

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 ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega88-20pu

If Timer/Counter2 is enabled, it will keep running during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK2, and the Global Interrupt Enable bit in SREG is set.

If Timer/Counter2 is not running, Power-down mode is recommended instead of Power-save mode.

The Timer/Counter2 can be clocked both synchronously and asynchronously in Power-save mode. If Timer/Counter2 is not using the asynchronous clock, the Timer/Counter Oscillator is stopped during sleep. If Timer/Counter2 is not using the synchronous clock, the clock source is stopped during sleep. Note that even if the synchronous clock is running in Power-save, this clock is only available for Timer/Counter2.

7.5 Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Table 7-2. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

Sleep Mode	Active Clock Domains					Oscillators		Wake-up Sources						
	clk _{CPU}	clk _{FLASH}	clk _{IO}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other/O
Idle			X	X	X	X	X ⁽²⁾	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X ⁽²⁾	X ⁽³⁾	X	X	X	X	X	
Power-down								X ⁽³⁾	X				X	
Power-save					X		X	X ⁽³⁾	X	X			X	
Standby ⁽¹⁾						X		X ⁽³⁾	X				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

7.6 Power Reduction Register

The Power Reduction Register, PRR, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. See ["Power-Down Supply Current" on page 315](#) for examples. In all other sleep modes, the clock is already stopped.

7.6.1 Power Reduction Register - PRR

Bit	7	6	5	4	3	2	1	0	
	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC	PRR
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 - PRTWI: Power Reduction TWI**

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

- **Bit 6 - PRTIM2: Power Reduction Timer/Counter2**

Writing a logic one to this bit shuts down the Timer/Counter2 module in synchronous mode (AS2 is 0). When the Timer/Counter2 is enabled, operation will continue like before the shutdown.

- **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 ATmega48/88/168 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**

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.

7.7 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 (clk_{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

- **Bit 5, 2..0 - WDP3..0: Watchdog Timer Prescaler 3, 2, 1 and 0**

The WDP3..0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is running. The different prescaling values and their corresponding time-out periods are shown in [Table 8-6 on page 53](#).

Table 8-6. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at $V_{CC} = 5.0V$
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

11.1.4 Pin Change Interrupt Control Register - PCICR

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

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

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

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

When the PCIE2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PC12 Interrupt Vector. PCINT23..16 pins are enabled individually by the PCMSK2 Register.

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

When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PC11 Interrupt Vector. PCINT14..8 pins are enabled individually by the PCMSK1 Register.

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

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PC10 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 Register.

11.1.5 Pin Change Interrupt Flag Register - PCIFR

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

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

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

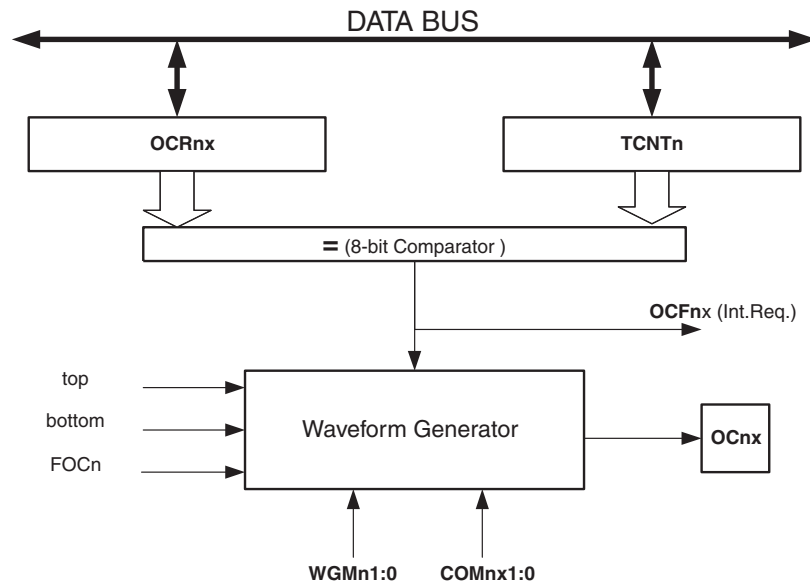
- **Bit 2 - PCIF2: Pin Change Interrupt Flag 2**

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

- **Bit 1 - PCIF1: Pin Change Interrupt Flag 1**

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

Figure 12-3. Output Compare Unit, Block Diagram



The OCR0x Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0x Buffer Register, and if double buffering is disabled the CPU will access the OCR0x directly.

12.4.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0x) bit. Forcing compare match will not set the OCF0x Flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the COM0x1:0 bits settings define whether the OC0x pin is set, cleared or toggled).

12.4.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

12.4.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is downcounting.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See [Table on page 129](#)). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

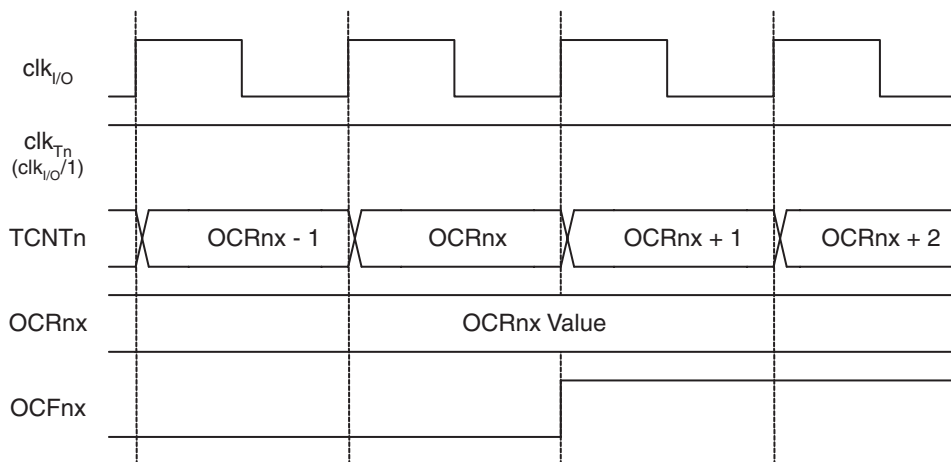
The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

13.9 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCR1x Register is updated with the OCR1x buffer value (only for modes utilizing double buffering). [Figure 13-10](#) shows a timing diagram for the setting of OCF1x.

Figure 13-10. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling



[Figure 13-11](#) shows the same timing data, but with the prescaler enabled.

14. Timer/Counter0 and Timer/Counter1 Prescalers

"8-bit Timer/Counter0 with PWM" on page 88 and "16-bit Timer/Counter1 with PWM" on page 106 share the same prescaler module, but the Timer/Counter can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

14.0.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$.

14.0.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler ($6 > CSn2:0 > 1$). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

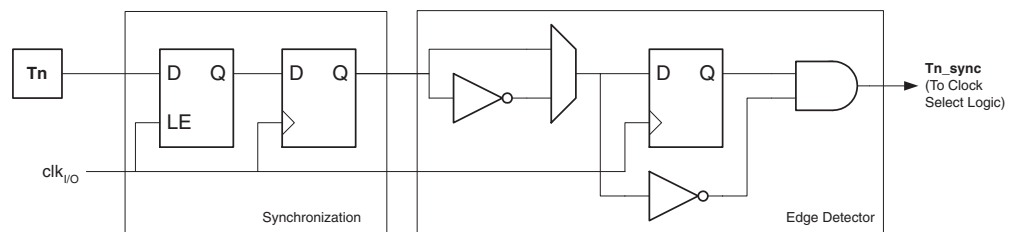
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counter it is connected to.

14.0.3 External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 14-1 shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T1}/clk_{T0} pulse for each positive ($CSn2:0 = 7$) or negative ($CSn2:0 = 6$) edge it detects.

Figure 14-1. T1/T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

17.4 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXCn Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter.

17.9 USART Register Description

17.9.1 USART I/O Data Register n– UDRn

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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 USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDRn Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDREN Flag in the UCSRnA Register is set. Data written to UDRn when the UDREN Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

17.9.2 USART Control and Status Register n A – UCSRnA

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCn: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn Flag can be used to generate a Receive Complete interrupt (see description of the RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDRn). The TXCn Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn Flag can generate a Transmit Complete interrupt (see description of the TXCIEn bit).

- **Bit 5 – UDREN: USART Data Register Empty**

The UDREN Flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREN is one, the buffer is empty, and therefore ready to be written. The UDREN Flag can generate a Data Register Empty interrupt (see description of the UDRIEn bit).

Table 17-9. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	$f_{osc} = 1.0000 \text{ MHz}$				$f_{osc} = 1.8432 \text{ MHz}$				$f_{osc} = 2.0000 \text{ MHz}$			
	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	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	—	—	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	—	—	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	—	—	—	—	—	—	0	0.0%	—	—	—	—
250k	—	—	—	—	—	—	—	—	—	—	0	0.0%
Max. ⁽¹⁾	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

Note: 1. UBRRn = 0, Error = 0.0%

Table 17-12. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	$f_{osc} = 16.0000 \text{ MHz}$				$f_{osc} = 18.4320 \text{ MHz}$				$f_{osc} = 20.0000 \text{ MHz}$			
	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	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	—	—	4	-7.8%	—	—	4	0.0%
1M	0	0.0%	1	0.0%	—	—	—	—	—	—	—	—
Max. ⁽¹⁾	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRRn = 0, Error = 0.0%

baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example⁽¹⁾

```

USART_Init:
    clr r18
    out UBRRnH,r18
    out UBRRnL,r18
    ; Setting the XCKn port pin as output, enables master mode.
    sbi XCKn_DDR, XCKn
    ; Set MSPI mode of operation and SPI data mode 0.
    ldi r18, (1<<UMSELn1) | (1<<UMSELn0) | (0<<UCPHAn) | (0<<UCPOLn)
    out UCSRnC,r18
    ; Enable receiver and transmitter.
    ldi r18, (1<<RXENn) | (1<<TXENn)
    out UCSRnB,r18
    ; Set baud rate.
    ; IMPORTANT: The Baud Rate must be set after the transmitter is enabled!
    out UBRRnH, r17
    out UBRRnL, r18
    ret

```

C Code Example⁽¹⁾

```

void USART_Init( unsigned int baud )
{
    UBRRn = 0;
    /* Setting the XCKn port pin as output, enables master mode. */
    XCKn_DDR |= (1<<XCKn);
    /* Set MSPI mode of operation and SPI data mode 0. */
    UCSRnC = (1<<UMSELn1) | (1<<UMSELn0) | (0<<UCPHAn) | (0<<UCPOLn);
    /* Enable receiver and transmitter. */
    UCSRnB = (1<<RXENn) | (1<<TXENn);
    /* Set baud rate. */
    /* IMPORTANT: The Baud Rate must be set after the transmitter is enabled
    */
    UBRRn = baud;
}

```

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

18.5 Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, i.e. the TXENn bit in the UCSRnB register is set to one. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the Receiver's serial input. The XCKn will in both cases be used as the transfer clock.

23.0.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- The CPU is halted during the Page Write operation.

23.1 Addressing the Flash During Self-Programming

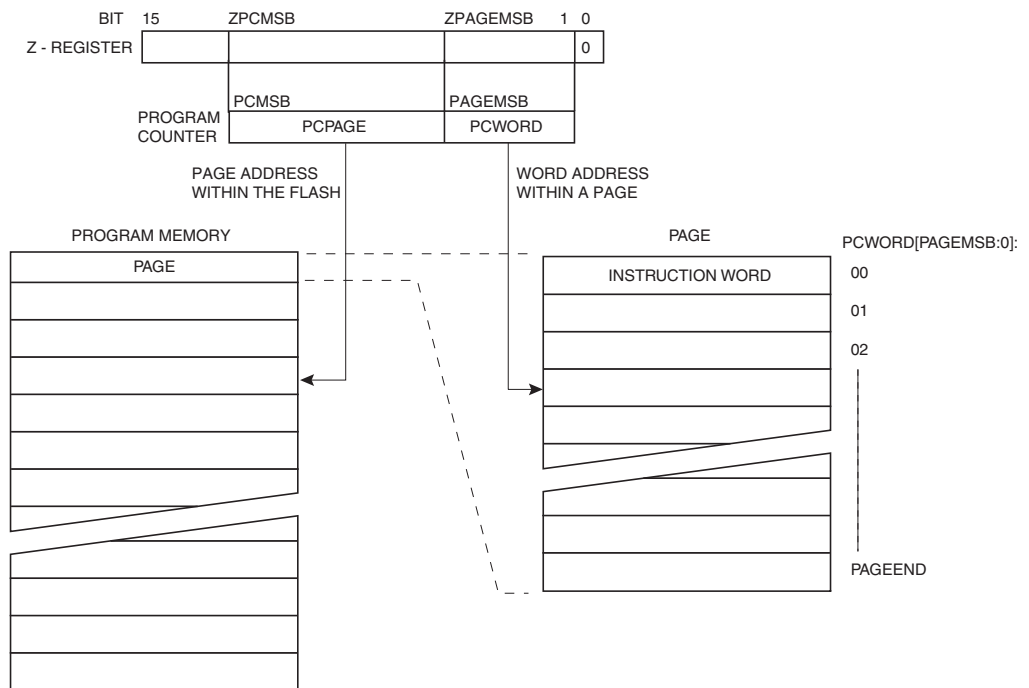
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see [Table 25-8 on page 284](#)), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in [Figure 24-3](#). Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 23-1. Addressing the Flash During SPM⁽¹⁾



Note: 1. The different variables used in [Figure 24-3](#) are listed in [Table 25-8 on page 284](#).

Note: The different BOOTSZ Fuse configurations are shown in [Figure 24-2](#).

Table 24-7. Read-While-Write Limit, ATmega88

Section	Pages	Address
Read-While-Write section (RWW)	96	0x000 - 0xBFF
No Read-While-Write section (NRWW)	32	0xC00 - 0xFFFF

For details about these two section, see ["NRWW – No Read-While-Write Section" on page 265](#) and ["RWW – Read-While-Write Section" on page 265](#)

Table 24-8. Explanation of Different Variables used in [Figure 24-3](#) and the Mapping to the Z-pointer, ATmega88

Variable		Corresponding Z-value ⁽¹⁾	Description
PCMSB	11		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	4		Most significant bit which is used to address the words within one page (32 words in a page requires 5 bits PC [4:0]).
ZPCMSB		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z5	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[11:5]	Z12:Z6	Program counter page address: Page select, for page erase and page write
PCWORD	PC[4:0]	Z5:Z1	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z13: always ignored
 Z0: should be zero for all SPM commands, byte select for the LPM instruction.
 See ["Addressing the Flash During Self-Programming" on page 270](#) for details about the use of Z-pointer during Self-Programming.

25.7.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to ["Programming the Flash" on page 287](#) for details on Command and Data loading):

1. A: Load Command "0100 0000".
2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.

25.7.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to ["Programming the Flash" on page 287](#) for details on Command and Data loading):

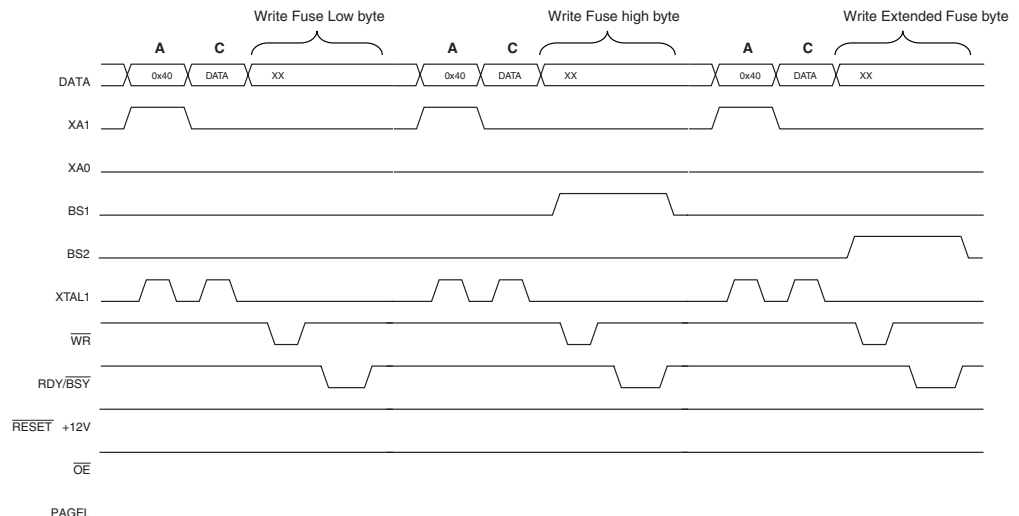
1. A: Load Command "0100 0000".
2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. Set BS1 to "1" and BS2 to "0". This selects high data byte.
4. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.
5. Set BS1 to "0". This selects low data byte.

25.7.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to ["Programming the Flash" on page 287](#) for details on Command and Data loading):

1. 1. A: Load Command "0100 0000".
2. 2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. 3. Set BS1 to "0" and BS2 to "1". This selects extended data byte.
4. 4. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.
5. 5. Set BS2 to "0". This selects low data byte.

Figure 25-5. Programming the FUSES Waveforms



25.7.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to ["Programming the Flash" on page 287](#) for details on Command and Data loading):

27. ATmega48/88/168 Typical Characteristics – Preliminary Data

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A square wave generator with rail-to-rail output is used as clock source.

All Active- and Idle current consumption measurements are done with all bits in the PRR register set and thus, the corresponding I/O modules are turned off. Also the Analog Comparator is disabled during these measurements. [Table 27-1 on page 313](#) and [Table 27-2 on page 314](#) show the additional current consumption compared to I_{CC} Active and I_{CC} Idle for every I/O module controlled by the Power Reduction Register. See ["Power Reduction Register" on page 39](#) for details.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as $C_L \cdot V_{CC} \cdot f$ where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

27.1 Active Supply Current

Figure 27-1. Active Supply Current vs. Frequency (0.1 - 1.0 MHz)

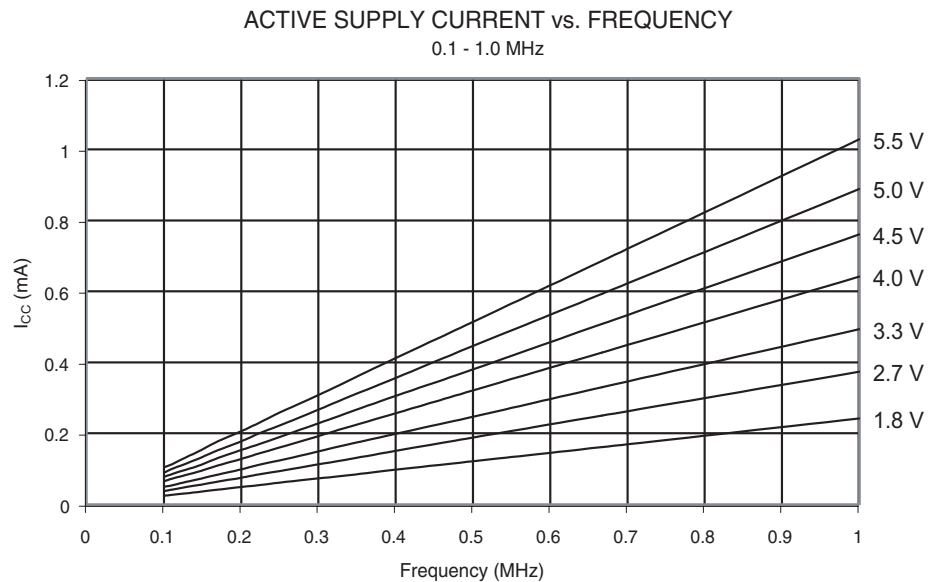


Figure 27-2. Active Supply Current vs. Frequency (1 - 24 MHz)

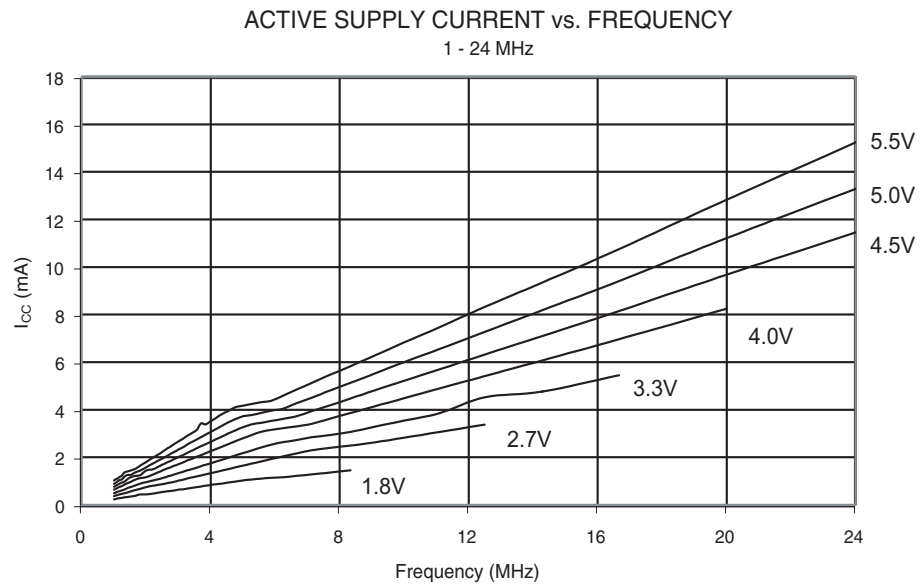


Figure 27-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 128 kHz)

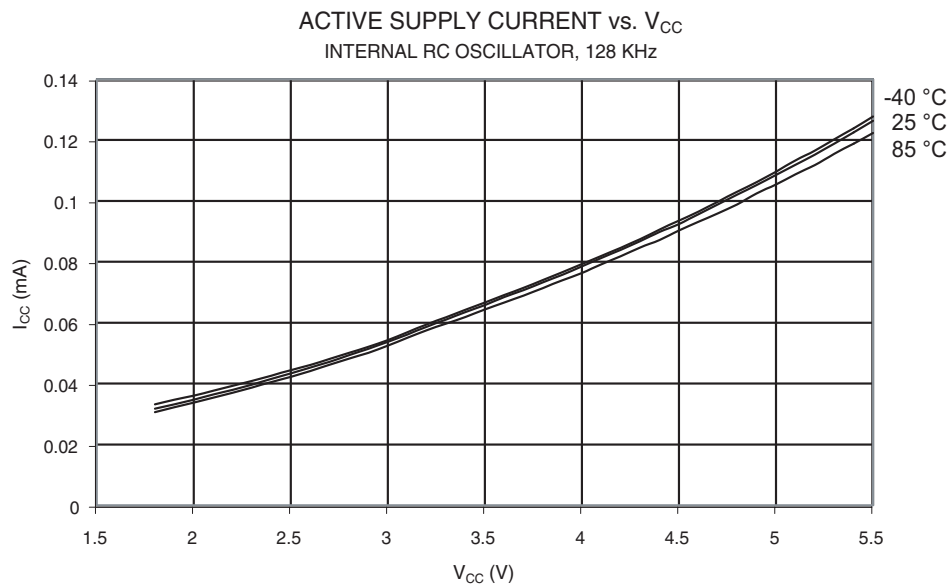


Figure 27-45. Aref Current vs. V_{CC} (ADC at 1 MHz)

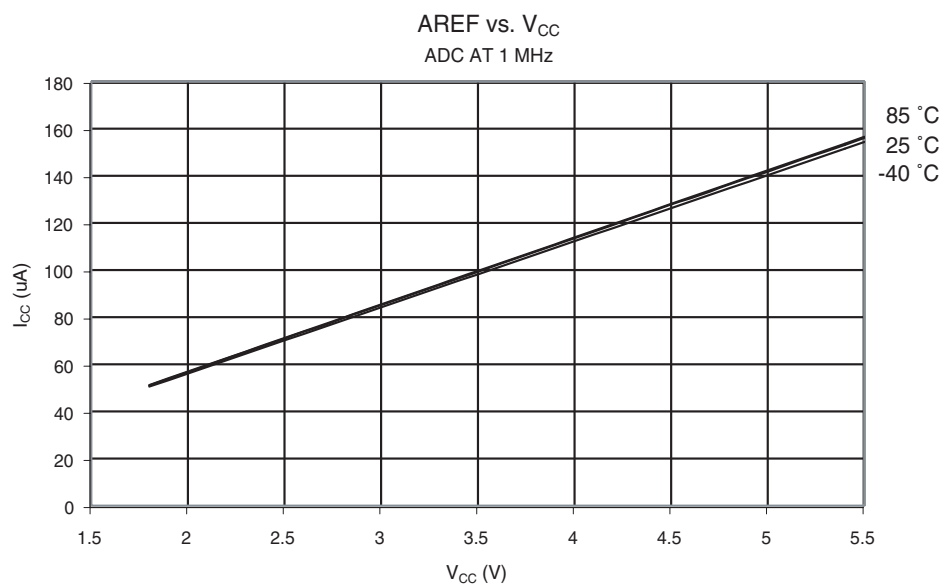
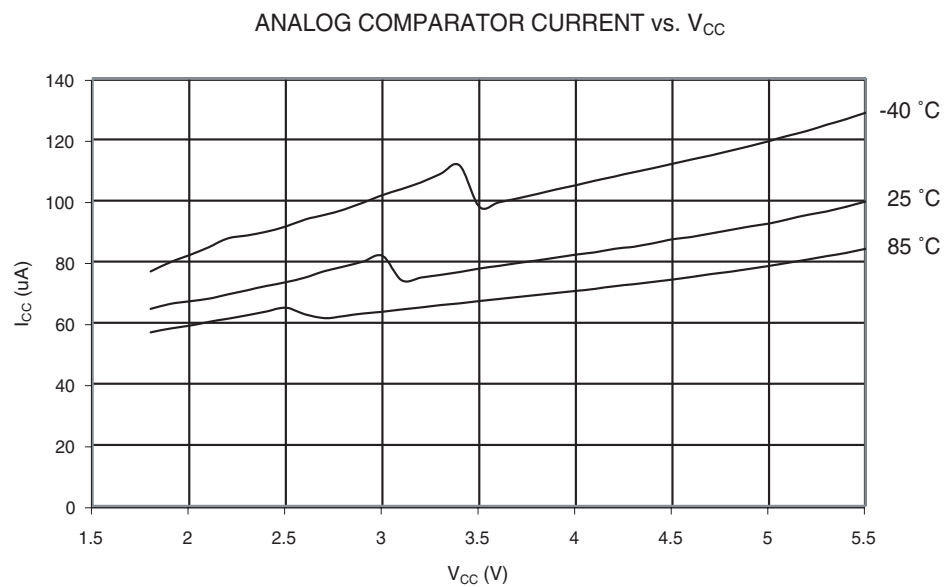


Figure 27-46. Analog Comparator Current vs. V_{CC}



29. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	RdI,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	RdI,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow 0xFF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow 0x00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (0xFF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
JMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP ⁽¹⁾	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL ⁽¹⁾	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow STACK$	None	4
RETI		Interrupt Return	$PC \leftarrow STACK$	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if $(Rd = Rr)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
CP	Rd,Rr	Compare	$Rd - Rr$	Z, N, V, C, H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z, N, V, C, H	1
CPI	Rd,K	Compare Register with Immediate	$Rd - K$	Z, N, V, C, H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if $(Rr(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBRs	Rr, b	Skip if Bit in Register is Set	if $(Rr(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if $(P(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if $(P(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if $(SREG(s) = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if $(SREG(s) = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BREQ	k	Branch if Equal	if $(Z = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRNE	k	Branch if Not Equal	if $(Z = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRCS	k	Branch if Carry Set	if $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRCC	k	Branch if Carry Cleared	if $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRSH	k	Branch if Same or Higher	if $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRLO	k	Branch if Lower	if $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRMI	k	Branch if Minus	if $(N = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRPL	k	Branch if Plus	if $(N = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if $(H = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if $(H = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTS	k	Branch if T Flag Set	if $(T = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTC	k	Branch if T Flag Cleared	if $(T = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if $(V = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if $(V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2