



MICROCHIP ATmega164A/PA/324A/PA/644A/PA/1284/P

megaAVR® Data Sheet

Introduction

The ATmega164A/PA/324A/PA/644A/PA/1284/P is a low power, CMOS 8-bit microcontrollers based on the AVR® enhanced RISC architecture. The ATmega164A/PA/324A/PA/644A/PA/1284/P is a 40/49-pins device ranging from 16 KB to 128 KB Flash, with 1 KB to 16 KB SRAM, 512 Bytes to 4 KB EEPROM. By executing instructions in a single clock cycle, the devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

Features

- High-performance, low-power 8-bit AVR® Microcontroller
- Advanced RISC architecture
 - 131 powerful Instructions – most single-clock cycle execution
 - 32 × 8 general purpose working registers
 - Fully static operation
 - Up to 20MIPS throughput at 20MHz
 - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
 - 16/32/64/128KBytes of In-System Self-programmable Flash program memory
 - 512/1K/2K/4KBytes EEPROM
 - 1/2/4/16KBytes Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
 - Data retention: 20 years at 85°C/ 100 years at 25°C⁽¹⁾
 - 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
- QTouch® Library Support
 - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix™ acquisition
 - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
 - One/two 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
 - Differential mode with selectable gain at 1×, 10× or 200×
 - Byte-oriented Two-wire Serial Interface
 - Two Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - 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 RC 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
 - 32 Programmable I/O Lines
 - 40-pin PDIP, 44-lead TQFP, 44-pad VQFN/QFN/MLF
 - 44-pad DRQFN
 - 49-ball VFBGA
- Operating Voltages
 - 1.8 - 5.5V
- Speed Grades
 - 0 - 4MHz @ 1.8 - 5.5V
 - 0 - 10MHz @ 2.7 - 5.5V
 - 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
 - Active: 0.4mA
 - Power-down Mode: 0.1µA
 - Power-save Mode: 0.6µA (Including 32kHz RTC)

Note: 1. See [“Data retention” on page 17](#) for details.

Table of Contents

1	<i>Pin configurations</i>	11
1.1	Pinout - PDIP/TQFP/VQFN/QFN/MLF for ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P	11
1.2	Pinout - DRQFN for ATmega164A/164PA/324A/324PA	12
1.3	Pinout - VFBGA for ATmega164A/164PA/324A/324PA	13
2	<i>Overview</i>	13
2.1	Block diagram	14
2.2	Comparison between ATmega164A, ATmega164PA, ATmega324A, ATmega324PA, ATmega644A, ATmega644PA, ATmega1284 and ATmega1284P	15
2.3	Pin Descriptions	15
3	<i>Resources</i>	17
4	<i>About code examples</i>	17
5	<i>Data retention</i>	17
6	<i>Capacitive touch sensing</i>	17
7	<i>AVR CPU Core</i>	18
7.1	Overview	18
7.2	ALU – Arithmetic Logic Unit	19
7.3	Status Register	19
7.4	General Purpose Register File	21
7.5	Stack Pointer	22
7.6	Instruction Execution Timing	23
7.7	Reset and interrupt handling	24
8	<i>AVR memories</i>	27
8.1	Overview	27
8.2	In-System Reprogrammable Flash Program Memory	27
8.3	SRAM data memory	28
8.4	EEPROM data memory	30
8.5	I/O memory	31
8.6	Register Description	32
9	<i>System clock and clock options</i>	38
9.1	Clock systems and their distribution	38
9.2	Clock Sources	39

9.3	Low Power Crystal Oscillator.....	41
9.4	Full swing Crystal Oscillator	42
9.5	Low Frequency Crystal Oscillator.....	43
9.6	Calibrated Internal RC Oscillator	44
9.7	128kHz internal oscillator	45
9.8	External clock	46
9.9	Timer/Counter Oscillator.....	46
9.10	Clock Output Buffer	47
9.11	System Clock Prescaler	47
9.12	Register description.....	48
10	Power management and sleep modes	50
10.1	Overview.....	50
10.2	Sleep Modes.....	50
10.3	BOD disable ⁽¹⁾	51
10.4	Idle mode.....	51
10.5	ADC Noise Reduction mode.....	51
10.6	Power-down mode.....	52
10.7	Power-save mode.....	52
10.8	Standby mode	52
10.9	Extended Standby mode	52
10.10	Power Reduction Register.....	53
10.11	Minimizing Power Consumption	53
10.12	Register description.....	55
11	System Control and Reset	58
11.1	Resetting the AVR.....	58
11.2	Internal Voltage Reference.....	62
11.3	Watchdog Timer	63
11.4	Register description.....	66
12	Interrupts	69
12.1	Overview.....	69
12.2	Interrupt Vectors in ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P	69
12.3	Register description.....	73
13	External Interrupts	75
13.1	Overview.....	75
13.2	Register description.....	76

14	I/O-Ports	80
14.1	Overview	80
14.2	Ports as General Digital I/O	81
14.3	Alternate Port Functions	85
14.4	Register description	97
15	8-bit Timer/Counter0 with PWM	99
15.1	Features	99
15.2	Overview	99
15.3	Timer/Counter Clock Sources	100
15.4	Counter Unit	100
15.5	Output Compare unit	101
15.6	Compare Match Output unit	102
15.7	Modes of operation	103
15.8	Timer/Counter Timing diagrams	107
15.9	Register description	109
16	16-bit Timer/Counter1 and Timer/Counter3⁽¹⁾ with PWM	115
16.1	Features	115
16.2	Overview	115
16.3	Accessing 16-bit Registers	117
16.4	Timer/Counter Clock Sources	120
16.5	Prescaler Reset	120
16.6	Counter Unit	121
16.7	Input Capture Unit	122
16.8	Output Compare units	123
16.9	Compare Match Output unit	126
16.10	Modes of Operation	127
16.11	Timer/Counter Timing diagrams	134
16.12	Register description	136
17	8-bit Timer/Counter2 with PWM and asynchronous operation	146
17.1	Features	146
17.2	Overview	146
17.3	Timer/Counter clock sources	147
17.4	Counter unit	148
17.5	Output Compare unit	148
17.6	Compare Match Output unit	150

17.7	Modes of operation	151
17.8	Timer/Counter Timing diagrams	155
17.9	Asynchronous Operation of Timer/Counter2	156
17.10	Timer/Counter Prescaler	158
17.11	Register description	159
18	<i>SPI – Serial Peripheral Interface</i>	166
18.1	Features	166
18.2	Overview	166
18.3	\overline{SS} pin functionality	170
18.4	Data modes	170
18.5	Register description	172
19	<i>USART</i>	175
19.1	Features	175
19.2	USART1 and USART0	175
19.3	Overview	175
19.4	Clock Generation	176
19.5	Frame formats	179
19.6	USART Initialization	180
19.7	Data Transmission – The USART Transmitter	181
19.8	Data Reception – The USART Receiver	184
19.9	Asynchronous Data Reception	188
19.10	Multi-processor Communication mode	191
19.11	Register description	193
19.12	Examples of Baud Rate Setting	198
20	<i>USART in SPI mode</i>	202
20.1	Features	202
20.2	Overview	202
20.3	Clock Generation	202
20.4	SPI Data Modes and Timing	203
20.5	Frame Formats	203
20.6	Data Transfer	205
20.7	AVR USART MSPIM vs. AVR SPI	207
20.8	Register description	208
21	<i>Two-wire Serial Interface</i>	211
21.1	Features	211

21.2	Two-wire Serial Interface bus definition.....	211
21.3	Data Transfer and Frame Format.....	212
21.4	Multi-master Bus Systems, Arbitration and Synchronization.....	214
21.5	Overview of the TWI Module	217
21.6	Using the TWI.....	219
21.7	Transmission modes	222
21.8	Multi-master Systems and Arbitration.....	234
21.9	Register description.....	236
22	<i>AC - Analog Comparator</i>	240
22.1	Overview.....	240
22.2	Analog Comparator Multiplexed Input	240
22.3	Register description.....	241
23	<i>ADC - Analog-to-digital converter</i>	243
23.1	Features	243
23.2	Overview.....	243
23.3	Operation.....	244
23.4	Starting a conversion.....	245
23.5	Prescaling and Conversion Timing.....	246
23.6	Changing Channel or Reference Selection	249
23.7	ADC Noise Canceler	250
23.8	ADC Conversion Result.....	255
23.9	Register description.....	257
24	<i>JTAG interface and on-chip debug system</i>	262
24.1	Features	262
24.2	Overview.....	262
24.3	TAP – Test Access Port	262
24.4	TAP controller.....	264
24.5	Using the Boundary-scan Chain.....	265
24.6	Using the On-chip Debug System	265
24.7	On-chip Debug Specific JTAG Instructions	266
24.8	Using the JTAG Programming Capabilities	266
24.9	Bibliography.....	267
24.10	Register description.....	267
25	<i>IEEE 1149.1 (JTAG) Boundary-scan</i>	268
25.1	Features	268

25.2	Overview	268
25.3	Data Registers	269
25.4	Boundary-scan Specific JTAG Instructions	270
25.5	Boundary-scan Chain	271
25.6	ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P Boundary-scan order	274
25.7	Boundary-scan Description Language Files	275
25.8	Register description	276
26	<i>Boot loader support – read-while-write self-programming</i>	277
26.1	Features	277
26.2	Overview	277
26.3	Application and Boot Loader Flash Sections	277
26.4	Read-While-Write and No Read-While-Write Flash Sections	278
26.5	Boot Loader Lock Bits	280
26.6	Entering the Boot Loader Program	281
26.7	Addressing the Flash During Self-Programming	282
26.8	Self-Programming the Flash	283
26.9	Register description	293
27	<i>Memory programming</i>	295
27.1	Program And Data Memory Lock Bits	295
27.2	Fuse bits	296
27.3	Signature Bytes	298
27.4	Calibration byte	298
27.5	Page Size	298
27.6	Parallel Programming Parameters, Pin Mapping, and Commands	299
27.7	Parallel programming	301
27.8	Serial downloading	309
27.9	Serial Programming Instruction set	312
27.10	Programming via the JTAG Interface	314
28	<i>Electrical characteristics (TA = -40°C to 85°C)</i>	326
28.1	DC Characteristics	326
28.2	Speed grades	332
28.3	Clock characteristics	333
28.4	System and reset characteristics	334
28.5	External interrupts characteristics	334
28.6	SPI timing characteristics	335

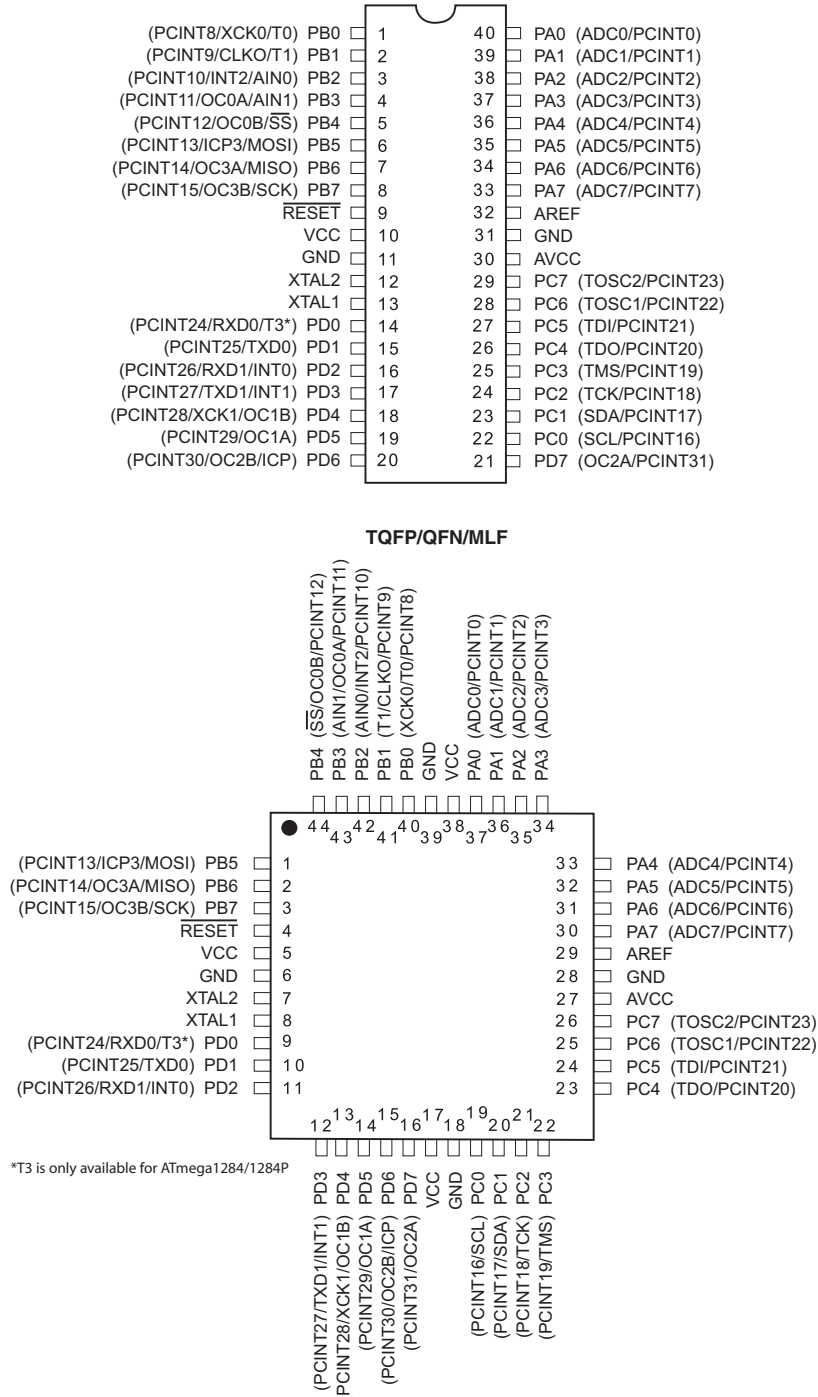
28.7	Two-wire Serial Interface Characteristics	336
28.8	ADC characteristics	338
29	Electrical Characteristics - TA = -40°C to 105°C	341
29.1	DC Characteristics	341
30	Typical characteristics -TA = -40°C to 85°C	344
30.1	ATmega164A typical characteristics - TA = -40°C to 85°C	344
30.2	ATmega164PA typical characteristics - TA = -40°C to 85°C	371
30.3	ATmega324A typical characteristics - TA = -40°C to 85°C	397
30.4	ATmega324PA typical characteristics - TA = -40°C to 85°C	423
30.5	ATmega644A typical characteristics - TA = -40°C to 85°C	449
30.6	ATmega644PA typical characteristics - TA = -40°C to 85°C	475
30.7	ATmega1284 typical characteristics - TA = -40°C to 85°C	501
30.8	ATmega1284P typical characteristics - TA = -40°C to 85°C	527
31	Typical Characteristics - TA = -40°C to 105°C	553
31.1	ATmega164PA Typical Characteristics - TA = -40°C to 105°C	554
31.2	ATmega324PA Typical Characteristics - TA = -40°C to 105°C	574
31.3	ATmega644PA Typical Characteristics - TA = -40°C to 105°C	594
31.4	ATmega1284P typical characteristics - TA = -40°C to 105°C	613
32	Register summary	636
33	Instruction set summary	640
34	Ordering information	643
34.1	ATmega164A.....	643
34.2	ATmega164PA	644
34.3	ATmega324A.....	645
34.4	ATmega324PA	646
34.5	ATmega644A.....	647
34.6	ATmega644PA	648
34.7	ATmega1284	649
34.8	ATmega1284P.....	650
35	Packaging information	651
35.1	44A	651
35.2	40P6	652
35.3	44M1.....	653
35.4	44MC	654

35.5	49C2	655
36	<i>Errata</i>	656
36.1	Errata for ATmega164A.....	656
36.2	Errata for ATmega164PA	656
36.3	Errata for ATmega324A.....	656
36.4	Errata for ATmega324PA	656
36.5	Errata for ATmega644A.....	656
36.6	Errata for ATmega644PA	656
36.7	Errata for ATmega1284	656
36.8	Errata for ATmega1284P.....	656
37	<i>Data sheet revision history</i>	657
37.1	Rev. A - 10/2018.....	657
37.2	Rev. 8272G - 01/2015	657
37.3	Rev. 8272F - 08/2014	657
37.4	Rev. 8272E - 04/2013.....	658
37.5	Rev. 8272D - 05/12	658
37.6	Rev. 8272C - 06/11	658
37.7	Rev. 8272B - 05/11.....	658
37.8	Rev. 8272A - 01/10.....	659

1. Pin configurations

1.1 Pinout - PDIP/TQFP/VQFN/QFN/MLF for ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P

Figure 1-1. Pinout



Note: The large center pad underneath the VQFN/QFN/MLF package should be soldered to ground on the board to ensure good mechanical stability.

1.2 Pinout - DRQFN for ATmega164A/164PA/324A/324PA

Figure 1-2. DRQFN - pinout

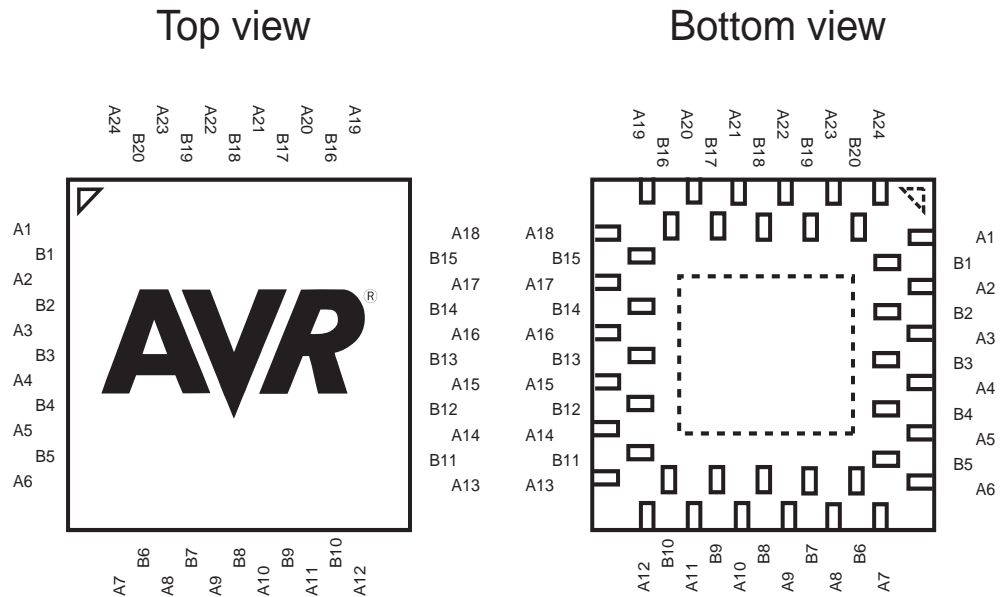


Table 1-1. DRQFN - pinout.

A1	PB5	A7	PD3	A13	PC4	A19	PA3
B1	PB6	B6	PD4	B11	PC5	B16	PA2
A2	PB7	A8	PD5	A14	PC6	A20	PA1
B2	RESET	B7	PD6	B12	PC7	B17	PA0
A3	VCC	A9	PD7	A15	AVCC	A21	VCC
B3	GND	B8	VCC	B13	GND	B18	GND
A4	XTAL2	A10	GND	A16	AREF	A22	PB0
B4	XTAL1	B9	PC0	B14	PA7	B19	PB1
A5	PD0	A11	PC1	A17	PA6	A23	PB2
B5	PD1	B10	PC2	B15	PA5	B20	PB3
A6	PD2	A12	PC3	A18	PA4	A24	PB4

1.3 Pinout - VFBGA for ATmega164A/164PA/324A/324PA

Figure 1-3. VFBGA - pinout

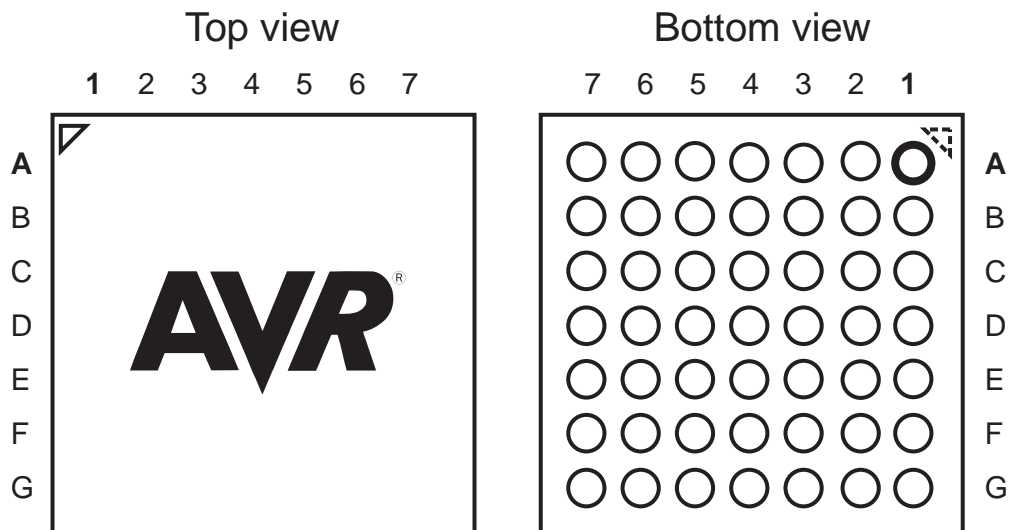


Table 1-2. BGA - pinout

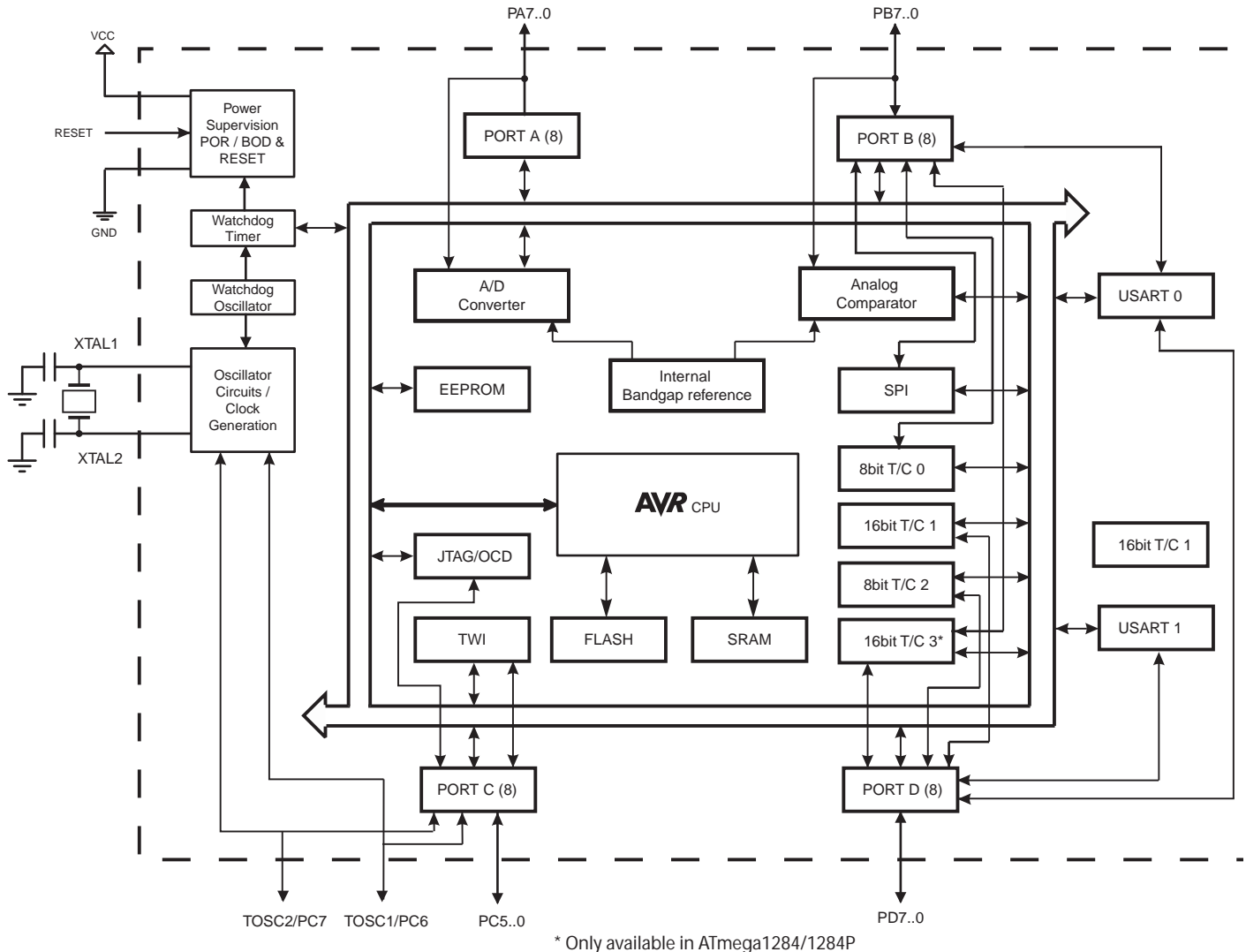
	1	2	3	4	5	6	7
A	GND	PB4	PB2	GND	VCC	PA2	GND
B	PB6	PB5	PB3	PB0	PA0	PA3	PA5
C	VCC	RESET	PB7	PB1	PA1	PA6	AREF
D	GND	XTAL2	PD0	GND	PA4	PA7	GND
E	XTAL1	PD1	PD5	PD7	PC5	PC7	AVCC
F	PD2	PD3	PD6	PC0	PC2	PC4	PC6
G	GND	PD4	VCC	GND	PC1	PC3	GND

2. Overview

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P 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 ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block diagram

Figure 2-1. Block diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P provide the following features:

16/32/64/128Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 512/1K/2K/4Kbytes EEPROM, 1/2/4/16Kbytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three (four for ATmega1284/1284P) flexible Timer/Counters with compare modes and PWM, 2 USARTs, a byte oriented two-wire Serial Interface, a 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a

timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

Microchip offers the QTouch library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression™ (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using the high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

2.2 Comparison between ATmega164A, ATmega164PA, ATmega324A, ATmega324PA, ATmega644A, ATmega644PA, ATmega1284 and ATmega1284P

Table 2-1. Differences between ATmega164A, ATmega164PA, ATmega324A, ATmega324PA, ATmega644A, ATmega644PA, ATmega1284 and ATmega1284P

Device	Flash	EEPROM	RAM	Units
ATmega164A	16K	512	1K	bytes
ATmega164PA	16K	512	1K	
ATmega324A	32K	1K	2K	
ATmega324PA	32K	1K	2K	
ATmega644A	64K	2K	4K	
ATmega644PA	64K	2K	4K	
ATmega1284	128K	4K	16K	
ATmega1284P	128K	4K	16K	

2.3 Pin Descriptions

2.3.1 VC

Digital supply voltage.

2.3.2 GND

Ground.

2.3.3 Port A (PA7:PA0)

Port A serves as analog inputs to the Analog-to-digital Converter.

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

Port A also serves the functions of various special features of the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P as listed on [page 87](#).

2.3.4 Port B (PB7:PB0)

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.

Port B also serves the functions of various special features of the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P as listed on [page 88](#).

2.3.5 Port C (PC7:PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C 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.

Port C also serves the functions of the JTAG interface, along with special features of the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P as listed on [page 91](#).

2.3.6 Port D (PD7:PD0)

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.

Port D also serves the functions of various special features of the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P as listed on [page 94](#).

2.3.7 RESET

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 "[System and reset characteristics](#)" on [page 334](#). Shorter pulses are not guaranteed to generate a reset.

2.3.8 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.3.9 XTAL2

Output from the inverting Oscillator amplifier.

2.3.10 AVCC

AVCC is the supply voltage pin for Port A and the Analog-to-digital Converter. It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

2.3.11 AREF

This is the analog reference pin for the Analog-to-digital Converter.

3. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on www.microchip.com

4. About code examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Confirm with the C compiler documentation for more details.

The code examples assume that the part specific header file is included before compilation. 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".

5. Data retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

6. Capacitive touch sensing

The QTouch Library provides a simple to use solution to realize touch sensitive interfaces on most AVR microcontrollers. The QTouch Library includes support for the QTouch and QMatrix acquisition methods.

Touch sensing can be added to any application by linking the appropriate QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

The QTouch Library is FREE and downloadable from www.microchip.com. For implementation details and other information, refer to the [QTouch Library User Guide](#) - also available for download from the microchip website.

8. AVR memories

8.1 Overview

This section describes the different memories in the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

8.2 In-System Reprogrammable Flash Program Memory

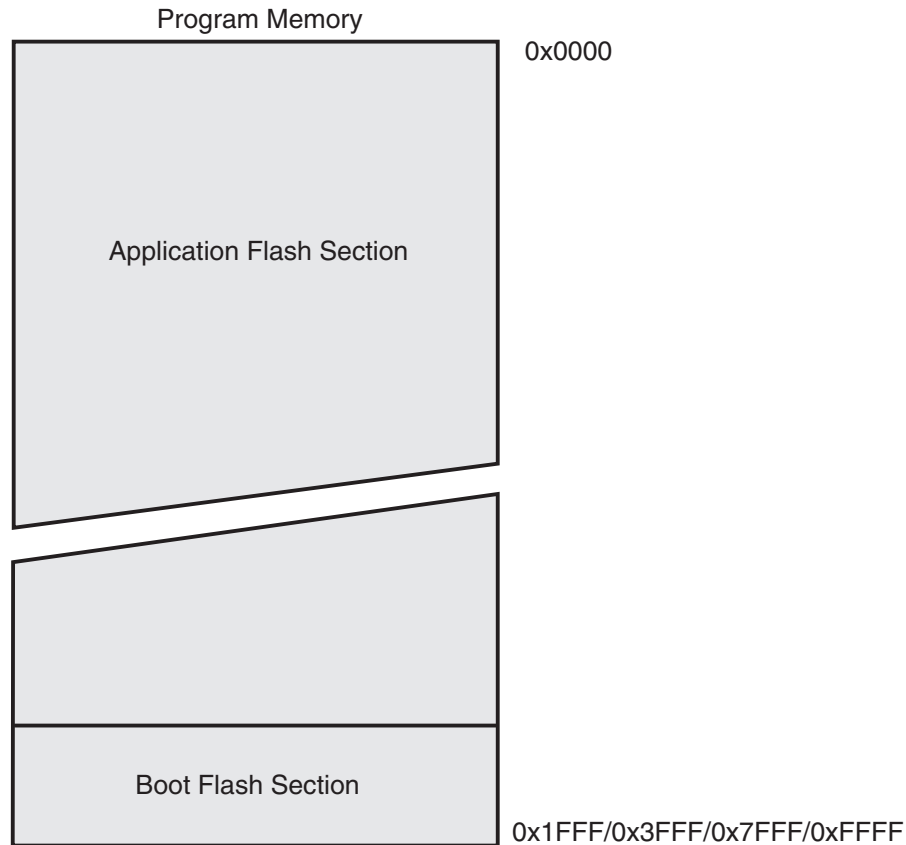
The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P contains 16/32/64/128Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 32/64 x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P Program Counter (PC) is 15/16 bits wide, thus addressing the 32/64K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in ["Memory programming" on page 295](#). ["Memory programming" on page 295](#) contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction description).

Timing diagrams for instruction fetch and execution are presented in ["Instruction Execution Timing" on page 23](#).

Figure 8-1. Program memory map



8.3 SRAM data memory

Figure 8-2 shows how the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P SRAM Memory is organized.

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from \$060 - \$FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The first 4,352 Data Memory locations address both the Register File, the I/O Memory, Extended I/O Memory, and the internal data SRAM. The first 32 locations address the Register file, the next 64 location the standard I/O Memory, then 160 locations of Extended I/O memory and the next 4,096 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register file, registers R26 to R31 feature the indirect addressing pointer registers.

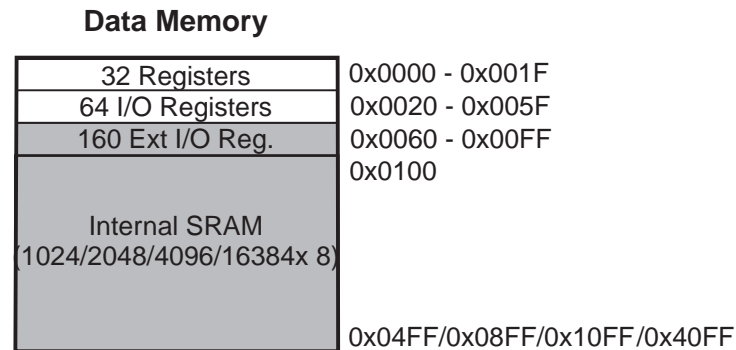
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O registers, 160 Extended I/O Registers and the 1024/2048/4096 bytes of internal data SRAM in the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P are all accessible through all these addressing modes. The Register File is described in ["General Purpose Register File" on page 21](#).

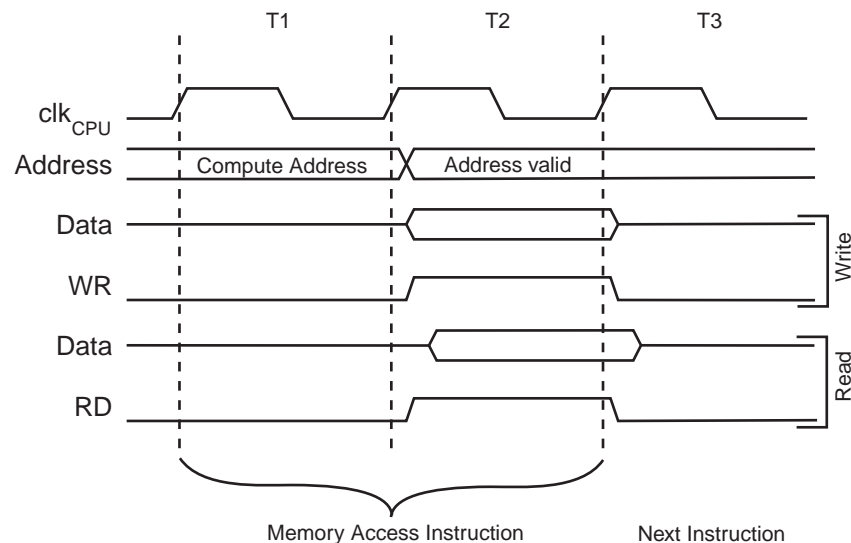
Figure 8-2. Data Memory Map for ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P



8.3.1 Data memory access times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in [Figure 8-3](#).

Figure 8-3. On-chip data SRAM access cycles



8.4 EEPROM data memory

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P contains 512/1K/2K/4Kbytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG and Parallel data downloading to the EEPROM, see [page 299](#), [page 309](#) and [page 314](#) respectively.

8.4.1 EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space. See ["Register Description" on page 32](#) for details.

The write access time for the EEPROM is given in [Table 8-2 on page 34](#). A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See [Section "8.4.2" on page 30](#) for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

8.4.2 Preventing EEPROM corruption

During periods of low V_{CC} , the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

8.5 I/O memory

The I/O space definition of the

ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P is shown in ["Register summary" on page 636](#).

All ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVR's, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P contains three General Purpose I/O Registers, see ["Register Description" on page 32](#). These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

Register description

14.3.5 MCUCR – MCU Control Register

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

Note: 1. Only available in the ATmega164PA/324PA/644PA/1284P.

• 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 ["Configuring the Pin" on page 81](#) for more details about this feature.

14.3.6 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.3.7 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.3.8 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14.3.9 PORTB – 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	

14.3.10 DDRB – 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	

14.3.11 PINB – 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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14.3.12 PORTC – Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.3.13 DDRC – Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.3.14 PINC – Port C Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14.3.15 PORTD – 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	

14.3.16 DDRD – 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	

14.3.17 PIND – 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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

15. 8-bit Timer/Counter0 with PWM

15.1 Features

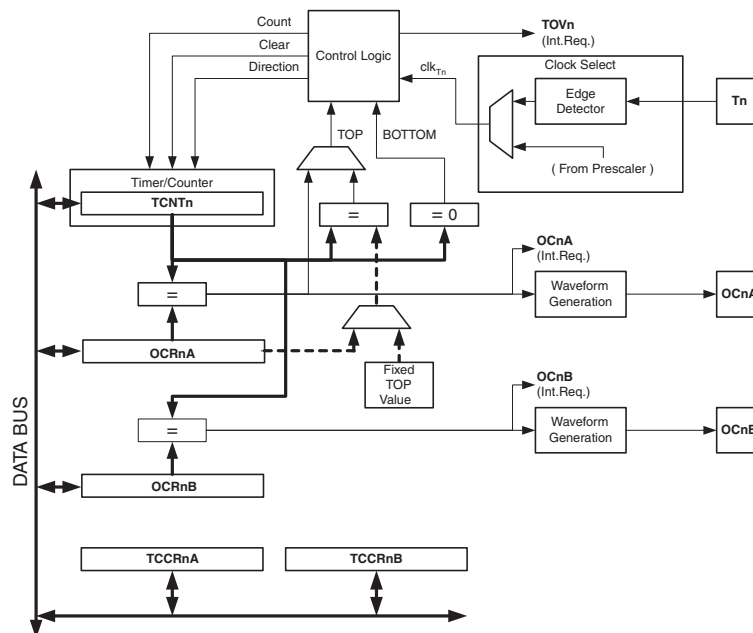
- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

15.2 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 15-1](#). For the actual placement of I/O pins, see ["Pin configurations" on page 11](#). 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 ["Register description" on page 109](#).

Figure 15-1. 8-bit Timer/Counter block diagram



15.2.1 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 “15.5” on page 101 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.

15.2.2 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, that is, TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 15-1 are also used extensively throughout the document.

Table 15-1. Definitions

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.

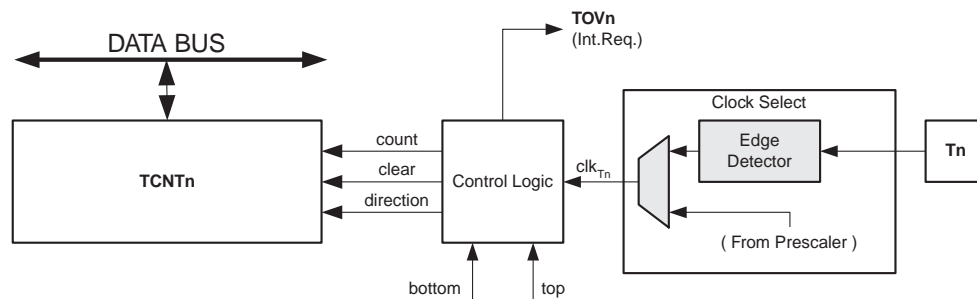
15.3 Timer/Counter Clock Sources

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 “Timer/Counter Prescaler” on page 158.

15.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable 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 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 ["Modes of operation" on page 103](#).

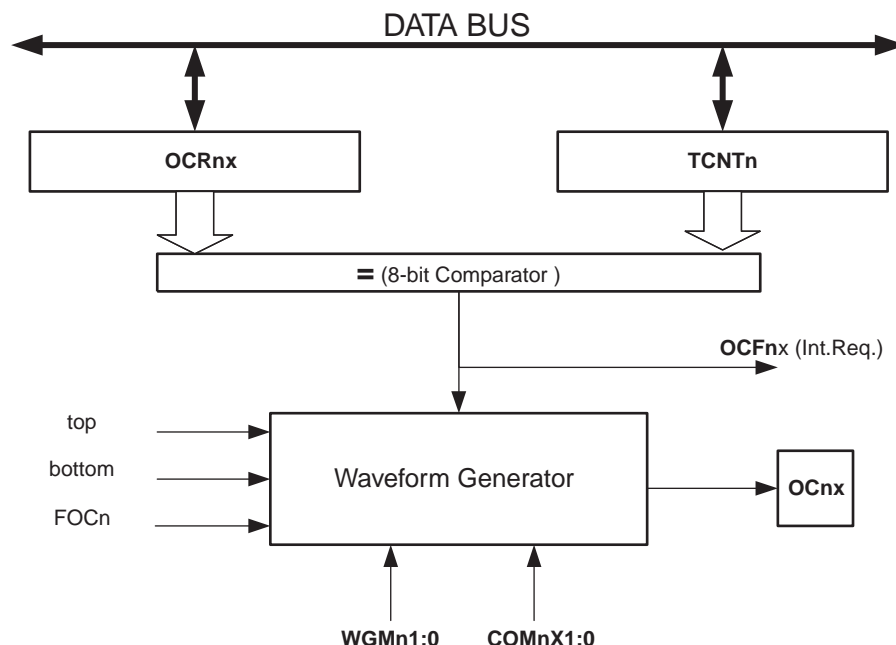
The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM02:0 bits. TOV0 can be used for generating a CPU interrupt.

15.5 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 (["Modes of operation" on page 103](#)).

[Figure 15-3](#) shows a block diagram of the Output Compare unit.

Figure 15-3. 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.

15.5.1 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).

15.5.2 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.

15.5.3 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.

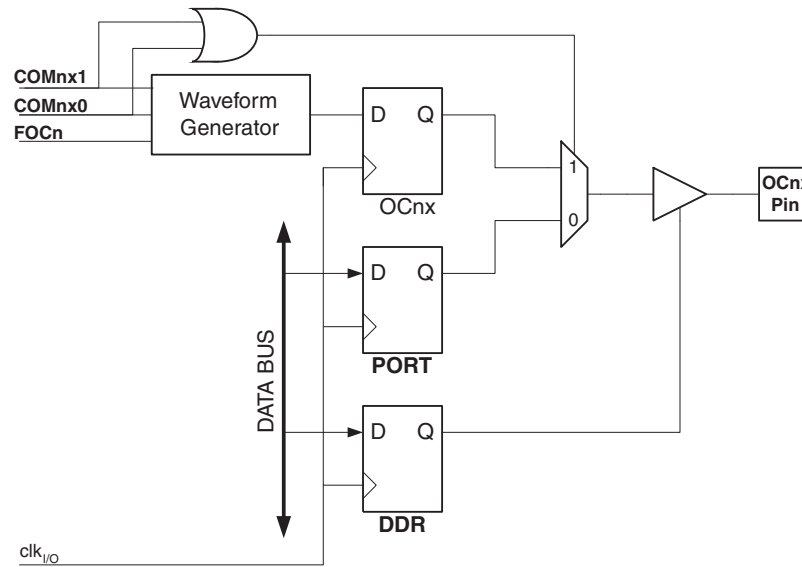
The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (FOC0x) strobe bits in Normal mode. The OC0x Registers keep their values even when changing between Waveform Generation 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.

15.6 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 15-4](#) 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 occur, the OC0x Register is reset to “0”.

Figure 15-4. 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 “15.9” on page 109.](#)

15.6.1 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 15-2 on page 109](#). For fast PWM mode, refer to [Table 15-3 on page 109](#), and for phase correct PWM refer to [Table 15-4 on page 110](#).

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.

15.7 Modes of operation

The mode of operation, that is, 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 “16.9” on page 126](#)).

For detailed timing information see [“Timer/Counter Timing diagrams” on page 107](#).

15.7.1 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.

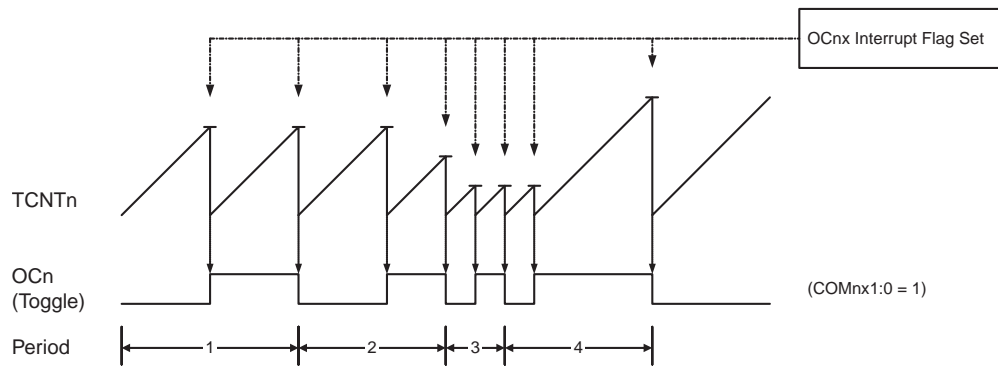
The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

15.7.2 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 15-5. The counter value (TCNT0) increases until a Compare Match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 15-5. 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:

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

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

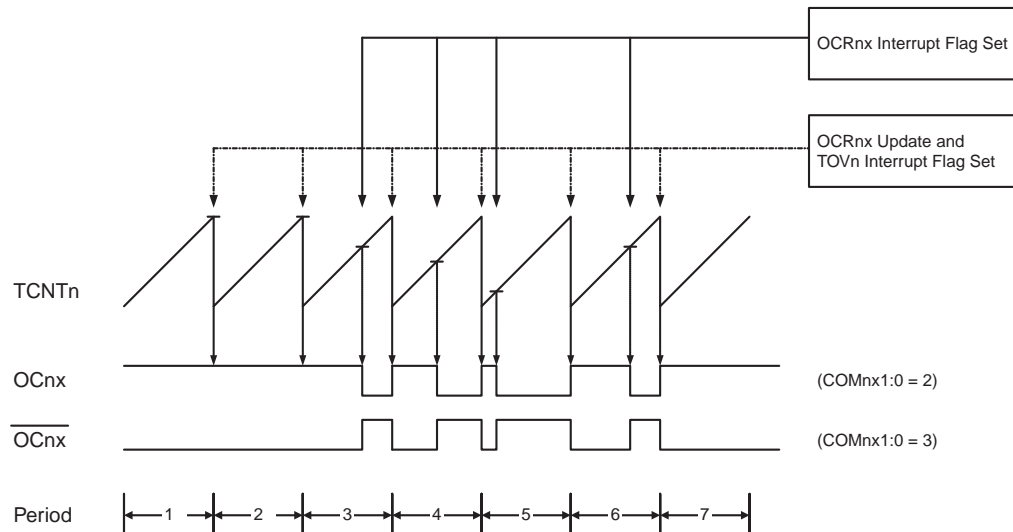
As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

15.7.3 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 15-6](#). 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 15-6. 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 15-3 on page 109](#)). 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:

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

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.)

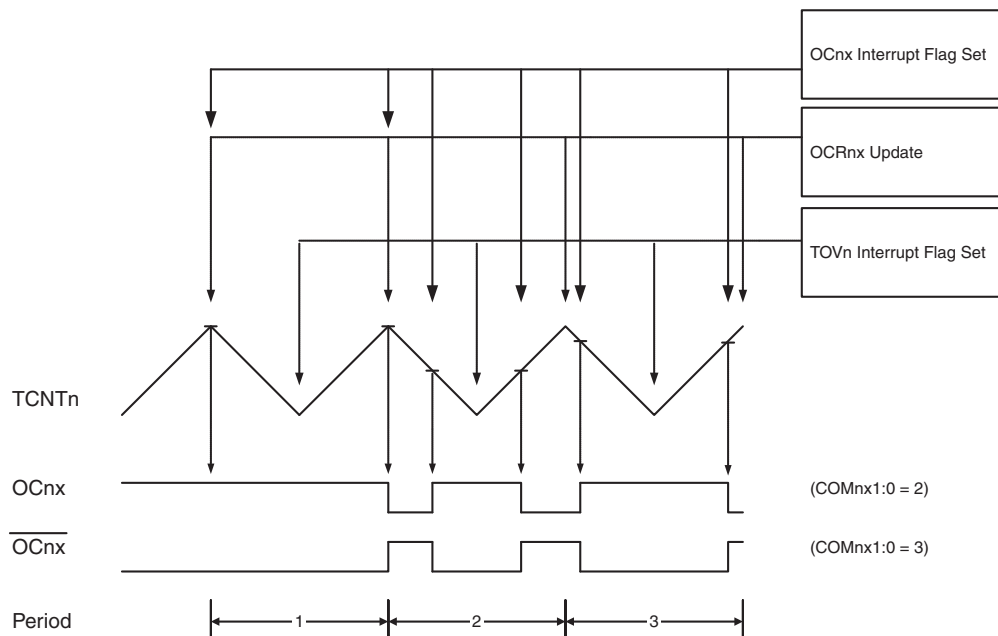
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each Compare Match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

15.7.4 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 down-counting. 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 15-7](#). The TCNT0 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 TCNT0 slopes represent Compare Matches between OCR0x and TCNT0.

Figure 15-7. 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 15-4 on page 110](#)). 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:

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

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.

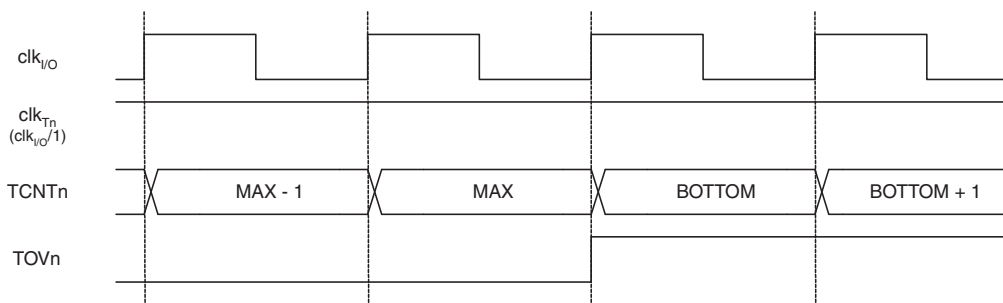
At the very start of period 2 in [Figure 15-7](#) OCnx has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR0A changes its value from MAX, like in [Figure 15-7](#). 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 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 OCR0A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

15.8 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 15-8](#) 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 15-8. Timer/Counter Timing diagram, no prescaling



[Figure 15-9 on page 108](#) shows the same timing data, but with the prescaler enabled.

Figure 15-9. Timer/Counter Timing diagram, with prescaler ($f_{clk_I/O}/8$)

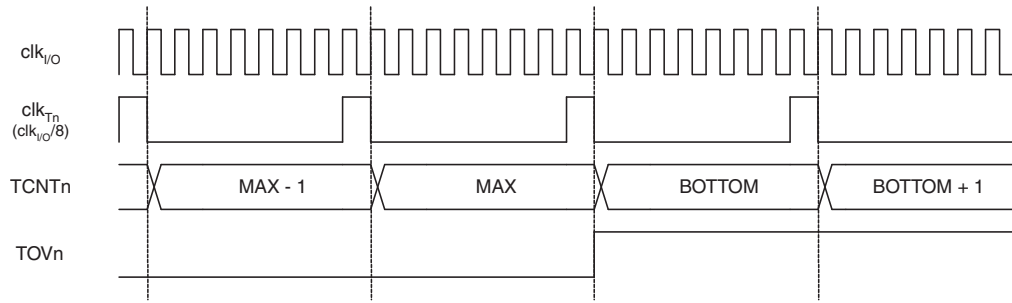


Figure 15-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 15-10. Timer/Counter Timing diagram, setting of OCF0x, with prescaler ($f_{clk_I/O}/8$)

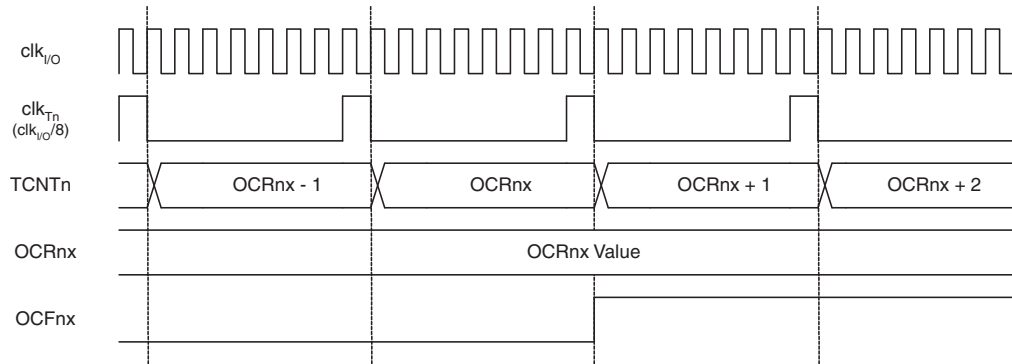
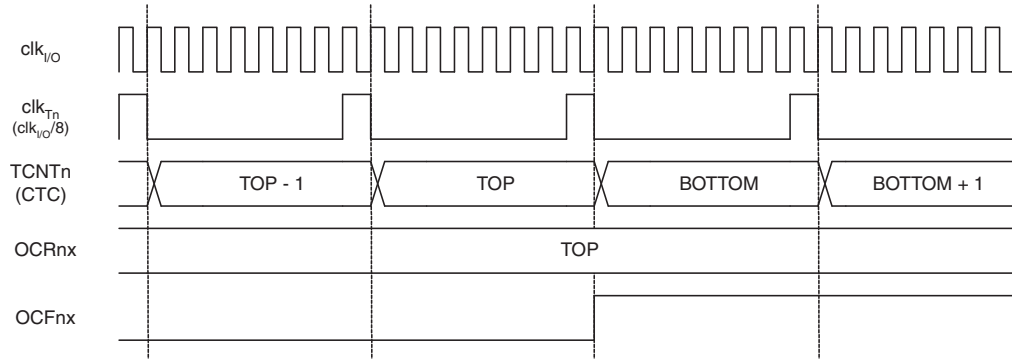


Figure 15-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 15-11. Timer/Counter Timing diagram, Clear Timer on Compare Match mode, with prescaler ($f_{clk_I/O}/8$)



15.9 Register description

15.9.1 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 15-2](#) shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 15-2. 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 15-3](#) shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 15-3. 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 ["Fast PWM mode" on page 105](#) for more details.

[Table 15-4 on page 110](#) shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 15-4. 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 ["Phase Correct PWM mode" on page 106](#) 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 15-2 on page 109](#) shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 15-5. 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 15-6](#) shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 15-6. 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 BOTTOM. See ["Fast PWM mode" on page 105](#) for more details.

Table 15-7 on page 111 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 15-7. 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 "Phase Correct PWM mode" on page 106 for more details.

• Bits 3:2 – Reserved

These bits are reserved bits in the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P 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 15-8 on page 111. 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 "Modes of Operation" on page 127).

Table 15-8. 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

15.9.2 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 – Reserved

These bits are reserved and will always read as zero.

• Bit 3 – WGM02: Waveform Generation Mode

See the description in the ["TCCR0A – Timer/Counter Control Register A" on page 109](#).

• Bits 2:0 – CS02:0: Clock Select

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

Table 15-9. Clock Select bit description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk _{I/O} /(No prescaling)
0	1	0	clk _{I/O} /8 (From prescaler)
0	1	1	clk _{I/O} /64 (From prescaler)
1	0	0	clk _{I/O} /256 (From prescaler)

Table 15-9. Clock Select bit description (Continued)

CS02	CS01	CS00	Description
1	0	1	clk _{I/O} /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.

15.9.3 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.

15.9.4 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.

15.9.5 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.

15.9.6 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 – Reserved**

These bits are reserved 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, that is, 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, that is, 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, that is, when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

15.9.7 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 – Reserved**

These bits are reserved 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 15-8, "Waveform Generation mode bit description" on page 111](#).

16. 16-bit Timer/Counter1 and Timer/Counter3⁽¹⁾ with PWM

Note: 1. Timer/Counter3 is only available in ATmega1284/1284P

16.1 Features

- True 16-bit design (that is, allows 16-bit PWM)
- Two independent Output Compare units
- Double Buffered Output Compare Registers
- One Input Capture unit
- Input Capture Noise Canceler
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)

16.2 Overview

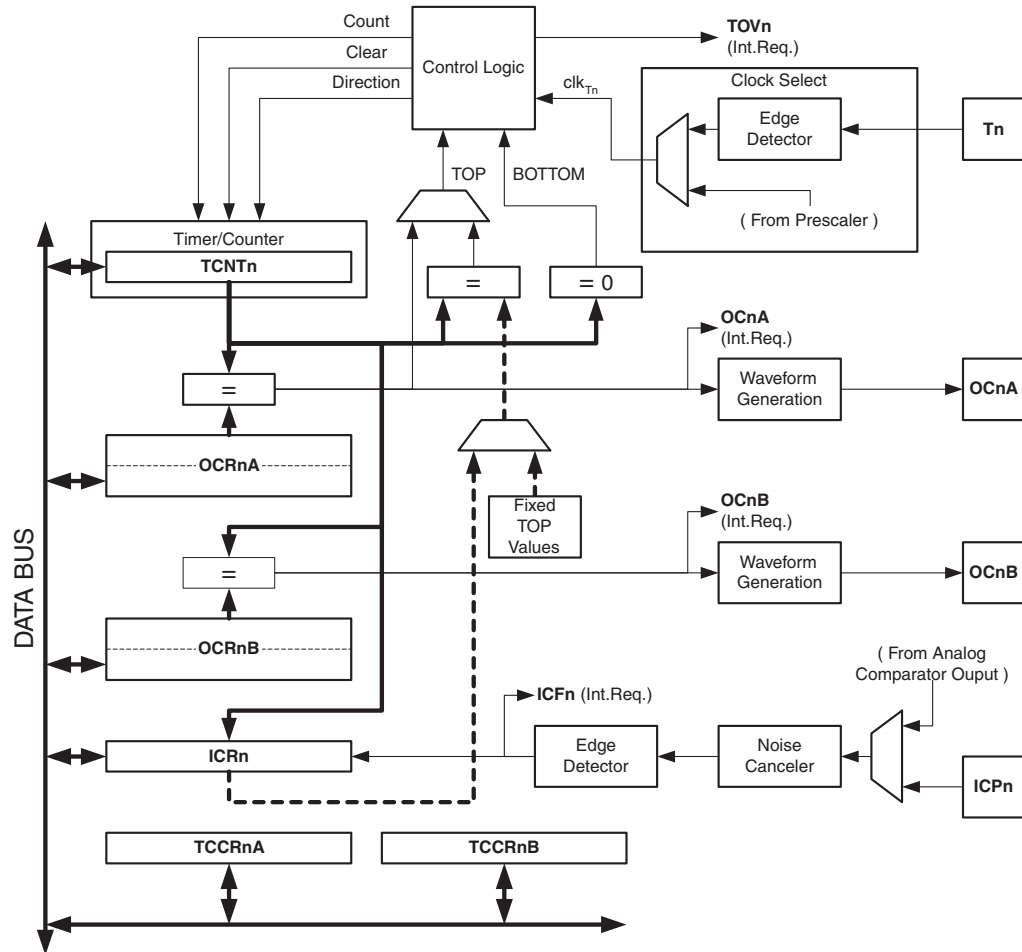
The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement.

Most register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the Output Compare unit channel. However, when using the register or bit defines in a program, the precise form must be used, that is, TCNT1 for accessing Timer/Counter1 counter value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in [Figure 16-1](#). For the actual placement of I/O pins, see ["Pin configurations" on page 11](#). 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 ["Register description" on page 136](#).

The PRTIM1 bit in ["PRR0 – Power Reduction Register 0" on page 56](#) must be written to zero to enable Timer/Counter1 module.

Figure 16-1. 16-bit Timer/Counter block diagram ⁽¹⁾



Note: 1. Refer to [Figure 1-1 on page 11](#) and ["Alternate Port Functions" on page 85](#) for Timer/Counter1 pin placement and description.

16.2.1 Registers

The *Timer/Counter* (TCNTn), *Output Compare Registers* (OCRnA/B/C), and *Input Capture Register* (ICRn) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section ["Accessing 16-bit Registers" on page 117](#). The *Timer/Counter Control Registers* (TCCRnA/B/C) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int. Req. in the figure) signals are all visible in the *Timer Interrupt Flag Register* (TIFRn). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSKn). TIFRn and TIMSKn are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the Tn 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_{Tn}).

The double buffered Output Compare Registers (OCRnA/B/C) 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 (OCnA/B/C). [See Section "16.8" on page 123](#). The compare match event will also set the Compare Match Flag (OCFnA/B/C) 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 (ICPn) or on the Analog Comparator pins (See Section “22.” on page 240) 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 OCRnA Register, the ICRn Register, or by a set of fixed values. When using OCRnA as TOP value in a PWM mode, the OCRnA 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 ICRn Register can be used as an alternative, freeing the OCRnA to be used as PWM output.

16.2.2 Definitions

The following definitions are used extensively throughout the section:

Table 16-1. Definitions

BOTTOM	The counter reaches the <i>BOTTOM</i> when it becomes 0x0000.
MAX	The counter reaches its <i>MAX</i> imum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the <i>TOP</i> 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 OCRnA or ICRn Register. The assignment is dependent of the mode of operation.

16.3 Accessing 16-bit Registers

The TCNTn, OCRnA/B/C, and ICRn 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 OCRnA/B/C 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 OCRnA/B/C and ICRn Registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly Code Examples ⁽¹⁾

```

...
; Set TCNTn to 0x01FF
ldi      r17,0x01
ldi      r16,0xFF
out      TCNTnH,r17
out      TCNTnL,r16
; Read TCNTn into r17:r16
in       r16,TCNTnL
in       r17,TCNTnH
...

```

C Code Examples ⁽¹⁾

```

unsigned int i;
...
/* Set TCNTn to 0x01FF */
TCNTn = 0x1FF;
/* Read TCNTn into i */
i = TCNTn;
...

```

Note: 1. The example code assumes that the part specific header file is included.
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 TCNTn 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 TCNTn Register contents. Reading any of the OCRnA/B/C or ICRn Registers can be done by using the same principle.

Assembly Code Example ⁽¹⁾

```
TIM16_ReadTCNTn:
    ; Save global interrupt flag
    in      r18,SREG
    ; Disable interrupts
    cli
    ; Read TCNTn into r17:r16
    in      r16,TCNTnL
    in      r17,TCNTnH
    ; Restore global interrupt flag
    out     SREG,r18
    ret
```

C Code Example ⁽¹⁾

```
unsigned int TIM16_ReadTCNTn( void )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Read TCNTn into i */
    i = TCNTn;
    /* Restore global interrupt flag */
    SREG = sreg;
    return i;
}
```

Note: 1. The example code assumes that the part specific header file is included.
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 assembly code example returns the TCNTn value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNTn Register contents. Writing any of the OCRnA/B/C or ICRn Registers can be done by using the same principle.

Assembly Code Example ⁽¹⁾

```
TIM16_WriteTCNTn:
    ; Save global interrupt flag
    in      r18,SREG
    ; Disable interrupts
    cli
    ; Set TCNTn to r17:r16
    out     TCNTnH,r17
    out     TCNTnL,r16
    ; Restore global interrupt flag
    out     SREG,r18
    ret
```

C Code Example ⁽¹⁾

```
void TIM16_WriteTCNTn( unsigned int i )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Set TCNTn to i */
    TCNTn = i;
    /* Restore global interrupt flag */
    SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included.
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 assembly code example requires that the r17:r16 register pair contains the value to be written to TCNTn.

16.3.1 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.

16.4 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* (CSn2:0) bits located in the *Timer/Counter control Register B* (TCCRnB). For details on clock sources and prescaler, see [“Timer/Counter Prescaler” on page 158](#).

16.5 Prescaler Reset

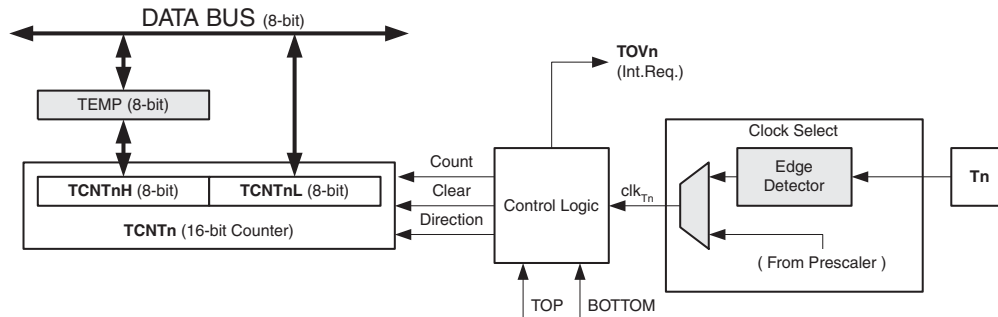
The prescaler is free running, that is, operates independently of the clock select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter’s clock select, the state of the prescaler will have implications for situations 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/Counters it is connected to.

16.6 Counter Unit

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

Figure 16-2. Counter unit block diagram



Signal description (internal signals):

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

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNTnH) containing the upper eight bits of the counter, and *Counter Low* (TCNTnL) containing the lower eight bits. The TCNTnH Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNTnH I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNTnH value when the TCNTnL is read, and TCNTnH is updated with the temporary register value when TCNTnL 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 TCNTn 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_{Tn}). The clk_{Tn} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CSn2:0). When no clock source is selected (CSn2:0 = 0) the timer is stopped. However, the TCNTn value can be accessed by the CPU, independent of whether clk_{Tn} 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 (WGMn3:0) located in the *Timer/Counter Control Registers A and B* (TCCRnA and TCCRnB). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OCnx. For more details about advanced counting sequences and waveform generation, see ["Modes of Operation" on page 127](#).

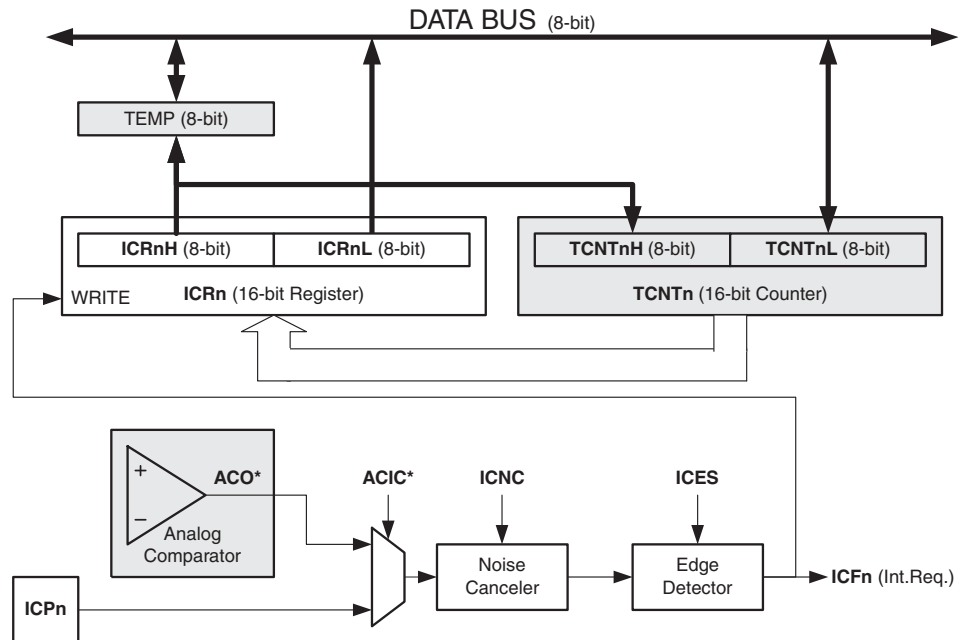
The Timer/Counter Overflow Flag (TOVn) is set according to the mode of operation selected by the WGMn3:0 bits. TOVn can be used for generating a CPU interrupt.

16.7 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 ICPn 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 16-3](#). 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 16-3. Input Capture unit block diagram



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICPn), 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 (TCNTn) is written to the *Input Capture Register* (ICRn). The *Input Capture Flag* (ICFn) is set at the same system clock as the TCNTn value is copied into ICRn Register. If enabled (ICIE_n = 1), the Input Capture Flag generates an Input Capture interrupt. The ICFn Flag is automatically cleared when the interrupt is executed. Alternatively the ICFn 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* (ICRn) is done by first reading the low byte (ICRnL) and then the high byte (ICRnH). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICRnH I/O location it will access the TEMP Register.

The ICRn Register can only be written when using a Waveform Generation mode that utilizes the ICRn Register for defining the counter's TOP value. In these cases the *Waveform Generation mode* (WGMn3:0) bits must be set before the TOP value can be written to the ICRn Register. When writing the ICRn Register the high byte must be written to the ICRnH I/O location before the low byte is written to ICRnL.

For more information on how to access the 16-bit registers refer to ["Accessing 16-bit Registers" on page 117](#).

16.7.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the *Input Capture pin* (ICPn). 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* (ICPn) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the Tn pin ([Figure 16-1 on page 116](#)). 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 ICRn to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICPn pin.

16.7.2 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* (ICNCn) bit in *Timer/Counter Control Register B* (TCCRnB). 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 ICRn Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

16.7.3 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 ICRn Register before the next event occurs, the ICRn 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 ICRn 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 ICRn Register has been read. After a change of the edge, the Input Capture Flag (ICFn) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICFn Flag is not required (if an interrupt handler is used).

16.8 Output Compare units

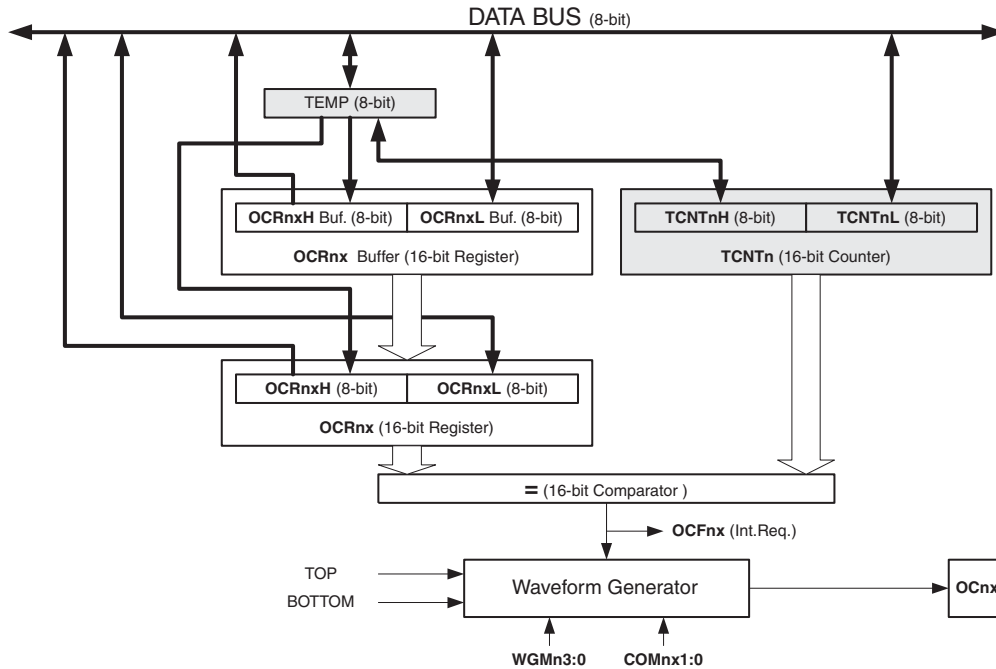
The 16-bit comparator continuously compares TCNTn with the *Output Compare Register* (OCRnx). If TCNT equals OCRnx the comparator signals a match. A match will set the *Output Compare Flag* (OCFnx) at the next timer clock cycle. If enabled (OCIE_n = 1), the Output Compare Flag generates an Output Compare interrupt. The OCFnx Flag is automatically cleared when the interrupt is executed. Alternatively the OCFnx 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* (WGMn3:0) bits and *Compare Output mode* (COMnx1:0) 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 “Modes of Operation” on page 127).

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 16-4 shows a block diagram of the Output Compare unit. The small “n” in the register and bit names indicates the device number (n = n for Timer/Counter n), and the “x” indicates Output Compare unit (A/B/C). The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

Figure 16-4. Output Compare unit, block diagram



The OCRnx 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 OCRnx 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 OCRnx Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCRnx Buffer Register, and if double buffering is disabled the CPU will access the OCRnx 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 OCRnx Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCRnxH) 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 (OCRnxL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCRnx buffer or OCRnx Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to ["Accessing 16-bit Registers" on page 117](#).

16.8.1 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* (FOCnx) bit. Forcing compare match will not set the OCFnx Flag or reload/clear the timer, but the OCnx pin will be updated as if a real compare match had occurred (the COMn1:0 bits settings define whether the OCnx pin is set, cleared or toggled).

16.8.2 Compare Match Blocking by TCNTn Write

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

16.8.3 Using the Output Compare Unit

Since writing TCNTn in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNTn when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNTn equals the OCRnx value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNTn 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 TCNTn value equal to BOTTOM when the counter is downcounting.

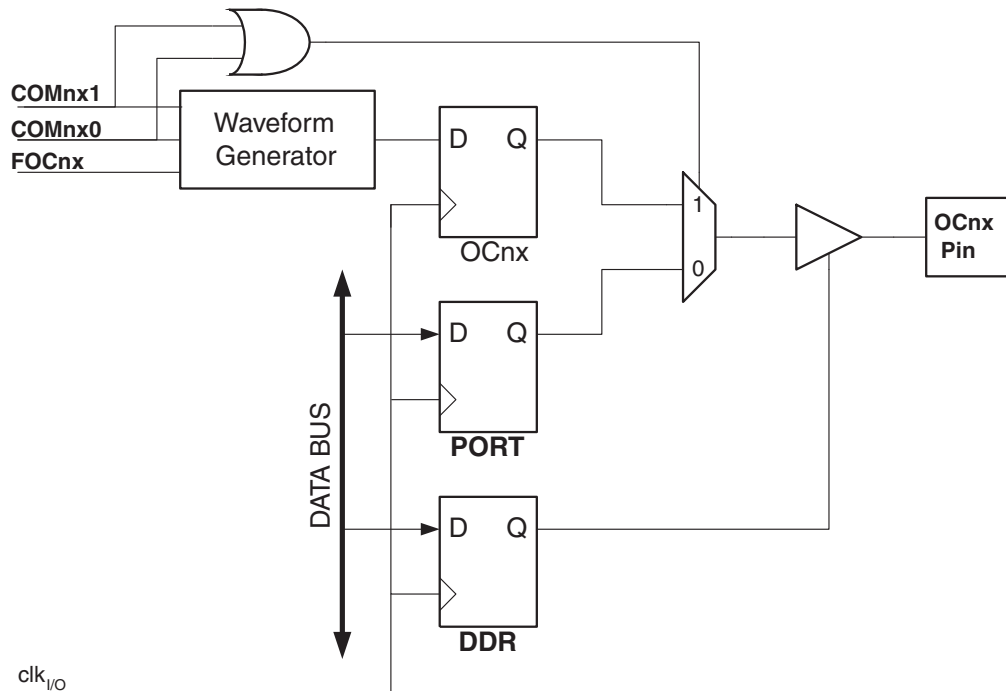
The setup of the OCnx should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OCnx value is to use the Force Output Compare (FOCnx) strobe bits in Normal mode. The OCnx Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COMnx1:0 bits are not double buffered together with the compare value. Changing the COMnx1:0 bits will take effect immediately.

16.9 Compare Match Output unit

The *Compare Output mode* (COMnx1:0) bits have two functions. The Waveform Generator uses the COMnx1:0 bits for defining the Output Compare (OCnx) state at the next compare match. Secondly the COMnx1:0 bits control the OCnx pin output source. [Figure 16-5](#) shows a simplified schematic of the logic affected by the COMnx1:0 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 COMnx1:0 bits are shown. When referring to the OCnx state, the reference is for the internal OCnx Register, not the OCnx pin. If a system reset occur, the OCnx Register is reset to “0”.

Figure 16-5. Compare Match Output unit, schematic



The general I/O port function is overridden by the Output Compare (OCnx) from the Waveform Generator if either of the COMnx1:0 bits are set. However, the OCnx 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 OCnx pin (DDR_OCnx) must be set as output before the OCnx 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 16-2](#), [Table 16-3](#) and [Table 16-4](#) for details.

The design of the Output Compare pin logic allows initialization of the OCnx state before the output is enabled. Note that some COMnx1:0 bit settings are reserved for certain modes of operation. See “Register description” on page 136.

The COMnx1:0 bits have no effect on the Input Capture unit.

16.9.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COMnx1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COMnx1:0 = 0 tells the Waveform Generator that no action on the OCnx Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to [Table 16-2 on page 136](#). For fast PWM mode refer to [Table 16-3 on page 136](#), and for phase correct and phase and frequency correct PWM refer to [Table 16-4 on page 137](#).

A change of the COMnx1:0 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 FOCnx strobe bits.

16.10 Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGMn3:0) and *Compare Output mode* (COMnx1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COMnx1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COMnx1:0 bits control whether the output should be set, cleared or toggle at a compare match (See “Compare Match Output unit” on page 126).

For detailed timing information refer to “Timer/Counter Timing diagrams” on page 134.

16.10.1 Normal mode

The simplest mode of operation is the *Normal mode* (WGMn3:0 = 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* (TOVn) will be set in the same timer clock cycle as the TCNTn becomes zero. The TOVn 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 TOVn 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.

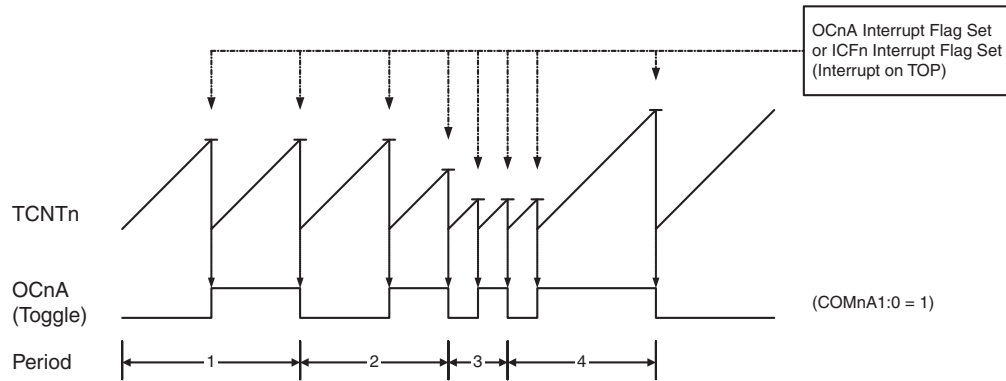
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

16.10.2 Clear Timer on Compare Match (CTC) mode

In *Clear Timer on Compare* or CTC mode (WGMn3:0 = 4 or 12), the OCRnA or ICRn Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches either the OCRnA (WGMn3:0 = 4) or the ICRn (WGMn3:0 = 12). The OCRnA or ICRn 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 16-6](#). The counter value (TCNTn) increases until a compare match occurs with either OCRnA or ICRn, and then counter (TCNTn) is cleared.

Figure 16-6. CTC mode, timing diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCFnA or ICFn 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 OCRnA or ICRn is lower than the current value of TCNTn, 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 OCRnA for defining TOP (WGMn3:0 = 15) since the OCRnA then will be double buffered.

For generating a waveform output in CTC mode, the OCnA output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COMnA1:0 = 1). The OCnA value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OCnA = 1). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk_I/O}/2$ when OCRnA is set to zero (0x0000). The waveform frequency is defined by the following equation:

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

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

As for the Normal mode of operation, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

16.10.3 Fast PWM mode

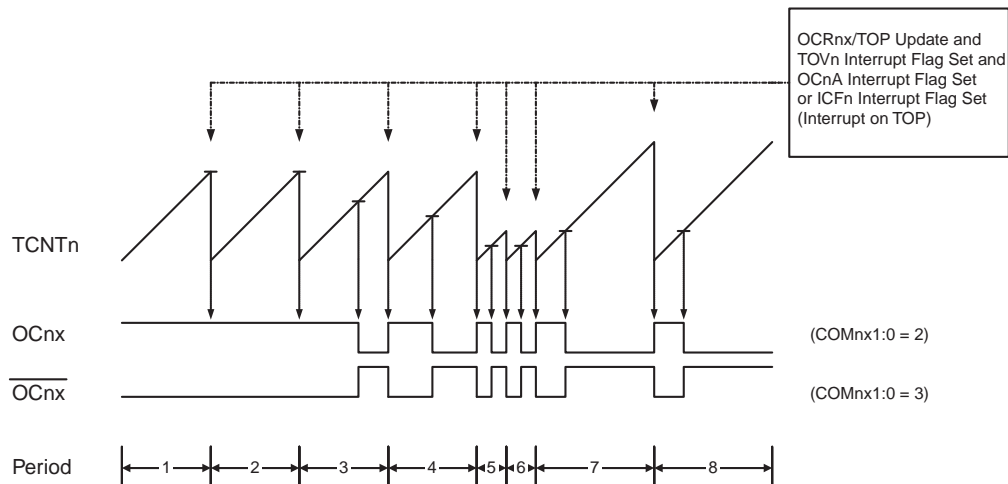
The *fast Pulse Width Modulation* or fast PWM mode (WGMn3:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the compare match between TCNTn and OCRnx, 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 ICRn or OCRnA. The minimum resolution allowed is 2-bit (ICRn or OCRnA set to 0x0003), and the maximum resolution is 16-bit (ICRn or OCRnA 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 (WGMn3:0 = 5, 6, or 7), the value in ICRn (WGMn3:0 = 14), or the value in OCRnA (WGMn3:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 16-7. The figure shows fast PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn 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 TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 16-7. Fast PWM mode, timing diagram



The Timer/Counter Overflow Flag (TOVn) is set each time the counter reaches TOP. In addition the OCnA or ICFn Flag is set at the same timer clock cycle as TOVn is set when either OCRnA or ICRn 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 TCNTn and the OCRnx. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCRnx Registers are written.

The procedure for updating ICRn differs from updating OCRnA when used for defining the TOP value. The ICRn Register is not double buffered. This means that if ICRn 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 ICRn value written is lower than the current value of TCNTn. 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 OCRnA Register however, is double buffered. This feature allows the OCRnA I/O location to be written anytime. When the OCRnA I/O location is written the value written will be put into the OCRnA Buffer Register. The OCRnA Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNTn matches TOP. The update is done at the same timer clock cycle as the TCNTn is cleared and the TOVn Flag is set.

Using the ICRn Register for defining TOP works well when using fixed TOP values. By using ICRn, the OCRnA Register is free to be used for generating a PWM output on OCnA. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCRnA 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 OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (see Table on page 136). The actual OCnx value will only be visible on the port

pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn, and clearing (or setting) the OCnx 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:

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

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

The extreme values for the OCRnx Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCRnx is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCRnx equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COMnx1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OCnA to toggle its logical level on each compare match (COMnA1:0 = 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 $f_{OCnA} = f_{clk_I/O}/2$ when OCRnA is set to zero (0x0000). This feature is similar to the OCnA toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

16.10.4 Phase Correct PWM mode

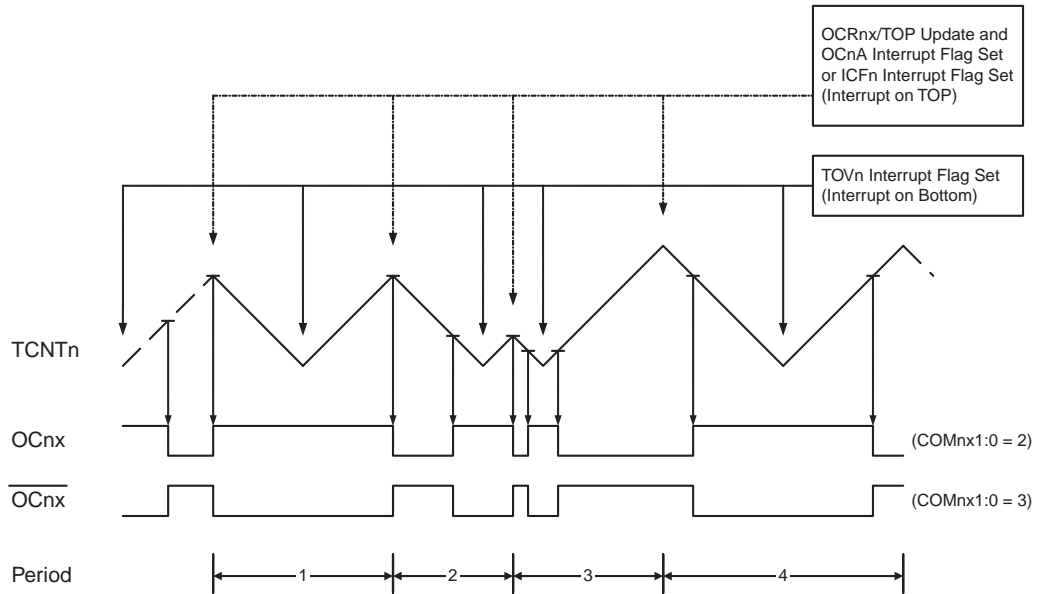
The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGMn3:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the compare match between TCNTn and OCRnx 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 ICRn or OCRnA. The minimum resolution allowed is 2-bit (ICRn or OCRnA set to 0x0003), and the maximum resolution is 16-bit (ICRn or OCRnA 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 (WGMn3:0 = 1, 2, or 3), the value in ICRn (WGMn3:0 = 10), or the value in OCRnA (WGMn3:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNTn value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on [Figure 16-8](#). The figure shows phase correct PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn 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 TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 16-8. Phase Correct PWM mode, timing diagram



The Timer/Counter Overflow Flag (TOVn) is set each time the counter reaches BOTTOM. When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag is set accordingly at the same timer clock cycle as the OCRnx 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 TCNTn and the OCRnx. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCRnx Registers are written. As the third period shown in [Figure 16-8](#) 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 OCRnx Register. Since the OCRnx 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 OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (See [Table on page 137](#)). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx Register at compare match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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

The extreme values for the OCRnx Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRnx 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.

16.10.5 Phase and Frequency Correct PWM mode

The *phase and frequency correct Pulse Width Modulation*, or phase and frequency correct PWM mode (WGMn3:0 = 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 (OCnx) is cleared on the compare match between TCNTn and OCRnx 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.

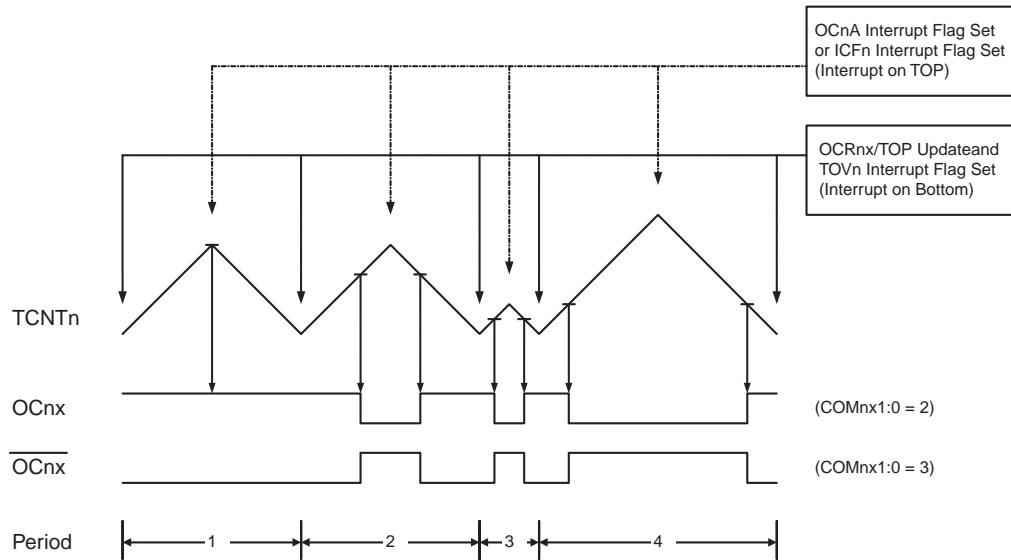
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCRnx Register is updated by the OCRnx Buffer Register, (see [Figure 16-8](#) and [Figure 16-9](#)).

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

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

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICRn (WGMn3:0 = 8), or the value in OCRnA (WGMn3:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNTn 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 16-9](#). The figure shows phase and frequency correct PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn 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 TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 16-9. Phase and Frequency Correct PWM mode, timing diagram



The Timer/Counter Overflow Flag (TOVn) is set at the same timer clock cycle as the OCRnx Registers are updated with the double buffer value (at BOTTOM). When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag set when TCNTn 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 TCNTn and the OCRnx.

As [Figure 16-9](#) shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCRnx 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 ICRn Register for defining TOP works well when using fixed TOP values. By using ICRn, the OCRnA Register is free to be used for generating a PWM output on OCnA. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCRnA 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 OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (See [Table 16-4 on page 137](#)). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx Register at compare match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

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

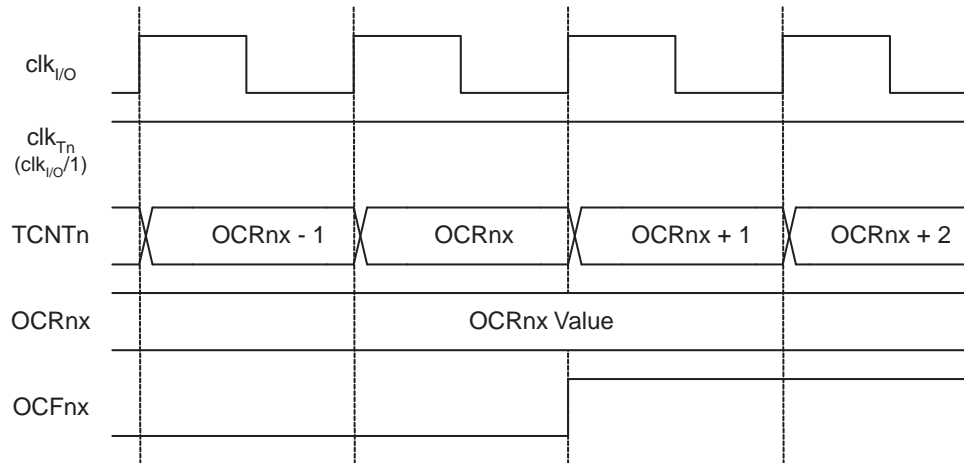
The extreme values for the OCRnx Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRnx is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted 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.

16.11 Timer/Counter Timing diagrams

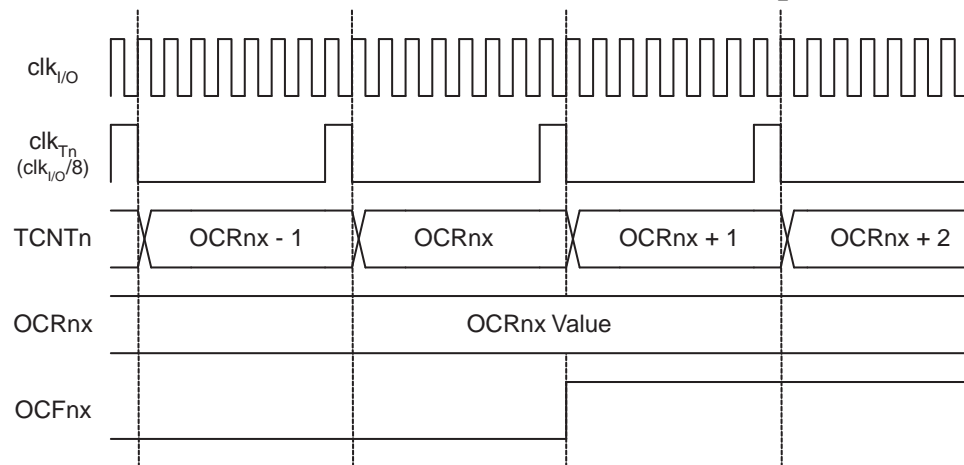
The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCRnx Register is updated with the OCRnx buffer value (only for modes utilizing double buffering). [Figure 16-10](#) shows a timing diagram for the setting of OCFnx.

Figure 16-10. Timer/Counter Timing diagram, setting of OCFnx, no prescaling



[Figure 16-11](#) shows the same timing data, but with the prescaler enabled.

Figure 16-11. Timer/Counter Timing diagram, setting of OCFnx, with prescaler ($f_{clk_{I/O}}/8$)



[Figure 16-12](#) shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCRnx 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 TOVn Flag at BOTTOM.

Figure 16-12. Timer/Counter Timing diagram, no prescaling

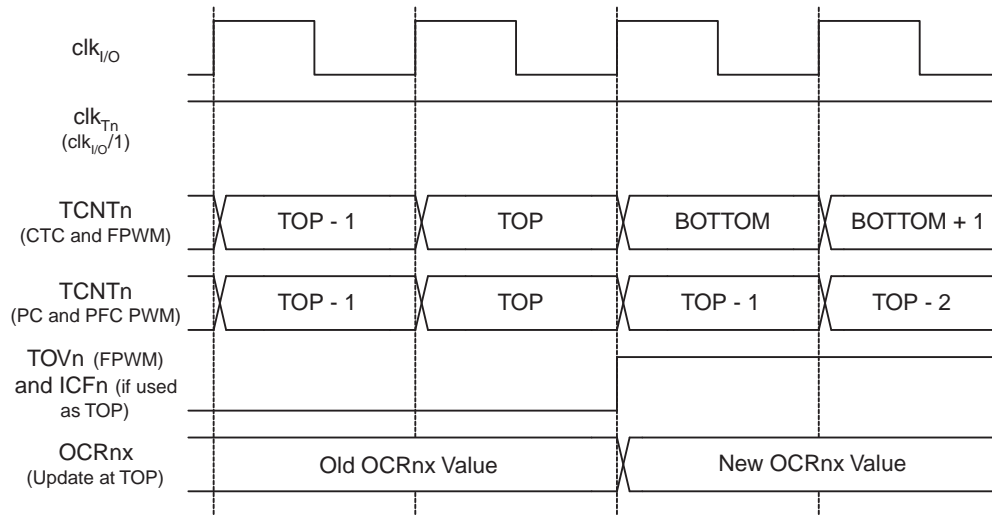
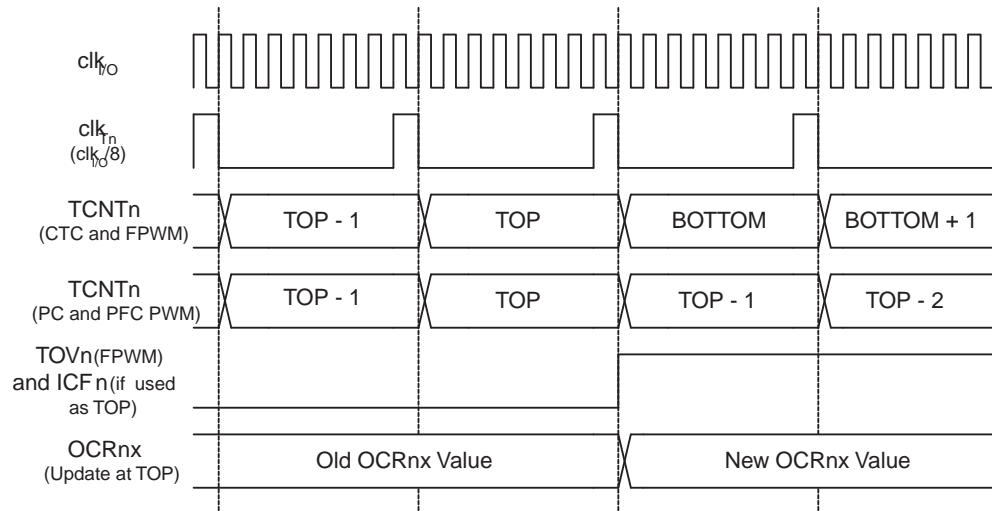


Figure 16-13 shows the same timing data, but with the prescaler enabled.

Figure 16-13. Timer/Counter Timing diagram, with prescaler ($f_{clk_I/O}/8$)



16.12 Register description

16.12.1 TCCRnA – Timer/Counter n Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COMnA1	COMnA0	COMnB1	COMnB0	–	–	WGMn1	WGMn0	TCCRnA
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 – COMnA1:0: Compare Output Mode for Channel A**

- **Bit 5:4 – COMnB1:0: Compare Output Mode for Channel B**

The COMnA1:0 and COMnB1:0 control the Output Compare pins (OCnA and OCnB respectively) behavior. If one or both of the COMnA1:0 bits are written to one, the OCnA output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COMnB1:0 bit are written to one, the OCnB 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 OCnA or OCnB pin must be set in order to enable the output driver.

When the OCnA or OCnB is connected to the pin, the function of the COMnx1:0 bits is dependent of the WGMn3:0 bits setting. [Table 16-2 on page 136](#) shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to a Normal or a CTC mode (non-PWM).

Table 16-2. Compare Output mode, non-PWM

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	Toggle OCnA/OCnB on Compare Match.
1	0	Clear OCnA/OCnB on Compare Match (Set output to low level).
1	1	Set OCnA/OCnB on Compare Match (Set output to high level).

[Table 16-3 on page 136](#) shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the fast PWM mode.

Table 16-3. Compare Output mode, fast PWM ⁽¹⁾

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	WGMn3:0 = 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 OCnA/OCnB on Compare Match, set OCnA/OCnB at BOTTOM (non-inverting mode)
1	1	Set OCnA/OCnB on Compare Match, clear OCnA/OCnB at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCRnA/OCRnB equals TOP and COMnA1/COMnB1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. [See Section “16.10.3” on page 128](#) for more details.

Table 16-4 on page 137 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 16-4. Compare Output mode, phase correct and phase and frequency correct PWM ⁽¹⁾

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	WGMn3:0 = 9 or 11: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OCnA/OCnB on Compare Match when up-counting. Set OCnA/OCnB on Compare Match when downcounting.
1	1	Set OCnA/OCnB on Compare Match when up-counting. Clear OCnA/OCnB on Compare Match when downcounting.

Note: 1. A special case occurs when OCRnA/OCRnB equals TOP and COMnA1/COMnB1 is set. See Section “16.10.4” on page 130 for more details.

• Bit 1:0 – WGMn1:0: Waveform Generation Mode

Combined with the WGMn3:2 bits found in the TCCRnB 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 16-5 on page 138. 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 “16.10” on page 127).

Table 16-5. Waveform Generation mode bit description ⁽¹⁾

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter mode of operation	TOP	Update of OCRnX at	TOVn 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	OCRnA	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	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

16.12.2 TCCRnB – Timer/Counter n Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNCn	ICESn	–	WGMn3	WGMn2	CSn2	CSn1	CSn0	TCCRnB
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 – ICNCn: 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 (ICPn) is filtered. The filter function requires four successive equal valued samples of the ICPn pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

• Bit 6 – ICESn: Input Capture Edge Select

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

When a capture is triggered according to the ICESn setting, the counter value is copied into the Input Capture Register (ICRn). The event will also set the Input Capture Flag (ICFn), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR_n is used as TOP value (see description of the WGM_n3:0 bits located in the TCCR_nA and the TCCR_nB Register), the ICP_n is disconnected and consequently the Input Capture function is disabled.

- **Bit 5 – Reserved**

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

- **Bit 4:3 – WGM_n3:2: Waveform Generation Mode**

See “TCCR_nA – Timer/Counter n Control Register A” description on page 136.

- **Bit 2:0 – CS_n2:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 16-10 and Figure 16-11.

Table 16-6. Clock Select bit description

CS _n 2	CS _n 1	CS _n 0	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{I/O} /1 (No prescaling)
0	1	0	clk _{I/O} /8 (From prescaler)
0	1	1	clk _{I/O} /64 (From prescaler)
1	0	0	clk _{I/O} /256 (From prescaler)
1	0	1	clk _{I/O} /1024 (From prescaler)
1	1	0	External clock source on T _n pin. Clock on falling edge.
1	1	1	External clock source on T _n pin. Clock on rising edge.

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

16.12.3 TCCR_nC – Timer/Counter n Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC _n A	FOC _n B	–	–	–	–	–	–	TCCR _n C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC_nA: Force Output Compare for Channel A**

- **Bit 6 – FOC_nB: Force Output Compare for Channel B**

The FOC_nA/FOC_nB bits are only active when the WGM_n3:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR_nA is written when operating in a PWM mode. When writing a logical one to the FOC_nA/FOC_nB bit, an immediate compare match is forced on the Waveform Generation unit. The OC_nA/OC_nB output is changed according to its COM_nx1:0 bits setting. Note that the FOC_nA/FOC_nB bits are implemented as strobes. Therefore it is the value present in the COM_nx1:0 bits that determine the effect of the forced compare.

A FOC_nA/FOC_nB strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR_nA as TOP.

The FOC_nA/FOC_nB bits are always read as zero.

16.12.4 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 [“Accessing 16-bit Registers” on page 117](#).

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

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

16.12.5 TCNT3H and TCNT3L –Timer/Counter3

Bit	7	6	5	4	3	2	1	0	
(0x95)	TCNT3[15:8]								TCNT3H
(0x94)	TCNT3[7:0]								TCNT3L
Read/Write	R/W	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 (TCNT3H and TCNT3L, combined TCNT3) 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 [“Accessing 16-bit Registers” on page 117](#).

Modifying the counter (TCNT3) while the counter is running introduces a risk of missing a compare match between TCNT3 and one of the OCRnx Registers.

Writing to the TCNT3 Register blocks (removes) the compare match on the following timer clock for all compare units.

16.12.6 OCR1AH and OCR1AL – Output Compare Register1 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	

16.12.7 OCR1BH and OCR1BL – Output Compare Register1 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 OCnx 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 [“Accessing 16-bit Registers” on page 117](#).

16.12.8 OCR3AH and OCR3AL – Output Compare Register3 A

Bit	7	6	5	4	3	2	1	0	
(0x99)	OCR3A[15:8]								OCR3AH
(0x98)	OCR3A[7:0]								OCR3AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

16.12.9 OCR3BH and OCR3BL – Output Compare Register3 B

Bit	7	6	5	4	3	2	1	0	
(0x9B)	OCR3B[15:8]								OCR3BH
(0x9A)	OCR3B[7:0]								OCR3BL
Read/Write	R/W	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 (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnx 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 [“Accessing 16-bit Registers” on page 117](#).

16.12.10 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 ICPn 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 [“Accessing 16-bit Registers” on page 117](#).

16.12.11 ICR3H and ICR3L – Input Capture Register 3

Bit	7	6	5	4	3	2	1	0	
(0x97)	ICR3[15:8]								ICR3H
(0x96)	ICR3[7:0]								ICR3L
Read/Write	R/W	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 (TCNT3) value each time an event occurs on the ICPn pin (or optionally on the Analog Comparator output for Timer/Counter3). 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 “Accessing 16-bit Registers” on page 117.

16.12.12TIMSK1 – 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 – Reserved

These bits are unused 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 “Interrupts” on page 69) is executed when the ICF1 Flag, located in TIFR1, is set.

• Bit 4:3 – Reserved

These bits are unused 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 “Interrupts” on page 69) 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 “Interrupts” on page 69) 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 “11.3” on page 63) is executed when the TOV1 Flag, located in TIFR1, is set.

16.12.13TIMSK3 – Timer/Counter3 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x71)	–	–	ICIE3	–	–	OCIE3B	OCIE3A	TOIE3	TIMSK3
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7:6 – Reserved

These bits are unused and will always read as zero.

- **Bit 5 – ICIE3: Timer/Counter3, 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 “Interrupts” on page 69) is executed when the ICF3 Flag, located in TIFR3, is set.

- **Bit 4:3 – Reserved**

These bits are unused and will always read as zero.

- **Bit 2 – OCIE3B: Timer/Counter3, 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/Counter3 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 69) is executed when the OCF3B Flag, located in TIFR3, is set.

- **Bit 1 – OCIE3A: Timer/Counter3, 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/Counter3 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 69) is executed when the OCF3A Flag, located in TIFR3, is set.

- **Bit 0 – TOIE3: Timer/Counter3, 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/Counter3 Overflow interrupt is enabled. The corresponding Interrupt Vector (See “Watchdog Timer” on page 63) is executed when the TOV3 Flag, located in TIFR3, is set.

16.12.14 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 – Reserved**

These bits are unused 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 WGMn3:0 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 – Reserved**

These bits are unused 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 WGMn3:0 bits setting. In Normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to [Table 16-5 on page 138](#) for the TOV1 Flag behavior when using another WGMn3:0 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.

16.12.15TIFR3 – Timer/Counter3 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	–	–	ICF3	–	–	OCF3B	OCF3A	TOV3	TIFR3
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Reserved**

These bits are unused and will always read as zero.

- **Bit 5 – ICF3: Timer/Counter3, Input Capture Flag**

This flag is set when a capture event occurs on the ICP3 pin. When the Input Capture Register (ICR1) is set by the WGMn3:0 to be used as the TOP value, the ICF3 Flag is set when the counter reaches the TOP value.

ICF3 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF3 can be cleared by writing a logic one to its bit location.

- **Bit 4:3 – Reserved**

These bits are unused and will always read as zero.

- **Bit 2 – OCF3B: Timer/Counter3, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register B (OCR3B).

Note that a Forced Output Compare (FOC3B) strobe will not set the OCF3B Flag.

OCF3B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF3B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF3A: Timer/Counter3, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register A (OCR3A).

Note that a Forced Output Compare (FOC3A) strobe will not set the OCF3A Flag.

OCF3A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF3A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV3: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGMn3:0 bits setting. In Normal and CTC modes, the TOV3 Flag is set when the timer overflows. Refer to [Table 16-5 on page 138](#) for the TOV3 Flag behavior when using another WGMn3:0 bit setting.

TOV3 is automatically cleared when the Timer/Counter3 Overflow Interrupt Vector is executed. Alternatively, TOV3 can be cleared by writing a logic one to its bit location.

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

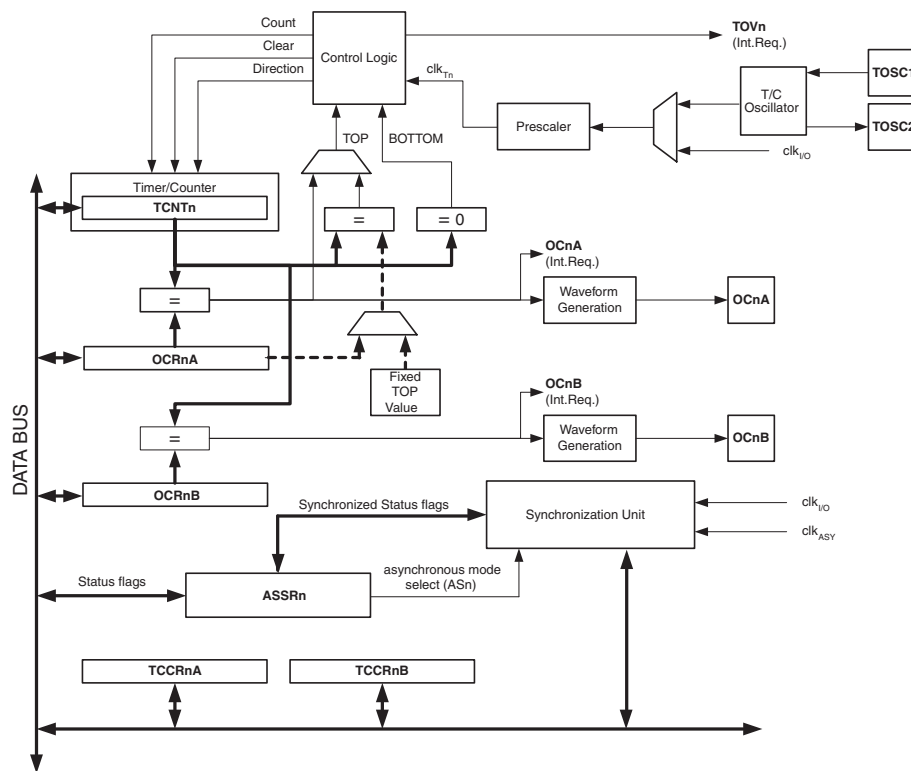
17.2 Overview

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 16-12](#). For the actual placement of I/O pins, see ["Pin configurations" on page 11](#). 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 ["Register description" on page 159](#).

The Power Reduction Timer/Counter2 bit, PRTIM2, in ["PRR0 – Power Reduction Register 0" on page 56](#) 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 (abbreviated to 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_{T2}).

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 [“Output Compare unit” on page 148](#) 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, that is, 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

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.

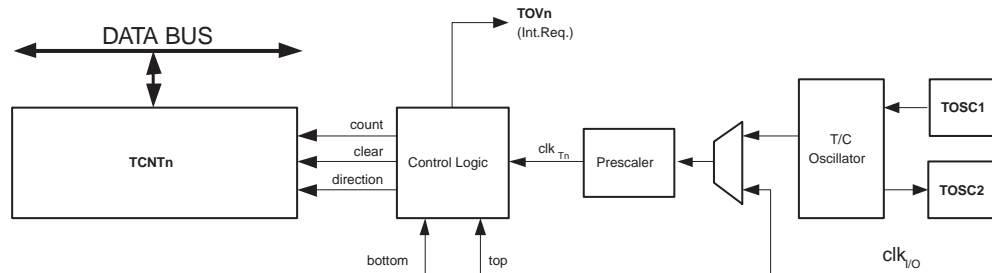
17.3 Timer/Counter clock sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{T2} is by default equal to the MCU clock, $\text{clk}_{I/O}$. 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 [“ASSR – Asynchronous Status Register” on page 163](#). For details on clock sources and prescaler, see [“Timer/Counter Prescaler” on page 158](#).

17.4 Counter unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. [Figure 17-2](#) 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).
clk_{Tn}	Timer/Counter clock, referred to as clk _{T2} 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_{T2}). clk_{T2} 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_{T2} 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 ["Modes of operation" on page 151](#).

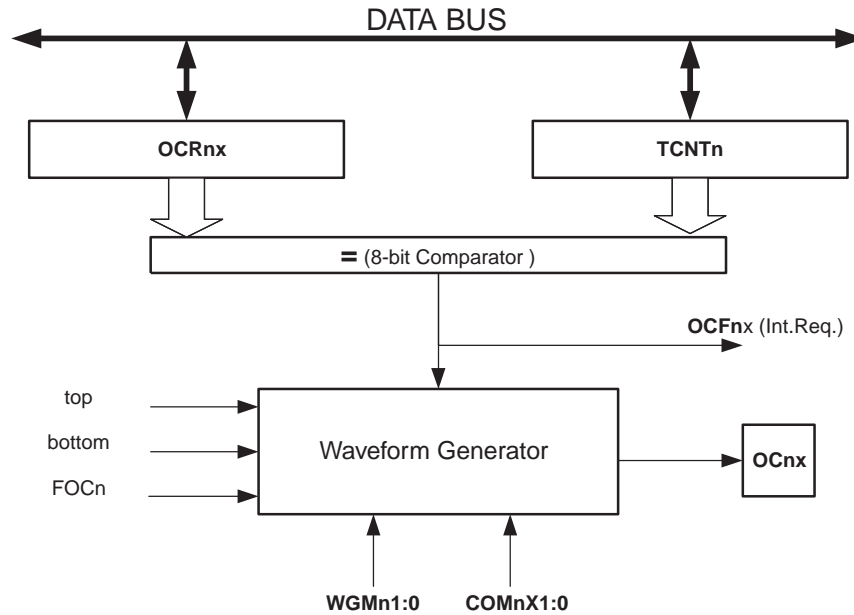
The Timer/Counter Overflow Flag (TOV2) is set according to the mode of operation selected by the WGM22:0 bits. TOV2 can be used for generating a CPU interrupt.

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 (see ["Modes of operation" on page 151](#)).

[Figure 16-10 on page 134](#) 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.

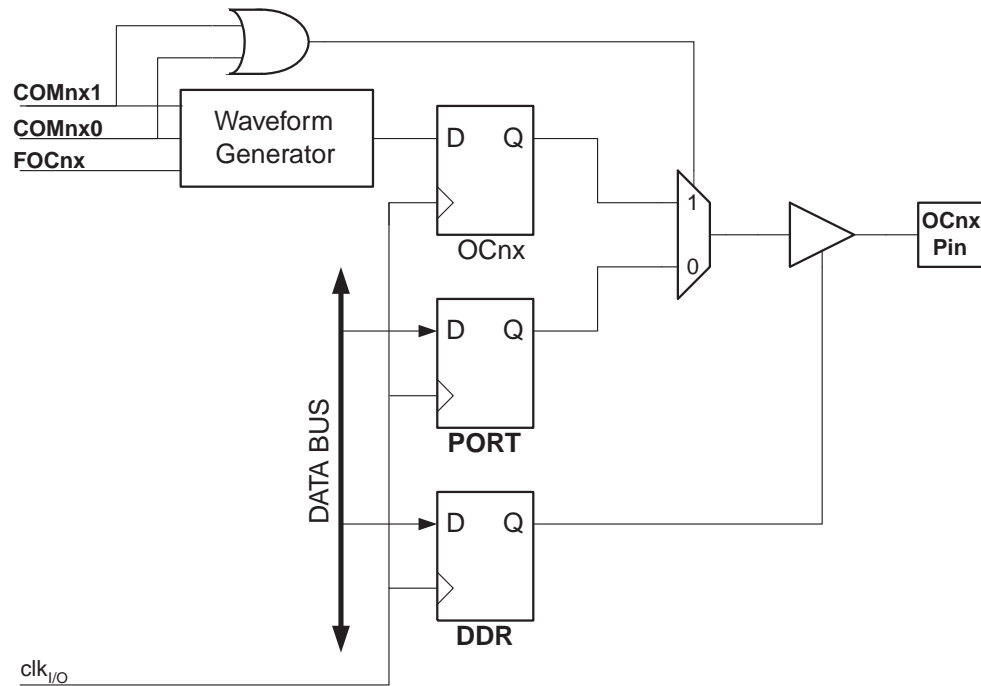
The setup of the OC2x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2x value is to use the Force Output Compare (FOC2x) strobe bit in Normal mode. The OC2x Register keeps its value even when changing between Waveform Generation 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 ["Register description" on page 159](#).

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 160](#). For fast PWM mode, refer to [Table 17-6 on page 160](#), and for phase correct PWM refer to [Table 17-7 on page 161](#).

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, that is, 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 “Compare Match Output unit” on page 150).

For detailed timing information refer to “Timer/Counter Timing diagrams” on page 155.

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.

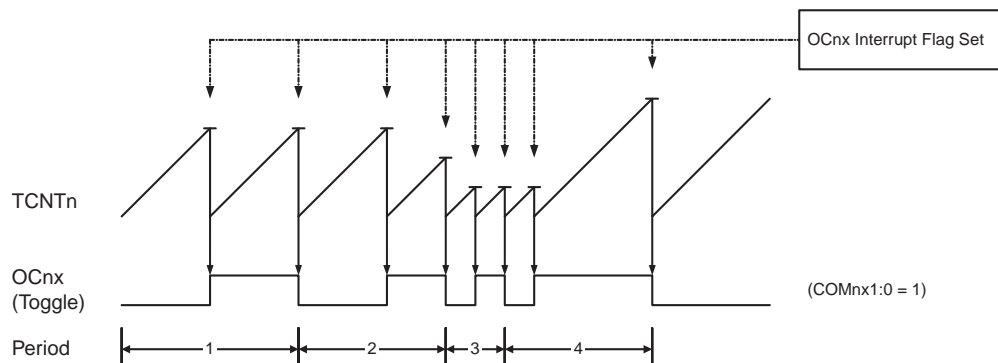
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

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 Table 17-5 on page 151. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and 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:

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

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

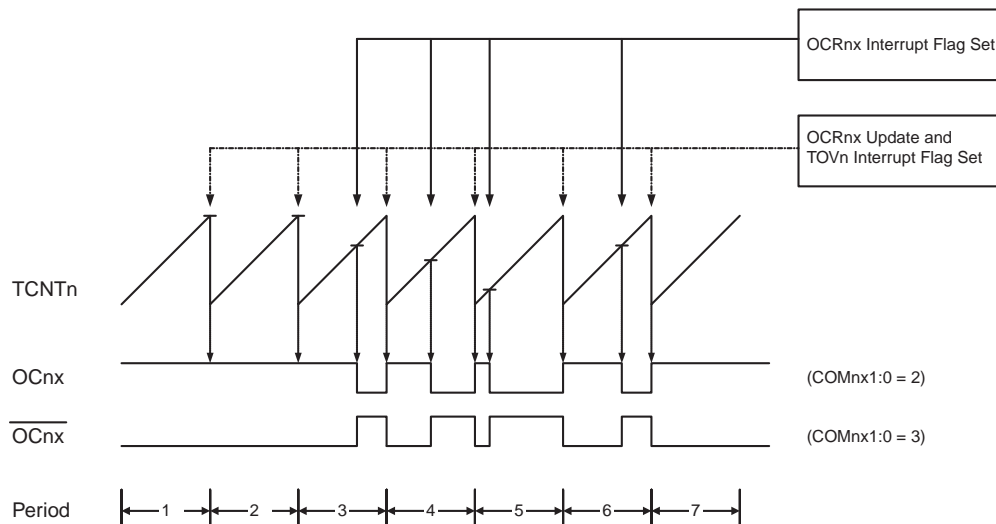
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

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 WGM22:0 = 3, and OCR2A when WGM22:0 = 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 on page 152](#). 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 159](#)). 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:

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

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.)

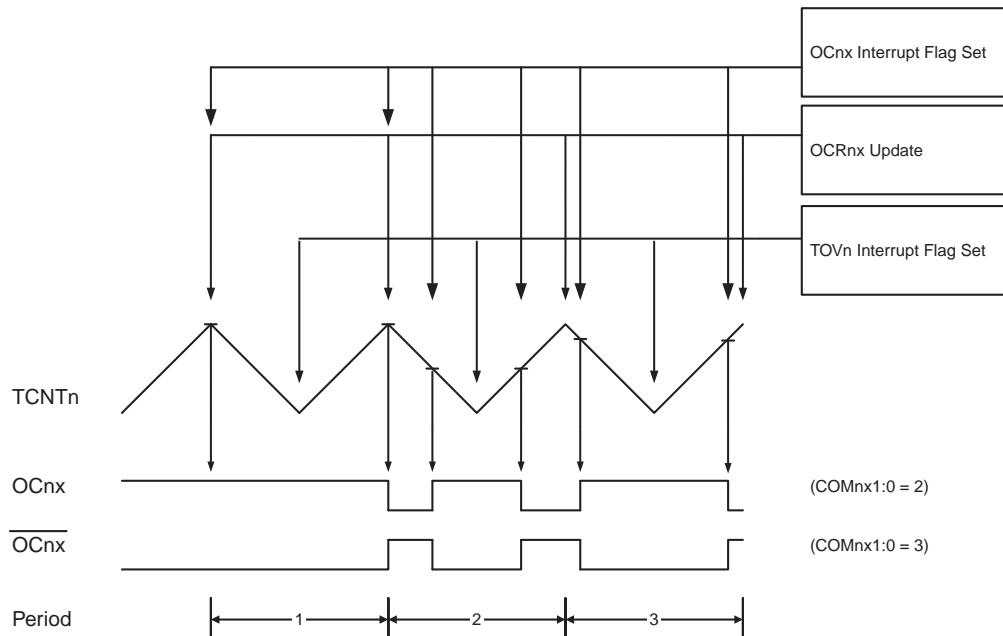
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2x to toggle its logical level on each compare match (COM2x1:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk_I/O}/2$ when OCR2A is set to zero. This feature is similar to 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 WGM22:0 = 1, and OCR2A when WGM22:0 = 5. 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 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 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 WGM2:0 = 7 (See [Table 17-4 on page 160](#)). 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:

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

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.

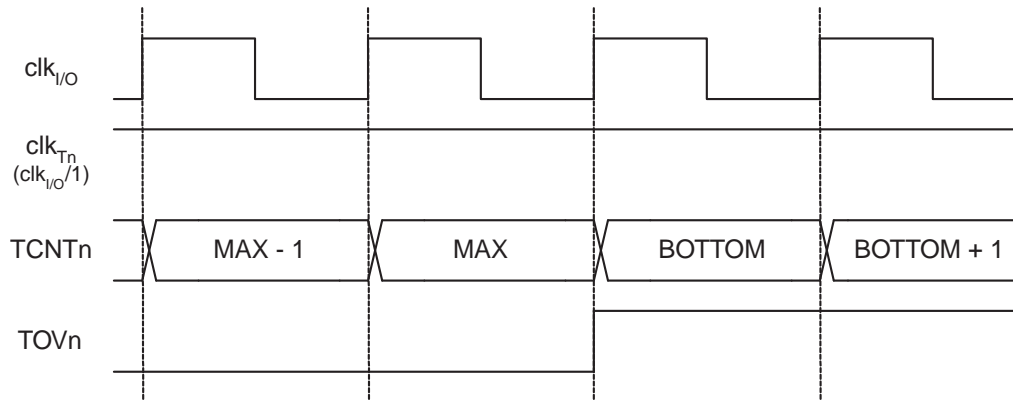
At the very start of period 2 in [Figure 17-7 on page 154](#) OCnx has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR2A changes its value from MAX, like in [Figure 17-7 on page 154](#). 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

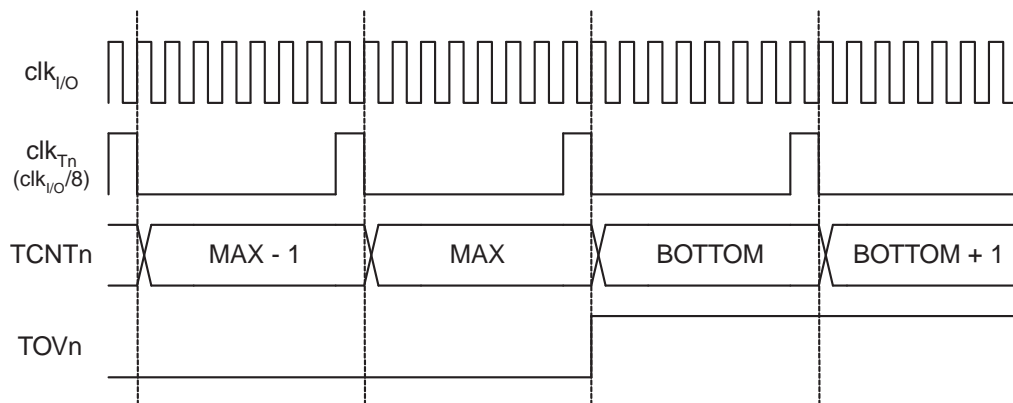
The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $\text{clk}_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. [Figure 17-8 on page 155](#) 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 on page 155](#) shows the same timing data, but with the prescaler enabled.

Figure 17-9. Timer/Counter Timing diagram, with prescaler ($f_{\text{clk}_{I/O}}/8$)



[Figure 17-10 on page 156](#) shows the setting of OCF2A in all modes except CTC mode.

Figure 17-10. Timer/Counter Timing diagram, setting of OCF2A, with prescaler ($f_{clk_I/O}/8$)

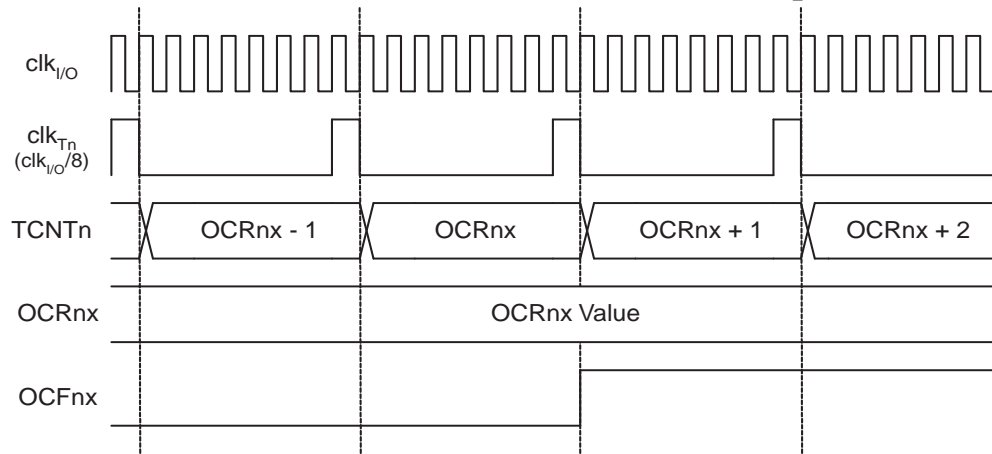
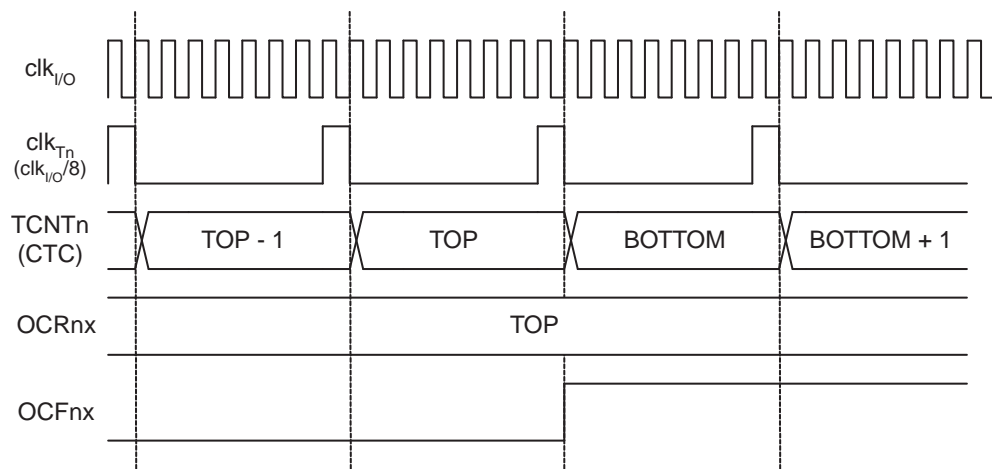


Figure 17-11 on page 156 shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

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.

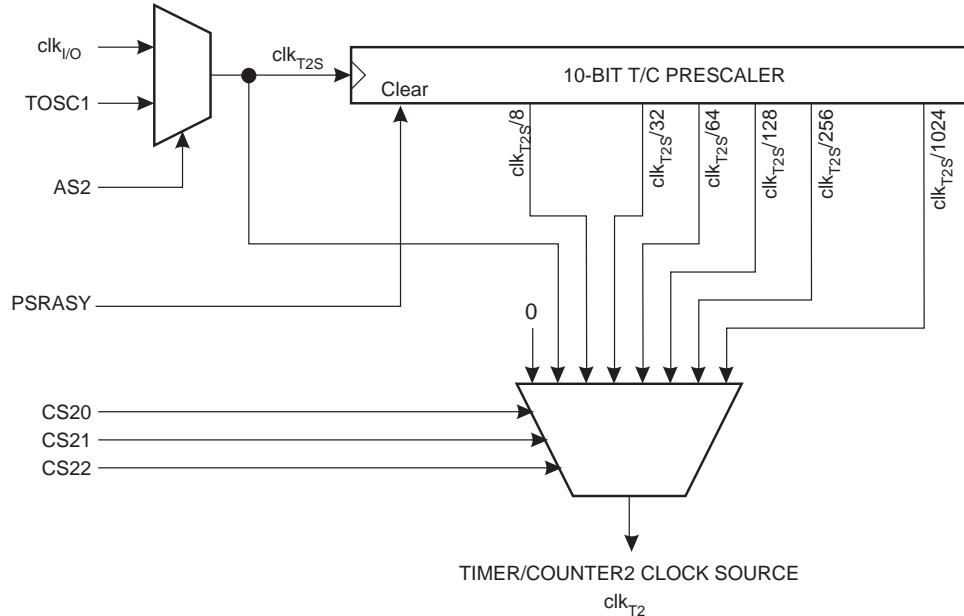
- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2x, and TCCR2x might be corrupted. A safe procedure for switching clock source is:
 1. Disable the Timer/Counter2 interrupts by clearing OCIE2x and TOIE2.
 2. Select clock source by setting AS2 as appropriate.
 3. Write new values to TCNT2, OCR2x, and TCCR2x.
 4. To switch to asynchronous operation: Wait for TCN2UB, OCR2xUB, and TCR2xUB.
 5. Clear the Timer/Counter2 Interrupt Flags.
 6. 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: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. 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:
 1. Write a value to TCCR2x, TCNT2, or OCR2x.
 2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
 3. 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:
 1. Write any value to either of the registers OCR2x or TCCR2x.
 2. Wait for the corresponding Update Busy Flag to be cleared.
 3. 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. By setting the EXCLK bit in the ASSR a 32kHz external clock can be applied. See ["ASSR – Asynchronous Status Register" on page 163](#) for details.

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, OC2A 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 ["Fast PWM mode" on page 152](#) 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 ["Phase Correct PWM mode" on page 153](#) 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 ["Fast PWM mode" on page 152](#) 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 "Phase Correct PWM mode" on page 153 for more details.

• Bits 3:2 – Reserved

These bits are reserved 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 "Modes of operation" on page 151).

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	TCCR2B
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 – 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 – Reserved

These bits are reserved and will always read as zero.

• Bit 3 – WGM22: Waveform Generation Mode

See the description in the ["TCCR2A – Timer/Counter Control Register A" on page 159](#).

• 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 on page 162](#).

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)

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 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 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.

17.11.7 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, that is, 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, that is, 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, that is, when the TOV2 bit is set in the Timer/Counter2 Interrupt Flag Register – TIFR2.

17.11.8 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.

17.11.9 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 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/Counters 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/Counters start counting simultaneously.

• 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 the “Bit 7 – TSM: Timer/Counter Synchronization Mode” on this page for a description of the Timer/Counter Synchronization mode.

• Bit 0 – PSRSYNC: Prescaler Reset

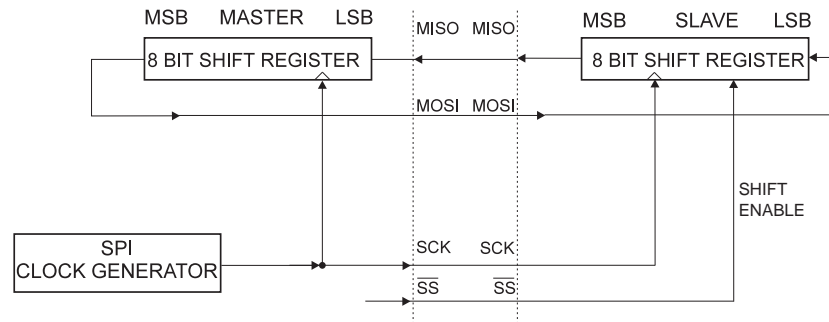
When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

The interconnection between Master and Slave CPUs with SPI is shown in [Figure 18-2](#). The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select \overline{SS} pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, \overline{SS} , line.

When configured as a Master, the SPI interface has no automatic control of the \overline{SS} line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, \overline{SS} line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the \overline{SS} pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the \overline{SS} pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 18-2. SPI Master-slave interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be:

Low period: longer than two CPU clock cycles.

High period: longer than two CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and \overline{SS} pins is overridden according to [Table 18-1](#). For more details on automatic port overrides, refer to ["Alternate Port Functions" on page 85](#).

Table 18-1. SPI pin overrides ⁽¹⁾

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
\overline{SS}	User Defined	Input

Note: 1. See ["Alternate Functions of Port B" on page 88](#) for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. For example, if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.

Assembly Code Example ⁽¹⁾

```

SPI_MasterInit:
    ; Set MOSI and SCK output, all others input
    ldi    r17, (1<<DD_MOSI) | (1<<DD_SCK)
    out    DDR_SPI, r17
    ; Enable SPI, Master, set clock rate fck/16
    ldi    r17, (1<<SPE) | (1<<MSTR) | (1<<SPR0)
    out    SPCR, r17
    ret

SPI_MasterTransmit:
    ; Start transmission of data (r16)
    out    SPDR, r16
Wait_Transmit:
    ; Wait for transmission complete
    sbis   SPSR, SPIF
    rjmp   Wait_Transmit
    ret
    
```

C Code Example ⁽¹⁾

```

void SPI_MasterInit(void)
{
    /* Set MOSI and SCK output, all others input */
    DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK);
    /* Enable SPI, Master, set clock rate fck/16 */
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
    /* Start transmission */
    SPDR = cData;
    /* Wait for transmission complete */
    while (!(SPSR & (1<<SPIF)))
        ;
}
    
```

Note: 1. [See “About code examples” on page 17.](#)

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example ⁽¹⁾

```
SPI_SlaveInit:
    ; Set MISO output, all others input
    ldi    r17, (1<<DD_MISO)
    out    DDR_SPI, r17
    ; Enable SPI
    ldi    r17, (1<<SPE)
    out    SPCR, r17
    ret

SPI_SlaveReceive:
    ; Wait for reception complete
    sbis   SPSR, SPIF
    rjmp   SPI_SlaveReceive
    ; Read received data and return
    in     r16, SPDR
    ret
```

C Code Example ⁽¹⁾

```
void SPI_SlaveInit(void)
{
    /* Set MISO output, all others input */
    DDR_SPI = (1<<DD_MISO);
    /* Enable SPI */
    SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
    /* Wait for reception complete */
    while(!(SPSR & (1<<SPIF)))
        ;
    /* Return Data Register */
    return SPDR;
}
```

Note: 1. [See “About code examples” on page 17.](#)

18.3 \overline{SS} pin functionality

18.3.1 Slave mode

When the SPI is configured as a Slave, the Slave Select (\overline{SS}) pin is always input. When \overline{SS} is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When \overline{SS} is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the \overline{SS} pin is driven high.

The \overline{SS} pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the \overline{SS} pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

18.3.2 Master mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the \overline{SS} pin.

If \overline{SS} is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the \overline{SS} pin of the SPI Slave.

If \overline{SS} is configured as an input, it must be held high to ensure Master SPI operation. If the \overline{SS} pin is driven low by peripheral circuitry when the SPI is configured as a Master with the \overline{SS} pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that \overline{SS} is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

18.4 Data modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in [Figure 18-3 on page 171](#) and [Figure 18-4 on page 171](#). Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing [Table 18-3 on page 172](#) and [Table 18-4 on page 172](#), as done in [Table 18-2 on page 171](#)

Table 18-2. SPI modes

SPI Mode	Conditions	Leading Edge	Trailing Edge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

Figure 18-3. SPI transfer format with CPHA = 0

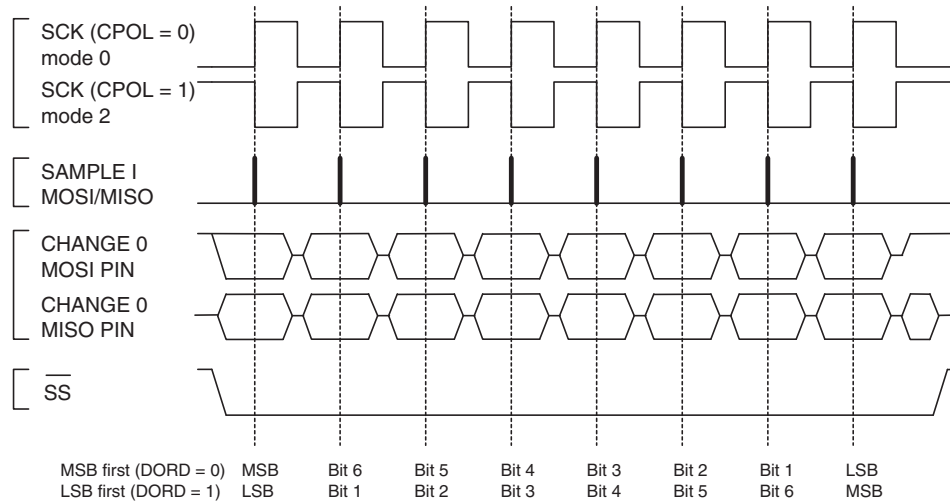
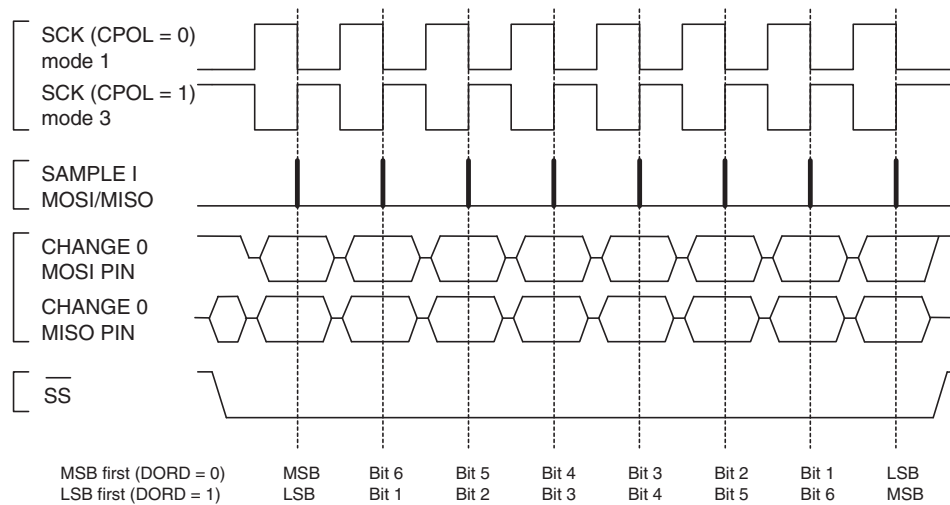


Figure 18-4. SPI transfer format with CPHA = 1



18.5 Register description

18.5.1 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the Global Interrupt Enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If \overline{SS} is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to [Figure 18-3](#) and [Figure 18-4](#) for an example. The CPOL functionality is summarized below:

Table 18-3. CPOL functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to [Figure 18-3](#) and [Figure 18-4](#) for an example. The CPOL functionality is summarized below:

Table 18-4. CPHA functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1:0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the following table:

Table 18-5. Relationship between SCK and the oscillator frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

18.5.2 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If \overline{SS} is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL: Write COLLision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5:1 – Reserved**

These bits are reserved and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see [Table 18-5](#)). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only ensured to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P is also used for program memory and EEPROM downloading or uploading. See [page 309](#) for serial programming and verification.

18.5.3 SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

19. USART

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 USART1 and USART0

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P has two USART's, USART0 and USART1.

The functionality for all USART's is described below, most register and bit references in this section are written in general form. A lower case "n" replaces the USART number.

USART0 and USART1 have different I/O registers as shown in ["Register summary" on page 636](#).

19.3 Overview

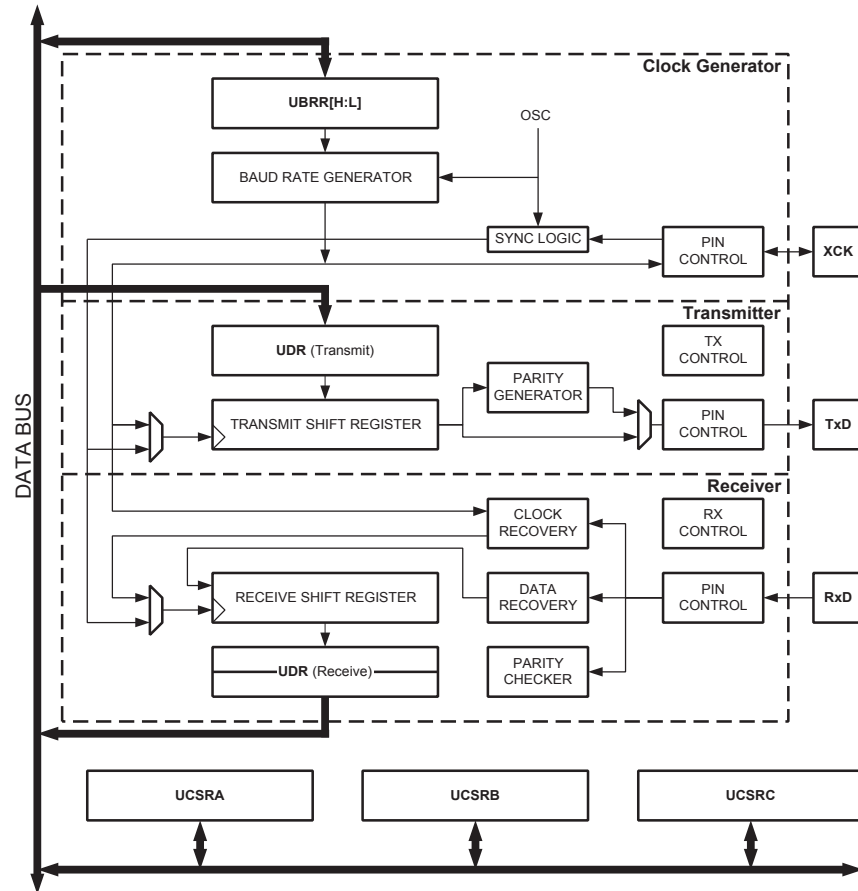
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device.

A simplified block diagram of the USART Transmitter is shown in [Figure 19-1 on page 176](#). CPU accessible I/O Registers and I/O pins are shown in bold.

The Power Reduction USART0 bit, PRUSART0, in ["PRR0 – Power Reduction Register 0" on page 56](#) must be disabled by writing a logical zero to it.

The Power Reduction USART1 bit, PRUSART1, in ["PRR1 – Power Reduction Register 1" on page 57](#) must be disabled by writing a logical zero to it.

Figure 19-1. USART block diagram ⁽¹⁾



Note: 1. See [Figure 1-1 on page 11](#) and ["Alternate Port Functions" on page 85](#) for USART pin placement.

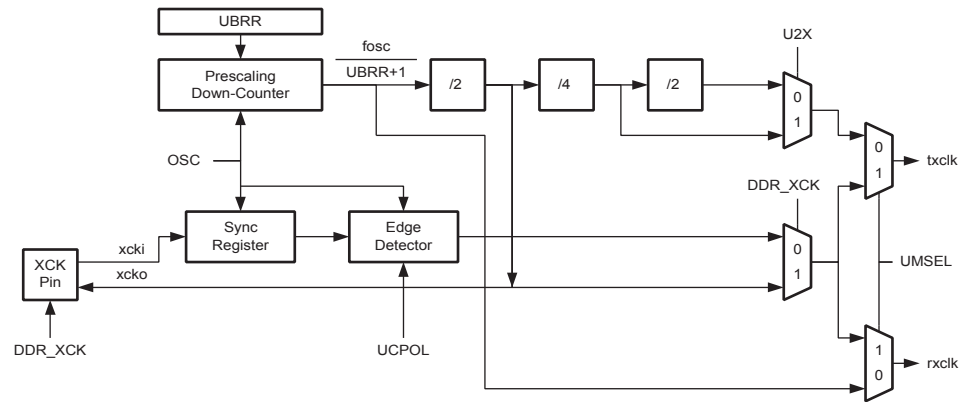
The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave 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.

19.4 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The USARTn 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.
f_{osc}	XTAL pin frequency (System Clock).

19.4.1 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 on page 177](#).

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 on page 178](#) 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 UBRR 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	$BAUD = \frac{f_{OSC}}{2(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

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

BAUDBaud rate (in bits per second, bps).

f_{osc}System Oscillator clock frequency.

UBRRnContents 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 198](#).

19.4.2 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.

19.4.3 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 177](#) for details.

External clock input from the XCKn pin is sampled by a synchronization register to minimize the chance of metastability. 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:

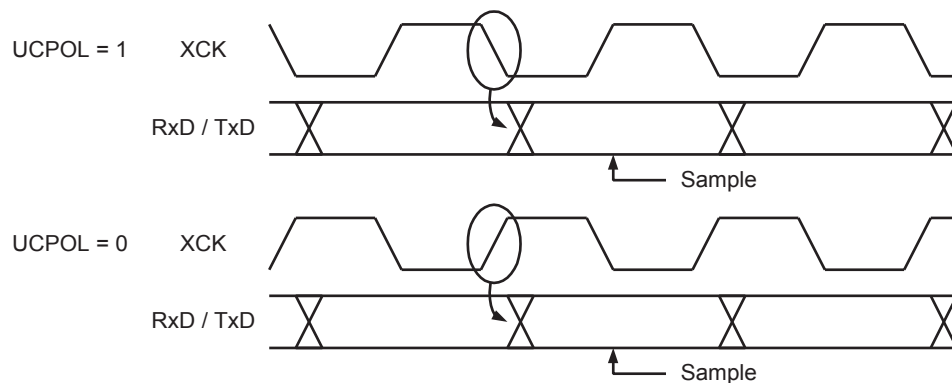
$$f_{XCK} < \frac{f_{OSC}}{4}$$

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.

19.4.4 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 on page 179](#) 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.

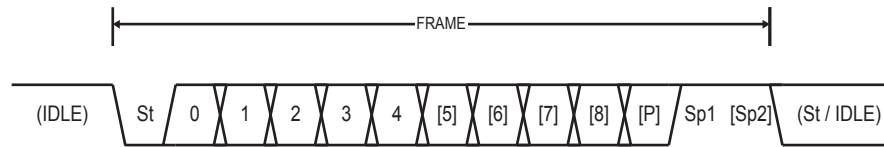
19.5 Frame formats

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:

- One start bit
- 5, 6, 7, 8, or 9 data bits
- No, even, or odd parity bit
- One or two 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 on page 180](#) illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 19-4. Frame formats



StStart bit, always low.

(n)Data bits (0 to 8).

PParity bit. Can be odd or even.

SpStop bit, always high.

IDLENo transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

The frame format used by the USART is set by the UCSZn2:0, UPMn1:0 and USBSn 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 (USBSn) 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.

19.5.1 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_nData 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.

19.6 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 the USART interrupts should be disabled.

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⁾

```

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
    
```

C Code Example ⁽¹⁾

```

void USART_Init( unsigned int baud )
{
    /* Set baud rate */
    UBRRnH = (unsigned char) (baud>>8);
    UBRRnL = (unsigned char) baud;
    /* Enable receiver and transmitter */
    UCSRnB = (1<<RXENn) | (1<<TXENn);
    /* Set frame format: 8data, 2stop bit */
    UCSRnC = (1<<USBSn) | (3<<UCSZn0);
}
    
```

Note: 1. See “About code examples” on page 17.

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.

19.7 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.

19.7.1 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⁾

```
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
```

C Code Example ⁽¹⁾

```
void USART_Transmit( unsigned char data )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSRnA & (1<<UDREn)) )
        ;
    /* Put data into buffer, sends the data */
    UDRn = data;
}
```

Note: 1. See [“About code examples” on page 17](#).

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.

19.7.2 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⁾

```

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

```

C Code Example ⁽¹⁾⁽²⁾

```

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;
}

```

- 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 “About code examples” on page 17.

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.

19.7.3 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.

19.7.4 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.

19.7.5 Disabling the Transmitter

The disabling of the Transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, that is, 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.

19.8 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.

19.8.1 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, that is, 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⁾

```

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

```

C Code Example ⁽¹⁾

```

unsigned char USART_Receive( void )
{
    /* Wait for data to be received */
    while ( !(UCSRnA & (1<<RXCn)) )
        ;
    /* Get and return received data from buffer */
    return UDRn;
}

```

Note: 1. See [“About code examples” on page 17](#).

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.

19.8.2 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⁾

```

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
    
```

C Code Example ⁽¹⁾

```

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);
}
    
```

Note: 1. See “About code examples” on page 17.

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.

19.8.3 Receive Compete 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.

19.8.4 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 ["Parity Bit Calculation" on page 180](#) and ["Parity Checker" on page 187](#).

19.8.5 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.

19.8.6 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 (that is, 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

19.8.7 Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, that is, 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 [“About code examples” on page 17](#).

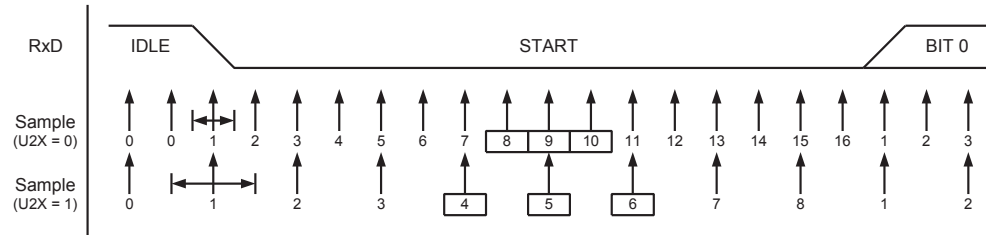
19.9 Asynchronous Data Reception

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.

19.9.1 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 (that is, no communication activity).

Figure 19-5. Start bit sampling

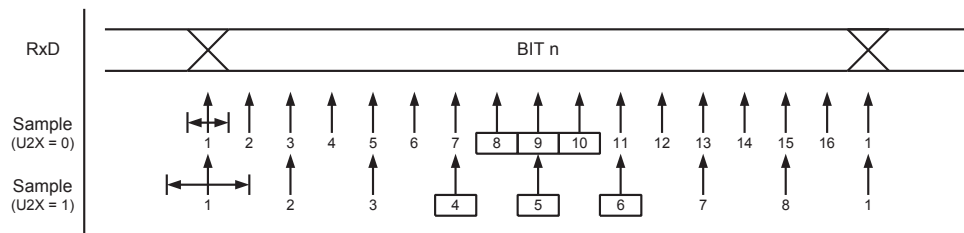


When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 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.

19.9.2 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.

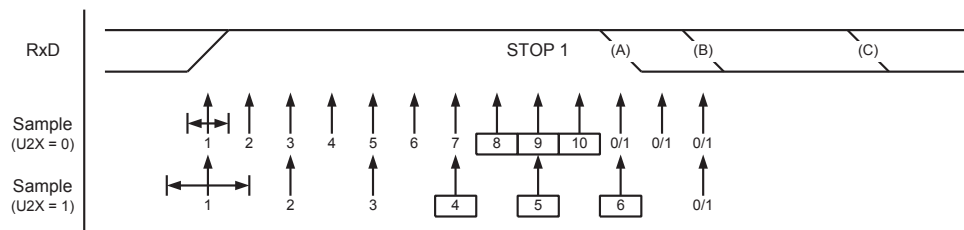
Figure 19-6. Sampling of Data and Parity bit



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 on page 189](#) 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 on page 189](#). 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.

19.9.3 Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If 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 on page 191](#)) base frequency, the Receiver will not be able to synchronize the frames to the start bit.

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

$$R_{slow} = \frac{(D+1)S}{S-1+D \cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D Sum of character size and parity size (D = 5 to 10 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 on page 191](#) and [Table 19-3 on page 191](#) 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} [%]	Maximum 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} [%]	Maximum 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 UBRR value that gives an acceptable low error can be used if possible.

19.10 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.

19.10.1 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.

19.11 Register description

19.11.1 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.

19.11.2 UCSRnA – USART Control and Status Register 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 (that is, 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 RXCIEn 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 TXCIEn 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 UDRIEn 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 ["Multi-processor Communication mode" on page 191](#).

19.11.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIE_n	TXCIE_n	UDRIE_n	RXEN_n	TXEN_n	UCSZ_{n2}	RXB8_n	TXB8_n	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_n: RX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the RXC_n Flag. A USART Receive Complete interrupt will be generated only if the RXCIE_n bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC_n bit in UCSRnA is set.

- **Bit 6 – TXCIE_n: TX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the TXC_n Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE_n bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC_n bit in UCSRnA is set.

- **Bit 5 – UDRIE_n: USART Data Register Empty Interrupt Enable n**

Writing this bit to one enables interrupt on the UDRE_n Flag. A Data Register Empty interrupt will be generated only if the UDRIE_n bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE_n bit in UCSRnA is set.

- **Bit 4 – RXENn: 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 – TXENn: 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 TXENn 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.

19.11.4 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 ["USART in SPI mode" on page 202](#) 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

19.11.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRnH
	UBRR[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**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL 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 UBRRL will trigger an immediate update of the baud rate prescaler.

19.12 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 UBRR settings in [Table 19-9](#) to [Table 19-12](#). UBRR 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 ["Asynchronous Operational Range" on page 190](#)). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1 \right) \cdot 100\%$$

Table 19-9. Examples of UBRRn settings for commonly used oscillator frequencies

Baud rate [bps]	f _{osc} = 1.0000MHz				f _{osc} = 1.8432MHz				f _{osc} = 2.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
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	

1. UBRR = 0, Error = 0.0%.

Table 19-10. Examples of UBRRn settings for commonly used oscillator frequencies. (Continued)

Baud rate [bps]												
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	

1. UBRR = 0, Error = 0.0%.

Table 19-11. Examples of UBRRn settings for commonly used oscillator frequencies (Continued)

Baud rate [bps]	$f_{osc} = 8.0000\text{MHz}$				$f_{osc} = 11.0592\text{MHz}$				$f_{osc} = 14.7456\text{MHz}$			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
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	

1. UBRR = 0, Error = 0.0%.

Table 19-12. Examples of UBRRn settings for commonly used oscillator frequencies (Continued)

Baud rate [bps]	$f_{osc} = 16.0000\text{MHz}$				$f_{osc} = 18.4320\text{MHz}$				$f_{osc} = 20.0000\text{MHz}$			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	—	—	4	-7.8%	—	—	4	0.0%
1M	0	0.0%	1	0.0%	—	—	—	—	—	—	—	—
Max. ⁽¹⁾	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

1. UBRR = 0, Error = 0.0%.

20. USART in SPI mode

20.1 Features

- Full duplex, three-wire synchronous data transfer
- Master Operation
- Supports all four SPI modes of operation (Mode 0, 1, 2, and 3)
- LSB first or MSB first data transfer (configurable data order)
- Queued operation (double buffered)
- High resolution baud rate generator
- High speed operation ($f_{XCKmax} = f_{CK}/2$)
- Flexible interrupt generation

20.2 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation.

Setting both UMSELn1:0 bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

20.3 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRRn setting can therefore be calculated using the same equations, see [Table 20-1](#):

Table 20-1. Equations for calculating baud rate register setting.

Operating mode	Equation for calculating baud rate ⁽¹⁾	Equation for calculating UBRRn value
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

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

BAUDBaud rate (in bits per second, bps).

f_{OSC}System oscillator clock frequency.

UBRRnContents of the UBRRnH and UBRRnL registers, (0-4095).

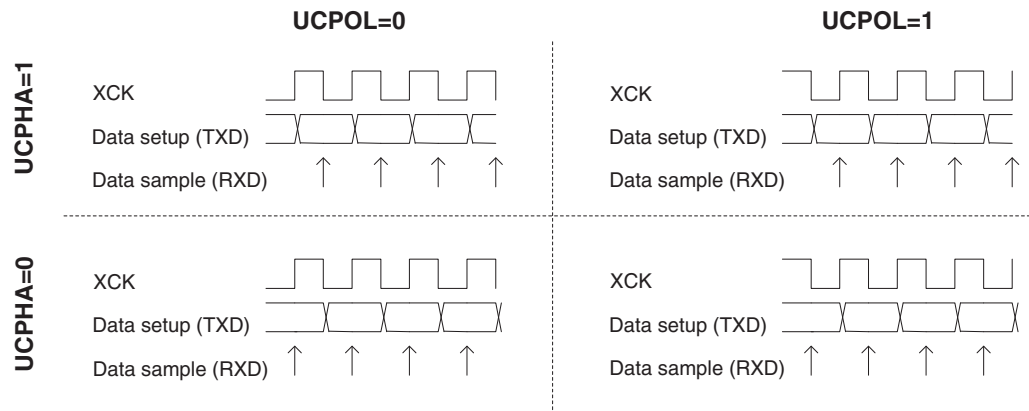
20.4 SPI Data Modes and Timing

There are four combinations of XCKn (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHAN and UCPOLn. The data transfer timing diagrams are shown in [Figure 20-1](#). Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAN functionality is summarized in [Table 20-2](#). Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

Table 20-2. UCPOLn and UCPHAN functionality

UCPOLn	UCPHAn	SPI mode	Leading edge	Trailing edge
0	0	0	Sample (Rising)	Setup (Falling)
0	1	1	Setup (Rising)	Sample (Falling)
1	0	2	Sample (Falling)	Setup (Rising)
1	1	3	Setup (Falling)	Sample (Rising)

Figure 20-1. UCPHAN and UCPOLn data transfer timing diagrams



20.5 Frame Formats

A serial frame for the MSPIM is defined to be one character of eight data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. Then the next data bits, up to a total of eight, are succeeding, ending with the most or least significant bit accordingly. When a complete frame is transmitted, a new frame can directly follow it, or the communication line can be set to an idle (high) state.

The UDORDn bit in UCSRnC sets the frame format used by the USART in MSPIM mode. 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.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

20.5.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate

independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there is 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 RXCn 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 polling (no interrupts enabled). 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⁾

```

USART_Init:
    clr r18
    out UBRRnH,r18
    out UBRRnL,r18
    ; Setting the XCKn port pin as output, enables master
mode.
    sbi XCKn_DDR, XCKn
    ; Set MSPI mode of operation and SPI data mode 0.
    ldi r18,
(1<<UMSELn1) | (1<<UMSELn0) | (0<<UCPHAn) | (0<<UCPOLn)
    out UCSRnC,r18
    ; Enable receiver and transmitter.
    ldi r18, (1<<RXENn) | (1<<TXENn)
    out UCSRnB,r18
    ; Set baud rate.
    ; IMPORTANT: The Baud Rate must be set after the
transmitter is enabled!
    out UBRRnH, r17
    out UBRRnL, r18
    ret

```

C Code Example ⁽¹⁾

```

void USART_Init( unsigned int baud )
{
    UBRRn = 0;
    /* Setting the XCKn port pin as output, enables master
mode. */
    XCKn_DDR |= (1<<XCKn);
    /* Set MSPI mode of operation and SPI data mode 0. */
    UCSRnC =
(1<<UMSELn1) | (1<<UMSELn0) | (0<<UCPHAn) | (0<<UCPOLn);
    /* Enable receiver and transmitter. */
    UCSRnB = (1<<RXENn) | (1<<TXENn);
    /* Set baud rate. */
    /* IMPORTANT: The Baud Rate must be set after the
transmitter is enabled */
    UBRRn = baud;
}

```

Note: 1. See [“About code examples” on page 17](#).

20.6 Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, that is, the TXENn bit in the UCSRnB register is set to one. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the Receiver's serial input. The XCKn will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDRn I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer

clock. The data written to UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, the UDRn register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDREN) Flag and the Receive Complete (RXCn) Flag. 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 and the data received will be available in the same register (R16) after the function returns.

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

Assembly Code Example ⁽¹⁾

```
USART_MSPIM_Transfer:
    ; Wait for empty transmit buffer
    sbis UCSRnA, UDREN
    rjmp USART_MSPIM_Transfer
    ; Put data (r16) into buffer, sends the data
    out UDRn,r16
    ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
    sbis UCSRnA, RXCn
    rjmp USART_MSPIM_Wait_RXCn
    ; Get and return received data from buffer
    in r16, UDRn
    ret
```

C Code Example ⁽¹⁾

```
unsigned char USART_Receive( void )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSRnA & (1<<UDREN)) );
    /* Put data into buffer, sends the data */
    UDRn = data;
    /* Wait for data to be received */
    while ( !(UCSRnA & (1<<RXCn)) );
    /* Get and return received data from buffer */
    return UDRn;
}
```

Note: 1. See [“About code examples” on page 17](#).

20.6.1 Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREN flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and is always read as zero.

20.6.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

20.7 AVR USART MSPIM vs. AVR SPI

The USART in MSPIM mode is fully compatible with the AVR SPI regarding:

- Master mode timing diagram.
- The UCPOLn bit functionality is identical to the SPI CPOL bit
- The UCPHAN bit functionality is identical to the SPI CPHA bit
- The UDORDn bit functionality is identical to the SPI DORD bit

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer
- The USART in MSPIM mode receiver includes an additional buffer level
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting UBRRn accordingly
- Interrupt timing is not compatible
- Pin control differs due to the master only operation of the USART in MSPIM mode

A comparison of the USART in MSPIM mode and the SPI pins is shown in [Table 20-3 on page 207](#).

Table 20-3. Comparison of USART in MSPIM mode and SPI pins

USART_MSPIM	SPI	Comment
TxDn	MOSI	Master Out only
RxDn	MISO	Master In only
XCKn	SCK	(Functionally identical)
(N/A)	\overline{SS}	Not supported by USART in MSPIM

20.8 Register description

The following section describes the registers used for SPI operation using the USART.

20.8.1 UDRn – USART MSPIM I/O Data Register

The function and bit description of the USART data register (UDRn) in MSPI mode is identical to normal USART operation. See “UDRn – USART I/O Data Register n” on page 193.

20.8.2 UCSRnA – USART MSPIM Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	–	–	–	–	–	UCSRnA
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	1	1	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 (that is, 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 RXCIEn 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 TXCIEn 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 4:0 – Reserved in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRnA is written.

20.8.3 UCSRnB – USART MSPIM Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIE	RXENn	TXENn	–	–	–	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXCn Flag. A USART Receive Complete interrupt will be generated only if the RXCIEn 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 – TXCIEn: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXCn Flag. A USART Transmit Complete interrupt will be generated only if the TXCIEn 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**

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 – RXENn: Receiver Enable**

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxDn pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXENn=1 and TXENn=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 – TXENn: Transmitter Enable**

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 TXENn to zero) will not become effective until ongoing and pending transmissions are completed, that is, 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 2:0 – Reserved in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRnB is written.

20.8.4 UCSRnC – USART MSPIM Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	–	–	–	UDORDn	UCPHAn	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7:6 – UMSELn1:0: USART Mode Select**

These bits select the mode of operation of the USART as shown in [Table 20-4](#). See "[UCSRnC – USART Control and Status Register n C](#)" on [page 195](#) for full description of the normal USART operation. The MSPIM is enabled when both UMSELn bits are set to one. The UDORDn, UCPHAn, and UCPOLn can be set in the same write operation where the MSPIM is enabled.

Table 20-4. UMSELn bits settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)

- **Bit 5:3 – Reserved in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRnC is written.

- **Bit 2 – UDORDn: Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to ["Frame Formats" on page 203](#) for details.

- **Bit 1 – UCPHAn: Clock Phase**

The UCPHAn bit setting determine if data is sampled on the leading edge (first) or trailing (last) edge of XCKn. Refer to ["SPI Data Modes and Timing" on page 203](#) for details.

- **Bit 0 – UCPOLn: Clock Polarity**

The UCPOLn bit sets the polarity of the XCKn clock. The combination of the UCPOLn and UCPHAn bit settings determine the timing of the data transfer. Refer to ["SPI Data Modes and Timing" on page 203](#) for details.

20.8.5 UBRRnL and UBRRnH –USART MSPIM Baud Rate Registers

The function and bit description of the baud rate registers in MSPI mode is identical to normal USART operation. See ["UBRRnL and UBRRnH – USART Baud Rate Registers" on page 197](#).

21. Two-wire Serial Interface

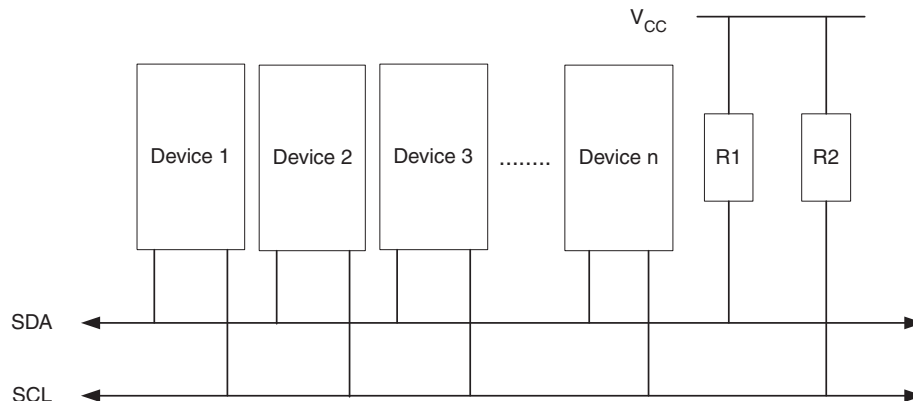
21.1 Features

- Simple yet powerful and flexible communication interface, only two bus lines needed
- Both Master and Slave operation supported
- Device can operate as transmitter or receiver
- 7-bit address space allows up to 128 different Slave addresses
- Multi-master arbitration support
- Up to 400kHz data transfer speed
- Slew-rate limited output drivers
- Noise suppression circuitry rejects spikes on bus lines
- Fully programmable Slave address with General Call support
- Address recognition causes wake-up when AVR is in Sleep mode

21.2 Two-wire Serial Interface bus definition

The Two-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 21-1. TWI bus interconnection



21.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 21-1. TWI terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

The Power Reduction TWI bit, PRTWI bit in "PRR0 – Power Reduction Register 0" on page 56 must be written to zero to enable the two-wire Serial Interface.

21.2.2 Electrical Interconnection

As depicted in Figure 21-1, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices trim-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

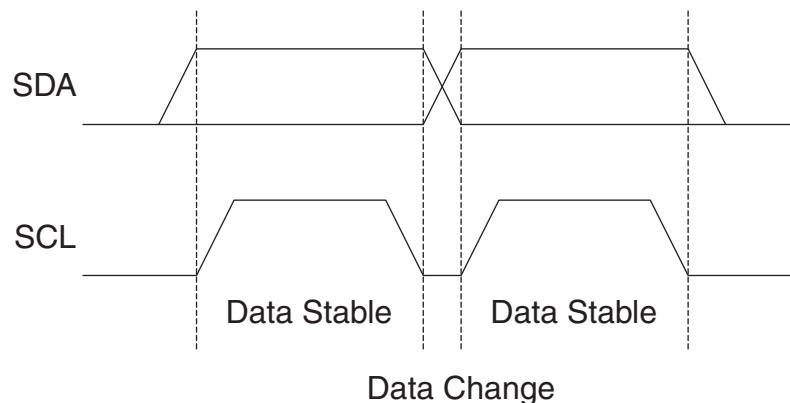
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400pF and the 7-bit slave address space. A detailed specification of the electrical characteristics of the TWI is given in "SPI timing characteristics" on page 335. Two different sets of specifications are presented there, one relevant for bus speeds below 100kHz, and one valid for bus speeds up to 400kHz.

21.3 Data Transfer and Frame Format

21.3.1 Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

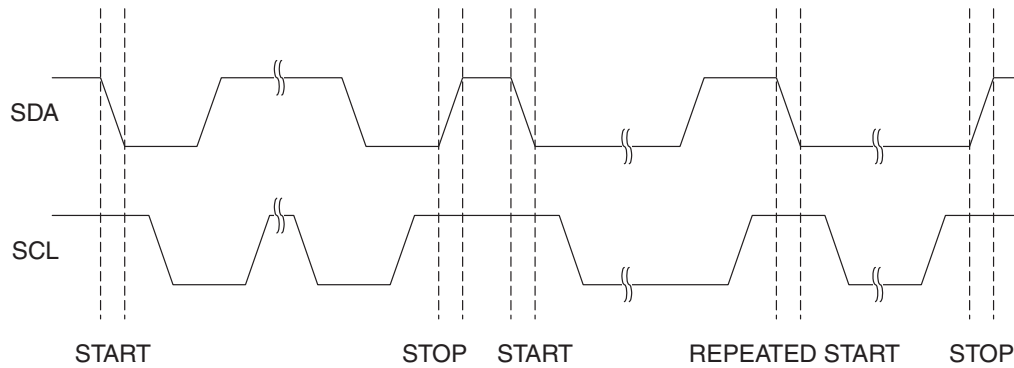
Figure 21-2. Data validity



21.3.2 START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without relinquishing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signalled by changing the level of the SDA line when the SCL line is high.

Figure 21-3. START, REPEATED START and STOP conditions



21.3.3 Address Packet Format

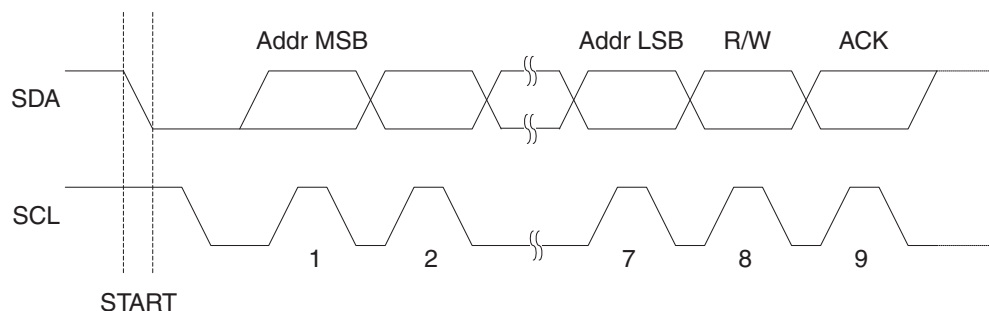
All address packets transmitted on the TWI bus are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 21-4. Address packet format

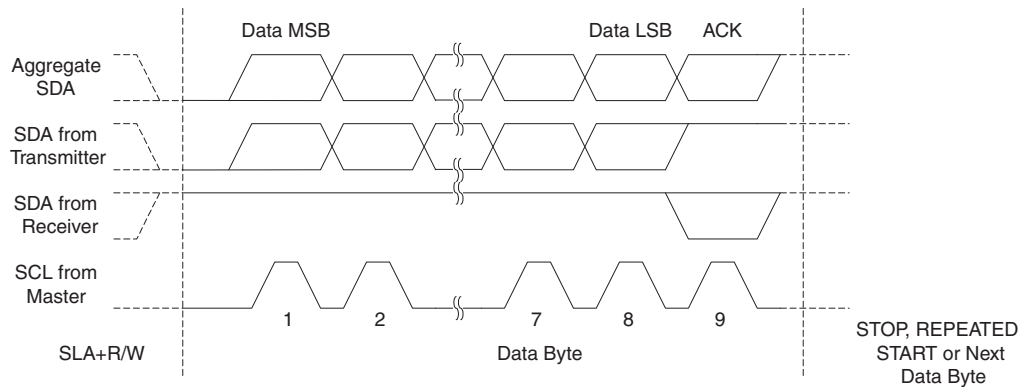


21.3.4 Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the Receiver pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signalled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it

should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 21-5. Data packet format

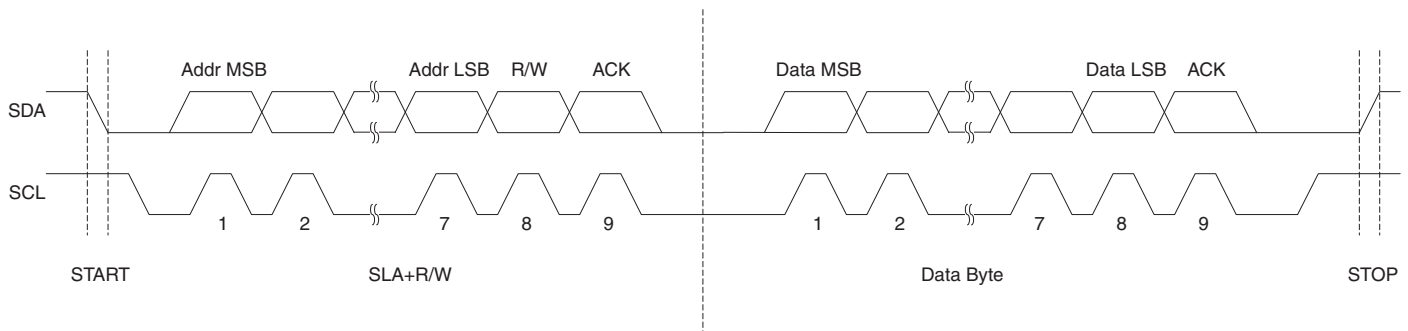


21.3.5 Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the Wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 21-6 on page 214 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 21-6. Typical data transmission



21.4 Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

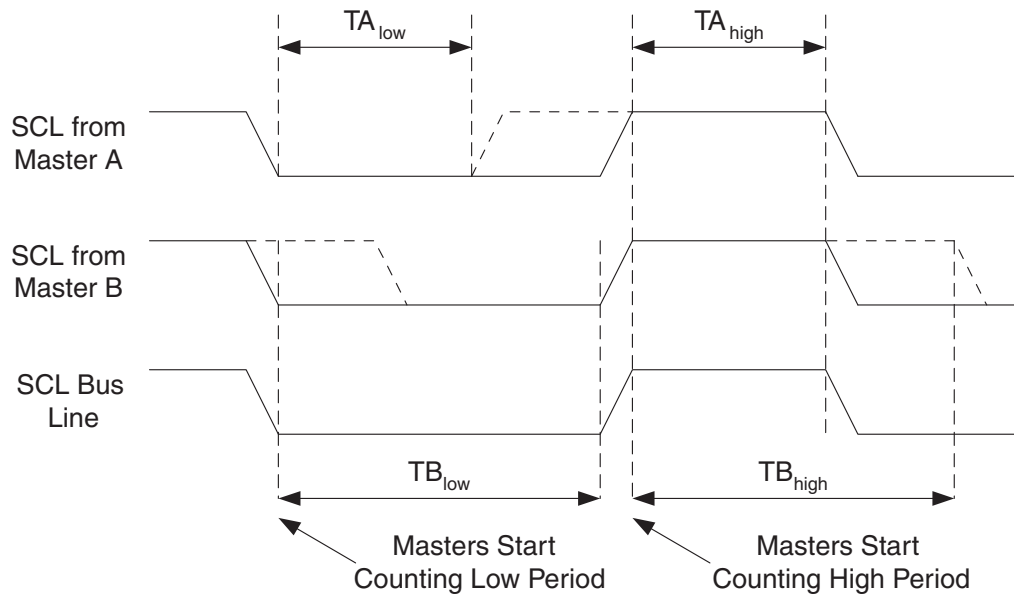
- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the

arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, that is the data being transferred on the bus must not be corrupted

- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process

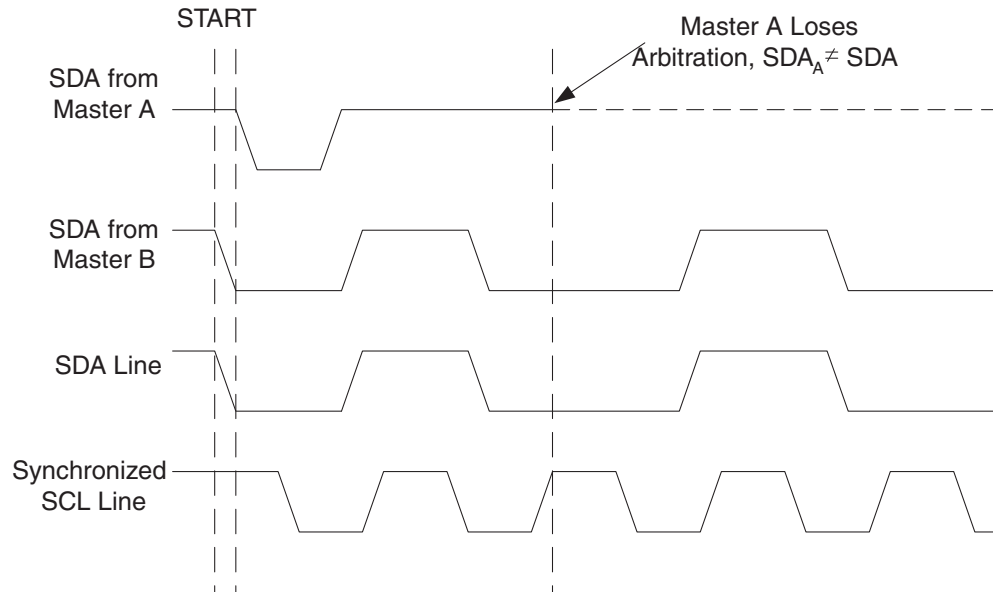
The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 21-7. SCL synchronization between multiple masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several masters are trying to address the same Slave, arbitration will continue into the data packet.

Figure 21-8. Arbitration between two masters



Note that arbitration is not allowed between:

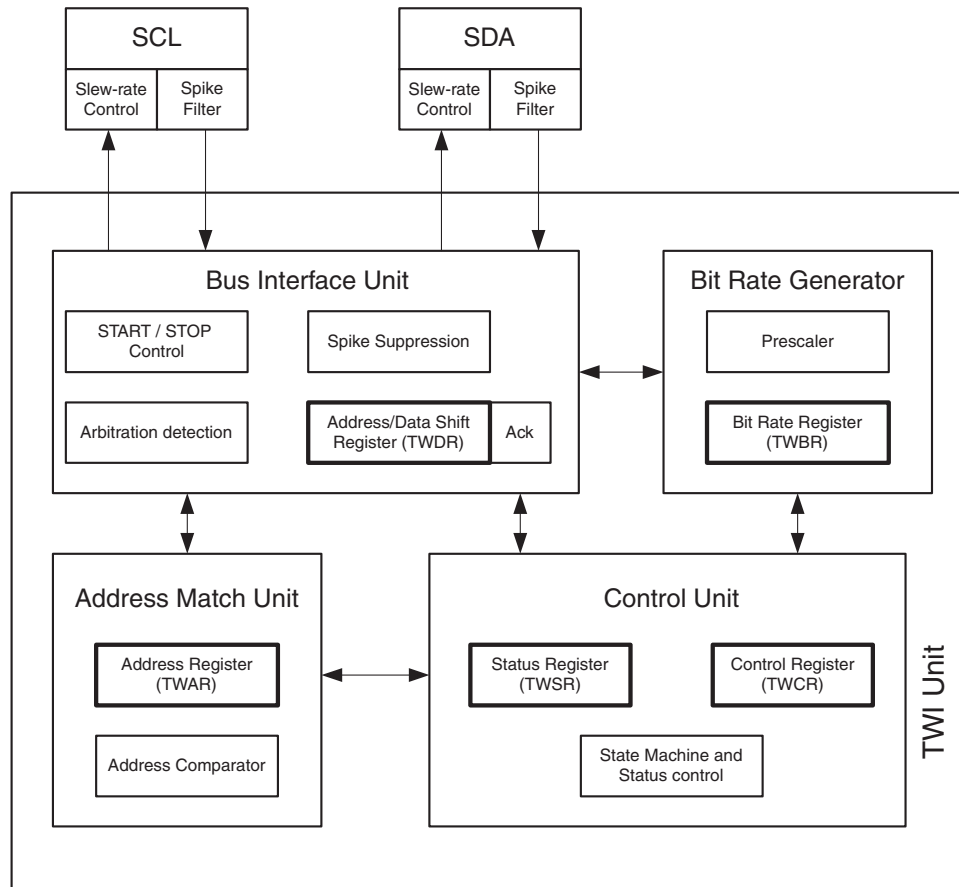
- A REPEATED START condition and a data bit
- A STOP condition and a data bit
- A REPEATED START and a STOP condition

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

21.5 Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 21-9. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 21-9. Overview of the TWI module



21.5.1 SCL and SDA pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

21.5.2 Bit Rate Generator unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{\text{TWPS}}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See two-wire Serial Bus Requirements in [Table 28-16 on page 336](#) for value of pull-up resistor.

21.5.3 Bus Interface unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted, or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

21.5.4 Address Match unit

The Address Match unit checks if received address bytes match the seven-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to compare addresses even when the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master.

21.5.5 Control unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition
- After the TWI has transmitted SLA+R/W
- After the TWI has transmitted an address byte
- After the TWI has lost arbitration
- After the TWI has been addressed by own slave address or general call
- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a Slave
- When a bus error has occurred due to an illegal START or STOP condition

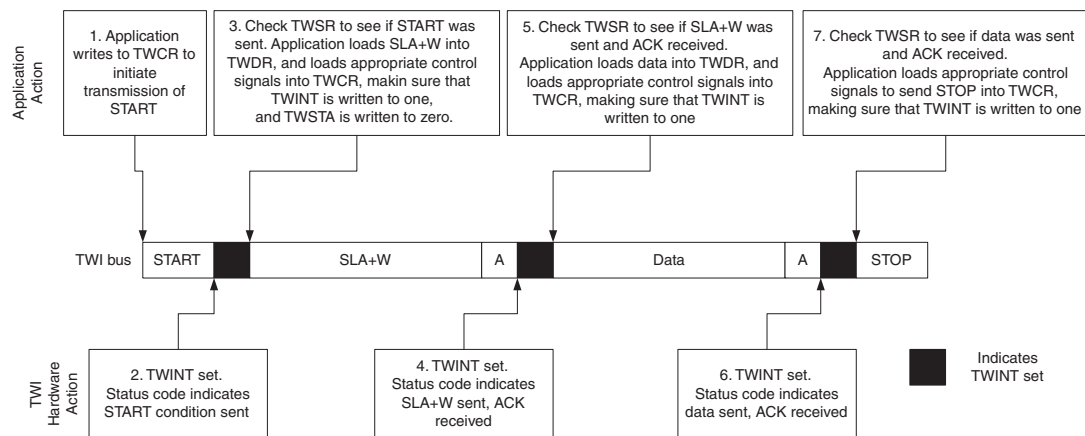
21.6 Using the TWI

The AVR TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 21-10 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

Figure 21-10. Interfacing the application to the TWI in a typical transmission.



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCR, instructing the TWI hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the START condition.
2. When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.

4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.
6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then commence executing whatever operation was specified by the TWCR setting

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.

Table 21-2. Assembly code and C code examples

	Assembly code example	C example	Comments
1	<pre>ldi r16, (1<<TWINT) (1<<TWSTA) (1<<TWEN) out TWCR, r16</pre>	<pre>TWCR = (1<<TWINT) (1<<TWSTA) (1<<TWEN)</pre>	Send START condition
2	<pre>wait1: in r16,TWCR sbrs r16,TWINT rjmp wait1</pre>	<pre>while (!(TWCR & (1<<TWINT))) ;</pre>	Wait for TWINT Flag set. This indicates that the START condition has been transmitted
3	<pre>in r16,TWSR andi r16, 0xF8 cpi r16, START brne ERROR</pre>	<pre>if ((TWSR & 0xF8) != START) ERROR();</pre>	Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR
	<pre>ldi r16, SLA_W out TWDR, r16 ldi r16, (1<<TWINT) (1<<TWEN) out TWCR, r16</pre>	<pre>TWDR = SLA_W; TWCR = (1<<TWINT) (1<<TWEN);</pre>	Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address
4	<pre>wait2: in r16,TWCR sbrs r16,TWINT rjmp wait2</pre>	<pre>while (!(TWCR & (1<<TWINT))) ;</pre>	Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.

Table 21-2. Assembly code and C code examples

	Assembly code example	C example	Comments
5	<pre> in r16, TWSR andi r16, 0xF8 cpi r16, MT_SLA_ACK brne ERROR </pre>	<pre> if ((TWSR & 0xF8) != MT_SLA_ACK) ERROR(); </pre>	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR
	<pre> ldi r16, DATA out TWDR, r16 ldi r16, (1<<TWINT) (1<<TWEN) out TWCR, r16 </pre>	<pre> TWDR = DATA; TWCR = (1<<TWINT) (1<<TWEN); </pre>	Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data
6	<pre> wait3: in r16, TWCR sbrs r16, TWINT rjmp wait3 </pre>	<pre> while (!(TWCR & (1<<TWINT))) ; </pre>	Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.
7	<pre> in r16, TWSR andi r16, 0xF8 cpi r16, MT_DATA_ACK brne ERROR </pre>	<pre> if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR(); </pre>	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR
	<pre> ldi r16, (1<<TWINT) (1<<TWEN) (1<<TWSTO) out TWCR, r16 </pre>	<pre> TWCR = (1<<TWINT) (1<<TWEN) (1<<TWSTO); </pre>	Transmit STOP condition

21.7 Transmission modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S:START condition

Rs:REPEATED START condition

R:Read bit (high level at SDA)

W:Write bit (low level at SDA)

A:Acknowledge bit (low level at SDA)

\overline{A} :Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P:STOP condition

SLA:Slave Address

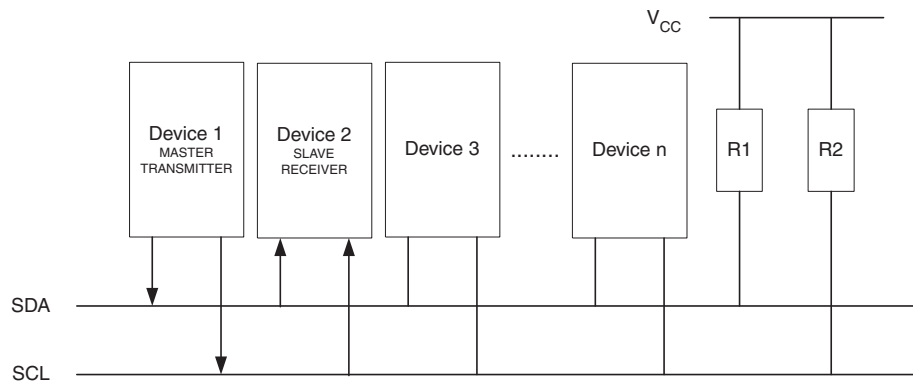
In [Figure 21-12 on page 225](#) to [Figure 21-18 on page 233](#), circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in [Table 21-3 on page 224](#) to [Table 21-6 on page 232](#). Note that the prescaler bits are masked to zero in these tables.

21.7.1 Master Transmitter mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see [Figure 21-11 on page 223](#)). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 21-11. Data transfer in Master Transmitter mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN must be set to enable the two-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT Flag. The TWI will then test the two-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (see [Table 21-3 on page 224](#)). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in [Table 21-3 on page 224](#).

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCR Register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) the two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

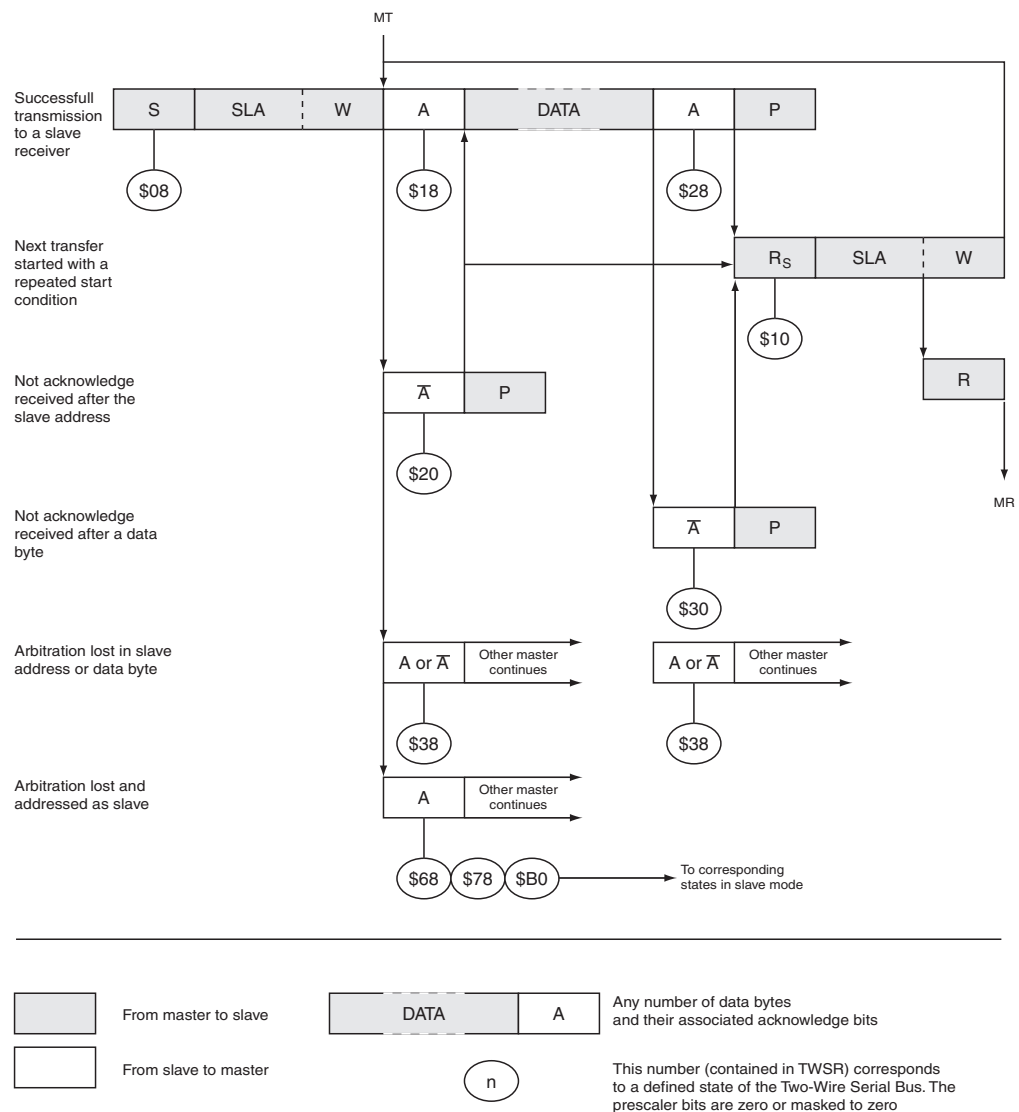
Table 21-3. Status codes for Master Transmitter mode

Status Code (TWSR) Prescaler Bits are 0	Status of the two-wire Serial Bus and two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received SLA+R will be transmitted; Logic will switch to Master Receiver mode
		Load SLA+R	0	0	1	X	
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or No TWDR action or	1 0	0 1	1 1	X X	
		No TWDR action	1	1	1	X	
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or No TWDR action or	1 0	0 1	1 1	X X	
		No TWDR action	1	1	1	X	

Table 21-3. Status codes for Master Transmitter mode

0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or	1	0	1	X	
		No TWDR action	0	1	1	X	
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
		No TWDR action or	1	0	1	X	
		No TWDR action	0	1	1	X	
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	two-wire Serial Bus will be released and not addressed Slave mode entered A START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	X	

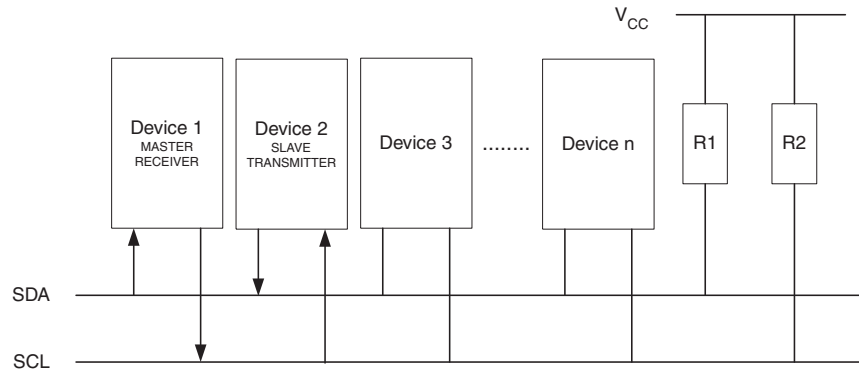
Figure 21-12. Formats and states in the Master Transmitter mode



21.7.2 Master Receiver mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (Slave see [Figure 21-13 on page 226](#)). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 21-13. Data transfer in Master Receiver mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN must be written to one to enable the two-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be set to clear the TWINT Flag. The TWI will then test the two-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (See [Table 21-3 on page 224](#)). In order to enter MR mode, SLA+R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

When SLA+R have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x38, 0x40, or 0x48. The appropriate action to be taken for each of these status codes is detailed in [Table 21-4 on page 227](#).

Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

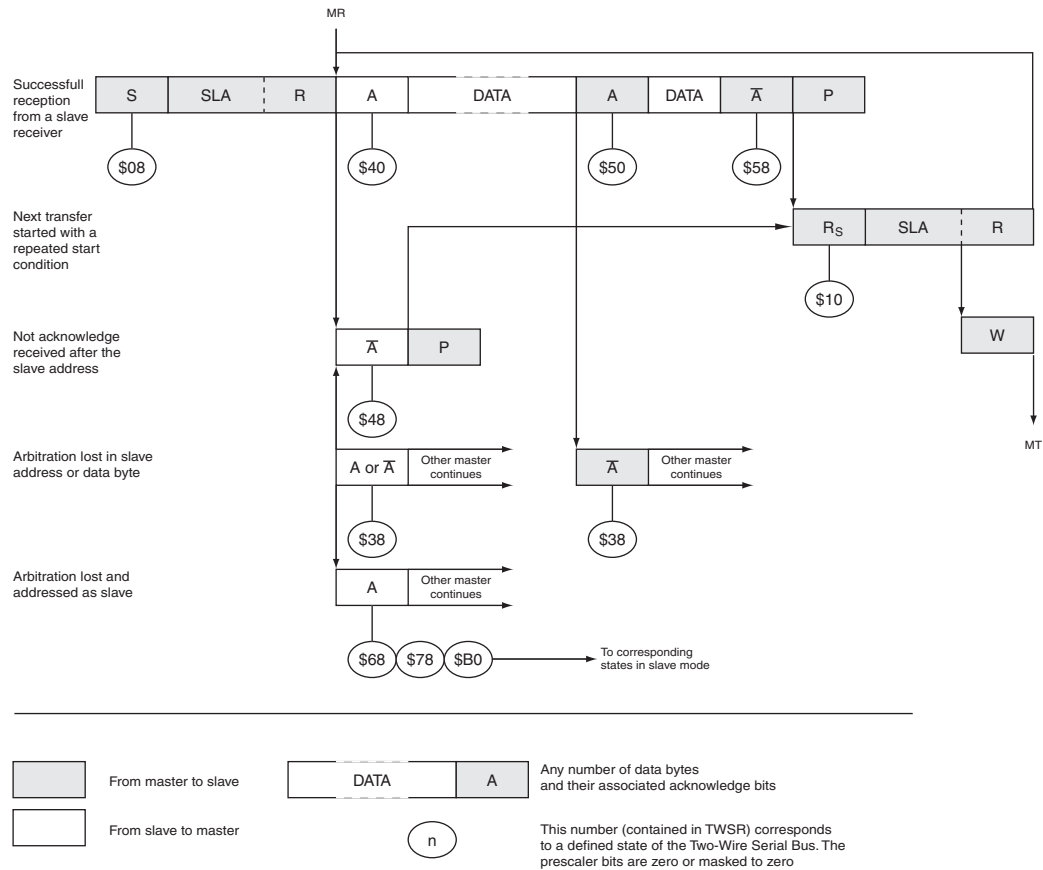
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) the two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 21-4. Status codes for Master Receiver mode

Status Code (TWSR) Prescaler Bits are 0	Status of the two-wire Serial Bus and two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	two-wire Serial Bus will be released and not addressed
		No TWDR action	1	0	1	X	Slave mode will be entered A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

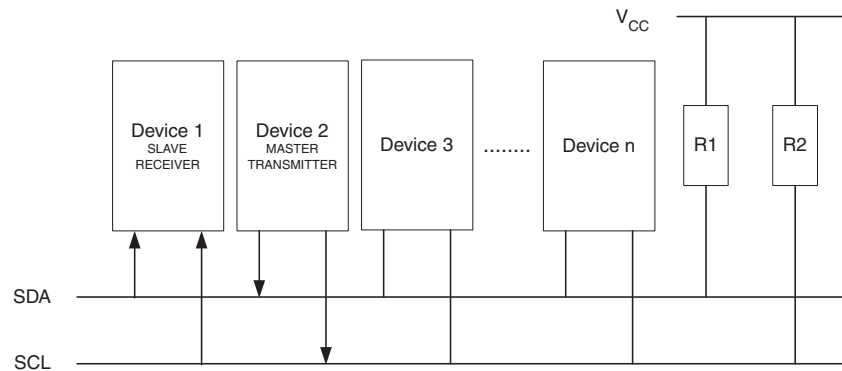
Figure 21-14. Formats and states in the Master Receiver mode



21.7.3 Slave Receiver mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see [Figure 21-15](#)). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 21-15. Data transfer in Slave Receiver mode



To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

The upper seven bits are the address to which the two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in [Table 21-5 on page 230](#). The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own slave address. However, the two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the two-wire Serial Bus.

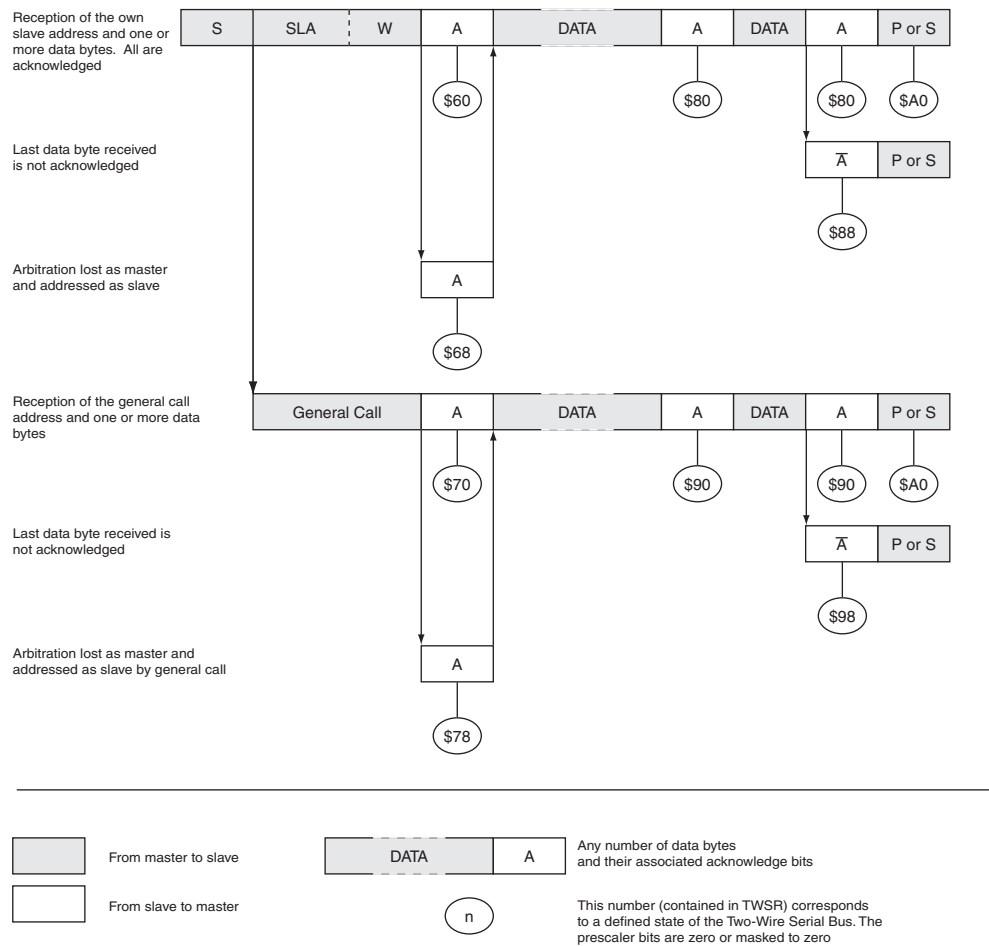
In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the two-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these Sleep modes.

Table 21-5. Status codes for Slave Receiver mode

Status Code (TWSR) Prescaler Bits are 0	Status of the two-wire Serial Bus and two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWIN T	TWE A	
0x60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized;
		Read data byte or	1	0	1	0	GCA will be recognized if TWGCE = “1” Switched to the not addressed Slave mode; no recognition of own SLA or GCA;
		Read data byte	1	0	1	1	a START condition will be transmitted when the bus becomes free Switched to the not addressed Slave mode; own SLA will be recognized;
0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
		Read data byte	X	0	1	1	
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte or	0	0	1	1	Data byte will be received and ACK will be returned
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized;
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	GCA will be recognized if TWGCE = “1”
			0	0	1	1	Switched to the not addressed Slave mode; no recognition of own SLA or GCA;
			1	0	1	0	a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized;
							GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

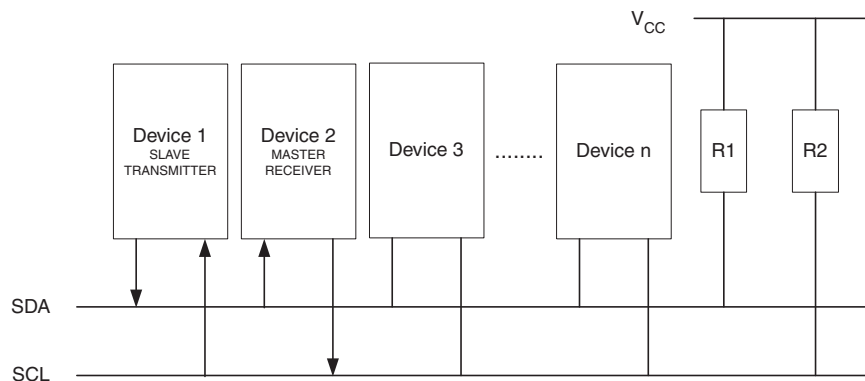
Figure 21-16. Formats and states in the Slave Receiver mode



21.7.4 Slave Transmitter mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see [Figure 21-17](#)). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 21-17. Data transfer in Slave Transmitter mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

The upper seven bits are the address to which the two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in [Table 21-6 on page 232](#). The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State 0xC8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own slave address. However, the two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the two-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the two-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the two-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.

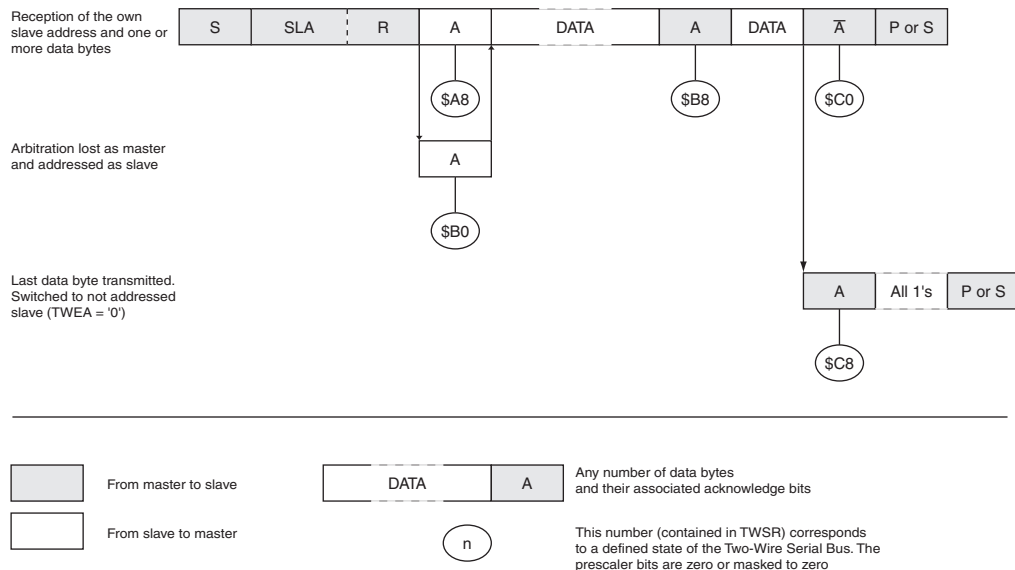
Table 21-6. Status codes for Slave Transmitter mode

Status Code (TWSR) Prescaler Bits are 0	Status of the two-wire Serial Bus and two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received Data byte will be transmitted and ACK should be received
		Load data byte	X	0	1	1	
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received Data byte will be transmitted and ACK should be received
		Load data byte	X	0	1	1	

Table 21-6. Status codes for Slave Transmitter mode

0xB8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or Load data byte	X X	0 0	1 1	0 1	Last data byte will be transmitted and NOT ACK should be received Data byte will be transmitted and ACK should be received
0xC0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or No TWDR action or No TWDR action or No TWDR action	0 0 1 1	0 0 0 0	1 1 1 1	0 1 0 1	Switched to the not addressed Slave mode; no recognition of own SLA or GCA Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1" Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
0xC8	Last data byte in TWDR has been transmitted (TWEA = "0"); ACK has been received	No TWDR action or No TWDR action or No TWDR action or No TWDR action	0 0 1 1	0 0 0 0	1 1 1 1	0 1 0 1	Switched to the not addressed Slave mode; no recognition of own SLA or GCA Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1" Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

Figure 21-18. Formats and states in the Slave Transmitter mode



21.7.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see [Table 21-7](#).

Status 0xF8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status 0x00 indicates that a bus error has occurred during a two-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by

writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 21-7. Miscellaneous states

Status Code (TWSR) Prescaler Bits are 0	Status of the two-wire Serial Bus and two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
STA	STO		TWINT	TWEA			
0xF8	No relevant state information available; TWINT = "0"	No TWDR action	No TWCR action				Wait or proceed current transfer
0x00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

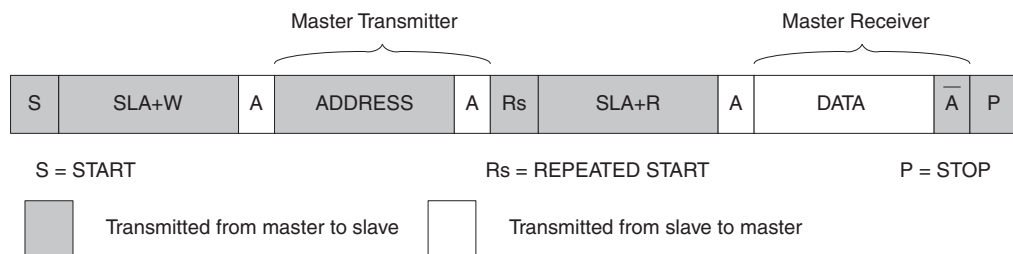
21.7.6 Combining several TWI modes

In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

1. The transfer must be initiated.
2. The EEPROM must be instructed what location should be read.
3. The reading must be performed.
4. The transfer must be finished.

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomical operation. If this principle is violated in a multimaster system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.

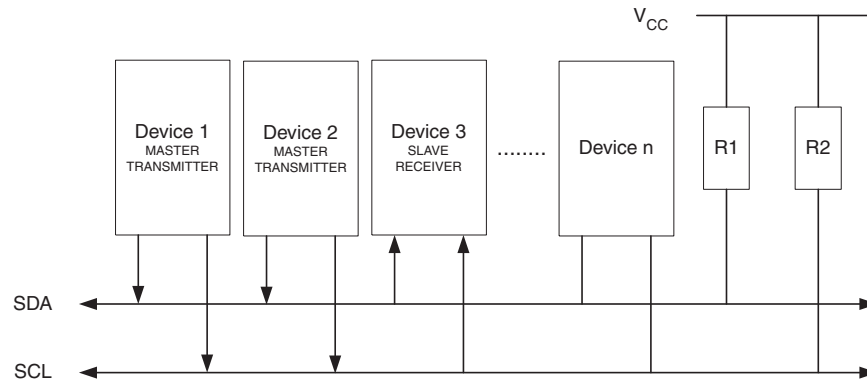
Figure 21-19. Combining several TWI modes to access a serial EEPROM



21.8 Multi-master Systems and Arbitration

If multiple masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two masters are trying to transmit data to a Slave Receiver.

Figure 21-20. An Arbitration example

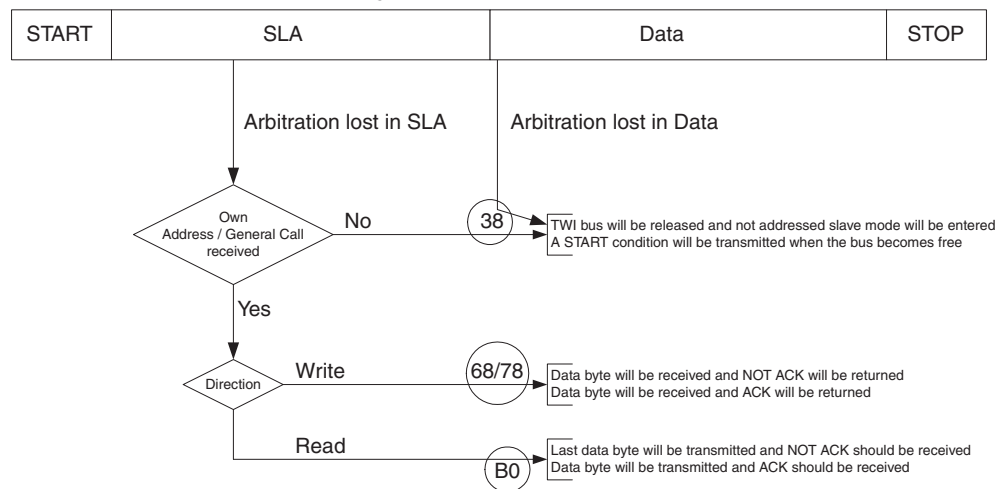


Several different scenarios may arise during arbitration, as described below:

- Two or more masters are performing identical communication with the same Slave. In this case, neither the Slave nor any of the masters will know about the bus contention
- Two or more masters are accessing the same Slave with different data or direction bit. In this case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Losing masters will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action
- Two or more masters are accessing different slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action

This is summarized in [Figure 21-21](#). Possible status values are given in circles.

Figure 21-21. Possible status codes caused by Arbitration



21.9 Register description

21.9.1 TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
(0xB8)	TWBR								TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bits 7:0 – TWI Bit Rate Register

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See ["Bit Rate Generator unit" on page 217](#) for calculating bit rates.

21.9.2 TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
(0xBC)	TWCR								TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a Receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

• Bit 7 – TWINT: TWI Interrupt Flag

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

• Bit 6 – TWEA: TWI Enable Acknowledge Bit

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met:

1. The device's own slave address has been received.
2. A general call has been received, while the TWGCE bit in the TWAR is set.
3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the two-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

• Bit 5 – TWSTA: TWI START Condition Bit

The application writes the TWSTA bit to one when it desires to become a Master on the two-wire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the two-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC: TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

- **Bit 1 – Reserved**

This bit is a reserved bit and will always read as zero.

- **Bit 0 – TWIE: TWI Interrupt Enable**

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

21.9.3 TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
(0xB9)	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7:3 – TWS: TWI Status**

These 5 bits reflect the status of the TWI logic and the two-wire Serial Bus. The different status codes are described "[Transmission modes](#)" on [page 222](#). Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Reserved**

This bit is reserved and will always read as zero.

- **Bits 1:0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

Table 21-8. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see ["Bit Rate Generator unit" on page 217](#). The value of TWPS1..0 is used in the equation.

21.9.4 TWDR – TWI Data Register

Bit (0xBB)	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

• Bits 7:0 – TWD: TWI Data Register

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the two-wire Serial Bus.

21.9.5 TWAR – TWI (Slave) Address Register

Bit (0xBA)	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. In multimaster systems, TWAR must be set in masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

• Bits 7:1 – TWA: TWI (Slave) Address Register

These seven bits constitute the slave address of the TWI unit.

- **Bit 0 – TWGCE: TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a General Call given over the two-wire Serial Bus.

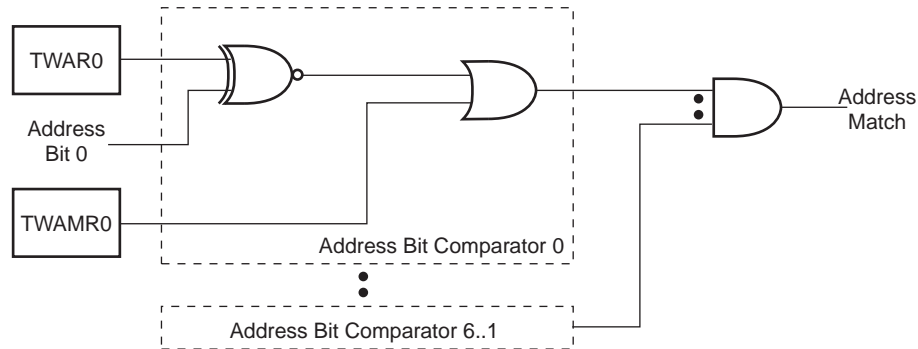
21.9.6 TWAMR – TWI (Slave) Address Mask Register

Bit	7	6	5	4	3	2	1	0	
(0xBD)	TWAM[6:0]							–	TWAMR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – TWAM: TWI Address Mask**

The TWAMR can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bit in the TWI Address Register (TWAR). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR. [Figure 21-22](#) shows the address match logic in detail.

Figure 21-22. TWI address match logic, block diagram



- **Bit 0 – Reserved**

This bit is reserved and will always read as zero.

23. ADC - Analog-to-digital converter

23.1 Features

- 10-bit resolution
- 0.5LSB integral non-linearity
- ± 2 LSB absolute accuracy
- 13 - 260 μ s conversion time
- Up to 15kSPS at maximum resolution
- Eight multiplexed single ended input channels
- Differential mode with selectable gain at 1 \times , 10 \times , or 200 \times
- Optional left adjustment for ADC result readout
- 0 - V_{CC} ADC input voltage range
- 2.7 - V_{CC} differential ADC voltage range
- Selectable 2.56V or 1.1V ADC reference voltage
- Free Running or Single Conversion mode
- ADC start conversion by auto triggering on interrupt sources
- Interrupt on ADC conversion complete
- Sleep mode noise canceler

Note: 1. The differential input channels are not tested for devices in PDIP package. This feature is only ensured to work for devices in TQFP, VQFN/QFN/MLF, VFBGA and DRQFN packages.

23.2 Overview

The ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

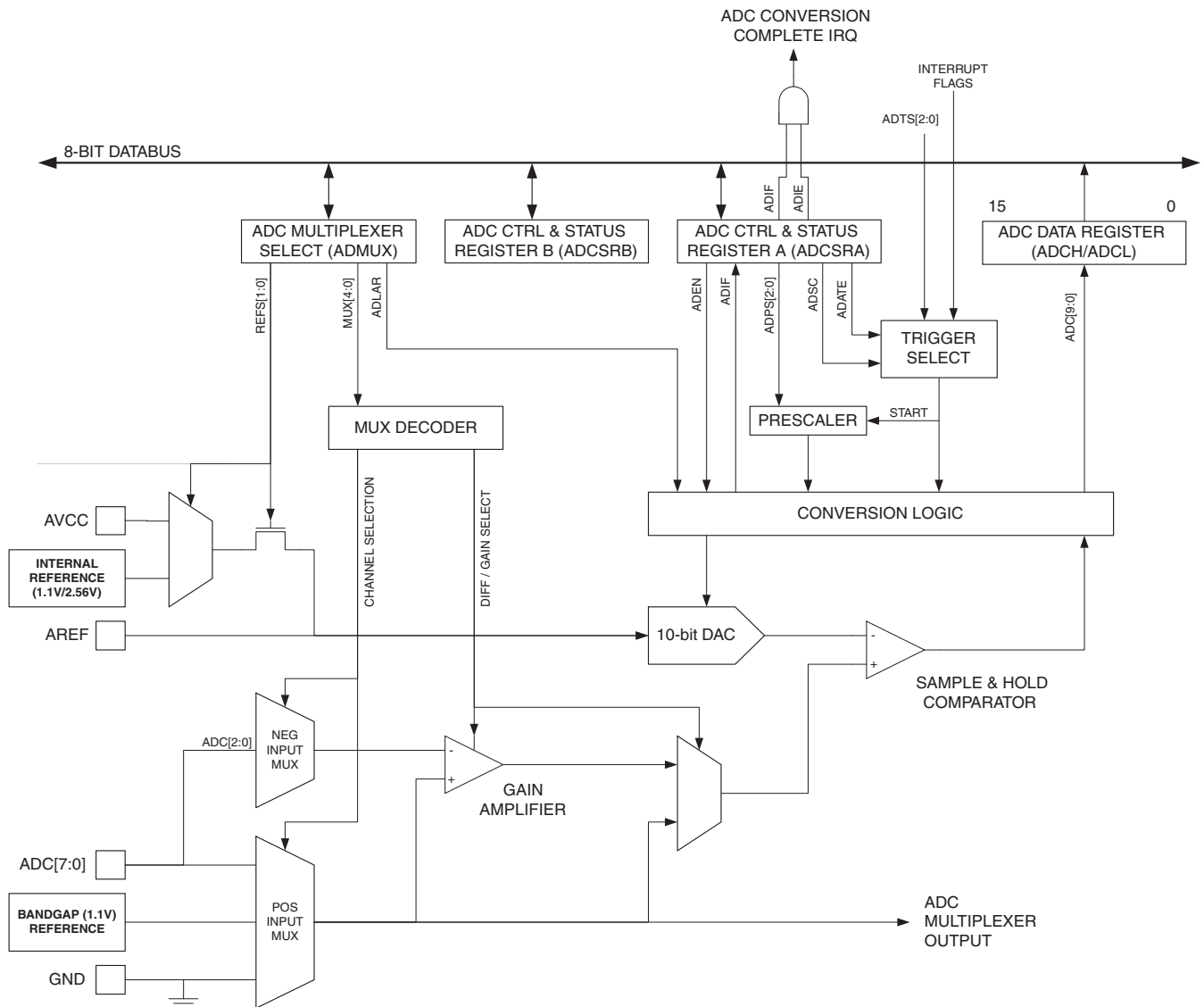
The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage. This provides amplification steps of 0 dB (1 \times), 20 dB (10 \times), or 46 dB (200 \times) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1 \times or 10 \times gain is used, 8-bit resolution can be expected. If 200 \times gain is used, 6-bit resolution can be expected. Note that internal references of 1.1V should not be used on 10 \times and 200 \times gain.

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 244](#).

The ADC has a separate analog supply voltage pin, AV_{CC} . AV_{CC} must not differ more than ± 0.3 V from V_{CC} . See the paragraph "[ADC Noise Canceler](#)" on [page 250](#) on how to connect this pin.

Internal reference voltages of nominally 1.1V, 2.56V, 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. If V_{CC} is below 2.1V, internal voltage reference of 1.1V should not be used on single ended channels.

Figure 23-1. Analog-to-digital Converter block schematic



23.3 Operation

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 1LSB. Optionally, AV_{CC} or an internal 2.56V 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.

The analog input channel and differential gain are 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. A selection of ADC input pins can be selected as positive and negative inputs to the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by the selected gain factor. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

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.

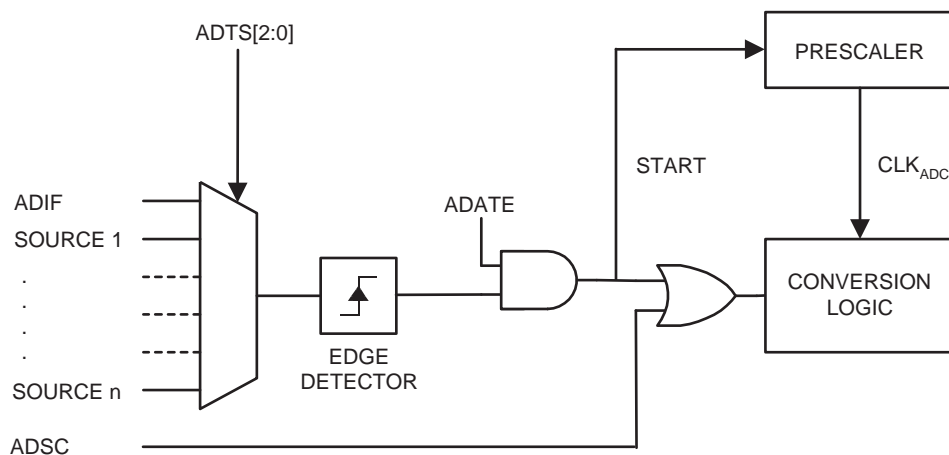
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

23.4 Starting a conversion

A single conversion is started by 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



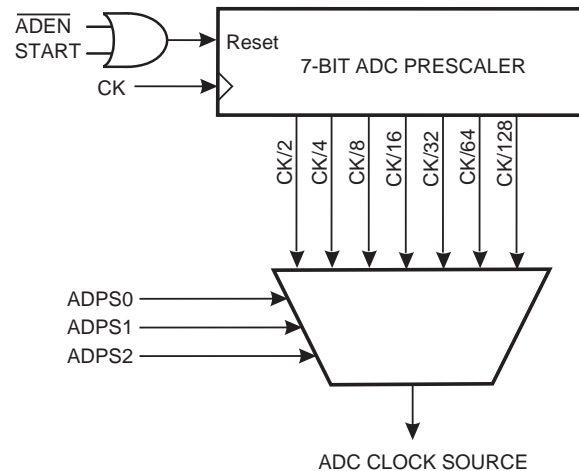
Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the 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.

23.5 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. See ["Differential Gain Channels" on page 248](#) for details on differential conversion timing.

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 a 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 2 ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

When using Differential mode, along with Auto Triggering from a source other than the ADC Conversion Complete, each conversion will require 25 ADC clocks. This is because the ADC must be disabled and re-enabled after every conversion.

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 248](#).

Figure 23-4. ADC Timing diagram, first conversion (Single Conversion mode)

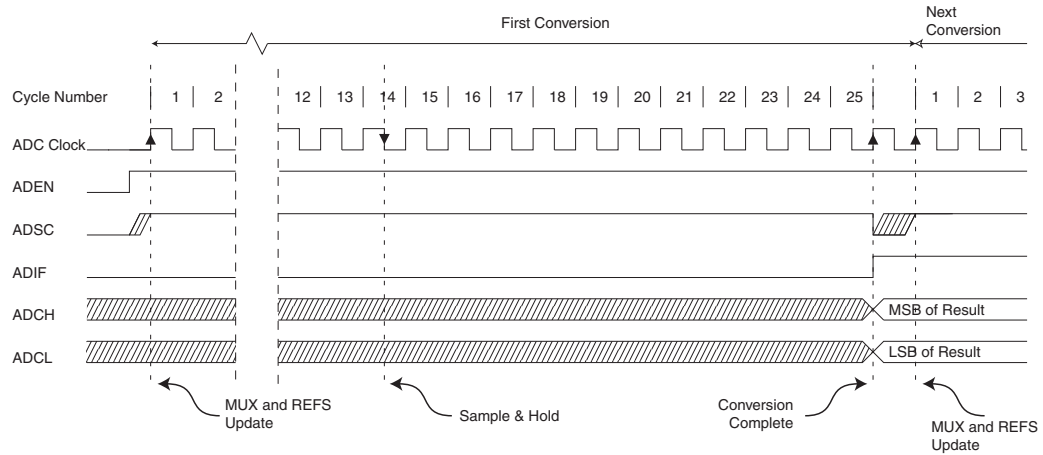


Figure 23-5. ADC Timing diagram, single conversion

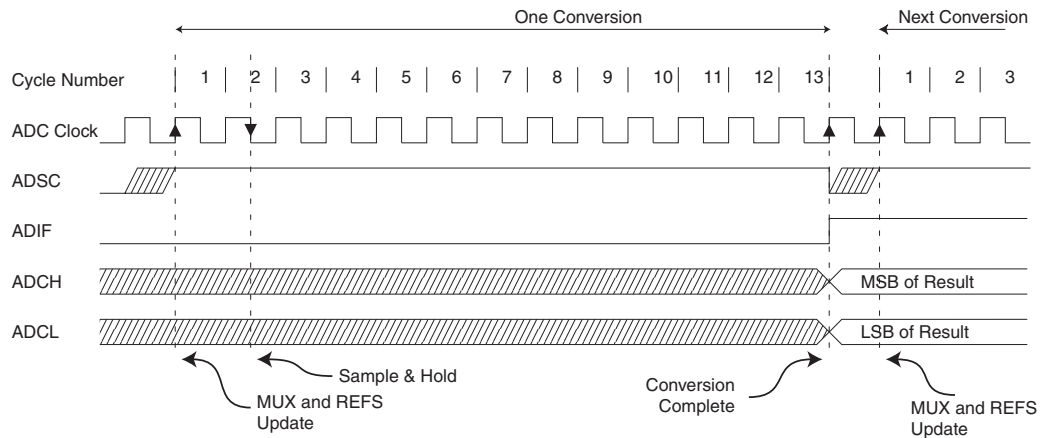


Figure 23-6. ADC Timing diagram, auto triggered conversion

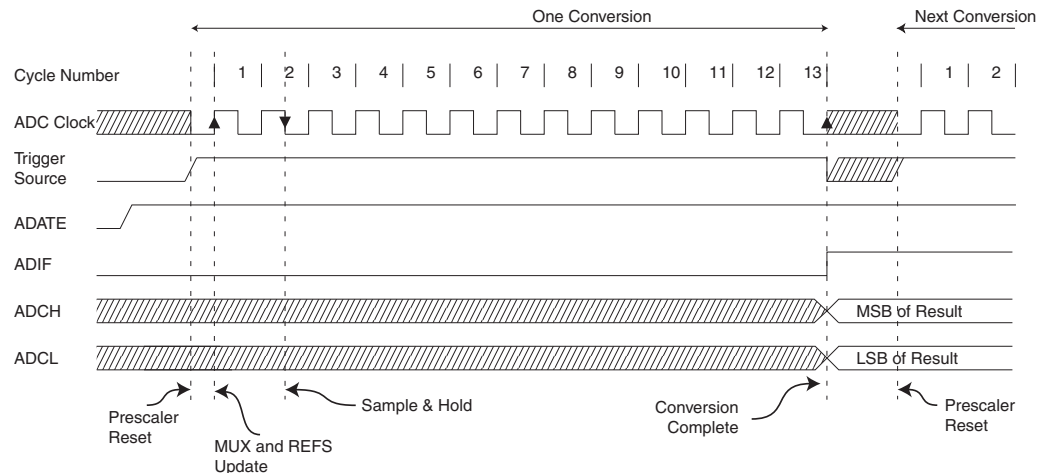


Figure 23-7. ADC Timing diagram, free running conversion

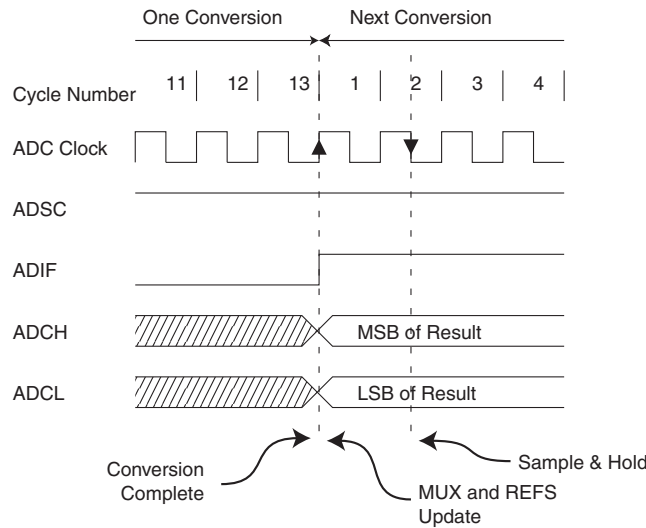


Table 23-1. ADC conversion time

Condition	Sample & Hold (cycles from start of conversion)	Conversion time (cycles)
First conversion	14.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

23.5.1 Differential Gain Channels

When using differential gain channels, certain aspects of the conversion need to be taken into consideration. Note that the differential channels should not be used with an $AREF < 2V$.

Differential conversions are synchronized to the internal clock CK_{ADC2} equal to half the ADC clock. This synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CK_{ADC2} . A conversion initiated by the user (that is, all single conversions, and the first free running conversion) when CK_{ADC2} is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CK_{ADC2} is high will take 14 ADC clock cycles due to the synchronization mechanism. In Free Running mode, a new conversion is initiated immediately after the previous conversion completes, and since CK_{ADC2} is high at this time, all automatically started (that is, all but the first) free running conversions will take 14 ADC clock cycles.

The gain stage is optimized for a bandwidth of 4kHz at all gain settings. Higher frequencies may be subjected to non-linear amplification. An external low-pass filter should be used if the input signal contains higher frequency components than the gain stage bandwidth. Note that the ADC clock frequency is independent of the gain stage bandwidth limitation. For example, the ADC clock period may be 6μs, allowing a channel to be sampled at 12kSPS, regardless of the bandwidth of this channel.

If differential gain channels are used and conversions are started by Auto Triggering, the ADC must be switched off between conversions. When Auto Triggering is used, the ADC prescaler is reset before the conversion is started. Since the gain stage is dependent of a stable ADC clock prior to the conversion, this conversion will not be valid. By disabling and then re-enabling the ADC between each conversion (writing ADEN in ADCSRA to “0” then to “1”), only extended conversions are performed. The result from the extended conversions will be valid. See [“Prescaling and Conversion Timing” on page 246](#) for timing details.

23.6 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:

1. When ADATE or ADEN is cleared.
2. During conversion, minimum one ADC clock cycle after the trigger event.
3. 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.

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 μ s to stabilize to the new value. Thus conversions should not be started within the first 125 μ s after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded.

The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS1:0 bits in ADMUX).

23.6.1 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.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

23.6.2 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 AVCC, internal 2.56V reference, or external AREF pin.

AVCC is connected to the ADC through a passive switch. The internal 2.56V 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 impedant voltmeter. Note that V_{REF} is a high impedant 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 2.56V 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.

If differential channels are used, the selected reference should not be closer to AVCC than indicated in [Table 28-17 on page 338](#).

23.7 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:

1. 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.
2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
3. 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. If the ADC is enabled in such sleep modes and the user wants to perform differential conversions, the user is advised to switch the ADC off and on after waking up from sleep to prompt an extended conversion to get a valid result.

23.7.1 Analog Input Circuitry

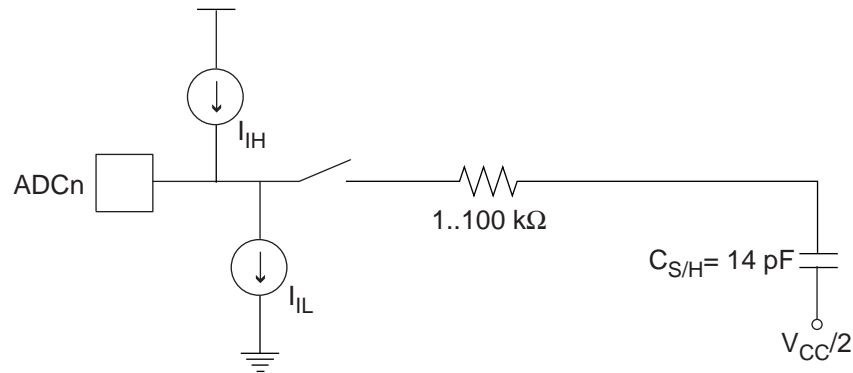
The Analog Input Circuitry for single ended channels is illustrated in [Figure 23-8 on page 251](#). An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the 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 impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

If differential gain channels are used, the input circuitry looks somewhat different, although source impedances of a few hundred k Ω or less is recommended.

Signal components higher than the Nyquist frequency ($f_{\text{ADC}}/2$) 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

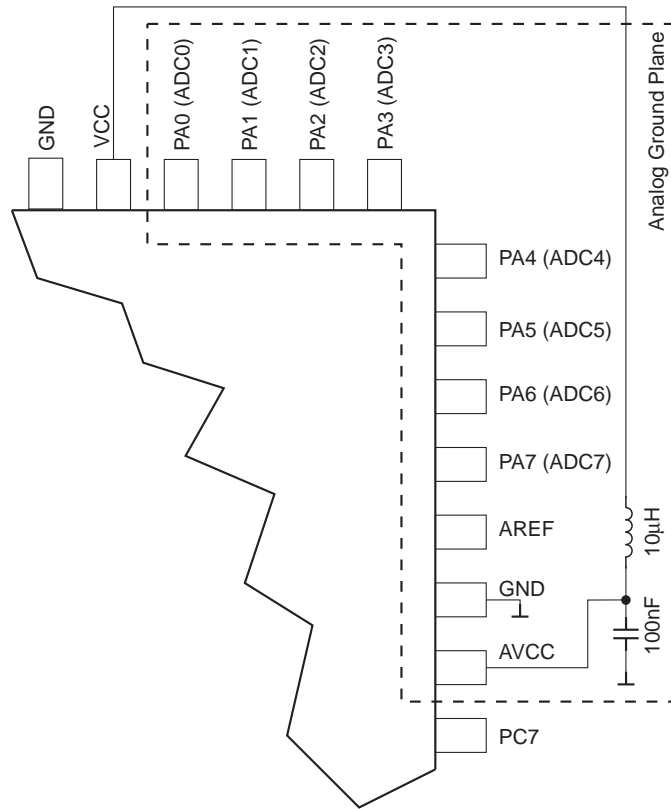


23.7.2 Analog Noise Canceling Techniques

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:

1. 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.
2. The AVCC pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in [Figure 23-9](#).
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Figure 23-9. ADC Power Connections



23.7.3 Offset Compensation Schemes

The gain stage has a built-in offset cancellation circuitry that nulls the offset of differential measurements as much as possible. The remaining offset in the analog path can be measured directly by selecting the same channel for both differential inputs. This offset residue can be then subtracted in software from the measurement results. Using this kind of software based offset correction, offset on any channel can be reduced below one LSB.

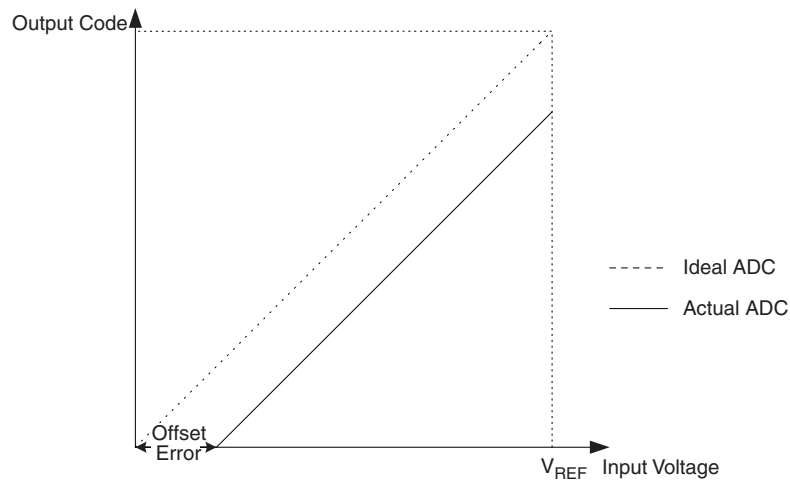
23.7.4 ADC Accuracy Definitions

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:

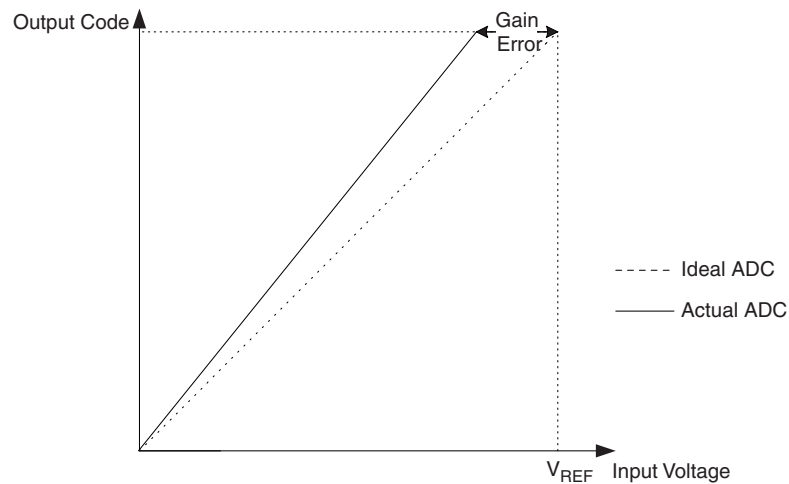
- **Offset:** The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5LSB). Ideal value: 0LSB

Figure 23-10. Offset error



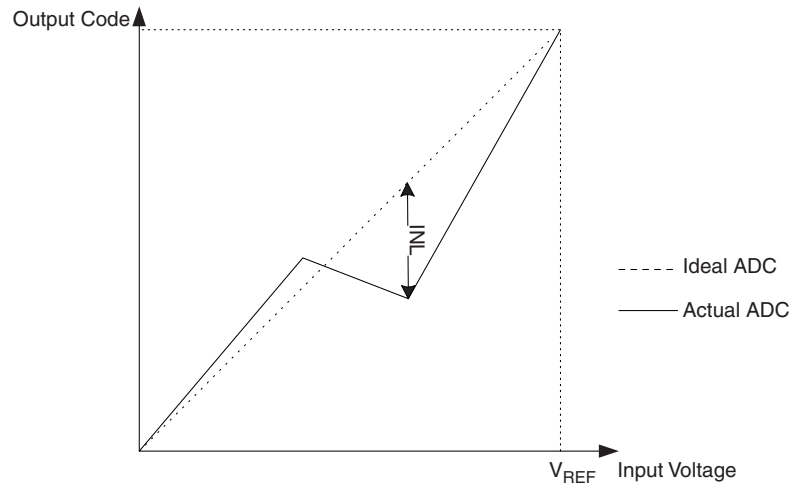
- **Gain Error:** After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5LSB below maximum). Ideal value: 0LSB

Figure 23-11. Gain error



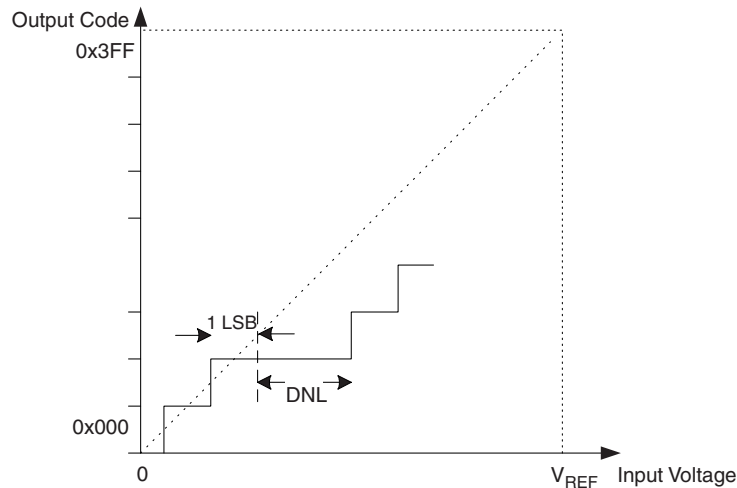
- **Integral Non-linearity (INL):** After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB

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 (1LSB). Ideal value: 0LSB

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 (1LSB wide) will code to the same value. Always $\pm 0.5\text{LSB}$
- **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: $\pm 0.5\text{LSB}$

23.8 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

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see [Table 23-3 on page 257](#) and [Table 23-4 on page 257](#)). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, GAIN the selected gain factor, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the results, it is sufficient to read the MSB of the result (ADC9 in ADCH). If this bit is one, the result is negative, and if this bit is zero, the result is positive. [Figure 23-14 on page 255](#) shows the decoding of the differential input range.

[Table 23-2 on page 256](#) shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a gain of GAIN and a reference voltage of V_{REF} .

Figure 23-14. Differential measurement range

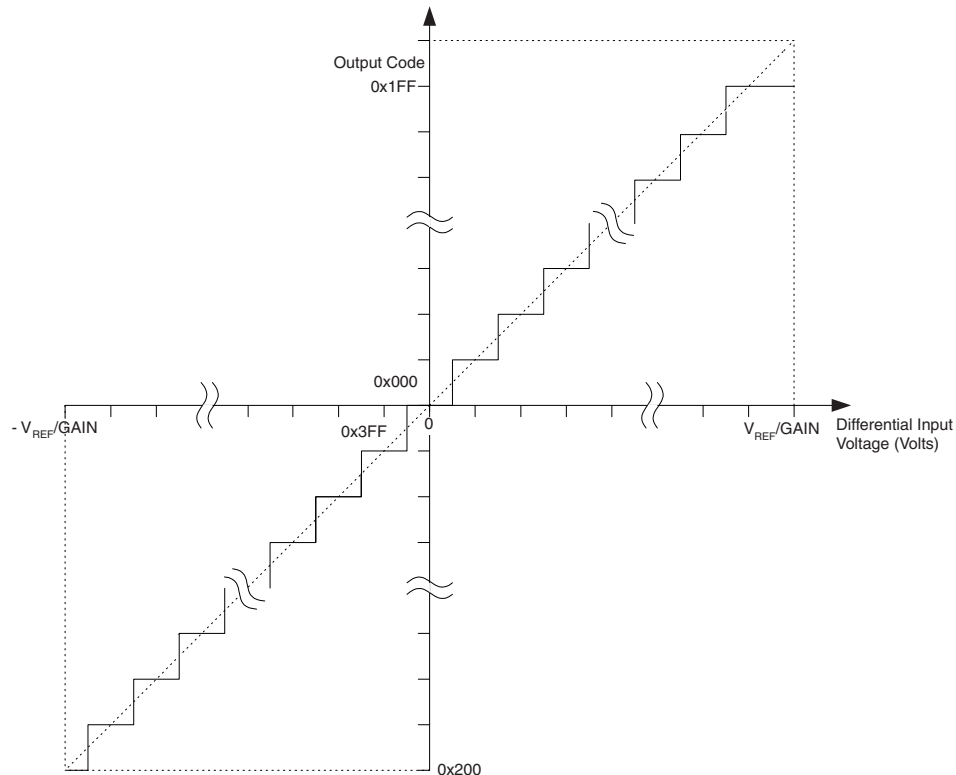


Table 23-2. Correlation between input voltage and output codes

V_{ADCn}	Read code	Corresponding decimal value
$V_{ADCm} + V_{REF}/GAIN$	0x1FF	511
$V_{ADCm} + 0.999V_{REF}/GAIN$	0x1FF	511
$V_{ADCm} + 0.998V_{REF}/GAIN$	0x1FE	510
...
$V_{ADCm} + 0.001V_{REF}/GAIN$	0x001	1
V_{ADCm}	0x000	0
$V_{ADCm} - 0.001V_{REF}/GAIN$	0x3FF	-1
...
$V_{ADCm} - 0.999V_{REF}/GAIN$	0x201	-511
$V_{ADCm} - V_{REF}/GAIN$	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10× gain, 2.56V reference, left adjusted result)

Voltage on ADC3 is 300mV, voltage on ADC2 is 500mV.

ADCR = $512 \times 10 \times (300 - 500) / 2560 = -400 = 0x270$

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result:

ADCL = 0x70, ADCH = 0x02.

23.9 Register description

23.9.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0									
(0x7C)	<table><tr><td>REFS1</td><td>REFS0</td><td>ADLAR</td><td>MUX4</td><td>MUX3</td><td>MUX2</td><td>MUX1</td><td>MUX0</td></tr></table>								REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0										
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W									
Initial Value	0	0	0	0	0	0	0	0									

• Bit 7:6 – 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 Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Note: If differential channels are selected, only 2.56V should be used as Internal Voltage Reference.

• 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 ["ADCL and ADCH – The ADC Data Register" on page 259](#).

• Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See [Table 23-4 on page 257](#) 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 and gain selections

MUX4..0	Single ended input	Positive differential input	Negative differential input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10×

Table 23-4. Input channel and gain selections

MUX4..0	Single ended input	Positive differential input	Negative differential input	Gain
01001	N/A	ADC1	ADC0	10×
01010 ⁽¹⁾		ADC0	ADC0	200×
01011 ⁽¹⁾		ADC1	ADC0	200×
01100		ADC2	ADC2	10×
01101		ADC3	ADC2	10×
01110 ⁽¹⁾		ADC2	ADC2	200×
01111 ⁽¹⁾		ADC3	ADC2	200×
10000		ADC0	ADC1	1×
10001		ADC1	ADC1	1×
10010		ADC2	ADC1	1×
10011		ADC3	ADC1	1×
10100		ADC4	ADC1	1×
10101		ADC5	ADC1	1×
10110		ADC6	ADC1	1×
10111		ADC7	ADC1	1×
11000		ADC0	ADC2	1×
11001		ADC1	ADC2	1×
11010		ADC2	ADC2	1×
11011		ADC3	ADC2	1×
11100		ADC4	ADC2	1×
11101		ADC5	ADC2	1×
11110	1.1V (V _{BG})	N/A		
11111	0 V (GND)			

Note: 1. The differential input channels are not tested for devices in PDIP Package. This feature is only ensured to work for devices in TQFP and VQFN/QFN/MLF Packages.

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 XTAL 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

ADLA

R	Bit	15	14	13	12	11	10	9	8	
	(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
=	(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
		7	6	5	4	3	2	1	0	
0	Read/Write	R	R	R	R	R	R	R	R	
		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	

ADLA

R	Bit	15	14	13	12	11	10	9	8	
	(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	
=	(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCH ADCL
		7	6	5	4	3	2	1	0	
1	Read/Write	R	R	R	R	R	R	R	R	
		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. If differential channels are used, the result is presented in two's complement form.

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 ["ADC Conversion Result" on page 255](#).

23.9.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	ADCSRB
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 5:3 – Reserved**

These bits are reserved for future use in the ATmega164A/164PA/324A/324PA/644A/644PA/1284/1284P. For ensuring compatibility with future devices, these bits must be written 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 ADTS[2:0] 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	7	6	5	4	3	2	1	0	
(0x7E)	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	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 – ADC7D..ADC0D: ADC7:0 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 ADC7:0 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.