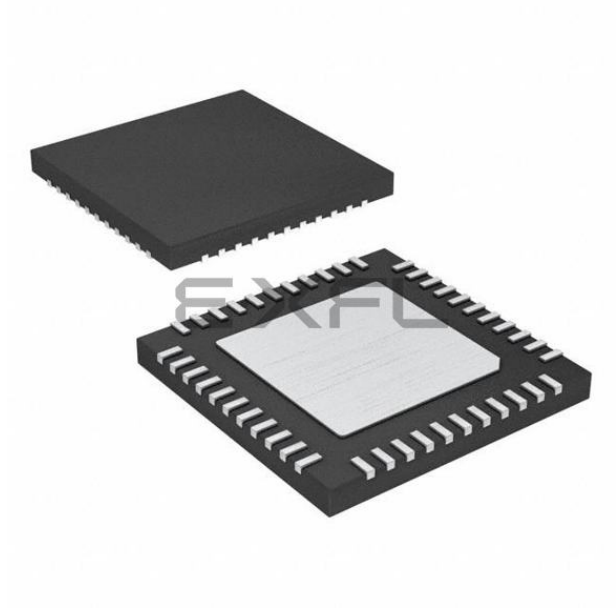


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8/16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	34
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.6V ~ 3.6V
Data Converters	A/D 12x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	44-VQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atxmega64d4-mnr

4.12.9 CTRLB – Control Register B

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

- **Bit 7:4 – Reserved**

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

- **Bit 3 – EEMAPEN: EEPROM Data Memory Mapping Enable**

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

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

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

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

Setting this bit enables power saving for the EEPROM. The EEPROM will then be turned off in a manner equal to entering sleep mode. If access is required, the bus master will be halted for a time equal to the start-up time from idle sleep mode.

- **Bit 0 – SPMLOCK: SPM Locked**

This bit can be written to prevent all further self-programming. The bit is cleared at reset, and cannot be cleared from software. This bit is protected by the configuration change protection (CCP) mechanism. Refer to “Configuration Change Protection” on page 13 for details on the CCP.

4.12.10 INTCTRL – Interrupt Control Register

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

- **Bit 7:4 – Reserved**

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

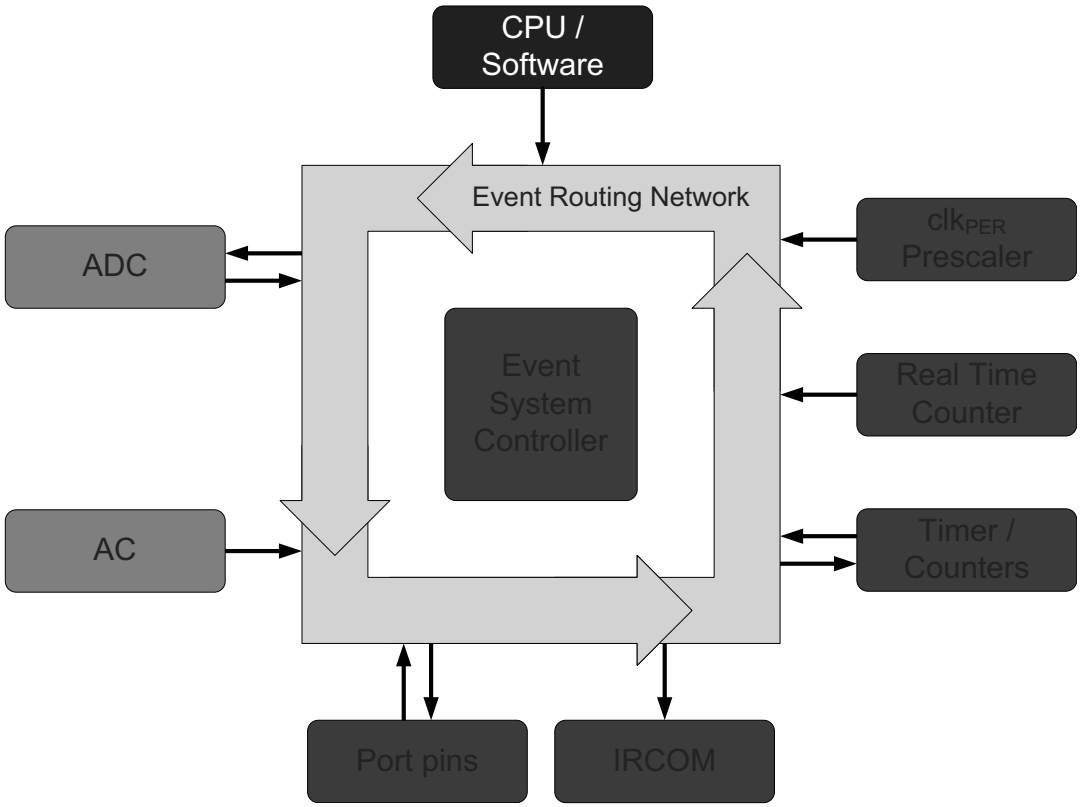
- **Bit 3:2 – SPMLVL[1:0]: SPM Ready Interrupt Level**

These bits enable the interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 94. This is a level interrupt that will be triggered only when the NVMBUSY flag in the STATUS register is set to zero. Thus, the interrupt should not be enabled before triggering an NVM command, as the NVMBUSY flag will not be set before the NVM command is triggered. The interrupt should be disabled in the interrupt handler.

- **Bit 1:0 – EELVL[1:0]: EEPROM Ready Interrupt Level**

These bits enable the EEPROM ready interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 94. This is a level interrupt that will be triggered only when the NVMBUSY flag in the STATUS register is set to zero. Thus, the interrupt should not be enabled before triggering an NVM command, as the NVMBUSY flag will not be set before the NVM command is triggered. The interrupt should be disabled in the interrupt handler.

Figure 5-1. Event System Overview and Connected Peripherals



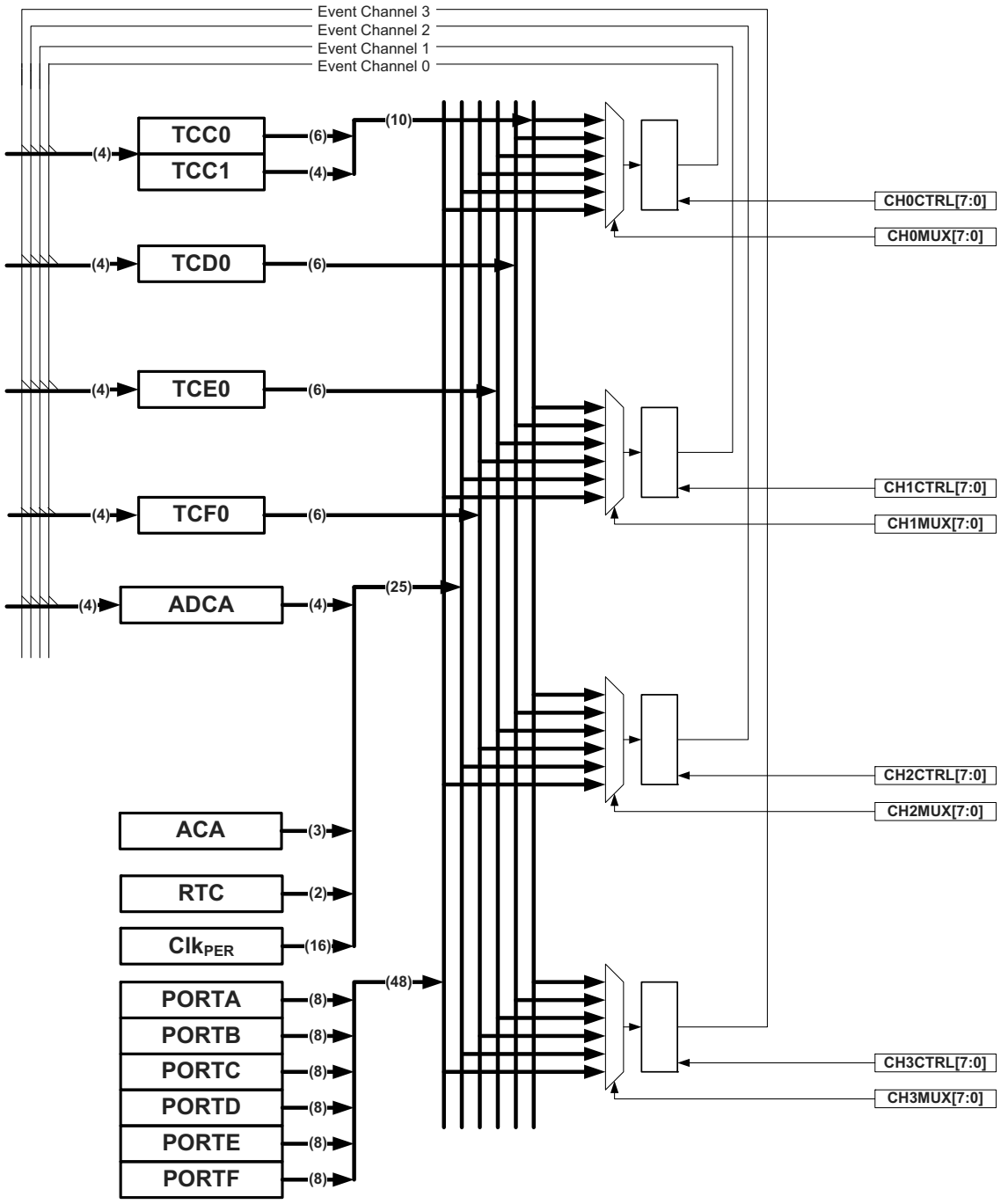
The event routing network consists of four software-configurable multiplexers that control how events are routed and used. These are called event channels, and allow for up to four parallel event configurations and routings. The maximum routing latency is two peripheral clock cycles. The event system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

5.3 Events

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

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

Figure 5-3. Event Routing Network



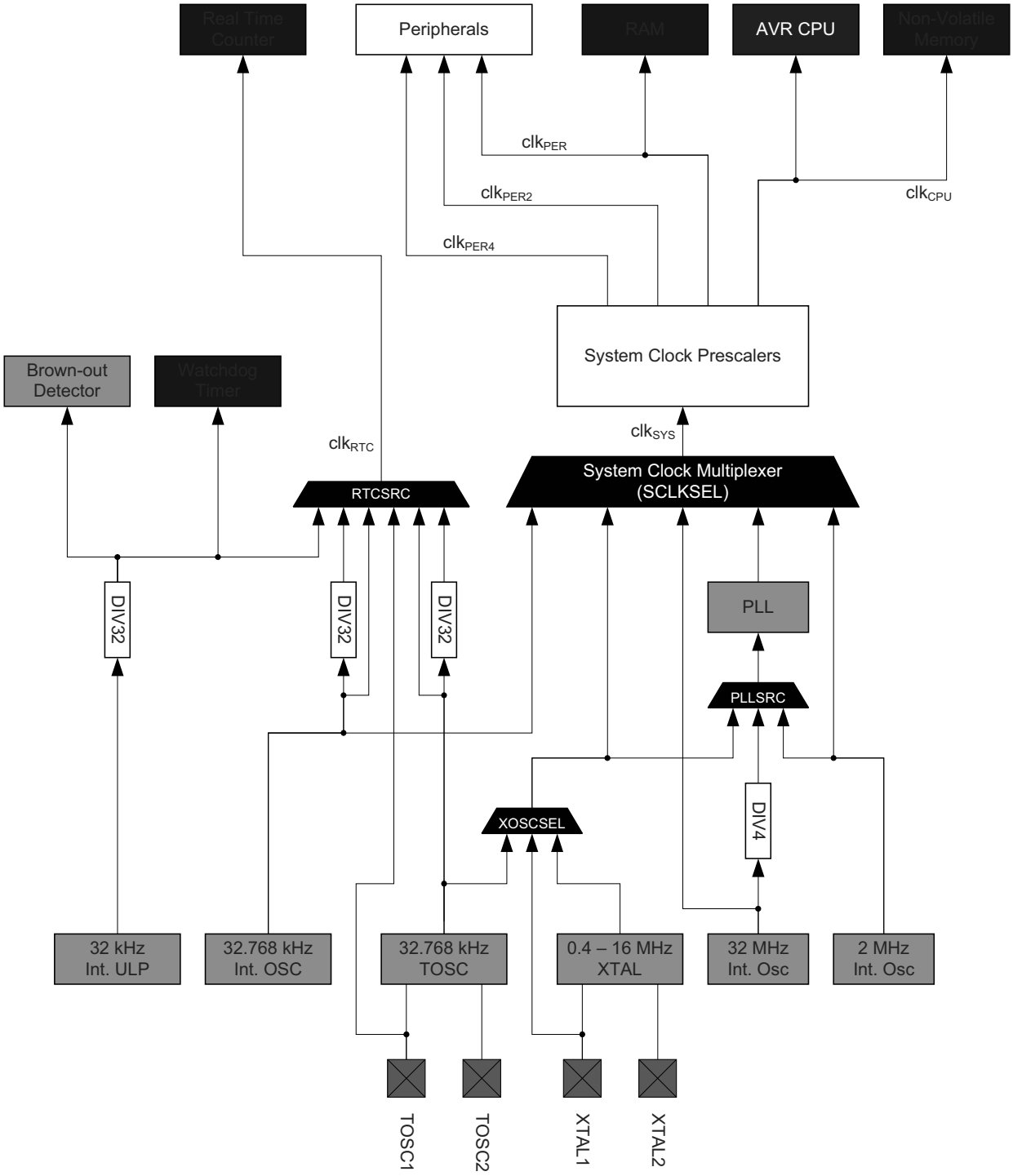
Four multiplexers means that it is possible to route up to four events at the same time. It is also possible to route one event through several multiplexers.

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

5.5 Event Timing

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

Figure 6-1. The Clock System, Clock Sources, and Clock Distribution



6.10 Register Description – Oscillator

6.10.1 CTRL – Oscillator Control Register

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

- **Bit 7:5 – Reserved**

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

- **Bit 4 – PLLEN: PLL Enable**

Setting this bit enables the PLL. Before the PLL is enabled, it must be configured with the desired multiplication factor and clock source. See ”STATUS – Oscillator Status Register” on page 67.

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

Setting this bit enables the selected external clock source. Refer to “XOSCCTRL – XOSC Control Register” on page 68 for details on how to select the external clock source. The external clock source should be allowed time to stabilize before it is selected as the source for the system clock. See ”STATUS – Oscillator Status Register” on page 67.

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

Setting this bit enables the 32.768kHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See ”STATUS – Oscillator Status Register” on page 67.

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

Setting this bit will enable the 32MHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See ”STATUS – Oscillator Status Register” on page 67.

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

Setting this bit enables the 2MHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See ”STATUS – Oscillator Status Register” on page 67.

By default, the 2MHz internal oscillator is enabled and this bit is set.

6.10.2 STATUS – Oscillator Status Register

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

- **Bit 7:5 – Reserved**

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

- **Bit 4 – PLLRDY: PLL Ready**

This flag is set when the PLL has locked on the selected frequency and is ready to be used as the system clock source.

- **Bit 3 – XOSCRDY: External Clock Source Ready**

This flag is set when the external clock source is stable and is ready to be used as the system clock source.

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

This flag is set when the 32.768kHz internal oscillator is stable and is ready to be used as the system clock source.

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

This flag is set when the 32MHz internal oscillator is stable and is ready to be used as the system clock source.

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

This flag is set when the 2MHz internal oscillator is stable and is ready to be used as the system clock source.

6.10.3 XOSCCTRL – XOSC Control Register

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

- **Bit 7:6 – FRQRANGE[1:0]: 0.4 - 16MHz Crystal Oscillator Frequency Range Select**

These bits select the frequency range for the connected crystal oscillator according to Table 6-5.

Table 6-5. 16MHz Crystal Oscillator Frequency Range Selection

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

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

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

- **Bit 4 – XOSCPWR: Crystal Oscillator Drive**

Setting this bit will increase the current in the 0.4MHz - 16MHz crystal oscillator and increase the swing on the XTAL2 pin. This allows for driving crystals with higher load or higher frequency than specified by the FRQRANGE bits.

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

These bits select the type and start-up time for the crystal or resonator that is connected to the XTAL or TOSC pins. See Table 6-6 for crystal selections. If an external clock or external oscillator is selected as the source for the system clock, see “CTRL – Oscillator Control Register” on page 67. This configuration cannot be changed.

Table 6-6. External Oscillator Selection and Start-up Time

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

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

6.12 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]			64
+0x01	PSCTRL	–	PSADIV[4:0]					PSBCDIV[1:0]		64
+0x02	LOCK	–	–	–	–	–	–	–	LOCK	66
+0x03	RTCCTRL	–	–	–	–	RTCSRC[2:0]			RTCEN	66
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	

6.13 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	67
+0x01	STATUS	–	–	–	PLLRDY	XOSCRDY	RC32KRD	R32MRDY	RC2MRDY	67
+0x02	XOSCCTR	FRQRANGE[1:0]		X32KLPM	XOSCPW	XOSCSEL[3:0]				68
+0x03	XOSCFAIL	–	–	–	–	PLLFDIF	PLLFDEN	XOSCFDIF	XOSCFDEN	69
+0x04	RC32KCAL	RC32KCAL[7:0]								69
+0x05	PLLCTRL	PLLSRC[1:0]		–	PLLFAC[4:0]					69
+0x06	DFLLCTRL	–	–	–	–	–	RC32MCREF[1:0]		RC2MCREF	70
+0x07	Reserved	–	–	–	–	–	–	–	–	

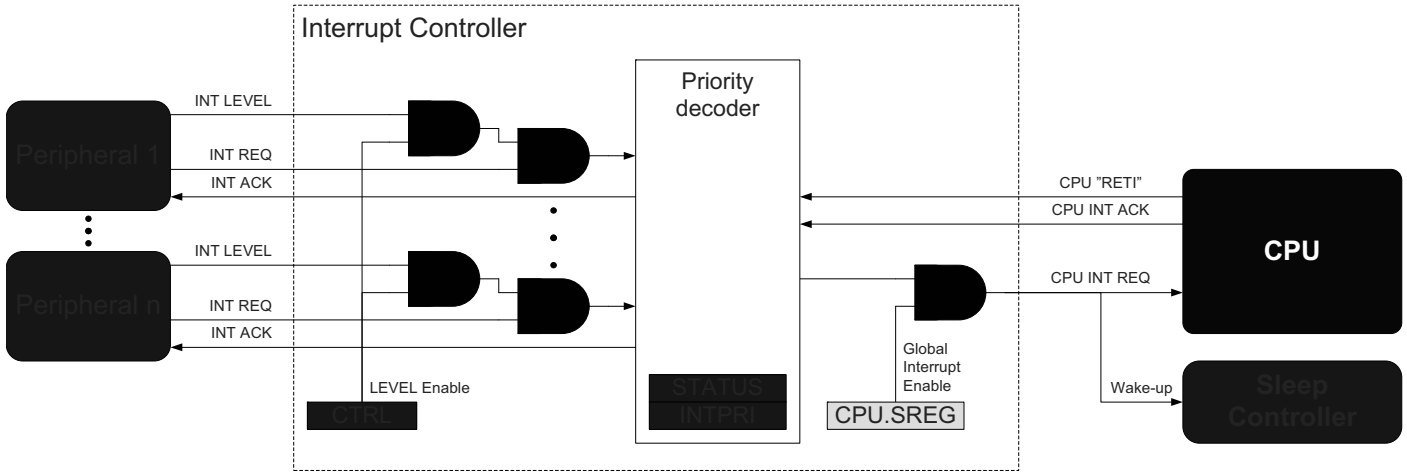
6.14 Register Summary - DFLL32M/DFLL2M

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRL	–	–	–	–	–	–	–	ENABLE	71
+0x01	Reserved	–	–	–	–	–	–	–	–	
+0x02	CALA	–	CALA[6:0]							71
+0x03	CALB	–	–	CALB[5:0]						71
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	COMP1	COMP[7:0]								72
+0x06	COMP2	COMP[15:8]								72
+0x07	Reserved	–	–	–	–	–	–	–	–	

6.15 Oscillator Failure Interrupt Vector Summary

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

Figure 10-1. Interrupt Controller Overview



10.4 Interrupts

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

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

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

Interrupts can be blocked when executing code from a locked section; e.g., when the boot lock bits are programmed. This feature improves software security. Refer to “Memory Programming” on page 274 for details on lock bit settings.

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

10.4.1 NMI – Non-Maskable Interrupts

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

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

10.4.2 Interrupt Response Time

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

10.5 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 10-1.

Table 10-1. Interrupt Levels

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

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

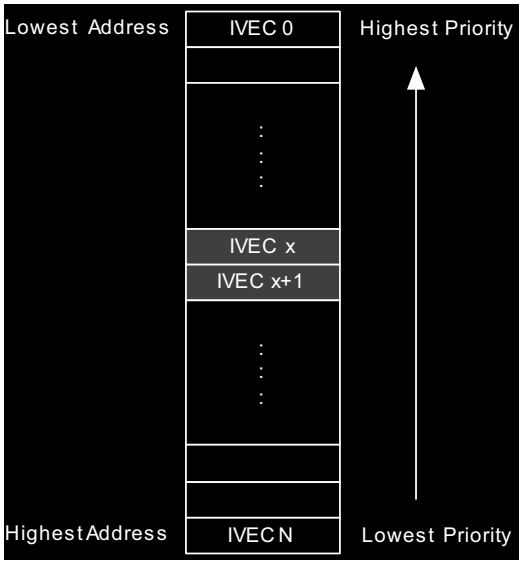
10.6 Interrupt Priority

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

10.6.1 Static Priority

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

Figure 10-3. Static Priority



- **Bit 2:0 – EVOUTSEL[2:0]: Event Channel Output Selection**

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

Table 11-11. Event Channel Output Selection

EVOUTSEL[2:0]	Group configuration	Description
000	0	Event channel 0 output to pin
001	1	Event channel 1 output to pin
010	2	Event channel 2 output to pin
011	3	Event channel 3 output to pin
100	4	Event channel 4 output to pin
101	5	Event channel 5 output to pin
110	6	Event channel 6 output to pin
111	7	Event channel 7 output to pin

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

12.1 Features

- 16-bit timer/counter
- 32-bit timer/counter support by cascading two timer/counters
- Up to four compare or capture (CC) channels
  - Four CC channels for timer/counters of type 0
  - Two CC channels for timer/counters of type 1
- Double buffered timer period setting
- Double buffered capture or compare channels
- Waveform generation:
  - Frequency generation
  - Single-slope pulse width modulation
  - Dual-slope pulse width modulation
- Input capture:
  - Input capture with noise cancelling
  - Frequency capture
  - Pulse width capture
  - 32-bit input capture
- Timer overflow and error interrupts/events
- One compare match or input capture interrupt/event per CC channel
- Can be used with event system for:
  - Quadrature decoding
  - Count and direction control
  - Capture
- High-resolution extension
  - Increases frequency and waveform resolution by 4x (2-bit) or 8x (3-bit)
- Advanced waveform extension:
  - Low- and high-side output with programmable dead-time insertion (DTI)
  - Event controlled fault protection for safe disabling of drivers

12.2 Overview

Atmel AVR XMEGA devices have a set of flexible, 16-bit timer/counters (TC). Their capabilities include accurate program execution timing, frequency and waveform generation, and input capture with time and frequency measurement of digital signals. Two timer/counters can be cascaded to create a 32-bit timer/counter with optional 32-bit capture.

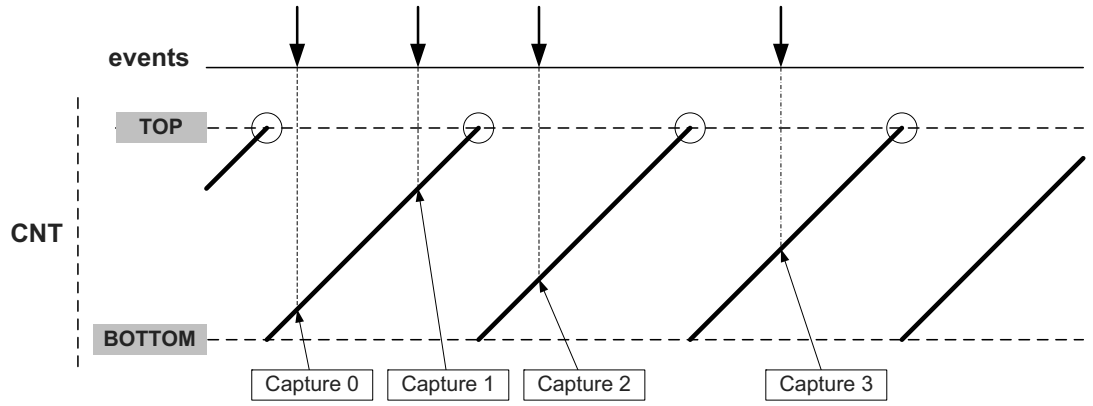
A timer/counter consists of a base counter and a set of compare or capture (CC) channels. The base counter can be used to count clock cycles or events. It has direction control and period setting that can be used for timing. The CC channels can be used together with the base counter to do compare match control, frequency generation, and pulse width waveform modulation, as well as various input capture operations. A timer/counter can be configured for either capture or compare functions, but cannot perform both at the same time.

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

There are two differences between timer/counter type 0 and type 1. Timer/counter 0 has four CC channels, and timer/counter 1 has two CC channels. All information related to CC channels 3 and 4 is valid only for timer/counter 0. Only Timer/Counter 0 has the split mode feature that split it into 2 8-bit Timer/Counters with four compare channels each.

Some timer/counters have extensions to enable more specialized waveform and frequency generation. The advanced waveform extension (AWeX) is intended for motor control and other power control applications. It enables low- and high-side output with dead-time insertion, as well as fault protection for disabling and shutting down external drivers. It can

Figure 12-11.Input Capture Timing



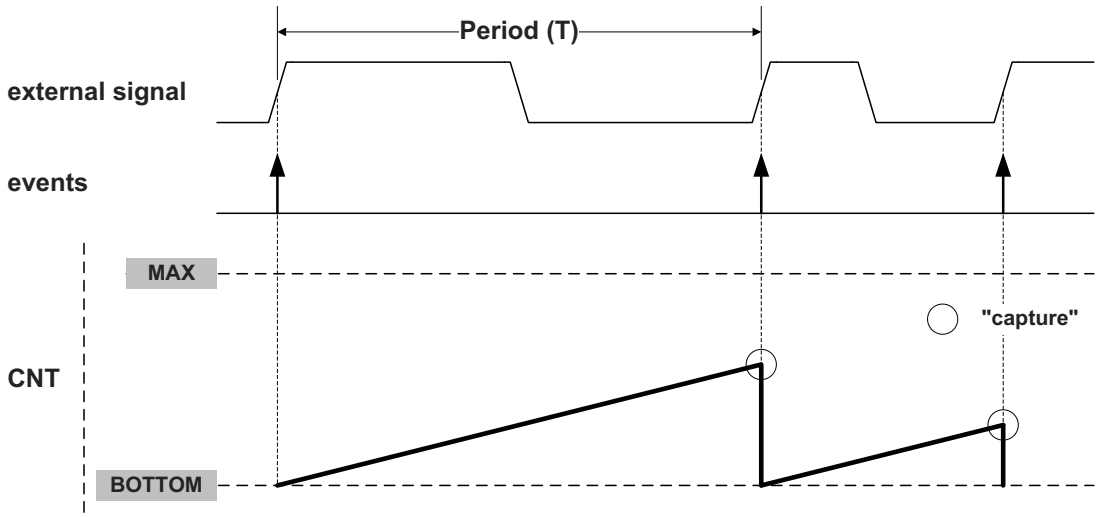
12.7.2 Frequency Capture

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

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

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

Figure 12-12.Frequency Capture of an External Signal



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

12.7.3 Pulse Width Capture

Selecting the pulse width measure event action makes the enabled compare channel perform the input capture action on falling edge events and the restart action on rising edge events. The counter will then restart on positive edge events, and the input capture will be performed on the negative edge event. The event source must be an I/O pin, and the sense configuration for the pin must be set to generate an event on both edges. Figure 12-13 on page 131 shows and example where the pulse width is measured twice for an external signal.

12.11 Register Description

12.11.1 CTRLA – Control Register A

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

- **Bit 7:4 – Reserved**

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

- **Bit 3:0 – CLKSEL[3:0]: Clock Select**

These bits select the clock source for the timer/counter according to Table 12-2.

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

Table 12-2. Clock Select Options

CLKSEL[3:0]	Group configuration	Description
0000	OFF	None (i.e, timer/counter in OFF state)
0001	DIV1	Prescaler: Clk
0010	DIV2	Prescaler: Clk/2
0011	DIV4	Prescaler: Clk/4
0100	DIV8	Prescaler: Clk/8
0101	DIV64	Prescaler: Clk/64
0110	DIV256	Prescaler: Clk/256
0111	DIV1024	Prescaler: Clk/1024
1nnn	EVCHn	Event channel n, n= [0,...,7]

12.11.2 CTRLB – Control Register B

Bit	7	6	5	4	3	2	1	0
+0x01	CCDEN	CCCEN	CCBEN	CCAEN	–	WGMODE[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – CCxEN: Compare or Capture Enable**

Setting these bits in the FRQ or PWM waveform generation mode of operation will override the port output register for the corresponding OCn output pin.

When input capture operation is selected, the CCxEN bits enable the capture operation for the corresponding CC channel.

- **Bit 3 – Reserved**

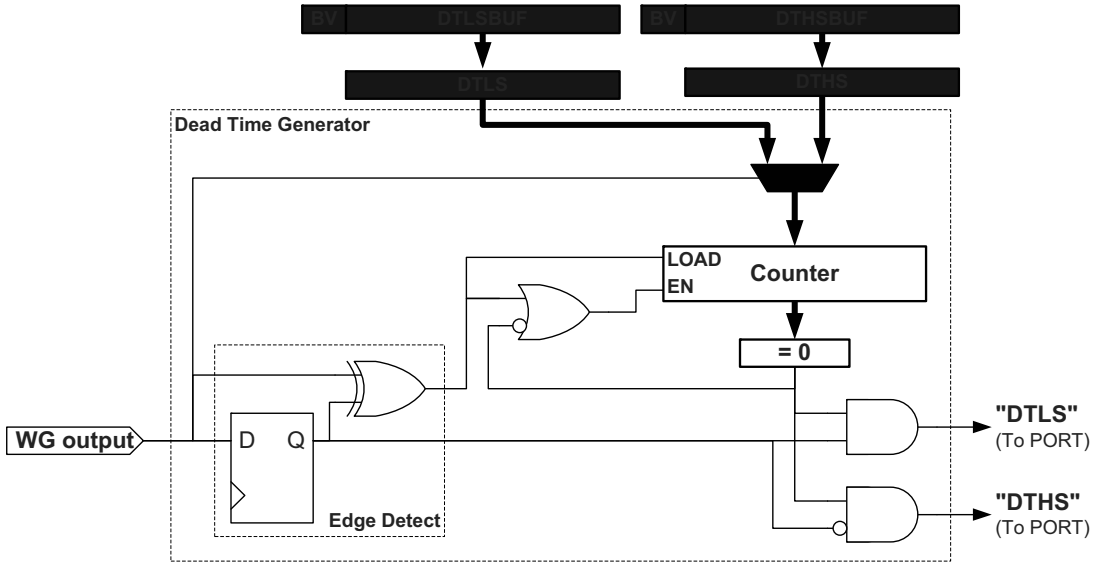
This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

15.4 Dead-time Insertion

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

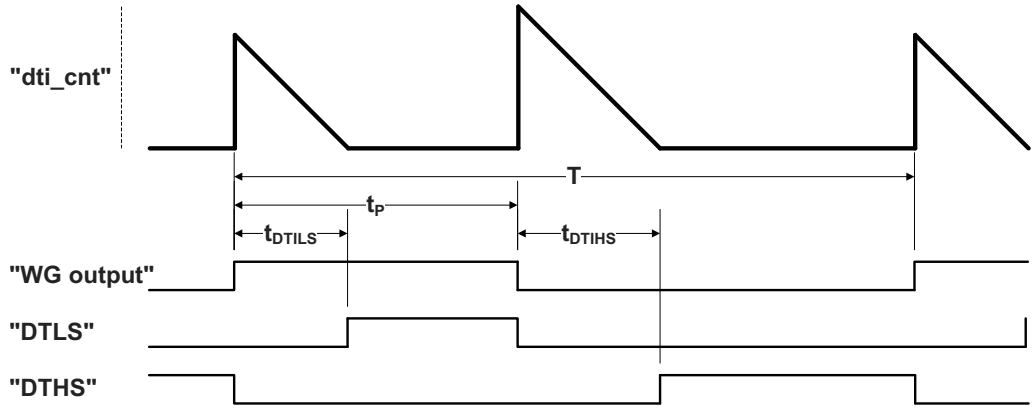
The DTI unit consists of four equal dead-time generators, one for each compare channel in timer/counter 0. Figure 15-3 shows the block diagram of one DTI generator. The four channels have a common register that controls the dead time. The high side and low side have independent dead-time setting, and the dead-time registers are double buffered.

Figure 15-3. Dead-time Generator Block Diagram



As shown in Figure 15-4, the 8-bit dead-time counter is decremented by one for each peripheral clock cycle, until it reaches zero. A nonzero counter value will force both the low side and high side outputs into their OFF state. When a change is detected on the WG output, the dead-time counter is reloaded according to the edge of the input. A positive edge initiates a counter reload of the DTLS register, and a negative edge a reload of DTHS register.

Figure 15-4. Dead-time Generator Timing Diagram



15.5 Pattern Generation

The pattern generator unit reuses the DTI registers to produce a synchronized bit pattern across the port it is connected to. In addition, the waveform generator output from compare channel A (CCA) can be distributed to and override all the port pins. These features are primarily intended for handling the commutation sequence in brushless DC motor (BLDC)

15.7.5 DTBOTH – Dead-time Concurrent Write to Both Sides

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

- **Bit 7:0 – DTBOTH: Dead-time Both Sides**

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

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

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

- **Bit 7:0 – DTBOTHBUF: Dead-time Both Sides Buffer**

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

15.7.7 DTLS – Dead-time Low Side Register

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

- **Bit 7:0 – DTLS: Dead-time Low Side**

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

15.7.8 DTHS – Dead-time High Side Register

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

- **Bit 7:0 – DTHS: Dead-time High Side**

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

15.7.9 DTLSBUF – Dead-time Low Side Buffer Register

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

- **Bit 7:0 – DTLSBUF: Dead-time Low Side Buffer**

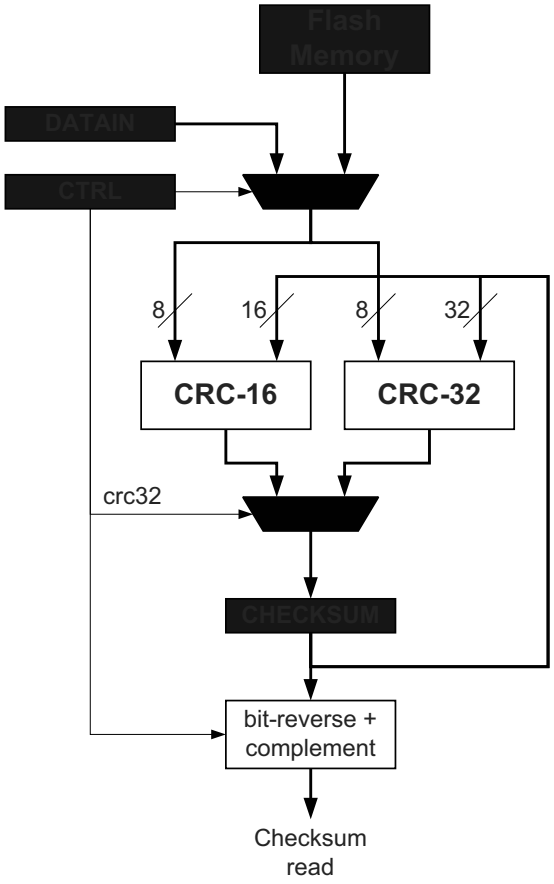
This register is the buffer for the DTLS register. If double buffering is used, valid content in this register is copied to the DTLS register on an UPDATE condition.

21.3 Operation

The data source for the CRC module must be selected in software as either flash memory or the I/O interface. The CRC module then takes data input from the selected source and generates a checksum based on these data. The checksum is available in the CHECKSUM registers in the CRC module. When CRC-32 polynomial is used, the final checksum read is bit reversed and complemented (see Figure 21-1).

For the I/O interface, which CRC polynomial is used is software selectable, but the default setting is CRC-16. CRC-32 is automatically used if Flash Memory is selected as the source. The CRC module operates on bytes only.

Figure 21-1. CRC Generator Block Diagram



21.4 CRC on Flash Memory

A CRC-32 calculation can be performed on the entire flash memory, on only the application section, on only the boot section, or on a software selectable range of the flash memory. Other than selecting the flash as the source, all further control and setup are done from the NVM controller. This means that the NVM controller configures the memory range to perform the CRC on, and the CRC is started using NVM commands. Once completed, the result is available in the checksum registers in the CRC module. For further details on setting up and performing CRC on flash memory, refer to “Memory Programming” on page 274.

21.5 CRC using the I/O Interface

CRC can be performed on any data by loading them into the CRC module using the CPU and writing the data to the DATAIN register. Using this method, an arbitrary number of bytes can be written to the register by the CPU, and CRC is done continuously for each byte. New data can be written for each cycle. The CRC complete is signaled by writing the BUSY bit in the STATUS register.

22.14.10.1 12-bit Mode, Left Adjusted

- **Bit 7:0 – CHRES[11:4]: Channel Result High byte**

These are the eight msbs of the 12-bit ADC result.

22.14.10.2 12-bit Mode, Right Adjusted

- **Bit 7:4 – Reserved**

These bits will in practice be the extension of the sign bit, CHRES11, when the ADC works in differential mode, and set to zero when the ADC works in signed mode.

- **Bit 3:0 – CHRES[11:8]: Channel Result High byte**

These are the four msbs of the 12-bit ADC result.

22.14.10.3 8-bit Mode

- **Bit 7:0 – Reserved**

These bits will in practice be the extension of the sign bit, CHRES7, when the ADC works in signed mode, and set to zero when the ADC works in single-ended mode.

22.14.11 CH0RESL – Channel 0 Result Register Low

	Bit	7	6	5	4	3	2	1	0
12-/8-bit, right		CHRES[7:0]							
12-bit, left		CHRES[3:0]				–	–	–	–
	Read/Write	R	R	R	R	R	R	R	R
	Initial Value	0	0	0	0	0	0	0	0

22.14.11.1 12-/8-bit Mode

- **Bit 7:0 – CHRES[7:0]: Channel Result Low byte**

These are the eight lsbs of the ADC result.

22.14.11.2 12-bit Mode, Left Adjusted

- **Bit 7:4 – CHRES[3:0]: Channel Result Low byte**

These are the four lsbs of the 12-bit ADC result.

- **Bit 3:0 – Reserved**

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

22.14.12 CMPH – Compare Register High

The CMPH and CMPL register pair represents the 16-bit value, CMP. For details on reading and writing 16-bit registers, refer to “Accessing 16-bit Registers” on page 11.

Bit	7	6	5	4	3	2	1	0
+0x19	CMP[15:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CMP[15:0]: Compare Value High byte**

These are the eight msbs of the 16-bit ADC compare value. In signed mode, the number representation is 2's complement, and the msb is the sign bit.

25.11.5.6 Erase EEPROM

The erase EEPROM command is used to erase all locations in all EEPROM pages that are loaded and tagged in the EEPROM page buffer.

1. Set up the NVM CMD register to the erase EPPROM command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

25.11.5.7 Read EEPROM

The read EEPROM command is used to read one byte from the EEPROM.

1. Load the NVM CMD register with the read EEPROM command.
2. Load the NVM ADDR register with the address to read.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The data byte read will be available in the NVM DATA0 register.

25.12 External Programming

External programming is the method for programming code and nonvolatile data into the device from an external programmer or debugger. This can be done by both in-system or in mass production programming.

For external programming, the device is accessed through the PDI and PDI controller, and using either the JTAG or PDI physical connection. For details on PDI and JTAG and how to enable and use the physical interface, refer to “Program and Debug Interface” on page 263. The remainder of this section assumes that the correct physical connection to the PDI is enabled. Doing this all data and program memory spaces are mapped into the linear PDI memory space. Figure 25-3 on page 288 shows the PDI memory space and the base address for each memory space in the device.

25.12.3.9 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.

25.12.3.10 Flash CRC

The flash CRC command can be used to verify the content of the flash program memory after programming. The command can be executed independently of the lock bit state.

1. Load the NVM CMD register with flash CRC command.
2. Set the CMDEX bit in the NVM CTRLA register.

Once this operation starts, the PDI bus between the PDI controller and the NVM is disabled, and the NVMEN bit in the PDI STATUS register is cleared until the operation is finished. Poll the NVMEN bit until this is set again, indicting the PDI bus is enabled.

The BUSY flag in the NVM STATUS register will be set until the operation is finished. The CRC checksum will be available in the NVM DATA register.

25.12.3.11 Write Fuse/ Lock Bit

The write fuse and write lock bit commands are used to write the fuses and the lock bits to a more secure setting.

1. Load the NVM CMD register with the write fuse/ lock bit command.
2. Write the selected fuse or lock bits by doing a PDI write operation.

The BUSY flag in the NVM STATUS register will be set until the command is finished.

For lock bit write, the lock bit write command can also be used.

25.13 Register Description

Refer to “Register Description – NVM Controller” on page 24 for a complete register description of the NVM controller.

Refer to “Register Description – PDI Control and Status Registers” on page 272 for a complete register description of the PDI.

25.14 Register Summary

Refer to “Register Description – NVM Controller” on page 24 for a complete register summary of the NVM controller.

Refer to “Register Summary” on page 273 for a complete register summary of the PDI.

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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.