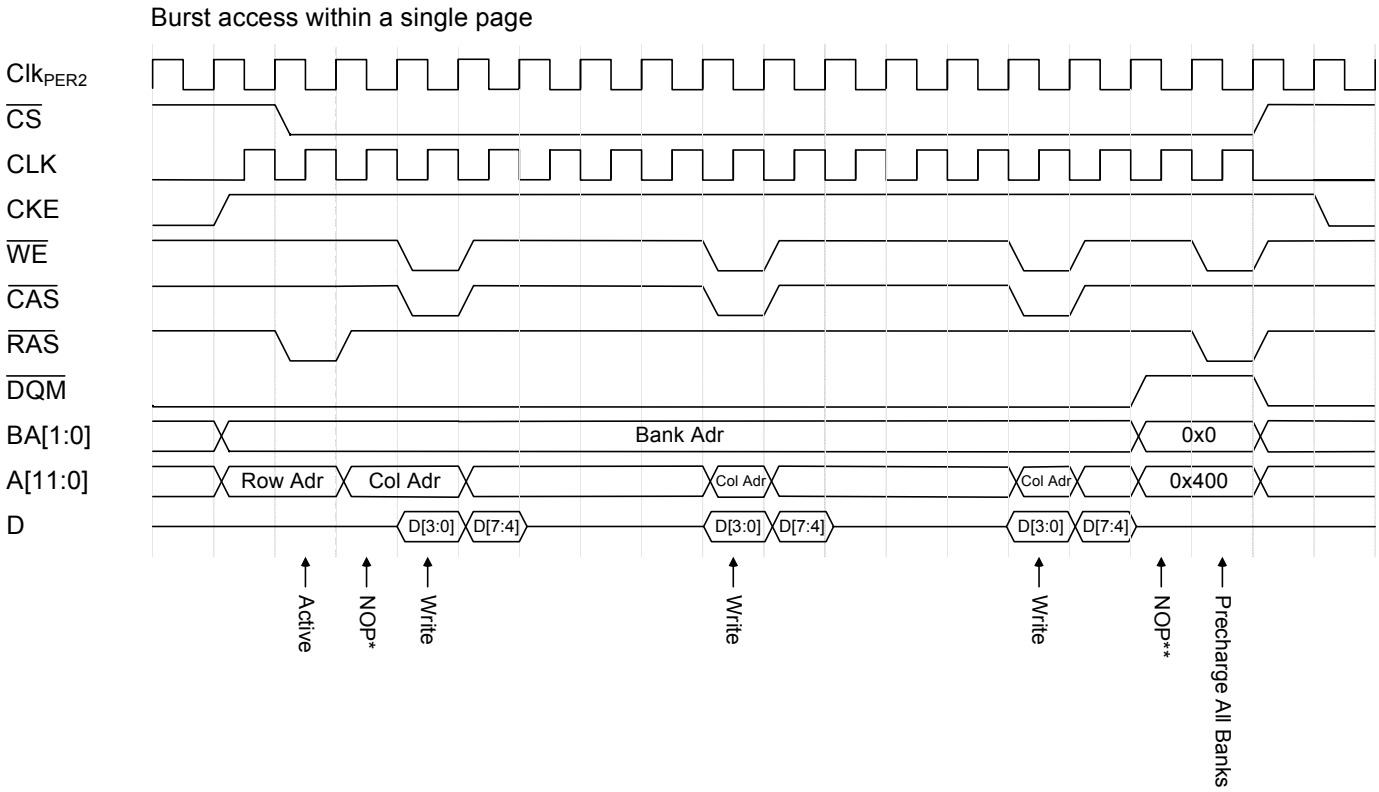


* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-46. Burst access within a single page

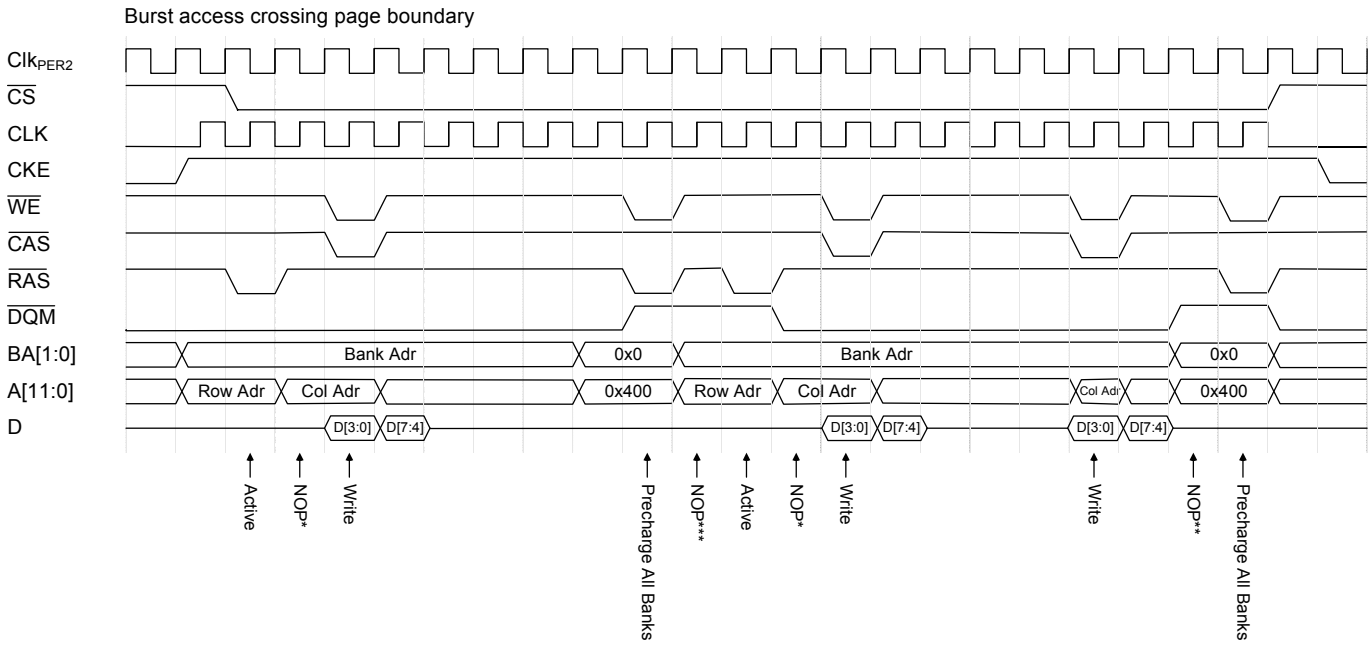


* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-47. Burst access crossing page boundary



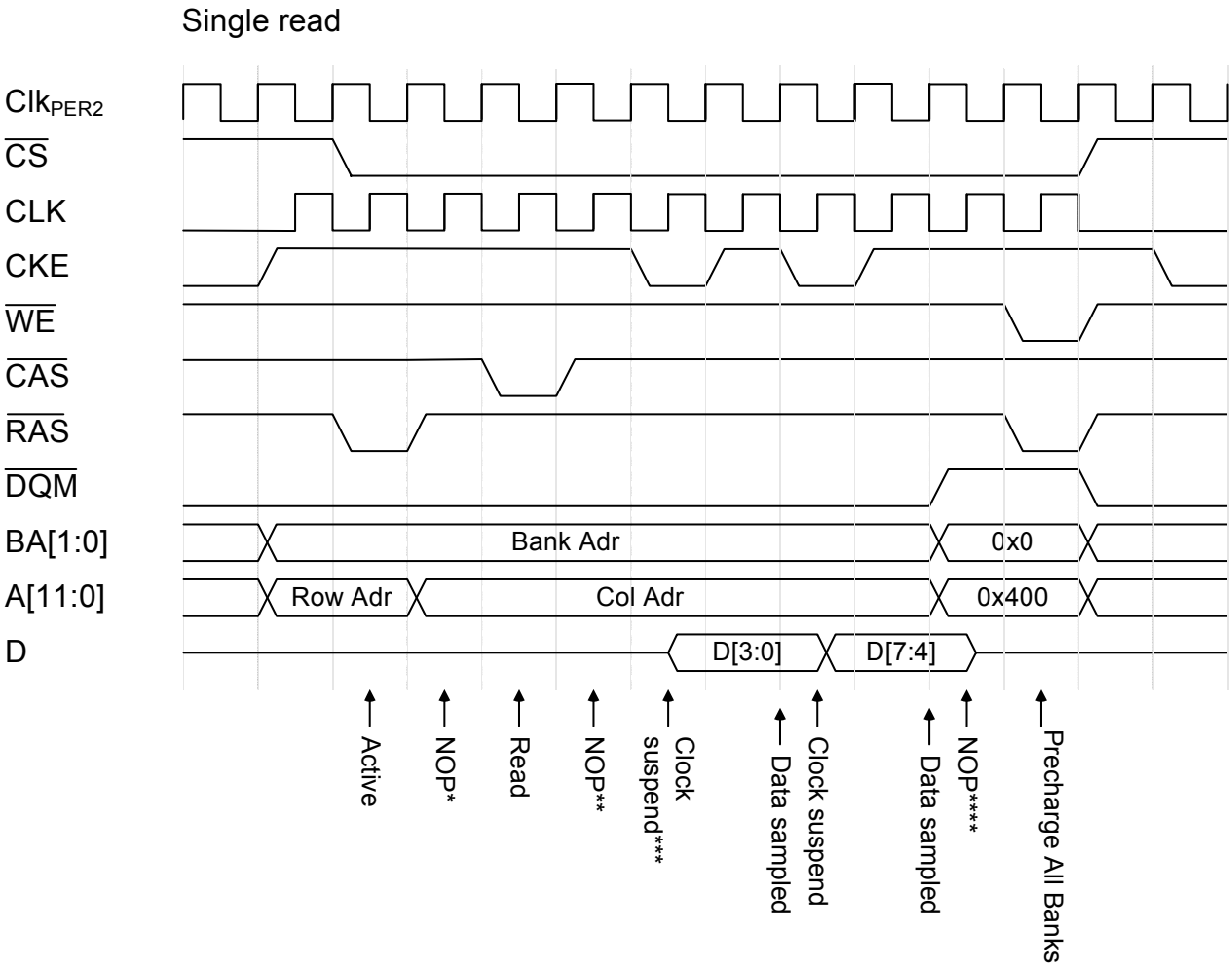
* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

*** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

33.15 SDRAM 4-bit read

Figure 33-48. Single read



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

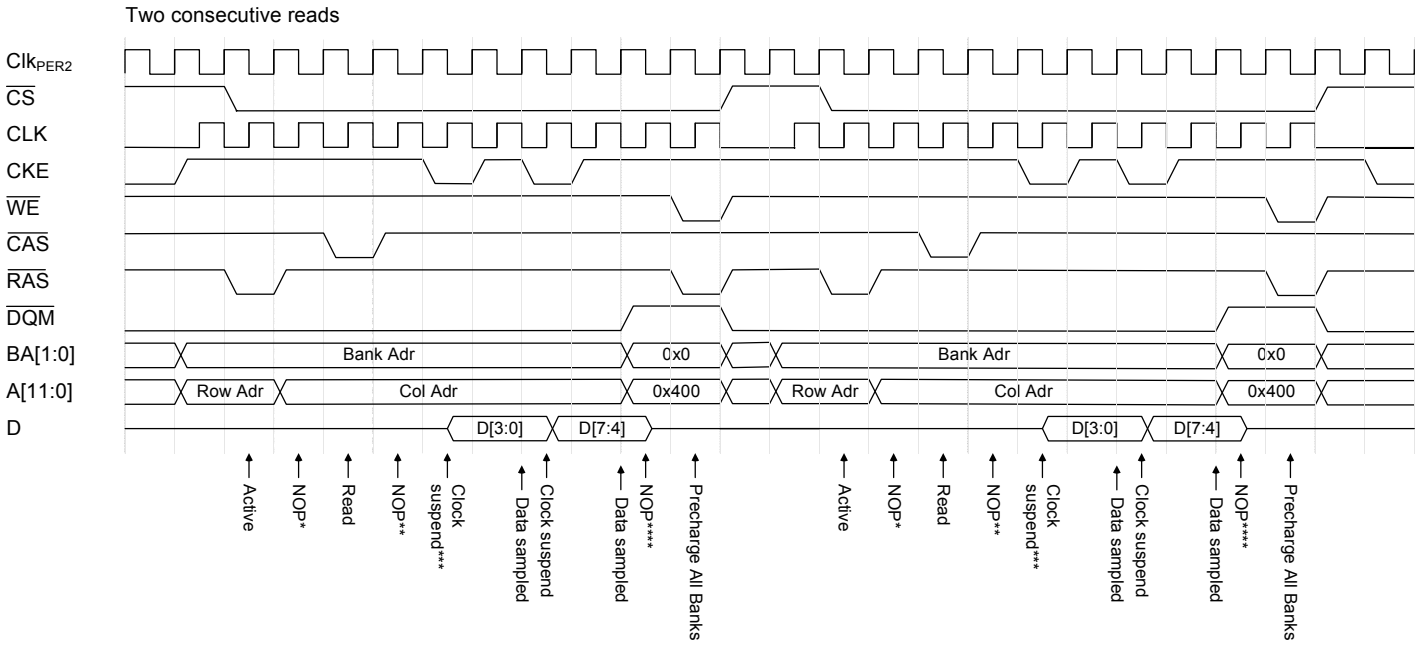
** NOP is only inserted for CAS3

*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-49. Two consecutive reads



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

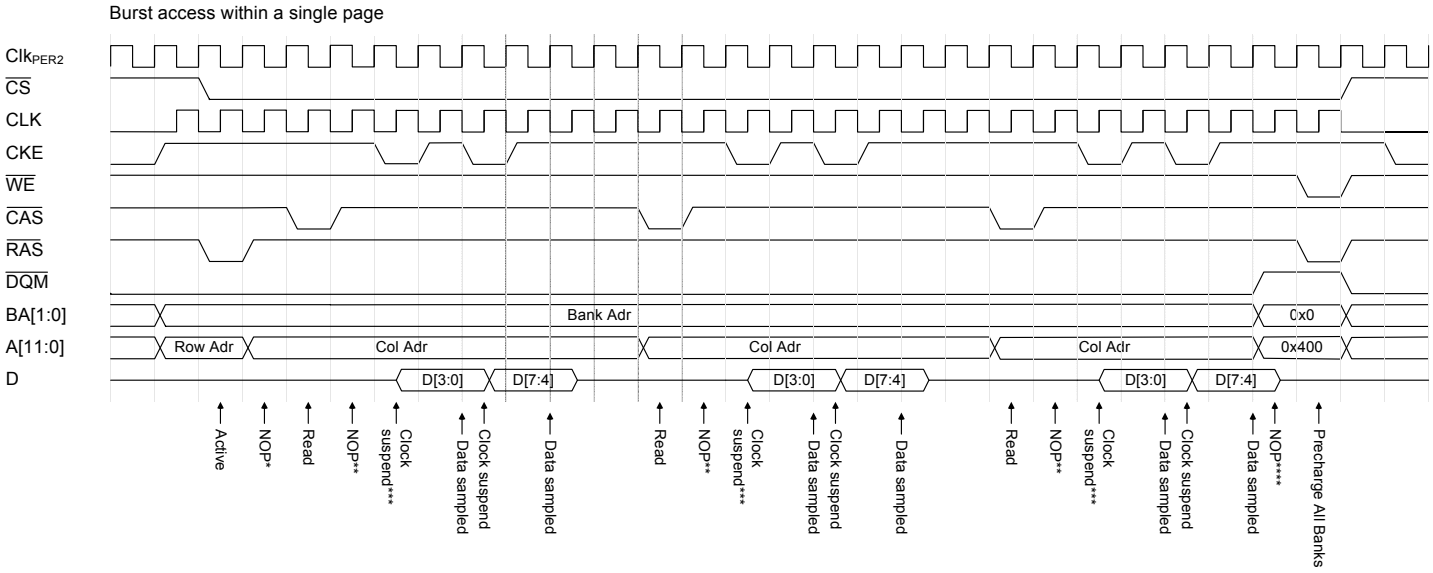
** NOP is only inserted for CAS3

*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-50. Burst access within a single page



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

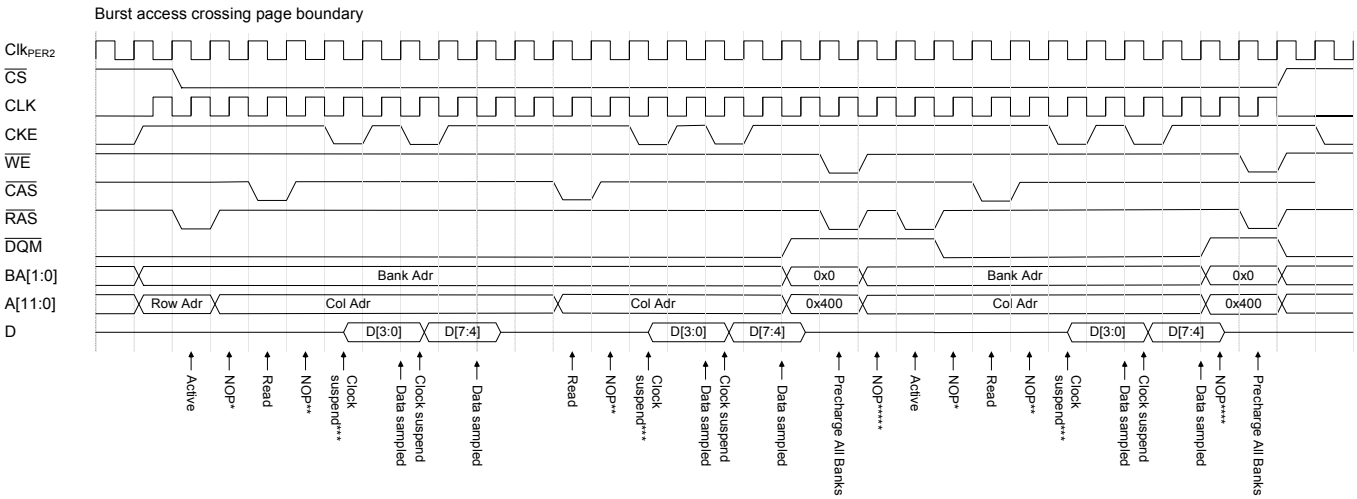
** NOP is only inserted for CAS3

*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

Figure 33-51. Burst access crossing page boundary



* The number of NOPs is equal to ROWCOLDLY[2:0] (ROWCOLDLY = 1 is shown)

** NOP is only inserted for CAS3

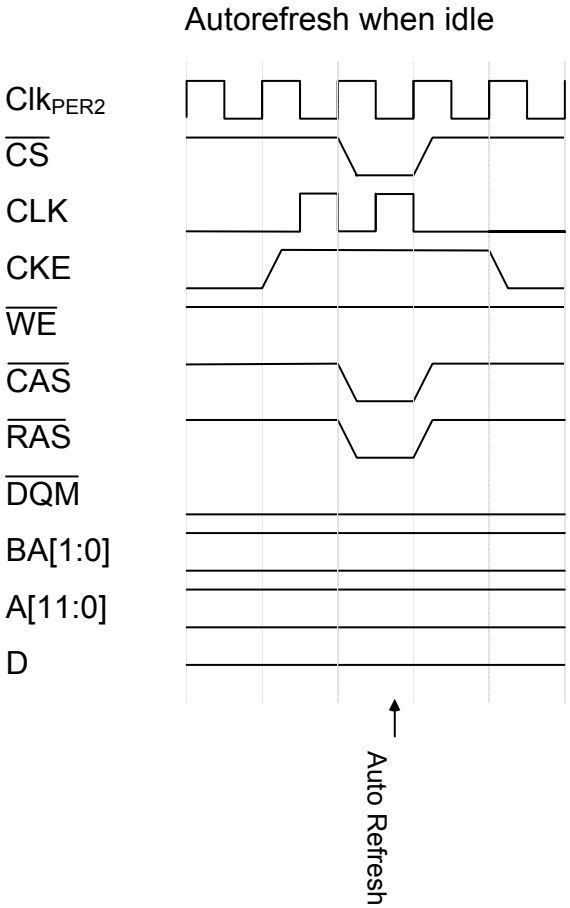
*** Clock suspended for 1 cycle when EBI is running at 1x and 1 or 2 cycles when EBI is running at 2x, to enable sampling of data on the positive edge of the 1x clock.

**** The number of NOPs is equal to WRDLY[1:0] + 1 (WRDLY = 0 is shown)

***** The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

33.16 SRAM refresh

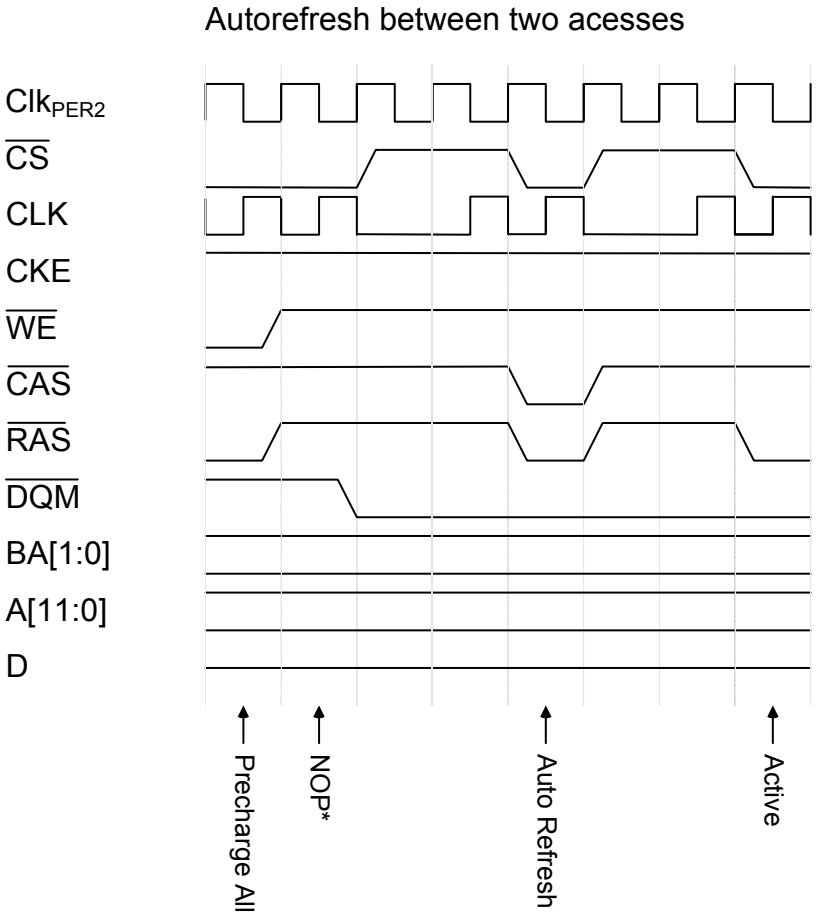
Figure 33-52. Autorefresh when idle



* The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

** The number of NOPs is equal to ESRDLY[2:0] (ESRDLY = 1 is shown)

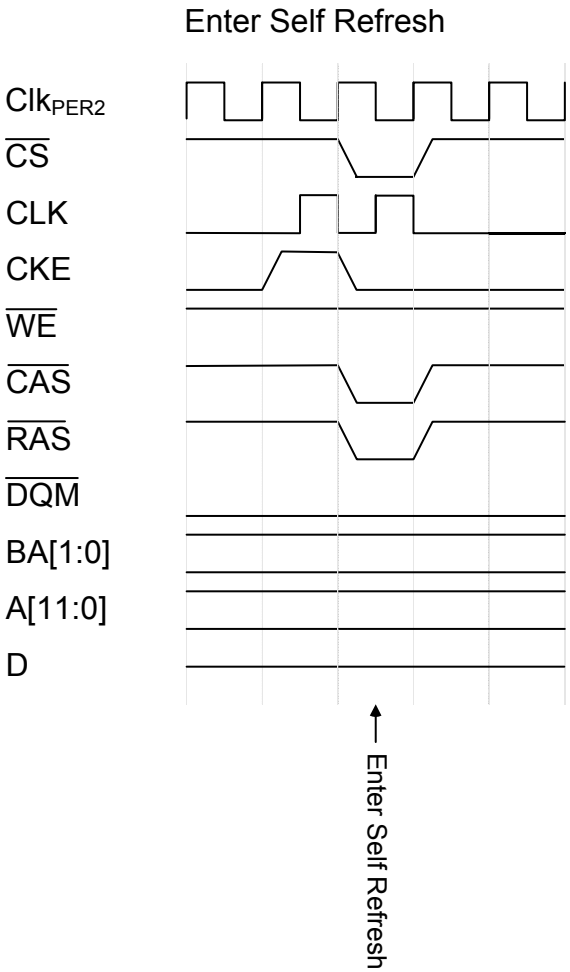
Figure 33-53. Autorefresh between two accesses



* The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

** The number of NOPs is equal to ESRDLY[2:0] (ESRDLY = 1 is shown)

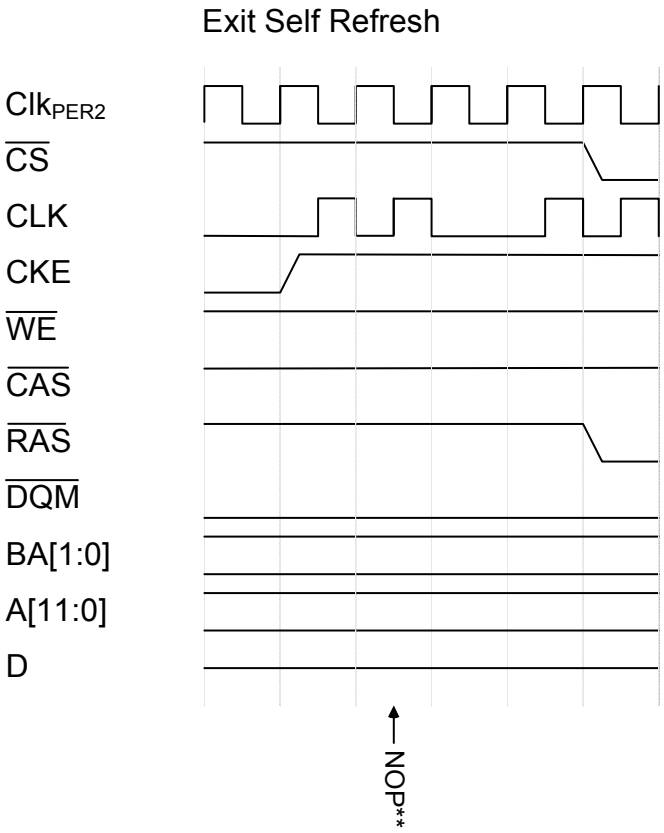
Figure 33-54. Enter Self Refresh



* The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

** The number of NOPs is equal to ESRDLY[2:0] (ESRDLY = 1 is shown)

Figure 33-55. Exit Self Refresh



* The number of NOPs is equal to RPDLY[1:0] (RPDLY = 1 is shown)

** The number of NOPs is equal to ESRDLY[2:0] (ESRDLY = 1 is shown)

34. 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.

34.1 8077H – 12/09

1. Updated "CTRLB - Non-Volatile Memory Control Register B" on page 27.
2. Removed Spike detector from XMEGA A Manual
3. Removed Table 4-9 on page 34.
4. Updated "System Clock and Clock options" on page 76.
5. Updated "Internal Oscillators" on page 78.
6. Updated Figure 7-5 on page 80.
7. Updated "Reset System" on page 102.
8. Updated Table 9-2 on page 106.
9. Updated "I/O Pin Configuration" on page 131.
10. Updated "Register Summary – Port Configuration" on page 148.
11. Updated "STATUS - Real Time Counter Status Register" on page 192.
12. Updated the description of "ADDR - TWI Slave Address Register" on page 226.
13. Updated "Register Description - TWI Master" on page 217
14. Changed in all datasheet Res: Reserved to Reserved.
15. Updated IRCOM registers in "Register Description" on page 249
16. Updated "ADC - Analog to Digital Converter" on page 289.
17. Updated "Program and Debug Interface" on page 344.

34.2 8077G – 04/09

1. Updated "Register Description – Fuses and Lockbits" on page 30.
2. Updated "Interrupt Vector Summary - NVM Controller" on page 48.
3. Updated "COMP2 - Oscillator Compare Register 2" on page 93.
4. Moved "TC - 16-bit Timer/Counter" on page 149 from section 26 to section 14.
5. Updated "Interrupt Vector Summary" on page 204.
6. Updated "EBI Timing" on page 277
7. Updated Figure 25-10 on page 295.
8. Updated "Appendix A: EBI Timing Diagrams" on page 392.

34.3 8077F – 02/09

1. Updated ”STATUS: Battery Backup Status Register” on page 115.
2. Updated Table 9-2 on page 106.
3. Updated ”Conversion Result” on page 294.
4. Added ADC section ”Interrupt vector Summary” on page 316.
5. Updated ”Overview” on page 149.
6. Updated ”Capture Channel” on page 156.
7. Updated data sheet cross-references.

34.4 8077E – 01/09

1. Updated Figure 9-5 External reset characteristics in Section 9.4.3 ”External reset” on page 107.
2. Updated ”Capture Channel” on page 156.
3. Added ADC section ”ADC Input Model” on page 300.
4. Added DAC section ”DAC Output model” on page 318.

34.5 8077D – 12/08

1. Updated the front page.
2. Updated Feature list in ”Memories” on page 18.
3. Updated ”Production Signature Row” on page 20 and inserted ”User Signature Row” on page 20.
4. Added Signtaure Row register description in Memory section, updated signature row summary with byte addresses.
5. Updated ”MCUCR – MCU Control Register” on page 44.
6 Updated lication of BODACT fuse bits in FUSEBYTE5 in Section 4.22 ”Register Summary - Fuses and Lockbits” on page 46.
7. Updated ”Interrupt Vector Summary - NVM Controller” on page 48.
8. Updated ”Priority Between Channels” on page 51.
9. Updated Overview in ”Event System” on page 65.
10. Updated ”Manually Generating Events” on page 67 and ”Event Routing Network” on page 67.
11. Updated ”Quadrature Decoder (QDEC)” on page 69.
12. Inserted a new section ”Battery Backup System” on page 111.
13. Updated title of the Table 13-7 on page 145: Event Channel 0 Output Configurations.
14. Updated Figure 13-1 ”General I/O pin functionality.” on page 130.
15. Updated Table 13-7 ”Event Channel 0 output configurations” on page 145.
16. Updated Figure 13-10 ”Port override signals and related logic” on page 137.

17. Added RTC section "Interrupts and events" on page 190.
18. Updated "Capture Channel" on page 156.
19. Updated "Frequency Capture" on page 157.
20. Updated "CTRLE - Control Register E" on page 167, "Register Summary" on page 174 and "TC - 16-bit Timer/Counter" on page 379 by removing DTHM-bit register.
21. Updated RTC32 "Overview" on page 196.
22. Inserted a new section "RTC32 - 32-bit Real Time Counter" on page 196.
23. Updated "STATUS– TWI Slave Status Register" on page 223, (Bit 7- DIF and Bit 6 - APIF).
24. Updated "IRCOM Mode of Operation" on page 248.
25. Updated "CTRLB - USART Control Register B" on page 250 (Bit 2).
26. Updated "TXPLCTRL - IRCOM Transmitter Pulse Length Control Register" on page 257 and "RXPLCTRL - IRCOM Receiver Pulse Length Control Register" on page 257 by inserting a paragraph note.
27. Updated Bit 2 - SDROW in "SDRAMCTRLA - SDRAM Control Register A" on page 279.
28. Updated EBI "CTRLA - Chip Select Control Register A" on page 283 (Bit 1:0)
29. Updated "Register Summary - EBI" on page 287 by inserting the page numbers in last column.
30. Added more details on unsigned input in the ADC section "Input sources" on page 289.
31. Added ADC result representation figures in section "Conversion Result" on page 294.
32. Added ADC section "Compare function" on page 295.
33. Updated ADC section "Calibration" on page 300.
34. Updated DAC section "Timing constraints" on page 317.
35. Added "Peripheral Module Address Map" on page 383.

34.6 8077C – 07/08

1. Updated Event System "Features" on page 57.
2. Updated "32 MHz Run-time Calibrated Internal Oscillator" on page 70.
3. Updated "STATUS - PMIC Status Register" on page 119.
4. Updated "Register Description" on page 173.
5. Updated "Register Summary" on page 178.
6. Updated "DES Instruction" on page 249.
7. Updated "SDRAMCTRLC - SDRAM Control Register C" on page 272.
8. Updated "TIMCTRL – DAC Timing Control Register" on page 309. Initial Value: 0110 0001.
9. Inserted general "Register Summary" on page 374.
10. Inserted Interrupt vectors in manual and "Interrupt Vector Summary" on page 383.
11. Inserted "Appendix A: EBI Timing Diagrams" on page 390.

34.7 8077B – 06/08

1. Updated “Overview” , “AVR CPU” , “DMAC - Direct Memory Access Controller” and “Memories” layout.
2. Updated bit names and register names in Section 5.14 ”Register Description – DMA Channel” on page 55.
3. Updated bit names and register names in Section 4.20 ”Register Description – MCU Control” on page 43.
4. Updated address register in Section 4.21 ”Register Summary - NVM Controller” on page 46.
5. Updated features in Section 6.1 ”Features” on page 65.
6. Updated bit name in Section 6.8.2 ”CHnCTRL – Event Channel n Control Register” on page 73.
7. Updated register format in Section 23.5 ”Register Description - AES” on page 263.
8. Updated register format, bit register and register names in Section 7. ”System Clock and Clock options” on page 76.
9. Updated the table layout in Table 8-1 on page 96.
10. Updated register description in Section 8.6 ”Register Description – Power Reduction” on page 98.
11. Updated register description, register names and bit register in Section 14. ”TC - 16-bit Timer/Counter” on page 149, in Section 15. ”AWeX – Advanced Waveform Extension” on page 175 and in Section 16. ”Hi-Res - High Resolution Extension” on page 187.
12. Updated register description, register names and bit register in Section 13. ”I/O Ports” on page 129, in Section 19. ”TWI – Two Wire Interface” on page 204, in Section 17. ”RTC - Real Time Counter” on page 189, in Section 20. ”SPI – Serial Peripheral Interface” on page 228, in Section 21. ”USART” on page 234, in Section 22. ”IRCOM - IR Communication Module” on page 255 and in Section 25. ”ADC - Analog to Digital Converter” on page 288.
13. Updated register description, register names and bit register in Section 26. ”DAC - Digital to Analog Converter” on page 316, in Section 27. ”AC - Analog Comparator” on page 328, in Section 28. ”IEEE 1149.1 JTAG Boundary Scan interface” on page 337, and in Section 29. ”Program and Debug Interface” on page 344.
14. Removed “Possibility to drive output to ground” from Section 26.1 ”Features” on page 316.
15. Added “Internal/External reference” in Section 26.1 ”Features” on page 316.
16. Updated “Bit utilization” in Section 26.9.6 ”STATUS – DAC Status Register” on page 322.
17. Removed parentheses from the section’s title Section 26.9.10 ”CH1DATAL – DAC Channel 1 Data Register Low byte” on page 325.
18. Added information text before the table in Section 26.10 ”Register Summary” on page 327.
19. Corrected the VSCALE formula in Section 27.9.4 ”CTRLB – Control Register B” on page 333.
20. Merged Section 23.3.2 and Section 23.3.3 in one Section 27.3.2 ”Internal Inputs” on page 330.
21. Changed “singed” to signed in Section 25.1 ”Features” on page 288.
22. Changed AWEXELOCK bit from position 4 to position 2 in Section 4.20.8 ”AWEXLOCK – Advanced Waveform Extension Lock Register” on page 45, and updated the entire section and Section 4.25 ”Register Summary - MCU Control” on page 48.
23. Added the last paragraph on Section 3.8 ”Stack and Stack Pointer” on page 9.
24. Inserted the correct Figure 14-1 on page 150.
25. Updated the Figure 14-2 on page 152.
26. Updated the Figure 13-10 on page 137.
27. Updated the Figure 25-18 on page 300 by removing the color.

28. Removed section 21.14.6. The removed section was only for the test.
29. Updated the Table 6-3 on page 71 with a footnote and cross-references to the footnote for PORTA_PINn to PORTF_PINn.
30. Changed connection description of CTRLA in Section 27.9.3 ”CTRLA – Control Register A” on page 333. AC0OUT is connected to pin 7, not on pin 0.
31. Updated the Section 4.1 ”Features” on page 18 with “Flexible Software CRC” in the feature list for the flash.
32. Updated Figure 7-1 on page 77.
33. Removed Figure 28-4 on page 343 and Figure 28-5 on page 343 and replaced by ones drawn in visio.
34. Updated Figure 4-1 on page 19, Figure 4-2 on page 22 and Figure 4-3 on page 24.
35. Updated Figure 19-11 on page 211.
36. Corrected the bit 0 in Section 8.6.1 ”PR - General Power Reduction Register” on page 98 and in Section 8.7 ”Register Summary - Sleep” on page 101.
37. Updated the text in Section 9.4.6 ”Software reset” on page 109.
38. Updated the table notes in the Table 7-6 on page 89.
39. Updated the Figure 6-1 on page 66 with IRCOM.
40. Updated the Figure 29-1 on page 345.
41. Changed the initial value to FF on the RTC PERH and PERL registers in Section 17.3.8 ”PERH - Real Time Counter Period Register High” on page 193 and in Section 17.3.9 ”PERL - Real Time Counter Period Register L” on page 193.
42. Added a footnote in Table 13-5 on page 143 that explains the low level.
43. Updated the high byte and low byte in Section 4.26 ”Interrupt Vector Summary - NVM Controller” on page 48.
44. Inserted a new Section 30. ”Memory Programming” on page 361.
45. Inserted a new figure Figure 2-1 on page 4.
46. Inserted a new figure Figure 3-1 on page 6.
47. Inserted new sections: ”EBI - External Bus Interface” on page 267, ”Memory Programming” on page 361, and ”Instruction Set Summary” on page 386.
48. Updated ”Virtual Registers” on page 138.
49. Deleted 2 Chapters: “Bootloader - Self-Programming” and Extern Programming.
50. Removed “TRUEGND” bit from ”Register Summary” on page 327.

34.8 8077A – 02/08

1. Initial revision

Table of Contents

1 About the Manual 2

1.1 Reading the Manual 2

1.2 Resources 2

1.3 Recommended Reading 2

2 Overview 3

2.1 Block Diagram 4

3 AVR CPU 5

3.1 Features 5

3.2 Overview 5

3.3 Architectural Overview 5

3.4 ALU - Arithmetic Logic Unit 7

3.5 Program Flow 7

3.6 Instruction Execution Timing 8

3.7 Status Register 9

3.8 Stack and Stack Pointer 9

3.9 Register File 9

3.10 RAMP and Extended Indirect Registers 11

3.11 Accessing 16-bits Registers 12

3.12 Configuration Change Protection 12

3.13 Fuse Lock 13

3.14 Register Description 13

3.15 Register Summary 17

4 Memories 18

4.1 Features 18

4.2 Overview 18

4.3 Flash Program Memory 19

4.4 Fuses and Lockbits 20

4.5 Data Memory 22

4.6 Internal SRAM 22

4.7 EEPROM 23

4.8 I/O Memory 23

4.9 External Memory 23

4.10 Data Memory and Bus Arbitration 23

4.11 Memory Timing 24
4.12 Device ID 24
4.13 JTAG Disable 24
4.14 IO Memory Protection 25
4.15 Register Description - NVM Controller 25
4.16 Register Description – Fuses and Lockbits 30
4.17 Register Description - Production Signature Row 36
4.18 Register Description – General Purpose I/O Memory 42
4.19 Register Description – External Memory 42
4.20 Register Description – MCU Control 42
4.21 Register Summary - NVM Controller 46
4.22 Register Summary - Fuses and Lockbits 46
4.23 Register Summary - Production Signature Row 47
4.24 Register Summary - General Purpose I/O Registers 48
4.25 Register Summary - MCU Control 48
4.26 Interrupt Vector Summary - NVM Controller 48

5 DMA - Direct Memory Access Controller 49

5.1 Features 49
5.2 Overview 49
5.3 DMA Transaction 50
5.4 Transfer Triggers 50
5.5 Addressing 51
5.6 Priority Between Channels 51
5.7 Double Buffering 51
5.8 Transfer Buffers 51
5.9 Error detection 52
5.10 Software Reset 52
5.11 Protection 52
5.12 Interrupts 52
5.13 Register Description – DMA Controller 53
5.14 Register Description – DMA Channel 55
5.15 Register Summary – DMA Controller 64
5.16 Register Summary – DMA Channel 64
5.17 DMA Interrupt Vector Summary 64

6 Event System 65

6.1 Features 65
6.2 Overview 65
6.3 Events 66
6.4 Event Routing Network 67
6.5 Event Timing 69
6.6 Filtering 69
6.7 Quadrature Decoder 69
6.8 Register Description 71
6.9 Register Summary 75

7 System Clock and Clock options 76

7.1 Features 76
7.2 Overview 76
7.3 Clock Distribution 77
7.4 Clock Sources 78
7.5 System Clock Selection and Prescalers 80
7.6 PLL with 1-31x Multiplication Factor 81
7.7 DFLL 2 MHz and DFLL 32 MHz 81
7.8 External Clock Source Failure Monitor 82
7.9 Register Description - Clock 84
7.10 Register Description - Oscillator 87
7.11 Register Description - DFLL32M/DFLL2M 92
7.12 Register Summary - Clock 94
7.13 Register Summary - Oscillator 94
7.14 Register Summary - DFLL32M/DFLL2M 94
7.15 Crystal Oscillator Failure Interrupt Vector Summary 94

8 Power Management and Sleep 95

8.1 Features 95
8.2 Overview 95
8.3 Sleep Modes 95
8.4 Power Reduction Registers 97
8.5 Register Description – Sleep 97
8.6 Register Description – Power Reduction 98
8.7 Register Summary - Sleep 101
8.8 Register Summary - Power Reduction 101

9 Reset System 102

9.1 Features 102
9.2 Overview 102
9.3 Reset Sequence 104
9.4 Reset Sources 104
9.5 Register Description 109
9.6 Register Summary 110

10 Battery Backup System 111

10.1 Features 111
10.2 Overview 111
10.3 Battery Backup Module 112
10.4 Main Power Loss 113
10.5 Main Reset and Start-up Sequence 113
10.6 Register Description 114
10.7 Register Summary 116

11 WDT – Watchdog Timer 117

11.1 Features 117
11.2 Overview 117
11.3 Normal Mode Operation 117
11.4 Window Mode Operation 118
11.5 Watchdog Timer clock 118
11.6 Configuration Protection and Lock 118
11.7 Registers Description 119
11.8 Register Summary 122

12 Interrupts and Programmable Multi-level Interrupt Controller 123

12.1 Features 123
12.2 Overview 123
12.3 Operation 123
12.4 Interrupts 124
12.5 Interrupt level 125
12.6 Interrupt priority 125
12.7 Moving Interrupts Between Application and Boot Section 127
12.8 Register Description 127
12.9 Register Summary 128

13 I/O Ports 129

13.1 Features 129

13.2 Overview 129
13.3 Using the I/O Pin 130
13.4 I/O Pin Configuration 131
13.5 Reading the Pin value 133
13.6 Input Sense Configuration 134
13.7 Port Interrupt 135
13.8 Port Event 136
13.9 Alternate Port Functions 136
13.10 Slew-rate Control 137
13.11 Clock and Event Output 137
13.12 Multi-configuration 138
13.13 Virtual Port Registers 138
13.14 Register Description – Ports 138
13.15 Register Description – Multiport Configuration 143
13.16 Register Description – Virtual Port 146
13.17 Register Summary – Ports 148
13.18 Register Summary – Port Configuration 148
13.19 Register Summary – Virtual Ports 148
13.20 Interrupt vector Summary - Ports 148

14 TC - 16-bit Timer/Counter 149

14.1 Features 149
14.2 Overview 149
14.3 Block Diagram 151
14.4 Clock and Event Sources 153
14.5 Double Buffering 153
14.6 Counter Operation 154
14.7 Capture Channel 156
14.8 Compare Channel 159
14.9 Interrupts and events 163
14.10 DMA Support. 164
14.11 Timer/Counter Commands 164
14.12 Register Description 165
14.13 Register Summary 175
14.14 Interrupt Vector Summary 175

15 AWeX – Advanced Waveform Extension 176

15.1 Features 176
15.2 Overview 176
15.3 Port Override 177
15.4 Dead Time Insertion 179
15.5 Pattern Generation 180
15.6 Fault Protection 181
15.7 Register Description 182
15.8 Register Summary 187

16 Hi-Res - High Resolution Extension 188

16.1 Features 188
16.2 Overview 188
16.3 Register Description 189
16.4 Register Summary 189

17 RTC - Real Time Counter 190

17.1 Features 190
17.2 Overview 190
17.3 Register Description 191
17.4 Register Summary 196
17.5 Interrupt Vector Summary 196

18 RTC32 - 32-bit Real Time Counter 197

18.1 Features 197
18.2 Overview 197
18.3 Register Description 198
18.4 Register Summary 204
18.5 Interrupt Vector Summary 204

19 TWI – Two Wire Interface 205

19.1 Features 205
19.2 Overview 205
19.3 General TWI Bus Concepts 206
19.4 TWI Bus State Logic 212
19.5 TWI Master Operation 213
19.6 TWI Slave Operation 215
19.7 Enabling External Driver Interface 216
19.8 Register Description - TWI 217
19.9 Register Description - TWI Master 217

19.10 Register Description - TWI Slave 223
19.11 Register Summary - TWI 228
19.12 Register Summary - TWI Master 228
19.13 Register Summary - TWI Slave 228
19.14 Interrupt Vector Summary 228

20 SPI – Serial Peripheral Interface 229

20.1 Features 229
20.2 Overview 229
20.3 Master Mode 230
20.4 Slave Mode 230
20.5 Data Modes 231
20.6 DMA Support 232
20.7 Register Description 232
20.8 Register Summary 234
20.9 SPI Interrupt vectors 234

21 USART 235

21.1 Features 235
21.2 Overview 235
21.3 Clock Generation 237
21.4 Frame Formats 240
21.5 USART Initialization 241
21.6 Data Transmission - The USART Transmitter 241
21.7 Data Reception - The USART Receiver 242
21.8 Asynchronous Data Reception 243
21.9 The Impact of Fractional Baud Rate Generation 246
21.10 USART in Master SPI Mode 247
21.11 USART SPI vs. SPI 247
21.12 Multi-processor Communication Mode 248
21.13 IRCOM Mode of Operation 249
21.14 DMA Support 249
21.15 Register Description 249
21.16 Register Summary 255
21.17 Interrupt Vector Summary 255

22 IRCOM - IR Communication Module 256

22.1 Features 256

22.2 Overview 256
22.3 Registers Description 258
22.4 Register Summary 259

23 Crypto Engines 260

23.1 Features 260
23.2 Overview 260
23.3 DES Instruction 260
23.4 AES Crypto Module 261
23.5 Register Description - AES 264
23.6 Register Summary - AES 267
23.7 AES Interrupt vector 267

24 EBI - External Bus Interface 268

24.1 Features 268
24.2 Overview 268
24.3 Chip Select 268
24.4 I/O Pin Configuration 269
24.5 EBI Clock 270
24.6 SRAM Configuration 270
24.7 SRAM LPC Configuration 272
24.8 SDRAM Configuration 273
24.9 Combined SRAM & SDRAM Configuration 275
24.10 EBI Timing 277
24.11 Register Description - EBI 279
24.12 Register Description - EBI Chip Select 284
24.13 Register Summary - EBI 288
24.14 Register Summary - EBI Chip Select 288

25 ADC - Analog to Digital Converter 289

25.1 Features 289
25.2 Overview 289
25.3 Input sources 290
25.4 ADC Channels 293
25.5 Voltage reference selection 294
25.6 Conversion Result 294
25.7 Compare function 296
25.8 Starting a conversion 296

25.9 ADC Clock and Conversion Timing 296
25.10 ADC Input Model 300
25.11 DMA transfer 301
25.12 Interrupts and events 301
25.13 Calibration 301
25.14 Channel priority 301
25.15 Synchronous sampling 301
25.16 Register Description - ADC 302
25.17 Register Description - ADC Channel 309
25.18 Register Summary - ADC 315
25.19 Register Summary - ADC Channel 315
25.20 Interrupt vector Summary 316

26 DAC - Digital to Analog Converter 317

26.1 Features 317
26.2 Overview 317
26.3 Starting a conversion 318
26.4 Output channels 318
26.5 DAC Output model 318
26.6 DAC clock 319
26.7 Timing constraints 319
26.8 Low Power mode 319
26.9 Calibration 319
26.10 Register Description 320
26.11 Register Summary 327

27 AC - Analog Comparator 328

27.1 Features 328
27.2 Overview 328
27.3 Input Channels 330
27.4 Start of Signal Compare 330
27.5 Generating Interrupts and Events 330
27.6 Window Mode 330
27.7 Input hysteresis 331
27.8 Power consumption vs. propagation delay 331
27.9 Register Description 331
27.10 Register Summary 336

27.11 Interrupt vector Summary 336

28 IEEE 1149.1 JTAG Boundary Scan interface 337

28.1 Features 337
28.2 Overview 337
28.3 TAP - Test Access Port 338
28.4 JTAG instructions 339
28.5 Data registers 341
28.6 Boundary-scan chain 342

29 Program and Debug Interface 344

29.1 Features 344
29.2 Overview 344
29.3 PDI Physical 345
29.4 JTAG Physical 350
29.5 PDI Controller 353
29.6 Register Description - PDI Instruction and Addressing Registers 357
29.7 Register Description - PDI Control and Status Register 359
29.8 Register Summary 361

30 Memory Programming 362

30.1 Features 362
30.2 Overview 362
30.3 NVM Controller 363
30.4 NVM Commands 363
30.5 NVM Controller Busy 363
30.6 Flash and EEPROM Page Buffers 364
30.7 Flash and EEPROM Programming Sequences 365
30.8 Protection of NVM 366
30.9 Preventing NVM Corruption 366
30.10 CRC Functionality 366
30.11 Self-Programming and Boot Loader Support 367
30.12 External Programming 379
30.13 Register Description 385
30.14 Register Summary 385

31 Peripheral Module Address Map 386

32 Instruction Set Summary 388

33 Appendix A: EBI Timing Diagrams 392

33.1 SRAM 3-Port ALE1 CS 392
33.2 SRAM 3-Port ALE12 CS 394
33.3 SRAM 4 -Port ALE2 CS 397
33.4 SRAM 4 - Port NOALE CS 399
33.5 LPC 2 - Port ALE12 CS 400
33.6 LPC 3 - Port ALE1 CS 402
33.7 LPC 2 - Port ALE1 CS 403
33.8 SRAM 3 - Port ALE1 noCS 404
33.9 SRAM 4 - Port NOALE no CS 406
33.10 LPC 2 - Port ALE12 no CS 407
33.11 SDRAM init 409
33.12 SDRAM 8-bit rwite 410
33.13 SDRAM 8-bit read 414
33.14 SDRAM 4-bit rwite 418
33.15 SDRAM 4-bit read 422
33.16 SRAM refresh 425

34 Datasheet Revision History 429

34.1 8077H – 12/09 429
34.2 8077G – 04/09 429
34.3 8077F – 02/09 430
34.4 8077E – 01/09 430
34.5 8077D – 12/08 430
34.6 8077C – 07/08 431
34.7 8077B – 06/08 432
34.8 8077A – 02/08 433

Table of Contents..... i

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2009 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others are registered trademarks, XMEGA™ and others are trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.