Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	12
Program Memory Size	2KB (1K x 16)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	14-SOIC (0.154", 3.90mm Width)
Supplier Device Package	14-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny24-15ssz

The fast-access register file contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle arithmetic logic unit (ALU) operation. In a typical ALU operation, two operands are output from the register file, the operation is executed, and the result is stored back in the register file, all in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for data space addressing, enabling efficient address calculations. One of the address pointers can also be used as an address pointer for look up tables in flash program memory. These added function registers are the 16-bit X-, Y-, and Z-registers, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single-register operations can also be executed in the ALU. After an arithmetic operation, the status register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions able to directly address the whole address space. Most AVR® instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the stack pointer (SP) in the reset routine (before subroutines or interrupts are executed). The SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the Atmel® AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions such as control registers, SPI, and other I/O functions. The I/O memory can be accessed directly or as the data space locations following those of the register file, 0x20 - 0x5F.

5.3 ALU – Arithmetic Logic Unit

The high-performance Atmel AVR ALU operates in direct connection with all 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories - arithmetic, logical, and bit functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

5.4 Status Register

The status register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the status register is updated after all ALU operations, as specified in the instruction set summary. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The status register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

6.5.4 EECR – EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

- **Bit 7 – Res: Reserved Bit**

This bit is reserved for future use, and will always read as 0 in Atmel® ATtiny24/44/84. For compatibility with future AVR® devices, always write this bit to a logical zero. After reading, mask out this bit.

- **Bit 6 – Res: Reserved Bit**

This bit is reserved in the Atmel ATtiny24/44/84 and will always read as zero.

- **Bits 5, 4 – EEPM1 and EEPM0: EEPROM Mode Bits**

The EEPROM programming mode bits define which programming action will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or split the erase and write operations into two separate operations. The programming times for the different modes are shown in Table 6-1. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 6-1. EEPROM Mode Bits

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4ms	Erase and write in one operation (atomic operation)
0	1	1.8ms	Erase only
1	0	1.8ms	Write only
1	1	–	Reserved for future use

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to logical one enables the EEPROM ready interrupt if the I-bit in SREG is set. Writing EERIE to logical zero disables the interrupt. The EEPROM ready interrupt generates a constant interrupt when non-volatile memory is ready for programming.

- **Bit 2 – EEMPE: EEPROM Master Program Enable**

The EEMPE bit determines whether writing EEPE to logical one will have effect or not. When EEMPE is set, setting EEPE within four clock cycles will program the EEPROM at the selected address. If EEMPE is logical zero, setting EEPE will have no effect. When EEMPE has been written to logical one by software, hardware clears the bit to logical zero after four clock cycles.

- **Bit 1 – EEPE: EEPROM Program Enable**

The EEPROM program enable bit, EEPE, is the programming enable signal to the EEPROM. When EEPE is written, the EEPROM will be programmed according to the EEPMn bits setting. The EEMPE bit must be written to logical one before a logical one is written to EEPE, otherwise no EEPROM write will take place. When the write access time has elapsed, the EEPE bit is cleared by hardware. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM read enable signal, EERE, is the read strobe for the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to logical one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data are available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is not possible to read the EEPROM or change the EEAR register.

7.9 System Clock Prescaler

The Atmel® ATtiny24/44/84 system clock can be divided by setting the clock prescaler register – CLKPR. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 7-10 on page 30.

7.9.1 Switching Time

When switching between prescaler settings, the system clock prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than either the clock frequency corresponding to the previous setting or the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler, even if it were readable, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2 × T2 before the new clock frequency is active. In this interval, twp active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

7.10 Register Description

7.10.1 Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0	
0x31 (0x51)	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								

The oscillator calibration register is used to trim the calibrated internal RC oscillator to remove process variations from the oscillator frequency. A pre-programmed calibration value is automatically written to this register during chip reset, giving the factory calibrated frequency as specified in Table 22-2 on page 157. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 22-2 on page 157. calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and flash write accesses, and these write times will be affected accordingly. If the EEPROM or flash are written, do not calibrate to more than 8.8MHz. Otherwise, the EEPROM or flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to logical zero gives the lowest frequency range, setting this bit to logical one gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80.

The CAL6..0 bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range.

7.10.2 Clock Prescaler Register – CLKPR

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	**CLKPCE**	–	–	–	**CLKPS3**	**CLKPS2**	**CLKPS1**	**CLKPS0**	**CLKPR**
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

The CLKPCE bit must be written to logical one to enable change of the CLKPS bits. The CLK- PCE bit is only updated when the other bits in CLKPR are simultaneously written to logical zero. CLKPCE is cleared by hardware four cycles after it is written or when the CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period nor clear the CLKPCE bit.+

8. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The Atmel® AVR® provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

8.1 Sleep Modes

Figure 7-1 on page 24 presents the different clock systems in the Atmel ATtiny24/44/84, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 8-1 shows the different sleep modes and their wake up sources

Table 8-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains				Oscillators	Wake-up Sources				
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	Main Clock Source Enabled	INT0 and Pin Change	SPM/ EEPROM Ready	ADC	Other I/O	Watchdog Interrupt
Idle			X	X	X	X	X	X	X	X
ADC noise Reduction				X	X	X(1)	X	X		X
Power-down						X(1)				X
Stand-by(2)					X	X	X(1)			

Notes: 1. For INT0, only level interrupt.
2. Only recommended with external crystal or resonator selected as clock source

To enter any of the three sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM1..0 bits in the MCUCR register select which sleep mode (idle, ADC noise reduction, standby or power-down) will be activated by the SLEEP instruction. See Table 8-2 on page 34 for a summary.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register File and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the reset vector.

8.2 Idle Mode

When the SM1..0 bits are written to "00", the SLEEP instruction makes the MCU enter idle mode, stopping the CPU but allowing the analog comparator, ADC, Timer/Counter, watchdog, and interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the timer overflow. If wake-up from the analog comparator interrupt is not required, the analog comparator can be powered down by setting the ACD bit in the analog comparator control and status register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

8.3 ADC Noise Reduction Mode

When the SM1..0 bits are written to 01, the SLEEP instruction makes the MCU enter ADC noise reduction mode, stopping the CPU but allowing the ADC, the external interrupts, and the Watchdog to continue operating (if enabled). This sleep mode halts $clk_{I/O}$, clk_{CPU}, and
clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC conversion complete interrupt, only an external reset, a watchdog reset, a brown-out reset, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or a pin change interrupt can wake up the MCU from ADC noise reduction mode.

12.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 12-2 on page 50, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 12.3 “Alternate Port Functions” on page 54.

If a logic high level (logical one) is present on an asynchronous external interrupt pin configured as “interrupt on rising edge, falling edge, or any logic change on pin” while the external interrupt is not enabled, the corresponding external interrupt flag will be set when resuming from the above mentioned sleep mode, as the clamping in these sleep mode produces the requested logic change.

12.2.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (reset, active mode and idle mode).

The simplest method to ensure a defined level of an unused pin is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to VCC or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

14. 16-bit Timer/Counter1

14.1 Features

- True 16-bit design (i.e., Allows 16-bit PWM)
- Two independent output compare units
- Double buffered output compare registers
- One input capture unit
- Input capture noise canceler
- Clear timer on compare match (auto reload)
- Glitch-free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- External event counter
- Four independent interrupt sources (TOV1, OCF1A, OCF1B, and ICF1)

14.2 Overview

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement.

Most register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the output compare unit channel. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 14-1 on page 79. For the actual placement of I/O pins, refer to Section 1-1 “Pinout Atmel ATtiny24/44/84” 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 Section 14.11 “Register Description” on page 97.

The input capture register can capture the Timer/Counter value at a given external (edge-triggered) event on either the input capture pin (ICP1) or on the analog comparator pins (see Section 17. “Analog Comparator” on page 115). The input capture unit includes a digital filtering unit (noise canceller) for reducing the chance of capturing noise spikes.

The top value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A register, the ICR1 register, or by a set of fixed values. When using OCR1A as top value in a PWM mode, the OCR1A register cannot be used for generating a PWM output. However, the top value will in this case be double buffered, allowing the top value to be changed at run time. If a fixed top value is required, the ICR1 register can be used as an alternative, freeing the OCR1A to be used as PWM output.

14.2.2 Definitions

The following definitions are used extensively throughout the section:

Table 14-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The top value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 register. The assignment is dependent on the mode of operation.

14.2.3 Compatibility

The 16-bit Timer/Counter has been updated and improved from previous versions of the 16-bit Atmel® AVR® Timer/Counter. This 16-bit Timer/Counter is fully compatible with the earlier version regarding:

- All 16-bit Timer/Counter related I/O register address locations, including timer interrupt registers.
- Bit locations inside all 16-bit Timer/Counter registers, including timer interrupt registers.
- Interrupt vectors.

The following control bits have changed name, but have same functionality and register location:

- PWM10 is changed to WGM10.
- PWM11 is changed to WGM11.
- CTC1 is changed to WGM12.

The following bits are added to the 16-bit Timer/Counter control registers:

- 1A and 1B are added to TCCR1A.
- WGM13 is added to TCCR1B.

The 16-bit Timer/Counter has improvements that will affect the compatibility in some special cases.

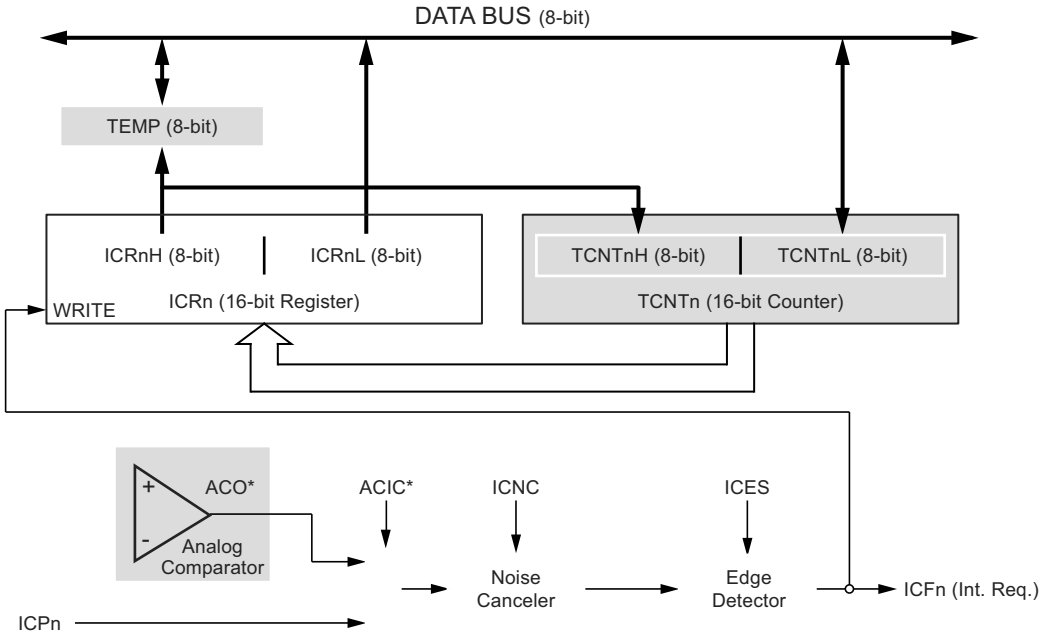
14.3 Accessing 16-bit Registers

TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the Atmel AVR CPU via the 8-bit data bus. The 16-bit registers must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storage of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses use the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

Figure 14-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the input capture pin (ICP1), or alternatively on the analog comparator output (ACO), and this change conforms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the input capture register (ICR1). The input capture flag (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 register. If enabled (ICIE1 = 1), the input capture flag generates an input capture interrupt. The ICF1 flag is automatically cleared when the interrupt is executed. Alternatively, the ICF1 flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the input capture register (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read, the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location, it will access the TEMP register.

The ICR1 register can only be written when using a waveform generation mode that utilizes the ICR1 register for defining the counter's top value. In these cases the waveform generation mode (WGM13:0) bits must be set before the top value can be written to the ICR1 register. When writing the ICR1 register, the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to Section 14.3 “Accessing 16-bit Registers” on page 80.

14.6.1 Input Capture Trigger Source

The main trigger source for the input capture unit is the input capture pin (ICP1). Timer/Counter 1 can alternatively use the analog comparator output as trigger source for the input capture unit. The analog comparator is selected as trigger source by setting the analog comparator input capture (ACIC) bit in the analog comparator control and status register (ACSR). Be aware that changing the trigger source can trigger a capture. The input capture flag must, therefore, be cleared after the change.

Both the input capture pin (ICP1) and the analog comparator output (ACO) are sampled using the same technique as for the T1 pin (Figure 15-1 on page 103). The edge detector is also identical. However, when the noise canceller is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the inputs of the noise canceller and edge detector are always enabled unless the Timer/Counter is set in a waveform generation mode that uses ICR1 to define top.

An input capture can be triggered by software by controlling the port of the ICP1 pin.

14.8.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the waveform generator that no action on the OC1x register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 14-2 on page 97. For fast PWM mode refer to Table 14-3 on page 97, and for phase correct and phase and frequency correct PWM refer to Table 14-4 on page 98.

A change of the COM1x1:0 bit states will have an effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the 1x strobe bits.

14.9 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM13:0) and compare output mode (COM1x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes, the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (Section 14.8 “Compare Match Output Unit” on page 88). For detailed timing information refer to Section 14.10 “Timer/Counter Timing Diagrams” on page 95.

14.9.1 Normal Mode

The simplest mode of operation is the normal mode (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (max = 0xFFFF), and then restarts from the bottom (0x0000). In normal operation, the Timer/Counter overflow flag (TOV1) will be set on the same timer clock cycle on which the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, when combined with the timer overflow interrupt that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode. A new counter value can be written anytime.

The input capture unit is easy to use in normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events is too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

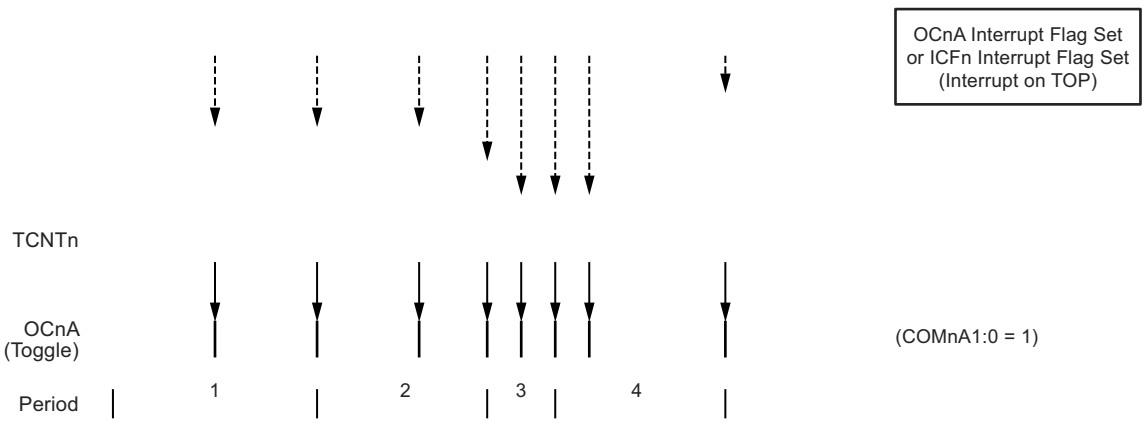
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended because this will occupy too much CPU time.

14.9.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare, or CTC, mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 register is used to manipulate the counter resolution. In CTC mode, the counter is cleared to zero when the counter value (TCNT1) matches either OCR1A (WGM13:0 = 4) or ICR1 (WGM13:0 = 12). OCR1A or ICR1 define the top value for the counter, and hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 14-6. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 14-6. CTC Mode, Timing Diagram



14.11 Register Description

14.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM1A1:0: Compare Output Mode for Channel A**

- **Bit 5:4 – COM1B1:0: Compare Output Mode for Channel B**

The COM1A1:0 and COM1B1:0 control the output compare pins' (OC1A and OC1B, respectively) behavior. If one or both of the COM1A1:0 bits are written to logical one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to logical one, the OC1B 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 OC1A or OC1B pin must be set in order to enable the output driver.

When OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent on the WGM13:0 bit settings. Table 14-2 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or CTC mode (non-PWM).

Table 14-2. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match.
1	0	Clear OC1A/OC1B on compare match (Set output to low level).
1	1	Set OC1A/OC1B on compare match (set output to high level).

Table 14-3 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 14-3. Compare Output Mode, Fast PWM(1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected. WGM13=1: Toggle OC1A on compare match, OC1B reserved.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See Section 14.9.3 “Fast PWM Mode” on page 90 for more details.

15. Timer/Counter Prescaler

Timer/Counter 0, and 1 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to all Timer/Counters. Tn is used as a general name, where n = 0, 1.

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

15.1 Prescaler Reset

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

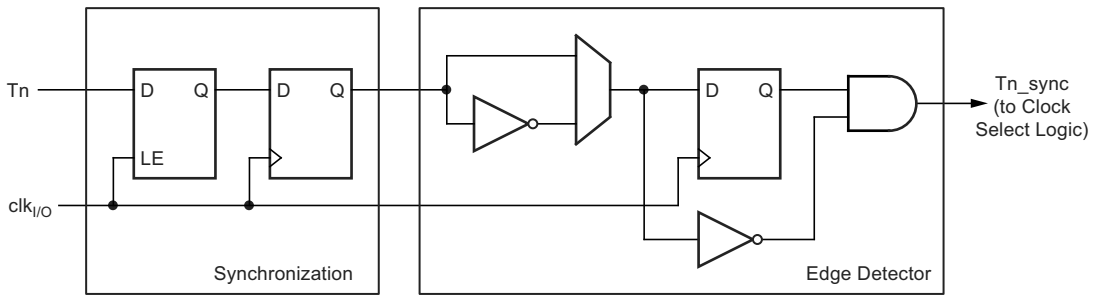
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution.

15.2 External Clock Source

An external clock source applied to the Tn pin can be used as Timer/Counter clock (clk_{Tn}). The Tn pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 15-1 shows a functional equivalent block diagram of the Tn synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

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

Figure 15-1. T0 Pin Sampling



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

Enabling and disabling of the clock input must be done when Tn has been stable for at least one system clock cycle, otherwise there is a risk that a false Timer/Counter clock pulse could be generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50 duty cycle. Because the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by oscillator source (crystal, resonator, and capacitor) tolerances, it is recommended that the maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

17.2 Register Description

17.2.1 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	BIN	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R	R/w	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logical one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the analog comparator. When this bit is written logical zero, AIN1 is applied to the negative input of the analog comparator. For a detailed description of this bit, see Section 17.1 “Analog Comparator Multiplexed Input” on page 115.

17.2.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	**ACD**	**ACBG**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

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

When this bit is written logical 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 ana- log comparator. When this bit is cleared, AIN0 is applied to the positive input of the analog comparator.

- **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 one to two 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 logical one to the flag.

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

When the ACIE bit is written logical one and the I-bit in the status register is set, the analog comparator interrupt is activated. When written logical zero, the interrupt is disabled.

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

When written logical one, this bit enables the input capture function in Timer/Counter 1 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 canceller and edge select features of the Timer/Counter 1 input capture interrupt. When written logical zero, no connection between the analog comparator and the input capture function exists. To make the comparator trigger the Timer/Counter 1 input capture interrupt, the ICIE1 bit in the timer interrupt mask register (TIMSK1) must be set.

19. debugWIRE On-chip Debug System

19.1 Features

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

19.2 Overview

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

19.3 Physical Interface

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

Figure 19-1. The debugWIRE Setup

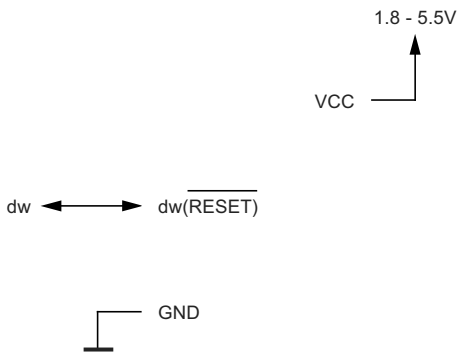


Figure 19-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

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

- Pull-up resistor on the dW/(RESET) line must be in the range of 10k to 20kΩ. However, the pull-up resistor is optional.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors inserted on the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

19.4 Software Break Points

debugWIRE supports program memory break points by the Atmel® AVR® BREAK instruction. Setting a break point in AVR Studio® will insert a BREAK instruction in the program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The flash must be re-programmed each time a break point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of break points will there- fore reduce the flash data retention. Devices used for debugging purposes should not be shipped to end customers.

19.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as the external reset (RESET). An external reset source is, therefore, not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O registers via the debugger (AVR Studio). See the debugWIRE documentation for a detailed description of the limitations.

A programmed DWEN fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN fuse should be disabled when debugWire is not used.

19.6 Register Description

The following section describes the registers used with the debugWire system.

19.6.1 DWDR – debugWire Data Register

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	DWDR[7:0]								DWDR
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 debugWire data register (DWDR) provides a communication channel from the program running in the MCU to the debugger. This register is only accessible by the debugWIRE system, and can, therefore, not be used as a general purpose register in normal operations.

Table 21-12. Pin Name Mapping

Signal Name in High-voltage Serial Programming Mode	Pin Name	I/O	Function
SDI	PA6	I	Serial data input
SII	PA5	I	Serial instruction input
SDO	PA4	O	Serial data output
SCI	PB0	I	Serial clock input (min. 220ns period)

The minimum period for the serial clock input (SCI) during high-voltage serial programming is 220ns.

Table 21-13. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PA0	Prog_enable[0]	0
PA1	Prog_enable[1]	0
PA2	Prog_enable[2]	0

21.8 High-voltage Serial Programming Algorithm

To program and verify the Atmel® AVR® ATtiny24/44/84 in the high-voltage serial programming mode, the following sequence is recommended (see instruction formats in Table 21-15 on page 152):

21.8.1 Enter High-voltage Serial Programming Mode

The following algorithm puts the device in high-voltage serial programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND.
2. Set RESET pin to “0” and toggle SCI at least six times.
3. Set the Prog_enable pins listed in Table 21-13 to “000” and wait at least 100ns.
4. Apply V_{HVRST} – 5.5V to RESET. Keep the Prog_enable pins unchanged for at least t_{HVRST} after the high-voltage has been applied to ensure the Prog_enable signature has been latched.
5. Shortly after latching the Prog_enable signature, the device will actively output data on the Prog_enable[2]/SDO pin, and the resulting drive contention may increase the power consumption. To minimize this drive contention, release the Prog_enable[2] pin after t_{HVRST} has elapsed.
6. Wait at least 50µs before giving any serial instructions on SDI/SII.

Table 21-14. High-voltage Reset Characteristics

Supply Voltage	RESET Pin High-voltage Threshold	Minimum High-voltage Period for Latching Prog_enable
V_{CC}	V_{HVRST}	t_{HVRST}
4.5V	11.5V	100ns
5.5V	11.5V	100ns

21.8.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF that is the contents of the entire EEPROM (unless the EESAVE fuse is programmed) and flash after a chip erase.
- Address high byte only needs be loaded before programming or reading a new 256-word window in flash or 256-byte EEPROM. This consideration also applies to reading signature bytes.

Table 22-7. ADC Characteristics, Differential Channels, T_A = –40°C to +125°C (Continued)

Parameter	Condition	Symbol	Min	Typ	Max	Units
Reference voltage		V_{REF}	2.56		AVCC – 0.5	V
Input voltage		V_{IN}	GND		AVCC	V
Input differential voltage		V_{DIFF}	$–V_{REF}$/gain		V_{REF}/gain	V

22.6 Serial Programming Characteristics

Figure 22-3. Serial Programming Timing

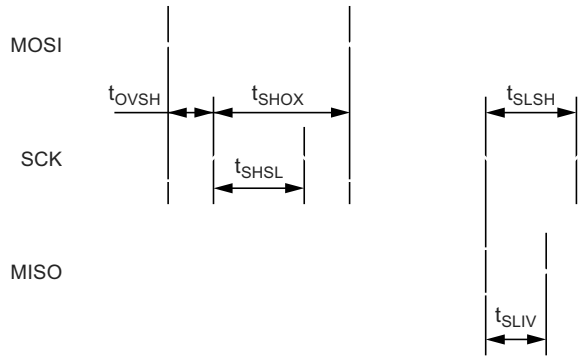


Figure 22-4. Serial Programming Waveforms

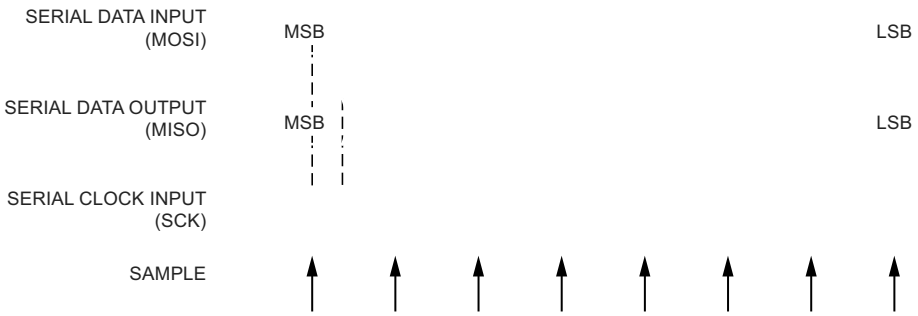


Table 22-8. Serial Programming Characteristics, T_A = –40°C to +125°C, V_{CC} = 2.7 to 5.5V (Unless Otherwise Noted)

Parameter	Symbol	Min	Typ	Max	Units
Oscillator frequency (Atmel® ATtiny24/44/84V)	$1/t_{CLCL}$	0		4	MHz
Oscillator period (Atmel ATtiny24/44/84V)	t_{CLCL}	250			ns
Oscillator frequency (ATtiny24/44/84, V_{CC} = 4.5V to 5.5V)	$1/t_{CLCL}$	0		20	MHz
Oscillator period (ATtiny24/44/84, V_{CC} = 4.5V to 5.5V)	t_{CLCL}	50			ns
SCK pulse width high	t_{SHSL}	2 t_{CLCL*}			ns
SCK pulse width low	t_{SLSH}	2 t_{CLCL*}			ns
MOSI setup to SCK high	t_{OVSH}	t_{CLCL}			ns
MOSI hold after SCK high	t_{SHOX}	2 t_{CLCL}			ns
SCK low to MISO valid	t_{SLIV}	TBD	TBD	TBD	ns

Note: 2 t_{CLCL} for f_{ck} < 12MHz, 3 t_{CLCL} for f_{ck} ≥ 12MHz

22.7 High-voltage Serial Programming Characteristics

Figure 22-5. High-voltage Serial Programming Timing

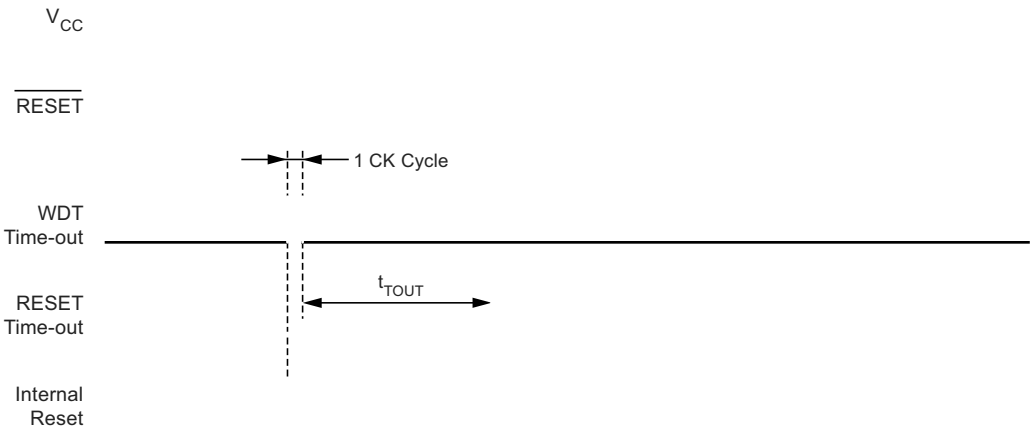


Table 22-9. High-voltage Serial Programming Characteristics
T_A = 25°C ±10%, V_{CC} = 5.0V ±10% (Unless otherwise noted)

Parameter	Symbol	Min	Typ	Max	Units
SCI (PB0) pulse width high	t_{SHSL}	110			ns
SCI (PB0) pulse width low	t_{SLSH}	110			ns
SDI (PA6), SII (PB1) valid to SCI (PB0) high	t_{IVSH}	50			ns
SDI (PA6), SII (PB1) hold after SCI (PB0) high	t_{SHIX}	50			ns
SCI (PB0) high to SDO (PA4) valid	t_{SHOV}		16		ns
Wait after Instr. 3 for write fuse bits	t_{WLWH_PFB}		2.5		ms

Figure 23-5. Active Supply Current versus V_{CC} (Internal RC Oscillator, 8MHz)

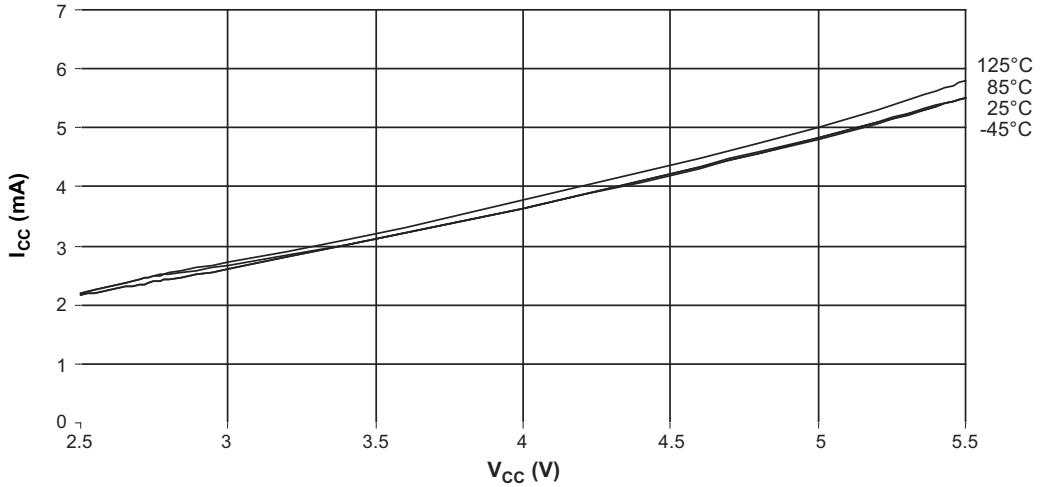


Figure 23-6. Active Supply Current versus V_{CC} (Internal RC Oscillator, 1MHz)

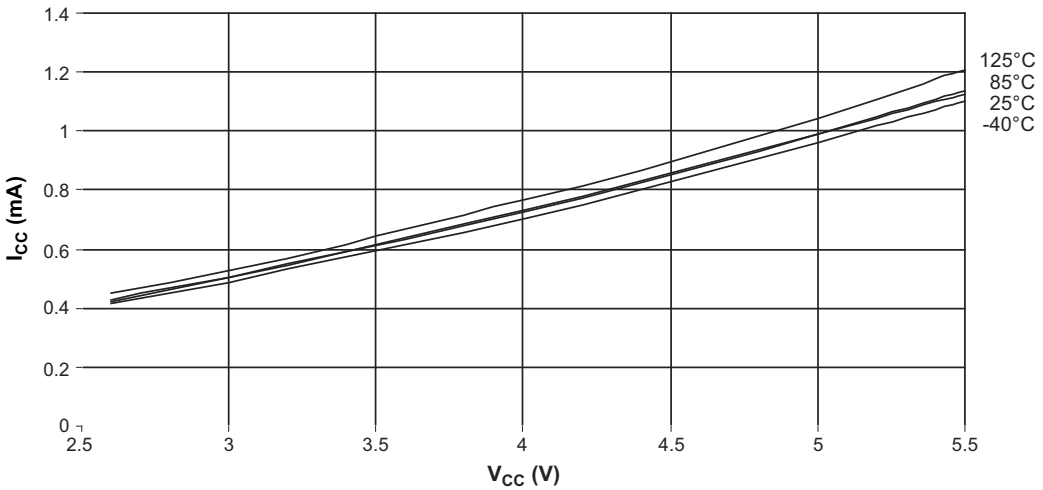
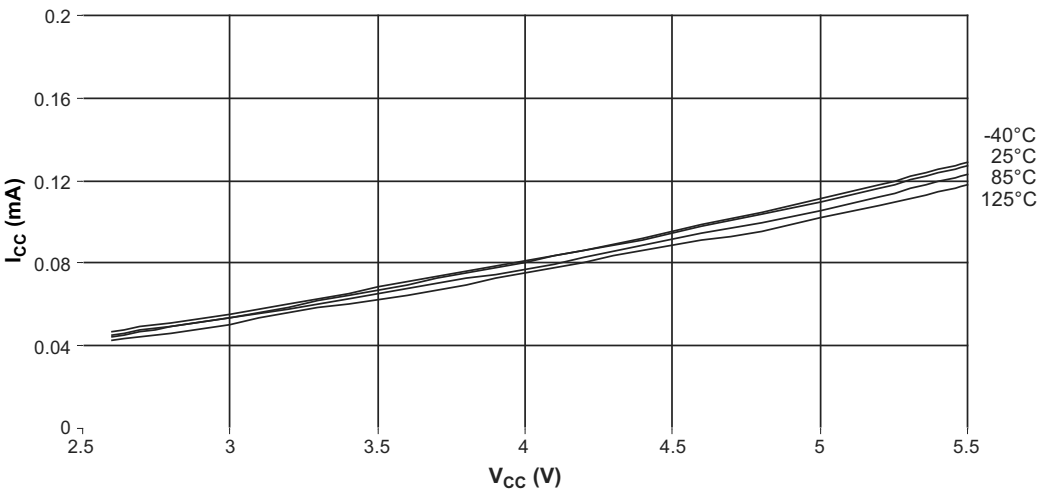


Figure 23-7. Active Supply Current versus V_{CC} (Internal RC Oscillator, 128kHz)



23.3 Supply Current of IO modules

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules are controlled by the power reduction register. See Section 8.7 “Power Reduction Register” on page 32 for details.

Table 23-1. Additional Current Consumption for the different I/O Modules (Absolute Values)

PRR bit	Typical numbers		
	V_{CC} = 2V, F = 1MHz	V_{CC} = 3V, F = 4MHz	V_{CC} = 5V, F = 8MHz
PRTIM1	6.6µA	26µA	106µA
PRTIM0	8.7µA	35µA	140µA
PRUSI	5.5µA	22µA	87µA
PRADC	22µA	87µA	340µA

23.4 Power-down Supply Current

Figure 23-13. Power-down Supply Current versus V_{CC} (Watchdog Timer Disabled)

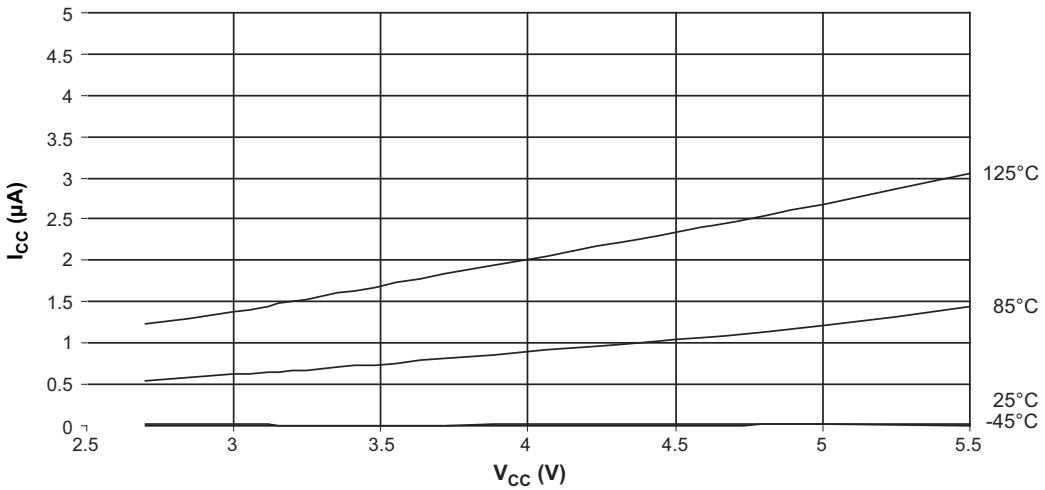
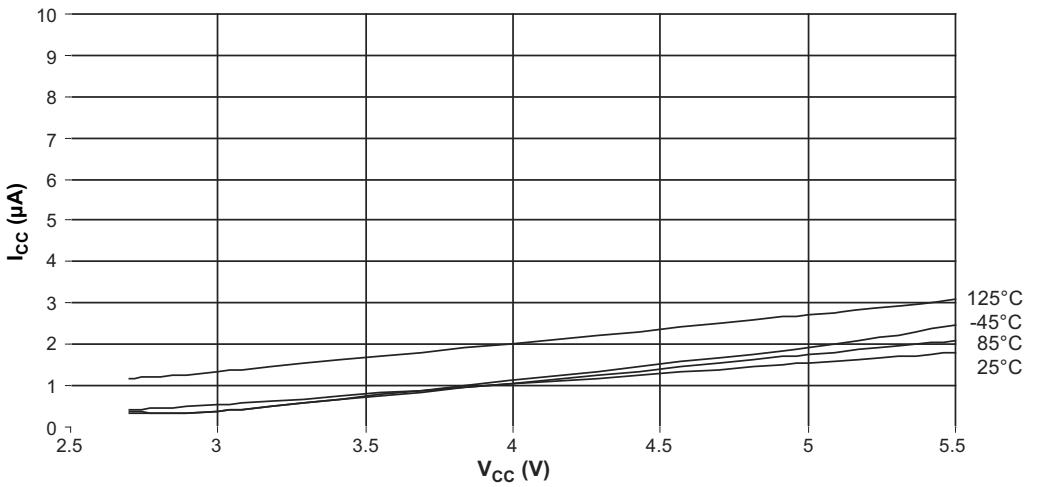
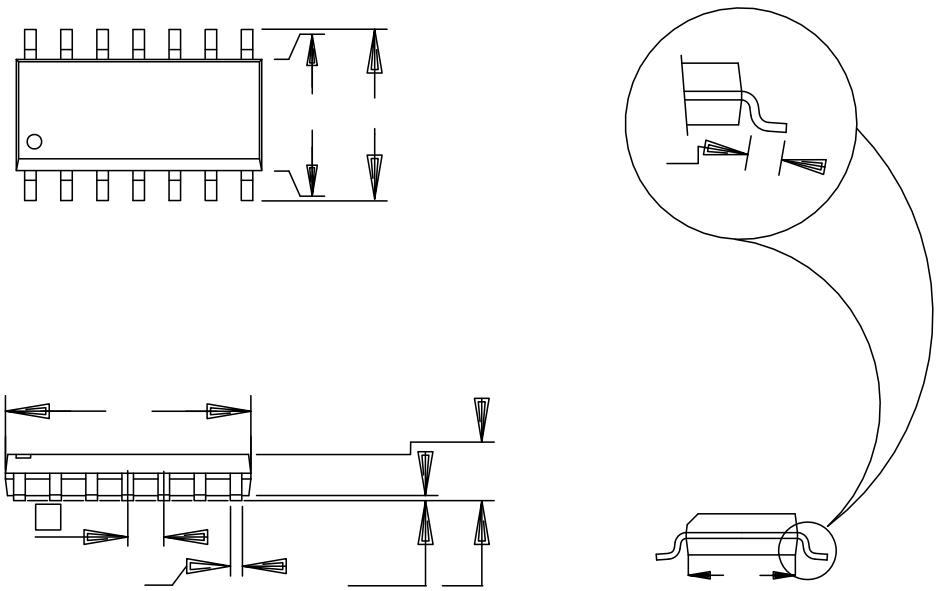


Figure 23-14. Power-down Supply Current versus V_{CC} (Watchdog Timer Enabled)



27.2 TU



		----			----	

07/27/07

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	TU, 14 Lead, 0.150" Body Width Plastic Gull Wing Small Outline Package (SOIC)		TU	C