Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega6490-16ai

priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). Refer to “Interrupts” on page 49 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see “Boot Loader Support – Read-While-Write Self-Programming” on page 278.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in  r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi EECR, EEMWE   ; start EEPROM write
sbi EECR, EEWE
out SREG, r16     ; restore SREG value (I-bit)
```

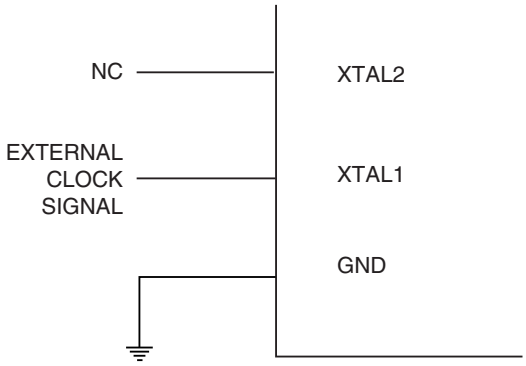
C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
__disable_interrupt();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

8.6 External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 8-3. To run the device on an external clock, the CKSEL Fuses must be programmed to “0000”.

Figure 8-3. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 8-10.

Table 8-9. Crystal Oscillator Clock Frequency

CKSEL3..0	Frequency Range
0000	0 - 16MHz

Table 8-10. Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4.1ms	Fast rising power
10	6 CK	14CK + 65ms	Slowly rising power
11	Reserved		

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to “System Clock Prescaler” on page 32 for details.

8.7 Clock Output Buffer

When the CKOUT Fuse is programmed, the system Clock will be output on CLKO. This mode is suitable when the chip clock is used to drive other circuits on the system. The clock will be output also during reset and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including internal RC Oscillator, can be selected when CLKO

12. External Interrupts

The External Interrupts are triggered by the INT0 pin or any of the PCINT30..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT30..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. The pin change interrupt PCI1 will trigger if any enabled PCINT15..8 pin toggles. Pin change interrupts PCI0 will trigger if any enabled PCINT7..0 pin toggles. The PCMSK3, PCMSK2, PCMSK1, and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT30..0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A – EICRA. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 requires the presence of an I/O clock, described in “Clock Systems and their Distribution” on page 26. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses as described in “System Clock and Clock Options” on page 26.

12.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 12-1.

Figure 12-1. Pin Change Interrupt

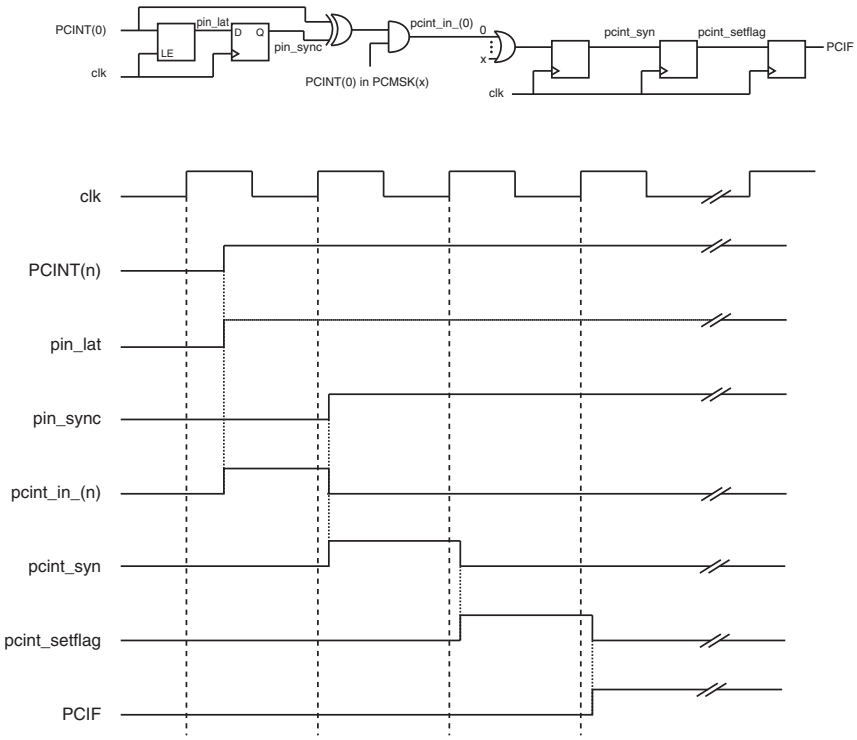


Table 13-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 13-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 13-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

Some pins are connected to different LCS segments on ATmega3290/6490 and ATmega3290/6490. See pinout on “Pinout ATmega3290/6490” on page 2 and “Pinout ATmega329/649” on page 3 for details.

- **$\overline{SS}$/PCINT8 – Port B, Bit 0**

$\overline{SS}$: Slave Port Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB0. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 bit

PCINT8, Pin Change Interrupt Source 8: The PB0 pin can serve as an external interrupt source.

Table 13-7 and Table 13-8 relate the alternate functions of Port B to the overriding signals shown in Figure 13-5 on page 65. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 13-7. Overriding Signals for Alternate Functions in PB7:PB4

Signal Name	PB7/OC2A/ PCINT15	PB6/OC1B/ PCINT14	PB5/OC1A/ PCINT13	PB4/OC0A/ PCINT12
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2A ENABLE	OC1B ENABLE	OC1A ENABLE	OC0A ENABLE
PVOV	OC2A	OC1B	OC1A	OC0A
PTOE	–	–	–	–
DIEOE	PCINT15 • PCIE1	PCINT14 • PCIE1	PCINT13 • PCIE1	PCINT12 • PCIE1
DIEOV	1	1	1	1
DI	PCINT15 INPUT	PCINT14 INPUT	PCINT13 INPUT	PCINT12 INPUT
AIO	–	–	–	–

14. 8-bit Timer/Counter0 with PWM

14.1 Features

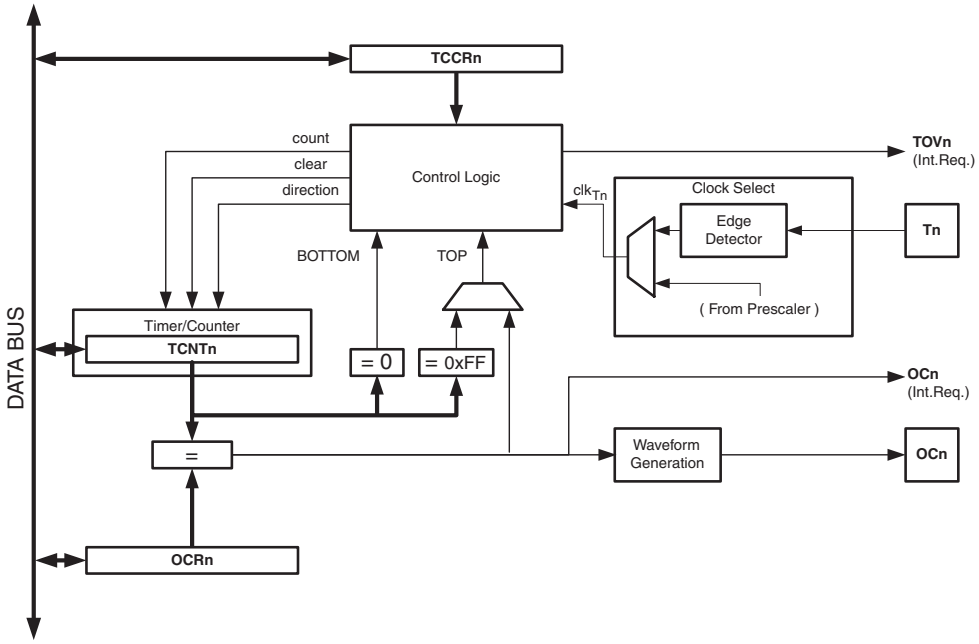
Timer/Counter0 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- **Single Compare Unit Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **External Event Counter**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV0 and OCF0A)**

14.2 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 14-1. For the actual placement of I/O pins, refer to "Pinout ATmega3290/6490" on page 2 and "Pinout ATmega329/649" on page 3. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "Register Description" on page 103.

Figure 14-1. 8-bit Timer/Counter Block Diagram



14.2.1 Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0A) 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

Table 14-2. Waveform Generation Mode Bit Description[1]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0A at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0A	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

- **Bit 5:4 – COM0A1:0: Compare Match Output Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM01:0 bit setting. Table 14-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM).

Table 14-3. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 14-4 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 14-4. Compare Output Mode, Fast PWM Mode[1]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Reserved
1	0	Clear OC0A on compare match, set OC0A at BOTTOM, (non-inverting mode)
1	1	Set OC0A on compare match, clear OC0A at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See “Fast PWM Mode” on page 98 for more details.

Table 14-5 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.

16.11.9 TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	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 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 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 16-5 on page 134 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.

When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM21:0 bit setting. Table 17-3 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to a normal or CTC mode (non-PWM).

Table 17-3. Compare Output Mode, non-PWM Mode

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Toggle OC2A on compare match.
1	0	Clear OC2A on compare match.
1	1	Set OC2A on compare match.

Table 17-4 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 17-4. Compare Output Mode, Fast PWM Mode(1)

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Reserved
1	0	Clear OC2A on compare match, set OC2A at BOTTOM, (non-inverting mode).
1	1	Set OC2A on compare match, clear OC2A at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See “Fast PWM Mode” on page 146 for more details.

Table 17-5 shows the COM21:0 bit functionality when the WGM21:0 bits are set to phase correct PWM mode.

Table 17-5. Compare Output Mode, Phase Correct PWM Mode(1)

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Reserved
1	0	Clear OC2A on compare match when up-counting. Set OC2A on compare match when counting down.
1	1	Set OC2A on compare match when up-counting. Clear OC2A on compare match when counting down.

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See “Phase Correct PWM Mode” on page 148 for more details.

18. SPI – Serial Peripheral Interface

18.1 Features

The ATmega329/3290/649/6490 SPI includes the following features:

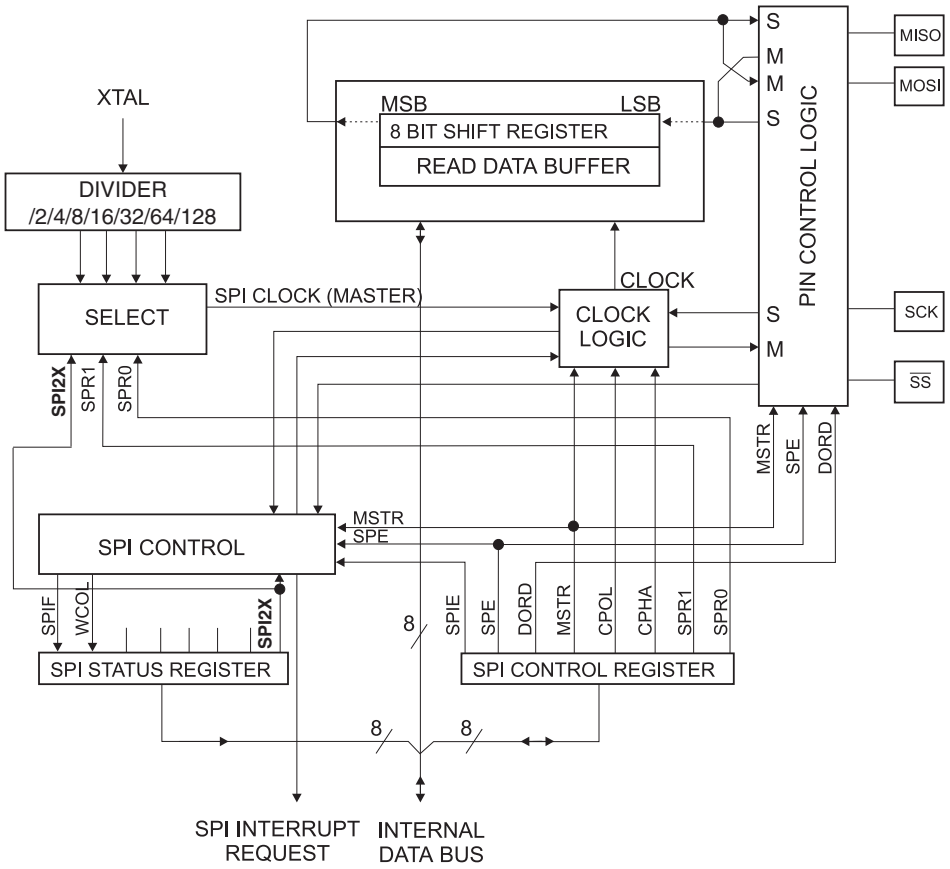
- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master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

18.2 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega329/3290/649/6490 and peripheral devices or between several AVR devices. A simplified block diagram of the Serial Peripheral Interface is shown in Figure 18-1.

The PRSPI bit in "Power Reduction Register" on page 37 must be written to zero to enable the SPI module.

Figure 18-1. SPI Block Diagram(1)



Note: 1. Refer to Figure 1-1 on page 2, and Table 13-6 on page 68 for SPI pin placement.

22.7 Register Description

22.7.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 22-3. 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 22-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC 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 “ADCL and ADCH – The ADC Data Register” on page 226.

- **Bits 4:0 – MUX4:0: Analog Channel Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 22-4 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).

Table 25-7. ATmega329/649 Boundary-scan Order, 64-pin (Continued)

Bit Number	Signal Name	Module
157	PE0.Data	Port E
156	PE0.Control	
155	PE0.Pull-up_Enable	
154	PE1.Data	
153	PE1.Control	
152	PE1.Pull-up_Enable	
151	PE2.Data	
150	PE2.Control	
149	PE2.Pull-up_Enable	
148	PE3.Data	
147	PE3.Control	
146	PE3.Pull-up_Enable	
145	PE4.Data	
144	PE4.Control	
143	PE4.Pull-up_Enable	
142	PE5.Data	
141	PE5.Control	
140	PE5.Pull-up_Enable	
139	PE6.Data	
138	PE6.Control	
137	PE6.Pull-up_Enable	
136	PE7.Data	
135	PE7.Control	
134	PE7.Pull-up_Enable	

used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

3. Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

26.7.11 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 26-5 shows the typical programming time for Flash accesses from the CPU.

Table 26-5. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7ms	4.5ms

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

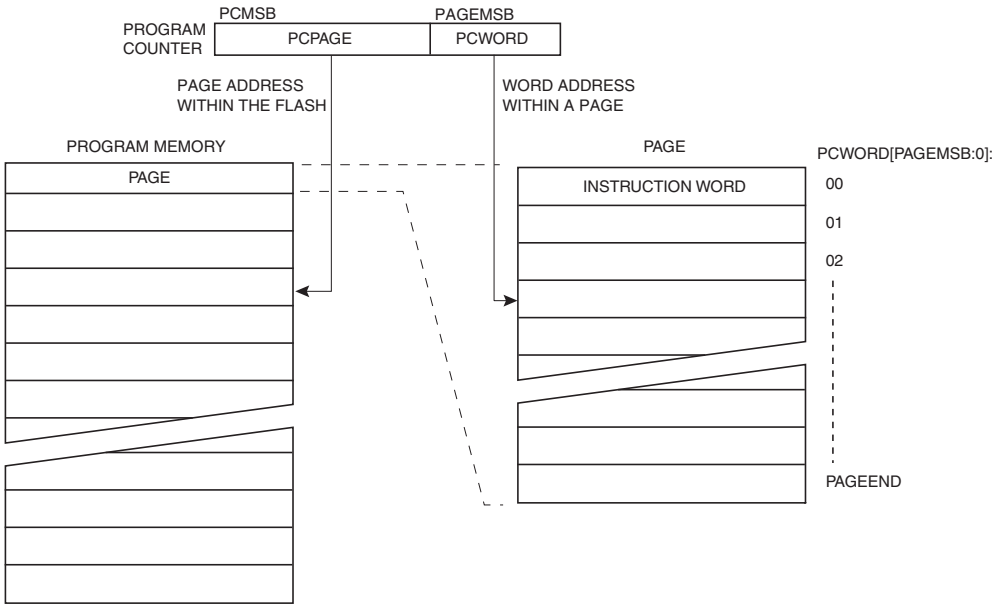
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (See Figure 27-3 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 27-2. Addressing the Flash Which is Organized in Pages[1]



Note: 1. PCPAGE and PCWORD are listed in Table 27-10 on page 298.

28.7 ADC Characteristics

Table 28-7. ADC Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution	Single Ended Conversion		10		Bits
		Differential Conversion		8		Bits
	Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		2	2.5	LSB
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1MHz		4.5		LSB
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz Noise Reduction Mode		2		LSB
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1MHz Noise Reduction Mode		4.5		LSB
	Integral Non-Linearity (INL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		0.5		LSB
	Differential Non-Linearity (DNL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		0.25		LSB
	Gain Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		2		LSB
	Offset Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		2		LSB
	Conversion Time	Free Running Conversion	13		260	µs
	Clock Frequency	Single Ended Conversion	50		1000	kHz
AVCC	Analog Supply Voltage		V_{CC} - 0.3		V_{CC} + 0.3	V
V_{REF}	Reference Voltage	Single Ended Conversion	1.0		AVCC	V
		Differential Conversion	1.0		AVCC - 0.5	V
V_{IN}	Pin Input Voltage	Single Ended Channels	GND		V_{REF}	V
		Differential Channels	GND		AVCC	V
	Input Range	Single Ended Channels	GND		V_{REF}	V
		Differential Channels[1]	-0.85V_{REF}		V_{REF}	V
	Input Bandwidth	Single Ended Channels		38.5		kHz
		Differential Channels		4		kHz

Figure 29-2. Active Supply Current vs. Frequency (1 - 16MHz))

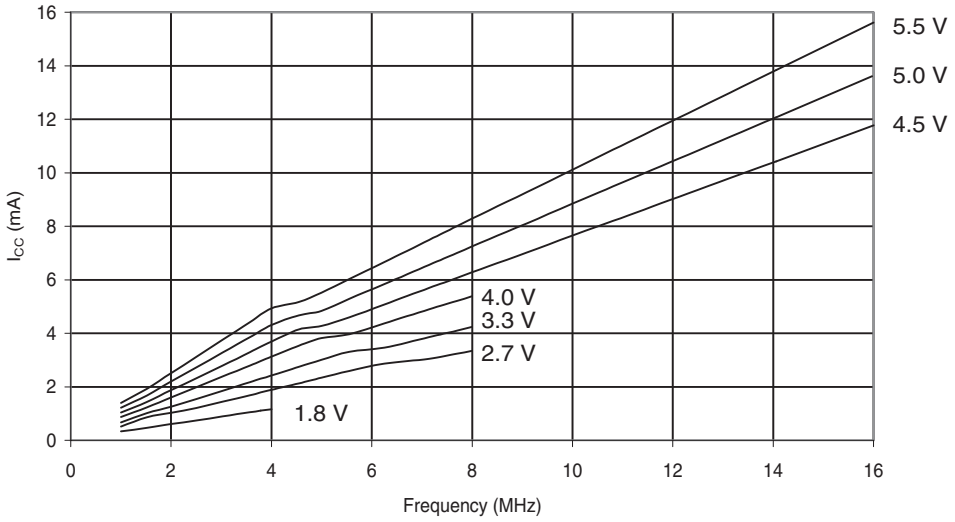
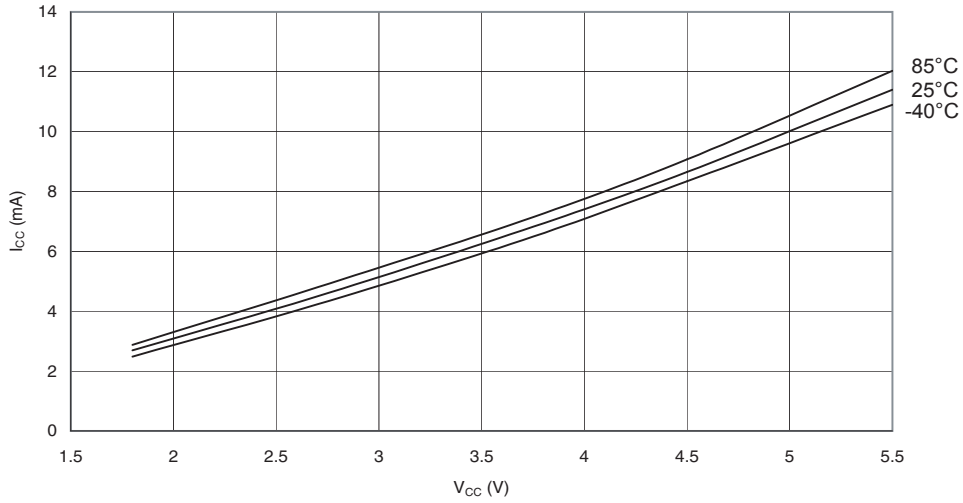


Figure 29-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8MHz)



29.0.9 Pin Thresholds and hysteresis

Figure 29-33. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as “1”)

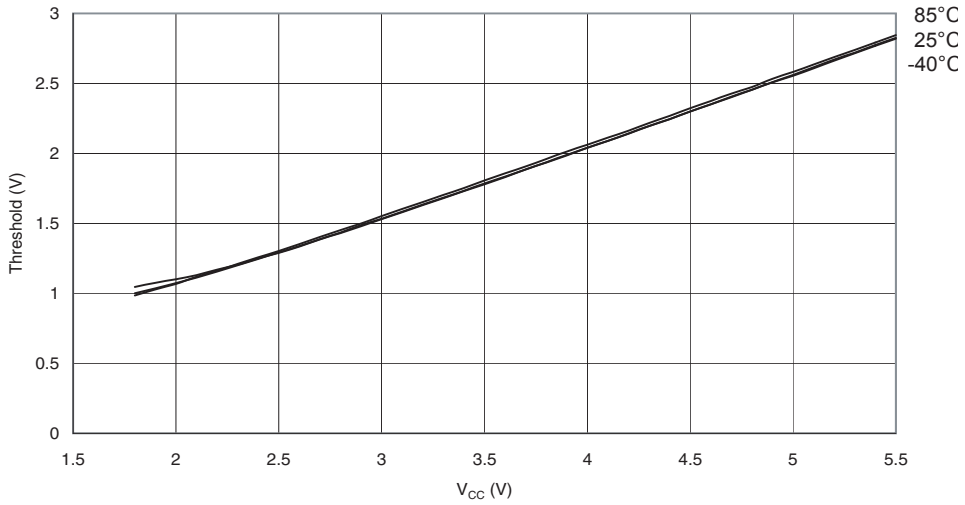
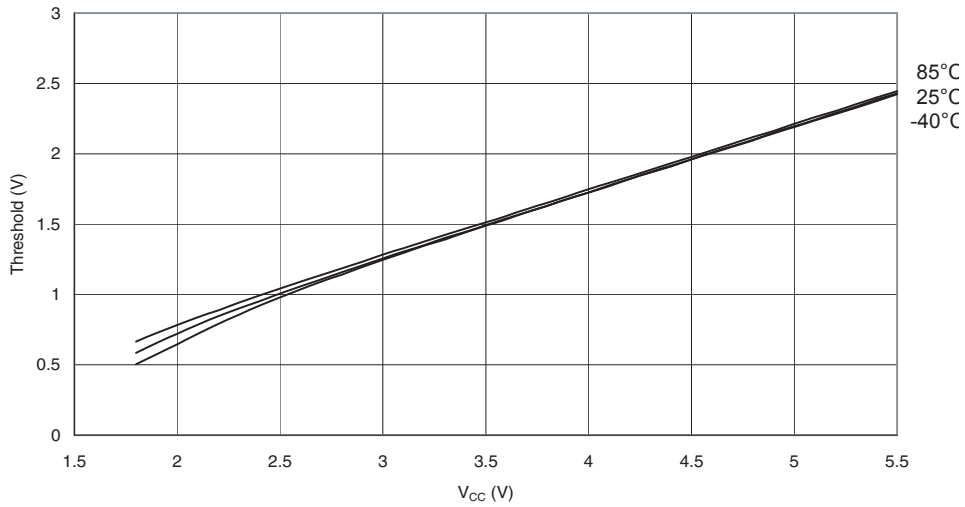


Figure 29-34. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as “0”)



29.0.11 Internal Oscillator Speed

Figure 29-45. Watchdog Oscillator Frequency vs. V_{CC}

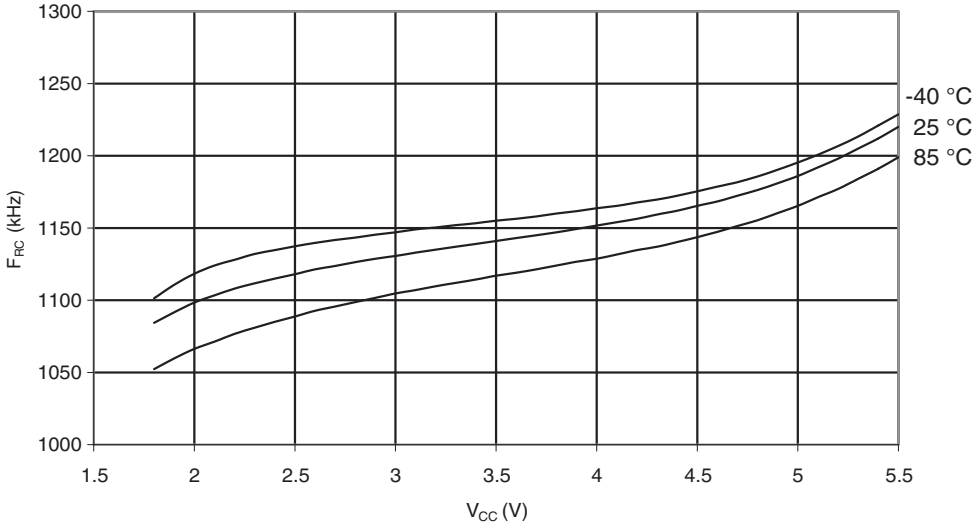


Figure 29-46. Calibrated 8MHz RC Oscillator Frequency vs. Temperature

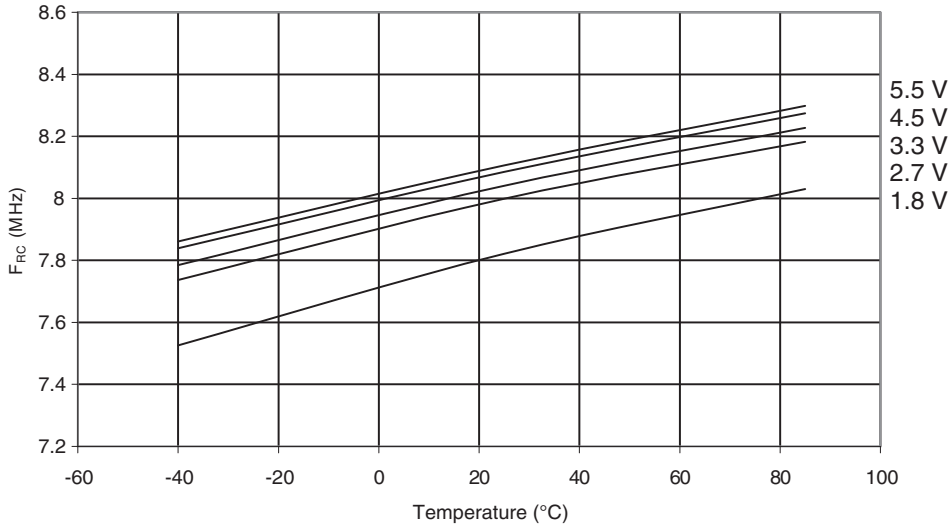
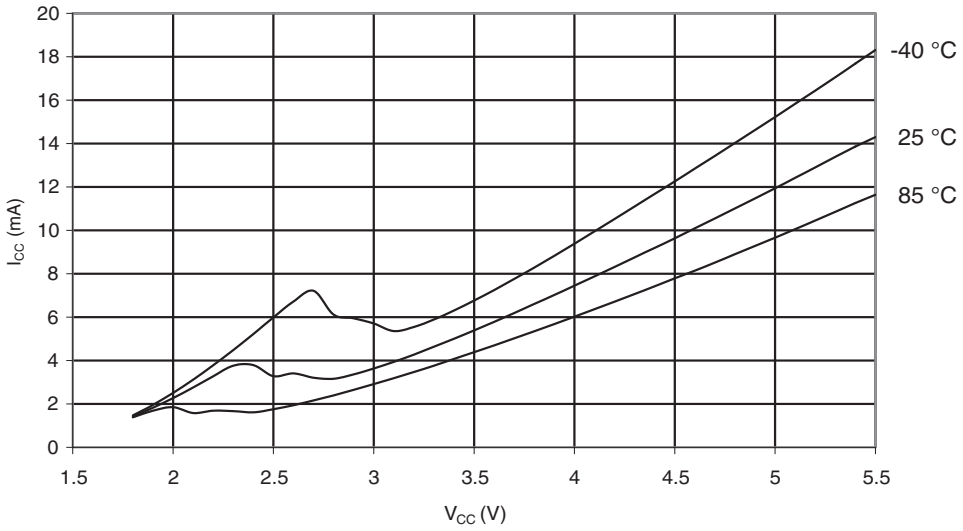
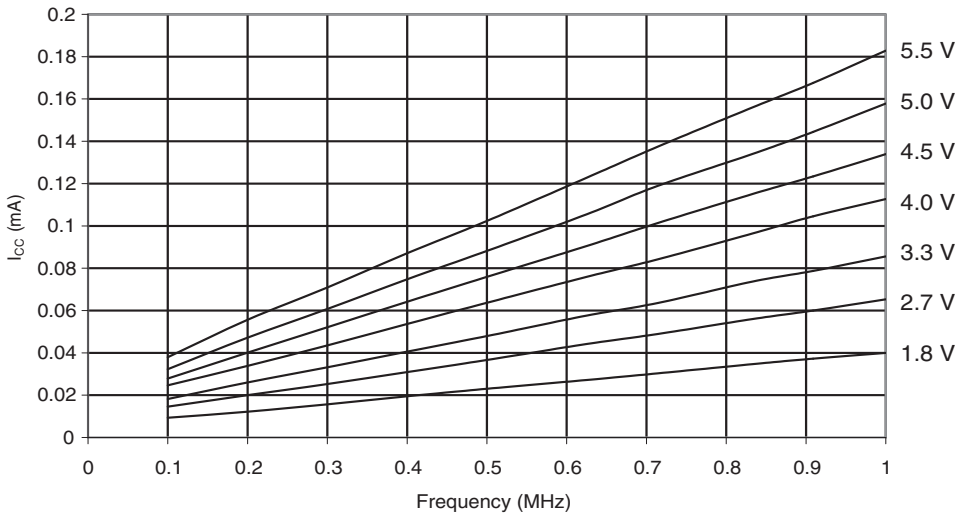


Figure 29-55. Programming Current vs. V_{CC}



29.0.13 Current Consumption in Reset and Reset Pulsewidth

Figure 29-56. Reset Supply Current vs. V_{CC} (0.1 - 1.0MHz, Excluding Current Through The Reset Pull-up)



31. Instruction Set Summary

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