

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

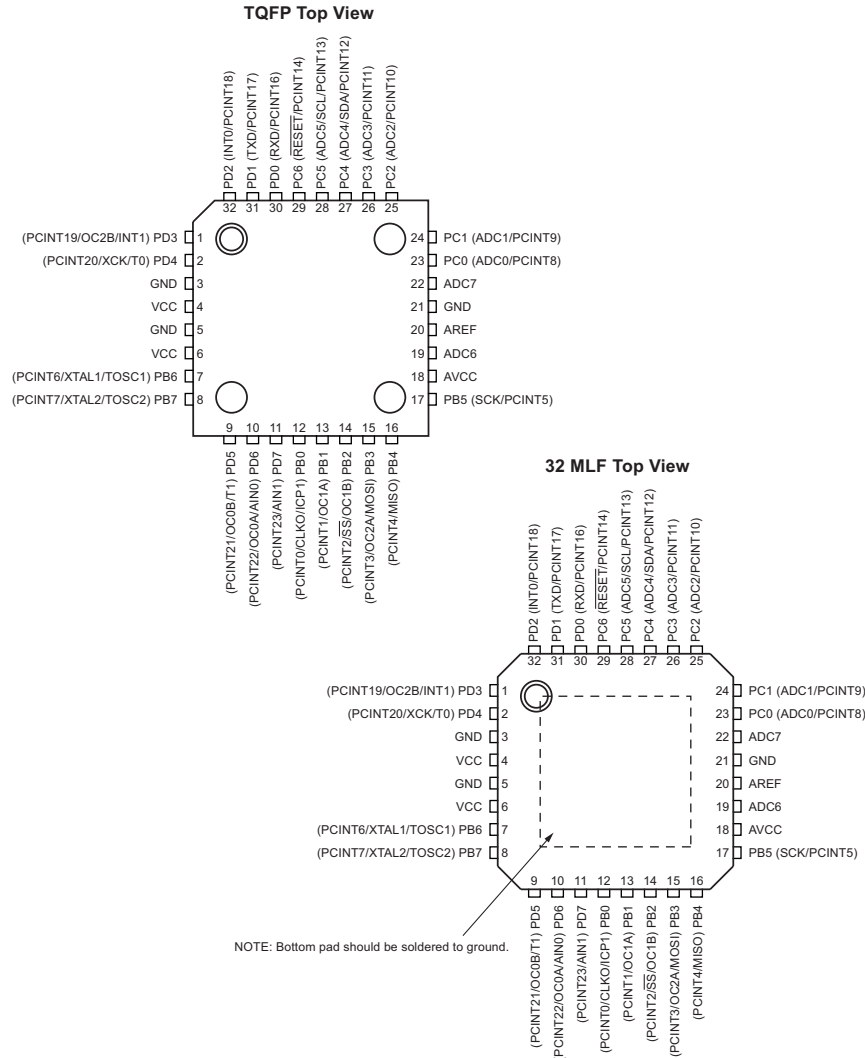
Features

- High performance, low power AVR[®] 8-bit microcontroller
- Advanced RISC architecture
 - 131 powerful instructions – most single clock cycle execution
 - 32 × 8 general purpose working registers
 - Fully static operation
 - Up to 16MIPS throughput at 16MHz
 - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
 - 32K bytes of in-system self-programmable flash program memory
 - 1Kbytes EEPROM
 - 2Kbytes internal SRAM
 - Write/erase cycles: 10,000 flash/100,000 EEPROM
 - Optional boot code section with independent lock bits
 - In-system programming by on-chip boot program
 - True read-while-write operation
 - Programming lock for software security
- Peripheral features
 - Two 8-bit Timer/Counters with separate prescaler and compare mode
 - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
 - Real time counter with separate oscillator
 - Six PWM channels
 - 8-channel 10-bit ADC in TQFP and QFN/MLF package
 - Temperature measurement
 - Programmable serial USART
 - Master/slave SPI serial interface
 - Byte-oriented 2-wire serial interface (Phillips I²C compatible)
 - Programmable watchdog timer with separate on-chip oscillator
 - On-chip analog comparator
 - Interrupt and wake-up on pin change
- Special microcontroller features
 - Power-on reset and programmable brown-out detection
 - Internal calibrated oscillator
 - External and internal interrupt sources
 - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

- I/O and packages
 - 23 programmable I/O lines
 - 32-lead TQFP, and 32-pad QFN/MLF
- Operating voltage:
 - 2.7V to 5.5V for ATmega328P
- Temperature range:
 - Automotive temperature range: –40°C to +125°C
- Speed grade:
 - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
 - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
 - Active mode: 1.5mA at 3V - 4MHz
 - Power-down mode: 1µA at 3V

1. Pin Configurations

Figure 1-1. Pinout



1.1 Pin Descriptions

1.1.1 VCC

Digital supply voltage.

1.1.2 GND

Ground.

1.1.3 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting oscillator amplifier.

If the internal calibrated RC oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of port B are elaborated in [Section 13.3.1 "Alternate Functions of Port B" on page 65](#) and [Section 8. "System Clock and Clock Options" on page 24](#).

1.1.4 Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, port C pins that are externally pulled low will source current if the pull-up resistors are activated. The port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

1.1.5 PC6/RESET

If the RSTDISBL fuse is programmed, PC6 is used as an input pin. If the RSTDISBL fuse is unprogrammed, PC6 is used as a reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in [Table 28-4 on page 261](#). Shorter pulses are not guaranteed to generate a reset.

The various special features of port C are elaborated in [Section 13.3.2 "Alternate Functions of Port C" on page 68](#).

1.1.6 Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, port D pins that are externally pulled low will source current if the pull-up resistors are activated. The port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

The various special features of port D are elaborated in [Section 13.3.3 "Alternate Functions of Port D" on page 70](#).

1.1.7 AVCC

AVCC is the supply voltage pin for the A/D converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

1.1.8 AREF

AREF is the analog reference pin for the A/D converter.

1.1.9 ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

1.2 Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of actual ATmega328P AVR® microcontrollers manufactured on the typical process technology. automotive min and max values are based on characterization of actual ATmega328P AVR microcontrollers manufactured on the whole process excursion (corner run).

1.3 Automotive Quality Grade

The ATmega328P have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the ATmega328P have been verified during regular product qualification as per AEC-Q100 grade 1. As indicated in the ordering information paragraph, the products are available in only one temperature.

Table 1-1. Temperature Grade Identification for Automotive Products

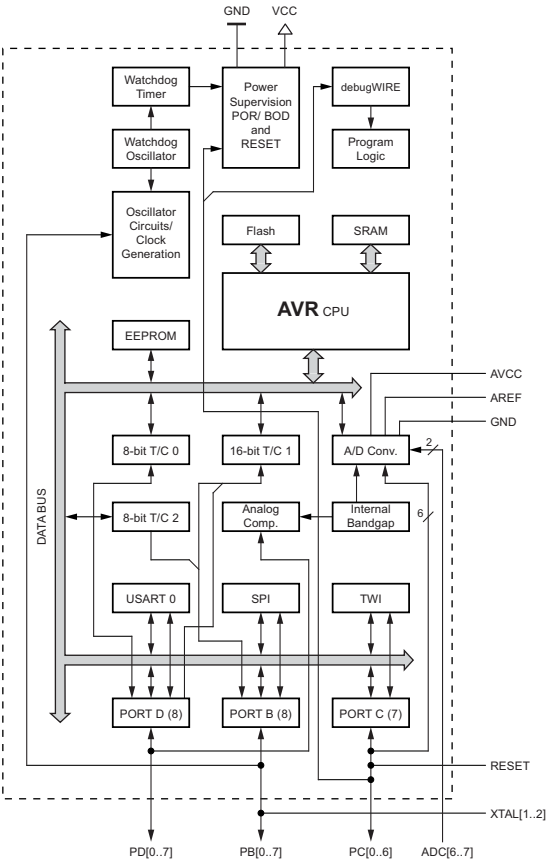
Temperature	Temperature Identifier	Comments
–40°C; +125°C	Z	Full automotive temperature range

2. Overview

The Atmel® ATmega328P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328P achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



11. Interrupts

This section describes the specifics of the interrupt handling as performed in Atmel® ATmega328P. For a general explanation of the AVR® interrupt handling, refer to [Section 6.7 "Reset and Interrupt Handling" on page 15](#).

- Each interrupt vector occupies two instruction words in Atmel ATmega328P.
- In Atmel ATmega328P, the reset vector is affected by the BOOTRST fuse, and the interrupt vector start address is affected by the IVSEL bit in MCUCR.

11.1 Interrupt Vectors in ATmega328P

Table 11-1. Reset and Interrupt Vectors in ATmega328P

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x0002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready

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

Table 11-2. Reset and Interrupt Vectors Placement in ATmega328P⁽¹⁾

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x0000	0x0002
1	1	0x0000	Boot reset address + 0x0002
0	0	Boot reset address	0x0002
0	1	Boot reset address	Boot reset address + 0x0002

Note: 1. For the BOOTRST fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATmega328P is:

```
Address      Labels Code      Comments
0x0000      jmp      RESET      ; Reset Handler
0x0002      jmp      EXT_INT0    ; IRQ0 Handler
0x0004      jmp      EXT_INT1    ; IRQ1 Handler
0x0006      jmp      PCINT0      ; PCINT0 Handler
0x0008      jmp      PCINT1      ; PCINT1 Handler
0x000A      jmp      PCINT2      ; PCINT2 Handler
0x000C      jmp      WDT         ; Watchdog Timer Handler
0x000E      jmp      TIM2_COMPA   ; Timer2 Compare A Handler
0x0010      jmp      TIM2_COMPB   ; Timer2 Compare B Handler
0x0012      jmp      TIM2_OVF     ; Timer2 Overflow Handler
0x0014      jmp      TIM1_CAPT    ; Timer1 Capture Handler
0x0016      jmp      TIM1_COMPA   ; Timer1 Compare A Handler
0x0018      jmp      TIM1_COMPB   ; Timer1 Compare B Handler
0x001A      jmp      TIM1_OVF     ; Timer1 Overflow Handler
0x001C      jmp      TIM0_COMPA   ; Timer0 Compare A Handler
0x001E      jmp      TIM0_COMPB   ; Timer0 Compare B Handler
0x0020      jmp      TIM0_OVF     ; Timer0 Overflow Handler
0x0022      jmp      SPI_STC      ; SPI Transfer Complete Handler
0x0024      jmp      USART_RXC    ; USART, RX Complete Handler
0x0026      jmp      USART_UDRE   ; USART, UDR Empty Handler
0x0028      jmp      USART_TXC    ; USART, TX Complete Handler
0x002A      jmp      ADC         ; ADC Conversion Complete Handler
0x002C      jmp      EE_RDY      ; EEPROM Ready Handler
0x002E      jmp      ANA_COMP     ; Analog Comparator Handler
0x0030      jmp      TWI         ; 2-wire Serial Interface Handler
0x0032      jmp      SPM_RDY     ; Store Program Memory Ready Handler
;
0x0033      RESET: ldi      r16, high(RAMEND); Main program start
0x0034      out      SPH,r16      ; Set Stack Pointer to top of RAM
0x0035      ldi      r16, low(RAMEND)
0x0036      out      SPL,r16
0x0037      sei                      ; Enable interrupts
0x0038      <instr> xxx
...      ...      ...      ...
```

When the BOOTRST fuse is unprogrammed, the boot section size set to 2Kbytes and the IVSEL bit in the MCUCR register is set before any interrupts are enabled, the most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATmega328P is:

```

Address      Labels Code      Comments
0x0000      RESET: ldi    r16,high(RAMEND); Main program start
0x0001              out    SPH,r16      ; Set Stack Pointer to top of RAM
0x0002              ldi    r16,low(RAMEND)
0x0003              out    SPL,r16
0x0004              sei                      ; Enable interrupts
0x0005              <instr> xxx
;
.org 0x3C02
0x3C02              jmp    EXT_INT0      ; IRQ0 Handler
0x3C04              jmp    EXT_INT1      ; IRQ1 Handler
...
0x3C32              jmp    SPM_RDY       ; Store Program Memory Ready Handler

```

When the BOOTRST fuse is programmed and the boot section size set to 2Kbytes, the most typical and general program setup for the reset and interrupt vector addresses in Atmel ATmega328P is:

```

Address      Labels Code      Comments
.org 0x0002
0x0002              jmp    EXT_INT0      ; IRQ0 Handler
0x0004              jmp    EXT_INT1      ; IRQ1 Handler
...
0x0032              jmp    SPM_RDY       ; Store Program Memory Ready Handler
;
.org 0x3C00
0x3C00      RESET: ldi    r16,high(RAMEND); Main program start
0x3C01              out    SPH,r16      ; Set Stack Pointer to top of RAM
0x3C02              ldi    r16,low(RAMEND)
0x3C03              out    SPL,r16
0x3C04              sei                      ; Enable interrupts
0x3C05              <instr> xxx

```

When the BOOTRST fuse is programmed, the boot section size set to 2Kbytes and the IVSEL bit in the MCUCR register is set before any interrupts are enabled, the most typical and general program setup for the reset and interrupt vector addresses in Atmel ATmega328P is:

```

Address      Labels Code      Comments
;
.org 0x3C00
0x3C00              jmp    RESET         ; Reset handler
0x3C02              jmp    EXT_INT0      ; IRQ0 Handler
0x3C04              jmp    EXT_INT1      ; IRQ1 Handler
...
0x3C32              jmp    SPM_RDY       ; Store Program Memory Ready Handler
;
0x3C33      RESET: ldi    r16,high(RAMEND); Main program start
0x3C34              out    SPH,r16      ; Set Stack Pointer to top of RAM
0x3C35              ldi    r16,low(RAMEND)
0x3C36              out    SPL,r16
0x3C37              sei                      ; Enable interrupts
0x3C38              <instr> xxx

```

11.2 Register Description

11.2.1 Moving Interrupts Between Application and Boot Space

The MCU control register controls the placement of the interrupt vector table.

11.2.2 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 1 – IVSEL: Interrupt Vector Select

When the IVSEL bit is cleared (zero), the interrupt vectors are placed at the start of the flash memory. When this bit is set (one), the interrupt vectors are moved to the beginning of the boot loader section of the flash. The actual address of the start of the boot flash section is determined by the BOOTSZ fuses. Refer to the [Section 26. “Boot Loader Support – Read-While-Write Self-Programming” on page 229](#) for details. To avoid unintentional changes of interrupt vector tables, a special write procedure must be followed to change the IVSEL bit:

- Write the interrupt vector change enable (IVCE) bit to one.
- Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the status register is unaffected by the automatic disabling.

Note: If interrupt vectors are placed in the boot loader section and boot lock bit BLB02 is programmed, interrupts are disabled while executing from the application section. If interrupt vectors are placed in the application section and boot lock bit BLB12 is programmed, interrupts are disabled while executing from the boot loader section. Refer to the [Section 26. “Boot Loader Support – Read-While-Write Self-Programming” on page 229](#) for details on boot lock bits.

This bit is not available in Atmel® ATmega328P.

• Bit 0 – IVCE: Interrupt Vector Change Enable

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See code example below.

Assembly Code Example
<pre> Move_interrupts: ; Enable change of Interrupt Vectors ldi r16, (1<<IVCE) out MCUCR, r16 ; Move interrupts to Boot Flash section ldi r16, (1<<IVSEL) out MCUCR, r16 ret </pre>
C Code Example
<pre> void Move_interrupts(void) { /* Enable change of Interrupt Vectors */ MCUCR = (1<<IVCE); /* Move interrupts to Boot Flash section */ MCUCR = (1<<IVSEL); } </pre>

This bit is not available in Atmel ATmega328P.

12. External Interrupts

The external interrupts are triggered by the INT0 and INT1 pins or any of the PCINT23..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 and INT1 or PCINT23..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. The pin change interrupt PC12 will trigger if any enabled PCINT23..16 pin toggles. The pin change interrupt PC11 will trigger if any enabled PCINT14..8 pin toggles. The pin change interrupt PC10 will trigger if any enabled PCINT7..0 pin toggles. The PCMSK2, PCMSK1 and PCMSK0 registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT23..0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

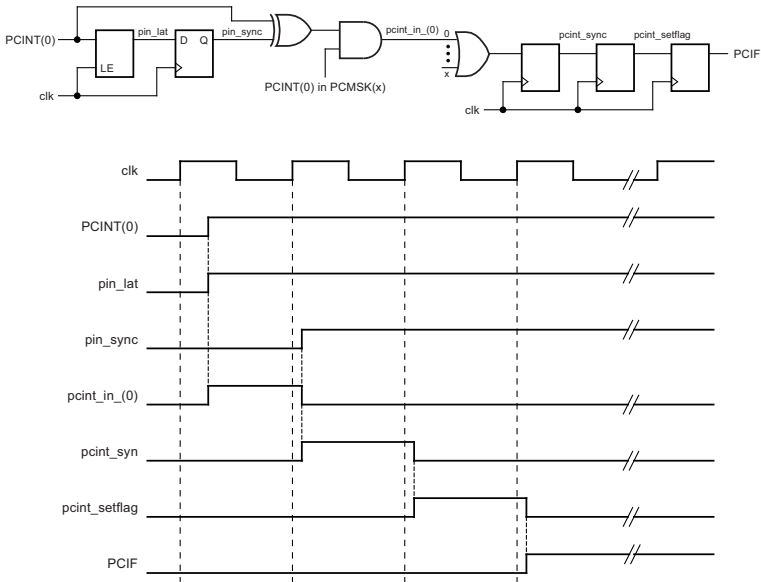
The INT0 and INT1 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the external interrupt control register A – EICRA. When the INT0 or INT1 interrupts are enabled and are configured as level triggered, the interrupts will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 or INT1 requires the presence of an I/O clock, described in [Section 8.1 “Clock Systems and their Distribution” on page 24](#). Low level interrupt on INT0 and INT1 is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL fuses as described in [Section 8. “System Clock and Clock Options” on page 24](#).

12.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in [Figure 12-1](#).

Figure 12-1. Timing of Pin Change Interrupts



12.2 Register Description

12.2.1 EICRA – External Interrupt Control Register A

The external interrupt control register A contains control bits for interrupt sense control.

Bit (0x69)	7	6	5	4	3	2	1	0	
	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
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 – Res: Reserved Bits**
These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- **Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**
The external interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in [Table 12-1](#). The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 12-1. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**
The external interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in [Table 12-2](#). The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 12-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

12.2.2 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
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 – Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- Bit 1 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the status register (SREG) is set (one), the external pin interrupt is enabled. The interrupt sense control1 bits 1/0 (ISC11 and ISC10) in the external interrupt control register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of external interrupt request 1 is executed from the INT1 interrupt vector.

- Bit 0 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the status register (SREG) is set (one), the external pin interrupt is enabled. The interrupt sense control0 bits 1/0 (ISC01 and ISC00) in the external interrupt control register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of external interrupt request 0 is executed from the INT0 interrupt vector.

12.2.3 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
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 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega328P, and will always read as zero.

- Bit 1 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in EIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

- Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

12.2.4 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
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 - Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- Bit 2 - PCIE2: Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCIE2 interrupt vector. PCINT23..16 pins are enabled individually by the PCMSK2 register.

- Bit 1 - PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCIE1 interrupt vector. PCINT14..8 pins are enabled individually by the PCMSK1 register.

- Bit 0 - PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCIE0 interrupt vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

12.2.5 PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
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 - Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega328P, and will always read as zero.

- Bit 2 - PCIF2: Pin Change Interrupt Flag 2**

When a logic change on any PCINT23..16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- Bit 1 - PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT14..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- Bit 0 - PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

12.2.6 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	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 – PCINT23..16: Pin Change Enable Mask 23..16**

Each PCINT23..16-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

12.2.7 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
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 – Res: Reserved Bit**

This bit is an unused bit in the Atmel® ATmega328P, and will always read as zero.

- Bit 6..0 – PCINT14..8: Pin Change Enable Mask 14..8**

Each PCINT14..8-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

12.2.8 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	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 – PCINT7..0: Pin Change Enable Mask 7..0**

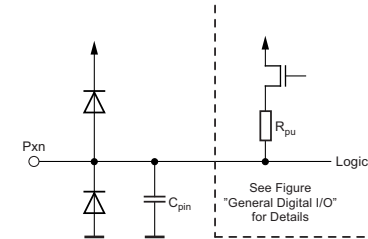
Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

13. I/O-Ports

13.1 Overview

All AVR® ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 13-1. Refer to Section 28. “Electrical Characteristics” on page 258 for a complete list of parameters.

Figure 13-1. I/O Pin Equivalent Schematic

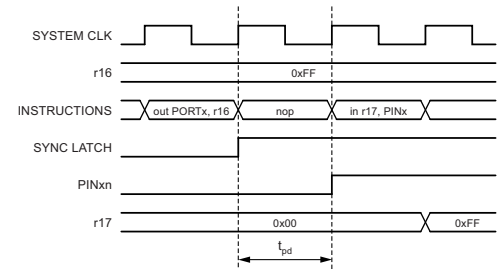


All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in port B, here documented generally as PORTxn. The physical I/O registers and bit locations are listed in Section 13.4 “Register Description” on page 72.

Three I/O memory address locations are allocated for each port, one each for the data register – PORTx, data direction register – DDRx, and the port input pins – PINx. The port input pins I/O location is read only, while the data register and the data direction register are read/write. However, writing a logic one to a bit in the PINx register, will result in a toggle in the corresponding bit in the data register. In addition, the pull-up disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as general digital I/O is described in Section 13.2 “Ports as General Digital I/O” on page 59. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 13.3 “Alternate Port Functions” on page 63. Refer to the individual module sections for a full description of the alternate functions. Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

Figure 13-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example ⁽¹⁾
<pre>... ; Define pull-ups and set outputs high ; Define directions for port pins ldi r16, (1<<PB7) (1<<PB6) (1<<PB1) (1<<PB0) ldi r17, (1<<DDB3) (1<<DDB2) (1<<DDB1) (1<<DDB0) out PORTB, r16 out DDRB, r17 ; Insert nop for synchronization nop ; Read port pins in r16, PINB ...</pre>
C Code Example
<pre>unsigned char i; ... /* Define pull-ups and set outputs high */ /* Define directions for port pins */ PORTB = (1<<PB7) (1<<PB6) (1<<PB1) (1<<PB0); DDRB = (1<<DDB3) (1<<DDB2) (1<<DDB1) (1<<DDB0); /* Insert nop for synchronization*/ __no_operation(); /* Read port pins */ i = PINB; ...</pre>

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

13.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 13-2, the digital input signal can be clamped to ground at the input of the Schmitt trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$. SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 13.3 “Alternate Port Functions” on page 63. If a logic high level (“one”) is present on an asynchronous external interrupt pin configured as “interrupt on rising edge, falling edge, or any logic change on pin” while the external interrupt is not enabled, the corresponding external interrupt flag will be set when resuming from the above mentioned sleep mode, as the clamping in these sleep mode produces the requested logic change.

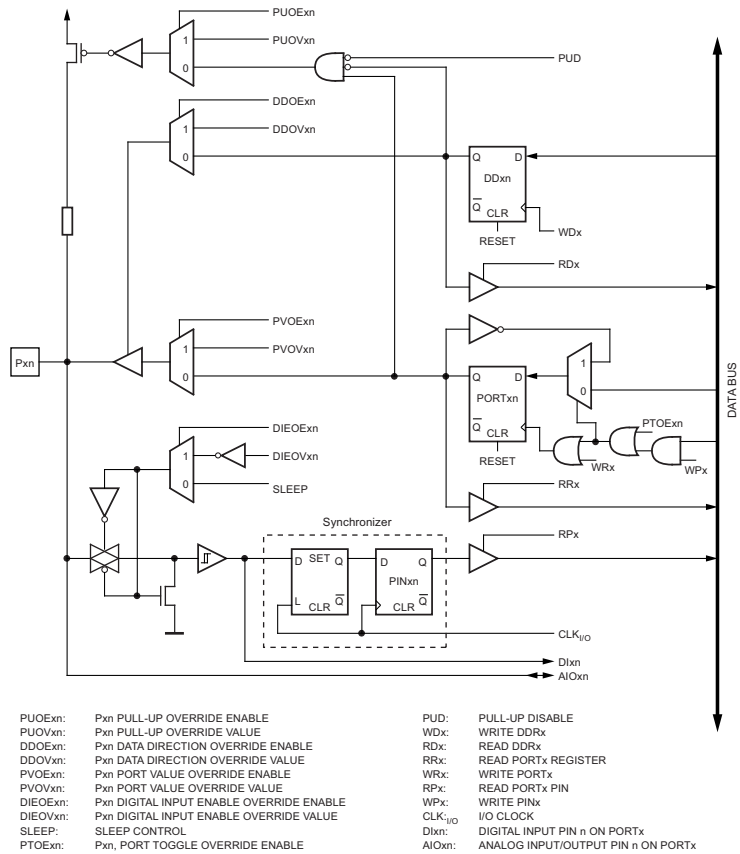
13.2.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (reset, active mode and idle mode). The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

13.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 13-5 shows how the port pin control signals from the simplified Figure 13-2 on page 59 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR® microcontroller family.

Figure 13-5. Alternate Port Functions⁽¹⁾



Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{IO}, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

Table 13-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 13-5 on page 63 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 13-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUEOE	Pull-up override enable	If this signal is set, the pull-up enable is controlled by the PUOE signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up override value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD register bits.
DDOE	Data direction override enable	If this signal is set, the output driver enable is controlled by the DDOV signal. If this signal is cleared, the output driver is enabled by the DDxn register bit.
DDOV	Data direction override value	If DDOE is set, the output driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn register bit.
PVOE	Port value override enable	If this signal is set and the output driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the output driver is enabled, the port value is controlled by the PORTxn register bit.
PVOV	Port value override value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn register bit.
PTOE	Port toggle override enable	If PTOE is set, the PORTxn register bit is inverted.
DIEOE	Digital input enable override enable	If this bit is set, the digital input enable is controlled by the DIEOV signal. If this signal is cleared, the digital input enable is determined by MCU state (normal mode, sleep mode).
DIEOV	Digital input enable override value	If DIEOE is set, the digital input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (normal mode, sleep mode).
DI	Digital input	This is the digital input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the digital input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog input/output	This is the analog input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

13.3.1 Alternate Functions of Port B

The port B pins with alternate functions are shown in [Table 13-3](#).

Table 13-3. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	XTAL2 (chip clock oscillator pin 2) TOSC2 (timer oscillator pin 2) PCINT7 (pin change interrupt 7)
PB6	XTAL1 (chip clock oscillator pin 1 or external clock input) TOSC1 (timer oscillator pin 1) PCINT6 (pin change interrupt 6)
PB5	SCK (SPI bus master clock input) PCINT5 (pin change interrupt 5)
PB4	MISO (SPI bus master input/slave output) PCINT4 (pin change interrupt 4)
PB3	MOSI (SPI bus master output/slave input) OC2A (Timer/Counter2 output compare match A output) PCINT3 (pin change interrupt 3)
PB2	\overline{SS} (SPI bus master slave select) OC1B (Timer/Counter1 output compare match B output) PCINT2 (pin change interrupt 2)
PB1	OC1A (Timer/Counter1 output compare match A output) PCINT1 (pin change interrupt 1)
PB0	ICP1 (Timer/Counter1 input capture input) CLKO (divided system clock output) PCINT0 (pin change interrupt 0)

The alternate pin configuration is as follows:

• XTAL2/TOSC2/PCINT7 – Port B, Bit 7

XTAL2: Chip clock oscillator pin 2. Used as clock pin for crystal oscillator or low-frequency crystal oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

TOSC2: Timer oscillator pin 2. Used only if internal calibrated RC oscillator is selected as chip clock source, and the asynchronous timer is enabled by the correct setting in ASSR. When the AS2 bit in ASSR is set (one) and the EXCLK bit is cleared (zero) to enable asynchronous clocking of Timer/Counter2 using the crystal oscillator, pin PB7 is disconnected from the port, and becomes the inverting output of the oscillator amplifier. In this mode, a crystal oscillator is connected to this pin, and the pin cannot be used as an I/O pin.

PCINT7: Pin change interrupt source 7. The PB7 pin can serve as an external interrupt source.

If PB7 is used as a clock pin, DDB7, PORTB7 and PINB7 will all read 0.

• XTAL1/TOSC1/PCINT6 – Port B, Bit 6

XTAL1: Chip clock oscillator pin 1. Used for all chip clock sources except internal calibrated RC oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

TOSC1: Timer oscillator pin 1. Used only if internal calibrated RC oscillator is selected as chip clock source, and the asynchronous timer is enabled by the correct setting in ASSR. When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PB6 is disconnected from the port, and becomes the input of the inverting oscillator amplifier. In this mode, a crystal oscillator is connected to this pin, and the pin can not be used as an I/O pin.

PCINT6: Pin change interrupt source 6. The PB6 pin can serve as an external interrupt source.

If PB6 is used as a clock pin, DDB6, PORTB6 and PINB6 will all read 0.

• SCK/PCINT5 – Port B, Bit 5

SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

PCINT5: Pin change interrupt source 5. The PB5 pin can serve as an external interrupt source.

• MISO/PCINT4 – Port B, Bit 4

MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB4. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit.

PCINT4: Pin change interrupt source 4. The PB4 pin can serve as an external interrupt source.

• MOSI/OC2/PCINT3 – Port B, Bit 3

MOSI: SPI master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB3. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB3. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB3 bit.

OC2, output compare match output: The PB3 pin can serve as an external output for the Timer/Counter2 compare match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC2 pin is also the output pin for the PWM mode timer function.

PCINT3: Pin change interrupt source 3. The PB3 pin can serve as an external interrupt source.

• \overline{SS} /OC1B/PCINT2 – Port B, Bit 2

\overline{SS} : Slave select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB2. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

OC1B, output compare match output: The PB2 pin can serve as an external output for the Timer/Counter1 compare match B. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT2: Pin change interrupt source 2. The PB2 pin can serve as an external interrupt source.

• OC1A/PCINT1 – Port B, Bit 1

OC1A, output compare match output: The PB1 pin can serve as an external output for the Timer/Counter1 compare match A. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

PCINT1: Pin change interrupt source 1. The PB1 pin can serve as an external interrupt source.

• ICP1/CLKO/PCINT0 – Port B, Bit 0

ICP1, input capture pin: The PB0 pin can act as an input capture pin for Timer/Counter1.

CLKO, divided system clock: The divided system clock can be output on the PB0 pin. The divided system clock will be output if the CKOUT fuse is programmed, regardless of the PORTB0 and DDB0 settings. It will also be output during reset.

PCINT0: Pin change interrupt source 0. The PB0 pin can serve as an external interrupt source.

[Table 13-4 on page 67](#) and [Table 13-5 on page 67](#) relate the alternate functions of Port B to the overriding signals shown in [Figure 13-5 on page 63](#). SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 13-4. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/XTAL2/ TOSC2/PCINT7 ⁽¹⁾	PB6/XTAL1/ TOSC1/PCINT6 ⁽¹⁾	PB5/SCK/ PCINT5	PB4/MISO/ PCINT4
PUOE	INTRC × EXTCK + AS2	INTRC + AS2	SPE × MSTR	SPE × MSTR
PUOV	0	0	PORTB5 × PUD	PORTB4 × PUD
DDOE	INTRC × EXTCK + AS2	INTRC + AS2	SPE × MSTR	SPE × MSTR
DDOV	0	0	0	0
PVOE	0	0	SPE × MSTR	SPE × MSTR
PVOV	0	0	SCK OUTPUT	SPI SLAVE OUTPUT
DIEOE	INTRC × EXTCK + AS2 + PCINT7 × PCIE0	INTRC + AS2 + PCINT6 × PCIE0	PCINT5 × PCIE0	PCINT4 × PCIE0
DIEOV	(INTRC + EXTCK) × AS2	INTRC × AS2	1	1
DI	PCINT7 INPUT	PCINT6 INPUT	PCINT5 INPUT SCK INPUT	PCINT4 INPUT SPI MSTR INPUT
AIO	Oscillator output	Oscillator/clock input	–	–

Note: 1. INTRC means that one of the internal RC oscillators are selected (by the CKSEL fuses), EXTCK means that external clock is selected (by the CKSEL fuses)

Table 13-5. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/MOSI/ OC2/PCINT3	PB2/SS/ OC1B/PCINT2	PB1/OC1A/ PCINT1	PB0/ICP1/ PCINT0
PUOE	SPE × MSTR	SPE × MSTR	0	0
PUOV	PORTB3 × PUD	PORTB2 × PUD	0	0
DDOE	SPE × MSTR	SPE × MSTR	0	0
DDOV	0	0	0	0
PVOE	SPE × MSTR + OC2A ENABLE	OC1B ENABLE	OC1A ENABLE	0
PVOV	SPI MSTR OUTPUT + OC2A	OC1B	OC1A	0
DIEOE	PCINT3 × PCIE0	PCINT2 × PCIE0	PCINT1 × PCIE0	PCINT0 × PCIE0
DIEOV	1	1	1	1
DI	PCINT3 INPUT SPI SLAVE INPUT	PCINT2 INPUT SPI SS	PCINT1 INPUT	PCINT0 INPUT ICP1 INPUT
AIO	–	–	–	–

13.3.2 Alternate Functions of Port C

The port C pins with alternate functions are shown in Table 13-6.

Table 13-6. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC6	RESET (reset pin) PCINT14 (pin change interrupt 14)
PC5	ADC5 (ADC input channel 5) SCL (2-wire serial bus clock line) PCINT13 (pin change interrupt 13)
PC4	ADC4 (ADC input channel 4) SDA (2-wire serial bus data input/output line) PCINT12 (pin change interrupt 12)
PC3	ADC3 (ADC input channel 3) PCINT11 (pin change interrupt 11)
PC2	ADC2 (ADC input channel 2) PCINT10 (pin change interrupt 10)
PC1	ADC1 (ADC input channel 1) PCINT9 (pin change interrupt 9)
PC0	ADC0 (ADC input channel 0) PCINT8 (pin change interrupt 8)

The alternate pin configuration is as follows:

• RESET/PCINT14 – Port C, Bit 6

RESET, Reset pin: When the RSTDISBL fuse is programmed, this pin functions as an input pin, and the part will have to rely on power-on reset and brown-out reset as its reset sources. When the RSTDISBL fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an input pin.

If PC6 is used as a reset pin, DDC6, PORTC6 and PINC6 will all read 0.

PCINT14: Pin change interrupt source 14. The PC6 pin can serve as an external interrupt source.

• SCL/ADC5/PCINT13 – Port C, Bit 5

SCL, 2-wire serial interface clock: When the TWEN bit in TWCR is set (one) to enable the 2-wire serial interface, pin PC5 is disconnected from the port and becomes the serial clock I/O pin for the 2-wire serial interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.

PC5 can also be used as ADC input channel 5. Note that ADC input channel 5 uses digital power.

PCINT13: Pin change interrupt source 13. The PC5 pin can serve as an external interrupt source.

• SDA/ADC4/PCINT12 – Port C, Bit 4

SDA, 2-wire serial interface data: When the TWEN bit in TWCR is set (one) to enable the 2-wire serial interface, pin PC4 is disconnected from the port and becomes the serial data I/O pin for the 2-wire serial interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.

PC4 can also be used as ADC input Channel 4. Note that ADC input channel 4 uses digital power.

PCINT12: Pin change interrupt source 12. The PC4 pin can serve as an external interrupt source.

• ADC3/PCINT11 – Port C, Bit 3

PC3 can also be used as ADC input Channel 3. Note that ADC input channel 3 uses analog power.

PCINT11: Pin change interrupt source 11. The PC3 pin can serve as an external interrupt source.

- **ADC2/PCINT10 – Port C, Bit 2**

PC2 can also be used as ADC input channel 2. Note that ADC input channel 2 uses analog power.
PCINT10: Pin change interrupt source 10. The PC2 pin can serve as an external interrupt source.

- **ADC1/PCINT9 – Port C, Bit 1**

PC1 can also be used as ADC input channel 1. Note that ADC input channel 1 uses analog power.
PCINT9: Pin change interrupt source 9. The PC1 pin can serve as an external interrupt source.

- **ADC0/PCINT8 – Port C, Bit 0**

PC0 can also be used as ADC input channel 0. Note that ADC input channel 0 uses analog power.
PCINT8: Pin change interrupt source 8. The PC0 pin can serve as an external interrupt source.

Table 13-7 and Table 13-8 relate the alternate functions of port C to the overriding signals shown in Figure 13-5 on page 63.

Table 13-7. Overriding Signals for Alternate Functions in PC6..PC4⁽¹⁾

Signal Name	PC6/RESET/PCINT14	PC5/SCL/ADC5/PCINT13	PC4/SDA/ADC4/PCINT12
PUOE	RSTDISBL	TWEN	TWEN
PUOV	1	PORTC5 × PUD	PORTC4 × PUD
DDOE	RSTDISBL	TWEN	TWEN
DDOV	0	SCL_OUT	SDA_OUT
PVOE	0	TWEN	TWEN
PVOV	0	0	0
DIOE	RSTDISBL + PCINT14 × PCIE1	PCINT13 × PCIE1 + ADC5D	PCINT12 × PCIE1 + ADC4D
DIOV	RSTDISBL	PCINT13 × PCIE1	PCINT12 × PCIE1
DI	PCINT14 INPUT	PCINT13 INPUT	PCINT12 INPUT
AIO	RESET INPUT	ADC5 INPUT / SCL INPUT	ADC4 INPUT / SDA INPUT

Note: 1. When enabled, the 2-wire serial interface enables slew-rate controls on the output pins PC4 and PC5. This is not shown in the figure. In addition, spike filters are connected between the AIO outputs shown in the port figure and the digital logic of the TWI module.

Table 13-8. Overriding Signals for Alternate Functions in PC3..PC0

Signal Name	PC3/ADC3/PCINT11	PC2/ADC2/PCINT10	PC1/ADC1/PCINT9	PC0/ADC0/PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIOE	PCINT11 × PCIE1 + ADC3D	PCINT10 × PCIE1 + ADC2D	PCINT9 × PCIE1 + ADC1D	PCINT8 × PCIE1 + ADC0D
DIOV	PCINT11 × PCIE1	PCINT10 × PCIE1	PCINT9 × PCIE1	PCINT8 × PCIE1
DI	PCINT11 INPUT	PCINT10 INPUT	PCINT9 INPUT	PCINT8 INPUT
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

13.3.3 Alternate Functions of Port D

The port D pins with alternate functions are shown in Table 13-9.

Table 13-9. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	AIN1 (analog comparator negative input) PCINT23 (Pin Change Interrupt 23)
PD6	AIN0 (analog comparator positive input) OC0A (Timer/Counter0 output compare match A output) PCINT22 (pin change interrupt 22)
PD5	T1 (Timer/Counter 1 external counter input) OC0B (Timer/Counter0 output compare match B output) PCINT21 (pin change interrupt 21)
PD4	XCK (USART external clock input/output) T0 (Timer/Counter 0 external counter input) PCINT20 (pin change interrupt 20)
PD3	INT1 (external interrupt 1 input) OC2B (Timer/Counter2 output compare match B output) PCINT19 (pin change interrupt 19)
PD2	INT0 (external interrupt 0 input) PCINT18 (pin change interrupt 18)
PD1	TXD (USART output pin) PCINT17 (pin change interrupt 17)
PD0	RXD (USART input pin) PCINT16 (pin change interrupt 16)

The alternate pin configuration is as follows:

- **AIN1/OC2B/PCINT23 – Port D, Bit 7**

AIN1, analog comparator negative input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

PCINT23: Pin change interrupt source 23. The PD7 pin can serve as an external interrupt source.

- **AIN0/OC0A/PCINT22 – Port D, Bit 6**

AIN0, analog comparator positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

OC0A, output compare match output: The PD6 pin can serve as an external output for the Timer/Counter0 compare match A. The PD6 pin has to be configured as an output (DDD6 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.

PCINT22: Pin change interrupt source 22. The PD6 pin can serve as an external interrupt source.

- **T1/OC0B/PCINT21 – Port D, Bit 5**

T1, Timer/Counter1 counter source.

OC0B, output compare match output: The PD5 pin can serve as an external output for the Timer/Counter0 compare match B. The PD5 pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.

PCINT21: Pin change interrupt source 21. The PD5 pin can serve as an external interrupt source.

- **XCK/T0/PCINT20 – Port D, Bit 4**

XCK, USART external clock.

T0, Timer/Counter0 counter source.

PCINT20: Pin change interrupt source 20. The PD4 pin can serve as an external interrupt source.

- **INT1/OC2B/PCINT19 – Port D, Bit 3**

INT1, external interrupt source 1: The PD3 pin can serve as an external interrupt source.

OC2B, output compare match output: The PD3 pin can serve as an external output for the Timer/Counter0 compare match B. The PD3 pin has to be configured as an output (DDD3 set (one)) to serve this function. The OC2B pin is also the output pin for the PWM mode timer function.

PCINT19: Pin change interrupt source 19. The PD3 pin can serve as an external interrupt source.

- **INT0/PCINT18 – Port D, Bit 2**

INT0, external interrupt source 0: The PD2 pin can serve as an external interrupt source.

PCINT18: Pin change interrupt source 18. The PD2 pin can serve as an external interrupt source.

- **TXD/PCINT17 – Port D, Bit 1**

TXD, transmit Data (data output pin for the USART). When the USART transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

PCINT17: Pin change interrupt source 17. The PD1 pin can serve as an external interrupt source.

- **RXD/PCINT16 – Port D, Bit 0**

RXD, Receive Data (data input pin for the USART). When the USART receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

PCINT16: Pin change interrupt source 16. The PD0 pin can serve as an external interrupt source.

Table 13-10 and Table 13-11 on page 72 relate the alternate functions of port D to the overriding signals shown in Figure 13-5 on page 63.

Table 13-10. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/AIN1/PCINT23	PD6/AIN0/OC0A/PCINT22	PD5/T1/OC0B/PCINT21	PD4/XCK/T0/PCINT20
PUOE	0	0	0	0
PUO	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	OC0A ENABLE	OC0B ENABLE	UMSEL
PVOV	0	OC0A	OC0B	XCK OUTPUT
DIEOE	PCINT23 × PCIE2	PCINT22 × PCIE2	PCINT21 × PCIE2	PCINT20 × PCIE2
DIEOV	1	1	1	1
DI	PCINT23 INPUT	PCINT22 INPUT	PCINT21 INPUT T1 INPUT	PCINT20 INPUT XCK INPUT T0 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

Table 13-11. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/OC2B/INT1/ PCINT19	PD2/INT0/ PCINT18	PD1/TXD/ PCINT17	PD0/RXD/ PCINT16
PUOE	0	0	TXEN	RXEN
PUO	0	0	0	PORTD0 × PUD
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	OC2B ENABLE	0	TXEN	0
PVOV	OC2B	0	TXD	0
DIEOE	INT1 ENABLE + PCINT19 × PCIE2	INT0 ENABLE + PCINT18 × PCIE1	PCINT17 × PCIE2	PCINT16 × PCIE2
DIEOV	1	1	1	1
DI	PCINT19 INPUT INT1 INPUT	PCINT18 INPUT INT0 INPUT	PCINT17 INPUT	PCINT16 INPUT RXD
AIO	–	–	–	–

13.4 Register Description

13.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ((DDxn, PORTxn) = 0b01). See Section 13.2.1 “Configuring the Pin” on page 59 for more details about this feature.

13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.4 PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

13.4.5 PORTC – The Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
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	

13.4.6 DDRC – The Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
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	

13.4.7 PINC – The Port C Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

13.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.9 DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.10 PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14. 8-bit Timer/Counter0 with PWM

14.1 Features

- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

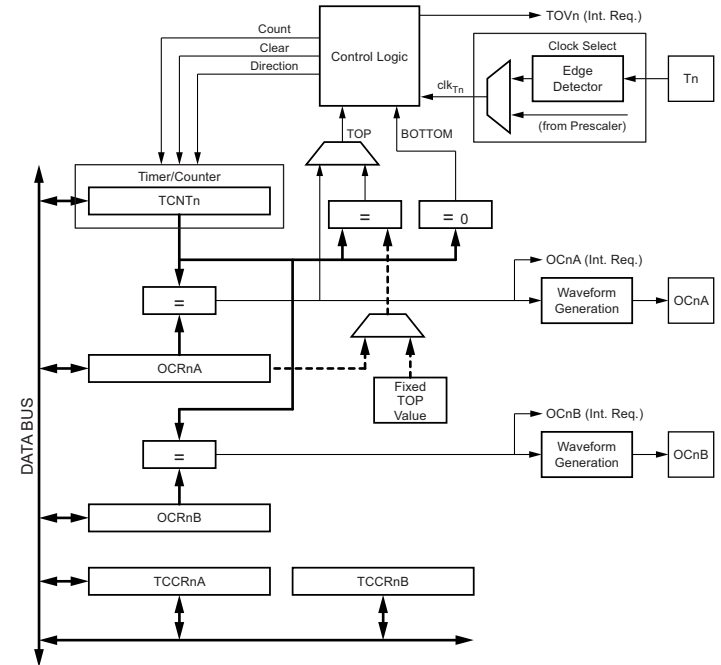
14.2 Overview

Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent output compare units, and with PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown in [Figure 14-1](#). For the actual placement of I/O pins, refer to [Section 1-1 “Pinout” on page 3](#). CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in the [Section 14.9 “Register Description” on page 84](#).

The PRTIM0 bit in [Section 9.10 “Minimizing Power Consumption” on page 36](#) must be written to zero to enable Timer/Counter0 module.

Figure 14-1. 8-bit Timer/Counter Block Diagram



14.2.1 Definitions

Many register and bit references in this section are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 0. A lower case "x" replaces the output compare unit, in this case compare unit A or compare unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 14-1 are also used extensively throughout the document.

Table 14-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A register. The assignment is dependent on the mode of operation.

14.2.2 Registers

The Timer/Counter (TCNT0) and output compare registers (OCR0A and OCR0B) are 8-bit registers. interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the timer interrupt flag register (TIFR0). All interrupts are individually masked with the timer interrupt mask register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The clock select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T0}).

The double buffered output compare registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins (OC0A and OC0B). See Section 15.7.3 "Using the Output Compare Unit" on page 99 for details. The compare match event will also set the compare flag (OCF0A or OCF0B) which can be used to generate an output compare interrupt request.

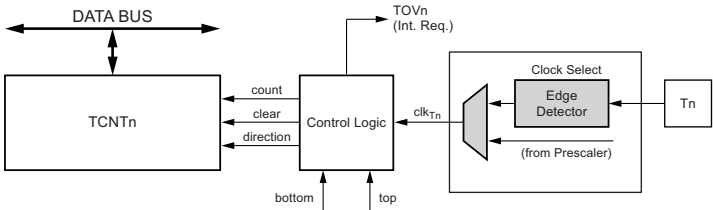
14.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter control register (TCCR0B). For details on clock sources and prescaler, see Section 16. "Timer/Counter0 and Timer/Counter1 Prescalers" on page 114.

14.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 14-2 shows a block diagram of the counter and its surroundings.

Figure 14-2. Counter Unit Block Diagram



Signal description (internal signals):

- count** Increment or decrement TCNT0 by 1.
- direction** Select between increment and decrement.
- clear** Clear TCNT0 (set all bits to zero).
- clk_{Tn}** Timer/Counter clock, referred to as clk_{T0} in the following.
- top** Signalize that TCNT0 has reached maximum value.
- bottom** Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the clock select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter control register (TCCR0A) and the WGM02 bit located in the Timer/Counter control register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see Section 14.7 "Modes of Operation" on page 78.

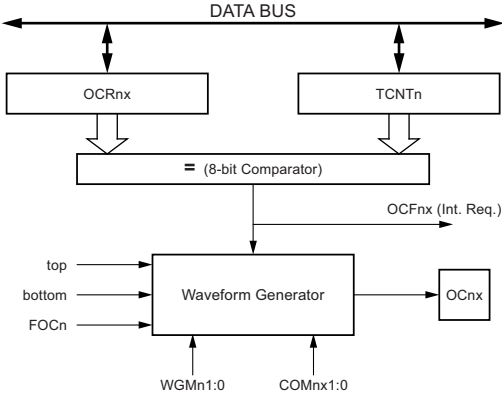
The Timer/Counter overflow flag (TOV0) is set according to the mode of operation selected by the WGM02:0 bits. TOV0 can be used for generating a CPU interrupt.

14.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the output compare flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM02:0 bits and compare output mode (COM0x1:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (Section 14.7 "Modes of Operation" on page 78).

Figure 14-3 shows a block diagram of the output compare unit.

Figure 14-3. Output Compare Unit, Block Diagram



The OCR0x registers are double buffered when using any of the pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x compare registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0x register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0x buffer register, and if double buffering is disabled the CPU will access the OCR0x directly.

14.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the force output compare (FOC0x) bit. Forcing compare match will not set the OCF0x flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the COM0x1:0 bits settings define whether the OC0x pin is set, cleared or toggled).

14.5.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

14.5.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down counting.

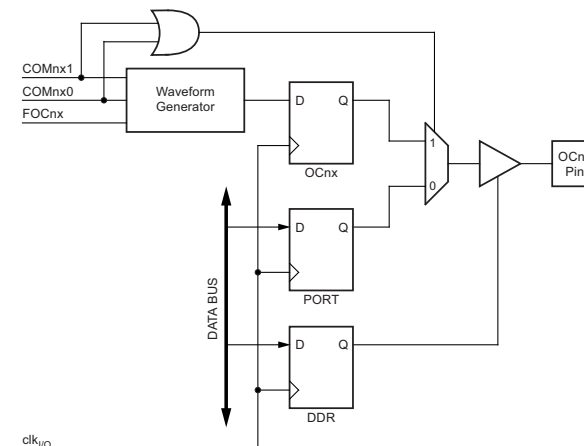
The setup of the OC0x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC0x value is to use the force output compare (FOC0x) strobe bits in normal mode. The OC0x registers keep their values even when changing between waveform generation modes.

Be aware that the COM0x1:0 bits are not double buffered together with the compare value. Changing the COM0x1:0 bits will take effect immediately.

14.6 Compare Match Output Unit

The compare output mode (COM0x1:0) bits have two functions. The waveform generator uses the COM0x1:0 bits for defining the output compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. [Figure 14-4 on page 78](#) shows a simplified schematic of the logic affected by the COM0x1:0 bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x register, not the OC0x pin. If a system reset occurs, the OC0x register is reset to "0".

Figure 14-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC0x) from the waveform generator if either of the COM0x1:0 bits are set. However, the OC0x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC0x pin (DDR_OC0x) must be set as output before the OC0x value is visible on the pin. The port override function is independent of the waveform generation mode.

The design of the output compare pin logic allows initialization of the OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation. See [Section 14.9 "Register Description" on page 84](#).

14.6.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM0x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0x1:0 = 0 tells the waveform generator that no action on the OC0x register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to [Table 14-2 on page 84](#). For fast PWM mode, refer to [Table 14-3 on page 84](#), and for phase correct PWM refer to [Table 14-4 on page 84](#).

A change of the COM0x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0x strobe bits.

14.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM02:0) and compare output mode (COM0x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM0x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (see [Section 14.6 "Compare Match Output Unit" on page 77](#)).

For detailed timing information refer to [Section 14.8 "Timer/Counter Timing Diagrams" on page 82](#).

14.7.1 Normal Mode

The simplest mode of operation is the normal mode (WGM02:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter overflow flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

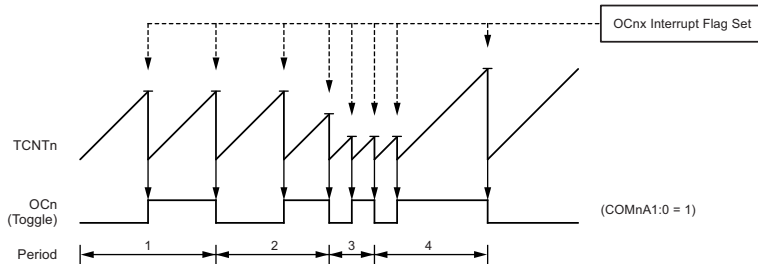
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

14.7.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM02:0 = 2), the OCR0A register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 14-5. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 14-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk_I/O} / 2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk_I/O}}{2 \times N \times (1 + OCRnx)}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

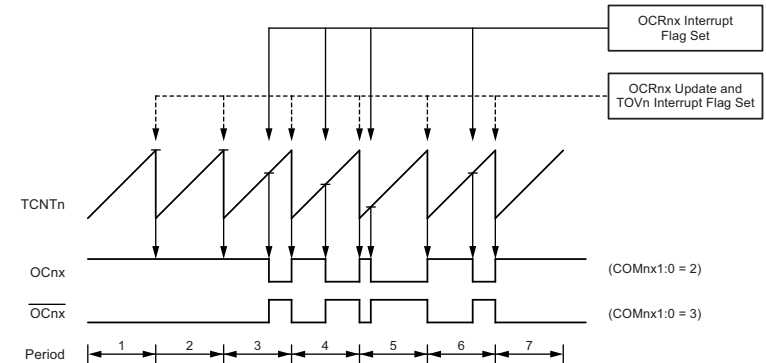
As for the normal mode of operation, the TOV0 flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

14.7.3 Fast PWM Mode

The fast pulse width modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting compare output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 14-6. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 14-6. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x1:0 to three. Setting the COM0A1:0 bits to one allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 14-6 on page 85). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \times 256}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

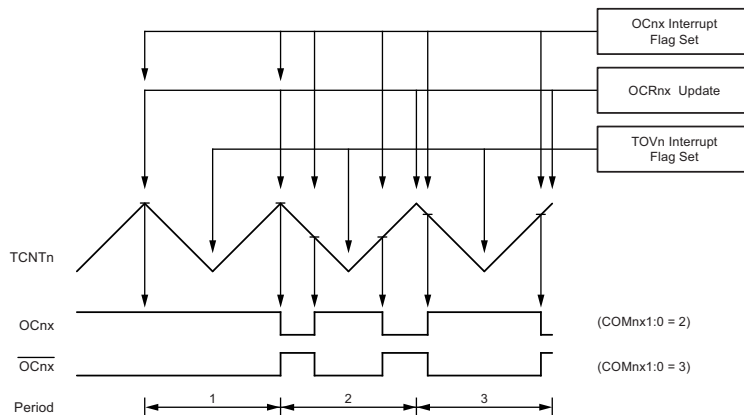
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each compare match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OCO} = f_{clk_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

14.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM02:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2:0 = 1, and OCR0A when WGM2:0 = 5. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x while upcounting, and set on the compare match while downcounting. In inverting output compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 14-7. The TCNT0 value in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 14-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches BOTTOM. The interrupt flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A0 bits to one allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 14-7 on page 85). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{N \times 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 14-7 on page 81 OCnx has a transition from high to low even though there is no compare match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without compare match.

- OCRnx changes its value from MAX, like in Figure 14-7 on page 81. When the OCR0A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCnx value at MAX must correspond to the result of an up-counting compare match.
- The timer starts counting from a value higher than the one in OCRnx, and for that reason misses the compare match and hence the OCnx change that would have happened on the way up.

14.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set. Figure 14-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 14-8. Timer/Counter Timing Diagram, no Prescaling

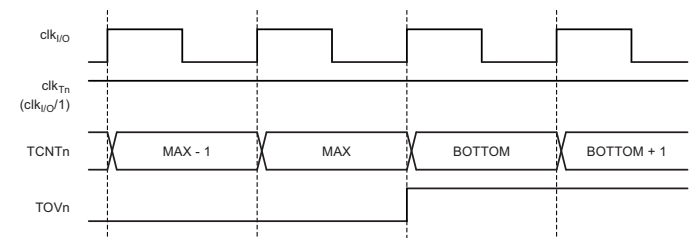


Figure 14-9 shows the same timing data, but with the prescaler enabled.

Figure 14-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)

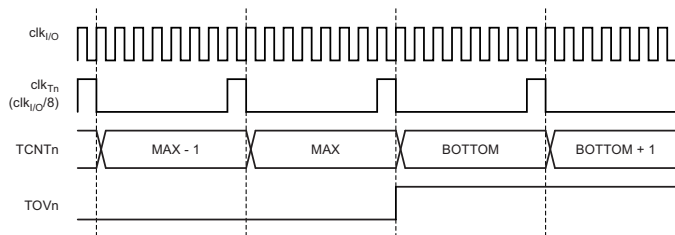


Figure 14-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 14-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk_I/O}/8$)

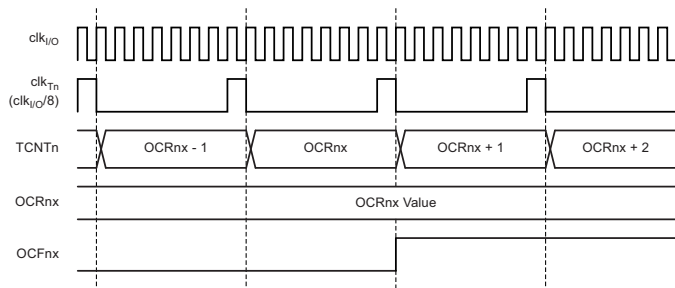
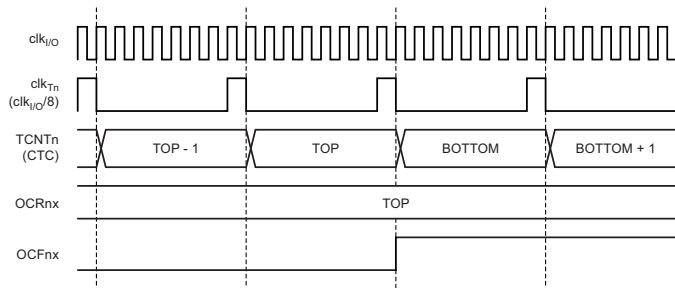


Figure 14-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 14-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk_I/O}/8$)



14.9 Register Description

14.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
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	

• Bits 7:6 – COM0A1:0: Compare Match Output A Mode

These bits control the output compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 14-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 14-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 14-3. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match, set OC0A at BOTTOM, (non-inverting mode).
1	1	Set OC0A on compare match, clear OC0A at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See Section 14.7.3 “Fast PWM Mode” on page 80 for more details.

Table 14-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 14-4. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match when up-counting. Set OC0A on compare match when down-counting.
1	1	Set OC0A on compare match when up-counting. Clear OC0A on compare match when down-counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 15.9.4 “Phase Correct PWM Mode” on page 103 for more details.

- **Bits 5:4 – COM0B1:0: Compare Match Output B Mode**

These bits control the output compare pin (OC0B) behavior. If one or both of the COM0B1:0 bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting. Table 14-5 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 14-5. Compare Output Mode, non-PWM Mode

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

Table 14-6 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 14-6. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match, set OC0B at BOTTOM, (non-inverting mode)
1	1	Set OC0B on compare match, clear OC0B at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 14.7.3 “Fast PWM Mode” on page 80 for more details.

Table 14-7 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 14-7. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match when up-counting. Set OC0B on compare match when down-counting.
1	1	Set OC0B on compare match when up-counting. Clear OC0B on compare match when down-counting.

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 14.7.4 “Phase Correct PWM Mode” on page 81 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

- **Bits 1:0 – WGM01:0: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 14-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 14.7 “Modes of Operation” on page 78).

Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on ⁽¹⁾⁽²⁾
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

14.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate compare match is forced on the waveform generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bit 6 – FOC0B: Force Output Compare B**

The FOC0B bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0B bit, an immediate compare match is forced on the waveform generation unit. The OC0B output is changed according to its COM0B1:0 bits setting. Note that the FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B1:0 bits that determines the effect of the forced compare.

A FOC0B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0B as TOP.

The FOC0B bit is always read as zero.

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega328P and will always read as zero.

- **Bit 3 – WGM02: Waveform Generation Mode**

See the description in the [Section 14.9.1 “TCR0A – Timer/Counter Control Register A”](#) on page 84.

- **Bits 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 14-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{IC}/(no\ prescaling)$
0	1	0	$clk_{IC}/8$ (from prescaler)
0	1	1	$clk_{IC}/64$ (from prescaler)
1	0	0	$clk_{IC}/256$ (from prescaler)
1	0	1	$clk_{IC}/1024$ (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

14.9.3 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	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 Timer/Counter register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0x registers.

14.9.4 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	OCR0A[7:0]								OCR0A
Read/Write	R/W	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 output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0A pin.

14.9.5 OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	OCR0B[7:0]								OCR0B
Read/Write	R/W	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 output compare register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0B pin.

14.9.6 TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
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 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

- **Bit 2 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the status register is set, the Timer/Counter compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter interrupt flag register – TIFR0.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

14.9.7 TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
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 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATmega328P and will always read as zero.

- **Bit 2 – OCF0B: Timer/Counter 0 Output Compare B Match Flag**

The OCF0B bit is set when a compare match occurs between the Timer/Counter and the data in OCR0B – output compare register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter compare B match interrupt enable), and OCF0B are set, the Timer/Counter compare match interrupt is executed.

- **Bit 1 – OCF0A: Timer/Counter 0 Output Compare A Match Flag**

The OCF0A bit is set when a compare match occurs between the Timer/Counter0 and the data in OCR0A – output compare register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 compare match interrupt enable), and OCF0A are set, the Timer/Counter0 compare match interrupt is executed.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 overflow interrupt enable), and TOV0 are set, the Timer/Counter0 overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to [Table 14-8 on page 86](#), [Section 14-8 “Waveform Generation Mode Bit Description”](#) on page 86.

The double buffered output compare registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pin (OC1A/B). See [Section 15.7 “Output Compare Units” on page 97](#). The compare match event will also set the compare match flag (OCF1A/B) which can be used to generate an output compare interrupt request.

The input capture register can capture the Timer/Counter value at a given external (edge triggered) event on either the input capture pin (ICP1) or on the analog comparator pins (see [Section 22. “Analog Comparator” on page 202](#)). The input capture unit includes a digital filtering unit (noise canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A register, the ICR1 register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 register can be used as an alternative, freeing the OCR1A to be used as PWM output.

15.2.2 Definitions

The following definitions are used extensively throughout the section:

Table 15-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 register. The assignment is dependent of the mode of operation.

15.3 Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR® CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit 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.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly Code Examples ⁽¹⁾
<pre>... ; Set TCNT1 to 0x01FF ldi r17,0x01 ldi r16,0xFF out TCNT1H,r17 out TCNT1L,r16 ; Read TCNT1 into r17:r16 in r16,TCNT1L in r17,TCNT1H ...</pre>
C Code Examples ⁽¹⁾
<pre>unsigned int i; ... /* Set TCNT1 to 0x01FF */ TCNT1 = 0x1FF; /* Read TCNT1 into i */ i = TCNT1; ...</pre>

Note: 1. See [Section 5. “About Code Examples” on page 8](#)
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBR”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNT1 register contents. Reading any of the OCR1A/B or ICR1 registers can be done by using the same principle.

Assembly Code Example⁽¹⁾ <pre> TIM16_ReadTCNT1: ; Save global interrupt flag in r18,SREG ; Disable interrupts cli ; Read TCNT1 into r17:r16 in r16,TCNT1L in r17,TCNT1H ; Restore global interrupt flag out SREG,r18 ret </pre>
C Code Example⁽¹⁾ <pre> unsigned int TIM16_ReadTCNT1(void) { unsigned char sreg; unsigned int i; /* Save global interrupt flag */ sreg = SREG; /* Disable interrupts */ _CLI(); /* Read TCNT1 into i */ i = TCNT1; /* Restore global interrupt flag */ SREG = sreg; return i; } </pre>

Note: 1. See [Section 5. “About Code Examples” on page 8](#).
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBR”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNT1 register contents. Writing any of the OCR1A/B or ICR1 registers can be done by using the same principle.

Assembly Code Example⁽¹⁾ <pre> TIM16_WriteTCNT1: ; Save global interrupt flag in r18,SREG ; Disable interrupts cli ; Set TCNT1 to r17:r16 out TCNT1H,r17 out TCNT1L,r16 ; Restore global interrupt flag out SREG,r18 ret </pre>
C Code Example⁽¹⁾ <pre> void TIM16_WriteTCNT1(unsigned int i) { unsigned char sreg; unsigned int i; /* Save global interrupt flag */ sreg = SREG; /* Disable interrupts */ _CLI(); /* Set TCNT1 to i */ TCNT1 = i; /* Restore global interrupt flag */ SREG = sreg; } </pre>

Note: 1. See [Section 5. “About Code Examples” on page 8](#).
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBR”, “SBRC”, “SBR”, and “CBR”.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

15.3.1 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

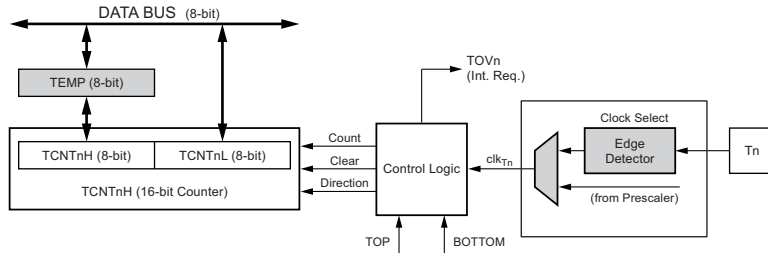
15.4 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS12:0) bits located in the Timer/Counter control register B (TCCR1B). For details on clock sources and prescaler, see [Section 16. “Timer/Counter0 and Timer/Counter1 Prescalers” on page 114](#).

15.5 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 15-2 shows a block diagram of the counter and its surroundings.

Figure 15-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_{T1}	Timer/Counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter high (TCNT1H) containing the upper eight bits of the counter, and counter low (TCNT1L) containing the lower eight bits. The TCNT1H register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the clock select bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the waveform generation mode bits (WGM13:0) located in the Timer/Counter control registers A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see [Section 15.9 "Modes of Operation" on page 100](#).

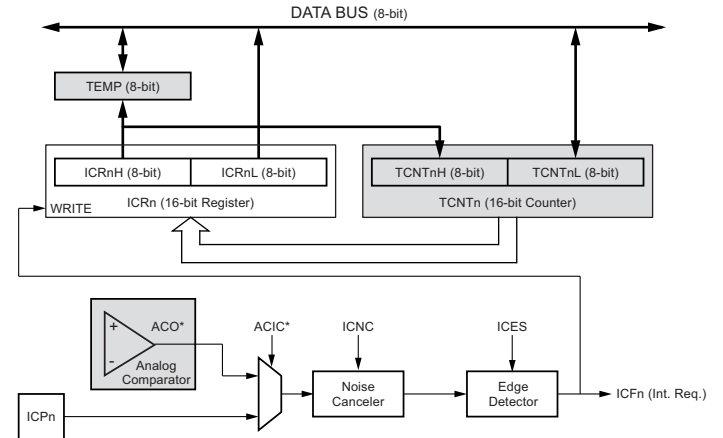
The Timer/Counter overflow flag (TOV1) is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

15.6 Input Capture Unit

The Timer/Counter incorporates an input capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The input capture unit is illustrated by the block diagram shown in [Figure 15-3 on page 96](#). The elements of the block diagram that are not directly a part of the input capture unit are gray shaded. The small "n" in register and bit names indicates the Timer/Counter number.

Figure 15-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the input capture pin (ICP1), alternatively on the analog comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the input capture register (ICR1). The input capture flag (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 register. If enabled (ICIE1 = 1), the input capture flag generates an input capture interrupt. The ICF1 flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the input capture register (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP register.

The ICR1 register can only be written when using a waveform generation mode that utilizes the ICR1 register for defining the counter's TOP value. In these cases the waveform generation mode (WGM13:0) bits must be set before the TOP value can be written to the ICR1 register. When writing the ICR1 register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to [Section 15.3 "Accessing 16-bit Registers" on page 91](#).

15.6.1 Input Capture Trigger Source

The main trigger source for the input capture unit is the input capture pin (ICP1). Timer/Counter1 can alternatively use the analog comparator output as trigger source for the input capture unit. The analog comparator is selected as trigger source by setting the analog comparator input capture (ACIC) bit in the analog comparator control and status register (ACSR). Be aware that changing trigger source can trigger a capture. The input capture flag must therefore be cleared after the change.

Both the input capture pin (ICP1) and the analog comparator output (ACO) inputs are sampled using the same technique as for the T1 pin ([Figure 16-1 on page 114](#)). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a waveform generation mode that uses ICR1 to define TOP.

An input capture can be triggered by software by controlling the port of the ICP1 pin.

15.6.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the input capture noise canceler (ICNC1) bit in Timer/Counter control register B (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

15.6.3 Using the Input Capture Unit

The main challenge when using the input capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the input capture interrupt, the ICR1 register should be read as early in the interrupt handler routine as possible. Even though the input capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the input capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 register has been read. After a change of the edge, the input capture flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 flag is not required (if an interrupt handler is used).

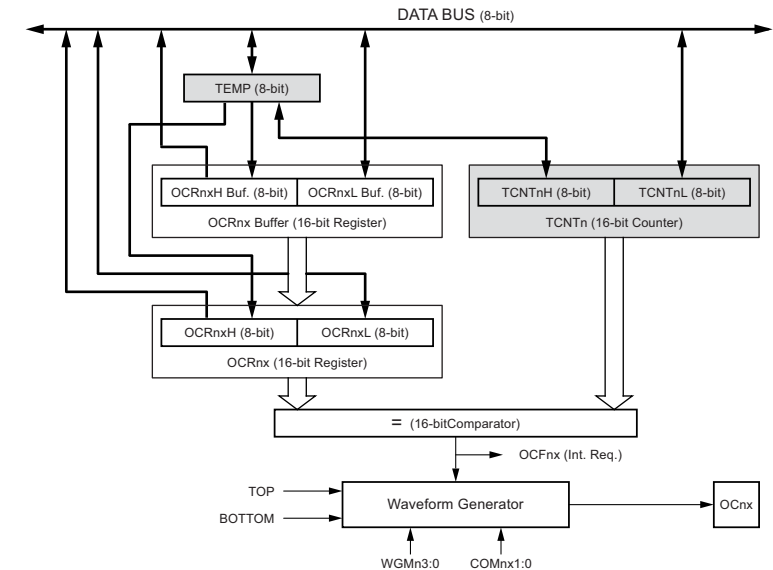
15.7 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the output compare register (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the output compare flag (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the output compare flag generates an output compare interrupt. The OCF1x flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the waveform generation mode (WGM13:0) bits and compare output mode (COM1x1:0) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (see [Section 15.9 "Modes of Operation" on page 100](#)).

A special feature of output compare unit A allows it to define the Timer/Counter TOP value (i.e., counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the waveform generator.

[Figure 15-4 on page 98](#) shows a block diagram of the output compare unit. The small "n" in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the "x" indicates output compare unit (A/B). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 15-4. Output Compare Unit, Block Diagram



The OCR1x register is double buffered when using any of the twelve pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x compare register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1x register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x buffer register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (buffer or compare) register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 register). Therefore OCR1x is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCR1x registers must be done via the TEMP register since the compare of all 16 bits is done continuously. The high byte (OCR1xH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP register will be updated by the value written. Then when the low byte (OCR1xL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x compare register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to [Section 15.3 "Accessing 16-bit Registers" on page 91](#).

15.7.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the force output compare (FOC1x) bit. Forcing compare match will not set the OCF1x flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the COM11:0 bits settings define whether the OC1x pin is set, cleared or toggled).

15.7.2 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

15.7.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the output compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

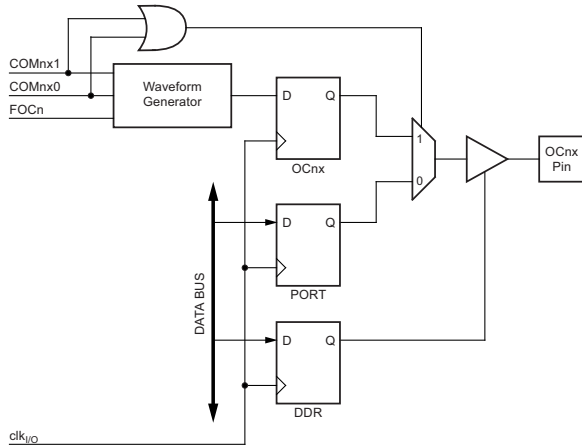
The setup of the OC1x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC1x value is to use the force output compare (FOC1x) strobe bits in normal mode. The OC1x register keeps its value even when changing between waveform generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

15.8 Compare Match Output Unit

The compare output mode (COM1x1:0) bits have two functions. The waveform generator uses the COM1x1:0 bits for defining the output compare (OC1x) state at the next compare match. Secondly the COM1x1:0 bits control the OC1x pin output source. [Figure 15-5](#) shows a simplified schematic of the logic affected by the COM1x1:0 bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x register, not the OC1x pin. If a system reset occurs, the OC1x register is reset to "0".

Figure 15-5. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC1x) from the waveform generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the waveform generation mode, but there are some exceptions. Refer to [Table 15-2 on page 108](#), [Table 15-3 on page 108](#) and [Table 15-4 on page 109](#) for details.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. See [Section 15.11 "Register Description" on page 108](#). The COM1x1:0 bits have no effect on the input capture unit.

15.8.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the waveform generator that no action on the OC1x register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to [Table 15-2 on page 108](#). For fast PWM mode refer to [Table 15-3 on page 108](#), and for phase correct and phase and frequency correct PWM refer to [Table 15-4 on page 109](#).

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

15.9 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM13:0) and compare output mode (COM1x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (see [Section 15.8 "Compare Match Output Unit" on page 99](#)). For detailed timing information refer to [Section 15.10 "Timer/Counter Timing Diagrams" on page 106](#).

15.9.1 Normal Mode

The simplest mode of operation is the normal mode (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the Timer/Counter overflow flag (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The input capture unit is easy to use in normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

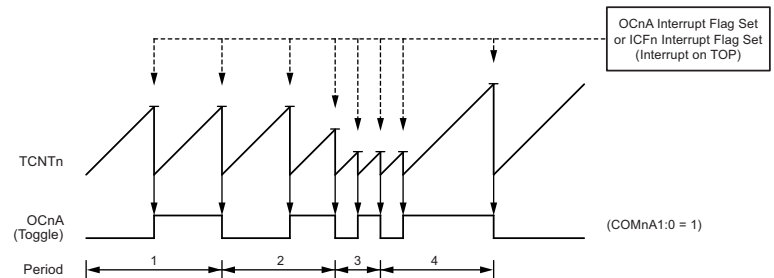
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

15.9.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in [Figure 15-6](#). The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 15-6. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk_I/O}}{2 \times N \times (1 + OCRnA)}$$

The N variable represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the normal mode of operation, the TOV1 flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

15.9.3 Fast PWM Mode

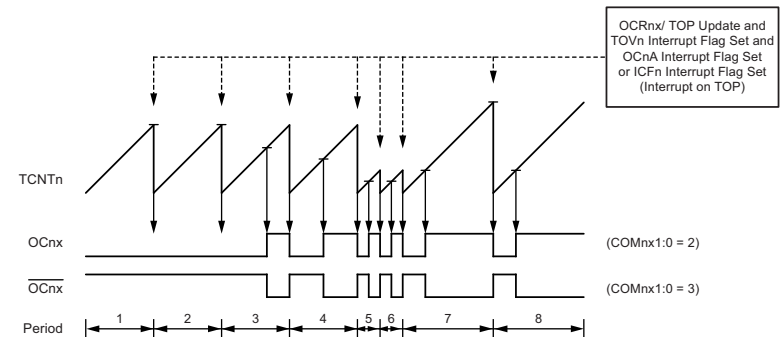
The fast pulse width modulation or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x, and set at BOTTOM. In inverting compare output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

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

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 15-7 on page 102. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 15-7. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set each time the counter reaches TOP. In addition the OC1A or ICF1 flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A buffer register. The OCR1A compare register will then be updated with the value in the buffer register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 flag is set.

Using the ICR1 register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce an inverted PWM and an non-inverted PWM output can be generated by setting the COM1x1:0 to three (see Table on page 108). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnXPWM} = \frac{f_{clk_I/O}}{N \times (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each compare match (COM1A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

15.9.4 Phase Correct PWM Mode

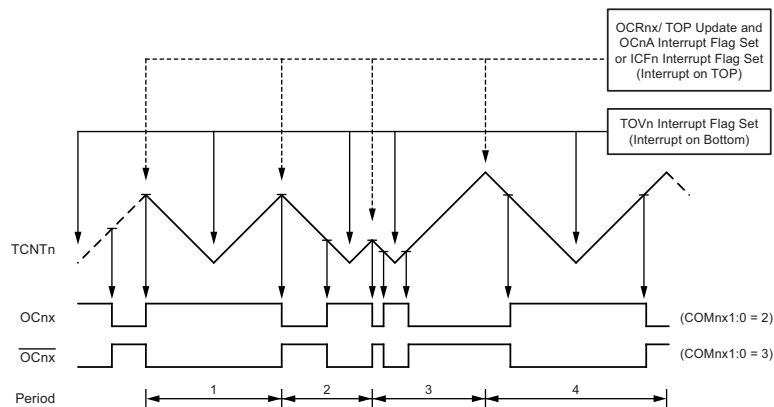
The phase correct pulse width modulation or phase correct PWM mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting output compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 15-8. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 15-8. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag is set accordingly at the same timer clock cycle as the OCR1x registers are updated with the double buffer value (at TOP). The interrupt flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x registers are written. As the third period shown in Figure 15-8 on page 103 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR1x register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table on page 109). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{2 \times N \times TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

15.9.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct pulse width modulation, or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting compare output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

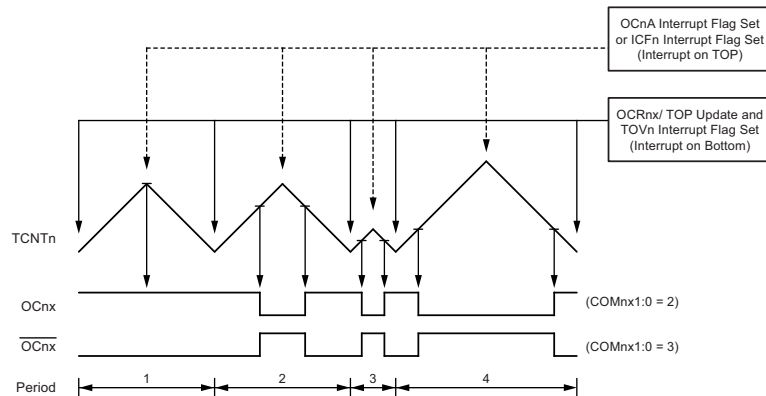
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x register is updated by the OCR1x buffer register, (see Figure 15-8 on page 103 and Figure 15-9 on page 105).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following equation:

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 15-9 on page 105. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 15-9. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set at the same timer clock cycle as the OCR1x registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag set when TCNT1 has reached TOP. The interrupt flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x.

As Figure 15-9 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

Using the ICR1 register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table on page 109). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{\text{OCnPF PWM}} = \frac{f_{\text{clk I/O}}}{2 \times N \times \text{TOP}}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

15.10 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR1x register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 15-10 shows a timing diagram for the setting of OCF1x.

Figure 15-10. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling

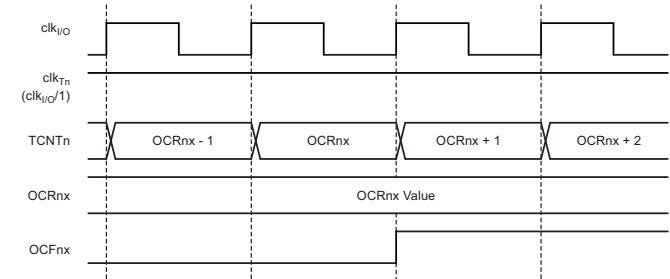


Figure 15-11 shows the same timing data, but with the prescaler enabled.

Figure 15-11. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{\text{clk I/O}}/8$)

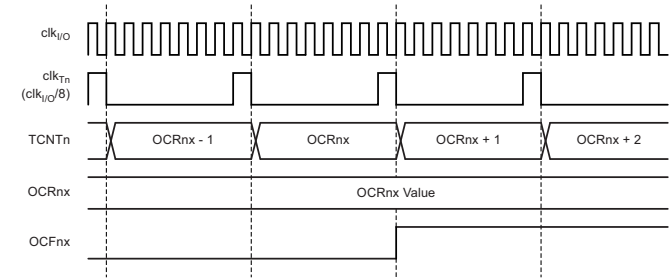


Figure 15-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 flag at BOTTOM.

Figure 15-12. Timer/Counter Timing Diagram, no Prescaling

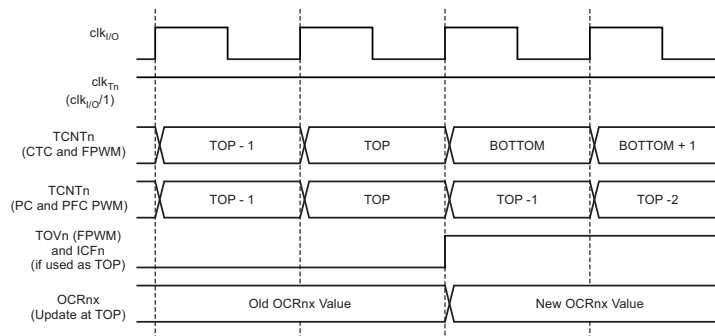
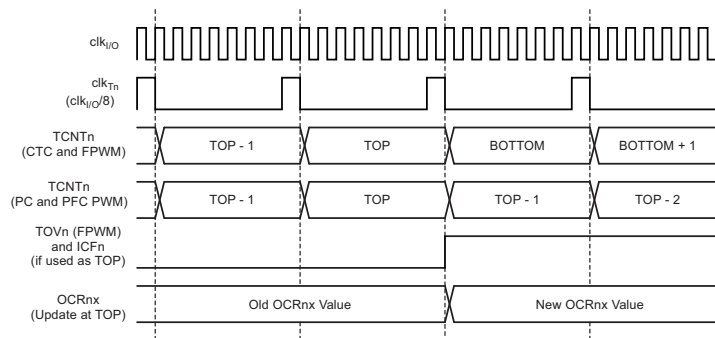


Figure 15-13 shows the same timing data, but with the prescaler enabled.

Figure 15-13. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)



15.11 Register Description

15.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
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 7:6 – COM1A1:0: Compare Output Mode for Channel A

• Bit 5:4 – COM1B1:0: Compare Output Mode for Channel B

The COM1A1:0 and COM1B1:0 control the output compare pins (OC1A and OC1B respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent of the WGM13:0 bits setting. Table 15-2 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 15-2. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match.
1	0	Clear OC1A/OC1B on compare match (set output to low level).
1	1	Set OC1A/OC1B on compare match (set output to high level).

Table 15-3 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 15-3. Compare Output Mode, Fast PWM⁽¹⁾

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on compare match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See Section 15.9.3 "Fast PWM Mode" on page 101 for more details.

Table 15-4 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 15-4. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM⁽¹⁾

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 11: Toggle OC1A on compare match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when down counting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when down counting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See [Section 15.9.4 "Phase Correct PWM Mode" on page 103](#) for more details.

• **Bit 1:0 – WGM11:0: Waveform Generation Mode**

Combined with the WGM13:2 bits found in the TCCR1B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see [Table 15-5](#). Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and three types of pulse width modulation (PWM) modes. See [Section 15.9 "Modes of Operation" on page 100](#).

Table 15-5. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

15.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
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 – ICNC1: Input Capture Noise Canceler**

Setting this bit (to one) activates the input capture noise canceler. When the noise canceler is activated, the input from the input capture pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The input capture is therefore delayed by four oscillator cycles when the noise canceler is enabled.

• **Bit 6 – ICES1: Input Capture Edge Select**

This bit selects which edge on the input capture pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the input capture register (ICR1). The event will also set the input capture flag (ICF1), and this can be used to cause an input capture interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B register), the ICP1 is disconnected and consequently the input capture function is disabled.

• **Bit 5 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

• **Bit 4:3 – WGM13:2: Waveform Generation Mode**

See TCCR1A register description.

• **Bit 2:0 – CS12:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter, see [Figure 15-10 on page 106](#) and [Figure 15-11 on page 106](#).

Table 15-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{IO} /1 (no prescaling)
0	1	0	clk _{IO} /8 (from prescaler)
0	1	1	clk _{IO} /64 (from prescaler)
1	0	0	clk _{IO} /256 (from prescaler)
1	0	1	clk _{IO} /1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

15.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**
- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in clear timer on compare match (CTC) mode using OCR1A as TOP. The FOC1A/FOC1B bits are always read as zero.

15.11.4 TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	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 two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See [Section 15.3 “Accessing 16-bit Registers” on page 91](#).

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x registers.

Writing to the TCNT1 register blocks (removes) the compare match on the following timer clock for all compare units.

15.11.5 OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

15.11.6 OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	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 output compare registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC1x pin.

The output compare registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See [Section 15.3 “Accessing 16-bit Registers” on page 91](#).

15.11.7 ICR1H and ICR1L – Input Capture Register 1

Bit	7	6	5	4	3	2	1	0	
(0x87)	ICR1[15:8]								ICR1H
(0x86)	ICR1[7:0]								ICR1L
Read/Write	R/W	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 input capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the analog comparator output for Timer/Counter1). The input capture can be used for defining the counter TOP value.

The input capture register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See [Section 15.3 “Accessing 16-bit Registers” on page 91](#).

15.11.8 TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- **Bit 5 – ICIE1: Timer/Counter1, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 input capture interrupt is enabled. The corresponding interrupt vector (see [Section 11. “Interrupts” on page 49](#)) is executed when the ICF1 flag, located in TIFR1, is set.

- **Bit 4, 3 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega328P, and will always read as zero.

- **Bit 2 – OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 output compare B match interrupt is enabled. The corresponding interrupt vector (see [Section 11. “Interrupts” on page 49](#)) is executed when the OCF1B flag, located in TIFR1, is set.

- **Bit 1 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 output compare A match interrupt is enabled. The corresponding interrupt vector (see [Section 11. “Interrupts” on page 49](#)) is executed when the OCF1A flag, located in TIFR1, is set.

- **Bit 0 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 overflow interrupt is enabled. The corresponding interrupt vector (see [Section 11. “Interrupts” on page 49](#)) is executed when the TOV1 flag, located in TIFR1, is set.

15.11.9 TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- **Bit 5 – ICF1: Timer/Counter1, Input Capture Flag**

This flag is set when a capture event occurs on the ICP1 pin. When the input capture register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the input capture interrupt vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

- **Bit 4, 3 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega328P, and will always read as zero.

- **Bit 2 – OCF1B: Timer/Counter1, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register B (OCR1B).

Note that a forced output compare (FOC1B) strobe will not set the OCF1B flag.

OCF1B is automatically cleared when the output compare match B interrupt vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF1A: Timer/Counter1, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register A (OCR1A).

Note that a forced output compare (FOC1A) strobe will not set the OCF1A flag.

OCF1A is automatically cleared when the output compare match A interrupt vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bits setting. In normal and CTC modes, the TOV1 flag is set when the timer overflows. Refer to [Table 15-5 on page 109](#) for the TOV1 flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

16. Timer/Counter0 and Timer/Counter1 Prescalers

[Section 14. “8-bit Timer/Counter0 with PWM” on page 74](#) and [Section 15. “16-bit Timer/Counter1 with PWM” on page 89](#) share the same prescaler module, but the Timer/Counter0 can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

16.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency (f_{CLK_IO}). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK_IO}/8$, $f_{CLK_IO}/64$, $f_{CLK_IO}/256$, or $f_{CLK_IO}/1024$.

16.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler ($6 > CSn2:0 > 1$). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

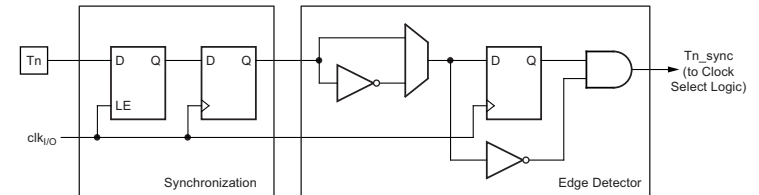
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counter0s it is connected to.

16.3 External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. [Figure 16-1](#) shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock (clk_{IO}). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T1}/clk_{T0} pulse for each positive ($CSn2:0 = 7$) or negative ($CSn2:0 = 6$) edge it detects.

Figure 16-1. T1/T0 Pin Sampling



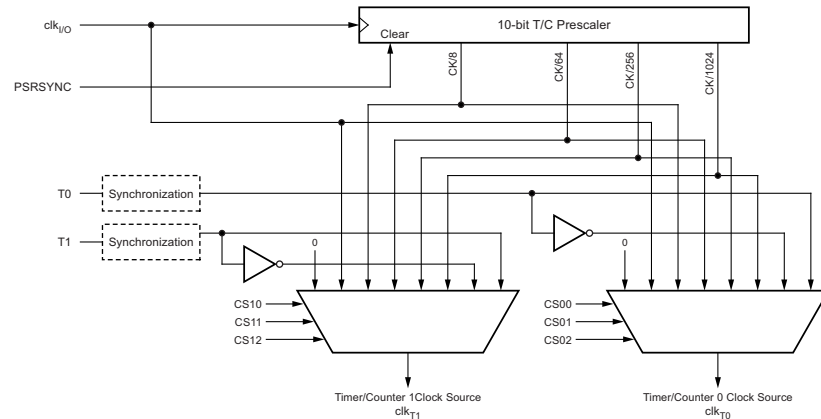
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk_IO}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk_IO}/2.5$.

An external clock source can not be prescaled.

Figure 16-2. Prescaler for Timer/Counter0 and Timer/Counter1⁽¹⁾



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 16-1.

16.4 Register Description

16.4.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	–	–	–	–	–	PSRASY	PSRSYNC	GTCCR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – TSM: Timer/Counter Synchronization Mode

Writing the TSM bit to one activates the Timer/Counter synchronization mode. In this mode, the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counter are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware, and the Timer/Counter start counting simultaneously.

• Bit 0 – PSRSYNC: Prescaler Reset

When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

17. 8-bit Timer/Counter2 with PWM and Asynchronous Operation

17.1 Features

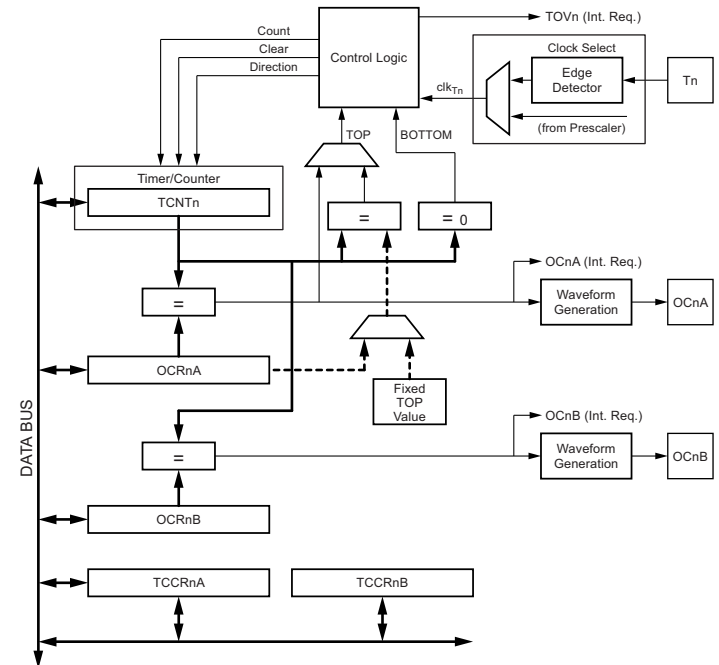
- Single channel counter
- Clear timer on compare match (auto reload)
- Glitch-free, phase correct pulse width modulator (PWM)
- Frequency generator
- 10-bit Clock prescaler
- Overflow and compare match interrupt sources (TOV2, OCF2A and OCF2B)
- Allows clocking from external 32kHz watch crystal independent of the I/O clock

17.2 Overview

Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 17-1. For the actual placement of I/O pins, refer to Section 1-1 “Pinout” on page 3. CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in the Section 17.11 “Register Description” on page 127.

The PRTIM2 bit in Section 9.10 “Minimizing Power Consumption” on page 36 must be written to zero to enable Timer/Counter2 module.

Figure 17-1. 8-bit Timer/Counter Block Diagram



17.2.1 Registers

The Timer/Counter (TCNT2) and output compare register (OCR2A and OCR2B) are 8-bit registers. Interrupt request (shorten as int.Req.) signals are all visible in the timer interrupt flag register (TIFR2). All interrupts are individually masked with the timer interrupt mask register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the asynchronous status register (ASSR). The clock select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{TC2}).

The double buffered output compare register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins (OC2A and OC2B). See [Section 17.5 "Output Compare Unit" on page 118](#) for details. The compare match event will also set the compare flag (OCF2A or OCF2B) which can be used to generate an output compare interrupt request.

17.2.2 Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT2 for accessing Timer/Counter2 counter value and so on.

The definitions in [Table 17-1](#) are also used extensively throughout the section.

Table 17-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A register. The assignment is dependent on the mode of operation.

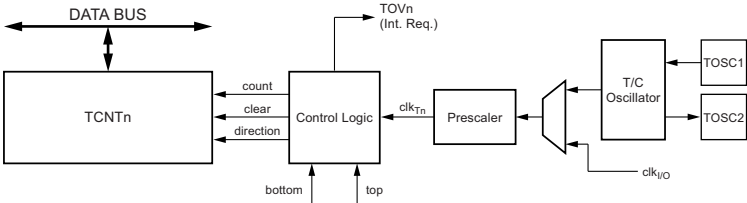
17.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{TC2} is by default equal to the MCU clock, clk_{IO}. When the AS2 bit in the ASSR register is written to logic one, the clock source is taken from the Timer/Counter oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see [Section 17.11.8 "ASSR – Asynchronous Status Register" on page 133](#). For details on clock sources and prescaler, see [Section 17.10 "Timer/Counter Prescaler" on page 127](#).

17.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. [Figure 17-2 on page 117](#) shows a block diagram of the counter and its surrounding environment.

Figure 17-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clTCn	Timer/Counter clock, referred to as clk _{TC2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{TC2}). clk_{TC2} can be generated from an external or internal clock source, selected by the clock select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{TC2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter control register (TCCR2A) and the WGM22 located in the Timer/Counter control register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see [Section 17.7 "Modes of Operation" on page 120](#).

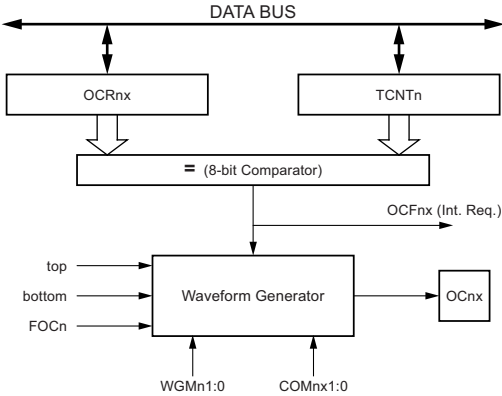
The Timer/Counter overflow flag (TOV2) is set according to the mode of operation selected by the WGM22:0 bits. TOV2 can be used for generating a CPU interrupt.

17.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the output compare register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the output compare flag (OCF2A or OCF2B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the output compare flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM22:0 bits and compare output mode (COM2x1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation ([Section 17.7 "Modes of Operation" on page 120](#)).

[Figure 17-3](#) shows a block diagram of the output compare unit.

Figure 17-3. Output Compare Unit, Block Diagram



The OCR2x register is double buffered when using any of the pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2x compare register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2x register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR2x buffer register, and if double buffering is disabled the CPU will access the OCR2x directly.

17.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the force output compare (FOC2x) bit. Forcing compare match will not set the OCF2x flag or reload/clear the timer, but the OC2x pin will be updated as if a real compare match had occurred (the COM2x1:0 bits settings define whether the OC2x pin is set, cleared or toggled).

17.5.2 Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2x to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

17.5.3 Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the output compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.

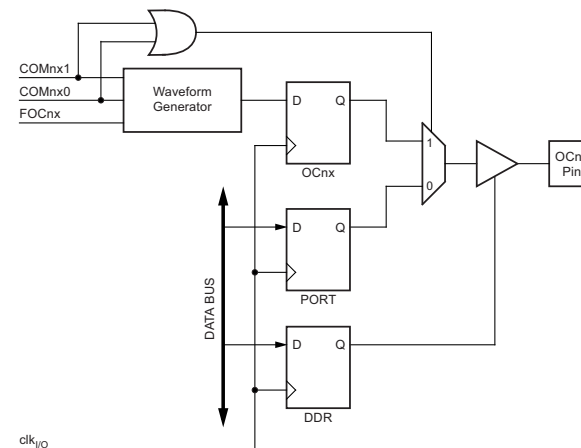
The setup of the OC2x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC2x value is to use the force output compare (FOC2x) strobe bit in normal mode. The OC2x register keeps its value even when changing between waveform generation modes.

Be aware that the COM2x1:0 bits are not double buffered together with the compare value. Changing the COM2x1:0 bits will take effect immediately.

17.6 Compare Match Output Unit

The compare output mode (COM2x1:0) bits have two functions. The waveform generator uses the COM2x1:0 bits for defining the output compare (OC2x) state at the next compare match. Also, the COM2x1:0 bits control the OC2x pin output source. [Figure 17-4](#) shows a simplified schematic of the logic affected by the COM2x1:0 bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM2x1:0 bits are shown. When referring to the OC2x state, the reference is for the internal OC2x register, not the OC2x pin.

Figure 17-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC2x) from the waveform generator if either of the COM2x1:0 bits are set. However, the OC2x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC2x pin (DDR_OC2x) must be set as output before the OC2x value is visible on the pin. The port override function is independent of the waveform generation mode.

The design of the output compare pin logic allows initialization of the OC2x state before the output is enabled. Note that some COM2x1:0 bit settings are reserved for certain modes of operation. See [Section 17.11 "Register Description" on page 127](#).

17.6.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM2x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM2x1:0 = 0 tells the waveform generator that no action on the OC2x register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to [Table 17-5 on page 129](#). For fast PWM mode, refer to [Table 17-6 on page 129](#), and for phase correct PWM refer to [Table 17-7 on page 129](#).

A change of the COM2x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC2x strobe bits.

17.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM22:0) and compare output mode (COM2x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM2x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM2x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (see [Section 17.6 "Compare Match Output Unit" on page 119](#)).

For detailed timing information refer to [Section 17.8 "Timer/Counter Timing Diagrams" on page 124](#).

17.7.1 Normal Mode

The simplest mode of operation is the normal mode (WGM22:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter overflow flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

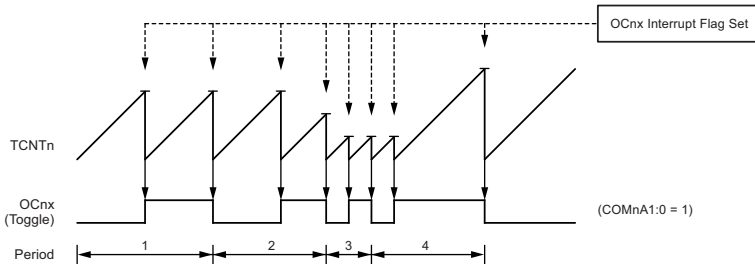
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

17.7.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM22:0 = 2), the OCR2A register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 17-5. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, then counter (TCNT2) is cleared.

Figure 17-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2A is lower than the current value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC2A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM2A1:0 = 1). The OC2A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC2A} = f_{clk_I/O} / 2$ when OCR2A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk_I/O}}{2 \times N \times (1 + OCRnx)}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

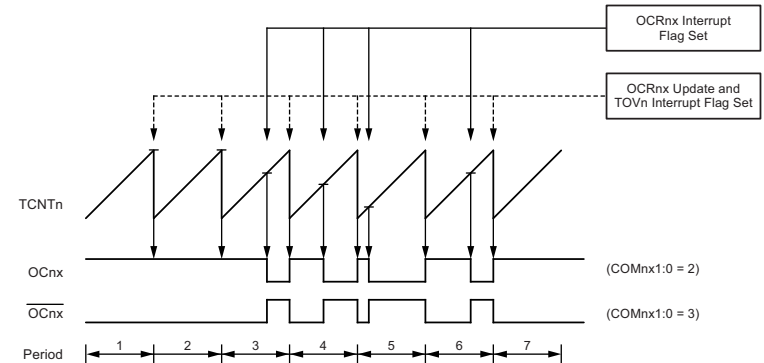
As for the normal mode of operation, the TOV2 flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

17.7.3 Fast PWM Mode

The fast pulse width modulation or fast PWM mode (WGM22:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR2A when WGM2:0 = 7. In non-inverting compare output mode, the output compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x, and set at BOTTOM. In inverting compare output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 17-6. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2x and TCNT2.

Figure 17-6. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV2) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM2:0 = 3, and OCR2A when WGM2:0 = 7. (See Table 17-3 on page 128). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2x register at the compare match between OCR2x and TCNT2, and clearing (or setting) the OC2x register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \times 256}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR2A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR2A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM2A1:0 bits.)

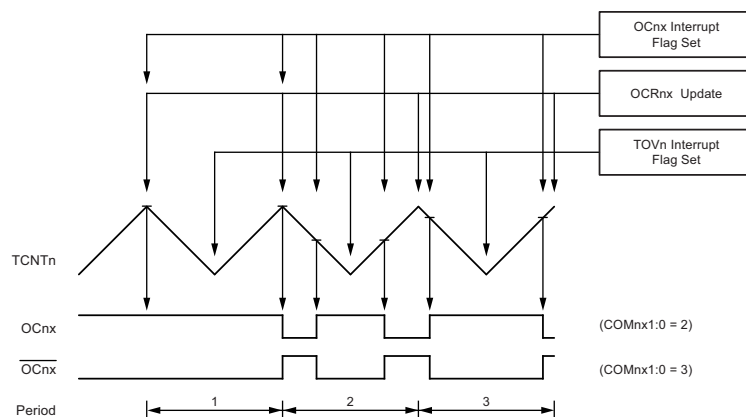
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2x to toggle its logical level on each compare match (COM2x1:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk_I/O}/2$ when OCR2A is set to zero. This feature is similar to the OC2A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

17.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM22:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR2A when MGM2:0 = 7. In non-inverting compare output mode, the output compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x while upcounting, and set on the compare match while downcounting. In inverting output compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT2 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on [Figure 17-7](#). The TCNT2 value in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2x and TCNT2.

Figure 17-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV2) is set each time the counter reaches BOTTOM. The interrupt flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM2:0 = 3, and OCR2A when MGM2:0 = 7 (See [Table 17-4 on page 128](#)). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2x register at the compare match between OCR2x and TCNT2 when the counter increments, and setting (or clearing) the OC2x register at compare match between OCR2x and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{N \times 510}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in [Figure 17-7 on page 123](#) OCnx has a transition from high to low even though there is no compare match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without compare match.

- OCR2A changes its value from MAX, like in [Figure 17-7 on page 123](#). When the OCR2A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting compare match.
- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the compare match and hence the OCn change that would have happened on the way up.

17.8 Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter oscillator clock. The figures include information on when interrupt flags are set. [Figure 17-8](#) contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 17-8. Timer/Counter Timing Diagram, no Prescaling

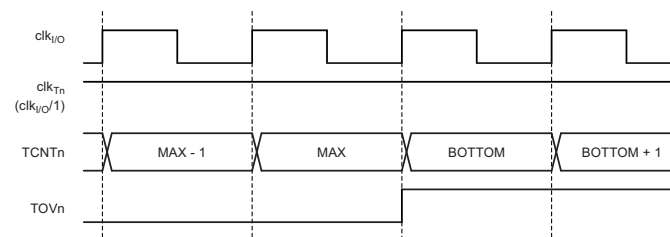


Figure 17-9 shows the same timing data, but with the prescaler enabled.

Figure 17-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_i/o/8}$)

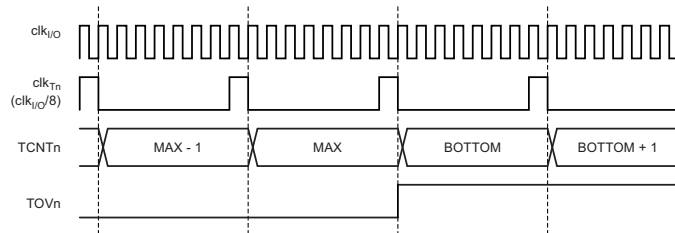


Figure 17-10 shows the setting of OCF2A in all modes except CTC mode.

Figure 17-10. Timer/Counter Timing Diagram, Setting of OCF2A, with Prescaler ($f_{clk_i/o/8}$)

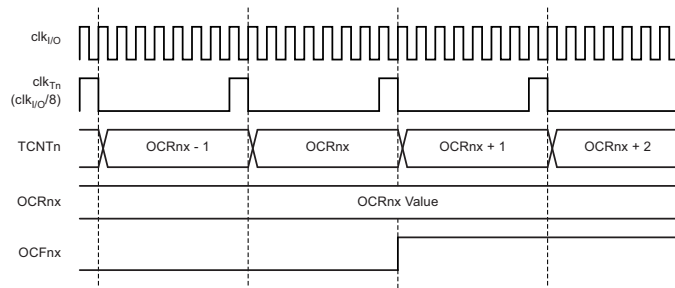
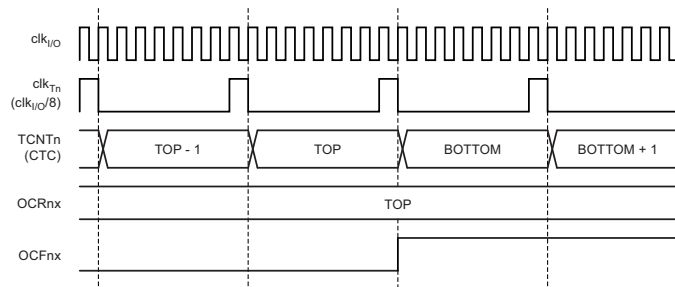


Figure 17-11 shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

Figure 17-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler ($f_{clk_i/o/8}$)



17.9 Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

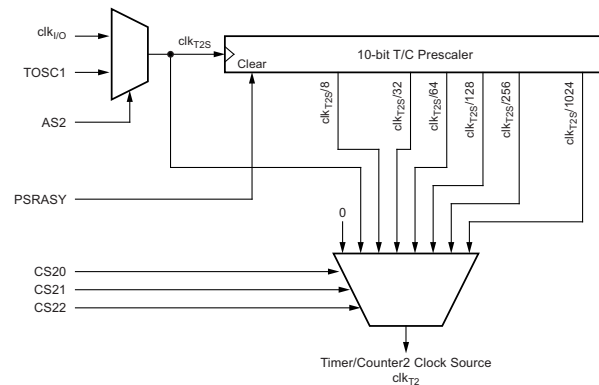
- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers $TCNT2$, $OCR2x$, and $TCCR2x$ might be corrupted. A safe procedure for switching clock source is:
 - a. Disable the Timer/Counter2 interrupts by clearing $OCIE2x$ and $TOIE2$.
 - b. Select clock source by setting $AS2$ as appropriate.
 - c. Write new values to $TCNT2$, $OCR2x$, and $TCCR2x$.
 - d. To switch to asynchronous operation: Wait for $TCN2xUB$, $OCR2xUB$, and $TCCR2xUB$.
 - e. Clear the Timer/Counter2 interrupt flags.
 - f. Enable interrupts, if needed.
- The CPU main clock frequency must be more than four times the oscillator frequency.
- When writing to one of the registers $TCNT2$, $OCR2x$, or $TCCR2x$, the value is transferred to a temporary register, and latched after two positive edges on $TOSC1$. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the five mentioned registers have their individual temporary register, which means that e.g. writing to $TCNT2$ does not disturb an $OCR2x$ write in progress. To detect that a transfer to the destination register has taken place, the asynchronous status register – $ASSR$ has been implemented.
- When entering power-save or ADC noise reduction mode after having written to $TCNT2$, $OCR2x$, or $TCCR2x$, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if any of the output compare2 interrupt is used to wake up the device, since the output compare function is disabled during writing to $OCR2x$ or $TCNT2$. If the write cycle is not finished, and the MCU enters sleep mode before the corresponding $OCR2xUB$ bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from power-save or ADC noise reduction mode, precautions must be taken if the user wants to re-enter one of these modes: If re-entering sleep mode within the $TOSC1$ cycle, the interrupt will immediately occur and the device wake up again. The result is multiple interrupts and wake-ups within one $TOSC1$ cycle from the first interrupt. If the user is in doubt whether the time before re-entering power-save or ADC noise reduction mode is sufficient, the following algorithm can be used to ensure that one $TOSC1$ cycle has elapsed:
 - a. Write a value to $TCCR2x$, $TCNT2$, or $OCR2x$.
 - b. Wait until the corresponding update busy flag in $ASSR$ returns to zero.
 - c. Enter power-save or ADC noise reduction mode.
- When the asynchronous operation is selected, the 32.768kHz oscillator for Timer/Counter2 is always running, except in power-down and standby modes. After a power-up reset or wake-up from power-down or standby mode, the user should be aware of the fact that this oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from power-down or standby mode. The contents of all Timer/Counter2 registers must be considered lost after a wake-up from power-down or standby mode due to unstable clock signal upon start-up, no matter whether the oscillator is in use or a clock signal is applied to the $TOSC1$ pin.
- Description of wake up from power-save or ADC noise reduction mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the $TCNT2$ register shortly after wake-up from power-save may give an incorrect result. Since $TCNT2$ is clocked on the asynchronous $TOSC$ clock, reading $TCNT2$ must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising $TOSC1$ edge. When waking up from power-save mode, and the I/O clock ($clk_{i/o}$) again becomes active, $TCNT2$ will read as the previous value (before entering sleep) until the next rising $TOSC1$ edge. The phase of the $TOSC$ clock after waking up from power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading $TCNT2$ is thus as follows:

- Write any value to either of the registers OCR2x or TCCR2x.
- Wait for the corresponding update busy flag to be cleared.
- Read TCNT2.

During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timer takes 3 processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the interrupt flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.

17.10 Timer/Counter Prescaler

Figure 17-12. Prescaler for Timer/Counter2



The clock source for Timer/Counter2 is named clk_{T2S} . clk_{T2S} is by default connected to the main system I/O clock $clk_{I/O}$. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the TOSC1 pin. This enables use of Timer/Counter2 as a real time counter (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from port C. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The oscillator is optimized for use with a 32.768kHz crystal.

For Timer/Counter2, the possible prescaled selections are: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, and $clk_{T2S}/1024$. Additionally, clk_{T2S} as well as 0 (stop) may be selected. Setting the PSRASY bit in GTCCR resets the prescaler. This allows the user to operate with a predictable prescaler.

17.11 Register Description

17.11.1 TCCR2A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
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	

• Bits 7:6 – COM2A1:0: Compare Match Output A Mode

These bits control the output compare pin (OC2A) behavior. If one or both of the COM2A1:0 bits are set, the OC2A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC2A pin must be set in order to enable the output driver.

When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM22:0 bit setting. Table 17-2 shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM).

Table 17-2. Compare Output Mode, non-PWM Mode

COM2A1	COM2A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC2A on compare match
1	0	Clear OC2A on compare match
1	1	Set OC2A on compare match

Table 17-3 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 17-3. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal port operation, OC0A disconnected. WGM22 = 1: Toggle OC2A on compare match.
1	0	Clear OC2A on compare match, set OC2A at BOTTOM, (non-inverting mode).
1	1	Set OC2A on compare match, clear OC2A at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See Section 17.7.3 “Fast PWM Mode” on page 122 for more details.

Table 17-4 shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 17-4. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal port operation, OC2A disconnected. WGM22 = 1: Toggle OC2A on compare match.
1	0	Clear OC2A on compare match when up-counting. Set OC2A on compare match when down-counting.
1	1	Set OC2A on compare match when up-counting. Clear OC2A on compare match when down-counting.

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 17.7.4 “Phase Correct PWM Mode” on page 123 for more details.

Bits 5:4 – COM2B1:0: Compare Match Output B Mode

These bits control the output compare pin (OC2B) behavior. If one or both of the COM2B1:0 bits are set, the OC2B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC2B pin must be set in order to enable the output driver.

When OC2B is connected to the pin, the function of the COM2B1:0 bits depends on the WGM22:0 bit setting. Table 17-5 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM).

Table 17-5. Compare Output Mode, non-PWM Mode

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Toggle OC2B on compare match
1	0	Clear OC2B on compare match
1	1	Set OC2B on compare match

Table 17-6 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to fast PWM mode.

Table 17-6. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on compare match, set OC2B at BOTTOM, (non-inverting mode).
1	1	Set OC2B on compare match, clear OC2B at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See Section 17.7.4 "Phase Correct PWM Mode" on page 123 for more details.

Table 17-7 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 17-7. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on compare match when up-counting. Set OC2B on compare match when down-counting.
1	1	Set OC2B on compare match when up-counting. Clear OC2B on compare match when down-counting.

Note: 1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 17.7.4 "Phase Correct PWM Mode" on page 123 for more details.

• Bits 3, 2 – Res: Reserved Bits

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

• Bits 1:0 – WGM21:0: Waveform Generation Mode

Combined with the WGM22 bit found in the TCCR2B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 17-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 17.7 "Modes of Operation" on page 120).

Table 17-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on ⁽¹⁾⁽²⁾
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

17.11.2 TCCR2B – Timer/Counter Control Register B

Bit (0xB1)	7	6	5	4	3	2	1	0
	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20
Read/Write	W	W	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

• Bit 7 – FOC2A: Force Output Compare A

The FOC2A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2A bit, an immediate compare match is forced on the waveform generation unit. The OC2A output is changed according to its COM2A1:0 bits setting. Note that the FOC2A bit is implemented as a strobe. Therefore it is the value present in the COM2A1:0 bits that determines the effect of the forced compare.

A FOC2A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2A as TOP.

The FOC2A bit is always read as zero.

• Bit 6 – FOC2B: Force Output Compare B

The FOC2B bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2B bit, an immediate compare match is forced on the waveform generation unit. The OC2B output is changed according to its COM2B1:0 bits setting. Note that the FOC2B bit is implemented as a strobe. Therefore it is the value present in the COM2B1:0 bits that determines the effect of the forced compare.

A FOC2B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2B as TOP.

The FOC2B bit is always read as zero.

• Bits 5:4 – Res: Reserved Bits

These bits are reserved bits in the ATmega328P and will always read as zero.

- **Bit 3 – WGM22: Waveform Generation Mode**

See the description in the [Section 17.11.1 “TCR2A – Timer/Counter Control Register A”](#) on page 127.

- **Bit 2:0 – CS22:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter, see [Table 17-9](#).

Table 17-9. Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$\text{clk}_{T2S}/(\text{no prescaling})$
0	1	0	$\text{clk}_{T2S}/8$ (from prescaler)
0	1	1	$\text{clk}_{T2S}/32$ (from prescaler)
1	0	0	$\text{clk}_{T2S}/64$ (from prescaler)
1	0	1	$\text{clk}_{T2S}/128$ (from prescaler)
1	1	0	$\text{clk}_{T2S}/256$ (from prescaler)
1	1	1	$\text{clk}_{T2S}/1024$ (from prescaler)

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

17.11.3 TCNT2 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
(0xB2)	TCNT2[7:0]								TCNT2
Read/Write	R/W	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 Timer/Counter register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT2 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a compare match between TCNT2 and the OCR2x registers.

17.11.4 OCR2A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
(0xB3)	OCR2A[7:0]								OCR2A
Read/Write	R/W	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 output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC2A pin.

17.11.5 OCR2B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
(0xB4)	OCR2B[7:0]								OCR2B
Read/Write	R/W	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 output compare register B contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC2B pin.

17.11.6 TIMSK2 – Timer/Counter2 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x70)	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2 – OCIE2B: Timer/Counter2 Output Compare Match B Interrupt Enable**

When the OCIE2B bit is written to one and the I-bit in the status register is set (one), the Timer/Counter2 compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCF2B bit is set in the Timer/Counter 2 interrupt flag register – TIFR2.

- **Bit 1 – OCIE2A: Timer/Counter2 Output Compare Match A Interrupt Enable**

When the OCIE2A bit is written to one and the I-bit in the status register is set (one), the Timer/Counter2 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCF2A bit is set in the Timer/Counter 2 interrupt flag register – TIFR2.

- **Bit 0 – TOIE2: Timer/Counter2 Overflow Interrupt Enable**

When the TOIE2 bit is written to one and the I-bit in the status register is set (one), the Timer/Counter2 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs, i.e., when the TOV2 bit is set in the Timer/Counter2 interrupt flag register – TIFR2.

17.11.7 TIFR2 – Timer/Counter2 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	–	–	–	–	–	OCF2B	OCF2A	TOV2	TIFR2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2 – OCF2B: Output Compare Flag 2 B**

The OCF2B bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2B – output compare register2. OCF2B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2B (Timer/Counter2 compare match interrupt enable), and OCF2B are set (one), the Timer/Counter2 compare match interrupt is executed.

- **Bit 1 – OCF2A: Output Compare Flag 2 A**

The OCF2A bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2A – output compare register2. OCF2A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2A (Timer/Counter2 compare match interrupt enable), and OCF2A are set (one), the Timer/Counter2 compare match interrupt is executed.

- **Bit 0 – TOV2: Timer/Counter2 Overflow Flag**

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE2A (Timer/Counter2 overflow interrupt enable), and TOV2 are set (one), the Timer/Counter2 overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 changes counting direction at 0x00.

17.11.8 ASSR – Asynchronous Status Register

Bit	7	6	5	4	3	2	1	0	
(0xB6)	–	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	ASSR
Read/Write	R	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RES: Reserved bit**

This bit is reserved and will always read as zero.

- **Bit 6 – EXCLK: Enable External Clock Input**

When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on timer oscillator 1 (TOSC1) pin instead of a 32kHz crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal oscillator will only run when this bit is zero.

- **Bit 5 – AS2: Asynchronous Timer/Counter2**

When AS2 is written to zero, Timer/Counter2 is clocked from the I/O clock, $clk_{I/O}$. When AS2 is written to one, Timer/Counter2 is clocked from a crystal oscillator connected to the timer oscillator 1 (TOSC1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B might be corrupted.

- **Bit 4 – TCN2UB: Timer/Counter2 Update Busy**

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

- **Bit 3 – OCR2AUB: Output Compare Register2 Update Busy**

When Timer/Counter2 operates asynchronously and OCR2A is written, this bit becomes set. When OCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2A is ready to be updated with a new value.

- **Bit 2 – OCR2BUB: Output Compare Register2 Update Busy**

When Timer/Counter2 operates asynchronously and OCR2B is written, this bit becomes set. When OCR2B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2B is ready to be updated with a new value.

- **Bit 1 – TCR2AUB: Timer/Counter Control Register2 Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2A is written, this bit becomes set. When TCCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2A is ready to be updated with a new value.

- **Bit 0 – TCR2BUB: Timer/Counter Control Register2 Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2B is written, this bit becomes set. When TCCR2B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2B is ready to be updated with a new value.

If a write is performed to any of the five Timer/Counter2 registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B are different. When reading TCNT2, the actual timer value is read. When reading OCR2A, OCR2B, TCCR2A and TCCR2B the value in the temporary storage register is read.

17.11.9 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	–	–	–	–	–	PSRASY	PSRSYNC	GTCCR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – PSRASY: Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set. Refer to the description of [“Bit 7 – TSM: Timer/Counter Synchronization Mode”](#) on page 115 for a description of the Timer/Counter synchronization mode.

19. USART0

19.1 Features

- Full duplex operation (independent serial receive and transmit registers)
- Asynchronous or synchronous operation
- Master or slave clocked synchronous operation
- High resolution 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 detection
- Framing error detection
- Noise filtering includes false start bit detection and digital low pass filter
- Three separate interrupts on TX complete, TX data register empty and RX complete
- Multi-processor communication mode
- Double speed asynchronous communication mode

19.2 Overview

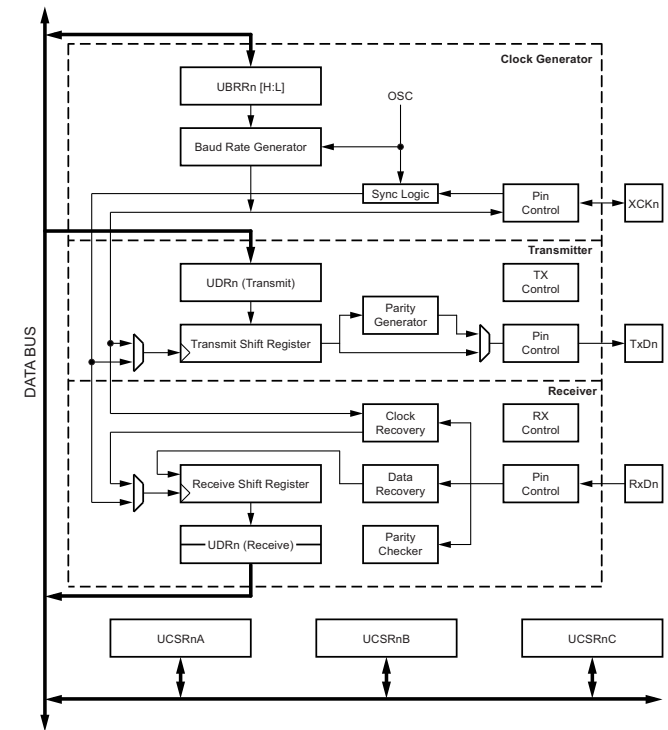
The universal synchronous and asynchronous serial receiver and transmitter (USART) is a highly flexible serial communication device.

The USART0 can also be used in master SPI mode, see [Section 20. "USART in SPI Mode" on page 166](#). The power reduction USART bit, PRUSART0, in [Section 9.10 "Minimizing Power Consumption" on page 36](#) must be disabled by writing a logical zero to it.

A simplified block diagram of the USART transmitter is shown in [Figure 19-1 on page 144](#). CPU accessible I/O registers and I/O pins are shown in bold.

The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock generator, transmitter and receiver. Control registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCKn (transfer clock) pin is only used by synchronous transfer mode. The transmitter consists of a single write buffer, a serial shift register, parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a shift register and a two level receive buffer (UDRn). The receiver supports the same frame formats as the transmitter, and can detect frame error, data overrun and parity errors.

Figure 19-1. USART Block Diagram⁽¹⁾



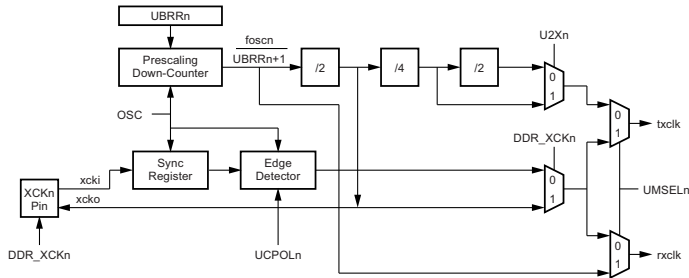
Note: 1. Refer to [Figure 1-1 on page 3](#) and [Table 13-9 on page 70](#) for USART0 pin placement.

19.3 Clock Generation

The clock generation logic generates the base clock for the transmitter and receiver. The USART supports four modes of clock operation: Normal asynchronous, double speed asynchronous, master synchronous and slave synchronous mode. The UMSELn bit in USART control and status register C (UCSRnC) selects between asynchronous and synchronous operation. Double speed (asynchronous mode only) is controlled by the U2Xn found in the UCSRnA register. When using synchronous mode (UMSELn = 1), the data direction register for the XCKn pin (DDR_XCKn) controls whether the clock source is internal (master mode) or external (slave mode). The XCKn pin is only active when using synchronous mode.

Figure 19-2 shows a block diagram of the clock generation logic.

Figure 19-2. Clock Generation Logic, Block Diagram



Signal description:

- txclk** Transmitter clock (Internal Signal).
- rxclk** Receiver base clock (Internal Signal).
- xcki** Input from XCK pin (internal Signal). Used for synchronous slave operation.
- xcko** Clock output to XCK pin (Internal Signal). Used for synchronous master operation.
- fosc** XTAL pin frequency (System Clock).

19.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 19-2.

The USART baud rate register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRnL register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output ($= f_{osc}/(UBRRn+1)$). The transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSELn, U2Xn and DDR_XCKn bits.

Table 19-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 19-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate ⁽¹⁾	Equation for Calculating UBRRn Value
Asynchronous normal mode (U2Xn = 0)	$BAUD = \frac{f_{osc}}{16(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{16BAUD} - 1$
Asynchronous double speed mode (U2Xn = 1)	$BAUD = \frac{f_{osc}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{8BAUD} - 1$
Synchronous master mode	$BAUD = \frac{f_{osc}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

- BAUD** Baud rate (in bits per second, bps)
- f_{osc}** System oscillator clock frequency
- UBRRn** Contents of the UBRRnH and UBRRnL registers, (0-4095)

Some examples of UBRRn values for some system clock frequencies are found in Table 19-9 on page 163.

19.3.2 Double Speed Operation (U2Xn)

The transfer rate can be doubled by setting the U2Xn bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the transmitter, there are no downsides.

19.3.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 19-2 on page 145 for details.

External clock input from the XCKn pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCKn clock frequency is limited by the following equation:

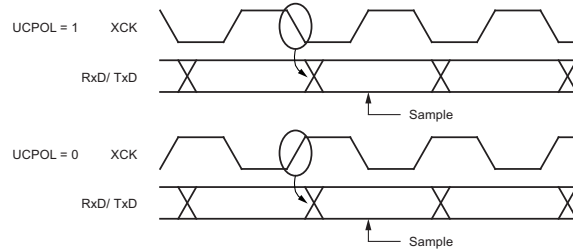
$$f_{XCK} < \frac{f_{osc}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

19.3.4 Synchronous Clock Operation

When synchronous mode is used (UMSELn = 1), the XCKn pin will be used as either clock input (Slave) or clock output (master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxDn) is sampled at the opposite XCKn clock edge of the edge the data output (TxDn) is changed.

Figure 19-3. Synchronous Mode XCKn Timing



The UCPOLn bit UCRSC selects which XCKn clock edge is used for data sampling and which is used for data change. As Figure 19-3 shows, when UCPOLn is zero the data will be changed at rising XCKn edge and sampled at falling XCKn edge. If UCPOLn is set, the data will be changed at falling XCKn edge and sampled at rising XCKn edge.

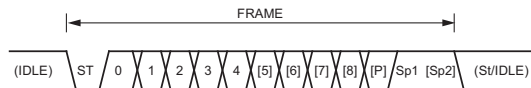
19.4 Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. 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. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 19-4 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 19-4. Frame Formats



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 (Rx/Dn or Tx/Dn). An IDLE line must be high.

The frame format used by the USART is set by the UCSZn2:0, UPMn1:0 and USBn bits in UCSRnB and UCSRnC. The receiver and transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the receiver and transmitter.

The USART character size (UCSZn2:0) bits select the number of data bits in the frame. The USART parity mode (UPMn1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART stop bit select (USBn) bit. The receiver ignores the second stop bit. An FE (frame error) will therefore only be detected in the cases where the first stop bit is zero.

19.4.1 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even}	Parity bit using even parity
P_{odd}	Parity bit using odd parity
d_n	Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

19.5 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the transmitter or the receiver depending on the usage. For interrupt driven USART operation, the global interrupt flag should be cleared (and interrupts globally disabled) when doing 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 TXCn flag can be used to check that the transmitter has completed all transfers, and the RXC flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example ⁽¹⁾
<pre> USART_Init: ; Set baud rate out UBRRnH, r17 out UBRRnL, r16 ; Enable receiver and transmitter ldi r16, (1<<RXENn) (1<<TXENn) out UCSRnB,r16 ; Set frame format: 8data, 2stop bit ldi r16, (1<<USBSn) (3<<UCSZn0) out UCSRnC,r16 ret </pre>
C Code Example ⁽¹⁾
<pre> #define FOSC 1843200 // Clock Speed #define BAUD 9600 #define MYUBRR FOSC/16/BAUD-1 void main(void) { ... USART_Init(MYUBRR) ... } void USART_Init(unsigned int ubrr) { /*Set baud rate */ UBRR0H = (unsigned char)(ubrr>>8); UBRR0L = (unsigned char)ubrr; Enable receiver and transmitter */ UCSR0B = (1<<RXEN0) (1<<TXEN0); /* Set frame format: 8data, 2stop bit */ UCSR0C = (1<<USBS0) (3<<UCSZ00); } </pre>

Note: 1. See [Section 5. "About Code Examples" on page 8](#).

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

19.6 Data Transmission – The USART Transmitter

The USART transmitter is enabled by setting the transmit enable (TXEN) bit in the UCSRnB register. When the transmitter is enabled, the normal port operation of the TxDn pin is overridden by the USART and given the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCKn pin will be overridden and used as transmission clock.

19.6.1 Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDRn I/O location. The buffered data in the transmit buffer will be moved to the shift register when the shift register is ready to send a new frame. The shift register is loaded with new data 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 new data, it will transfer one complete frame at the rate given by the baud register, U2Xn bit or by XCKn depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the data register empty (UDREN) flag. When using frames with less than eight bits, the most significant bits written to the UDRn are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register R16.

Assembly Code Example ⁽¹⁾
<pre> USART_Transmit: ; Wait for empty transmit buffer sbis UCSRnA,UDREN rjmp USART_Transmit ; Put data (r16) into buffer, sends the data out UDRn,r16 ret </pre>
C Code Example ⁽¹⁾
<pre> void USART_Transmit(unsigned char data) { /* Wait for empty transmit buffer */ while (!(UCSRnA & (1<<UDREN))) ; /* Put data into buffer, sends the data */ UDRn = data; } </pre>

Note: 1. See [Section 5. "About Code Examples" on page 8](#)

The function simply waits for the transmit buffer to be empty by checking the UDREN flag, before loading it with new data to be transmitted. If the data register empty interrupt is utilized, the interrupt routine writes the data into the buffer.

19.6.2 Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZn = 7), the ninth bit must be written to the TXB8 bit in UCSRnB before the low byte of the character is written to UDRn. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

Assembly Code Example ⁽¹⁾⁽²⁾
<pre>USART_Transmit: ; Wait for empty transmit buffer sbis UCSRnA,UDREN rjmp USART_Transmit ; Copy 9th bit from r17 to TXB8 cbi UCSRnB,TXB8 sbrc r17,0 sbi UCSRnB,TXB8 ; Put LSB data (r16) into buffer, sends the data out UDRn,r16 ret</pre>
C Code Example ⁽¹⁾⁽²⁾
<pre>void USART_Transmit(unsigned int data) { /* Wait for empty transmit buffer */ while (!(UCSRnA & (1<<UDREN))) ; /* Copy 9th bit to TXB8 */ UCSRnB &= ~(1<<TXB8); if (data & 0x0100) UCSRnB = (1<<TXB8); /* Put data into buffer, sends the data */ UDRn = data; }</pre>

- Notes:
1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB Register is used after initialization.
 2. See [Section 5. "About Code Examples" on page 8](#).

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

19.6.3 Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART data register empty (UDREN) and transmit complete (TXCn). Both flags can be used for generating interrupts.

The data register empty (UDREN) flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. For compatibility with future devices, always write this bit to zero when writing the UCSRnA register.

When the data register empty interrupt enable (UDRIEn) bit in UCSRnB is written to one, the USART data register empty interrupt will be executed as long as UDREN is set (provided that global interrupts are enabled). UDREN is cleared by writing UDRn. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to UDRn in order to clear UDREN or disable the data register empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The transmit complete (TXCn) flag bit is set one when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXCn flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the transmit complete interrupt enable (TXCIEn) bit in UCSRnB is set, the USART transmit complete interrupt will be executed when the TXCn flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXCn flag, this is done automatically when the interrupt is executed.

19.6.4 Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPMn1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

19.6.5 Disabling the Transmitter

The disabling of the transmitter (setting the 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 TxDn pin.

19.7 Data Reception – The USART Receiver

The USART receiver is enabled by writing the receive enable (RXENn) bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCKn pin will be used as transfer clock.

19.7.1 Receiving Frames with 5 to 8 Data Bits

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 XCKn 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, i.e., 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 buffer can then be read by reading the UDRn I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXCn) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDRn will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example ⁽¹⁾
<pre>USART_Receive: ; Wait for data to be received sbis UCSRnA, RXCn rjmp USART_Receive ; Get and return received data from buffer in r16, UDRn ret</pre>
C Code Example ⁽¹⁾
<pre>unsigned char USART_Receive(void) { /* Wait for data to be received */ while (!(UCSRnA & (1<<RXCN))) ; /* Get and return received data from buffer */ return UDRn; }</pre>

- Note:
1. See [Section 5. "About Code Examples" on page 8](#).
For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBR", "SBR", and "CBR".

The function simply waits for data to be present in the receive buffer by checking the RXCn flag, before reading the buffer and returning the value.

19.7.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZn=7) the ninth bit must be read from the RXB8n bit in UCSRnB **before** reading the low bits from the UDRn. This rule applies to the FEn, DORn and UPEn status flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8n, FEn, DORn and UPEn bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example ⁽¹⁾
<pre>USART_Receive: ; Wait for data to be received sbis UCSRnA, RXCn rjmp USART_Receive ; Get status and 9th bit, then data from buffer in r18, UCSRnA in r17, UCSRnB in r16, UDRn ; If error, return -1 andi r18, (1<<FEn) (1<<DORn) (1<<UPEn) breq USART_ReceiveNoError ldi r17, HIGH(-1) ldi r16, LOW(-1) USART_ReceiveNoError: ; Filter the 9th bit, then return lsr r17 andi r17, 0x01 ret</pre>
C Code Example ⁽¹⁾
<pre>unsigned int USART_Receive(void) { unsigned char status, resh, resl; /* Wait for data to be received */ while (!(UCSRnA & (1<<RXCn))) ; /* Get status and 9th bit, then data */ /* from buffer */ status = UCSRnA; resh = UCSRnB; resl = UDRn; /* If error, return -1 */ if (status & (1<<FEn) (1<<DORn) (1<<UPEn)) return -1; /* Filter the 9th bit, then return */ resh = (resh >> 1) & 0x01; return ((resh << 8) resl); }</pre>

Note: 1. See [Section 5, "About Code Examples" on page 8](#)
For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBRC", "SBR", and "CBR".

The receive function example reads all the I/O registers into the register file before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

19.7.3 Receive Complete Flag and Interrupt

The USART receiver has one flag that indicates the receiver state.

The receive complete (RXCn) flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled (RXENn = 0), the receive buffer will be flushed and consequently the RXCn bit will become zero.

When the receive complete interrupt enable (RXCIEn) in UCSRnB is set, the USART receive complete interrupt will be executed as long as the RXCn flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDRn in order to clear the RXCn flag, otherwise a new interrupt will occur once the interrupt routine terminates.

19.7.4 Receiver Error Flags

The USART receiver has three error flags: Frame error (FEn), data overrun (DORn) and parity error (UPEn). All can be accessed by reading UCSRnA. Common for the error flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the error flags, the UCSRnA must be read before the receive buffer (UDRn), since reading the UDRn I/O location changes the buffer read location. Another equality for the error flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRnA is written for upward compatibility of future USART implementations. None of the error flags can generate interrupts.

The frame error (FEn) flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FEn flag is zero when the stop bit was correctly read (as one), and the FEn flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FEn flag is not affected by the setting of the USBn bit in UCSRnC since the receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRnA.

The data overrun (DORn) flag indicates data loss due to a receiver buffer full condition. A data overrun 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. If the DORn flag is set there was one or more serial frame lost between the frame last read from UDRn, and the next frame read from UDRn. For compatibility with future devices, always write this bit to zero when writing to UCSRnA. The DORn flag is cleared when the frame received was successfully moved from the shift register to the receive buffer.

The parity error (UPEn) flag indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the UPEn bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRnA. For more details see [Section 19.4.1 "Parity Bit Calculation" on page 148](#) and [Section 19.7.5 "Parity Checker" on page 154](#).

19.7.5 Parity Checker

The parity checker is active when the high USART parity mode (UPMn1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPMn0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The parity error (UPEn) flag can then be read by software to check if the frame had a parity error.

The UPEn bit is set if the next character that can be read from the receive buffer had a parity error when received and the parity checking was enabled at that point (UPMn1 = 1). This bit is valid until the receive buffer (UDRn) is read.

19.7.6 Disabling the Receiver

In contrast to the transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., the RXENn is set to zero) the receiver will no longer override the normal function of the RxDn port pin. The receiver buffer FIFO will be flushed when the receiver is disabled. Remaining data in the buffer will be lost.

19.7.7 Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDRn I/O location until the RXCn flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example ⁽¹⁾
<pre> USART_Flush: sbis UCSRnA, RXCn ret in r16, UDRn rjmp USART_Flush </pre>
C Code Example ⁽¹⁾
<pre> void USART_Flush(void) { unsigned char dummy; while (UCSRnA & (1<<RXCn)) dummy = UDRn; } </pre>

Note: 1. See Section 5, “About Code Examples” on page 8.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBR”, “SBR”, “SBR”, and “CBR”.

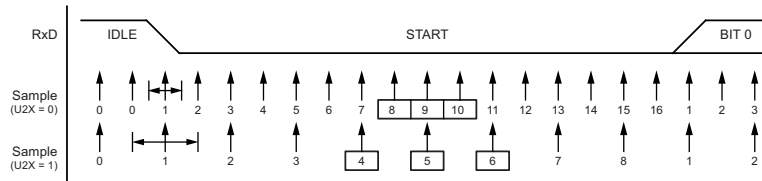
19.8 Asynchronous Data Reception

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 internally generated baud rate clock to the incoming asynchronous serial frames at the RxDn pin. 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.

19.8.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 19-5 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 (U2Xn = 1) of operation. Samples denoted zero are samples done when the RxDn line is idle (i.e., no communication activity).

Figure 19-5. Start Bit Sampling

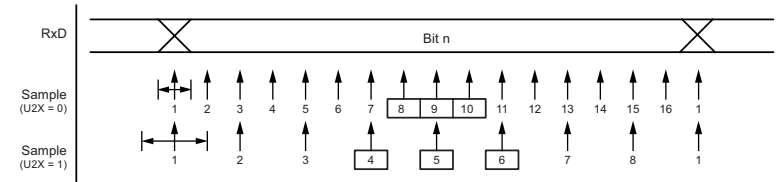


When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for normal mode, and samples 4, 5, and 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 logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

19.8.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and eight states for each bit in double speed mode. Figure 19-6 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

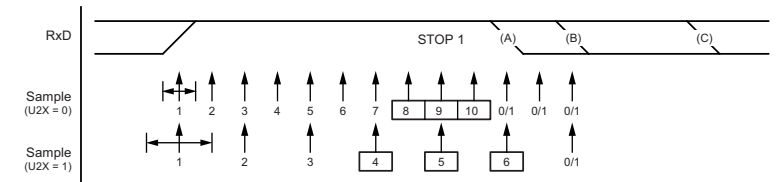
Figure 19-6. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxDn pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 19-7 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 19-7. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the frame error (FEn) flag will be set.

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 Figure 19-7. For double speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

19.8.3 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 the transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have a similar (see [Table 19-2](#)) base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{\text{slow}} = \frac{(D+1)S}{S-1+D \times S + S_F} \quad R_{\text{fast}} = \frac{(D+2)S}{(D+1)S + S_M}$$

- D** Sum of character size and parity size (D = 5 to 10 bit)
- S** Samples per bit. S = 16 for normal speed mode and S = 8 for double speed mode.
- S_F** First sample number used for majority voting. 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 19-2](#) and [Table 19-3](#) list the maximum receiver baud rate error that can be tolerated. Note that normal speed mode has higher toleration of baud rate variations.

Table 19-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2Xn = 0)

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	±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 19-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2Xn = 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
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 (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRRn value that gives an acceptable low error can be used if possible.

19.9 Multi-processor Communication Mode

Setting the multi-processor communication mode (MPCMn) bit in UCSRnA enables a filtering function of incoming frames received by the USART receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The transmitter is unaffected by the MPCMn setting, but has to be used differently when it is a part of a system utilizing the multi-processor communication mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the receiver is set up for frames with nine data bits, then the ninth bit (RXB8n) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The multi-processor communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. 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.

19.9.1 Using MPCMn

For an MCU to act as a master MCU, it can use a 9-bit character frame format (UCSZn = 7). The ninth bit (TXB8n) must be set when an address frame (TXB8n = 1) or cleared when a data frame (TXB = 0) 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 (MPCMn in UCSRnA is set).
2. The master MCU sends an address frame, and all slaves receive and read this frame. In the Slave MCUs, the RXCn flag in UCSRnA will be set as normal.
3. Each slave MCU reads the UDRn register and determines if it has been selected. If so, it clears the MPCMn bit in UCSRnA, otherwise it waits for the next address byte and keeps the MPCMn setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other slave MCUs, which still have the MPCMn bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCMn bit and waits for a new address frame from 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. If 5- to 8-bit character frames are used, the transmitter must be set to use two stop bit (USBSn = 1) since the first stop bit is used for indicating the frame type.

Do not use read-modify-write instructions (SBI and CBI) to set or clear the MPCMn bit. The MPCMn bit shares the same I/O location as the TXCn flag and this might accidentally be cleared when using SBI or CBI instructions.

19.10 Register Description

19.10.1 UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	R/W	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 or UDRn. The transmit data buffer register (TXB) will be the destination for data written to the UDRn register location. Reading the UDRn 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 UDREN flag in the UCSRnA register is set. Data written to UDRn when the UDREN 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. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read-modify-write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

19.10.2 UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

• Bit 7 – RXCn: USART Receive Complete

This flag bit 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). If the receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn flag can be used to generate a receive complete interrupt (see description of the RXCIE bit).

• Bit 6 – TXCn: USART Transmit Complete

This flag bit is set when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer (UDRn). The TXCn flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn flag can generate a transmit complete interrupt (see description of the TXCIE bit).

• Bit 5 – UDREN: USART Data Register Empty

The UDREN flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREN is one, the buffer is empty, and therefore ready to be written. The UDREN flag can generate a data register empty interrupt (see description of the UDRIE bit). UDREN is set after a reset to indicate that the transmitter is ready.

• Bit 4 – FEn: Frame Error

This bit is set if the next character in the receive buffer had a frame error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDRn) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRnA.

• Bit 3 – DORn: Data OverRun

This bit is set if a data overrun condition is detected. A data overrun 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 bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

• Bit 2 – UPEn: USART Parity Error

This bit is set if the next character in the receive buffer had a parity error when received and the parity checking was enabled at that point (UPMn1 = 1). This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

• Bit 1 – U2Xn: Double the USART Transmission Speed

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

• Bit 0 – MPCMn: Multi-processor Communication Mode

This bit enables the multi-processor communication mode. When the MPCMn bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCMn setting. For more detailed information see [Section 19.9 "Multi-processor Communication Mode" on page 158](#).

19.10.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – RXCIE: RX Complete Interrupt Enable n

Writing this bit to one enables interrupt on the RXCn flag. A USART receive complete interrupt will be generated only if the RXCIE bit is written to one, the global interrupt flag in SREG is written to one and the RXCn bit in UCSRnA is set.

• Bit 6 – TXCIE: TX Complete Interrupt Enable n

Writing this bit to one enables interrupt on the TXCn flag. A USART transmit complete interrupt will be generated only if the TXCIE bit is written to one, the global interrupt flag in SREG is written to one and the TXCn bit in UCSRnA is set.

• Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable n

Writing this bit to one enables interrupt on the UDREN flag. A data register empty interrupt will be generated only if the UDRIE bit is written to one, the global interrupt flag in SREG is written to one and the UDREN bit in UCSRnA is set.

• Bit 4 – RXEN: Receiver Enable n

Writing this bit to one enables the USART receiver. The receiver will override normal port operation for the RxDn pin when enabled. Disabling the receiver will flush the receive buffer invalidating the FEn, DORn, and UPEn flags.

• Bit 3 – TXEN: Transmitter Enable n

Writing this bit to one enables the USART transmitter. The transmitter will override normal port operation for the TxDn pin when enabled. The disabling of 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 TxDn port.

• Bit 2 – UCSZn2: Character Size n

The UCSZn2 bits combined with the UCSZn1:0 bit in UCSRnC sets the number of data bits (character size) in a frame the receiver and transmitter use.

• Bit 1 – RXB8n: Receive Data Bit 8 n

RXB8n is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDRn.

• Bit 0 – TXB8n: Transmit Data Bit 8 n

TXB8n is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDRn.

19.10.4 UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/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	

• Bits 7:6 – UMSELn1:0 USART Mode Select

These bits select the mode of operation of the USARTn as shown in Table 19-4.

Table 19-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM) ⁽¹⁾

Note: 1. See Section 20. “USART in SPI Mode” on page 166 for full description of the master SPI mode (MSPIM) operation

• Bits 5:4 – UPMn1:0: Parity Mode

These bits enable and set type of parity generation and check. If 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 UPMn setting. If a mismatch is detected, the UPEn flag in UCSRnA will be set.

Table 19-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, even parity
1	1	Enabled, odd parity

• Bit 3 – USBSn: Stop Bit Select

This bit selects the number of stop bits to be inserted by the transmitter. The receiver ignores this setting.

Table 19-6. USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

• Bit 2:1 – UCSZn1:0: Character Size

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (character size) in a frame the receiver and transmitter use.

Table 19-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

• Bit 0 – UCPOLn: Clock Polarity

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 19-8. UCPOLn Bit Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn edge	Falling XCKn edge
1	Falling XCKn edge	Rising XCKn edge

19.10.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	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 15:12 – Reserved Bits

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRnH is written.

• Bit 11:0 – UBRR11:0: USART Baud Rate Register

This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits, and the UBRRnL 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 UBRRnL will trigger an immediate update of the baud rate prescaler.

19.11 Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRRn settings in [Table 19-9](#). UBRRn values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see [Section 19.8.3 "Asynchronous Operational Range" on page 157](#)). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1 \right) \times 100\%$$

Table 19-9. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	$f_{\text{osc}} = 1.0000\text{MHz}$				$f_{\text{osc}} = 1.8432\text{MHz}$				$f_{\text{osc}} = 2.0000\text{MHz}$			
	$U2Xn = 0$		$U2Xn = 1$		$U2Xn = 0$		$U2Xn = 1$		$U2Xn = 0$		$U2Xn = 1$	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	—	—	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	—	—	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	—	—	—	—	—	—	0	0.0%	—	—	—	—
250k	—	—	—	—	—	—	—	—	—	—	0	0.0%
Max. ⁽¹⁾	62.5kbps		125kbps		115.2kbps		230.4kbps		125kbps		250kbps	

Note: 1. UBRRn = 0, error = 0.0%

Table 19-10. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	$f_{\text{osc}} = 3.6864\text{MHz}$				$f_{\text{osc}} = 4.0000\text{MHz}$				$f_{\text{osc}} = 7.3728\text{MHz}$			
	$U2Xn = 0$		$U2Xn = 1$		$U2Xn = 0$		$U2Xn = 1$		$U2Xn = 0$		$U2Xn = 1$	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	—	—	0	-7.8%	—	—	0	0.0%	0	-7.8%	1	-7.8%
1M	—	—	—	—	—	—	—	—	—	—	0	-7.8%
Max. ⁽¹⁾	230.4kbps		460.8kbps		250kbps		0.5Mbps		460.8kbps		921.6kbps	

Note: 1. UBRRn = 0, error = 0.0%

Table 19-11. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	$f_{\text{osc}} = 8.0000\text{MHz}$				$f_{\text{osc}} = 11.0592\text{MHz}$				$f_{\text{osc}} = 14.7456\text{MHz}$			
	$U2Xn = 0$		$U2Xn = 1$		$U2Xn = 0$		$U2Xn = 1$		$U2Xn = 0$		$U2Xn = 1$	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	—	—	2	-7.8%	1	-7.8%	3	-7.8%
1M	—	—	0	0.0%	—	—	—	—	0	-7.8%	1	-7.8%
Max. ⁽¹⁾	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

Note: 1. UBRRn = 0, error = 0.0%

Table 19-12. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	$f_{osc} = 16.0000\text{MHz}$			
	U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. ⁽¹⁾	1Mbps		2Mbps	

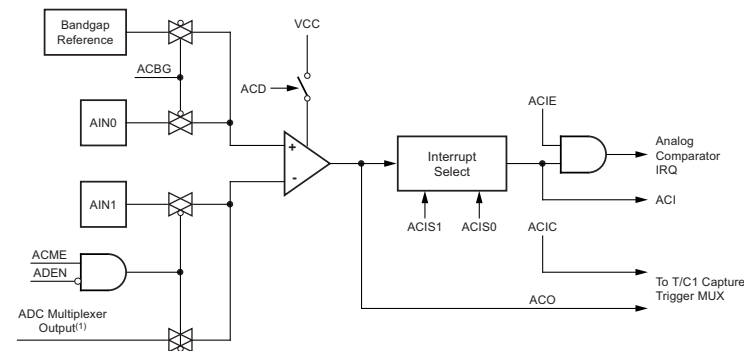
Note: 1. UBRRn = 0, error = 0.0%

22. Analog Comparator

22.1 Overview

The analog comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the analog comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 input capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the analog comparator. The user can select interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 22-1.

The power reduction ADC bit, PRADC, in Section 9.10 "Minimizing Power Consumption" on page 36 must be disabled by writing a logical zero to be able to use the ADC input MUX.

Figure 22-1. Analog Comparator Block Diagram⁽²⁾

Notes: 1. See Table 22-1.

2. Refer to Figure 1-1 on page 3 and Table 13-9 on page 70 for analog comparator pin placement.

22.2 Analog Comparator Multiplexed Input

It is possible to select any of the ADC7..0 pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 22-1. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the analog comparator.

Table 22-1. Analog Comparator Multiplexed Input

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

22.3 Register Description

22.3.1 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see [Section 22.2 "Analog Comparator Multiplexed Input" on page 202](#).

22.3.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. This will reduce power consumption in active and idle mode. When changing the ACD bit, the analog comparator interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the analog comparator. When this bit is cleared, AIN0 is applied to the positive input of the analog comparator. When the bandgap reference is used as input to the analog comparator, it will take a certain time for the voltage to stabilize. If not stabilized, the first conversion may give a wrong value. See [Section 10.7 "Internal Voltage Reference" on page 43](#)

- Bit 5 – ACO: Analog Comparator Output**

The output of the analog comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

- Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The analog comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the status register is set, the analog comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 input capture interrupt. When written logic zero, no connection between the analog comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 input capture interrupt, the ICIE1 bit in the timer interrupt mask register (TIMSK1) must be set.

- Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the analog comparator interrupt. The different settings are shown in [Table 22-2](#).

Table 22-2. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator interrupt on output toggle.
0	1	Reserved
1	0	Comparator interrupt on falling output edge.
1	1	Comparator interrupt on rising output edge.

When changing the ACIS1/ACIS0 bits, the analog comparator interrupt must be disabled by clearing its interrupt enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

22.3.3 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
(0x7F)	–	–	–	–	–	–	AIN1D	AIN0D	DIDR1
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 – Res: Reserved Bits**

These bits are unused bits in the ATmega328P, and will always read as zero.

- Bit 1, 0 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

23. Analog-to-Digital Converter

23.1 Features

- 10-bit resolution
- 0.5 LSB integral non-linearity
- ± 2 LSB absolute accuracy
- 65 to 260 μ s conversion time
- Up to 15kSPS
- 6 multiplexed single ended input channels
- 2 additional multiplexed single ended input channels
- Temperature sensor input channel
- Optional left adjustment for ADC result readout
- 0 to V_{CC} ADC input voltage range
- Selectable 1.1V ADC reference voltage
- Free running or single conversion mode
- Interrupt on ADC conversion complete
- Sleep mode noise canceler

23.2 Overview

The Atmel® ATmega328P features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel analog multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

The ADC contains a sample and hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in [Figure 23-1 on page 206](#).

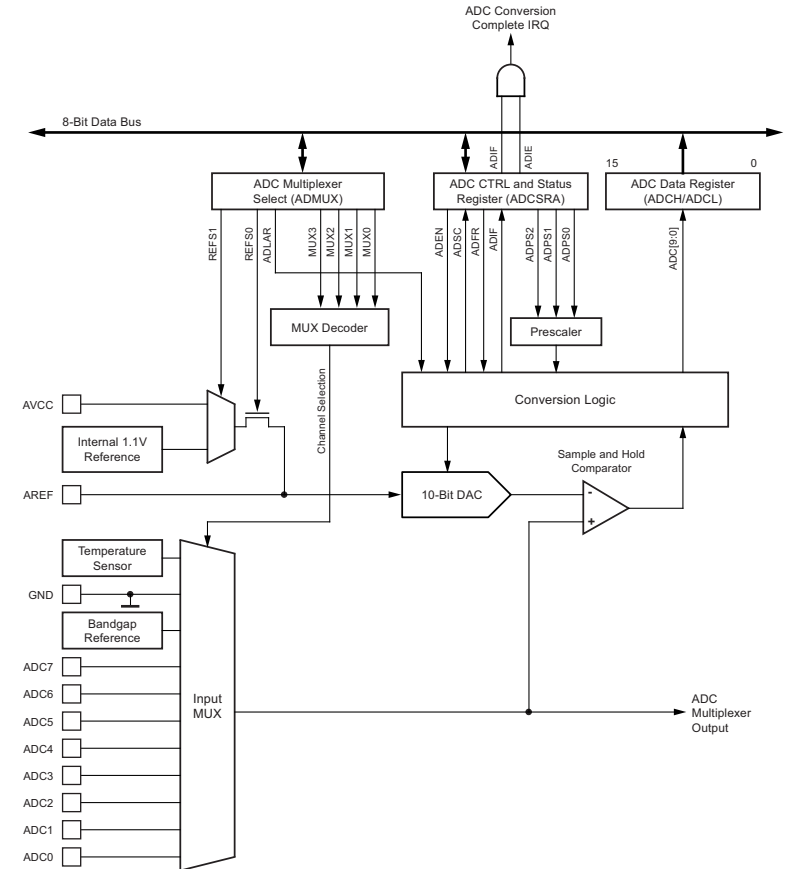
The ADC has a separate analog supply voltage pin, AV_{CC} . AV_{CC} must not differ more than $\pm 0.3V$ from V_{CC} . See [Section 23.6 "ADC Noise Canceler" on page 211](#) on how to connect this pin.

Internal reference voltages of nominally 1.1V or AV_{CC} are provided on-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

The power reduction ADC bit, PRADC, in [Section 9.10 "Minimizing Power Consumption" on page 36](#) must be disabled by writing a logical zero to enable the ADC.

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AV_{CC} or an internal 1.1V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

Figure 23-1. Analog to Digital Converter Block Schematic Operation



The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC data registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the data registers belongs to the same conversion. Once ADCL is read, ADC access to data registers is blocked.

This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL registers is re-enabled.

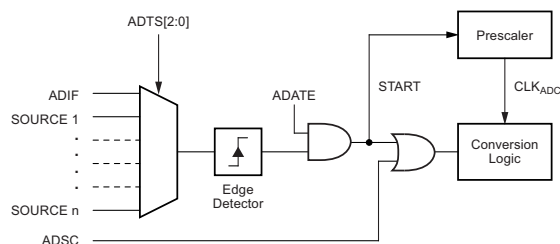
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

23.3 Starting a Conversion

A single conversion is started by disabling the power reduction ADC bit, PRADC, in [Section 9.10 "Minimizing Power Consumption" on page 36](#) by writing a logical zero to it and writing a logical one to the ADC start conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto triggering is enabled by setting the ADC auto trigger enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC trigger select bits, ADTS in ADCSRB (See description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the interrupt flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 23-2. ADC Auto Trigger Logic

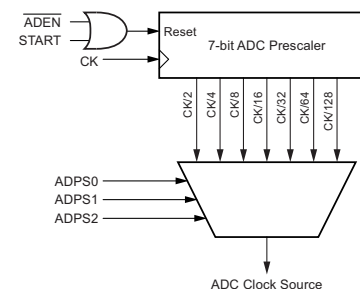


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC interrupt flag, ADIF is cleared or not.

If auto triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

23.4 Prescaling and Conversion Timing

Figure 23-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

When the bandgap reference voltage is used as input to the ADC, it will take a certain time for the voltage to stabilize. If not stabilized, the first value read after the first conversion may be wrong.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC data registers, and ADIF is set. In single conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When auto triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In free running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see [Table 23-1 on page 210](#).

Figure 23-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

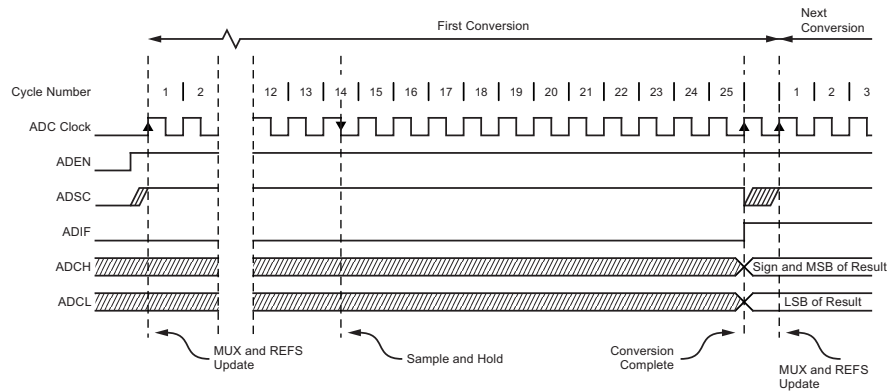


Figure 23-5. ADC Timing Diagram, Single Conversion

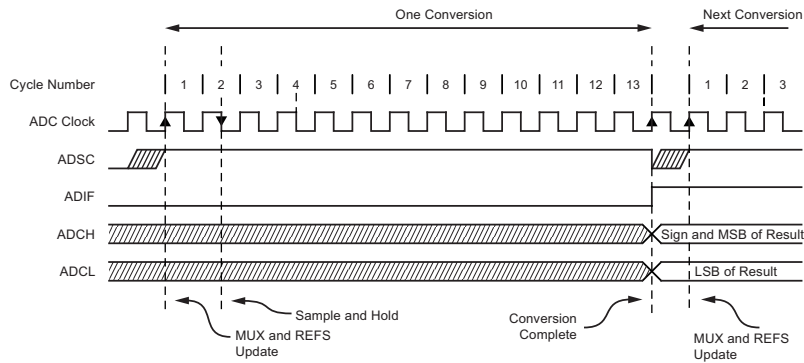


Figure 23-6. ADC Timing Diagram, Auto Triggered Conversion

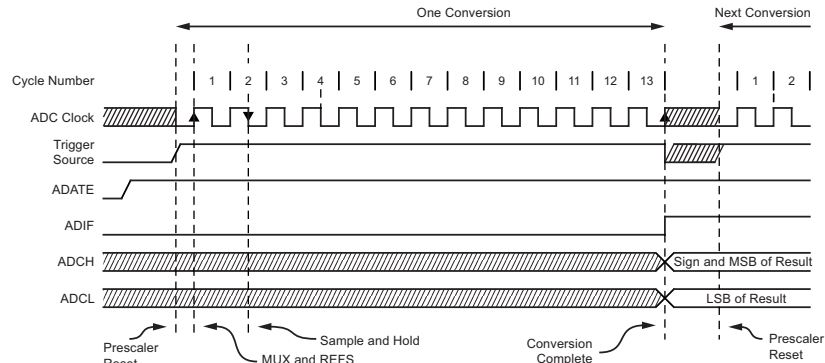


Figure 23-7. ADC Timing Diagram, Free Running Conversion

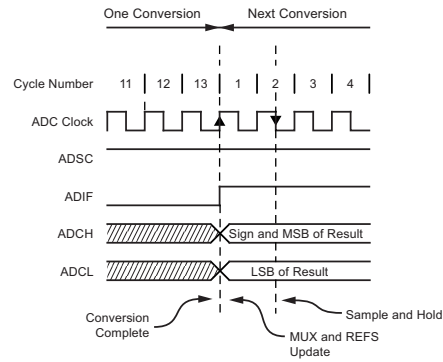


Table 23-1. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5

23.5 Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the ADMUX register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If auto triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

- When ADATE or ADEN is cleared.
- During conversion, minimum one ADC clock cycle after the trigger event.
- After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

23.5.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In single conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In free running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

23.5.2 ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AV_{CC} , internal 1.1V reference, or external AREF pin.

AV_{CC} is connected to the ADC through a passive switch. The internal 1.1V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedance voltmeter. Note that V_{REF} is a high impedance source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AV_{CC} and 1.1V as reference selection. The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

23.6 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC noise reduction and idle mode. To make use of this feature, the following procedure should be used:

- Make sure that the ADC is enabled and is not busy converting. Single Conversion mode must be selected and the ADC conversion complete interrupt must be enabled.
- Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.

- If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than idle mode and ADC noise reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption.

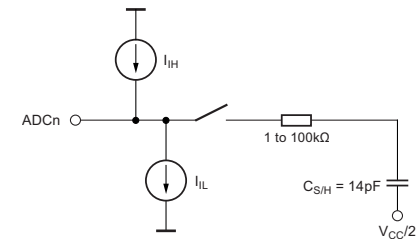
23.6.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in [Figure 23-8 on page 212](#). An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10k Ω or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 23-8. Analog Input Circuitry

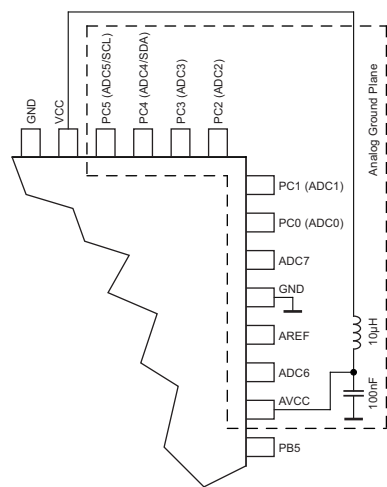


23.6.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

- Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
- The AV_{CC} pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in [Figure 23-9 on page 213](#).
- Use the ADC noise canceler function to reduce induced noise from the CPU.
- If any ADC [3..0] port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress. However, using the 2-wire interface (ADC4 and ADC5) will only affect the conversion on ADC4 and ADC5 and not the other ADC channels.

Figure 23-9. ADC Power Connections



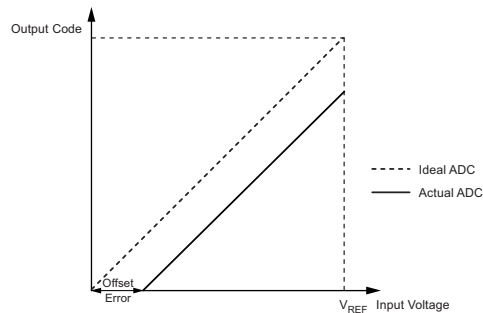
23.6.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n-1 .

Several parameters describe the deviation from the ideal behavior:

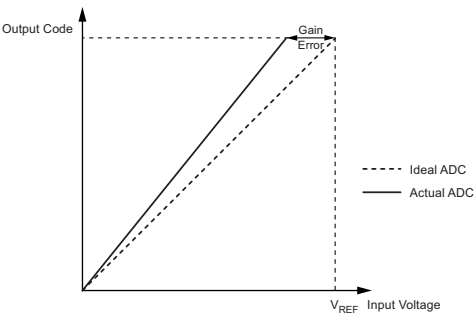
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 23-10. Offset Error



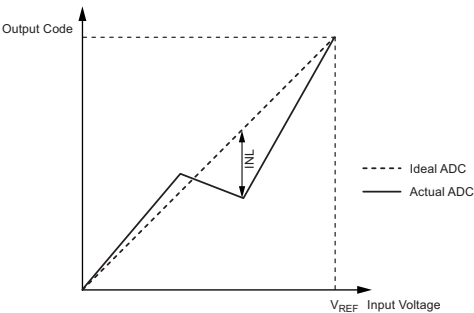
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 23-11. Gain Error



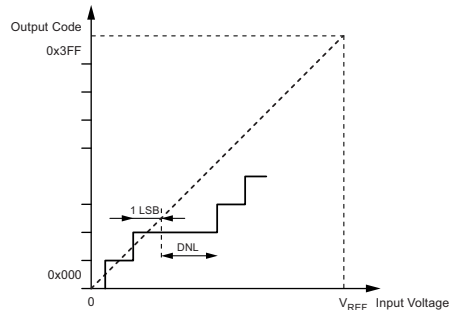
- Integral non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 23-12. Integral Non-linearity (INL)



- Differential non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 23-13. Differential Non-linearity (DNL)



- Quantization error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ± 0.5 LSB.
- Absolute accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ± 0.5 LSB.

23.7 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result registers (ADCL, ADCH).

For single ended conversion, the result is:

$$ADC = \frac{V_{IN} \times 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 23-3 on page 217 and Table 23-4 on page 218). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

23.8 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC input. MUX[4..0] bits in ADMUX register enables the temperature sensor. The internal 1.1V voltage reference must also be selected for the ADC voltage reference source in the temperature sensor measurement. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor.

The measured voltage has a linear relationship to the temperature as described in Table 23-2 on page 215.

The voltage sensitivity is approximately 1LSB/ $^{\circ}$ C and the accuracy of the temperature measurement is $\pm 10^{\circ}$ C using manufacturing calibration values (TS_GAIN, TS_OFFSET).

The values described in Table 23-2 are typical values. However, due to the process variation the temperature sensor output varies from one chip to another.

Table 23-2. Sensor Output Code versus Temperature (Typical Values)

Temperature/ $^{\circ}$ C	-40 $^{\circ}$ C	+25 $^{\circ}$ C	+125 $^{\circ}$ C
	0x010D	0x0160	0x01E0

23.8.1 Manufacturing Calibration

Calibration values determined during test are available in the signature row.

The temperature in degrees celsius can be calculated using the formula:

$$\frac{(((ADCH \ll 8) + ADCL) - (273 + 100 - TS_OFFSET)) \times 128}{TS_GAIN} + 25$$

Where:.

- ADCH and ADCL are the ADC data registers,
 - is the temperature sensor gain
 - TS_OFFSET is the temperature sensor offset correction term
- TS_GAIN is the unsigned fixed point 8-bit temperature sensor gain factor in 1/128th units stored in the signature row.
- TS_OFFSET is the signed twos complement temperature sensor offset reading stored in the signature row. See Table 26-5 on page 236 for signature row parameter address

The following code example allows to read signature row data

```
.equ TS_GAIN = 0x0003
.equ TS_OFFSET = 0x0002
LDI R30,LOW(TS_GAIN)
LDI R31,HIGH (TS_GAIN)
RCALL Read_signature_row
MOV R17,R16; Save R16 result
LDI R30,LOW(TS_OFFSET)
LDI R31,HIGH (TS_OFFSET)
RCALL Read_signature_row
; R16 holds TS_OFFSET and R17 holds TS_GAIN
Read_signature_row:
IN R16,SPMCSR; Wait for SPMEN ready
SBRC R16,SPMEN; Exit loop here when SPMCSR is free
RJMP Read_signature_row
LDI R16,((1<<SIGRD)|(1<<SPMEN)); We need to set SIGRD and SPMEN together
OUT SPMCSR,R16; and execute the LPM within 3 cycles
LPM R16,Z
RET
```

23.9 Register Description

23.9.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
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 – REFS1:0: Reference Selection Bits

These bits select the voltage reference for the ADC, as shown in [Table 23-3](#). If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 23-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V voltage reference with external capacitor at AREF pin

• Bit 5 – ADLAR: ADC Left Adjust Result

The ADLAR bit affects the presentation of the ADC conversion result in the ADC data register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC data register immediately, regardless of any ongoing conversions. For a complete description of this bit, see [Section 23.9.3 “ADCL and ADCH – The ADC Data Register” on page 219](#).

• Bit 4 – Res: Reserved Bit

This bit is an unused bit in the Atmel® ATmega328P, and will always read as zero.

• Bits 3:0 – MUX3:0: Analog Channel Selection Bits

The value of these bits selects which analog inputs are connected to the ADC. See [Table 23-4 on page 218](#) for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 23-4. Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8 ⁽¹⁾
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Note: 1. For temperature sensor.

23.9.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – ADEN: ADC Enable

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

• Bit 6 – ADSC: ADC Start Conversion

In single conversion mode, write this bit to one to start each conversion. In free running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

• Bit 5 – ADATE: ADC Auto Trigger Enable

When this bit is written to one, auto triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC trigger select bits, ADTS in ADCSRB.

• Bit 4 – ADIF: ADC Interrupt Flag

This bit is set when an ADC conversion completes and the data registers are updated. The ADC conversion complete interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a read-modify-write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC conversion complete interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 23-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

23.9.3 ADCL and ADCH – The ADC Data Register

23.9.3.1 ADLAR = 0

Bit (0x79) (0x78)	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

23.9.3.2 ADLAR = 1

Bit (0x79) (0x78)	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC data register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in [Section 23.7 “ADC Conversion Result” on page 215](#).

23.9.4 ADCSRB – ADC Control and Status Register B

Bit (0x7B)	7	6	5	4	3	2	1	0	
	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 5:3 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be written to zero when ADCSRB is written.

- **Bit 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected interrupt flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to free running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC interrupt flag is set.

Table 23-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free running mode
0	0	1	Analog comparator
0	1	0	External interrupt request 0
0	1	1	Timer/Counter0 compare match A
1	0	0	Timer/Counter0 overflow
1	0	1	Timer/Counter1 compare match B
1	1	0	Timer/Counter1 overflow
1	1	1	Timer/Counter1 capture event

23.9.5 DIDR0 – Digital Input Disable Register 0

Bit (0x7E)	7	6	5	4	3	2	1	0	
	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
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 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be written to zero when DIDR0 is written.

- **Bit 5:0 – ADC5D..ADC0D: ADC5..0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC5..0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Note that ADC pins ADC7 and ADC6 do not have digital input buffers, and therefore do not require digital input disable bits.

36. Table of Contents

Features	1
1. Pin Configurations	3
1.1 Pin Descriptions	4
1.2 Disclaimer	5
1.3 Automotive Quality Grade	5
2. Overview	6
2.1 Block Diagram	6
3. Resources	8
4. Data Retention	8
5. About Code Examples	8
6. AVR CPU Core	9
6.1 Overview	9
6.2 ALU – Arithmetic Logic Unit	10
6.3 Status Register	10
6.4 General Purpose Register File	12
6.5 Stack Pointer	13
6.6 Instruction Execution Timing	14
6.7 Reset and Interrupt Handling	15
7. AVR Memories	17
7.1 Overview	17
7.2 In-System Reprogrammable Flash Program Memory	17
7.3 SRAM Data Memory	18
7.4 EEPROM Data Memory	19
7.5 I/O Memory	19
7.6 Register Description	20
8. System Clock and Clock Options	24
8.1 Clock Systems and their Distribution	24
8.2 Clock Sources	25
8.3 Low Power Crystal Oscillator	26
8.4 Full Swing Crystal Oscillator	27
8.5 Low Frequency Crystal Oscillator	29
8.6 Calibrated Internal RC Oscillator	29
8.7 128 kHz Internal Oscillator	30
8.8 External Clock	31
8.9 Clock Output Buffer	31
8.10 Timer/Counter Oscillator	32
8.11 System Clock Prescaler	32
8.12 Register Description	32

9. Power Management and Sleep Modes	34
9.1 Sleep Modes	34
9.2 BOD Disable	35
9.3 Idle Mode	35
9.4 ADC Noise Reduction Mode	35
9.5 Power-down Mode	35
9.6 Power-save Mode	36
9.7 Standby Mode	36
9.8 Extended Standby Mode	36
9.9 Power Reduction Register	36
9.10 Minimizing Power Consumption	36
9.11 Register Description	37
10. System Control and Reset	40
10.1 Resetting the AVR	40
10.2 Reset Sources	40
10.3 Power-on Reset	41
10.4 External Reset	42
10.5 Brown-out Detection	42
10.6 Watchdog System Reset	43
10.7 Internal Voltage Reference	43
10.8 Watchdog Timer	43
10.9 Register Description	46
11. Interrupts	49
11.1 Interrupt Vectors in ATmega328P	49
11.2 Register Description	52
12. External Interrupts	53
12.1 Pin Change Interrupt Timing	53
12.2 Register Description	54
13. I/O-Ports	58
13.1 Overview	58
13.2 Ports as General Digital I/O	59
13.3 Alternate Port Functions	63
13.4 Register Description	72
14. 8-bit Timer/Counter0 with PWM	74
14.1 Features	74
14.2 Overview	74
14.3 Timer/Counter Clock Sources	75
14.4 Counter Unit	75
14.5 Output Compare Unit	76
14.6 Compare Match Output Unit	77
14.7 Modes of Operation	78
14.8 Timer/Counter Timing Diagrams	82
14.9 Register Description	84

15.	16-bit Timer/Counter1 with PWM	89
15.1	Features	89
15.2	Overview	89
15.3	Accessing 16-bit Registers	91
15.4	Timer/Counter Clock Sources	94
15.5	Counter Unit	95
15.6	Input Capture Unit	95
15.7	Output Compare Units	97
15.8	Compare Match Output Unit	99
15.9	Modes of Operation	100
15.10	Timer/Counter Timing Diagrams	106
15.11	Register Description	108
16.	Timer/Counter0 and Timer/Counter1 Prescalers	114
16.1	Internal Clock Source	114
16.2	Prescaler Reset	114
16.3	External Clock Source	114
16.4	Register Description	115
17.	8-bit Timer/Counter2 with PWM and Asynchronous Operation	116
17.1	Features	116
17.2	Overview	116
17.3	Timer/Counter Clock Sources	117
17.4	Counter Unit	117
17.5	Output Compare Unit	118
17.6	Compare Match Output Unit	119
17.7	Modes of Operation	120
17.8	Timer/Counter Timing Diagrams	124
17.9	Asynchronous Operation of Timer/Counter2	126
17.10	Timer/Counter Prescaler	127
17.11	Register Description	127
18.	SPI – Serial Peripheral Interface	135
18.1	Features	135
18.2	Overview	135
18.3	SS Pin Functionality	138
18.4	Data Modes	139
18.5	Register Description	140
19.	USART0	143
19.1	Features	143
19.2	Overview	143
19.3	Clock Generation	144
19.4	Frame Formats	147
19.5	USART Initialization	148
19.6	Data Transmission – The USART Transmitter	150
19.7	Data Reception – The USART Receiver	152
19.8	Asynchronous Data Reception	155
19.9	Multi-processor Communication Mode	158
19.10	Register Description	159
19.11	Examples of Baud Rate Setting	163

20.	USART in SPI Mode	166
20.1	Features	166
20.2	Overview	166
20.3	Clock Generation	166
20.4	SPI Data Modes and Timing	167
20.5	Frame Formats	167
20.6	Data Transfer	169
20.7	AVR USART MSPIM versus AVR SPI	170
20.8	Register Description	170
21.	2-wire Serial Interface	173
21.1	Features	173
21.2	2-wire Serial Interface Bus Definition	173
21.3	Data Transfer and Frame Format	174
21.4	Multi-master Bus Systems, Arbitration and Synchronization	177
21.5	Overview of the TWI Module	179
21.6	Using the TWI	181
21.7	Transmission Modes	184
21.8	Multi-master Systems and Arbitration	197
21.9	Register Description	198
22.	Analog Comparator	202
22.1	Overview	202
22.2	Analog Comparator Multiplexed Input	202
22.3	Register Description	203
23.	Analog-to-Digital Converter	205
23.1	Features	205
23.2	Overview	205
23.3	Starting a Conversion	207
23.4	Prescaling and Conversion Timing	208
23.5	Changing Channel or Reference Selection	211
23.6	ADC Noise Canceler	211
23.7	ADC Conversion Result	215
23.8	Temperature Measurement	215
23.9	Register Description	217
24.	debugWIRE On-chip Debug System	221
24.1	Features	221
24.2	Overview	221
24.3	Physical Interface	221
24.4	Software Break Points	222
24.5	Limitations of debugWIRE	222
24.6	Register Description	222
25.	Self-Programming the Flash, ATmega328P	223
25.1	Overview	223
25.2	Addressing the Flash During Self-Programming	224
25.3	Register Description	228

26.	Boot Loader Support – Read-While-Write Self-Programming	229
26.1	Features	229
26.2	Overview	229
26.3	Application and Boot Loader Flash Sections	229
26.4	Read-While-Write and No Read-While-Write Flash Sections	229
26.5	Boot Loader Lock Bits	231
26.6	Entering the Boot Loader Program	232
26.7	Addressing the Flash During Self-Programming	233
26.8	Self-Programming the Flash	234
26.9	Register Description	239
27.	Memory Programming	241
27.1	Program And Data Memory Lock Bits	241
27.2	Fuse Bits	242
27.3	Signature Bytes	244
27.4	Calibration Byte	244
27.5	Page Size	245
27.6	Parallel Programming Parameters, Pin Mapping, and Commands	245
27.7	Parallel Programming	247
27.8	Serial Downloading	254
28.	Electrical Characteristics	258
28.1	Absolute Maximum Ratings	258
28.2	DC Characteristics	258
28.3	DC Characteristics	260
28.4	Speed Grades	260
28.5	Clock Characteristics	260
28.6	System and Reset Characteristics	261
28.7	SPI Timing Characteristics	262
28.8	2-wire Serial Interface Characteristics	264
28.9	ADC Characteristics	265
28.10	Parallel Programming Characteristics	265
29.	Typical Characteristics	268
29.1	ATmega328P Typical Characteristics	268
30.	Register Summary	275
31.	Instruction Set Summary	281
32.	Ordering Information	285
32.1	ATmega328P	285
33.	Packaging Information	286
33.1	MA	286
33.2	PN	287
34.	Errata	288
34.1	Errata ATmega328P	288
35.	Revision History	288
36.	Table of Contents	289



Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA T: (+1)(408) 441.0311 F: (+1)(408) 436.4200 | www.atmel.com

© 2015 Atmel Corporation. / Rev.: 7810D-AVR-01/15

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, AVR®, AVR Studio®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 AND 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 products 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.