

Welcome to [E-XFL.COM](https://www.e-xfl.com)

### What is "[Embedded - Microcontrollers](#)"?

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

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	20MHz
Connectivity	I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	16KB (8K 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-VFQFN Exposed Pad
Supplier Device Package	32-VQFN (5x5)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atmega168-20mq">https://www.e-xfl.com/product-detail/microchip-technology/atmega168-20mq</a>

When the write access time has elapsed, the EEPB bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPB has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEPB bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. [Table 5-2](#) lists the typical programming time for EEPROM access from the CPU.

**Table 5-2.** EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ Programming Time
EEPROM write (from CPU)	26,368	3.3 ms

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

## 7.7 Minimizing Power Consumption

There are several possibilities to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

### 7.7.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to ["Analog-to-Digital Converter" on page 239](#) for details on ADC operation.

### 7.7.2 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to ["Analog Comparator" on page 235](#) for details on how to configure the Analog Comparator.

### 7.7.3 Brown-out Detector

If the Brown-out Detector is not needed by the application, this module should be turned off. If the Brown-out Detector is enabled by the BODLEVEL Fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to ["Brown-out Detection" on page 46](#) for details on how to configure the Brown-out Detector.

### 7.7.4 Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detection, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to ["Internal Voltage Reference" on page 48](#) for details on the start-up time.

### 7.7.5 Watchdog Timer

If the Watchdog Timer is not needed in the application, the module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes and hence always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to ["Watchdog Timer" on page 49](#) for details on how to configure the Watchdog Timer.

### 7.7.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ( $\text{clk}_{\text{I/O}}$ ) and the ADC clock ( $\text{clk}_{\text{ADC}}$ ) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In

## 8.0.5 Brown-out Detection

ATmega48/88/168 has an On-chip Brown-out Detection (BOD) circuit for monitoring the  $V_{CC}$  level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as  $V_{BOT+} = V_{BOT} + V_{HYST}/2$  and  $V_{BOT-} = V_{BOT} - V_{HYST}/2$ .

**Table 8-2.** BODLEVEL Fuse Coding<sup>(1)</sup>

BODLEVEL 2..0 Fuses	Min V <sub>BOT</sub>	Typ V <sub>BOT</sub>	Max V <sub>BOT</sub>	Units
111	BOD Disabled			
110	1.7 <sup>(2)</sup>	1.8	2.0 <sup>(2)</sup>	V
101	2.5 <sup>(2)</sup>	2.7	2.9 <sup>(2)</sup>	
100	4.1 <sup>(2)</sup>	4.3	4.5 <sup>(2)</sup>	
011	Reserved			
010				
001				
000				

- Notes:
- $V_{BOT}$  may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to  $V_{CC} = V_{BOT}$  during the production test. This guarantees that a Brown-Out Reset will occur before  $V_{CC}$  drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 110 and BODLEVEL = 101 for ATmega48V/88V/168V, and BODLEVEL = 101 and BODLEVEL = 101 for ATmega48/88/168.
  - Min/Max values applicable for ATmega48.

**Table 8-3.** Brown-out Characteristics

Symbol	Parameter	Min	Typ	Max	Units
$V_{HYST}$	Brown-out Detector Hysteresis		50		mV
$t_{BOD}$	Min Pulse Width on Brown-out Reset		2		$\mu s$

When the BOD is enabled, and  $V_{CC}$  decreases to a value below the trigger level ( $V_{BOT-}$  in [Figure 8-5](#)), the Brown-out Reset is immediately activated. When  $V_{CC}$  increases above the trigger level ( $V_{BOT+}$  in [Figure 8-5](#)), the delay counter starts the MCU after the Time-out period  $t_{TOUT}$  has expired.

The BOD circuit will only detect a drop in  $V_{CC}$  if the voltage stays below the trigger level for longer than  $t_{BOD}$  given in [Table 8-1](#).

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

## 8.1 Internal Voltage Reference

ATmega48/88/168 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC.

### 8.1.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in [Table 8-4](#). To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2..0] Fuses).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

**Table 8-4.** Internal Voltage Reference Characteristics

Symbol	Parameter	Condition	Min <sup>(1)</sup>	Typ <sup>(1)</sup>	Max <sup>(1)</sup>	Units
$V_{BG}$	Bandgap reference voltage	$V_{CC}=2.7$ $T_A=25^{\circ}\text{C}$	1.0	1.1	1.2	V
$t_{BG}$	Bandgap reference start-up time	$V_{CC}=2.7$ $T_A=25^{\circ}\text{C}$		40	70	$\mu\text{s}$
$I_{BG}$	Bandgap reference current consumption	$V_{CC}=2.7$ $T_A=25^{\circ}\text{C}$		10		$\mu\text{A}$

Note: 1. Values are guidelines only. Actual values are TBD.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

### 10.2.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

### 10.2.3 Switching Between Input and Output

When switching between tri-state ( $\{DDRxn, PORTxn\} = 0b00$ ) and output high ( $\{DDRxn, PORTxn\} = 0b11$ ), an intermediate state with either pull-up enabled ( $\{DDRxn, PORTxn\} = 0b01$ ) or output low ( $\{DDRxn, PORTxn\} = 0b10$ ) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ( $\{DDRxn, PORTxn\} = 0b00$ ) or the output high state ( $\{DDRxn, PORTxn\} = 0b11$ ) as an intermediate step.

Table 10-1 summarizes the control signals for the pin value.

**Table 10-1.** Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

### 10.2.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDRxn, the port pin can be read through the PINxn Register bit. As shown in Figure 10-2, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 10-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted  $t_{pd,max}$  and  $t_{pd,min}$  respectively.

When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

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

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

- **OC1A/PCINT1 – Port B, Bit 1**

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

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

- **ICP1/CLKO/PCINT0 – Port B, Bit 0**

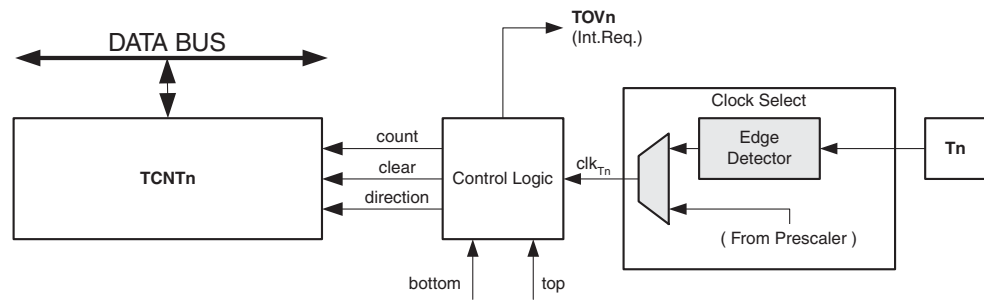
ICP1, Input Capture Pin: The PB0 pin can act as an Input Capture Pin for Timer/Counter1.

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

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

[Table 10-4](#) and [Table 10-5](#) relate the alternate functions of Port B to the overriding signals shown in [Figure 10-5 on page 69](#). SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

**Figure 12-2.** Counter Unit Block Diagram



Signal description (internal signals):

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

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

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see ["Modes of Operation" on page 93](#).

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

## 12.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM02:0 bits and Compare Output mode (COM0x1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (["Modes of Operation" on page 93](#)).

[Figure 12-3](#) shows a block diagram of the Output Compare unit.



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

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

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to [Table 12-8, "Waveform Generation Mode Bit Description"](#) on page 101.

**Table 15-9.** Clock Select Bit Description

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

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

### 15.8.3 Timer/Counter Register – TCNT2

Bit	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT2 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a Compare Match between TCNT2 and the OCR2x Registers.

### 15.8.4 Output Compare Register A – OCR2A

Bit	7	6	5	4	3	2	1	0	
	OCR2A[7:0]								OCR2A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2A pin.

### 15.8.5 Output Compare Register B – OCR2B

Bit	7	6	5	4	3	2	1	0	
	OCR2B[7:0]								OCR2B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2B pin.

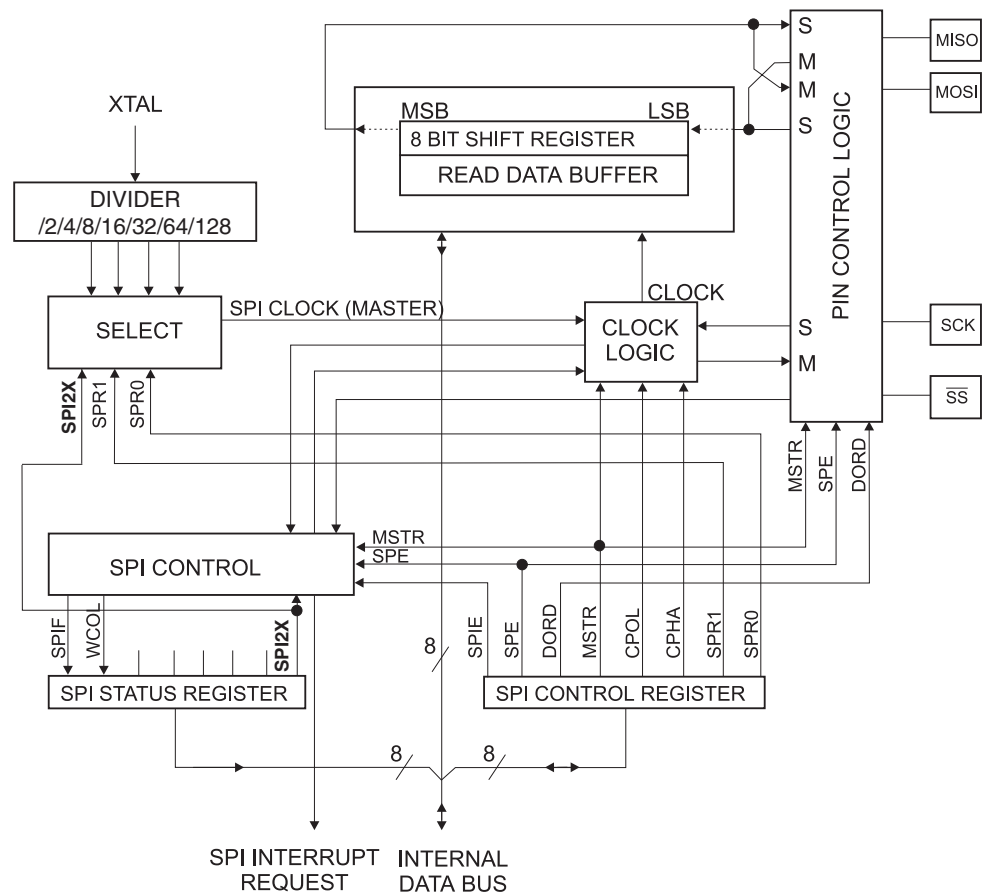
## 16. Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega48/88/168 and peripheral devices or between several AVR devices. The ATmega48/88/168 SPI includes the following features:

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

The USART can also be used in Master SPI mode, see “USART in SPI Mode” on page 196. The PRSPI bit in “Power Reduction Register - PRR” on page 40 must be written to zero to enable SPI module.

Figure 16-1. SPI Block Diagram<sup>(1)</sup>



Note: 1. Refer to Figure 1-1 on page 2, and Table 10-3 on page 71 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 16-2. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the

### 16.1.4 SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	<b>SPIF</b>	<b>WCOL</b>	–	–	–	–	–	<b>SPI2X</b>	<b>SPSR</b>
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

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

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

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

- **Bit 5..1 – Res: Reserved Bits**

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

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

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see [Table 16-4](#)). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at  $f_{osc}/4$  or lower.

The SPI interface on the ATmega48/88/168 is also used for program memory and EEPROM downloading or uploading. See [page 295](#) for serial programming and verification.

### 16.1.5 SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	<b>MSB</b>							<b>LSB</b>	<b>SPDR</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

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

## 16.2 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in [Figure 16-3](#) and [Figure 16-4](#). Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing [Table 16-2](#) and [Table 16-3](#), as done below.

## 17.5.2 Sending Frames with 9 Data Bit

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

### Assembly Code Example<sup>(1)(2)</sup>

```
USART_Transmit:
    ; Wait for empty transmit buffer
    sbis UCSRnA, UDREn
    rjmp USART_Transmit
    ; Copy 9th bit from r17 to TXB8
    cbi UCSRnB, TXB8
    sbrc r17, 0
    sbi UCSRnB, TXB8
    ; Put LSB data (r16) into buffer, sends the data
    out UDRn, r16
    ret
```

### C Code Example<sup>(1)(2)</sup>

```
void USART_Transmit( unsigned int data )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSRnA & (1<<UDREn)) )
        ;
    /* Copy 9th bit to TXB8 */
    UCSRnB &= ~(1<<TXB8);
    if ( data & 0x0100 )
        UCSRnB |= (1<<TXB8);
    /* Put data into buffer, sends the data */
    UDRn = data;
}
```

- Notes:
1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB Register is used after initialization.
  2. [See "About Code Examples" on page 6.](#)  
For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBRC", "SBR", and "CBR".

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

## 17.5.3 Transmitter Flags and Interrupts

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDREn) and Transmit Complete (TXCn). Both flags can be used for generating interrupts.

The Data Register Empty (UDREn) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer

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<sup>(1)</sup>

```
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<sup>(1)</sup>

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

Note: 1. See ["About Code Examples" on page 6](#).

For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBR", "SBR", and "CBR".

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

### 17.6.2 Receiving Frames with 9 Data Bits

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

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

### Assembly Code Example<sup>(1)</sup>

```

USART_Receive:
    ; Wait for data to be received
    sbis UCSRnA, RXCn
    rjmp USART_Receive
    ; Get status and 9th bit, then data from buffer
    in    r18, UCSRnA
    in    r17, UCSRnB
    in    r16, UDRn
    ; If error, return -1
    andi r18, (1<<FEn) | (1<<DORn) | (1<<UPEn)
    breq USART_ReceiveNoError
    ldi   r17, HIGH(-1)
    ldi   r16, LOW(-1)
USART_ReceiveNoError:
    ; Filter the 9th bit, then return
    lsr   r17
    andi r17, 0x01
    ret

```

### C Code Example<sup>(1)</sup>

```

unsigned int USART_Receive( void )
{
    unsigned char status, resh, resl;
    /* Wait for data to be received */
    while ( !(UCSRnA & (1<<RXCn)) )
        ;
    /* Get status and 9th bit, then data */
    /* from buffer */
    status = UCSRnA;
    resh = UCSRnB;
    resl = UDRn;
    /* If error, return -1 */
    if ( status & (1<<FEn) | (1<<DORn) | (1<<UPEn) )
        return -1;
    /* Filter the 9th bit, then return */
    resh = (resh >> 1) & 0x01;
    return ((resh << 8) | resl);
}

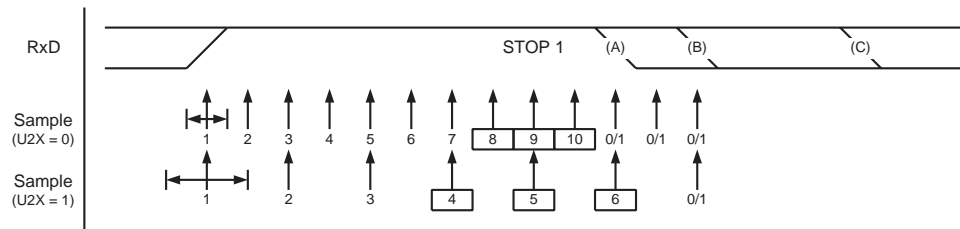
```

Note: 1. See "About Code Examples" on page 6.

For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBRC", "SBR", and "CBR".

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

**Figure 17-7.** Stop Bit Sampling and Next Start Bit Sampling



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

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be at point marked (A) in [Figure 17-7](#). For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

### 17.7.3 Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar (see [Table 17-2](#)) base frequency, the Receiver will not be able to synchronize the frames to the start bit.

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

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

<b>D</b>	Sum of character size and parity size (D = 5 to 10 bit)
<b>S</b>	Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
<b>S<sub>F</sub></b>	First sample number used for majority voting. S <sub>F</sub> = 8 for normal speed and S <sub>F</sub> = 4 for Double Speed mode.
<b>S<sub>M</sub></b>	Middle sample number used for majority voting. S <sub>M</sub> = 9 for normal speed and S <sub>M</sub> = 5 for Double Speed mode.
<b>R<sub>slow</sub></b>	is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R <sub>fast</sub> is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

[Table 17-2](#) and [Table 17-3](#) list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

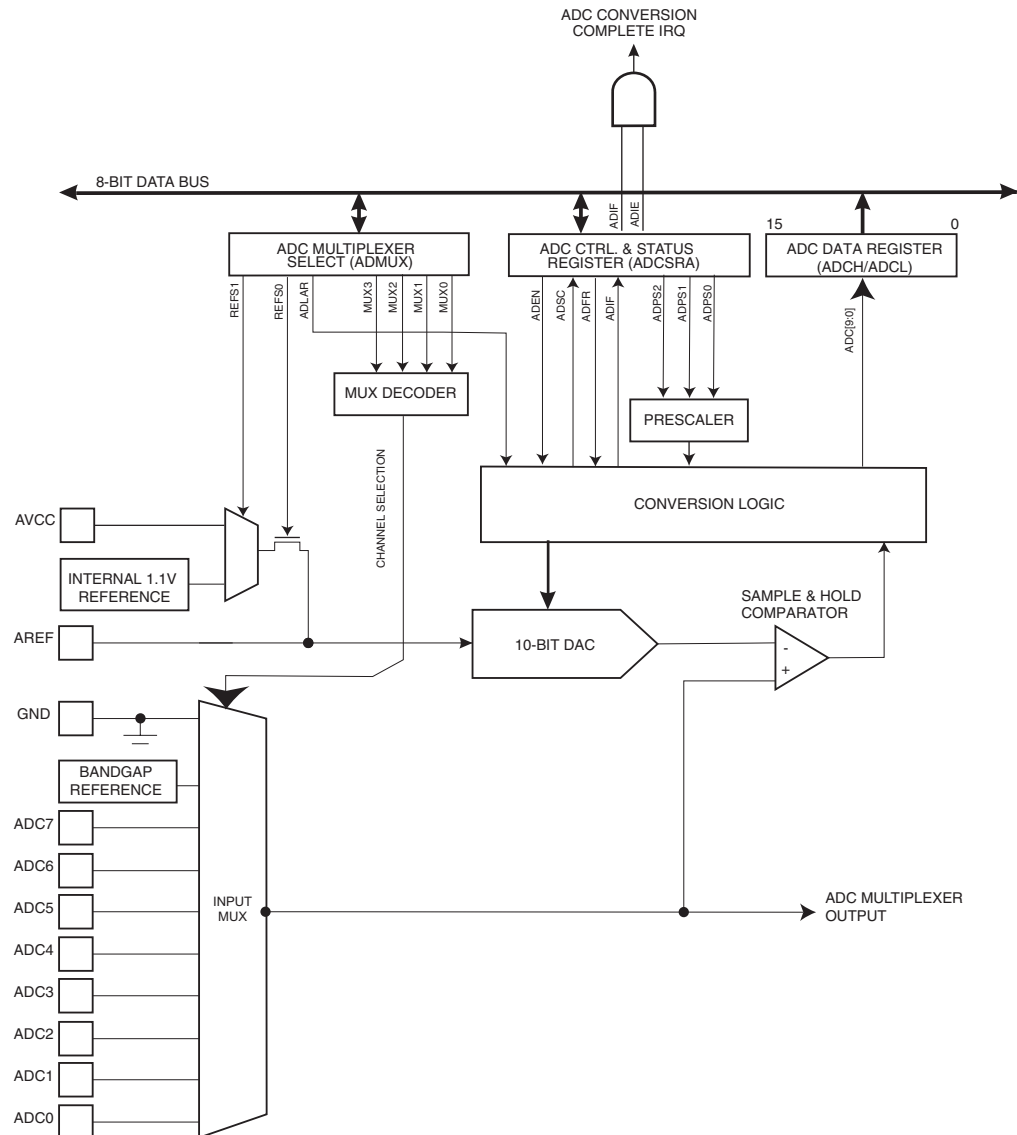


**Table 17-11.** Examples of UBRRn Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	$f_{osc} = 8.0000 \text{ MHz}$				$f_{osc} = 11.0592 \text{ MHz}$				$f_{osc} = 14.7456 \text{ MHz}$			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR n	Error	UBRR n	Error	UBRR n	Error	UBRR n	Error	UBRR n	Error	UBRR n	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	—	—	2	-7.8%	1	-7.8%	3	-7.8%
1M	—	—	0	0.0%	—	—	—	—	0	-7.8%	1	-7.8%
Max. <sup>(1)</sup>	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRRn = 0, Error = 0.0%

**Figure 21-1.** Analog to Digital Converter Block Schematic Operation



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

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is

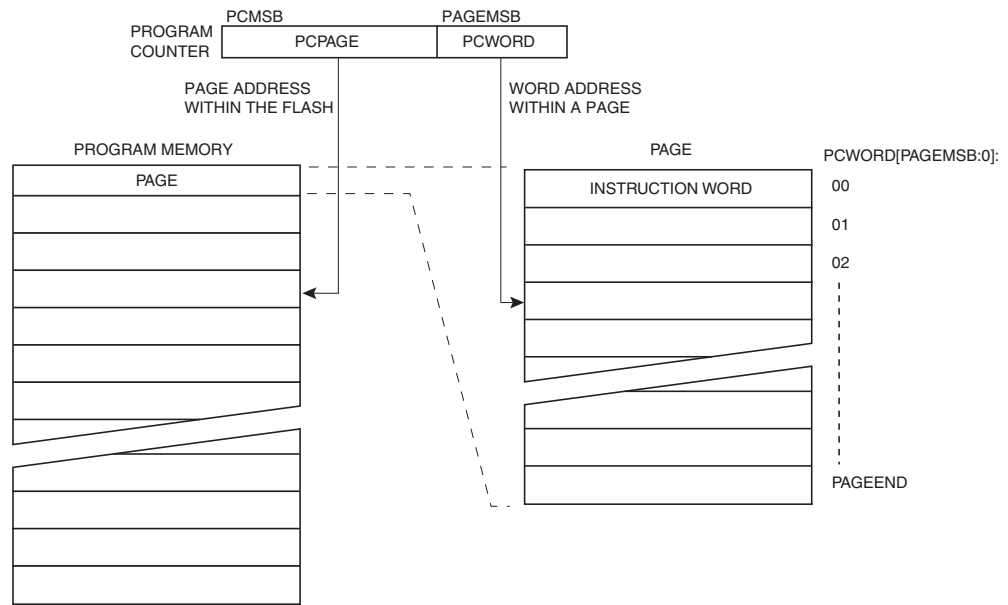
```

    ; return to RWW section
    ; verify that RWW section is safe to read
Return:
    in    temp1, SPMCSR
    sbrs temp1, RWWSB      ; If RWWSB is set, the RWW section is not ready yet
    ret
    ; re-enable the RWW section
    ldi   spmcrval, (1<<RWWSRE) | (1<<SELFPRGEN)
    rcall Do_spm
    rjmp  Return

Do_spm:
    ; check for previous SPM complete
Wait_spm:
    in    temp1, SPMCSR
    sbrc temp1, SELFPRGEN
    rjmp Wait_spm
    ; input: spmcrval determines SPM action
    ; disable interrupts if enabled, store status
    in    temp2, SREG
    cli
    ; check that no EEPROM write access is present
Wait_ee:
    sbic EECR, EEPE
    rjmp Wait_ee
    ; SPM timed sequence
    out   SPMCSR, spmcrval
    spm
    ; restore SREG (to enable interrupts if originally enabled)
    out   SREG, temp2
    ret

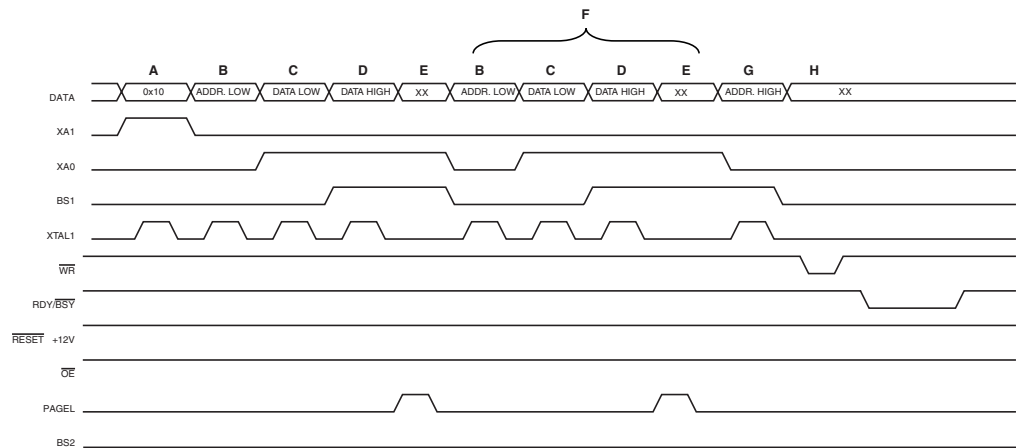
```

**Figure 25-2.** Addressing the Flash Which is Organized in Pages<sup>(1)</sup>



Note: 1. PCPAGE and PCWORD are listed in [Table 25-8 on page 284](#).

**Figure 25-3.** Programming the Flash Waveforms<sup>(1)</sup>



Note: 1. "XX" is don't care. The letters refer to the programming description above.

## 25.7.5 Programming the EEPROM

The EEPROM is organized in pages, see [Table 25-9 on page 284](#). When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to ["Programming the Flash" on page 287](#) for details on Command, Address and Data loading):

1. A: Load Command "0001 0001".
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. C: Load Data (0x00 - 0xFF).

22.6	debugWIRE Related Register in I/O Memory .....	256
<b>23</b>	<b><i>Self-Programming the Flash, ATmega48 .....</i></b>	<b>256</b>
23.1	Addressing the Flash During Self-Programming .....	258
<b>24</b>	<b><i>Boot Loader Support – Read-While-Write Self-Programming, ATmega88 and ATmega168 .....</i></b>	<b>264</b>
24.1	Boot Loader Features .....	264
24.2	Application and Boot Loader Flash Sections .....	264
24.3	Read-While-Write and No Read-While-Write Flash Sections .....	265
24.4	Boot Loader Lock Bits .....	267
24.5	Entering the Boot Loader Program .....	268
24.6	Addressing the Flash During Self-Programming .....	270
24.7	Self-Programming the Flash .....	271
<b>25</b>	<b><i>Memory Programming .....</i></b>	<b>280</b>
25.1	Program And Data Memory Lock Bits .....	280
25.2	Fuse Bits .....	281
25.3	Signature Bytes .....	283
25.4	Calibration Byte .....	284
25.5	Page Size .....	284
25.6	Parallel Programming Parameters, Pin Mapping, and Commands .....	284
25.7	Parallel Programming .....	286
25.8	Serial Downloading .....	295
25.9	Serial Programming Pin Mapping .....	296
<b>26</b>	<b><i>Electrical Characteristics .....</i></b>	<b>299</b>
26.1	Absolute Maximum Ratings* .....	299
26.2	DC Characteristics .....	299
26.3	External Clock Drive Waveforms .....	301
26.4	External Clock Drive .....	301
26.5	Maximum Speed vs. $V_{CC}$ .....	301
26.6	2-wire Serial Interface Characteristics .....	302
26.7	SPI Timing Characteristics .....	304
26.8	ADC Characteristics – Preliminary Data .....	306
<b>27</b>	<b><i>ATmega48/88/168 Typical Characteristics – Preliminary Data .....</i></b>	<b>307</b>
27.1	Active Supply Current .....	307
27.2	Idle Supply Current .....	310
27.3	Supply Current of I/O modules .....	313