Microchip Technology - ATMEGA88-15MT2 Datasheet





Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega88-15mt2

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

4.7 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR[®] CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 4-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 4-4. The Parallel Instruction Fetches and Instruction Executions



Figure 4-5 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 4-5. Single Cycle ALU Operation

	T1	T2	Т3	T4
clk _{CPU}		, 1 1 1 1 1	- 	
Total Execution Time		 	1 	
Register Operands Fetch				
ALU Operation Execute		1 1 1 1 1		
Result Write Back				
				:

4.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the global interrupt enable bit in the status register in order to enable the interrupt. Depending on the program counter value, interrupts may be automatically disabled when boot lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the Section 24. "Memory Programming" on page 234 for details.

The lowest addresses in the program memory space are by default defined as the reset and interrupt vectors. The complete list of vectors is shown in Section 9. "Interrupts" on page 48. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the external interrupt request 0. The interrupt vectors can be moved to the start of the boot flash section by setting the IVSEL bit in the MCU control register (MCUCR). Refer to Section 9. "Interrupts" on page 48 for more information. The reset vector can also be moved to the start of the boot flash section by programming the BOOTRST fuse, see Section 23. "Boot Loader Support – Read-While-Write Self-Programming, ATmega88 and ATmega168" on page 221.

When an interrupt occurs, the global interrupt enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a return from interrupt instruction – RETI – is executed.

Atmel

7. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR[®] provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the five sleep modes, the SE bit in SMCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the SMCR register select which sleep mode (idle, ADC noise reduction, power-down, power-save, or standby) will be activated by the SLEEP instruction. See Table 7-1 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the reset vector.

Figure 6-1 on page 23 presents the different clock systems in the Atmel[®] ATmega88/168, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

7.1 Sleep Mode Control Register – SMCR

The sleep mode control register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	0	

• Bits 7..4 Res: Reserved Bits

These bits are unused bits in the Atmel ATmega88/168, and will always read as zero.

• Bits 3..1 - SM2..0: Sleep Mode Select Bits 2, 1, and 0

These bits select between the five available sleep modes as shown in Table 7-1.

SM2	SM1	SM0	Sleep Mode
0	0	0	ldle
0	0	1	ADC noise reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby ⁽¹⁾
1	1	1	Reserved

Table 7-1. Sleep Mode Select

Note: 1. Standby mode is only recommended for use with external crystals or resonators.

• Bit 0 – SE: Sleep Enable

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the sleep enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

10. I/O-Ports

10.1 Introduction

All AVR[®] ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and ground as indicated in Figure 10-1. Refer to Section 25. "Electrical Characteristics" on page 252 for a complete list of parameters.

Figure 10-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in port B, here documented generally as PORTxn. The physical I/O registers and bit locations are listed in Section 10.4 "Register Description for I/O Ports" on page 69.

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

Using the I/O port as general digital I/O is described in Section 10.2 "Ports as General Digital I/O" on page 56. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 10.3 "Alternate Port Functions" on page 60. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

• Bit 1 - PCIF1: Pin Change Interrupt Flag 1

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

• Bit 0 - PCIF0: Pin Change Interrupt Flag 0

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

11.6 Pin Change Mask Register 2 – PCMSK2

Bit	7	6	5	4	3	2	1	0	_
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7..0 – PCINT23..16: Pin Change Enable Mask 23..16

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

11.7 Pin Change Mask Register 1 – PCMSK1

Bit	7	6	5	4	3	2	1	0	_
	_	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - Res: Reserved Bit

This bit is an unused bit in the Atmel[®] ATmega88/168, and will always read as zero.

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

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

11.8 Pin Change Mask Register 0 – PCMSK0

Bit	7	6	5	4	3	2	1	0	
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7..0 – PCINT7..0: Pin Change Enable Mask 7..0

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



12.6.3 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 12-6. 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 12-6. 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 12-6 on page 86). 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:

$$f_{OCnxPWM} = \frac{f_{\text{clk}_I/O}}{N \cdot 256}$$

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.)

Atmel

12.8 8-bit Timer/Counter Register Description

12.8.1 Timer/Counter Control Register A – TCCR0A

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A 0	COM0B1	COM0B 0	_	-	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 12-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 12-2. Compare Output Mode, non-F will 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 12-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 12-3.	Compare	Output	Mode,	Fast	PWM	Mode ⁽¹⁾
Table 12-J.	Compare	Output	woue,	ιαδι		MOUE

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 TOP
1	1	Set OC0A on compare match, clear OC0A at TOP
0 1 1	1 0 1	WGM02 = 0: normal port operation, OC0A disconnected. WGM02 = 1: toggle OC0A on compare match. Clear OC0A on compare match, set OC0A at TOP Set OC0A on compare match, clear OC0A at TOP

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 12.6.3 "Fast PWM Mode" on page 81 for more details.

Table 12-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 12-4.	Compare Output Mode, Phas	se Correct PWM Mode ⁽¹⁾
-------------	---------------------------	------------------------------------

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

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 14.8.4 "Phase Correct PWM Mode" on page 106 for more details.

The double buffered output compare registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pin (OC1A/B). See Section 14.6 "Output Compare Units" on page 100. The compare match event will also set the compare match flag (OCF1A/B) which can be used to generate an output compare interrupt request.

The input capture register can capture the Timer/Counter value at a given external (edge triggered) event on either the input capture pin (ICP1) or on the analog comparator pins (see Section 20. "Analog Comparator" on page 201) The input capture unit includes a digital filtering unit (noise canceler) for reducing the chance of capturing noise spikes.

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

14.1.2 Definitions

The following definitions are used extensively throughout the section:

Parameter	Definitions
BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
ТОР	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 OCR1A or ICR1 register. The assignment is dependent of the mode of operation.

Table 14-1. Definitions

14.2 Accessing 16-bit Registers

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

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

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



Figure 14-13 shows the same timing data, but with the prescaler enabled.





14.10 16-bit Timer/Counter Register Description

14.10.1 Timer/Counter1 Control Register A – TCCR1A



• Bit 7:6 - COM1A1:0: Compare Output Mode for Channel A

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

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

When the OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent of the WGM13:0 bits setting. Table 14-2 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

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

15.6.1 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.

15.6.2 Clear Timer on Compare Match (CTC) Mode

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

The timing diagram for the CTC mode is shown in Figure 15-5. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and then counter (TCNT2) is cleared.



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 $f_{OC2A} = f_{elk \mid UO}/2$ when OCR2A is set to zero (0x00). The waveform frequency is defined by the following equation:

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

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.

Table 17-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 17-1.	Equations for	Calculating	Baud Rate	Register	Setting
-------------	---------------	-------------	-----------	----------	---------

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

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

BAUD Baud rate (in bits per second, bps)

fosc System oscillator clock frequency

UBRRn Contents of the UBRRnH and UBRRnL registers, (0-4095)

Some examples of UBRRn values for some system clock frequencies are found in Table 17-9 on page 163.

17.2.2 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.

17.2.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 17-2 on page 146 for details.

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

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

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

17.2.4 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.





19.3.5 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 19-6 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 19-6. Typical Data Transmission

19.4 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, i.e. 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.



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.

19.5 Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 19-9. All registers drawn in a thick line are accessible through the AVR[®] data bus.

Figure 19-9. Overview of the TWI Module



19.5.1 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.



Figure 21-1. Analog to Digital Converter Block Schematic 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 1 LSB. Optionally, AV_{CC} or an internal 1.1V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. The ADC is enabled by setting the ADC enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

Atmel





Figure 21-7. ADC Timing Diagram, Free Running Conversion



Table 21-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5

21.5 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:

- a. 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.
- b. Enter ADC noise reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
- c. 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.

21.5.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 21-8 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 $10k\Omega$ 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.

Signal components higher than the nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 21-8. Analog Input Circuitry



23.7.8 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to flash. Reading the fuses and lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR register and verifies that the bit is cleared before writing to the SPMCSR register.

23.7.9 Reading the Fuse and Lock Bits from Software

It is possible to read both the fuse and lock bits from software. To read the lock bits, load the Z-pointer with 0x0001 and set the BLBSET and SELFPRGEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the BLBSET and SELFPRGEN bits are set in SPMCSR, the value of the lock bits will be loaded in the destination register. The BLBSET and SELFPRGEN bits will auto-clear upon completion of reading the lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLBSET and SELFPRGEN are cleared, LPM will work as described in the Instruction set manual.



The algorithm for reading the fuse low byte is similar to the one described above for reading the lock bits. To read the fuse low byte, load the Z-pointer with 0x0000 and set the BLBSET and SELFPRGEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the BLBSET and SELFPRGEN bits are set in the SPMCSR, the value of the fuse low byte (FLB) will be loaded in the destination register as shown below. Refer to Table 24-4 on page 235 for a detailed description and mapping of the fuse low byte.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the fuse high byte, load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SELFPRGEN bits are set in the SPMCSR, the value of the fuse high byte (FHB) will be loaded in the destination register as shown below. Refer to Table 24-5 on page 236 for detailed description and mapping of the fuse high byte.



When reading the extended fuse byte, load 0x0002 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SELFPRGEN bits are set in the SPMCSR, the value of the extended fuse byte (EFB) will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	-	-	EFB3	EFB2	EFB1	EFB0

Fuse and lock bits that are programmed, will be read as zero. Fuse and lock bits that are unprogrammed, will be read as one.

23.7.10 Preventing Flash Corruption

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

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

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

- 1. If there is no need for a boot loader update in the system, program the boot loader lock bits to prevent any boot loader software updates.
- 2. 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 operating voltage matches the detection level. If not, 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.
- Keep the AVR[®] core in power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR register and thus the flash from unintentional writes.

23.7.11 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 23-5 shows the typical programming time for flash accesses from the CPU.

Table 23-5. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7ms	4.5ms

23.7.12 Simple Assembly Code Example for a Boot Loader

```
;-the routine writes one page of data from RAM to Flash
      ; the first data location in RAM is pointed to by the Y pointer
      ; the first data location in Flash is pointed to by the Z-pointer
      ;-error handling is not included
      ;-the routine must be placed inside the Boot space
      ; (at least the Do_spm sub routine). Only code inside NRWW section can
      ; be read during Self-Programming (Page Erase and Page Write).
      ;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
      ; loophi (r25), spmcrval (r20)
      ; storing and restoring of registers is not included in the routine
      ; register usage can be optimized at the expense of code size
      ;-It is assumed that either the interrupt table is moved to the Boot
      ; loader section or that the interrupts are disabled.
.eau
      PAGESIZEB = PAGESIZE*2 ; PAGESIZEB is page size in BYTES, not words
.org SMALLBOOTSTART
Write_page:
      ;Page Erase
      ldi spmcrval, (1<<PGERS) | (1<<SELFPRGEN)
      call Do_spm
      ;
            re-enable the RWW section
            spmcrval, (1<<RWWSRE) | (1<<SELFPRGEN)</pre>
      ldi
      call Do_spm
             transfer data from RAM to Flash page buffer
      ;
      ldi
             looplo, low(PAGESIZEB) ;init loop variable
      ldi
            loophi, high(PAGESIZEB) ;not required for PAGESIZEB<=256</pre>
Wrloop:
      ld r0, Y+
      ld
           r1, Y+
      ldi
            spmcrval, (1<<SELFPRGEN)
      call Do_spm
      adiw ZH:ZL, 2
      sbiw loophi:looplo, 2 ;use subi for PAGESIZEB<=256
      brne Wrloop
```



Table 24-11. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega88	4Kwords (8Kbytes)	32 words	PC[4:0]	128	PC[11:5]	11
ATmega168	8Kwords (16Kbytes)	64 words	PC[5:0]	128	PC[12:6]	12

Table 24-12. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATmega88	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8
ATmega168	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

24.6 Serial Programming Pin Mapping

Table 24-13. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB3	I	Serial data in
MISO	PB4	0	Serial data out
SCK	PB5	I	Serial clock

24.7 Parallel Programming

24.7.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

- 1. Apply 4.5 5.5V between V_{CC} and GND.
- 2. Set RESET to "0" and toggle XTAL1 at least six times.
- 3. Set the prog_enable pins listed in Table 24-8 on page 238 to "0000" and wait at least 100ns.
- 4. Apply 11.5 12.5V to RESET. Any activity on prog_enable pins within 100 ns after +12V has been applied to RESET, will cause the device to fail entering programming mode.
- 5. Wait at least 50µs before sending a new command.

24.7.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE fuse is programmed) and flash after a chip erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in flash or 256 byte EEPROM. This consideration also applies to signature bytes reading.

Figure 24-4. Programming the EEPROM Waveforms



24.7.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to Section 24.7.4 "Programming the Flash" on page 240 for details on command and address loading):

- 1. A: Load command "0000 0010".
- 2. G: Load address high byte (0x00 0xFF).
- 3. B: Load address low byte (0x00 0xFF).
- 4. Set OE to "0", and BS1 to "0". The flash word low byte can now be read at DATA.
- 5. Set BS1 to "1". The flash word high byte can now be read at DATA.
- 6. Set OE to "1".

24.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Section 24.7.4 "Programming the Flash" on page 240 for details on command and address loading):

- 1. A: Load command "0000 0011".
- 2. G: Load address high byte (0x00 0xFF).
- 3. B: Load address low byte (0x00 0xFF).
- 4. Set OE to "0", and BS1 to "0". The EEPROM data byte can now be read at DATA.
- 5. Set OE to "1".

24.7.8 Programming the Fuse Low Bits

The algorithm for programming the fuse low bits is as follows (refer to Section 24.7.4 "Programming the Flash" on page 240 for details on command and data loading):

- 1. A: Load command "0100 0000".
- 2. C: Load data low byte. Bit n = "0" programs and bit n = "1" erases the fuse bit.
- 3. Give WR a negative pulse and wait for RDY/BSY to go high.

31.	Packaging Information	282
32.	Errata ATmega88	284
33.	Errata ATmega168	285
34.	Table of Contents	286