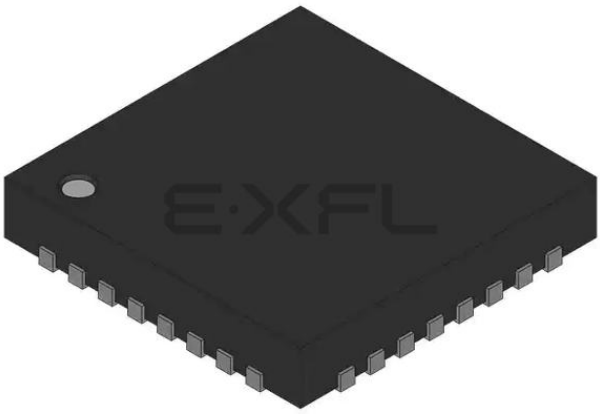


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

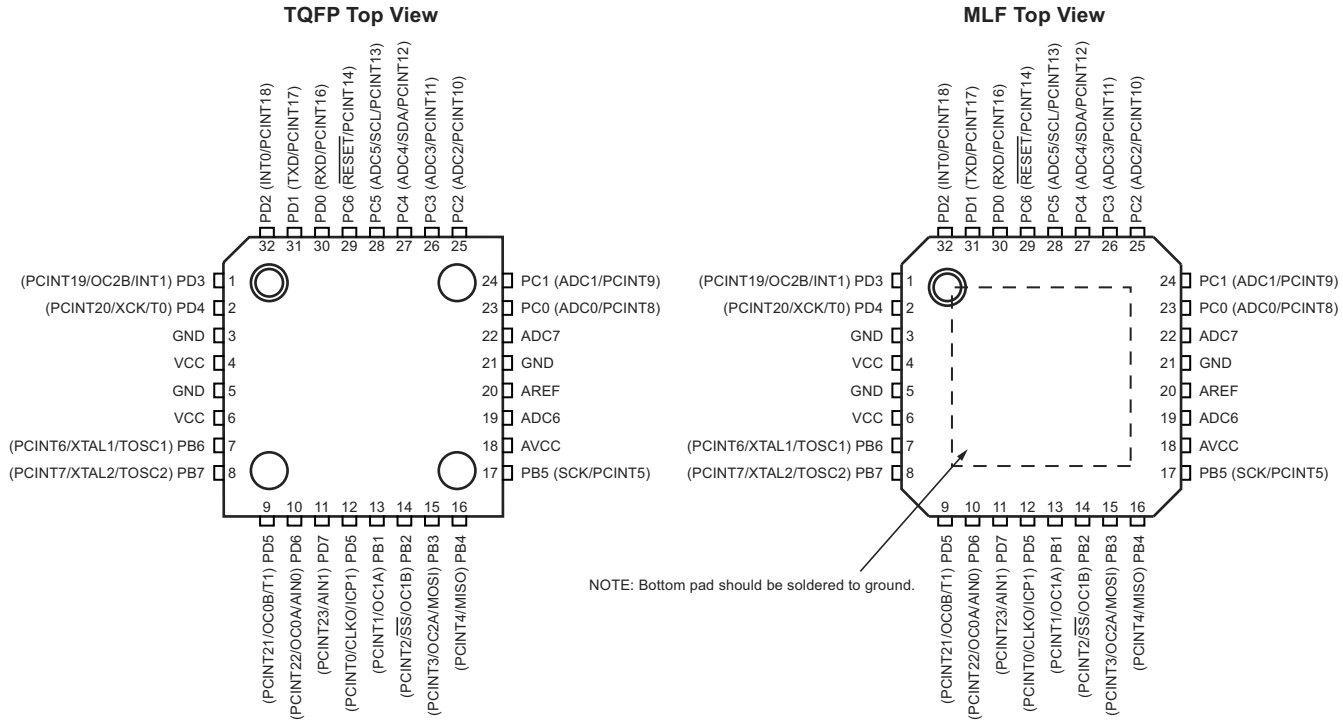
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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 ~ 85°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/atmel/atmega88-15mt

1. Pin Configurations

Figure 1-1. Pinout ATmega48/88/168



1.1 Disclaimer

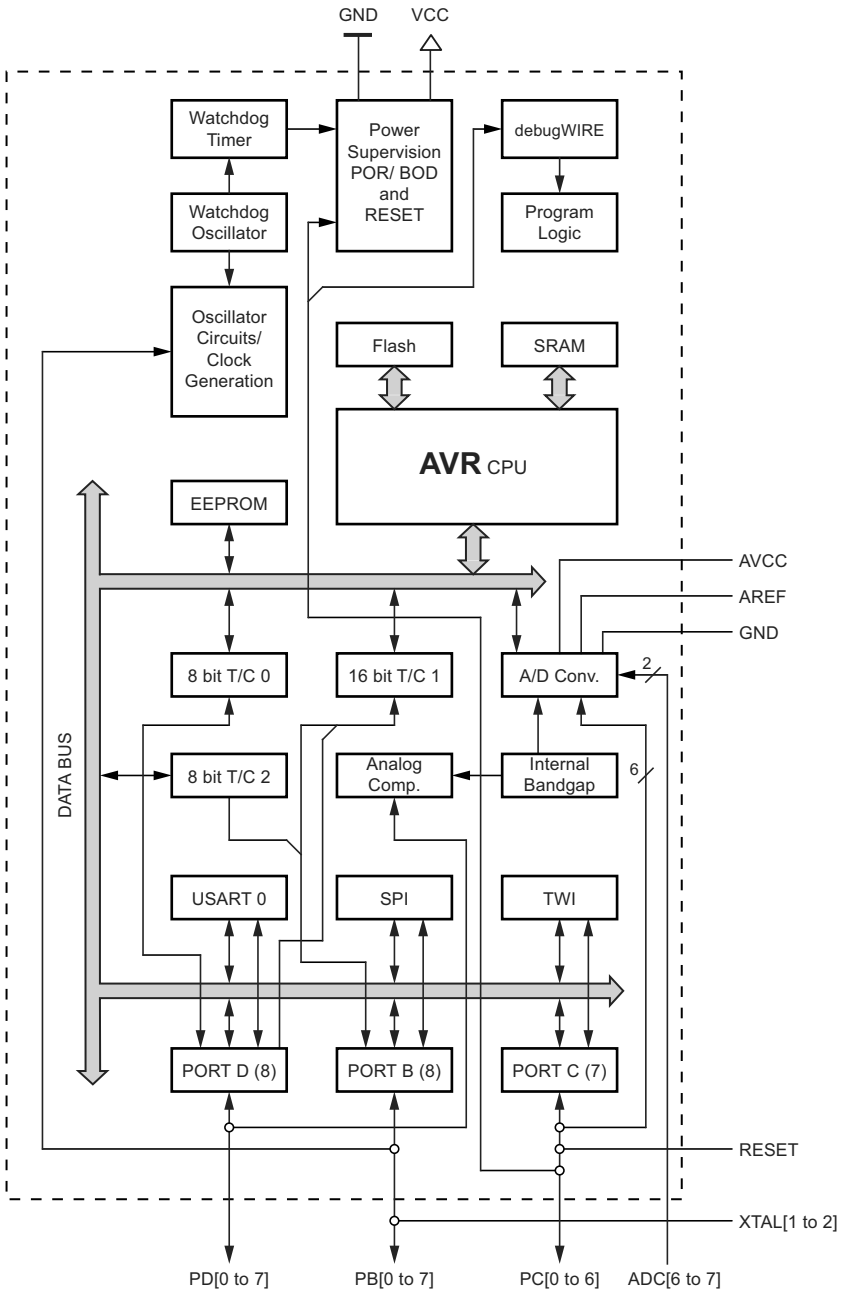
Typical values contained in this datasheet are based on simulations and characterization of other AVR® microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

2. Overview

The Atmel® ATmega48/88/168 is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the Atmel ATmega48/88/168 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



8.7 MCU Status Register – MCUSR

The MCU status register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7..4: Res: Reserved Bits**

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

- **Bit 3 – WDRF: Watchdog System Reset Flag**

This bit is set if a watchdog system reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a brown-out reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **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.

8.8 Internal Voltage Reference

Atmel 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.

8.8.1 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(1)

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Bandgap reference voltage	TBD	V_{BG}	1.0	1.1	1.2	V
Bandgap reference start-up time	TBD	t_{BG}		40	70	µs
Bandgap reference current consumption	TBD	I_{BG}		10	TBD	µA

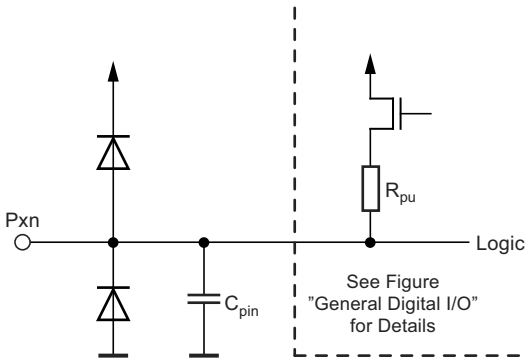
Note: 1. Values are guidelines only. Actual values are TBD.

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 26. “Electrical Characteristics” on page 260 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 71.

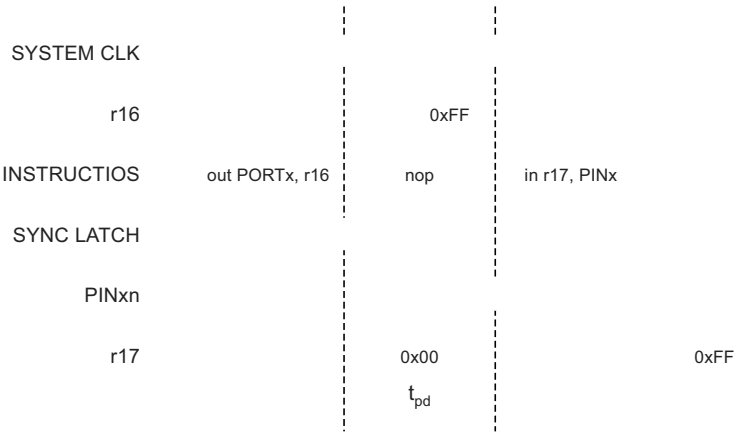
Three I/O memory address locations are allocated for each port, one each for the data register – PORTx, data direction register – DDRx, and the port input pins – PINx. The port input pins I/O location is read only, while the data register and the data direction register are read/write. However, writing 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 58. 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 62. 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.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 10-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 10-4. Synchronization when Reading a Software Assigned Pin Value



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

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi    r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi    r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out    PORTB,r16
out    DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in     r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
__no_operation();
/* Read port pins */
i = PINB;
...
```

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

- **SCK/PCINT5 – Port B, Bit 5**

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

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

- **MISO/PCINT4 – Port B, Bit 4**

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

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

- **MOSI/OC2/PCINT3 – Port B, Bit 3**

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

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

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

- **$\overline{\text{SS}}$/OC1B/PCINT2 – Port B, Bit 2**

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

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

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

- **OC1A/PCINT1 – Port B, Bit 1**

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

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

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

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

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

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

- **ADC2/PCINT10 – Port C, Bit 2**

PC2 can also be used as ADC input channel 2. Note that ADC input channel 2 uses analog power.

PCINT10: pin change interrupt source 10. The PC2 pin can serve as an external interrupt source.

- **ADC1/PCINT9 – Port C, Bit 1**

PC1 can also be used as ADC input channel 1. Note that ADC input channel 1 uses analog power.

PCINT9: pin change interrupt source 9. The PC1 pin can serve as an external interrupt source.

- **ADC0/PCINT8 – Port C, Bit 0**

PC0 can also be used as ADC input channel 0. Note that ADC input channel 0 uses analog power.

PCINT8: pin change interrupt source 8. The PC0 pin can serve as an external interrupt source.

Table 10-7 and Table 10-8 on page 68 relate the alternate functions of port C to the overriding signals shown in Figure 10-5 on page 62.

Table 10-7. Overriding Signals for Alternate Functions in PC6..PC4[1]

Signal Name	PC6/$\overline{\text{RESET}}$/PCINT14	PC5/SCL/ADC5/PCINT13	PC4/SDA/ADC4/PCINT12
PUOE	RSTDISBL	TWEN	TWEN
PUOV	1	PORTC5 × $\overline{\text{PUD}}$	PORTC4 × $\overline{\text{PUD}}$
DDOE	RSTDISBL	TWEN	TWEN
DDOV	0	SCL_OUT	SDA_OUT
PVOE	0	TWEN	TWEN
PVOV	0	0	0
DIEOE	RSTDISBL + PCINT14 × PCIE1	PCINT13 × PCIE1 + ADC5D	PCINT12 × PCIE1 + ADC4D
DIEOV	RSTDISBL	PCINT13 × PCIE1	PCINT12 × PCIE1
DI	PCINT14 INPUT	PCINT13 INPUT	PCINT12 INPUT
AIO	RESET INPUT	ADC5 INPUT / SCL INPUT	ADC4 INPUT / SDA INPUT

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

Table 10-8. Overriding Signals for Alternate Functions in PC3..PC0

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

Bits 5:4 – COM0**B1:0: Compare Match Output B Mode**

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

When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting. Table 12-5 on page 88 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 12-5. Compare Output Mode, non-PWM Mode

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

Table 12-6 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 12-6. Compare Output Mode, Fast PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match, set OC0B at TOP
1	1	Set OC0B on compare match, clear OC0B at TOP

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 12.6.3 “Fast PWM Mode” on page 83 for more details.

Table 12-7 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 12-7. Compare Output Mode, Phase Correct PWM Mode(1)

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

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 12.6.4 “Phase Correct PWM Mode” on page 84 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

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

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

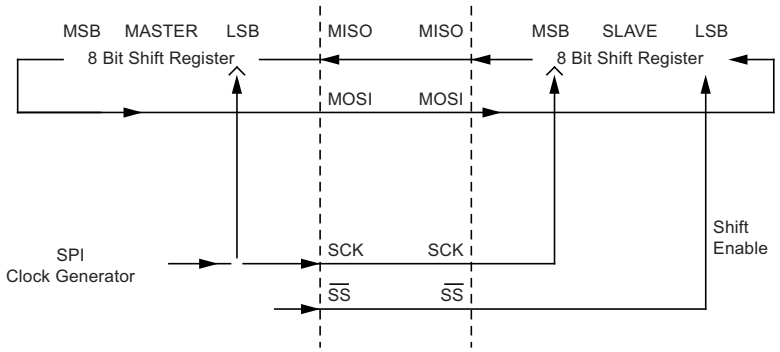
Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 12-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 12.6 “Modes of Operation” on page 81).

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 communication cycle when pulling low the slave select $\overline{SS}$ pin of the desired slave. Master and slave prepare the data to be sent in their respective shift registers, and the master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from master to slave on the master out – slave In, MOSI, line, and from slave to master on the master In – slave out, MISO, line. After each data packet, the master will synchronize the slave by pulling high the slave select, $\overline{SS}$, line.

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

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

Figure 16-2. SPI Master-slave Interconnection



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

In SPI slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed $f_{osc}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 16-1 on page 139. For more details on automatic port overrides, refer to Section 10.3 “Alternate Port Functions” on page 62.

Table 16-1. SPI Pin Overrides(1)

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

Note: 1. See Section 10.3.2 “Alternate Functions of Port B” on page 64 for a detailed description of how to define the direction of the user defined SPI pins.

- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® 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 on page 143). 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 Atmel ATmega48/88/168 is also used for program memory and EEPROM downloading or uploading. See Section 25.8 “Serial Downloading” on page 256 for serial programming and verification.

16.1.5 SPI Data Register – SPDR

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

The SPI data register is a read/write register used for data transfer between the register file and the SPI shift register. Writing to the register initiates data transmission. Reading the register causes the shift register receive buffer to be read.

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 on page 145 and Figure 16-4 on page 145. 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 Figure 16-2 on page 142 and Table 16-3 on page 143, as done below.

Table 16-5. CPOL Functionality

	Leading Edge	Trailing eDge	SPI Mode
CPOL=0, CPHA=0	Sample (rising)	Setup (falling)	0
CPOL=0, CPHA=1	Setup (rising)	Sample (falling)	1
CPOL=1, CPHA=0	Sample (falling)	Setup (rising)	2
CPOL=1, CPHA=1	Setup (falling)	Sample (rising)	3

Figure 16-3. SPI Transfer Format with CPHA = 0

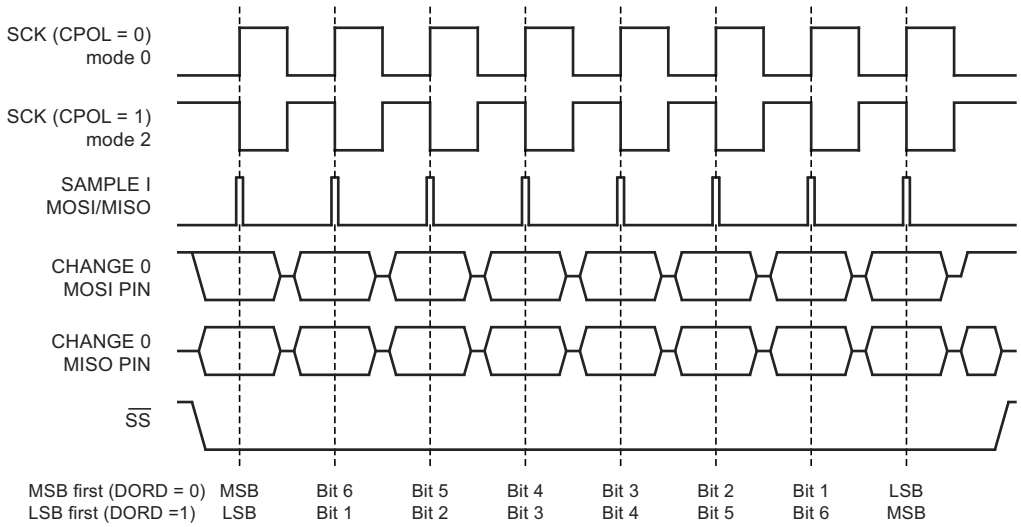
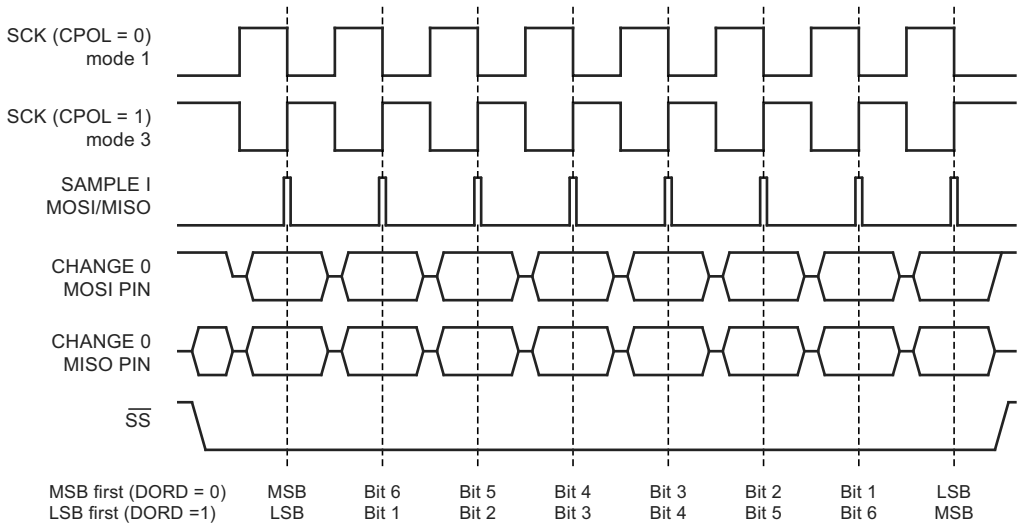


Figure 16-4. SPI Transfer Format with CPHA = 1



Note that arbitration is not allowed between:

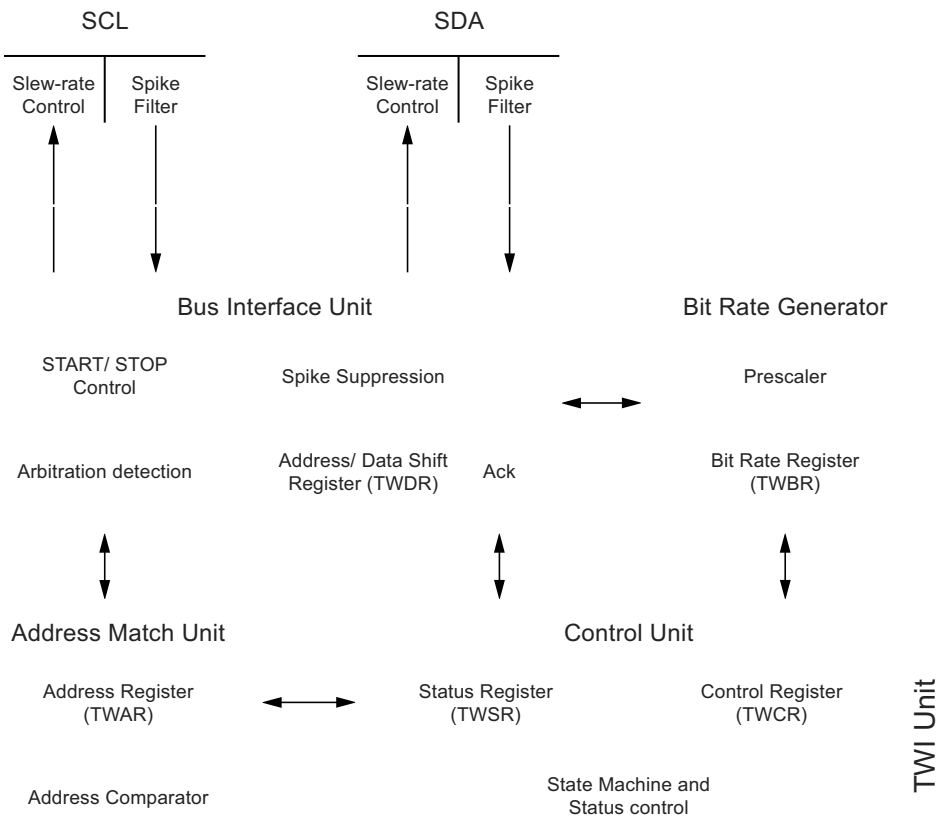
- A REPEATED START condition and a data bit.
- A STOP condition and a data bit.
- A REPEATED START and a STOP condition.

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.

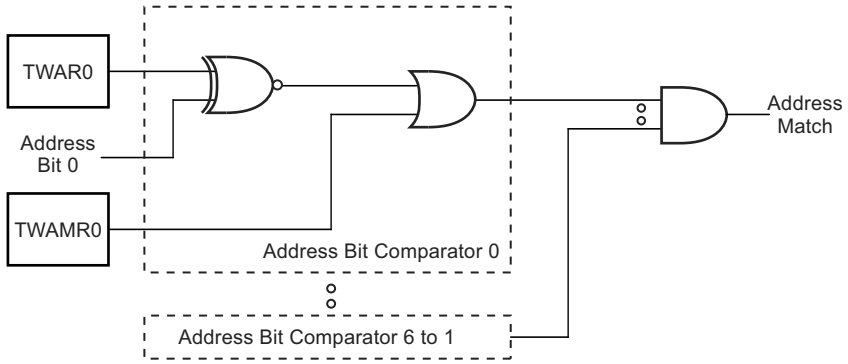
19.6.6 TWI (Slave) Address Mask Register – TWAMR

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

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

The TWAMR can be loaded with a 7-bit salve address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bits in the TWI address register (TWAR). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR. Figure 19-10 shown the address match logic in detail.

Figure 19-10. TWI Address Match Logic, Block Diagram



- **Bit 0 – Res: Reserved Bit**

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

19.7 Using the TWI

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

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

Figure 19-11 on page 186 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

After a repeated START condition (state 0x10) the 2-wire serial interface can access the same slave again, or a new slave without transmitting a STOP condition. Repeated START enables the master to switch between slaves, master transmitter mode and master receiver mode without losing control over the bus.

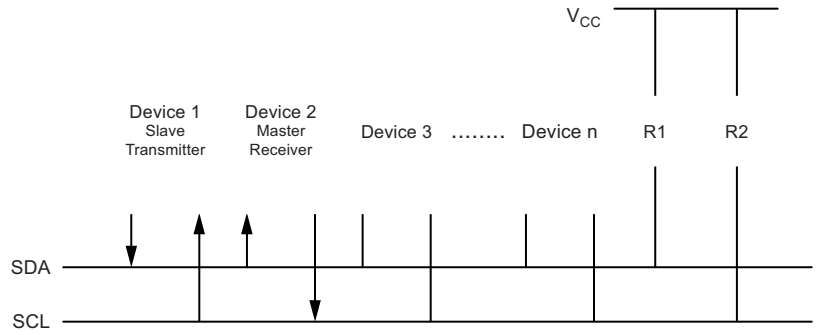
Table 19-5. Status Codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	0 0	0 0	1 1	X X	SLA+R will be transmitted ACK or NOT ACK will be received SLA+W will be transmitted Logic will switch to master transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or No TWDR action	0 0	0 0	1 1	X X	2-wire serial bus will be released and not addressed Slave mode will be entered A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or No TWDR action	0 0	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or No TWDR action or No TWDR action	1 0 1	0 1 1	1 1 1	X X X	Repeated START will be transmitted STOP condition will be transmitted and TWSTO flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or Read data byte	0 0	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or Read data byte or Read data byte	1 0 1	0 1 1	1 1 1	X X X	Repeated START will be transmitted STOP condition will be transmitted and TWSTO flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset

19.8.4 Slave Transmitter Mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 19-18). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 19-18. Data Transfer in Slave Transmitter Mode



To initiate the slave transmitter mode, TWAR and TWCR must be initialized as follows:

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

The upper seven bits are the address to which the 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 "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 19-7 on page 199. The slave transmitter mode may also be entered if arbitration is lost while the TWI is in the master mode (see state 0xB0).

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

While TWEA is zero, the TWI does not respond to its own slave address. However, the 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 will low during the wake up and until the TWINT flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the AVR® clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire serial interface data register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.

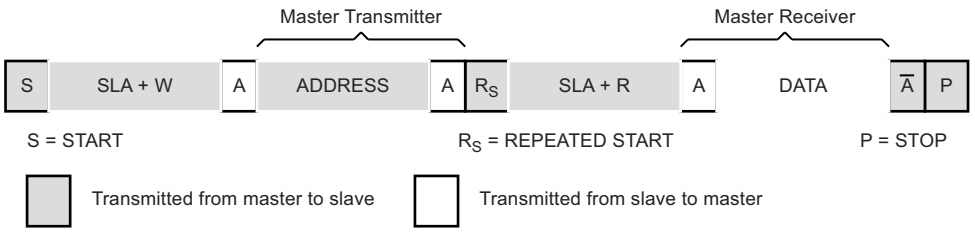
19.8.6 Combining Several TWI Modes

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

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

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

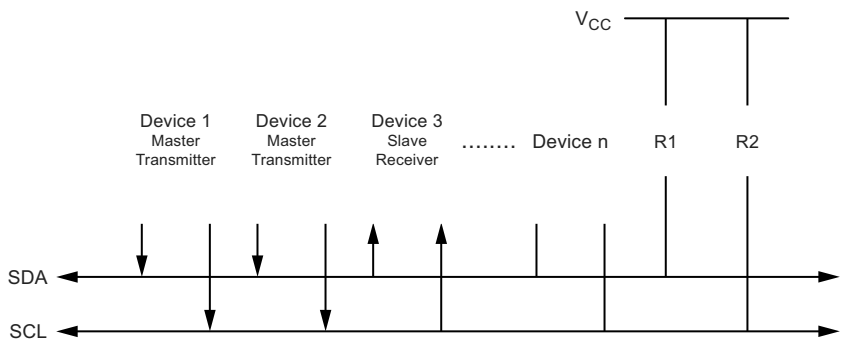
Figure 19-20. Combining Several TWI Modes to Access a Serial EEPROM



19.9 Multi-master Systems and Arbitration

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

Figure 19-21. An Arbitration Example



24.7.14 ATmega168 Boot Loader Parameters

In Table 24-9 through Table 24-11, the parameters used in the description of the self programming are given.

Table 24-9. Boot Size Configuration, ATmega168

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 - 0x1FFF	0x1F7F	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 - 0x1FFF	0x1EFF	0x1F00
0	1	512 words	8	0x0000 - 0x1DFF	0x1E00 - 0x1FFF	0x1DFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00 - 0x1FFF	0x1BFF	0x1C00

Note: The different BOOTSZ fuse configurations are shown in Figure 24-2 on page 231.

Table 24-10. Read-While-Write Limit, ATmega168

Section	Pages	Address
Read-while-write section (RWW)	112	0x0000 - 0x1BFF
No read-while-rite section (NRWW)	16	0x1C00 - 0x1FFF

For details about these two section, see Section 24.3.2 “NRWW – No Read-While-Write Section” on page 230 and Section 24.3.1 “RWW – Read-While-Write Section” on page 230.

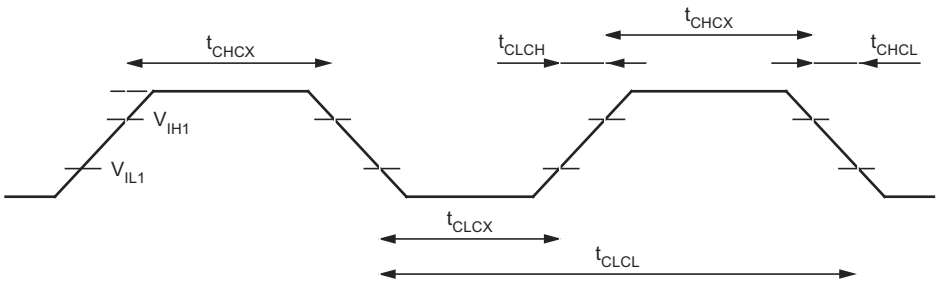
Table 24-11. Explanation of Different Variables used in Figure 24-3 and the Mapping to the Z-pointer, ATmega168

Variable		Corresponding Z-value(1)	Description
PCMSB	12		Most significant bit in the program counter. (The program counter is 12 bits PC[11:0])
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires 6 bits PC [5:0])
ZPCMSB		Z13	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[12:6]	Z13:Z7	Program counter page address: page select, for page erase and page write
PCWORD	PC[5:0]	Z6:Z1	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z14: always ignored
Z0: should be zero for all SPM commands, byte select for the LPM instruction. See Section 24.6 “Addressing the Flash During Self-Programming” on page 234 for details about the use of Z-pointer during self-Programming.

26.3 External Clock Drive Waveforms

Figure 26-1. External Clock Drive Waveforms



26.4 External Clock Drive

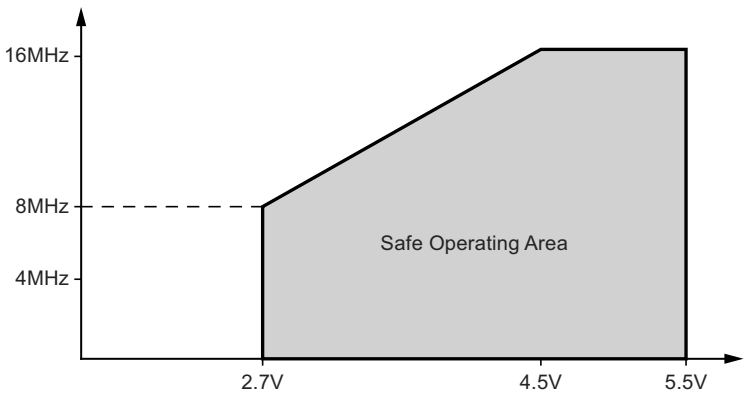
Table 26-1. External Clock Drive

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



28.1.5 Internal Oscillator Speed

Figure 28-19. Watchdog Oscillator Frequency versus V_{CC}

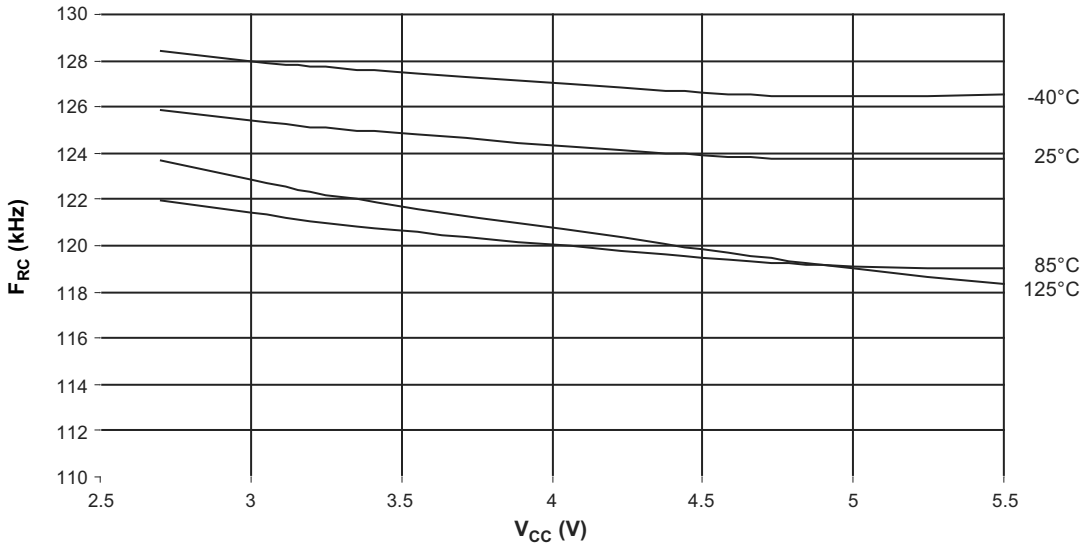
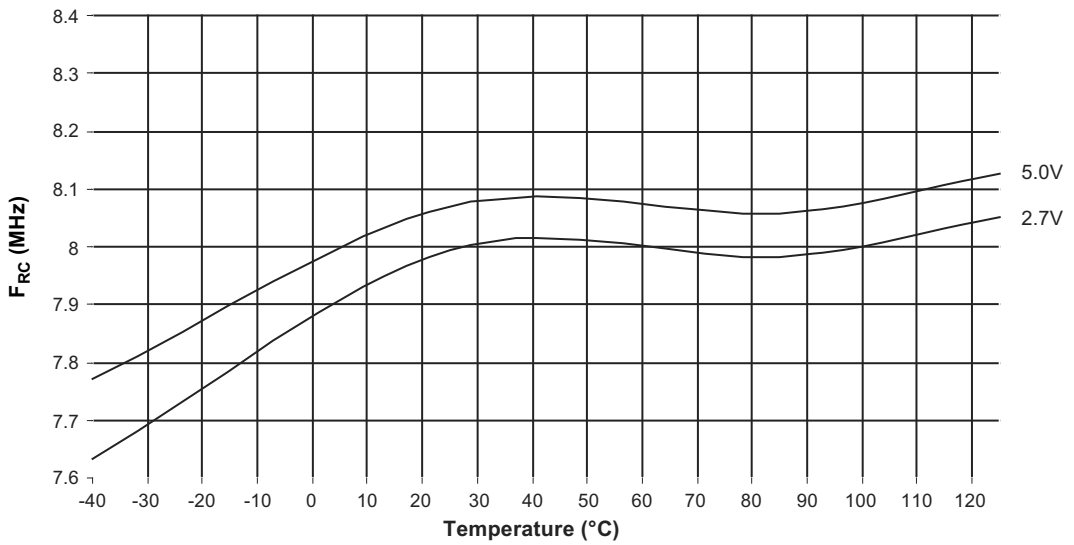
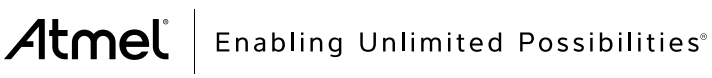


Figure 28-20. Calibrated 8MHz RC Oscillator Frequency versus Temperature





Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**