



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	16MHz
Connectivity	I <sup>2</sup> C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atmega8-16au">https://www.e-xfl.com/product-detail/microchip-technology/atmega8-16au</a>

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “[Instruction Set Description](#)” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a Carry in an arithmetic or logic operation. See the “[Instruction Set Description](#)” for detailed information.

## General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

[Figure 3](#) shows the structure of the 32 general purpose working registers in the CPU.

**Figure 3.** AVR CPU General Purpose Working Registers

	7	0	Addr.	
	R0		0x00	
	R1		0x01	
	R2		0x02	
	...			
	R13		0x0D	
	R14		0x0E	
	R15		0x0F	
General Purpose Working Registers	R16		0x10	
	R17		0x11	
	...			
	R26		0x1A	X-register Low Byte
	R27		0x1B	X-register High Byte
	R28		0x1C	Y-register Low Byte
	R29		0x1D	Y-register High Byte
	R30		0x1E	Z-register Low Byte
	R31		0x1F	Z-register High Byte

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in [Figure 3](#), each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-pointer, Y-pointer, and Z-pointer Registers can be set to index any register in the file.

## Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

### • Bits 7..0 – CAL7..0: Oscillator Calibration Value

Writing the calibration byte to this address will trim the Internal Oscillator to remove process variations from the Oscillator frequency. During Reset, the 1MHz calibration value which is located in the signature row High byte (address 0x00) is automatically loaded into the OSCCAL Register. If the internal RC is used at other frequencies, the calibration values must be loaded manually. This can be done by first reading the signature row by a programmer, and then store the calibration values in the Flash or EEPROM. Then the value can be read by software and loaded into the OSCCAL Register. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the Internal Oscillator. Writing 0xFF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0MHz, 2.0MHz, 4.0MHz, or 8.0MHz. Tuning to other values is not guaranteed, as indicated in [Table 11](#).

**Table 11.** Internal RC Oscillator Frequency Range

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)
0x00	50	100
0x7F	75	150
0xFF	100	200

## Alternate Functions of Port D

The Port D pins with alternate functions are shown in [Table 28](#).

**Table 28.** Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input)
PD6	AIN0 (Analog Comparator Positive Input)
PD5	T1 (Timer/Counter 1 External Counter Input)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

The alternate pin configuration is as follows:

- **AIN1 – Port D, Bit 7**

AIN1, Analog Comparator Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

- **AIN0 – Port D, Bit 6**

AIN0, Analog Comparator Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

- **T1 – Port D, Bit 5**

T1, Timer/Counter1 counter source.

- **XCK/T0 – Port D, Bit 4**

XCK, USART external clock.

T0, Timer/Counter0 counter source.

- **INT1 – Port D, Bit 3**

INT1, External Interrupt source 1: The PD3 pin can serve as an external interrupt source.

- **INT0 – Port D, Bit 2**

INT0, External Interrupt source 0: The PD2 pin can serve as an external interrupt source.

- **TXD – Port D, Bit 1**

TXD, Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

- **RXD – Port D, Bit 0**

RXD, Receive Data (Data input pin for the USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

[Table 29 on page 64](#) and [Table 30 on page 64](#) relate the alternate functions of Port D to the overriding signals shown in [Figure 25 on page 56](#).

## Register Description for I/O Ports

### The Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	<b>PORTB7</b>	<b>PORTB6</b>	<b>PORTB5</b>	<b>PORTB4</b>	<b>PORTB3</b>	<b>PORTB2</b>	<b>PORTB1</b>	<b>PORTB0</b>	<b>PORTB</b>
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 Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	<b>DDB7</b>	<b>DDB6</b>	<b>DDB5</b>	<b>DDB4</b>	<b>DDB3</b>	<b>DDB2</b>	<b>DDB1</b>	<b>DDB0</b>	<b>DDRB</b>
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 Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	<b>PINB7</b>	<b>PINB6</b>	<b>PINB5</b>	<b>PINB4</b>	<b>PINB3</b>	<b>PINB2</b>	<b>PINB1</b>	<b>PINB0</b>	<b>PINB</b>
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

### The Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	–	<b>PORTC6</b>	<b>PORTC5</b>	<b>PORTC4</b>	<b>PORTC3</b>	<b>PORTC2</b>	<b>PORTC1</b>	<b>PORTC0</b>	<b>PORTC</b>
Read/Write	R	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 Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	–	<b>DDC6</b>	<b>DDC5</b>	<b>DDC4</b>	<b>DDC3</b>	<b>DDC2</b>	<b>DDC1</b>	<b>DDC0</b>	<b>DDRC</b>
Read/Write	R	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 Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	–	<b>PINC6</b>	<b>PINC5</b>	<b>PINC4</b>	<b>PINC3</b>	<b>PINC2</b>	<b>PINC1</b>	<b>PINC0</b>	<b>PINC</b>
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

### The Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	<b>PORTD7</b>	<b>PORTD6</b>	<b>PORTD5</b>	<b>PORTD4</b>	<b>PORTD3</b>	<b>PORTD2</b>	<b>PORTD1</b>	<b>PORTD0</b>	<b>PORTD</b>
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 Port D Data Direction Register – DDRD

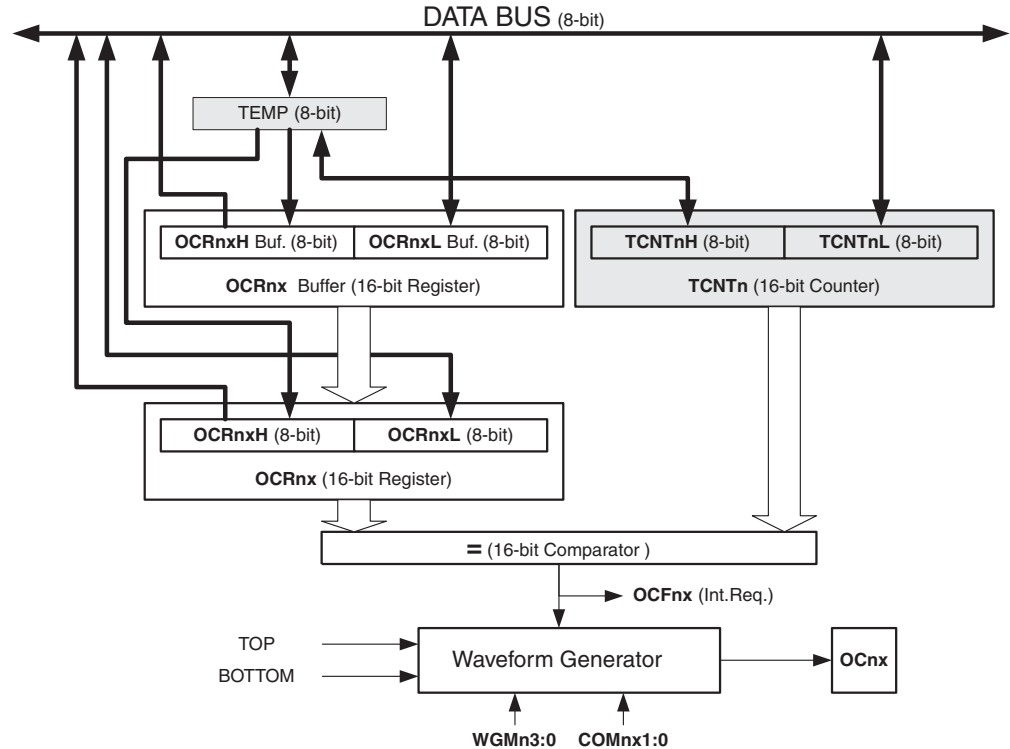
Bit	7	6	5	4	3	2	1	0	
	<b>DDD7</b>	<b>DDD6</b>	<b>DDD5</b>	<b>DDD4</b>	<b>DDD3</b>	<b>DDD2</b>	<b>DDD1</b>	<b>DDD0</b>	<b>DDRD</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### The Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	<b>PIND7</b>	<b>PIND6</b>	<b>PIND5</b>	<b>PIND4</b>	<b>PIND3</b>	<b>PIND2</b>	<b>PIND1</b>	<b>PIND0</b>	<b>PIND</b>
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figure 35 shows a block diagram of the Output Compare unit. The small “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B). The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

**Figure 35.** Output Compare Unit, Block Diagram



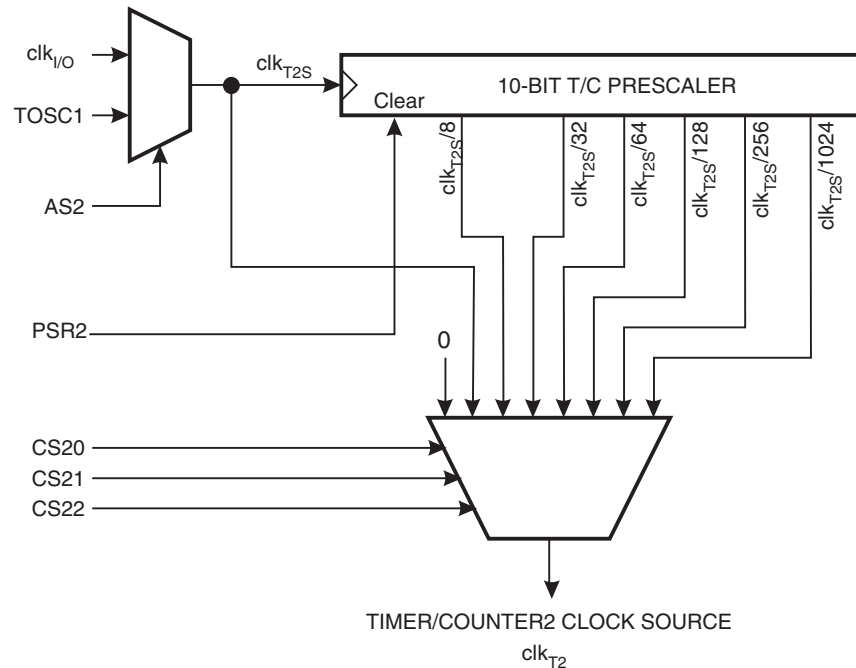
The OCR1x Register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

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

For more information of how to access the 16-bit registers refer to [“Accessing 16-bit Registers” on page 77](#).

## Timer/Counter Prescaler

**Figure 56.** Prescaler for Timer/Counter2



The clock source for Timer/Counter2 is named  $clk_{T2S}$ .  $clk_{T2S}$  is by default connected to the main system I/O clock  $clk_{I/O}$ . By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the TOSC1 pin. This enables use of Timer/Counter2 as a Real Time Counter (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from Port B. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768kHz crystal. Applying an external clock source to TOSC1 is not recommended.

For Timer/Counter2, the possible prescaled selections are:  $clk_{T2S}/8$ ,  $clk_{T2S}/32$ ,  $clk_{T2S}/64$ ,  $clk_{T2S}/128$ ,  $clk_{T2S}/256$ , and  $clk_{T2S}/1024$ . Additionally,  $clk_{T2S}$  as well as 0 (stop) may be selected. Setting the PSR2 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler.

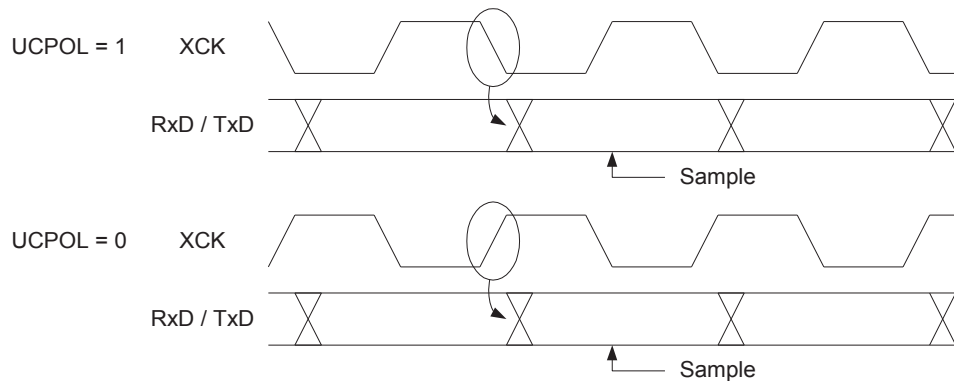
## Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### • Bit 1 – PSR2: Prescaler Reset Timer/Counter2

When this bit is written to one, the Timer/Counter2 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always be read as zero if Timer/Counter2 is clocked by the internal CPU clock. If this bit is written when Timer/Counter2 is operating in Asynchronous mode, the bit will remain one until the prescaler has been reset.

**Figure 63.** Synchronous Mode XCK Timing



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As [Figure 63](#) shows, when UCPOL is zero the data will be changed at rising XCK edge and sampled at falling XCK edge. If UCPOL is set, the data will be changed at falling XCK edge and sampled at rising XCK edge.

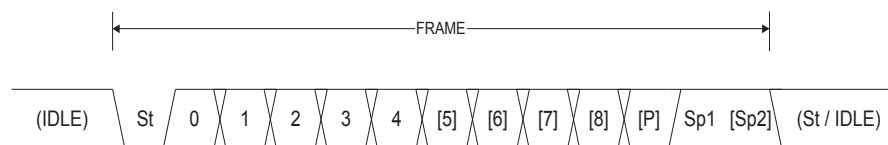
## Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. [Figure 64](#) illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

**Figure 64.** Frame Formats



**St** Start bit, always low

**(n)** Data bits (0 to 8)

**P** Parity bit. Can be odd or even

**Sp** Stop bit, always high

**IDLE** No transfers on the communication line (RxD or TxD). An IDLE line must be high

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.



Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC Flag and this might accidentally be cleared when using SBI or CBI instructions.

## Accessing UBRRH/UCSRC Registers

### Write Access

The UBRRH Register shares the same I/O location as the UCSRC Register. Therefore some special consideration must be taken when accessing this I/O location.

When doing a write access of this I/O location, the high bit of the value written, the USART Register Select (URSEL) bit, controls which one of the two registers that will be written. If URSEL is zero during a write operation, the UBRRH value will be updated. If URSEL is one, the UCSRC setting will be updated.

The following code examples show how to access the two registers.

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

```
...
; Set UBRRH to 2
ldi r16,0x02
out UBRRH,r16
...
; Set the USBS and the UCSZ1 bit to one, and
; the remaining bits to zero.
ldi r16,(1<<URSEL) | (1<<USBS) | (1<<UCSZ1)
out UCSRC,r16
...
```

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

```
...
/* Set UBRRH to 2 */
UBRRH = 0x02;
...
/* Set the USBS and the UCSZ1 bit to one, and */
/* the remaining bits to zero. */
UCSRC = (1<<URSEL) | (1<<USBS) | (1<<UCSZ1);
...
```

Note: 1. See [“About Code Examples” on page 8](#)

As the code examples illustrate, write accesses of the two registers are relatively unaffected of the sharing of I/O location.

The TWINT Flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition
- After the TWI has transmitted SLA+R/W
- After the TWI has transmitted an address byte
- After the TWI has lost arbitration
- After the TWI has been addressed by own slave address or general call
- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a Slave
- When a bus error has occurred due to an illegal START or STOP condition

## TWI Register Description

### TWI Bit Rate Register – TWBR

Bit	7	6	5	4	3	2	1	0	
	<b>TWBR7</b>	<b>TWBR6</b>	<b>TWBR5</b>	<b>TWBR4</b>	<b>TWBR3</b>	<b>TWBR2</b>	<b>TWBR1</b>	<b>TWBR0</b>	<b>TWBR</b>
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	

#### • Bits 7..0 – TWI Bit Rate Register

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See [“Bit Rate Generator Unit” on page 164](#) for calculating bit rates.

### TWI Control Register – TWCR

Bit	7	6	5	4	3	2	1	0	
	<b>TWINT</b>	<b>TWEA</b>	<b>TWSTA</b>	<b>TWSTO</b>	<b>TWWC</b>	<b>TWEN</b>	<b>–</b>	<b>TWIE</b>	<b>TWCR</b>
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a Receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

#### • Bit 7 – TWINT: TWI Interrupt Flag

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. See “Internal Voltage Reference” on page 42.

- **Bit 5 – ACO: Analog Comparator Output**

The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt Handling Vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the Input Capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the Input Capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set.

- **Bits 1,0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

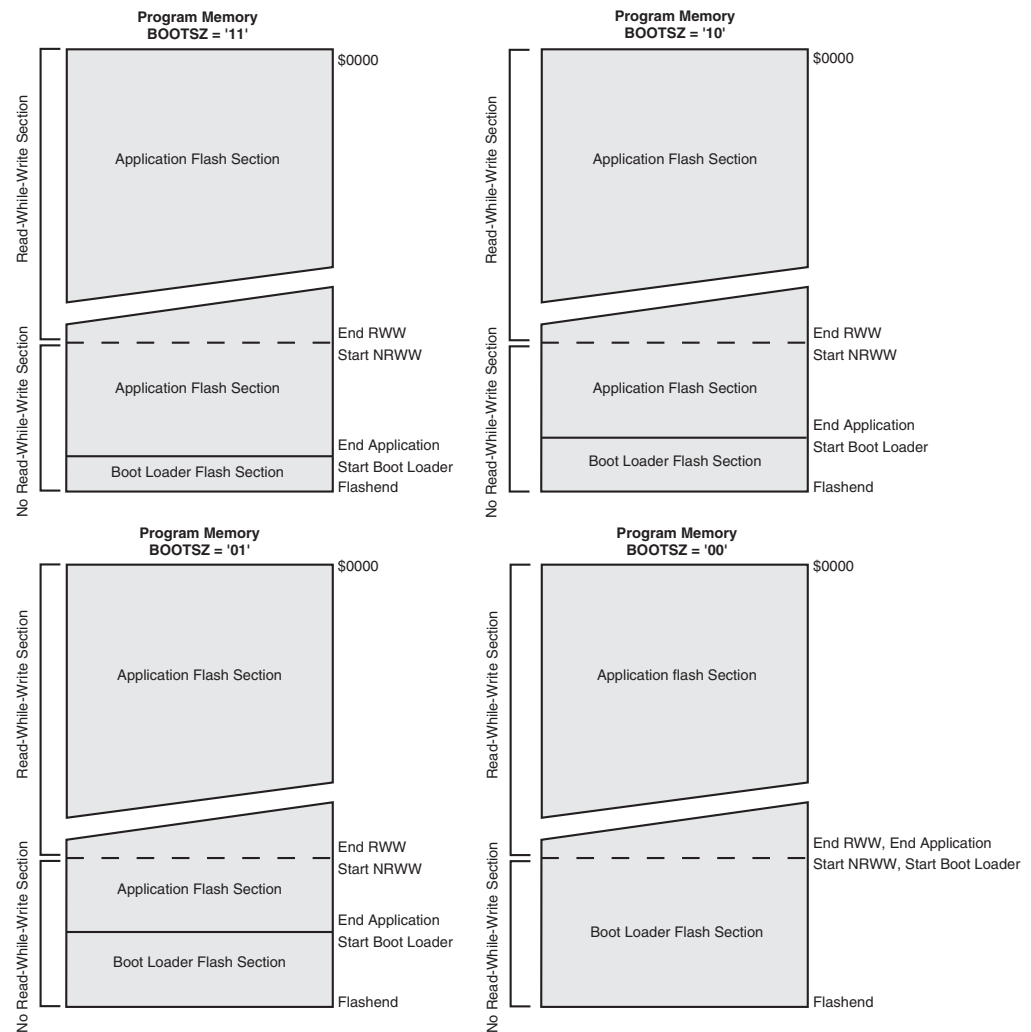
These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 71.

**Table 71.** ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

**Figure 102. Memory Sections<sup>(1)</sup>**



Note: 1. The parameters in the figure are given in [Table 82 on page 213](#)

## Boot Loader Lock Bits

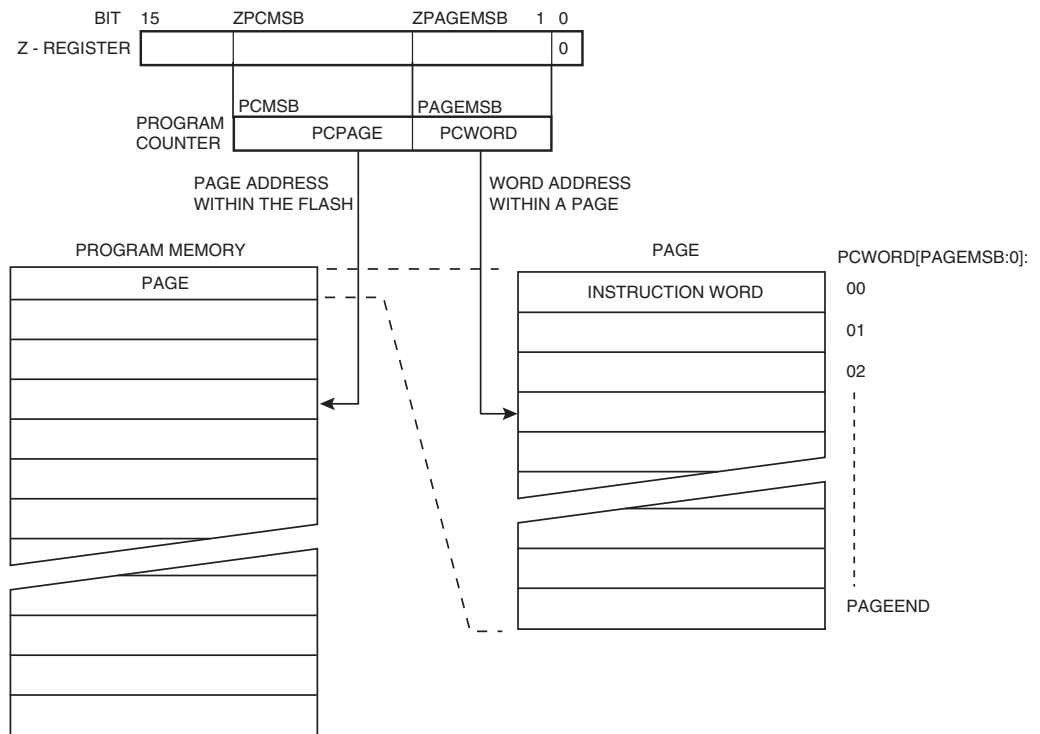
If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock Bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU
- To protect only the Boot Loader Flash section from a software update by the MCU
- To protect only the Application Flash section from a software update by the MCU
- Allow software update in the entire Flash

See [Table 78 on page 205](#) and [Table 79 on page 205](#) for further details. The Boot Lock Bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a chip erase command only. The general Write Lock (Lock bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock bit mode 3) does not control reading nor writing by LPM/SPM, if it is attempted.

**Figure 103.** Addressing the Flash during SPM<sup>(1)</sup>



- Notes:
1. The different variables used in the figure are listed in [Table 84 on page 214](#)
  2. PCPAGE and PCWORD are listed in [Table 89 on page 218](#)

## Self-Programming the Flash

The Program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

Alternative 1, fill the buffer before a page erase.

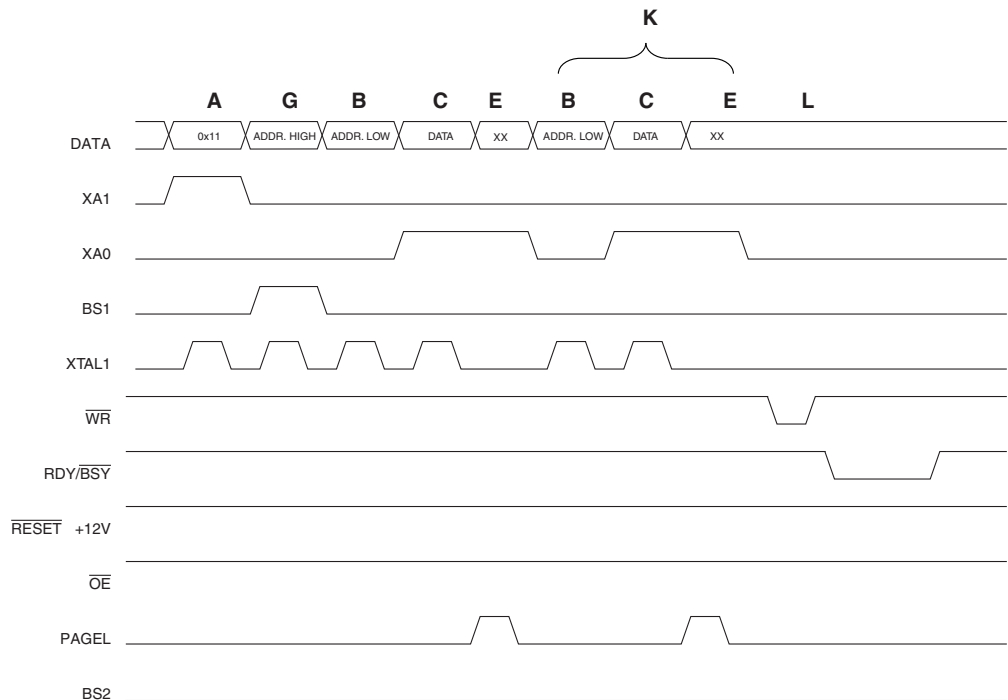
- Fill temporary page buffer
- Perform a page erase
- Perform a page write

Alternative 2, fill the buffer after page erase.

- Perform a page erase
- Fill temporary page buffer
- Perform a page write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the boot loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page. See [“Simple Assembly Code Example for a Boot Loader” on page 212](#) for an assembly code example.

**Figure 107. Programming the EEPROM Waveforms**



## Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to [“Programming the Flash” on page 222](#) for details on Command and Address loading):

1. A: Load Command “0000 0010”
2. G: Load Address High byte (0x00 - 0xFF)
3. B: Load Address Low byte (0x00 - 0xFF)
4. Set  $\overline{OE}$  to “0”, and BS1 to “0”. The Flash word Low byte can now be read at DATA
5. Set BS1 to “1”. The Flash word High byte can now be read at DATA
6. Set  $\overline{OE}$  to “1”

## Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to [“Programming the Flash” on page 222](#) for details on Command and Address loading):

1. A: Load Command “0000 0011”
2. G: Load Address High byte (0x00 - 0xFF)
3. B: Load Address Low byte (0x00 - 0xFF)
4. Set  $\overline{OE}$  to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA
5. Set  $\overline{OE}$  to “1”

## Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to [“Programming the Flash” on page 222](#) 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 and BS2 to “0”
4. Give  $\overline{WR}$  a negative pulse and wait for RDY/BSY to go high

## Serial Programming Algorithm

When writing serial data to the ATmega8, data is clocked on the rising edge of SCK.

When reading data from the ATmega8, data is clocked on the falling edge of SCK. See [Figure 113 on page 232](#) for timing details.

To program and verify the ATmega8 in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in [Table 98 on page 233](#)):

1. Power-up sequence:  
Apply power between  $V_{CC}$  and GND while  $\overline{RESET}$  and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during Power-up. In this case,  $\overline{RESET}$  must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”
2. Wait for at least 20ms and enable Serial Programming by sending the Programming Enable serial instruction to pin MOSI
3. The Serial Programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give  $\overline{RESET}$  a positive pulse and issue a new Programming Enable command
4. The Flash is programmed one page at a time. The page size is found in [Table 89 on page 218](#). The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the Load Program memory Page instruction. To ensure correct loading of the page, the data Low byte must be loaded before data High byte is applied for a given address. The Program memory Page is stored by loading the Write Program memory Page instruction with the 7MSB of the address. If polling is not used, the user must wait at least  $t_{WD\_FLASH}$  before issuing the next page (see [Table 97 on page 232](#)).

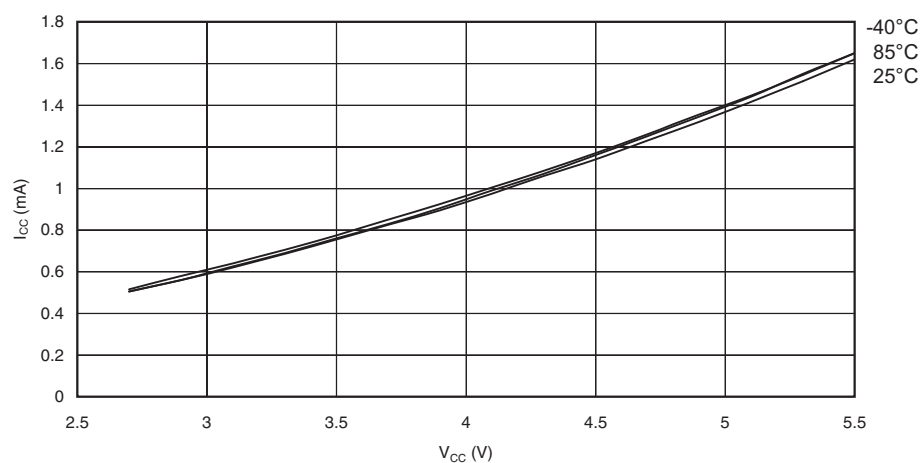
Note: If other commands than polling (read) are applied before any write operation (FLASH, EEPROM, Lock Bits, Fuses) is completed, it may result in incorrect programming

5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least  $t_{WD\_EEPROM}$  before issuing the next byte (see [Table 97 on page 232](#)). In a chip erased device, no 0xFFs in the data file(s) need to be programmed
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO
7. At the end of the programming session,  $\overline{RESET}$  can be set high to commence normal operation
8. Power-off sequence (if needed):  
Set  $\overline{RESET}$  to “1”  
Turn  $V_{CC}$  power off

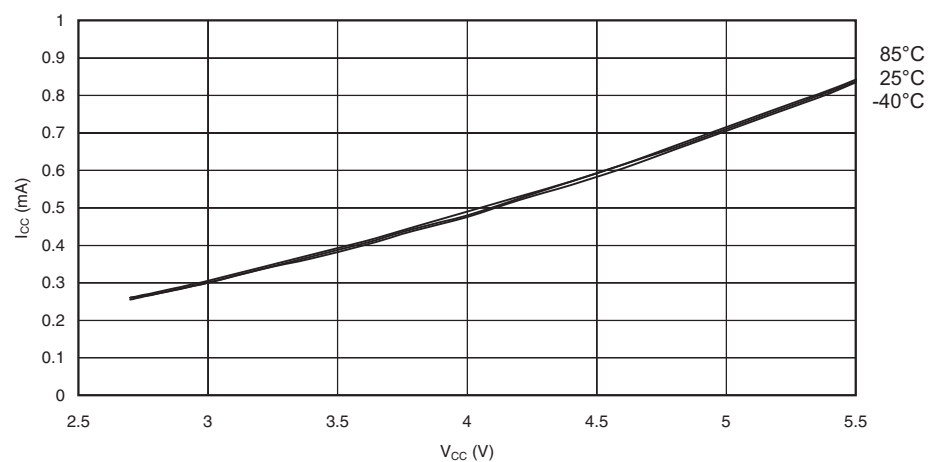
## Data Polling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value 0xFF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value 0xFF, so when programming this value, the user will have to wait for at least  $t_{WD\_FLASH}$  before programming the next page. As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. See [Table 97 on page 232](#) for  $t_{WD\_FLASH}$  value.

**Figure 129.** Idle Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 2MHz)

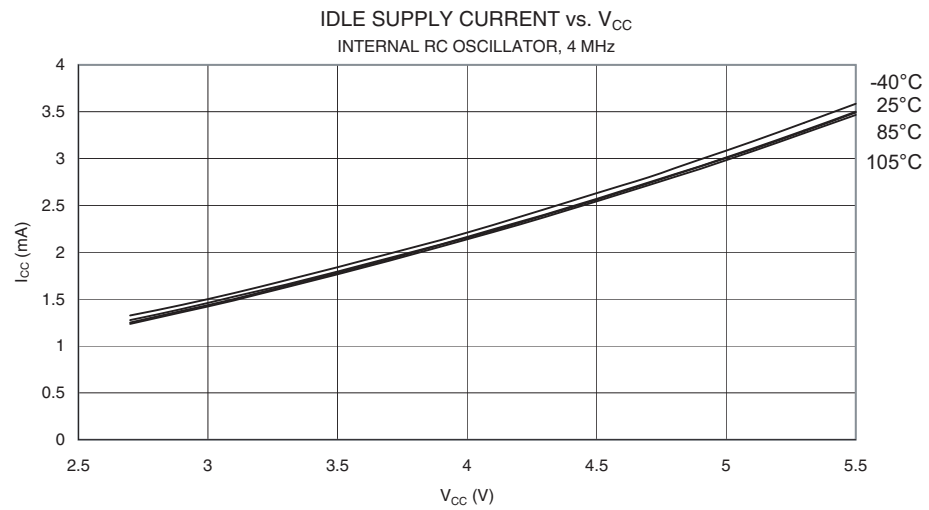


**Figure 130.** Idle Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 1MHz)

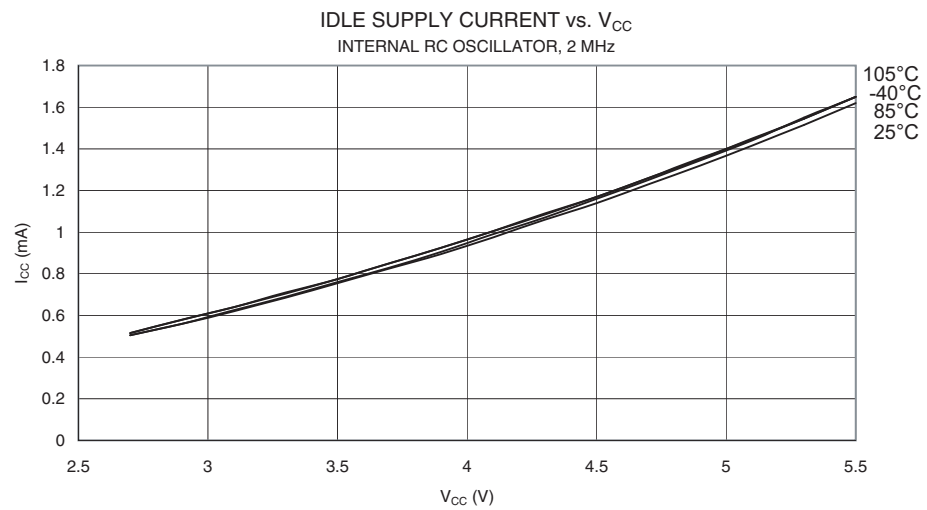




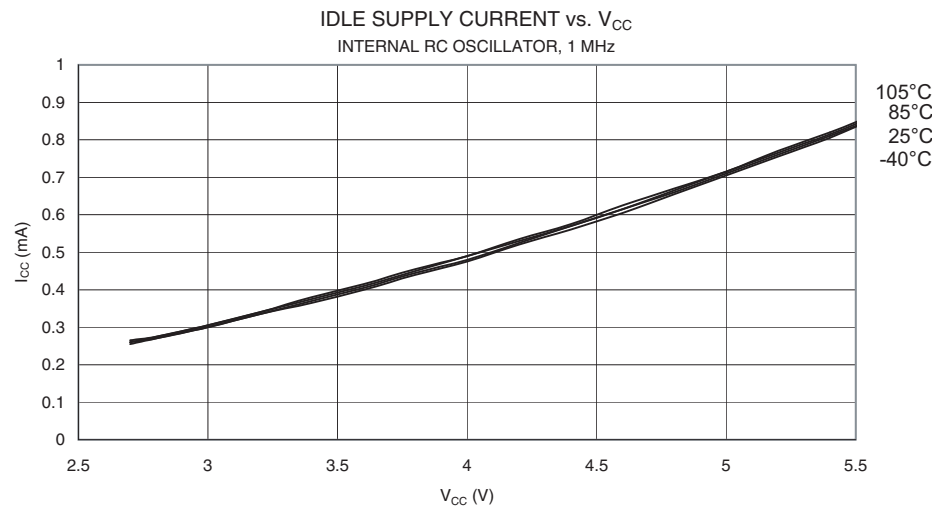
**Figure 0-6.** Idle Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 4 MHz)



**Figure 0-7.** Idle Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 2 MHz)

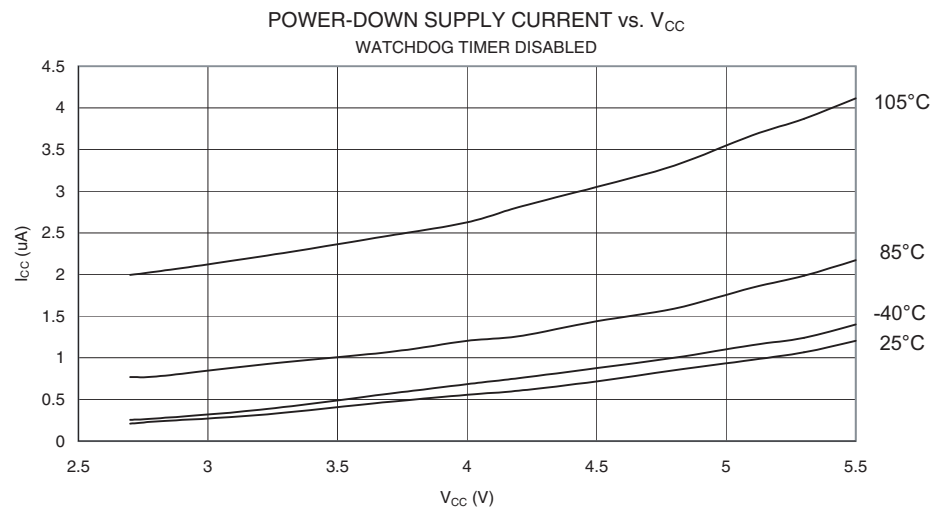


**Figure 0-8.** Idle Supply Current vs.  $V_{CC}$  (Internal RC Oscillator, 1 MHz)



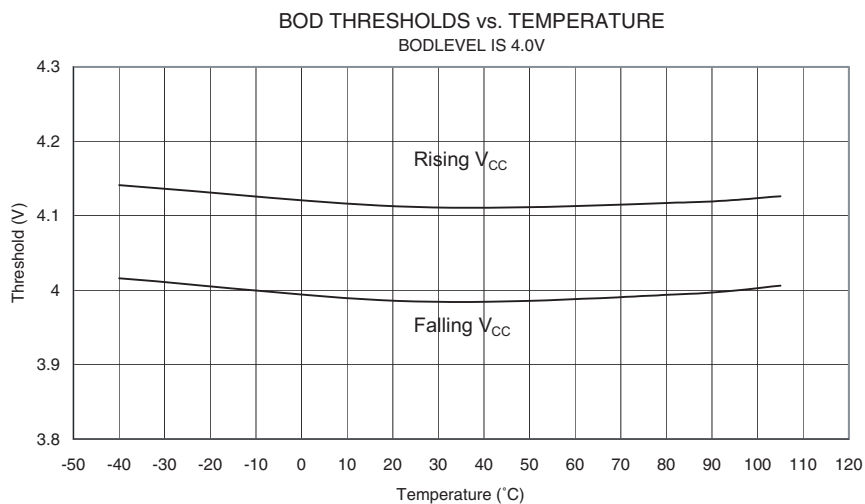
## Power-down Supply Current

**Figure 0-9.** Power-down Supply Current vs.  $V_{CC}$  (Watchdog Timer Disabled)

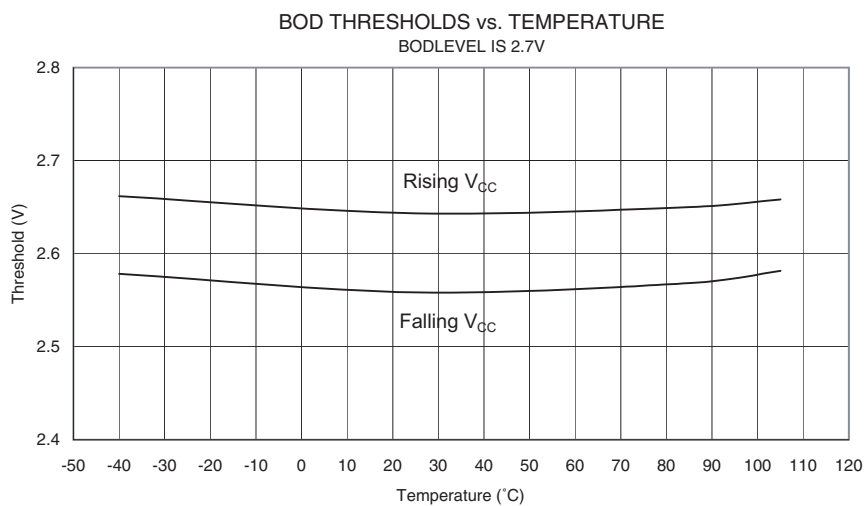


## Bod Thresholds and Analog Comparator Offset

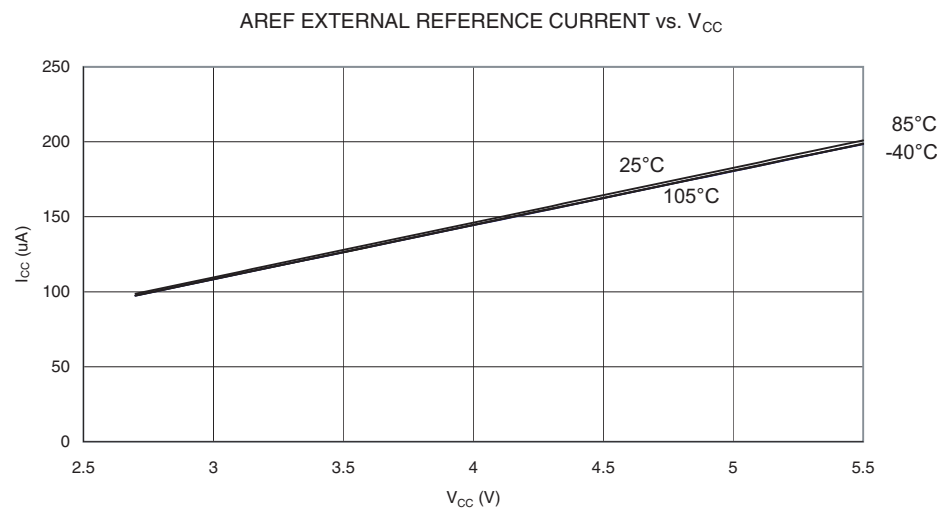
**Figure 0-32.** BOD Thresholds vs. Temperature (BOD Level is 4.0V)



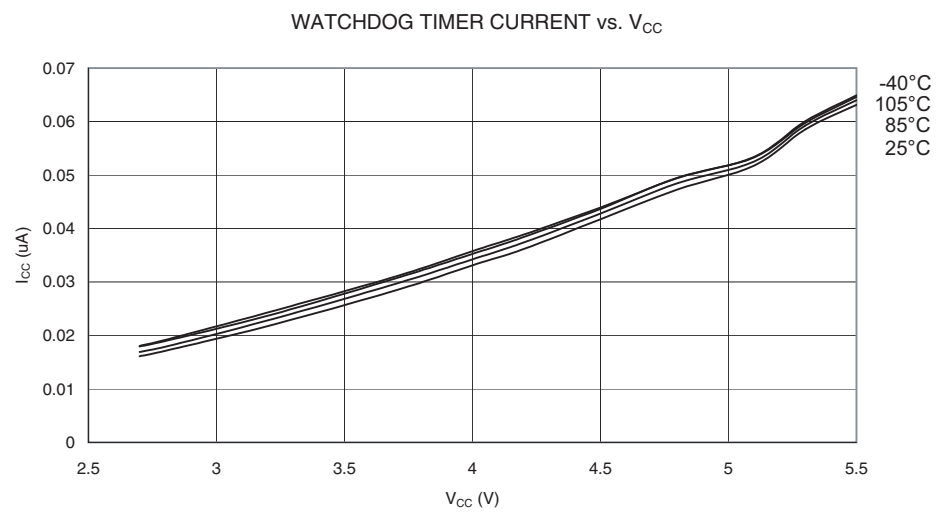
**Figure 0-33.** BOD Thresholds vs. Temperature (BOD Level is 2.7V)



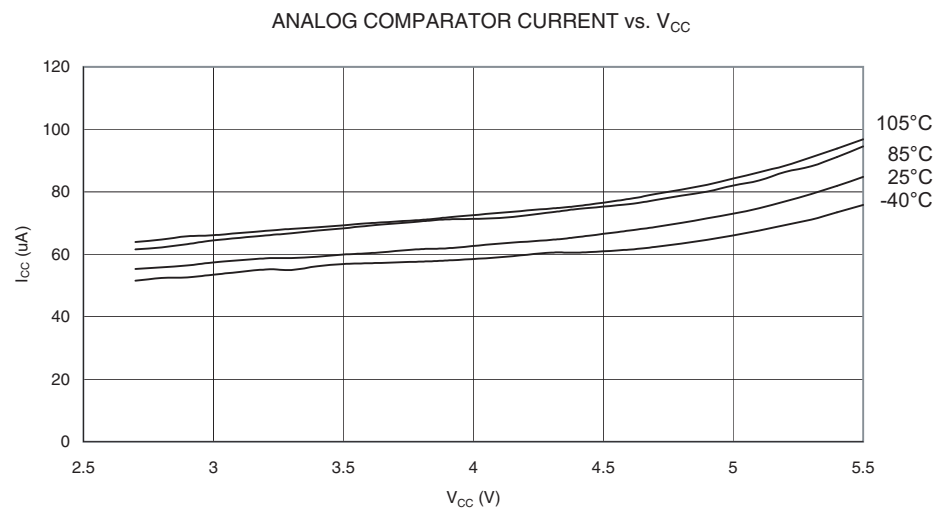
**Figure 0-48.** AREF External Reference Current vs.  $V_{CC}$



**Figure 0-49.** Watchdog Timer Current vs.  $V_{CC}$



**Figure 0-50.** Analog Comparator Current vs.  $V_{CC}$



**Figure 0-51.** Programming Current vs.  $V_{CC}$

