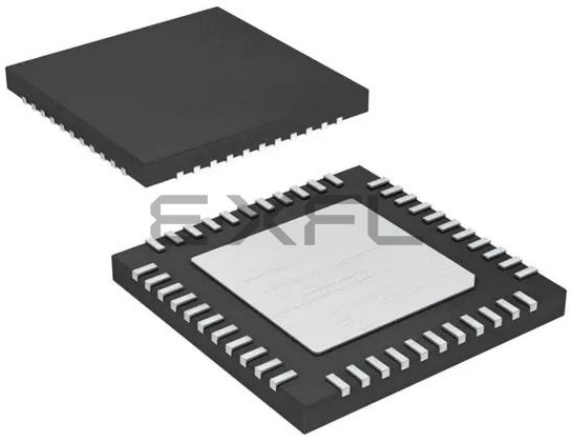


Welcome to E-XFL.COM

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	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K 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	44-VQFN Dual Rows, Exposed Pad
Supplier Device Package	44-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega324p-20mcu

1. Description

The Atmel® ATmega324P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega324P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega324P provides the following features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, two serial programmable USARTs , one byte-oriented 2-wire Serial Interface (I2C), a 8-channel 10-bit ADC with optional differential input stage with programmable gain, a programmable Watchdog Timer with internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run.

Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega324P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega324P is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

32-pin TQFP/ QFN/ MLF Pin #	40-pin PDIP Pin #	PAD	EXTINT	PCINT	ADC/AC	OSC	T/C # 0	T/C # 1	USART	I2C	SPI	JTAG
35	38	PA[2]		PCINT2	ADC2							
36	39	PA[1]		PCINT1	ADC1							
37	40	PA[0]		PCINT0	ADC0							
38	-	VCC								SDA1		
39	-	GND								SCL1		
40	1	PB[0]		PCINT8			T0		XCK0			
41	2	PB[1]		PCINT9		CLKO		T1				
42	3	PB[2]	INT2	PCINT10	AIN0							
43	4	PB[3]		PCINT11	AIN1		OC0A					
44	5	PB[4]		PCINT12			OC0B				$\overline{SS}$	
-	-	GND										
-	-	GND										
-	-	GND										
-	-	GND										
-	-	GND										

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. Even if the synchronous clock is running in Power-save, this clock is only available for Timer/Counter2.

11.8. Standby Mode

When the SM[2:0] bits are written to '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.

11.9. Extended Standby Mode

When the SM[2:0] bits are written to '111' and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-Save mode with the exception that the Oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles.

11.10. 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 corresponding bit in the 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. In all other sleep modes, the clock is already stopped.

Related Links

PRR0 on page 66

11.11. 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.

11.11.1. 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.

Related Links

Analog-to-Digital Converter on page 303

11.12.1. Sleep Mode Control Register

The Sleep Mode Control Register contains control bits for power management.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: SMCR
Offset: 0x53
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x33

Bit	7	6	5	4	3	2	1	0
					SM2	SM1	SM0	SE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – SM2: Sleep Mode Select 2
The SM[2:0] bits select between the five available sleep modes.

Table 11-2. Sleep Mode Select

SM2,SM1,SM0	Sleep Mode
000	Idle
001	ADC Noise Reduction
010	Power-down
011	Power-save
100	Reserved
101	Reserved
110	Standby(1)
111	Extended Standby(1)

Note:
1. Standby mode is only recommended for use with external crystals or resonators.

Bit 2 – SM1: Sleep Mode Select 1
Refer to SM2.

Bit 1 – SM0: Sleep Mode Select 0
Refer to SM2.

Bit 0 – SE: Sleep Enable
The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose,

14. External Interrupts

14.1. EXINT - External Interrupts

The External Interrupts are triggered by the INT pin or any of the PCINT pins. Observe that, if enabled, the interrupts will trigger even if the INT or PCINT pins are configured as outputs. This feature provides a way of generating a software interrupt.

The Pin Change Interrupt Request 3 (PCI3) will trigger if any enabled PCINT[31:24] pin toggles. The Pin Change Interrupt Request 2 (PCI2) will trigger if any enabled PCINT[23:16] pin toggles. The Pin Change Interrupt Request 1 (PCI1) will trigger if any enabled PCINT[15:8] pin toggles. The Pin Change Interrupt Request 0 (PCI0) will trigger if any enabled PCINT[7:0] pin toggles. The PCMSK3, PCMSK2, PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The external 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 external interrupts are enabled and are configured as level triggered, the interrupts will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT requires the presence of an I/O clock. Low level interrupt on INT 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: 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.

Related Links

System Clock and Clock Options on page 42

14.1.1. Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in the following figure.

14.1.2.5. Pin Change Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PCIFR
Offset: 0x3B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1B

Bit	7	6	5	4	3	2	1	0
					PCIF3	PCIF2	PCIF1	PCIF0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – PCIF3: Pin Change Interrupt Flag 3
When a logic change on any PCINT[31:24] pin triggers an interrupt request, PCIF3 will be set. If the I-bit in SREG and the PCIE3 bit in PCICR are set, 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 '1' to it.

Bit 2 – PCIF2: Pin Change Interrupt Flag 2
When a logic change on any PCINT[23:16] pin triggers an interrupt request, PCIF2 will be set. If the I-bit in SREG and the PCIE2 bit in PCICR are set, 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 '1' to it.

Bit 1 – PCIF1: Pin Change Interrupt Flag 1
When a logic change on any PCINT[15:8] pin triggers an interrupt request, PCIF1 will be set. If the I-bit in SREG and the PCIE1 bit in PCICR are set, 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 '1' to it.

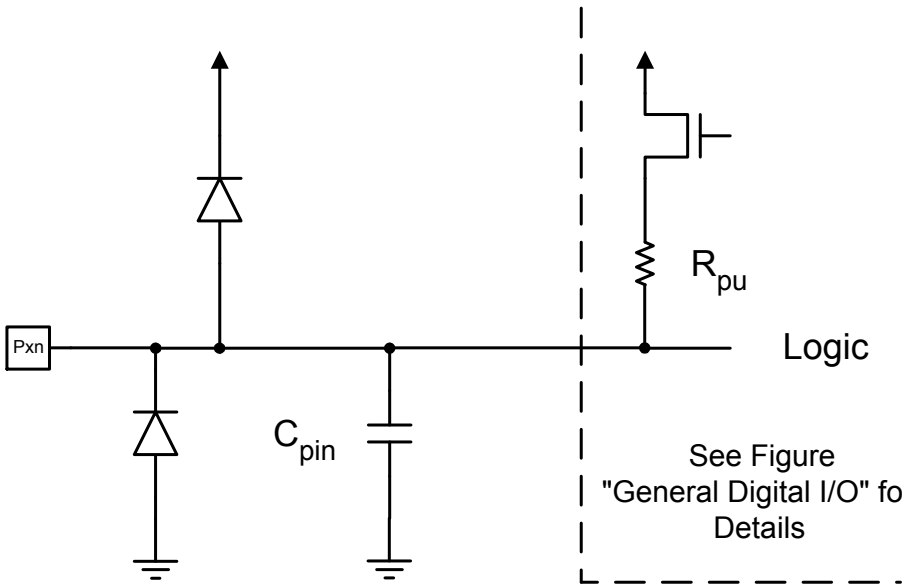
Bit 0 – PCIF0: Pin Change Interrupt Flag 0
When a logic change on any PCINT[7:0] pin triggers an interrupt request, PCIF0 will be set. If the I-bit in SREG and the PCIE0 bit in PCICR are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it.

15. I/O-Ports

15.1. Overview

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in the following figure.

Figure 15-1. I/O Pin Equivalent Schematic



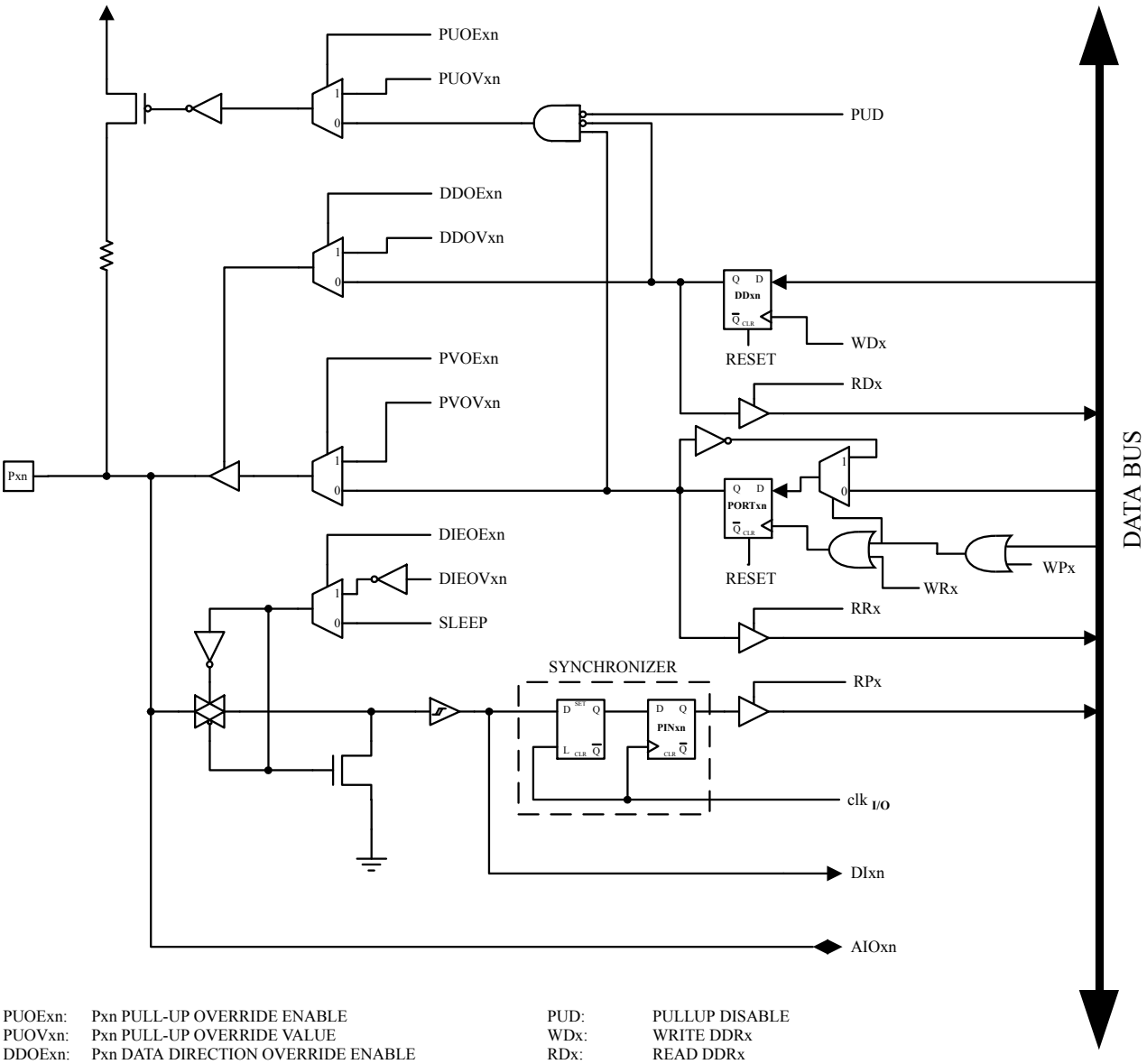
All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing '1' to a bit in the PINx Register will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer to the individual module sections for a full description of the alternate functions.

Enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

Figure 15-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE
SLEEP:	SLEEP CONTROL

PUD:	PULLUP DISABLE
WDx:	WRITE DDRx
RDx:	READ DDRx
RRx:	READ PORTx REGISTER
WRx:	WRITE PORTx
RPx:	READ PORTx PIN
WPx:	WRITE PINx
$clk_{I/O}$:	I/O CLOCK
DIxn:	DIGITAL INPUT PIN n ON PORTx
AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

The following table summarizes the function of the overriding signals. The pin and port indexes from previous figure are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

set. This option is not available for the OC0B pin. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x Register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using Phase Correct PWM can be calculated by:

$$f_{\mathrm{OCnxPCPWM}} = \frac{f_{\mathrm{clk\_I/O}}}{N \cdot 510}$$

N represents the prescaler factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the Phase Correct PWM mode: If the OCR0A register is written equal to BOTTOM, the output will be continuously low. If OCR0A is written to MAX, the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in the timing diagram above, OC0x has a transition from high to low even though there is no Compare Match. This transition serves to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

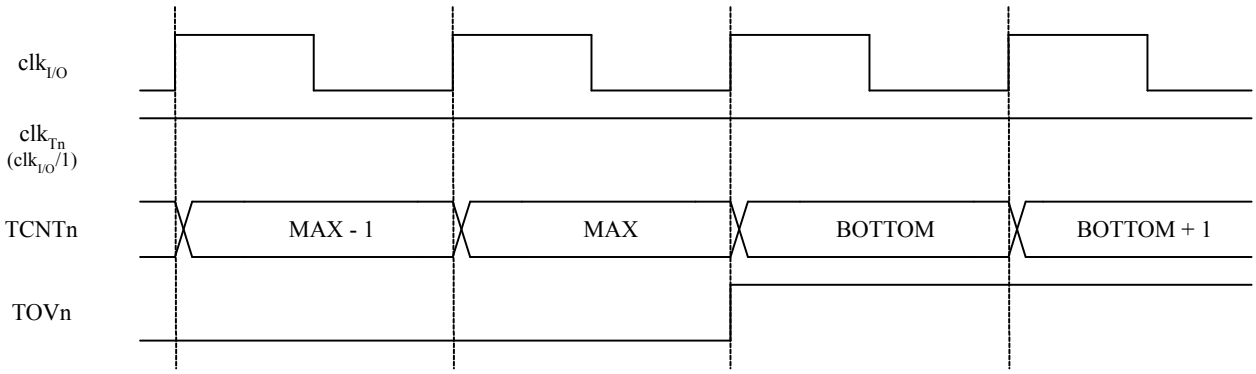
- OCR0x changes its value from MAX, as in the timing diagram. When the OCR0A value is MAX, the OC0 pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OC0x value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts up-counting from a value higher than the one in OCR0x, and for that reason misses the Compare Match and consequently, the OC0x does not undergo the change that would have happened on the way up.

16.8. Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. If the given instance of the TC0 supports an asynchronous mode, $clk_{I/O}$ should be replaced by the TC oscillator clock.

The figures include information on when interrupt flags are set. The first figure below illustrates timing data for basic Timer/Counter operation close to the MAX value in all modes other than Phase Correct PWM mode.

Figure 16-8. Timer/Counter Timing Diagram, no Prescaling



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The next figure shows the same timing data, but with the prescaler enabled.

17.14.4. TC1 Counter Value Low and High byte

The TCNT1L and TCNT1H register pair represents the 16-bit value, TCNT1.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: TCNT1L and TCNT1H
Offset: 0x84
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	TCNT1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCNT1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – TCNT1[15:0]: Timer/Counter 1 Counter Value

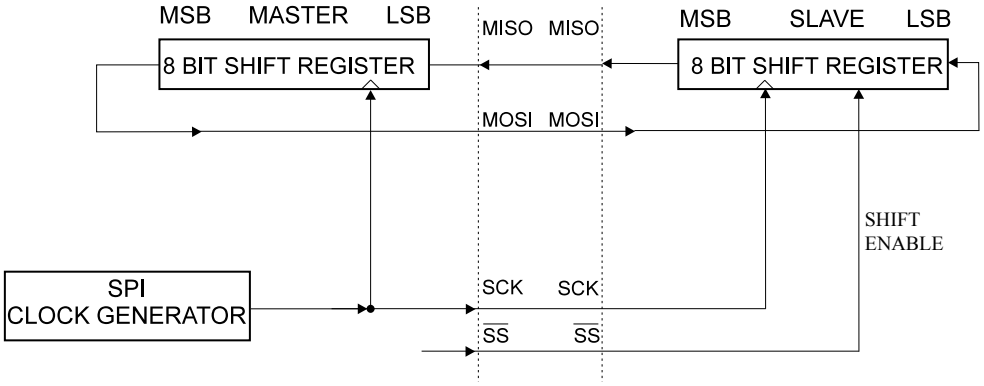
The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Registers for details.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 20-2. SPI Master-slave Interconnection



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

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be longer than two CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to the table below. For more details on automatic port overrides, refer to the IO Port description.

Table 20-1. SPI Pin Overrides

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

Note: 1. See the IO Port description for how to define the SPI pin directions.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. `DDR_SPI` in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. `DD_MOSI`, `DD_MISO` and `DD_SCK` must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB5, replace `DD_MOSI` with `DDB5` and `DDR_SPI` with `DDRB`.

Assembly Code Example

```
SPI_MasterInit:
   ; Set MOSI and SCK output, all others input
   ldi     r17,(1<<DD_MOSI)|(1<<DD_SCK)
   out     DDR_SPI,r17
   ; Enable SPI, Master, set clock rate fck/16
   ldi     r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
   out     SPCR,r17
   ret
```

BAUD	Baud rate (in bits per second, bps)
f_{OSC}	System oscillator clock frequency
UBRRn	Contents of the UBRRnH and UBRRnL Registers, (0-4095). Some examples of UBRRn values for some system clock frequencies are found in Examples of Baud Rate Settings.

21.4.2. Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. However, in this case, the Receiver will only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used.

For the Transmitter, there are no downsides.

21.4.3. External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to the Clock Generation Logic block diagram in the previous section.

External clock input from the XCKn pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and Receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCKn clock frequency is limited by the following equation:

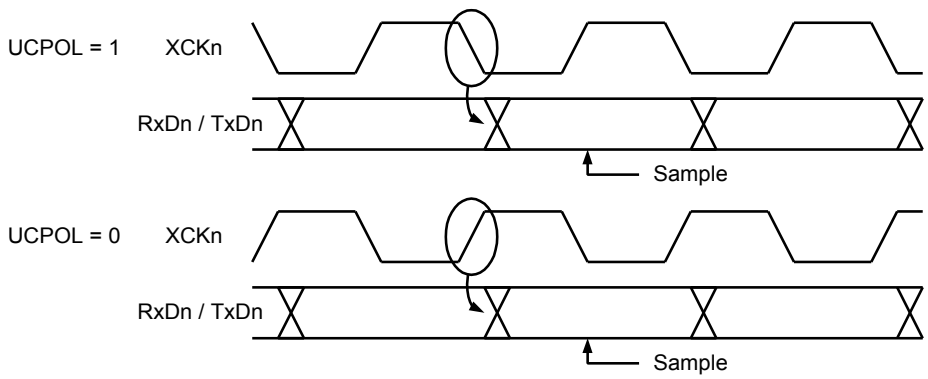
$$f_{XCKn} < \frac{f_{OSC}}{4}$$

The value of f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

21.4.4. Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCKn pin will be used as either clock input (Slave) or clock output (Master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxDn) is sampled at the opposite XCKn clock edge of the edge the data output (TxDn) is changed.

Figure 21-3. Synchronous Mode XCKn Timing



The UCPOL bit UCRSC selects which XCKn clock edge is used for data sampling and which is used for data change. As the above timing diagram shows, when UCPOL is zero, the data will be changed at

ADPS[2:0]	Division Factor
010	4
011	8
100	16
101	32
110	64
111	128

implemented, but all outputs with tri-state capability can be set in high-impedant state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this data sheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which data register is selected as path between TDI and TDO for each instruction.

26.12.1. EXTEST; 0x0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

26.12.2. IDCODE; 0x1

Optional JTAG instruction selecting the 32-bit ID Register as Data Register. The ID Register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE Register is sampled into the Boundary-scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

26.12.3. SAMPLE_PRELOAD; 0x2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan Chain is applied to the output latches. However, the output latches are not connected to the pins.

26.12.4. AVR_RESET; 0xC

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG Reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the Reset will be active as long as there is a logic 'one' in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

26.12.5. BYPASS; 0xF

Mandatory JTAG instruction selecting the Bypass Register for Data Register.

28.8.3. Chip Erase

The Chip Erase will erase the Flash, the SRAM and the EEPROM memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

Note: The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”:

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

28.8.4. Programming the Flash

The Flash is organized in pages as number of Words in a Page and number of Pages in the Flash. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

Step A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

Step B. Load Address Low Byte

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

Step C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

Step D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

Step E. Latch Data

1. Set BS1 to “1”. This selects high data byte.

4. Load address low byte using programming instruction 2c.
5. Load data using programming instructions 2d, 2e and 2f.
6. Repeat steps 4 and 5 for all instruction words in the page.
7. Write the page using programming instruction 2g.
8. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
9. Repeat steps 3 to 7 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load the page address using programming instructions 2b and 2c. PCWORD (refer to Command Byte Bit Coding table in Signal Names section) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGELOAD.
5. Load the entire page by shifting in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page.
6. Enter JTAG instruction PROG_COMMANDS.
7. Write the page using programming instruction 2g.
8. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
9. Repeat steps 3 to 8 until all data have been programmed.

28.10.18. Reading the Flash

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load address using programming instructions 3b and 3c.
4. Read data using programming instruction 3d.
5. Repeat steps 3 and 4 until all data have been read.

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load the page address using programming instructions 3b and 3c. PCWORD (refer to table *Command Byte Bit Coding* in section *Parallel Programming Parameters, Pin Mapping, and Commands*) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGEREAD.
5. Read the entire page by shifting out all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. Remember that the first 8 bits shifted out should be ignored.
6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

28.10.19. Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed. See Performing Chip Erase.

Figure 30-2. Active Supply Current vs. Frequency (1 - 20MHz)

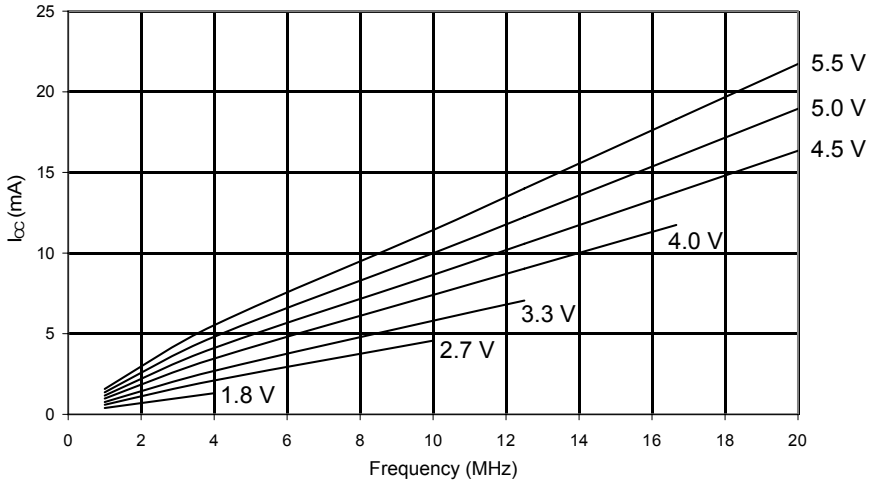


Figure 30-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

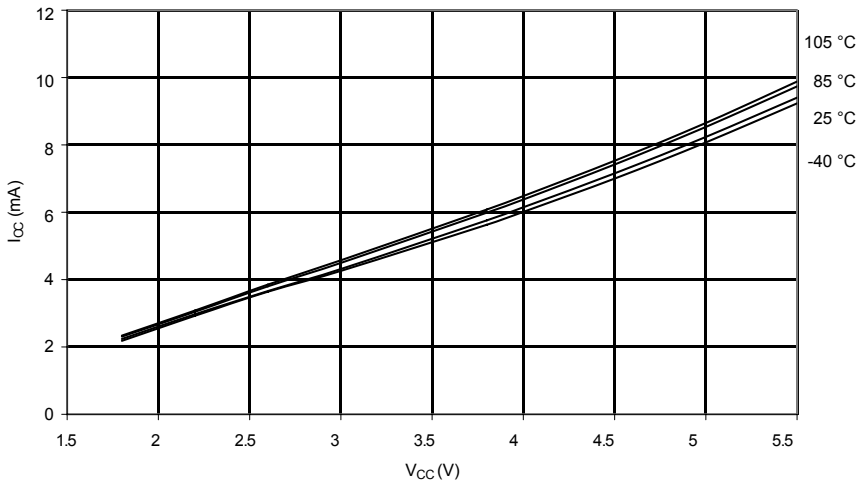


Figure 30-4. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

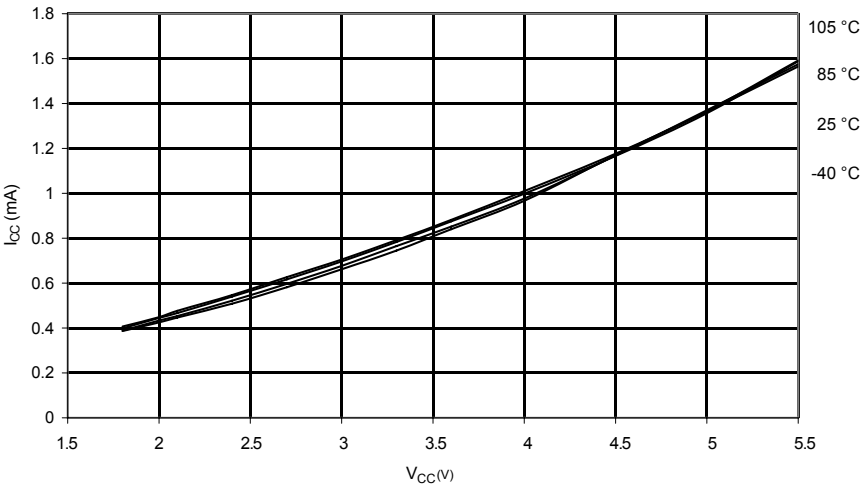
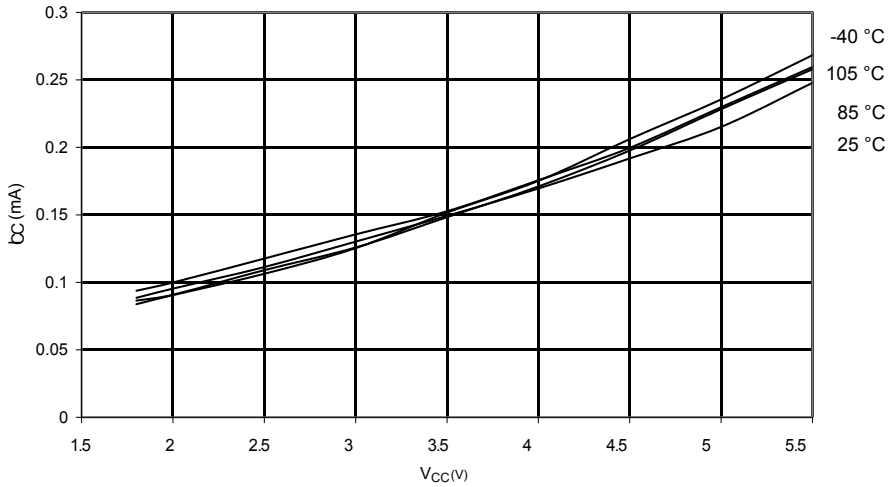
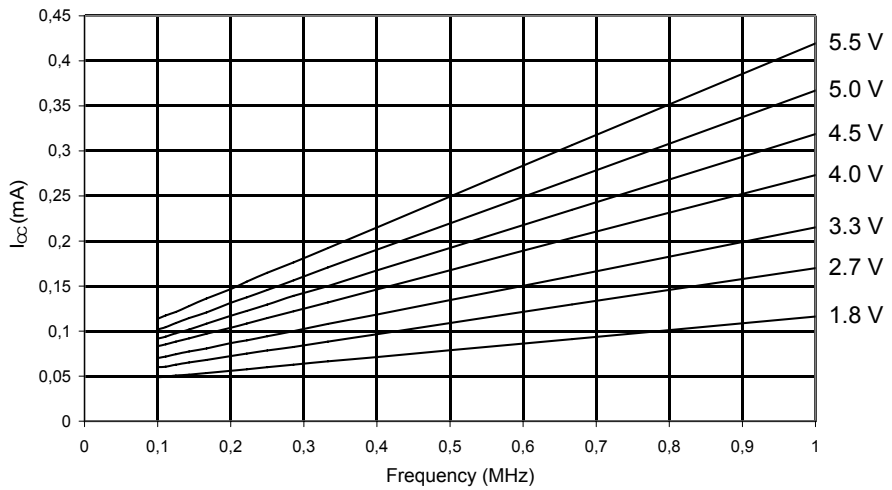


Figure 30-5. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 128 kHz)



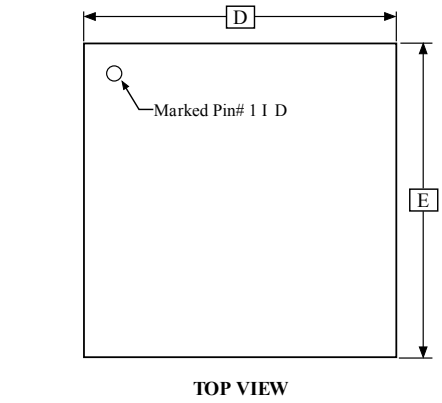
30.2. Idle Supply Current

Figure 30-6. Idle Supply Current vs. Low Frequency (0.1 - 1.0MHz)

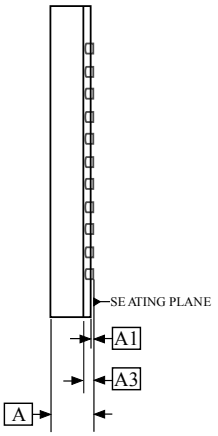


Offset	Name	Bit Pos.								
0x52	Reserved									
0x53	SMCR	7:0					SM2	SM1	SM0	SE
0x54	MCUSR	7:0				JTRF	WDRF	BORF	EXTRF	PORF
0x55	MCUCR	7:0	JTD	BODS	BODSE	PUD			IVSEL	IVCE
0x56	Reserved									
0x57	SPMCSR	7:0	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
0x58 ... 0x5C	Reserved									
0x5D	SPL and SPH	7:0	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x5E		15:8					SP11	SP10	SP9	SP8
0x5F	SREG	7:0	I	T	H	S	V	N	Z	C
0x60	WDTCSR	7:0	WDIF	WDIE	WDP[3]	WDCE	WDE	WDP[2:0]		
0x61	CLKPR	7:0	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
0x62 ... 0x63	Reserved									
0x64	PRR0	7:0	PRTWI	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI0	PRUSART0	PRADC
0x65	Reserved									
0x66	OSCCAL	7:0	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
0x67	Reserved									
0x68	PCICR	7:0					PCIE3	PCIE2	PCIE1	PCIE0
0x69	EICRA	7:0			ISC21	ISC20	ISC11	ISC10	ISC01	ISC00
0x6A	Reserved									
0x6B	PCMSK0	7:0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
0x6C	PCMSK1	7:0	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6D	PCMSK2	7:0	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x6E	TIMSK0	7:0						OCIEB	OCIEA	TOIE
0x6F	TIMSK1	7:0			ICIE			OCIEB	OCIEA	TOIE
0x70	TIMSK2	7:0						OCIEB	OCIEA	TOIE
0x71 ... 0x72	Reserved									
0x73	PCMSK3	7:0	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
0x74 ... 0x77	Reserved									
0x78	ADCL and ADCH	7:0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
0x79		15:8							ADC9	ADC8
0x7A	ADCSRA	7:0	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
0x7B	ADCSRB	7:0		ACME				ADTS2	ADTS1	ADTS0
0x7C	ADMUX	7:0	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
0x7D	Reserved									
0x7E	DIDR0	7:0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
0x7F	DIDR1	7:0	Reserved5	Reserved4	Reserved3	Reserved2	Reserved1	Reserved0	AIN1D	AIN0D
0x80	TCCR1A	7:0	COM1	COM1	COM1	COM1			WGM11	WGM10
0x81	TCCR1B	7:0	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10

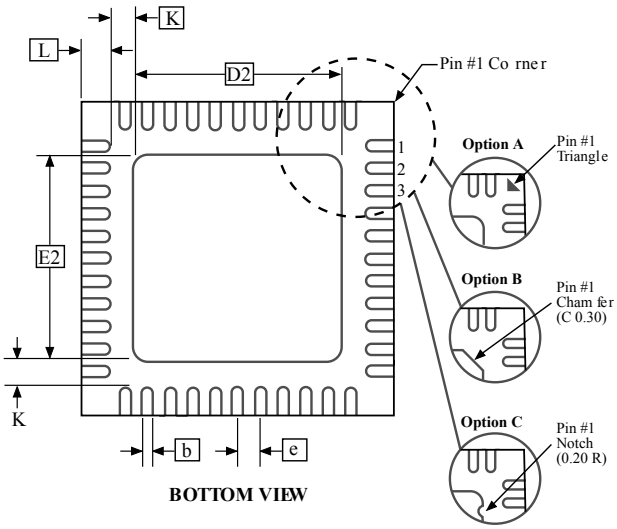
33.3. 44-pin VQFN



TOP VIEW



SIDE VIEW



BOTTOM VIEW

Note : JEDEC Standard MO-220, Fig . 1 (S AW Singulation) VKKD-3 .

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	6.90	7.00	7.10	
D2	5.00	5.20	5.40	
E	6.90	7.00	7.10	
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.59	0.64	0.69	
K	0.20	0.26	0.41	

9/26/08

Package Drawing Contact: avr@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	44M1, 44-pad, 7 x 7 x 1.0mm body, lead pitch 0.50mm, 5.20mm exposed pad, thermally enhanced plastic very thin quad flat no lead package (VQFN)	ZWS	44M1	H