Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

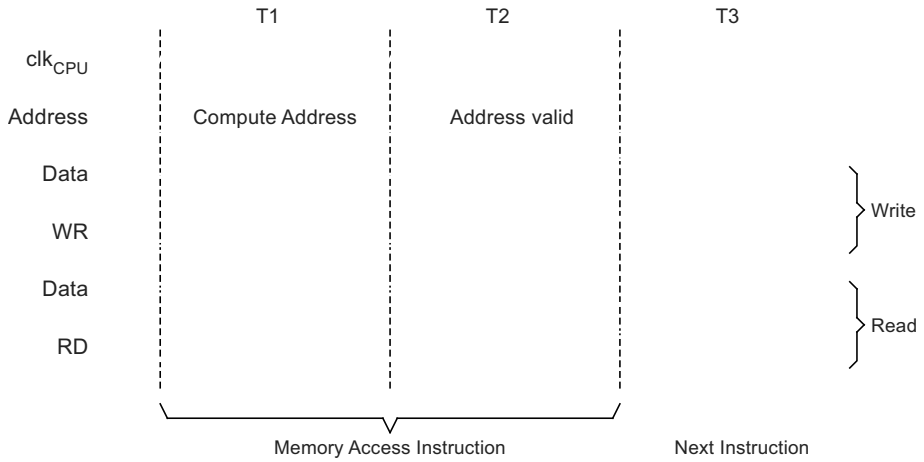
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, 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 ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega88-15at1

5.2.1 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 5-4.

Figure 5-4. On-chip Data SRAM Access Cycles



5.3 EEPROM Data Memory

The Atmel ATmega48/88/168 contains 256/512/512 bytes 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.

Section 25. “Memory Programming” on page 242 contains a detailed description on EEPROM programming in SPI or parallel programming mode.

5.3.1 EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 5-2 on page 19. 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 5.3.5 “Preventing EEPROM Corruption” on page 21 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.

5.3.2 The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	**EEAR8**	**EEARH**
	EEAR7	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	**EEARL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	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	X	
	X	X	X	X	X	X	X	X	

The most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATmega168 is:

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

When the BOOTRST fuse is unprogrammed, the boot section size set to 2K bytes and the IVSEL bit in the MCUCR register is set before any interrupts are enabled, the most typical and general program setup for the reset and interrupt vector addresses in ATmega168 is:

```
Address      Labels Code                Comments
0x0000              RESET: ldi          r16,high(RAMEND); Main program start
0x0001              out    SPH,r16      ; Set Stack Pointer to top of RAM
0x0002              ldi    r16,low(RAMEND)
0x0003              out    SPL,r16
0x0004              sei                 ; Enable interrupts
0x0005              <instr>  xxx
;
.org 0xC02
0x1C02              jmp    EXT_INT0     ; IRQ0 Handler
0x1C04              jmp    EXT_INT1     ; IRQ1 Handler
...          ...    ...                 ;
0x1C32              jmp    SPM_RDY      ; Store Program Memory Ready Handler
```

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

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

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

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

11.4 Pin Change Interrupt Control Register - PCICR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega48/88/168, and will always read as zero.

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

When the PCIE2 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI2 interrupt vector. PCINT23..16 pins are enabled individually by the PCMSK2 register.

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

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT14..8 pins are enabled individually by the PCMSK1 register.

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

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 interrupt vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

11.5 Pin Change Interrupt Flag Register - PCIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega48/88/168, and will always read as zero.

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

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

Table 12-9. 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)
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.

12.8.3 Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	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.

12.8.4 Output Compare Register A – OCR0A

Bit	7	6	5	4	3	2	1	0	
	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.

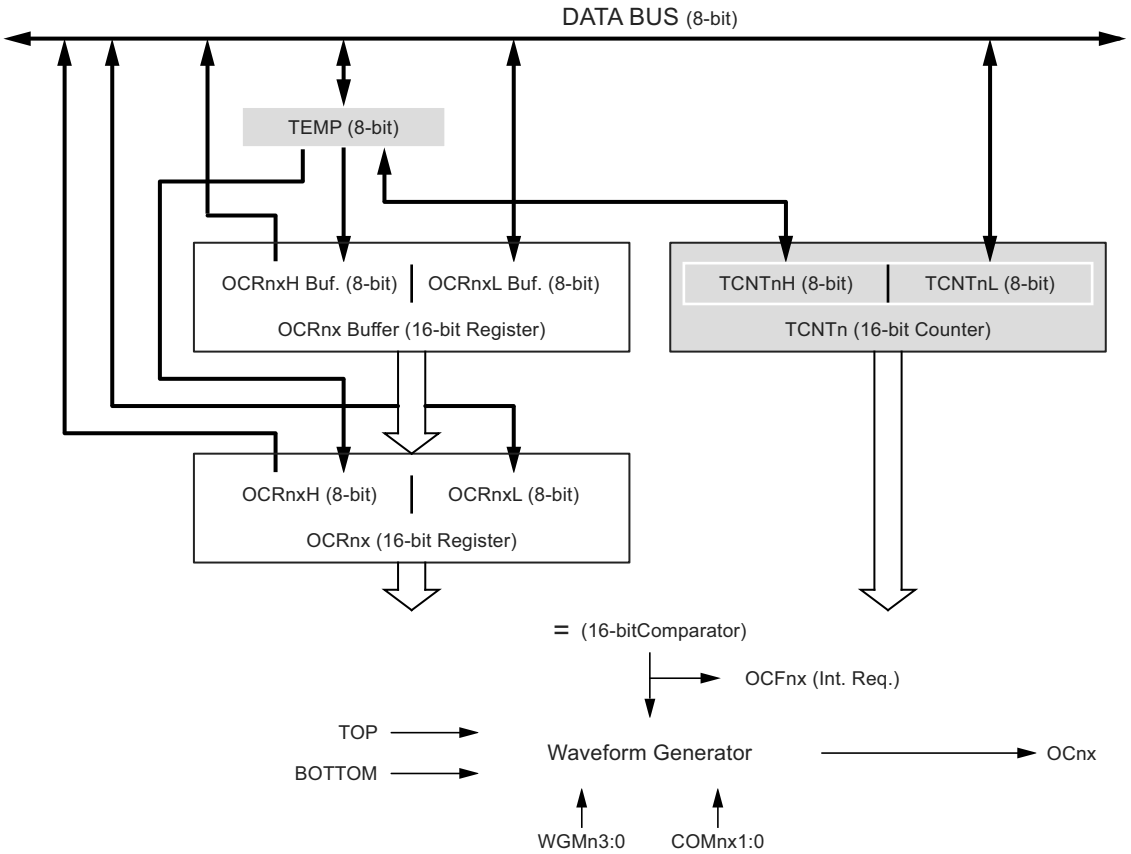
12.8.5 Output Compare Register B – OCR0B

Bit	7	6	5	4	3	2	1	0	
	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.

Figure 14-4 shows a block diagram of the output compare unit. The small “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 14-4. Output Compare Unit, Block Diagram



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

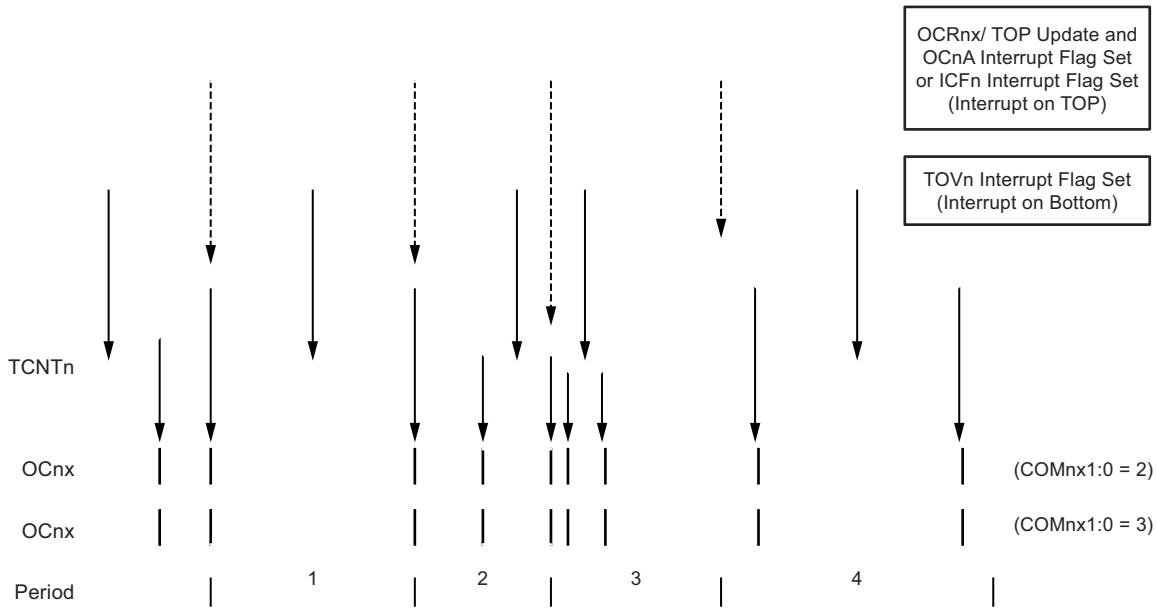
The OCR1x register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x buffer register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (buffer or compare) register is only changed by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

For more information of how to access the 16-bit registers refer to Section 14.2 “Accessing 16-bit Registers” on page 96.

14.6.1 Force Output Compare

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

Figure 14-8. Phase Correct PWM Mode, Timing Diagram



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

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x registers are written. As the third period shown in Figure 14-8 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 OCR1x register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

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

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table on page 114). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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

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

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 compete 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.

17.5.4 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.

17.5.5 Disabling the Transmitter

The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit shift register and transmit buffer register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxDn pin.

17.6 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.

17.6.1 Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCKn clock, and shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, i.e., a complete serial frame is present in the receive shift register, the contents of the shift register will be moved into the receive buffer. The receive buffer can then be read by reading the UDRn I/O location.

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

Assembly Code Example(1)

```
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. 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 "SBRS", "SBRC", "SBR", and "CBR".

19. 2-wire Serial Interface

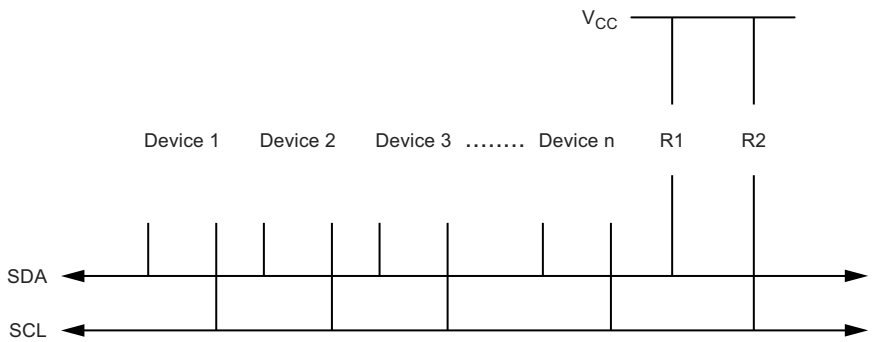
19.1 Features

- 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

19.2 2-wire Serial Interface Bus Definition

The 2-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 19-1. TWI Bus Interconnection



19.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 19-1. 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 PRTWI bit in Section 7.7.1 “Power Reduction Register - PRR” on page 35 must be written to zero to enable the 2-wire serial interface.

19.5.2 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:

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

- TWBR = value of the TWI bit rate register.
- Prescaler value = value of the prescaler, see Table 19-2 on page 184.

Note: TWBR should be 10 or higher if the TWI operates in master mode. If TWBR is lower than 10, the master may produce an incorrect output on SDA and SCL for the reminder of the byte. The problem occurs when operating the TWI in master mode, sending start + SLA + R/W to a slave (a slave does not need to be connected to the bus for the condition to happen).

19.5.3 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.

19.5.4 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. If another interrupt (e.g., INT0) occurs during TWI power-down address match and wakes up the CPU, the TWI aborts operation and return to it’s idle state. If this cause any problems, ensure that TWI address match is the only enabled interrupt when entering power-down.

19.5.5 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.

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 7 bits are the address to which the 2-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 19-6 on page 196. 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 2-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 2-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 2-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 2-wire serial interface data register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.

21.6 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 21-2 and Table 21-3 on page 218). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

21.6.1 ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	**REFS0**	**ADLAR**	**–**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 21-2. 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 21-2. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC data register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC data register immediately, regardless of any ongoing conversions. For a complete description of this bit, see Section 21.6.3 “The ADC Data Register – ADCL and ADCH” on page 219.

- **Bit 4 – Res: Reserved Bit**

This bit is an unused bit in the Atmel® ATmega48/88/168, and will always read as zero.

- **Bits 3:0 – MUX3:0: Analog Channel Selection Bits**

The value of these bits selects which analog inputs are connected to the ADC. See Table 21-3 on page 218 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Figure 24-1. Read-While-Write versus No Read-While-Write

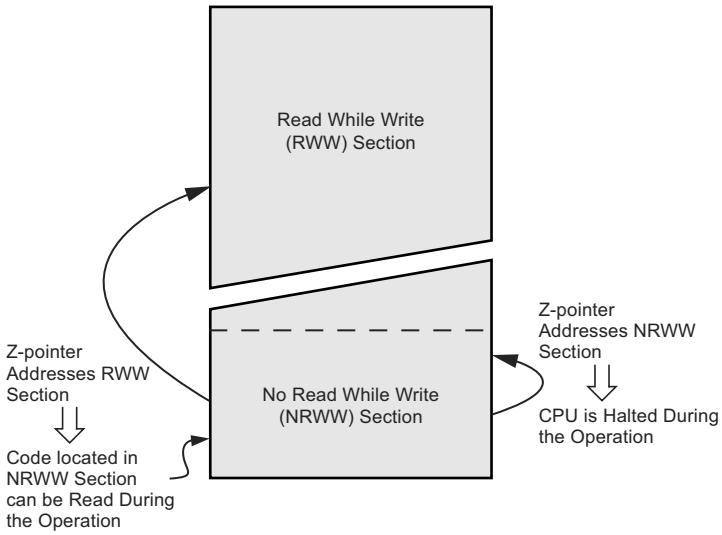
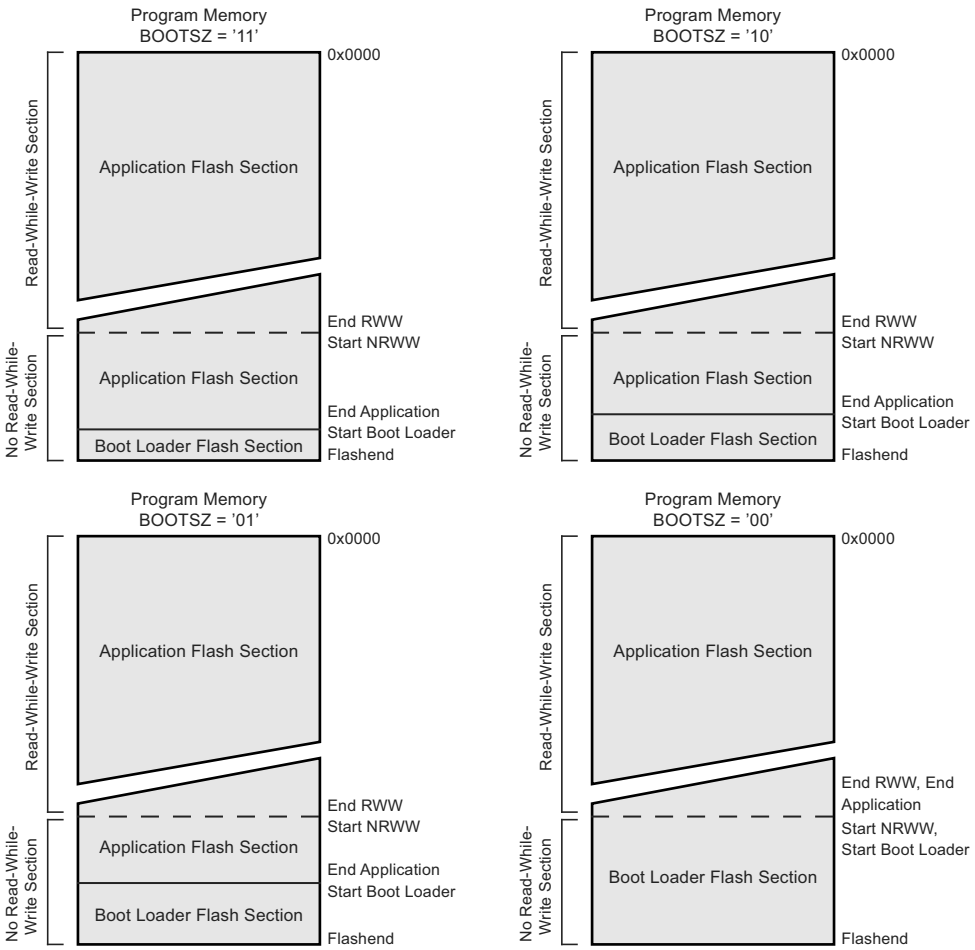


Figure 24-2. Memory Sections



Note: 1. The parameters in the figure above are given in Table 24-6 on page 240.

24.5 “Entering the Boot Loader Program

Entering the boot loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the boot reset fuse can be programmed so that the reset vector is pointing to the boot flash start address after a reset. In this case, the boot loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the boot reset fuse is programmed, the reset vector will always point to the boot loader reset and the fuse can only be changed through the serial or parallel programming interface.

Table 24-4. Boot Reset Fuse(1)

BOOTRST	Reset Address
1	Reset vector = application reset (address 0x0000)
0	Reset vector = boot loader reset (see Table 24-6 on page 240)

Note: 1. “1” means unprogrammed, “0” means programmed

24.5.1 Store Program Memory Control and Status Register – SPMCSR

The store program memory control and status register contains the control bits needed to control the boot loader operations.

Bit	7	6	5	4	3	2	1	0	
	SPMIE	**RWWSB**	–	RWWSRE	**BLBSET**	**PGWRT**	**PGERS**	**SELFPRGEN**	**SPMCSR**
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPMIE: SPM Interrupt Enable**

When the SPMIE bit is written to one, and the I-bit in the status register is set (one), the SPM ready interrupt will be enabled. The SPM ready interrupt will be executed as long as the SELFPRGEN bit in the SPMCSR register is cleared.

- **Bit 6 – RWWSB: Read-While-Write Section Busy**

When a self-programming (page erase or page write) operation to the RWW section is initiated, the RWWSB will be set (one) by hardware. When the RWWSB bit is set, the RWW section cannot be accessed. The RWWSB bit will be cleared if the RWWSRE bit is written to one after a self-programming operation is completed. Alternatively the RWWSB bit will automatically be cleared if a page load operation is initiated.

- **Bit 5 – Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATmega48/88/168 and always read as zero.

- **Bit 4 – RWWSRE: Read-While-Write Section Read Enable**

When programming (page erase or page write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SELFPRGEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SELFPRGEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the flash is busy with a page erase or a page write (SELFPRGEN is set). If the RWWSRE bit is written while the flash is being loaded, the flash load operation will abort and the data loaded will be lost.

- **Bit 3 – BLBSET: Boot Lock Bit Set**

If this bit is written to one at the same time as SELFPRGEN, the next SPM instruction within four clock cycles sets boot lock bits and memory lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the lock bit set, or if no SPM instruction is executed within four clock cycles.

An LPM instruction within three cycles after BLBSET and SELFPRGEN are set in the SPMCSR register, will read either the lock bits or the fuse bits (depending on Z0 in the Z-pointer) into the destination register. See Section 24.7.9 “Reading the Fuse and Lock Bits from Software” on page 237 for details.

26.2 DC Characteristics (Continued)

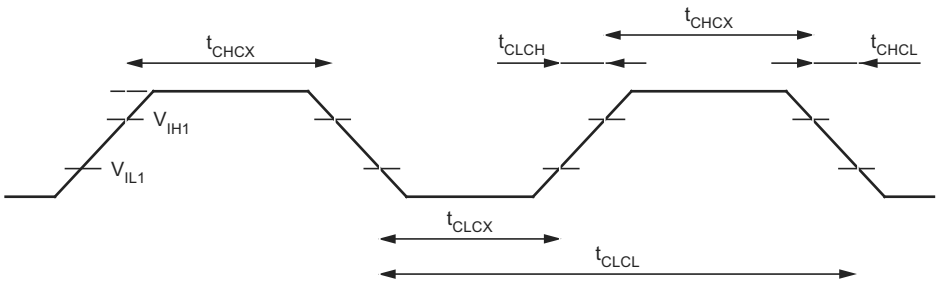
T_A = –40°C to +125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Parameter	Condition	Symbol	Min.[5]	Typ.	Max.[5]	Unit
Power supply current[6]	Active 4MHz, V_{CC} = 3V (ATmega48/88/168L)	I_{CC}		1.8	3.0	mA
	Active 8MHz, V_{CC} = 5V (ATmega48/88/168)			6.0	10	mA
	Active 15MHz, V_{CC} = 5V (ATmega48/88/168)			10.0	16	mA
	Idle 4MHz, V_{CC} = 3V (ATmega48/88/168V)			0.4	1	mA
	Idle 8MHz, V_{CC} = 5V (ATmega48/88/168L)			1.4	2.4	mA
	Idle 15MHz, V_{CC} = 5V (ATmega48/88/168)			2.8	4	mA
Power-down mode	WDT enabled, V_{CC} = 3V			8	30	µA
	WDT enabled, V_{CC} = 5V			12.6	50	µA
	WDT disabled, V_{CC} = 3V			5	24	µA
	WDT disabled, V_{CC} = 5V			6.6	36	µA
Analog comparator input offset voltage	V_{CC} = 5V V_{in} = V_{CC}/2	V_{ACIO}		10	40	mV
Analog comparator input leakage current	V_{CC} = 5V V_{in} = V_{CC}/2	I_{ACLK}	–50		50	nA
Analog comparator propagation delay	V_{CC} = 4.5V	t_{ACID}		140		ns

Notes:
1. “Max” means the highest value where the pin is guaranteed to be read as low
2. “Min” means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   Atmel ATmega48:
   1] The sum of all IOL, for ports C0 - C5, should not exceed 70mA.
   2] The sum of all IOL, for ports C6, D0 - D4, should not exceed 70mA.
   3] The sum of all IOL, for ports B0 - B7, D5 - D7, should not exceed 70mA.
   ATmega88/168:
   1] The sum of all IOL, for ports C0 - C5, should not exceed 100mA.
   2] The sum of all IOL, for ports C6, D0 - D4, should not exceed 100mA.
   3] The sum of all IOL, for ports B0 - B7, D5 - D7, should not exceed 100mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega48:
   1] The sum of all IOH, for ports C0 - C5, should not exceed 70mA.
   2] The sum of all IOH, for ports C6, D0 - D4, should not exceed 70mA.
   3] The sum of all IOH, for ports B0 - B7, D5 - D7, should not exceed 70mA.
   ATmega88/168:
   1] The sum of all IOH, for ports C0 - C5, should not exceed 100mA.
   2] The sum of all IOH, for ports C6, D0 - D4, should not exceed 100mA.
   3] The sum of all IOH, for ports B0 - B7, D5 - D7, should not exceed 100mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. All DC characteristics contained in this datasheet are based on actual ATmega88 microcontrollers characterization.
6. Values with Section 7.7.1 “Power Reduction Register - PRR” on page 35 enabled (0xEF).

26.3 External Clock Drive Waveforms

Figure 26-1. External Clock Drive Waveforms



26.4 External Clock Drive

Table 26-1. External Clock Drive

Parameter	Symbol	V_{CC}=2.7 to 5.5V		V_{CC}=4.5 to 5.5V		Unit
		Min.	Max.	Min.	Max.	
Oscillator frequency	$1/t_{CLCL}$	0	8	0	16	MHz
Clock period	t_{CLCL}	125		62.5		ns
High time	t_{CHCX}	50		25		ns
Low time	t_{CLCX}	50		25		ns
Rise time	t_{CLCH}		1.6		0.5	µs
Fall time	t_{CHCL}		1.6		0.5	µs
Change in period from one clock cycle to the next	Dt_{CLCL}		2		2	%

26.5 Maximum Speed versus V_{CC}

Maximum frequency is dependent on $V_{CC.}$ As shown in Figure 26-2, the maximum frequency versus V_{CC} curve is linear between 2.7V < V_{CC} < 4.5V.

Figure 26-2. Maximum Frequency versus V_{CC}, ATmega48/88/168

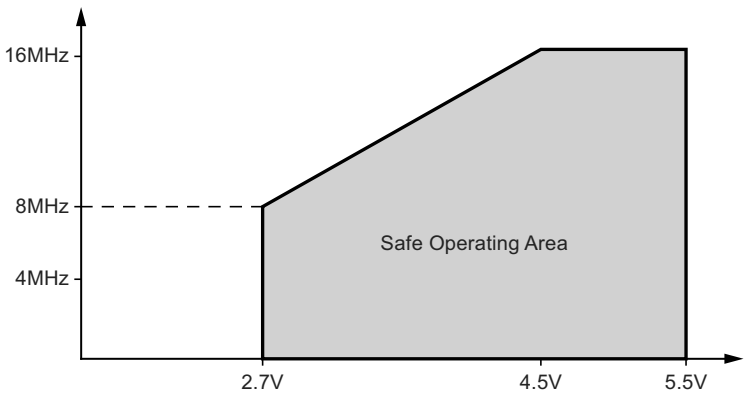
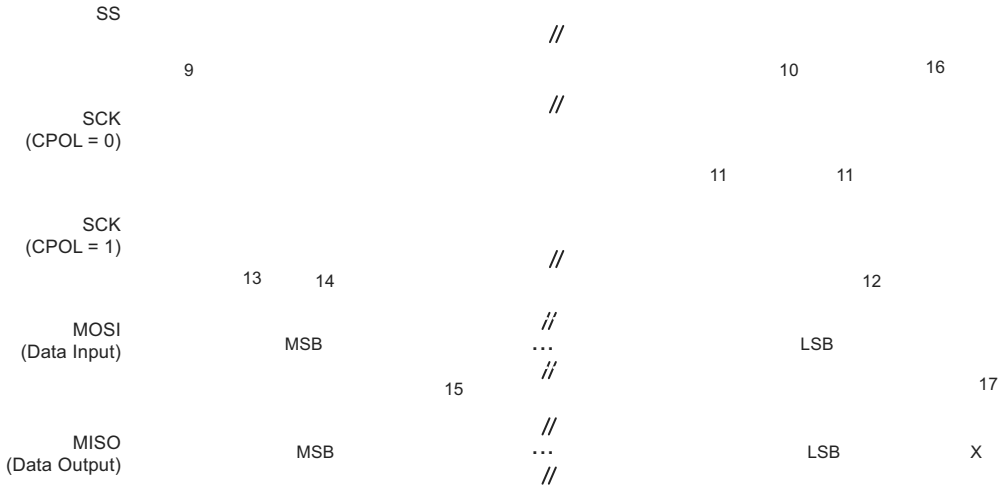


Figure 27-3. SPI Interface Timing Requirements (Slave Mode)



27.2 ADC Characteristics

Table 27-3. ADC Characteristics

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Resolution				10		Bits
Absolute accuracy (including INL, DNL, quantization error, gain and offset error)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			2	3.5	LSB
	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz noise reduction mode			2	3.5	LSB
Integral non-linearity (INL)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			0.6	2.5	LSB
Differential non-linearity (DNL)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			0.30	1.0	LSB
Gain error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		–3.5	–1.3	3.5	LSB
Offset error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			1.8	3.5	LSB
Conversion time	Free running conversion		13 cycles			µs
Clock frequency			50		200	kHz
Analog supply voltage		AV_{CC}	V_{CC} – 0.3		V_{CC} + 0.3	V
Reference voltage		V_{REF}	1.0		AV_{CC}	V
Input voltage		V_{IN}	GND		V_{REF}	V
Input bandwidth				38.5		kHz
Internal voltage reference		V_{INT}	1.0	1.1	1.2	V
Reference input resistance		R_{REF}	25.6	32	38.4	kΩ
Analog input resistance		R_{AIN}		100		MΩ

Figure 28-9. Output High Voltage versus Output High Current (V_{CC} = 3V)

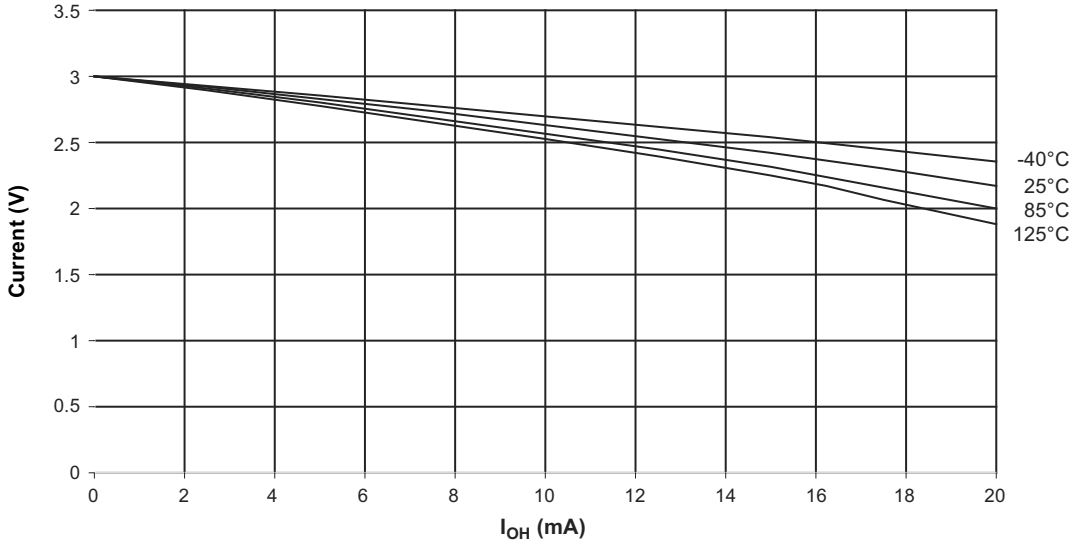
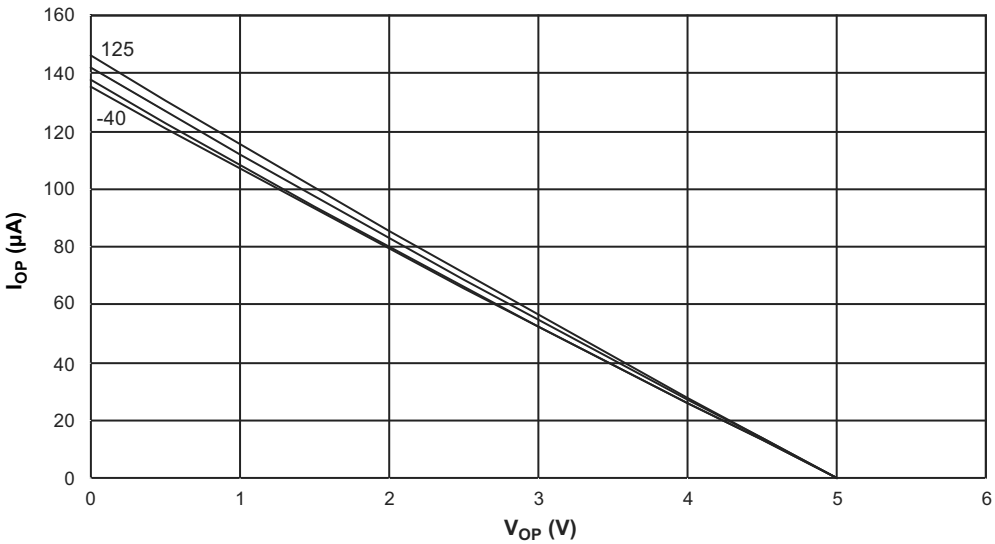


Figure 28-10. Reset Pull-up Resistor Current versus Reset Pin Voltage (V_{CC} = 5V)



29. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	Reserved	–	–	–	–	–	–	–	–	
(0xD1)	Reserved	–	–	–	–	–	–	–	–	
(0xD0)	Reserved	–	–	–	–	–	–	–	–	
(0xCF)	Reserved	–	–	–	–	–	–	–	–	
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	Reserved	–	–	–	–	–	–	–	–	
(0xCA)	Reserved	–	–	–	–	–	–	–	–	
(0xC9)	Reserved	–	–	–	–	–	–	–	–	
(0xC8)	Reserved	–	–	–	–	–	–	–	–	
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	UDR0	USART I/O data register								161
(0xC5)	UBRR0H	–	–	–	–	USART baud rate register high				164
(0xC4)	UBRR0L	USART Baud Rate Register Low								164
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 /UDORD0	UCSZ00 / UCPHA0	UCPOL0	163/173
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	162
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	161
(0xBF)	Reserved	–	–	–	–	–	–	–	–	
(0xBE)	Reserved	–	–	–	–	–	–	–	–	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	–	185
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	182
(0xBB)	TWDR	2-wire serial interface data register								184
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	184
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	183

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. 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.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, 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.
4. 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 ATmega48/88/168 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.
5. Only valid for Atmel® ATmega88/168

30. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CPI	Rd, K	Compare register with immediate	Rd – K	Z,N,V,C,H	1
SBRC	Rr, b	Skip if bit in register cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if bit in register is set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if bit in I/O register cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if bit in I/O register is set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if status flag set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1/2
BRBC	s, k	Branch if status flag cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1/2
BREQ	k	Branch if equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if greater or equal, signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if less than zero, signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag is cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
Bit and Bit-test Instructions					
SBI	P, b	Set bit in I/O register	I/O(P,b) ← 1	None	2
CBI	P, b	Clear Bit in I/O register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical shift left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical shift right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate left through carry	Rd(0) ← C,Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate right through carry	Rd(7) ← C,Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic shift right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap nibbles	Rd(3..0) ← Rd(7..4), Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag clear	SREG(s) ← 0	SREG(s)	1

Note: 1. These instructions are only available in Atmel® ATmega168.

8. Minimizing Power Consumption . . . 36
8.1 System Control and Reset . . . 38
8.2 Resetting the AVR . . . 38
8.3 Reset Sources . . . 38
8.4 Power-on Reset . . . 39
8.5 External Reset . . . 41
8.6 Brown-out Detection . . . 41
8.7 Watchdog System Reset . . . 42
8.8 MCU Status Register – MCUSR . . . 43
8.9 Internal Voltage Reference . . . 43
8.10 Watchdog Timer . . . 44

9. Interrupts . . . 48
9.1 Interrupt Vectors in ATmega48 . . . 48
9.2 Interrupt Vectors in ATmega88 . . . 49
9.3 Interrupt Vectors in ATmega168 . . . 52

10. I/O-Ports . . . 57
10.1 Introduction . . . 57
10.2 Ports as General Digital I/O . . . 58
10.3 Alternate Port Functions . . . 62
10.4 Register Description for I/O Ports . . . 71

11. External Interrupts . . . 73
11.1 External Interrupt Control Register A – EICRA . . . 73
11.2 External Interrupt Mask Register – EIMSK . . . 74
11.3 External Interrupt Flag Register – EIFR . . . 74
11.4 Pin Change Interrupt Control Register - PCICR . . . 75
11.5 Pin Change Interrupt Flag Register - PCIFR . . . 75
11.6 Pin Change Mask Register 2 – PCMSK2 . . . 76
11.7 Pin Change Mask Register 1 – PCMSK1 . . . 76
11.8 Pin Change Mask Register 0 – PCMSK0 . . . 76

12. 8-bit Timer/Counter0 with PWM . . . 77
12.1 Overview . . . 77
12.2 Timer/Counter Clock Sources . . . 78
12.3 Counter Unit . . . 78
12.4 Output Compare Unit . . . 79
12.5 Compare Match Output Unit . . . 80
12.6 Modes of Operation . . . 81
12.7 Timer/Counter Timing Diagrams . . . 85
12.8 8-bit Timer/Counter Register Description . . . 87

13. Timer/Counter0 and Timer/Counter1 Prescalers . . . 92
13.1 Internal Clock Source . . . 92
13.2 Prescaler Reset . . . 92
13.3 External Clock Source . . . 92

14. 16-bit Timer/Counter1 with PWM . . . 94
14.1 Overview . . . 94
14.2 Accessing 16-bit Registers . . . 96
14.3 Timer/Counter Clock Sources . . . 99
14.4 Counter Unit . . . 100
14.5 Input Capture Unit . . . 101
14.6 Output Compare Units . . . 102
14.7 Compare Match Output Unit . . . 104