Microchip Technology - ATMEGA168-15AD Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
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	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 ~ 150°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/atmega168-15ad

- I/O and packages
  - 23 programmable I/O lines
  - Green/ROHS 32-lead TQFP and 32-pad QFN
- Operating voltage:
  - 2.7 - 5.5V
- Temperature range:
  - –40°C to 150°C
- Speed grade:
  - ATmega88/168: 0 to 8MHz at 2.7 to 5.5V, 0 - 16MHz at 4.5 to 5.5V
- Low power consumption
  - Active mode:
    - 4MHz, 3.0V: 1.8mA
  - Power-down mode:
    - 5µA at 3.0V
- AEC-Q100 Grade 0 qualified

- **Bit 5 - PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

- **Bit 4 - Res: Reserved bit**

This bit is reserved in Atmel® ATmega88/168 and will always read as zero.

- **Bit 3 - PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2 - PRSPI: Power Reduction Serial Peripheral Interface**

If using debugWIRE on-chip debug system, this bit should not be written to one. Writing a logic one to this bit shuts down the serial peripheral interface by stopping the clock to the module. When waking up the SPI again, the SPI should be re-initialized to ensure proper operation.

- **Bit 1 - PRUSART0: Power Reduction USART0**

Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re-initialized to ensure proper operation.

- **Bit 0 - PRADC: Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

7.8 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.

7.8.1 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 Section 21. “Analog-to-Digital Converter” on page 204 for details on ADC operation.

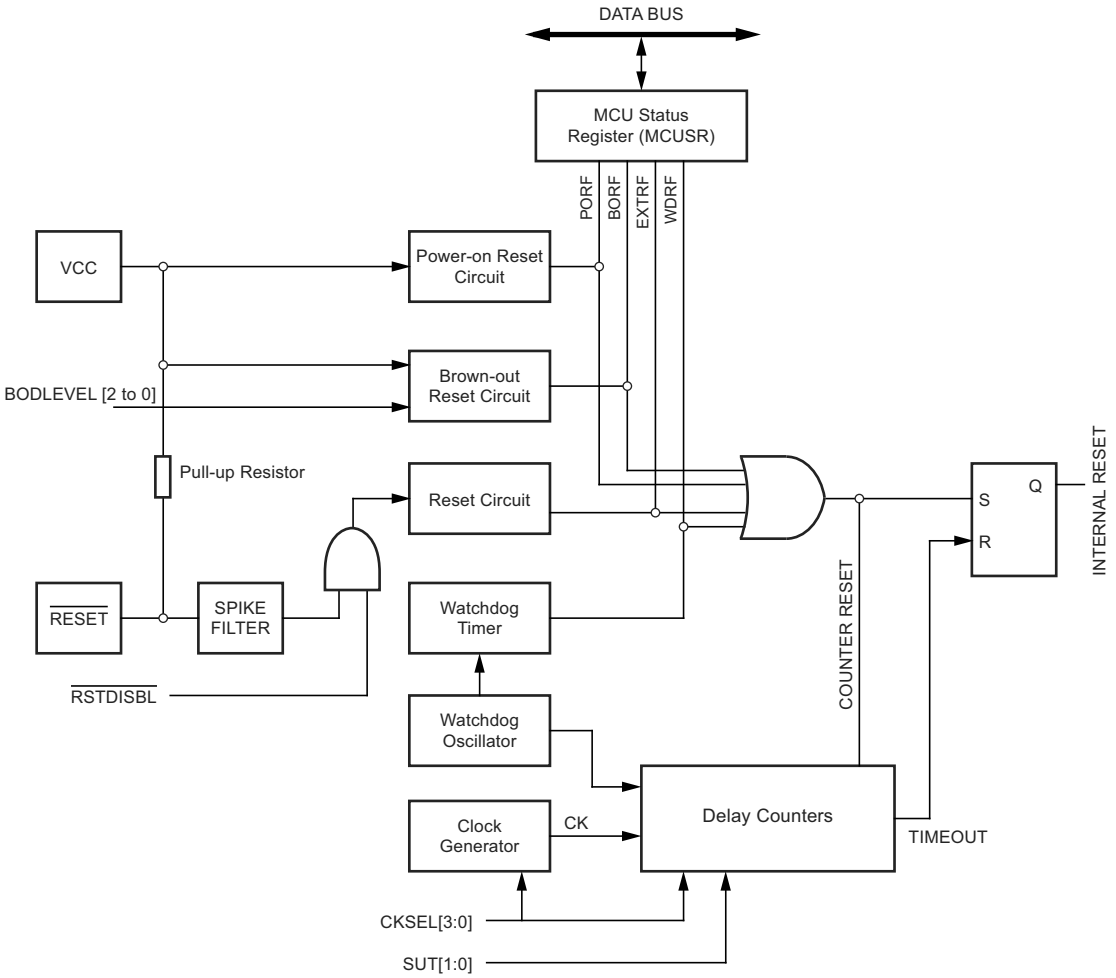
7.8.2 Analog Comparator

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 Section 20. “Analog Comparator” on page 201 for details on how to configure the analog comparator.

7.8.3 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 Section 8.5 “Brown-out Detection” on page 41 for details on how to configure the brown-out detector.

Figure 8-1. Reset Logic



8.3 Power-on Reset

A power-on reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in Table 8-1 on page 40. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the start-up Reset, as well as to detect a failure in supply voltage.

A power-on reset (POR) circuit ensures that the device is reset from power-on. Reaching the power-on reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

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
```

When the BOOTRST fuse is programmed and the boot section size set to 2Kbytes, the most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATmega168 is:

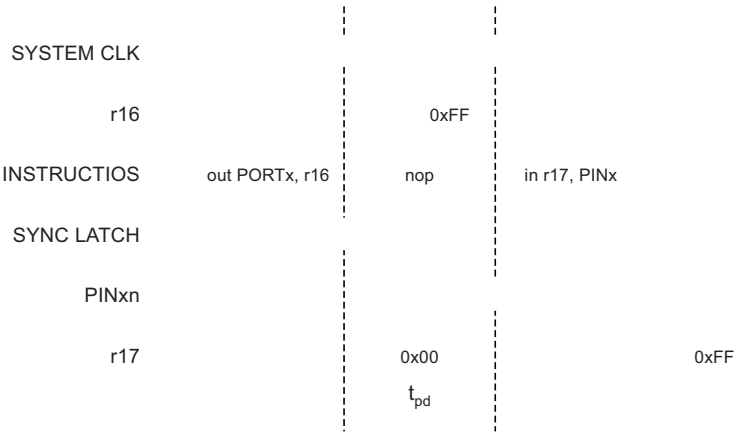
```
Address      Labels Code               Comments
.org 0x0002
0x0002              jmp    EXT_INT0    ; IRQ0 Handler
0x0004              jmp    EXT_INT1    ; IRQ1 Handler
...                 ...    ...         ;
0x0032              jmp    SPM_RDY     ; Store Program Memory Ready Handler
;
.org 0x1C00
0x1C00       RESET: ldi    r16,high(RAMEND); Main program start
0x1C01              out    SPH,r16     ; Set Stack Pointer to top of RAM
0x1C02              ldi    r16,low(RAMEND)
0x1C03              out    SPL,r16
0x1C04              sei                ; Enable interrupts
0x1C05              <instr> xxx
```

When the BOOTRST Fuse is programmed, 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
;
.org 0x1C00
0x1C00              jmp    RESET       ; Reset handler
0x1C02              jmp    EXT_INT0    ; IRQ0 Handler
0x1C04              jmp    EXT_INT1    ; IRQ1 Handler
...          ...    ...                ;
0x1C32              jmp    SPM_RDY     ; Store Program Memory Ready Handler
;
0x1C33       RESET: ldi    r16,high(RAMEND); Main program start
0x1C34              out    SPH,r16     ; Set Stack Pointer to top of RAM
0x1C35              ldi    r16,low(RAMEND)
0x1C36              out    SPL,r16
0x1C37              sei                ; Enable interrupts
0x1C38              <instr> xxx
```

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.

12.1.1 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the output compare unit, in this case compare unit A or compare unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in the following table are also used extensively throughout the document.

Table 12-1. Definitions

Parameters	Definitions
BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A register. The assignment is dependent on the mode of operation.

12.1.2 Registers

The Timer/Counter (TCNT0) and output compare registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to int.req. in the figure) signals are all visible in the timer interrupt flag register (TIFR0). All interrupts are individually masked with the timer interrupt mask register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The clock select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T0}).

The double buffered output compare registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins (OC0A and OC0B). See Section 14.6.3 “Using the Output Compare Unit” on page 102 for details. The compare match event will also set the compare flag (OCF0A or OCF0B) which can be used to generate an output compare interrupt request.

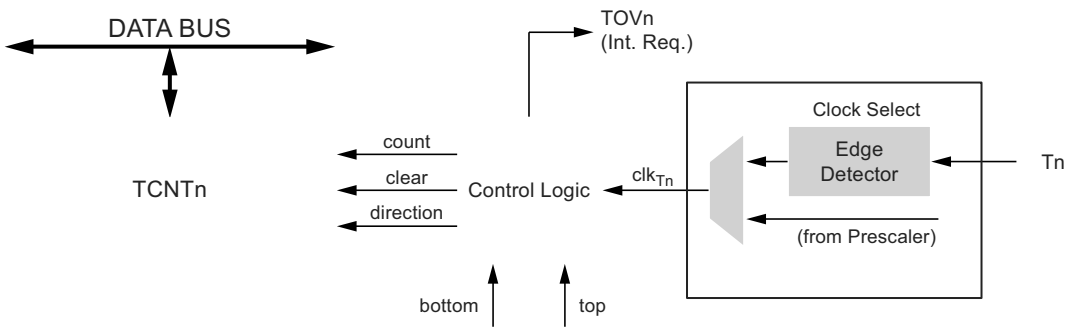
12.2 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter control register (TCCR0B). For details on clock sources and prescaler, see Section 13. “Timer/Counter0 and Timer/Counter1 Prescalers” on page 90.

12.3 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 12-2 shows a block diagram of the counter and its surroundings.

Figure 12-2. Counter Unit Block Diagram



12.8.6 Timer/Counter Interrupt Mask Register – TIMSK0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

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

- **Bit 2 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the status register is set, the Timer/Counter compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter interrupt flag register – TIFR0.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

12.8.7 Timer/Counter 0 Interrupt Flag Register – TIFR0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

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

- **Bit 2 – OCF0B: Timer/Counter 0 Output Compare B Match Flag**

The OCF0B bit is set when a compare match occurs between the Timer/Counter and the data in OCR0B – output compare register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter compare B match interrupt enable), and OCF0B are set, the Timer/Counter compare match interrupt is executed.

- **Bit 1 – OCF0A: Timer/Counter 0 Output Compare A Match Flag**

The OCF0A bit is set when a compare match occurs between the Timer/Counter0 and the data in OCR0A – output compare register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 compare match interrupt enable), and OCF0A are set, the Timer/Counter0 compare match interrupt is executed.

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

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

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to Table 12-8 on page 87 in Section 12-8 “Waveform Generation Mode Bit Description” on page 87.

- **Bit 2 – OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 output compare B match interrupt is enabled. The corresponding interrupt vector (see Section 9. “Interrupts” on page 48) is executed when the OCF1B flag, located in TIFR1, is set.

- **Bit 1 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 output compare A match interrupt is enabled. The corresponding interrupt vector (see Section 9. “Interrupts” on page 48) is executed when the OCF1A flag, located in TIFR1, is set.

- **Bit 0 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 overflow interrupt is enabled. The corresponding interrupt vector (see Section 8.9 “Watchdog Timer” on page 44) is executed when the TOV1 flag, located in TIFR1, is set.

14.10.9 Timer/Counter1 Interrupt Flag Register – TIFR1

Bit	7	6	5	4	3	2	1	0	
	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – Res: Reserved Bits**

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

- **Bit 5 – ICF1: Timer/Counter1, Input Capture Flag**

This flag is set when a capture event occurs on the ICP1 pin. When the input capture register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the input capture interrupt vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

- **Bit 4, 3 – Res: Reserved Bits**

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

- **Bit 2 – OCF1B: Timer/Counter1, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register B (OCR1B).

Note that a forced output compare (FOC1B) strobe will not set the OCF1B flag.

OCF1B is automatically cleared when the output compare match B interrupt vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF1A: Timer/Counter1, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register A (OCR1A).

Note that a forced output compare (FOC1A) strobe will not set the OCF1A flag.

OCF1A is automatically cleared when the output compare match A interrupt vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bits setting. In normal and CTC modes, the TOV1 flag is set when the timer overflows. Refer to Table 14-5 on page 112 for the TOV1 flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

Signal description (internal signals):

count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the clock select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter control register (TCCR2A) and the WGM22 located in the Timer/Counter control register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see Section 15.6 “Modes of Operation” on page 121.

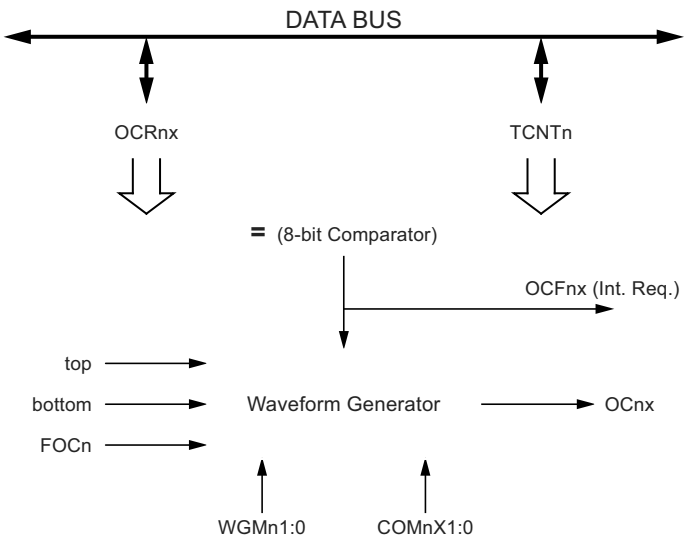
The Timer/Counter overflow flag (TOV2) is set according to the mode of operation selected by the WGM22:0 bits. TOV2 can be used for generating a CPU interrupt.

15.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the output compare register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the output compare flag (OCF2A or OCF2B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the output compare flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM22:0 bits and compare output mode (COM2x1:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (Section 15.6 “Modes of Operation” on page 121).

Figure 15-3 shows a block diagram of the output compare unit.

Figure 15-3. Output Compare Unit, Block Diagram



15.6.1 Normal Mode

The simplest mode of operation is the normal mode (WGM22:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter overflow flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

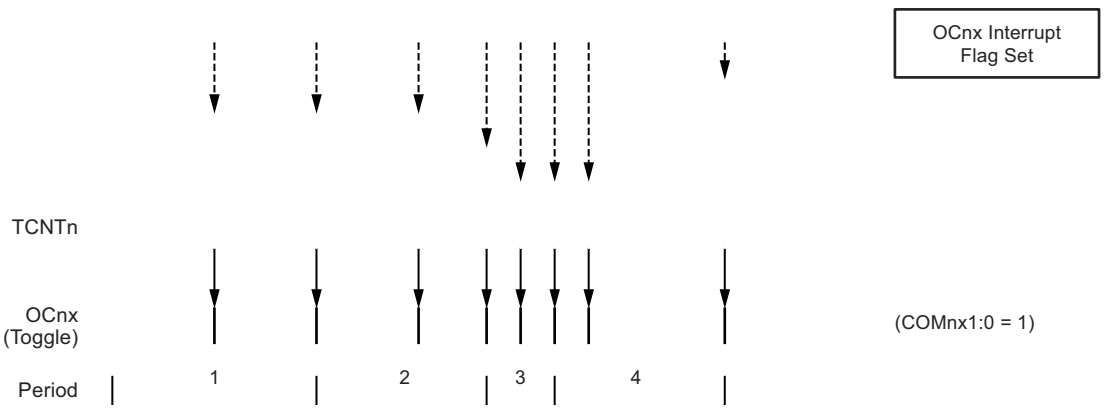
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

15.6.2 Clear Timer on Compare Match (CTC) Mode

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

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

Figure 15-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2A is lower than the current value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC2A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM2A1:0 = 1). The OC2A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC2A} = f_{clk\_I/O}/2$ when OCR2A is set to zero (0x00). The waveform frequency is defined by the following equation:

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

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

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

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

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

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

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

Table 19-3. Code Example

No.	Assembly Code Example	C Example	Comments
1	**ldi** r16, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN) **out** TWCR, r16	TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)	Send START condition
2	wait1: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait1	**while** (!(TWCR & (1<<TWINT)));	Wait for TWINT flag set. This indicates that the START condition has been transmitted
3	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, START **brne** ERROR	**if** ((TWSR & 0xF8) != START ERROR();	Check value of TWI status register. Mask prescaler bits. If status different from START go to ERROR
	ldi r16, SLA_W **out** TWDR, r16 **ldi** r16, (1<<TWINT) | (1<<TWEN) **out** TWCR, r16	TWDR = SLA_W; TWCR = (1<<TWINT) | (1<<TWEN);	Load SLA_W into TWDR register. Clear TWINT bit in TWCR to start transmission of address
4	wait2: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait2	**while** (!(TWCR & (1<<TWINT))) ;	Wait for TWINT flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.
5	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_SLA_ACK **brne** ERROR	**if** ((TWSR & 0xF8) != MT_SLA_ACK) ERROR();	Check value of TWI status register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR
	ldi r16, DATA **out** TWDR, r16 **ldi** r16, (1<<TWINT) | (1<<TWEN) **out** TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) | (1<<TWEN);	Load DATA into TWDR register. Clear TWINT bit in TWCR to start transmission of data

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 194. 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 216). 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 217.

- **Bit 4 – Res: Reserved Bit**

This bit is an unused bit in the Atmel® ATmega88/168, 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 216 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).

22. debugWIRE On-chip Debug System

22.1 Features

- Complete program flow control
- Emulates all on-chip functions, both digital and analog, except RESET pin
- Real-time operation
- Symbolic debugging support (both at C and assembler source Level, or for other HLLs)
- Unlimited number of program break points (using software break points)
- Non-intrusive operation
- Electrical characteristics identical to real device
- Automatic configuration system
- High-speed operation
- Programming of non-volatile memories

22.2 Overview

The debugWIRE on-chip debug system uses a one-wire, bi-directional interface to control the program flow, execute AVR® instructions in the CPU and to program the different non-volatile memories.

22.3 Physical Interface

When the debugWIRE enable (DWEN) fuse is programmed and lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 22-1. The debugWIRE Setup

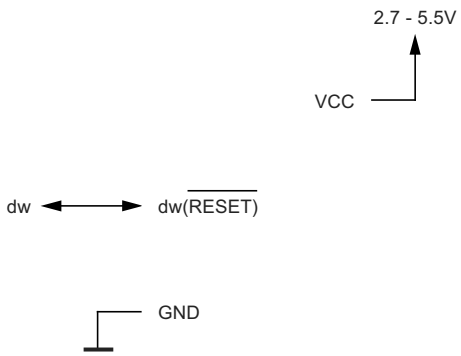


Figure 22-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

Table 24-16. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming after RESET goes low.
Chip erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase EEPROM and flash.
Read program memory	0010 **H**000	000**a aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from program memory at word address **a**:**b**.
Load program memory page	0100 **H**000	000x xxxx	xx**bb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to program memory page at word address **b**. Data low byte must be loaded before data high byte is applied within the same address.
Write program memory page	0100 1100	000**a aaaa**	**bb**xx xxxx	xxxx xxxx	Write program memory page at address **a**:**b**.
Read EEPROM memory	1010 0000	000x xx**aa**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM memory	1100 0000	000x xx**aa**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Load EEPROM memory page (page access)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Write EEPROM memory page (page access)	1100 0010	00xx xx**aa**	**bbbb bb00**	xxxx xxxx	Write EEPROM page at address **a**:**b**.
Read lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read lock bits. “0” = programmed, “1” = unprogrammed. See Table 24-1 on page 234 for details.
Write lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write lock bits. Set bits = “0” to program lock bits. See Table 24-1 on page 234 for details.
Read signature byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read signature byte **o** at address **b**.
Write fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 21-1 on page 209 for details.
Write fuse high bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 21-1 on page 209 for details.
Write extended fuse bits	1010 1100	1010 0100	xxxx xxxx	**xxxx xxii**	Set bits = “0” to program, “1” to unprogram.
Read fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. “0” = programmed, “1” = unprogrammed. See Table 21-1 on page 209 for details.
Read fuse high bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read fuse high bits. “0” = programmed, “1” = unprogrammed. See Table 21-1 on page 209 for details.

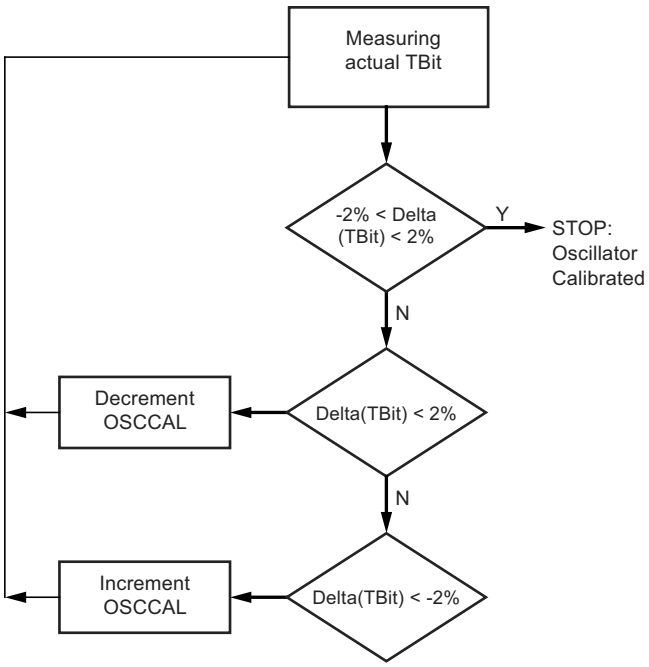
Note: **a** = address high bits, **b** = address low bits, **H** = 0 - low byte, 1 - high byte, **o** = data out, **i** = data in, x = don’t care

25.7 LIN Re-synchronization Algorithm

25.8 Synchronization Algorithm

The possibility to change the value of OSCCAL during the oscillator operation allows for in-situ calibration of the slave node to entering Master frames. The principle of operation is to measure the TBit during the SYNCH byte and to change the calibration value of OSCCAL to recover from local frequency drifts due to local voltage or temperature deviation. The algorithm used for the synchronization of the internal RC oscillator is depicted in Figure 25-3 on page 255.

Figure 25-3. Dichotomic Algorithm Used for LIN Slave Clock Re-synchronization



26.1 SPI Timing Characteristics

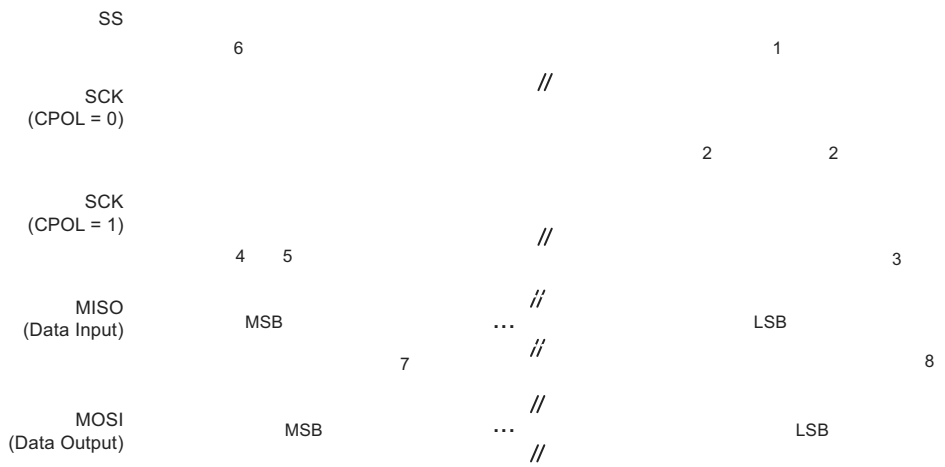
See Figure 26-2 and Figure 26-3 for details.

Table 26-2. SPI Timing Parameters

No.	Description	Mode	Min.	Typ.	Max.	Unit
1	SCK period	Master		See Table 16-4		ns
2	SCK high/low	Master		50% duty cycle		
3	Rise/fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		$0.5 \times t_{sck}$		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	$\overline{SS}$ low to out	Slave		15		
10	SCK period	Slave	$4 \times t_{ck}$			
11	SCK high/low(1)	Slave	$2 \times t_{ck}$			
12	Rise/fall time	Slave		1600		
13	Setup	Slave	10			
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	$\overline{SS}$ low to SCK	Slave	20			

Note: 1. In SPI programming mode the minimum SCK high/low period is:
– 2 t_{CLCL} for f_{CK} < 12MHz
– 3 t_{CLCL} for f_{CK} > 12MHz

Figure 26-2. SPI Interface Timing Requirements (Master Mode)



27.6 BOD Thresholds and Analog Comparator Offset

Figure 27-19. BOD Threshold versus Temperature (BODLEVEL is 4.0V)

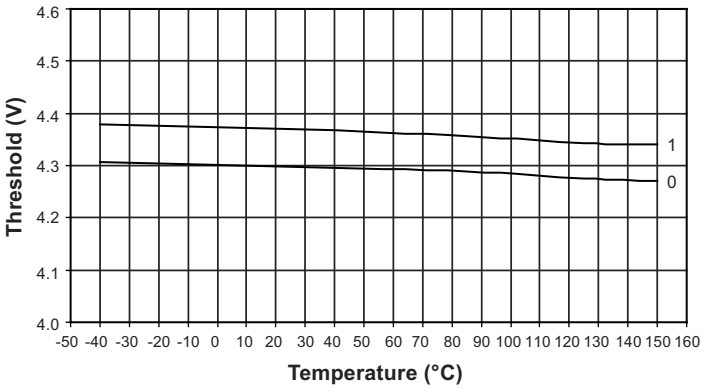


Figure 27-20. BOD Threshold versus Temperature (BODLEVEL is 2.7V)

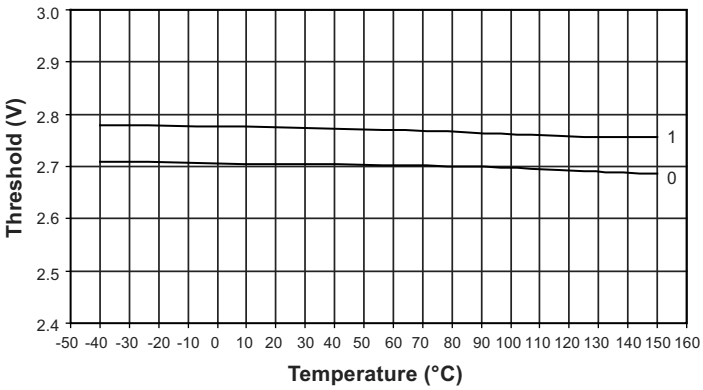


Figure 27-21. Bandgap Voltage versus V_{CC}

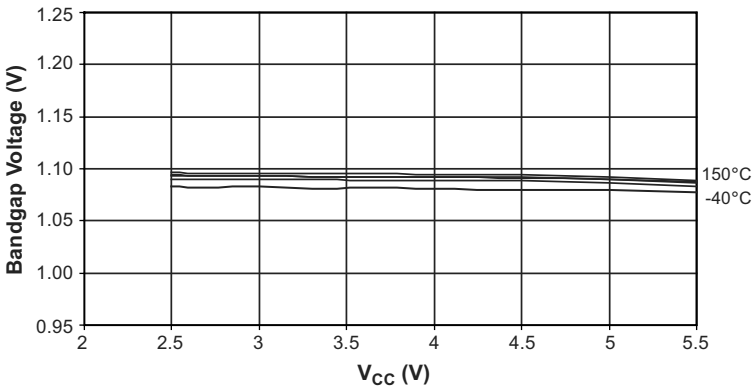
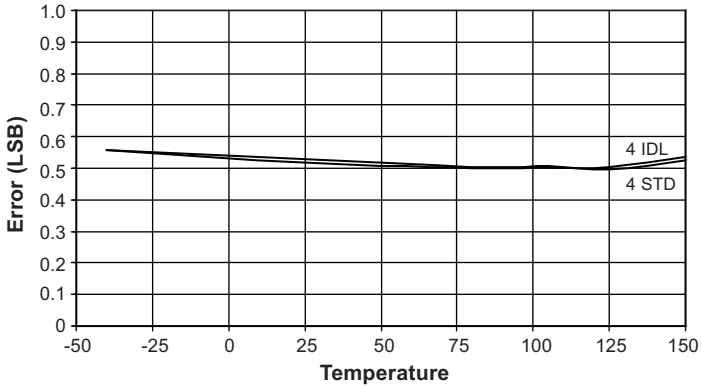


Figure 27-25. Analog to Digital Converter INL versus V_{CC}



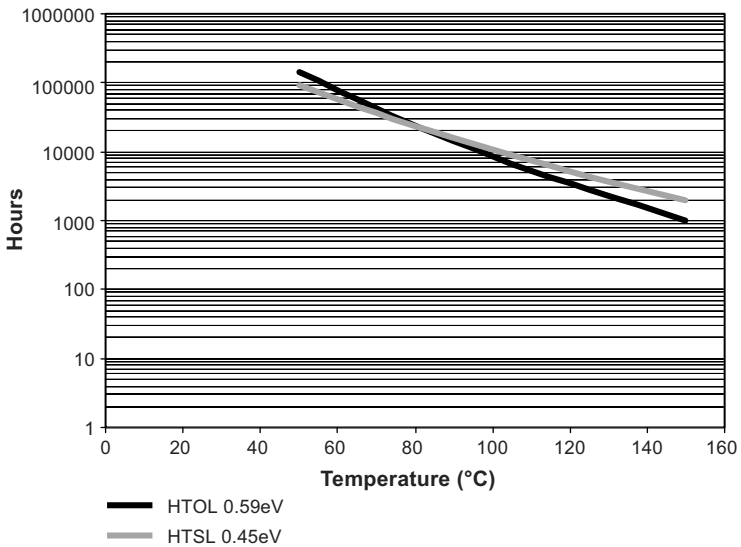
27.8 Grade 0 Qualification

The ATmega88/ATmega168 Automotive has been developed and manufactured according to the most stringent quality assurance requirements of ISO-TS-16949 and verified during product qualification as per AEC-Q100 grade 0.

AEC-Q100 qualification relies on temperature accelerated stress testing. High temperature field usage however may result in less significant stress test acceleration. In order to prevent the risk that ATmega88/ATmega168 Automotive lifetime would not satisfy the application end-of-life reliability requirements, Atmel® has extended the testing, whenever applicable (High Temperature Operating Life Test, High Temperature Storage Life, Data Retention, Thermal Cycles), far beyond the AEC-Q100 requirements. Thereby, Atmel verified the ATmega88/ATmega168 Automotive has a long safe lifetime period after the grade 0 qualification acceptance limits.

The valid domain calculation depends on the activation energy of the potential failure mechanism that is considered. Examples are given in Figure 27-26. Therefore any temperature mission profile which could exceed the AEC-Q100 equivalence domain shall be submitted to Atmel for a thorough reliability analysis.

Figure 27-26. AEC-Q100 Lifetime Equivalence



29. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ST	Z, Rr	Store indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store indirect and post-inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store ndirect and pre-dec.	Z ← Z – 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store indirect with displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store direct to SRAM	(k) ← Rr	None	2
LPM		Load program memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load program memory and post-inc	Rd ← (Z), Z ← Z + 1	None	3
SPM		Store program memory	(Z) ← R1:R0	None	-
IN	Rd, P	In port	Rd ← P	None	1
OUT	P, Rr	Out port	P ← Rr	None	1
PUSH	Rr	Push register on stack	STACK ← Rr	None	2
POP	Rd	Pop register from stack	Rd ← STACK	None	2
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep	(see specific description for sleep function)	None	1
WDR		Watchdog reset	(see specific description for WDR/timer)	None	1
BREAK		Break	For on-chip debug Only	None	N/A

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