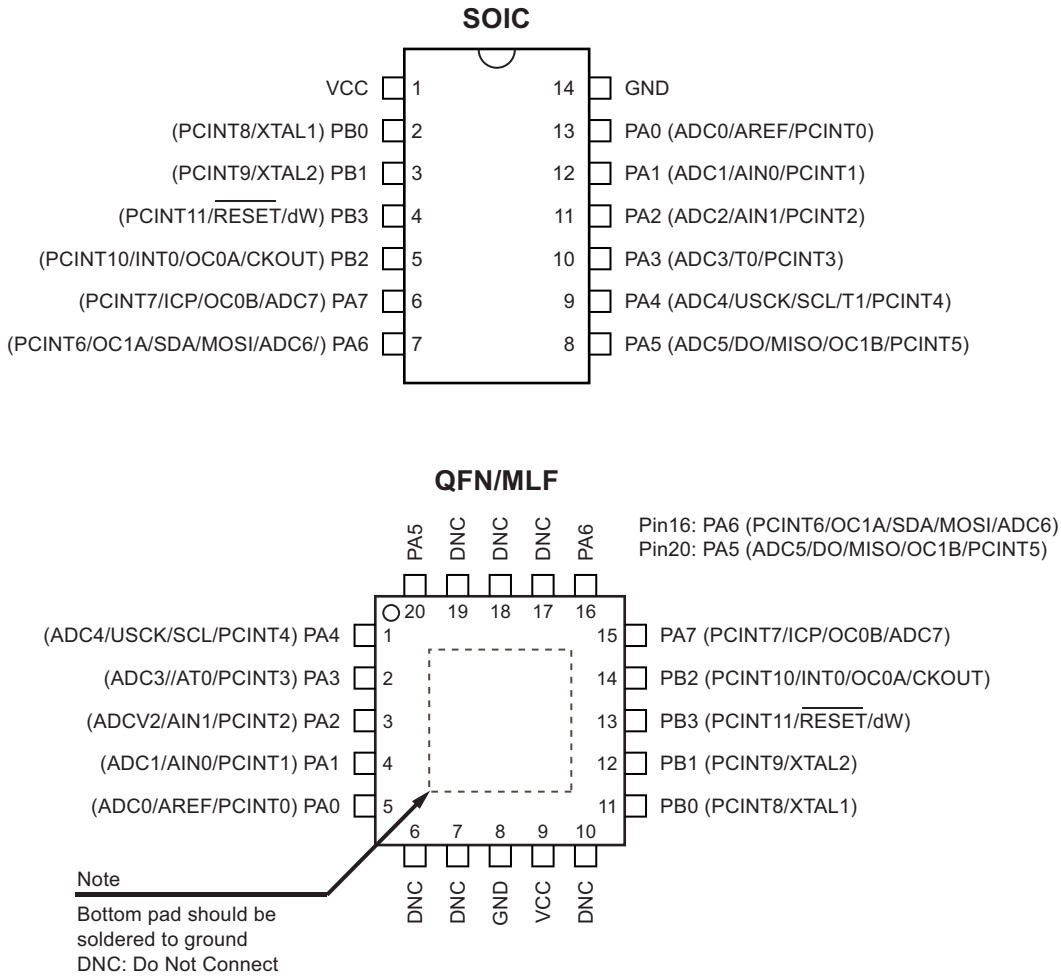
- Operating voltage:
  - 2.7 - 5.5V for Atmel ATtiny24/44/84
- Speed grade
  - Atmel ATtiny24/44/84: 0 - 8MHz at 2.7 - 5.5V, 0 - 16MHz at 4.5 - 5.5V
- Automotive temperature range
- Low power consumption
  - Active mode:
    - 1MHz, 2.7V: 800µA
  - Power-down mode:
    - 2.7V: 2.0µA

1. Pin Configurations

Figure 1-1. Pinout Atmel ATtiny24/44/84



1.1 Disclaimer

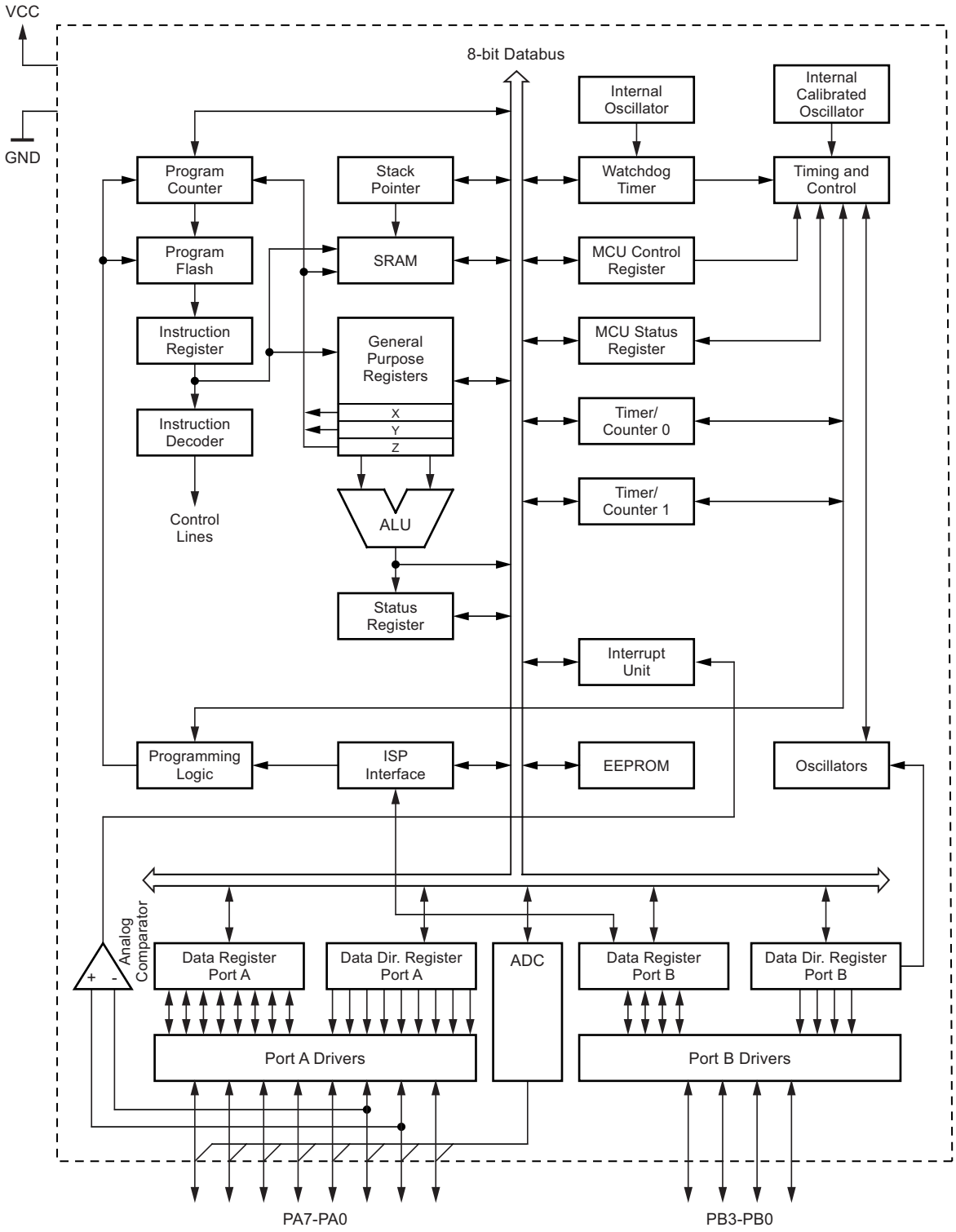
Typical values contained in this data sheet are based on simulations and characterization of actual Atmel® ATtiny24/44/84 AVR® microcontrollers manufactured on the typical process technology. Applicable automotive min. and max. values will be available after devices representative of the whole process excursion (corner run) have been characterized.

2. Overview

The Atmel® ATtiny24/44/84 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 Atmel ATtiny24/44/84 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the arithmetic logic unit (ALU), allowing two independent registers to be accessed in one 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 Atmel ATtiny24/44/84 provides the following features: 2/4/8K bytes of in-system programmable flash, 128/256/512 bytes EEPROM, 128/256/512 bytes SRAM, 12 general purpose I/O lines, 32 general purpose working registers, an 8-bit Timer/Counter with two PWM channels, a 16-bit Timer/Counter with two PWM channels, internal and external interrupts, an 8-channel 10-bit ADC, programmable gain stage (1x, 20x) for 12 differential ADC channel pairs, a programmable watchdog timer with internal oscillator, internal calibrated oscillator, and three software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, analog comparator, and interrupt system to continue functioning. The power-down mode saves the register contents, disabling all chip functions until the next interrupt or hardware reset. The ADC noise reduction mode stops the CPU and all I/O modules except 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.

The device is manufactured using Atmel's high-density non-volatile memory technology. The on-chip ISP flash allows the program memory to be re-programmed in-system through an SPI serial interface, by a conventional non-volatile memory programmer or by an on-chip boot code running on the AVR core.

The Atmel ATtiny24/44/84 AVR 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.

2.2 Automotive Quality Grade

The Atmel ATtiny24/44/84 has been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949 grade 1. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the Atmel ATtiny24/44/84 have been verified during regular product qualification as per AEC-Q100.

As indicated in the ordering information paragraph, the product is available in only one temperature grade.

Table 2-1. Temperature Grade Identification for Automotive Products

Temperature	Temperature Identifier	Comments
–40; +125	Z	Full automotive temperature range

2.3 Pin Descriptions

2.3.1 VCC

Supply voltage.

2.3.2 GND

Ground.

2.3.3 Port B (PB3...PB0)

Port B is a 4-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability except PB3, which has the RESET capability. To use pin PB3 as an I/O pin instead of RESET pin, program (‘0’) RSTDISBL fuse. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the Atmel ATtiny24/44/84 as listed on Section 12.3 “Alternate Port Functions” on page 54.

2.3.4 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 9-1 on page 37. Shorter pulses are not guaranteed to generate a reset.

2.3.5 Port A (PA7...PA0)

Port A is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A has an alternate function as analog input for the ADC, analog comparator, Timer/Counter, SPI and pin change interrupt as described in Section 12.3 “Alternate Port Functions” on page 54.

3. Resources

A comprehensive set of development tools, driver and application notes, and datasheets are available for download on http://www.atmel.com/avr.

4. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part-specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files, and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

For I/O registers located in the extended I/O map, IN, OUT, SBIS, SBIC, CBI, and SBI instructions must be replaced with instructions that allow access to extended I/O, typically LDS and STS combined with SBRS, SBRC, SBR, and CBR.

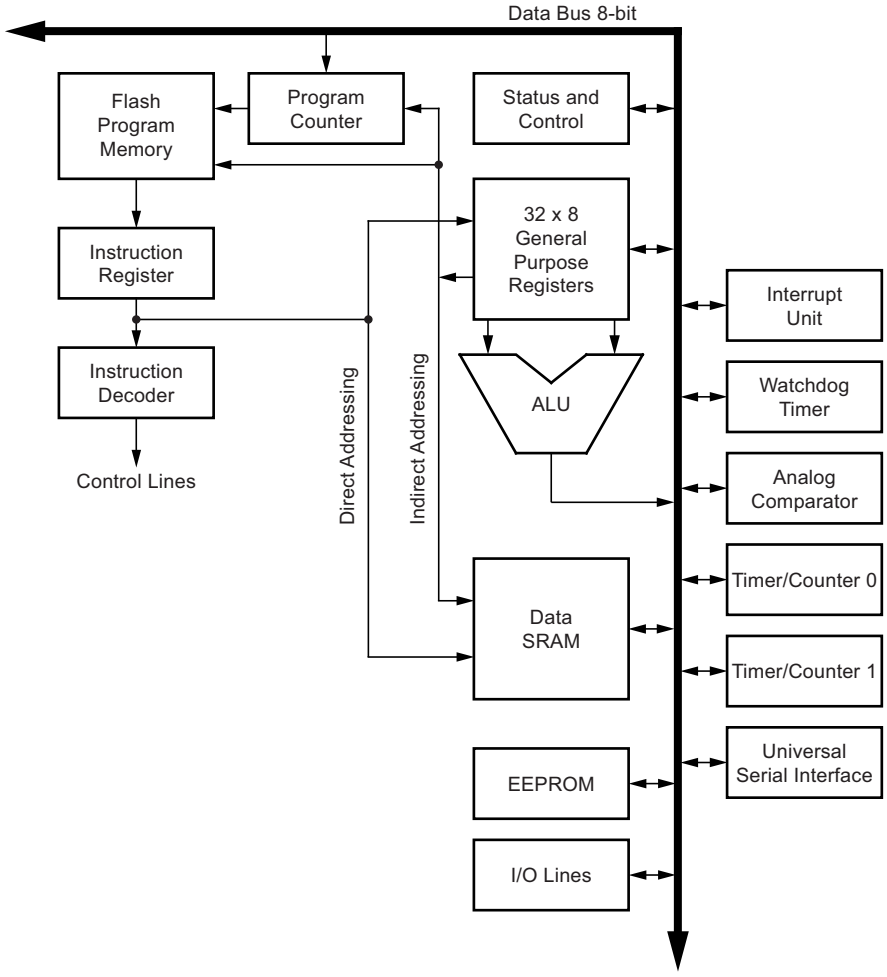
5. CPU Core

5.1 Overview

This section discusses the Atmel® AVR® core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must, therefore, be able to access memories, perform calculations, control peripherals, and handle interrupts.

5.2 Architectural Overview

Figure 5-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture, with separate memories and buses for program and data. Instructions in the program memory are executed with a single-level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system reprogrammable flash memory.

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.

5.4.1 SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
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 – I: Global Interrupt Enable**

The global interrupt enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the global interrupt enable register is cleared, none of the interrupts are enabled independently of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set summary.

- **Bit 6 – T: Bit Copy Storage**

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The half carry flag H indicates a half carry in some arithmetic operations. Half carry is useful in BCD arithmetic. See the instruction set reference for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive OR between the negative flag N and the two's complement overflow flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The two's complement overflow flag V supports two's complement arithmetic. See the instruction set summary for detailed information.

- **Bit 2 – N: Negative Flag**

The negative flag N indicates a negative result in an arithmetic or logic operation. See the instruction set summary for detailed information.

- **Bit 1 – Z: Zero Flag**

The zero flag Z indicates a zero result in an arithmetic or logic operation. See the instruction set summary for detailed information.

- **Bit 0 – C: Carry Flag**

The carry flag C indicates a carry in an arithmetic or logic operation. See the instruction set summary for detailed information.

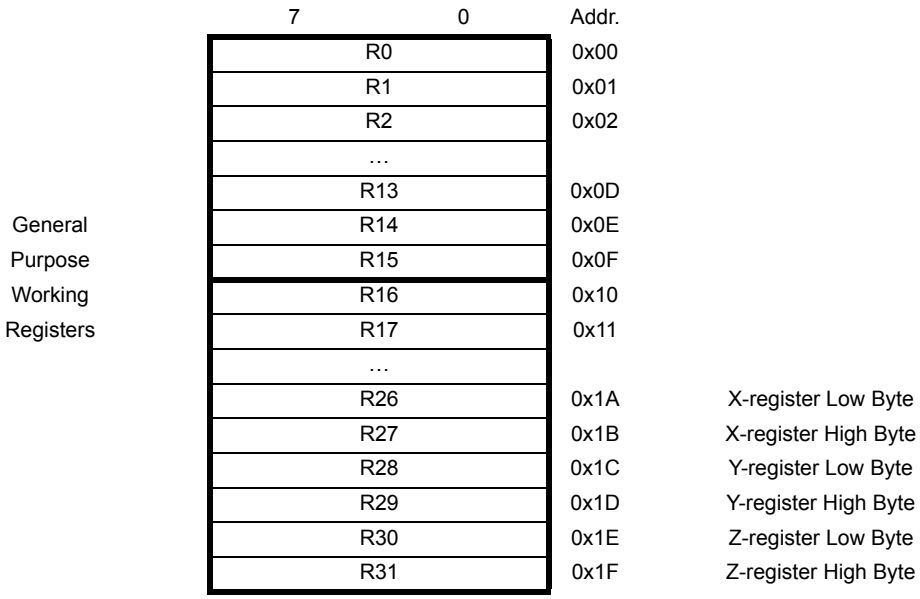
5.5 General Purpose Register File

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

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

Figure 5-2 on page 12 shows the structure of the 32 general purpose working registers in the CPU.

Figure 5-2. AVR CPU General Purpose Working Registers



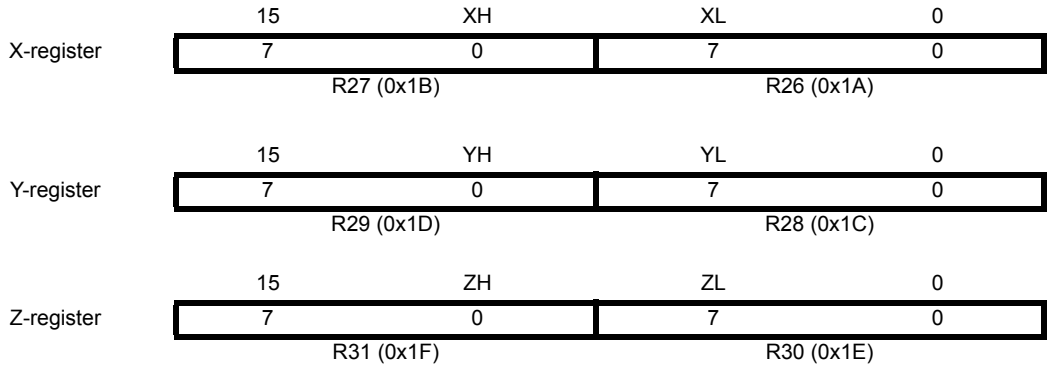
Most of the instructions operating on the register file have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 5-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-registers can be set to index any register in the file.

5.5.1 The X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 5-3 on page 13.

Figure 5-3. The X-, Y-, and Z-registers



In the different addressing modes, these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set summary for details).

5.6 Stack Pointer

The stack is mainly used for storing temporary data, for storing local variables, and for storing return addresses after interrupts and subroutine calls. The stack pointer register always points to the top of the stack. Note that the stack is implemented as growing from higher memory locations to lower memory locations. This implies that a stack PUSH instruction decreases the stack pointer.

The stack pointer points to the data SRAM stack area where the subroutine and interrupt stacks are located. This stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The stack pointer must be set to point above 0x60. The stack pointer is decremented by one when data are pushed onto the stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the stack with a subroutine call or interrupt. The stack pointer is incremented by one when data are popped from the stack with the POP instruction, and it is incremented by two when data are popped from the stack with a return from subroutine RET instruction or return from interrupt RETI instruction.

The Atmel AVR® stack pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only the SPL register is needed. In this case, the SPH register will not be present.

5.6.1 SPH and SPL – Stack Pointer High and Low

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

5.7 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The Atmel® AVR® CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 5-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast access register file concept. This is the basic pipelining concept to obtain up to 1MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 5-4. The Parallel Instruction Fetches and Instruction Executions

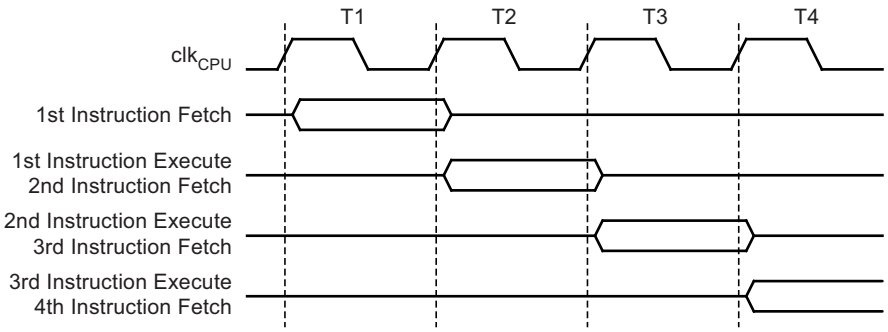
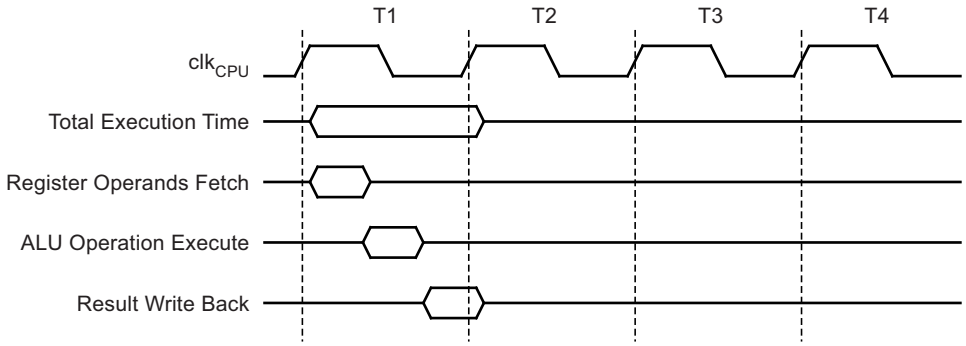


Figure 5-5 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 5-5. Single-cycle ALU Operation



5.8 Reset and Interrupt Handling

The Atmel AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written with a logical one together with the global interrupt enable bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are by default defined as the reset and interrupt vectors. The complete list of vectors is shown in Section 10. “Interrupts” on page 44. The list also determines the priority levels of the different interrupts. The lower the address, the higher the priority level. RESET has the highest priority, and next is INT0, the external interrupt request 0.

When an interrupt occurs, the global interrupt enable I-bit is cleared and all interrupts are disabled. The user software can write a logical 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 interrupt 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 Atmel® 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, EEMPE  ; start EEPROM write
sbi   EECR, EEPE
out   SREG, r16    ; restore SREG value (I-bit)
```

C Code Example

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

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei   ; set Global Interrupt Enable
sleep ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set Global Interrupt Enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

5.8.1 Interrupt Response Time

The interrupt execution response for all the enabled Atmel AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock-cycle period, the program counter is pushed onto the stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the program counter (two bytes) is popped back from the stack, the stack pointer is incremented by two, and the I-bit in SREG is set.

6. Memories

This section describes the different memories in the Atmel® ATtiny24/44/84. The AVR® architecture has two main memory spaces, the data memory space and the program memory space. In addition, the Atmel ATtiny24/44/84 features an EEPROM memory for data storage. All three memory spaces are linear and regular.

6.1 In-System Re-programmable Flash Program Memory

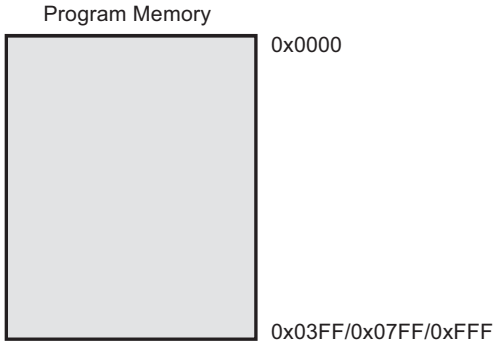
The Atmel ATtiny24/44/84 contains 2/4/8K bytes of on-chip, in-system, reprogrammable flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the flash memory is organized as 1024/2048/4096 x 16.

The flash memory has an endurance of at least 10,000 write/erase cycles. The Atmel ATtiny24/44/84 program counter (PC) is 10/11/12 bits wide, thus addressing the 1024/2048/4096 program memory locations. Section 21. “Memory Programming” on page 141 contains a detailed description on flash data serial downloading using the SPI pins.

Constant tables can be allocated within the entire program memory address space (see the LPM – load program memory instruction description).

Timing diagrams for instruction fetch and execution are presented in Section 5.7 “Instruction Execution Timing” on page 14.

Figure 6-1. Program Memory Map



6.2 SRAM Data Memory

Figure 6-2 on page 17 shows how the ATtiny24/44/84 SRAM memory is organized.

The lower 160 data memory locations address the register file, the I/O memory, and the internal data SRAM. The first 32 locations address the register file, the next 64 locations the standard I/O memory, and the last 128/256/512 locations address the internal data SRAM.

The five different addressing modes for the data memory are: direct, indirect with displacement, indirect, indirect with pre-decrement, and indirect with post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

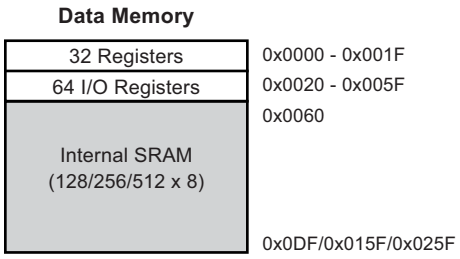
The direct addressing mode reaches the entire data space.

The indirect with displacement mode reaches 63 address locations starting from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O registers, and 128/256/512 bytes of internal data SRAM in the Atmel ATtiny24/44/84 are all accessible through all these addressing modes. The register file is described in Section 5.5 “General Purpose Register File” on page 12.

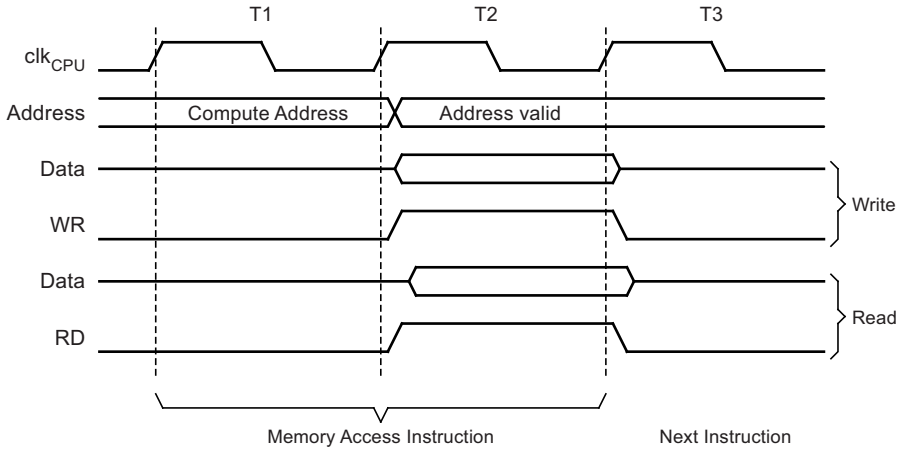
Figure 6-2. Data Memory Map



6.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 6-3.

Figure 6-3. On-chip Data SRAM Access Cycles



6.3 EEPROM Data Memory

The Atmel® ATtiny24/44/84 contains 128/256/512 bytes of EEPROM data memory. It is organized as a separate data space in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following sections specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register. For a detailed description of serial data downloading to the EEPROM, see Section 21.6 “Serial Downloading” on page 144.

6.3.1 EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access times for the EEPROM are given in Table 6-1 on page 22. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write to the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than the specified minimum for the clock frequency used. See Section 6.3.6 “Preventing EEPROM Corruption” on page 20 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. See Section 6.3.2 “Atomic Byte Programming” on page 18 and Section 6.3.3 “Split Byte Programming” on page 18 for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

6.3.2 Atomic Byte Programming

Atomic byte programming is the simplest mode. When writing a byte to the EEPROM, the user must write the address into the EEARL register and data into EEDR register. If the EEPMn bits are zero, writing EEPE (within four cycles after EEMPE is written) will trigger the erase/write operation. Both the erase and write cycle are done in one operation and the total programming time is given in Table 1. The EEPE bit remains set until the erase and write operations are completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

6.3.3 Split Byte Programming

It is possible to split the erase and write cycle into two different operations. This may be useful if the system requires short access time for some limited period of time (typically if the power supply voltage falls). In order to take advantage of this method, the locations to be written are required to have been erased before the write operation. But because the erase and write operations are split, it is possible to do the erase operations when the system allows time-critical operations to be done (typically after power-up).

6.3.4 Erase

To erase a byte, the address must be written to EEAR. If the EEPMn bits are 0b01, writing the EEPE (within four cycles after EEMPE is written) will trigger the erase operation only (programming time is given in Table 1). The EEPE bit remains set until the erase operation completes. While the device is busy programming, it is not possible to do any other EEPROM operations.

6.3.5 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEPMn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in Table 1). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated oscillator is used to time the EEPROM accesses. Make sure the oscillator frequency is within the requirements described in Section 7.10.1 “Oscillator Calibration Register – OSCCAL” on page 29.

The following code examples show one assembly function and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_write:
        ; Wait for completion of previous write
        sbic    EECR,EEPE
        rjmp    EEPROM_write
        ; Set Programming mode
        ldi     r16, (0<<EEPM1)|(0<<EEPM0)
        out     EECR, r16
        ; Set up address (r17) in address register
        out     EEARL, r17
        ; Write data (r16) to data register
        out     EEDR,r16
        ; Write logical one to EEMPE
        sbi     EECR,EEMPE
        ; Start eeprom write by setting EEPE
        sbi     EECR,EEPE
        ret
```

C Code Example

```
void EEPROM_write(unsigned char ucAddress, unsigned char ucData)
{
        /* Wait for completion of previous write */
        while(EECR & (1<<EEPE))
        ;
        /* Set Programming mode */
        EECR = (0<<EEPM1)|(0>>EEPM0)
        /* Set up address and data registers */
        EEARL = ucAddress;
        EEDR = ucData;
        /* Write logical one to EEMPE */
        EECR |= (1<<EEMPE);
        /* Start eeprom write by setting EEPE */
        EECR |= (1<<EEPE);
}
```

Note: The code examples are only valid for Atmel® ATtiny24 and Atmel ATtiny44, using 8-bit addressing mode.

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
        ; Wait for completion of previous write
        sbic    EECR,EEPE
        rjmp    EEPROM_read
        ; Set up address (r17) in address register
        out     EEARL, r17
        ; Start eeprom read by writing EERE
        sbi     EECR,EERE
        ; Read data from data register
        in      r16,EEDR
        ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned char ucAddress)
{
        /* Wait for completion of previous write */
        while(EECR & (1<<EEPE))
        ;
        /* Set up address register */
        EEARL = ucAddress;
        /* Start eeprom read by writing EERE */
        EECR |= (1<<EERE);
        /* Return data from data register */
        return EEDR;
}
```

Note: The code examples are only valid for Atmel® ATtiny24 and Atmel ATtiny44, using 8-bit addressing mode.

6.3.6 Preventing EEPROM Corruption

During periods of low VCC, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. A regular write sequence to the EEPROM requires a minimum voltage to operate correctly, and the CPU itself can execute instructions incorrectly if the supply voltage is too low. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

EEPROM data corruption can easily be avoided by following this design recommendation: Keep the AVR® RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal brown-out detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low $-V_{CC}$ reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided the power supply voltage is sufficient.

6.4 I/O Memory

The I/O space definition of the Atmel ATtiny24/44/84 is shown in Section 23-43 “Minimum Reset Pulse Width versus V_{CC}” on page 181.

All Atmel ATtiny24/44/84 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. See the instruction set summary for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written with a logical zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®s, the CBI and SBI instructions will only operate on the specified bit, and can, therefore, be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

6.4.1 General Purpose I/O Registers

The ATtiny24/44/84 contains three general purpose I/O registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags. General purpose I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

6.5 Register Description

6.5.1 EEARH – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	–	–	–	–	–	–	–	**EEAR8**	EEARH
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	X	

- **Bits 7..1 – Res: Reserved Bits**

These are reserved bits in the ATtiny24/44/84, and will always read as zero.

- **Bit 0 – EEAR8: EEPROM Address**

The EEPROM address register, EEARH, specifies the most-significant bit for EEPROM address in the 512-byte EEPROM space for Tiny84. This bit is reserved in the ATtiny24/44, and will always read as zero. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

6.5.2 EEARL – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	**EEAR7**	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	EEARL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	

- **Bits 7..0 – EEAR7..0: EEPROM Address**

The EEPROM address register, EEARL, specifies the EEPROM address. In the 128-byte EEPROM space in ATiny24, bit 7 is reserved and will always read as zero. The EEPROM data bytes are addressed linearly between 0 and 128/256/512. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

6.5.3 EEDR – EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	**EEDR7**	**EEDR6**	**EEDR5**	**EEDR4**	**EEDR3**	**EEDR2**	**EEDR1**	**EEDR0**	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – EEDR7..0: EEPROM Data**

For the EEPROM write operation the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

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.

6.5.5 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	MSB							LSB	GPIOR2
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	

6.5.6 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x14 (0x34)	MSB							LSB	GPIOR1
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	

6.5.7 GPIOR0 – General Purpose I/O Register 0

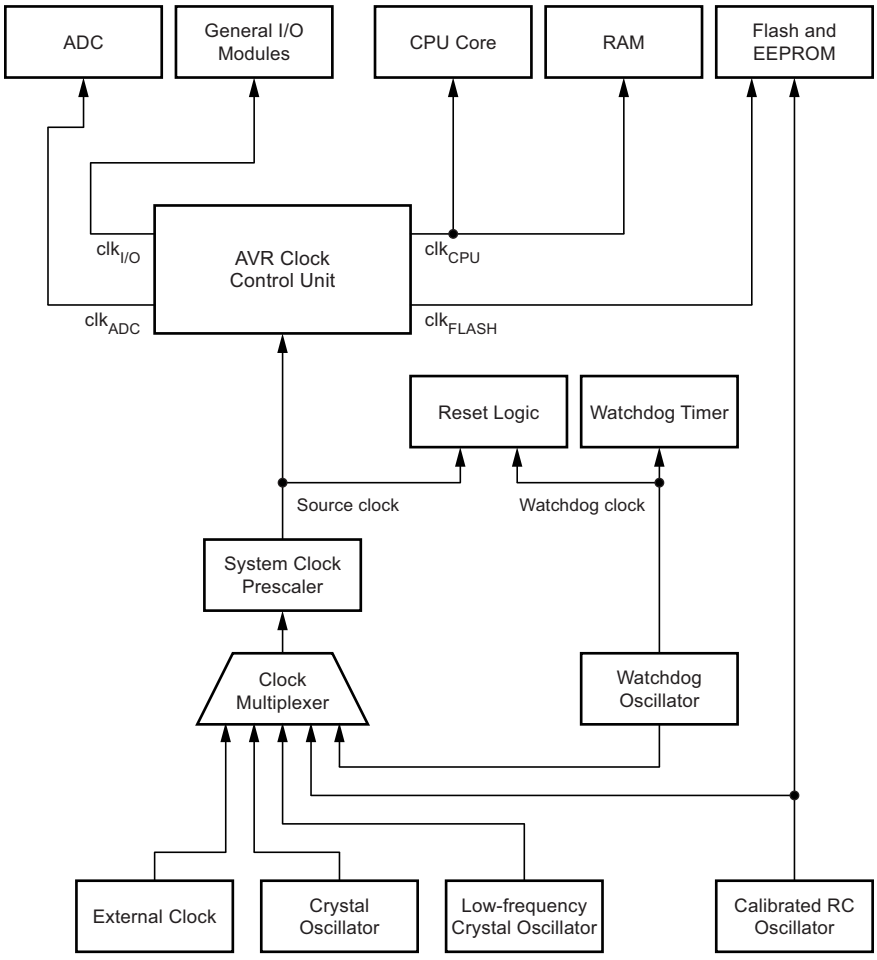
Bit	7	6	5	4	3	2	1	0	
0x13 (0x33)	MSB							LSB	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

7. System Clock and Clock Options

7.1 Clock Systems and their Distribution

Figure 7-1 presents the principal clock systems in the AVR® and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in Section 8. “Power Management and Sleep Modes” on page 31. The clock systems are detailed below.

Figure 7-1. Clock Distribution



7.1.1 CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the general purpose register file, the status register and the data memory holding the stack pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

7.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like the Timer/Counter. The I/O clock is also used by the external interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted.

7.1.3 Flash Clock – clk_{FLASH}

The flash clock controls operation of the flash interface. The flash clock is usually active simultaneously with the CPU clock.

7.1.4 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O

clocks in order to reduce noise generated by their digital circuitry. This gives more accurate ADC conversion results.

7.2 Clock Sources

The device has the following clock source options, selectable by flash fuse bits as shown below. The clock from the selected source is input to the Atmel® AVR® clock generator, and routed to the appropriate modules.

Table 7-1. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3..0
External clock	0000
Calibrated internal RC oscillator 8.0MHz	0010
Watchdog oscillator 128kHz	0100
External low-frequency oscillator	0110
External crystal/ceramic resonator	1000-1111
Reserved	0101, 0111, 0011,0001

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from power-down or power-save, the selected clock source is used to time the start-up, ensuring stable oscillator operation before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before commencing normal operation. The watchdog oscillator is used for timing this real-time part of the start-up time. The number of WDT oscillator cycles used for each time-out is shown in Table 7-2.

Table 7-2. Number of Watchdog Oscillator Cycles

Typ Time-out	Number of Cycles
4ms	512
64ms	8K (8,192)

7.3 Default Clock Source

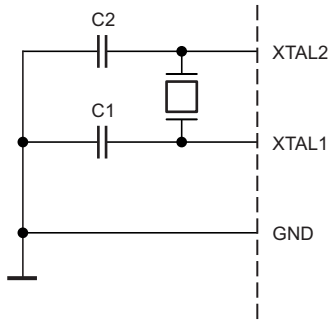
The device is shipped with CKSEL = “0010”, SUT = “10”, and CKDIV8 programmed. The default clock source setting is therefore the internal RC oscillator running at 8.0MHz with longest start-up time and an initial system clock prescaling of 8, resulting in 1.0MHz system clock. This default setting ensures that all users can make their desired clock source setting using an in-system or high-voltage programmer.

7.4 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 7-2 Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 7-3 on page 26. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 7-2. Crystal Oscillator Connections



The oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 7-3.

Table 7-3. Crystal Oscillator Operating Modes

CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100(1)	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 fuse together with the SUT1..0 fuses select the start-up times as shown in Table 7-4.

Table 7-4. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258CK(1)	14CK + 4.1ms	Ceramic resonator, fast rising power
0	01	258CK(1)	14CK + 65ms	Ceramic resonator, slowly rising power
0	10	1KCK(2)	14CK	Ceramic resonator, BOD enabled
0	11	1KCK(2)	14CK + 4.1ms	Ceramic resonator, fast rising power
1	00	1KCK(2)	14CK + 65ms	Ceramic resonator, slowly rising power
1	01	16KCK	14CK	Crystal oscillator, BOD enabled
1	10	16KCK	14CK + 4.1ms	Crystal oscillator, fast rising power
1	11	16KCK	14CK + 65ms	Crystal oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.

2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

7.5 Low-frequency Crystal Oscillator

To use a 32.768kHz watch crystal as the clock source for the device, the low-frequency crystal oscillator must be selected by setting CKSEL fuses to “0110”. The crystal should be connected as shown in Figure 7-2 on page 26. See the 32kHz crystal oscillator application note for details on oscillator operation and how to choose appropriate values for C1 and C2.

When this oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 7-5.

Table 7-5. Start-up Times for the Low Frequency Crystal Oscillator Clock Selection

SUT1..0	Start-up Time from Power Down and Power Save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended usage
00	1KCK(1)	4ms	Fast rising power or BOD enabled
01	1KCK(1)	64ms	Slowly rising power
10	32KCK	64ms	Stable frequency at start-up
11	Reserved		

Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

7.6 Calibrated Internal RC Oscillator

By default, the internal RC oscillator provides an approximate 8MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. See Table 22-2 on page 157 and Section 23.9 “Internal Oscillator Speed” on page 176 for more details. The device is shipped with the CKDIV8 fuse programmed. See Section 7.9 “System Clock Prescaler” on page 29 for more details.

This clock may be selected as the system clock by programming the CKSEL fuses as shown in Table 7-6. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL register and thereby automatically calibrates the RC oscillator. The accuracy of this calibration is shown as factory calibration in Table 22-2 on page 157.

By changing the OSCCAL register from SW, see Section 7.10.1 “Oscillator Calibration Register – OSCCAL” on page 29, it is possible to get higher calibration accuracy than by using the factory calibration. The accuracy of this calibration is shown as User calibration in Table 22-2 on page 157.When this oscillator is used as the chip clock, the watchdog oscillator will still be used for the watchdog timer and for the reset time-out. For more information on the pre-programmed calibration value, see the section Section 21.4 “Calibration Byte” on page 143.

Table 7-6. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency
0010(1)	8.0 MHz

Note: 1. The device is shipped with this option selected.

When this oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 7-7.

Table 7-7. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

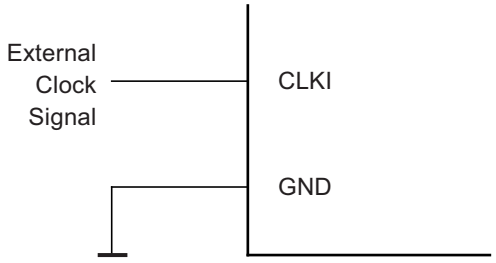
SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10(1)	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

Note: 1. The device is shipped with this option selected.

7.7 External Clock

To drive the device from an external clock source, CLKI should be driven as shown in Figure 7-3. To run the device on an external clock, the CKSEL fuses must be programmed to “0000”.

Figure 7-3. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 7-8.

Table 7-8. Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset	Recommended Usage
00	6CK	14CK	BOD enabled
01	6CK	14CK + 4ms	Fast rising power
10	6CK	14CK + 64ms	Slowly rising power
11	Reserved		

When applying an external clock, sudden changes in the applied clock frequency must be avoided 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. See Section 7.9 “System Clock Prescaler” on page 29 for details.

7.8 128kHz Internal Oscillator

The 128kHz internal oscillator is a low power oscillator providing a 128kHz clock. The frequency is nominal at 3V and 25°C. This clock may be selected as the system clock by programming the CKSEL fuses to "0100".

When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 7-9.

Table 7-9. Start-up Times for the 128kHz Internal Oscillator

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset	Recommended Usage
00	6CK	14CK	BOD enabled
01	6CK	14CK + 4ms	Fast rising power
10	6CK	14CK + 64ms	Slowly rising power
11	Reserved		

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.+

- **Bits 6..4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84 and will always read as zero.

- **Bits 3..0 – CLKPS3..0: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written at run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 7-10.

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

The CKDIV8 fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of eight at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 fuse setting. The application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 fuse programmed.

Table 7-10. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

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.

8.4 Power-down Mode

When the SM1..0 bits are written to “10”, the SLEEP instruction makes the MCU enter power-down mode. In this mode, the oscillator is stopped, while the external interrupts and the watchdog continue operating (if enabled). Only an external reset, a watchdog reset, a brown-out reset, an external level interrupt on INT0, or a pin change interrupt can wake up the MCU. This sleep mode halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. See Section 11. “External Interrupts” on page 46 for details

8.5 Standby Mode

When the SM1..0 bits are 11 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.

8.6 Software BOD Disable

When the brown-out detector (BOD) is enabled by BODLEVEL fuses (see Table 21-4 on page 142), the BOD is actively monitoring the supply voltage during a sleep period. In some devices it is possible to save power by disabling the BOD by software in power-down and stand-by sleep modes. The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses.

If BOD is disabled by software, the BOD function is turned off immediately after entering the sleep mode. Upon wake-up from sleep, BOD is automatically enabled again. This ensures safe operation in case the V_{CC} level has dropped during the sleep period.

When the BOD has been disabled, the wake-up time from sleep mode will be approximately 60µs to ensure that the BOD is working correctly before the MCU continues executing code.

BOD disable is controlled by the BODS (BOD Sleep) bit of MCU control register, see Section 8.9.1 “MCUCR – MCU Control Register” on page 34. Writing this bit to one turns off BOD in power-down and stand-by, while writing a zero keeps the BOD active. The default setting is zero, i.e. BOD active.

Writing to the BODS bit is controlled by a timed sequence and an enable bit, see Section 8.9.1 “MCUCR – MCU Control Register” on page 34.

8.7 Power Reduction Register

The power reduction register (PRR), see Section 8.9.2 “PRR – Power Reduction Register” on page 35, 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 cannot be read or written. Resources used by the peripheral when stopping the clock will remain occupied; hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in idle mode and active mode to significantly reduce the overall power consumption. See Section 23.4 “Power-down Supply Current” on page 168 for examples. In all other sleep modes, the clock is already stopped.

8.8 Minimizing Power Consumption

There are several issues 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 of the device's functions as possible 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.

8.8.1 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. See Section 18. “Analog-to-Digital Converter” on page 118 for details on ADC operation.

8.8.2 Analog Comparator

When entering idle mode, the analog comparator should be disabled if not used. When entering ADC noise reduction mode, the analog comparator should be disabled. In the other sleep modes, the analog comparator is automatically disabled. However, if the analog comparator is set up to use the internal voltage reference as input, the analog comparator should be disabled in all sleep modes. Otherwise, the internal voltage reference will be enabled, independent of sleep mode. See Section 17. “Analog Comparator” on page 115 for details on how to configure the analog comparator.

8.8.3 Brown-out Detector

If the brown-out detector is not needed in the application, this module should be turned off. If the brown-out detector is enabled by the BODLEVEL fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. See Section 9.5 “Brown-out Detection” on page 38 for details on how to configure the brown-out detector.

8.8.4 Internal Voltage Reference

The internal voltage reference will be enabled when needed by the brown-out detection, the analog comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. See Section 9.7 “Internal Voltage Reference” on page 39 for details on the start-up time.

8.8.5 Watchdog Timer

If the watchdog timer is not needed in the application, this module should be turned off. If the watchdog timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. See Section 9.8 “Watchdog Timer” on page 39 for details on how to configure the watchdog timer.

8.8.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. See the section Section 12.2.5 “Digital Input Enable and Sleep Modes” on page 53 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or has an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. digital input buffers can be disabled by writing to the digital input disable register (DIDR0). See Section 18.10.5 “DIDR0 – Digital Input Disable Register 0” on page 134 for details.

8.9 Register Description

8.9.1 MCUCR – MCU Control Register

The MCU control register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
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 – BODS: BOD Sleep**

In order to disable BOD during sleep (see Table 8-1 on page 31) the BODS bit must be written to logic one. This is controlled by a timed sequence and the enable bit, BODSE in MCUCR. First, both BODS and BODSE must be set to one. Second, within four clock cycles, BODS must be set to one and BODSE must be set to zero. The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

- **Bit 5 – SE: Sleep Enable**

The SE bit must be written to logical 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, it is recommended to set the sleep enable (SE) bit just before the execution of the SLEEP instruction and to clear it immediately after waking up.

- **Bits 4, 3 – SM1..0: Sleep Mode Select Bits 2..0**

These bits select between the three available sleep modes as shown in Table 8-2.

Table 8-2. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	ADC Noise Reduction
1	0	Power-down
1	1	Standby(1)

Note: 1. Only recommended with external crystal or resonator selected as clock source

- **Bit 2 – BODSE: BOD Sleep Enable**

The BODSE bit enables setting of BODS control bit, as explained on BODS bit description. BOD disable is controlled by a timed sequence.

8.9.2 PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	PRTIM1	PRTIM0	PRUSI	PRADC	PRR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 6, 5, 4- Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84 and will always read as zero.

- **Bit 3- PRTIM1: Power Reduction Timer/Counter1**

Writing a logical one to this bit shuts down the Timer/Counter 1 module. When the Timer/Counter 1 is enabled, operation will continue like before the shutdown.

- **Bit 2- PRTIM0: Power Reduction Timer/Counter0**

Writing a logical one to this bit shut s down the Timer/Counter 0 module. When the Timer/Counter 0 is enabled, operation will continue like before the shutdown.

- **Bit 1 - PRUSI: Power Reduction USI**

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

- **Bit 0 - PRADC: Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shutdown. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

9. System Control and Reset

9.1 Resetting the AVR

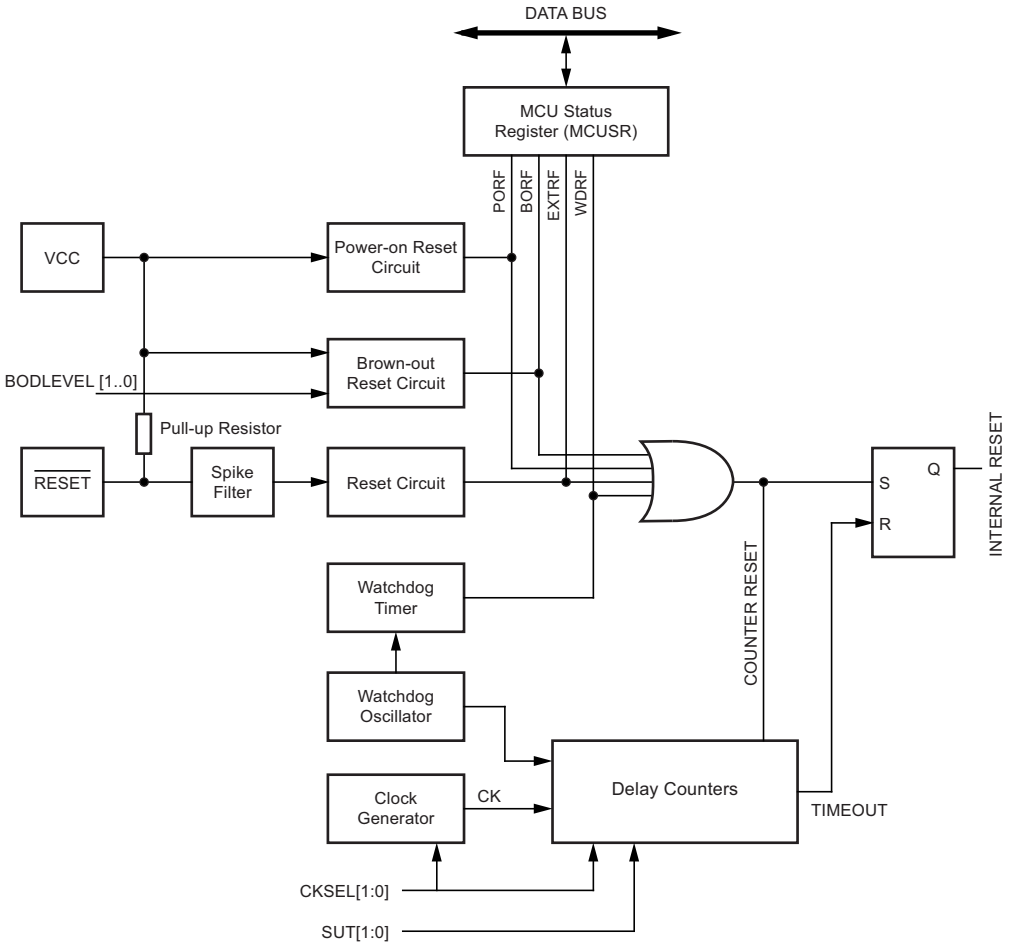
During reset, all I/O registers are set to their initial values, and the program starts execution from the reset vector. The instruction placed at the reset vector must be a RJMP – relative jump – instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 9-1 shows the reset logic. Table 9-1 on page 37 defines the electrical parameters of the reset circuitry. The I/O ports of the Atmel® AVR® are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running. After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 7.2 “Clock Sources” on page 25.

9.2 Reset Sources

The Atmel ATtiny24/44/84 has four sources of reset:

- Power-on reset. The MCU is reset when the supply voltage is below the power-on reset threshold (V_{POT}).
- External reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length when $\overline{\text{RESET}}$ function is enabled.
- Watchdog reset. The MCU is reset when the watchdog timer period expires and the watchdog is enabled.
- Brown-out reset. The MCU is reset when the supply voltage V_{CC} is below the brown-out reset threshold (V_{BOT}) and the brown-out detector is enabled.

Figure 9-1. Reset Logic



9.3 Power-on Reset

A power-on reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in Section 22.4 “System and Reset Characterizations” on page 158. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the start-up reset, as well as to detect a failure in supply voltage.

A POR circuit ensures that the device is reset from power-on. Reaching the power-on reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 9-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

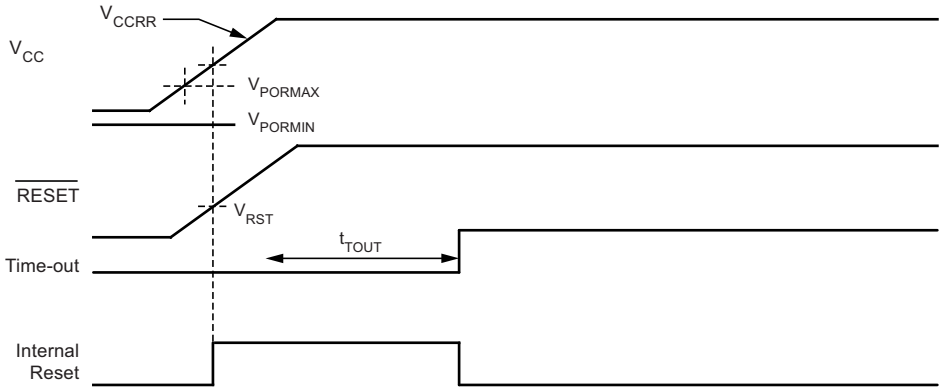


Figure 9-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally

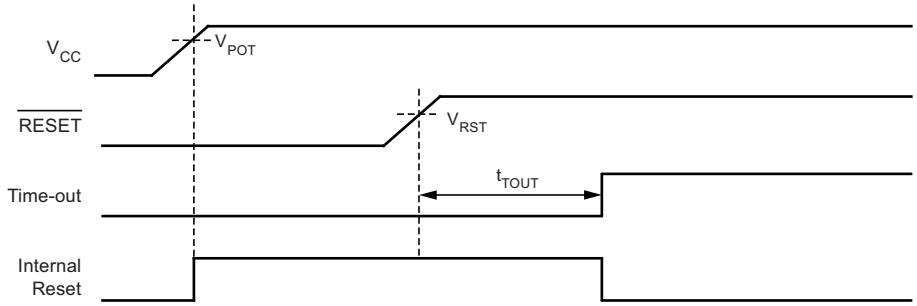


Table 9-1. Power On Reset Specifications

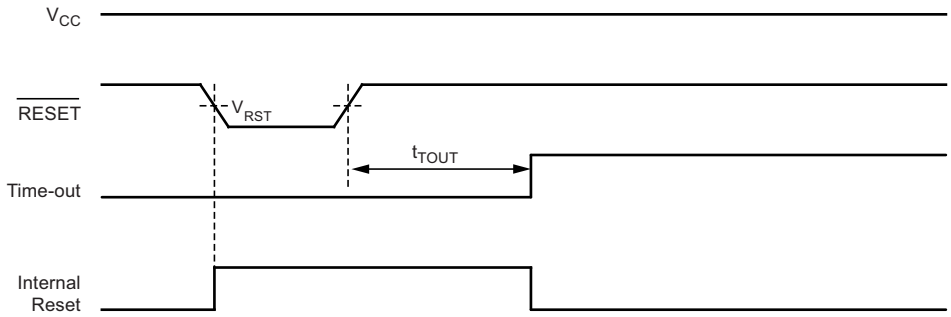
Parameter	Symbol	Min	Typ	Max	Unit
Power-on reset threshold voltage (rising)	V_{POT}	1.1	1.4	1.7	V
Power-on reset threshold voltage (falling)(1)		0.8	1.3	1.6	V
VCC max. start voltage to ensure internal power-on reset signal	V_{PORMAX}			0.4	V
VCC Min. start voltage to ensure internal power-on reset signal	V_{PORMIN}	-0.1			V
VCC rise rate to ensure power-on reset	V_{CCRR}	0.01			V/ms
$\overline{\text{RESET}}$ pin threshold voltage	V_{RST}	0.1 V_{CC}		0.9V_{CC}	V

Note: 1. Before rising, the supply has to be between V_{PORMIN} and V_{PORMAX} to ensure a reset.

9.4 External Reset

An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin if enabled. Reset pulses longer than the minimum pulse width (see Section 22.4 “System and Reset Characterizations” on page 158) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the reset threshold voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the time-out period – t_{TOUT} – has expired.

Figure 9-4. External Reset During Operation



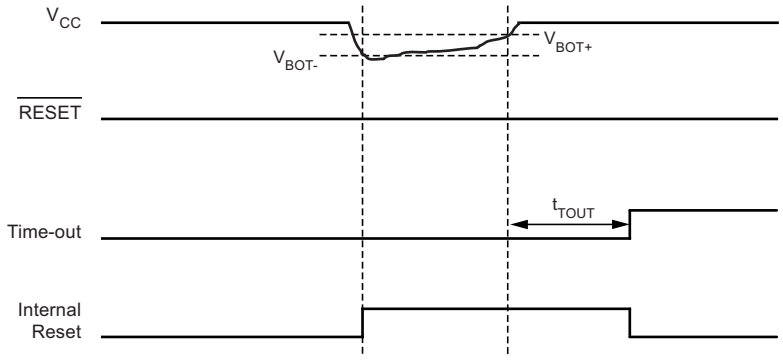
9.5 Brown-out Detection

Atmel® ATtiny24/44/84 has an on-chip brown-out detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL fuses. The trigger level has a hysteresis to ensure spike free brown-out detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} – V_{HYST}/2$.

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-5), the brown-out reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 9-5), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Section 22.4 “System and Reset Characterizations” on page 158.

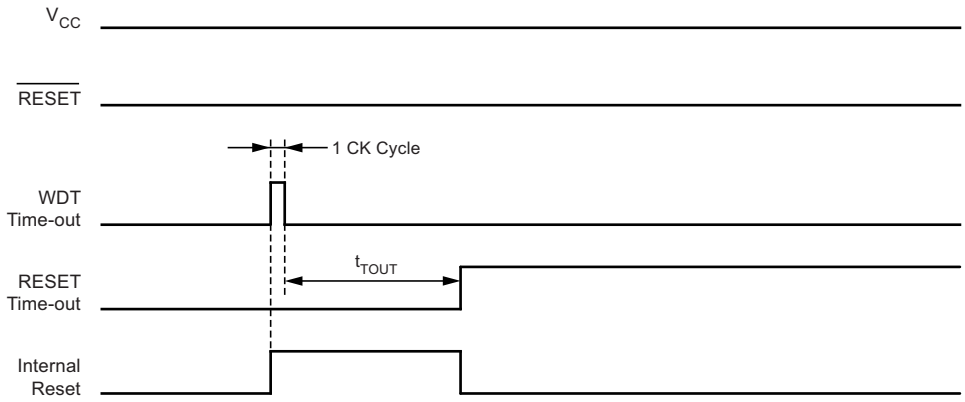
Figure 9-5. Brown-out Reset During Operation



9.6 Watchdog Reset

When the watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the time-out period t_{TOUT}. See Section 9.8 “Watchdog Timer” on page 39 for details on operation of the watchdog timer.

Figure 9-6. Watchdog Reset During Operation



9.7 Internal Voltage Reference

The Atmel® ATtiny24/44/84 features an internal bandgap reference. This reference is used for brown-out detection, and it can be used as an input to the analog comparator or the ADC.

9.7.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Section 22.4 “System and Reset Characterizations” on page 158. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2..0] fuse).
2. When the bandgap reference is connected to the analog comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the analog comparator or ADC is used. To reduce power consumption in power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering power-down mode.

9.8 Watchdog Timer

The watchdog timer is clocked from an on-chip oscillator which runs at 128kHz. By controlling the watchdog timer prescaler, the watchdog reset interval can be adjusted as shown in Table 9-4 on page 43. The WDR – watchdog reset – instruction resets the watchdog timer. The watchdog timer is also reset when it is disabled and when a chip reset occurs. Ten different clock cycle periods can be selected to determine the reset period. If the reset period expires without another watchdog reset, the Atmel ATtiny24/44/84 resets and executes from the reset vector. For timing details on the watchdog reset, refer to Table 9-4 on page 43.

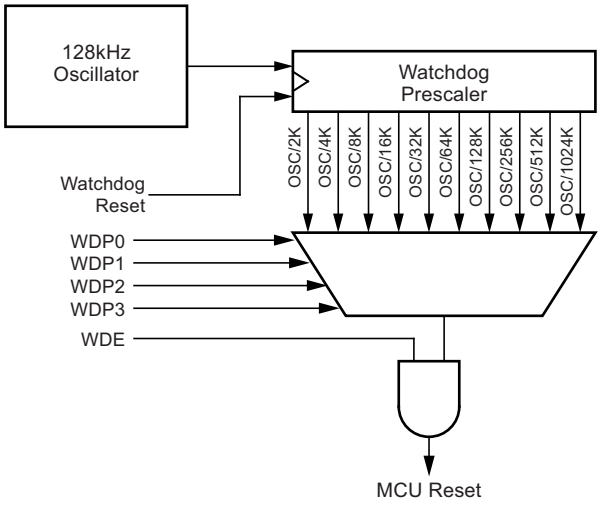
The watchdog timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the watchdog to wake-up from power-down.

To prevent unintentional disabling of the watchdog or unintentional change of time-out period, two different safety levels are selected by the WDTON fuse, as shown in Table 9-2 See Section 9.9 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 40 for details.

Table 9-2. WDT Configuration as a Function of the Fuse Settings of WDTON

WDTON	Safety Level	WDT Initial State	How to Disable the WDT	How to Change Time-out
Unprogrammed	1	Disabled	Timed sequence	No limitations
Programmed	2	Enabled	Always enabled	Timed sequence

Figure 9-7. Watchdog Timer



9.9 Timed Sequences for Changing the Configuration of the Watchdog Timer

The sequence for changing configuration differs slightly between the two safety levels. Separate procedures are described for each level.

9.9.1 Safety Level 1

In this mode, the watchdog timer is initially disabled, but can be enabled by writing the WDE bit to logical one without any restriction. A timed sequence is needed when disabling an enabled watchdog timer. To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDCE and WDE. A logical one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, in the same operation, write the WDE and WDP bits as desired, but with the WDCE bit cleared.

9.9.2 Safety Level 2

In this mode, the watchdog timer is always enabled, and the WDE bit will always read as logical one. A timed sequence is needed when changing the watchdog time-out period. To change the watchdog time-out, the following procedure must be followed:

1. In the same operation, write a logical one to WDCE and WDE. Even though the WDE always is set, the WDE must be written to one to start the timed sequence.
2. Within the next four clock cycles, in the same operation, write the WDP bits as desired, but with the WDCE bit cleared. The value written to the WDE bit is irrelevant.

9.10 Register Description

9.10.1 MCUSR – MCU Status Register

The MCU status register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84 and will always read as zero.

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a watchdog reset occurs. The bit is reset by a power-on reset, or by writing a logical zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a brown-out reset occurs. The bit is reset by a power-on reset, or by writing a logical zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an external reset occurs. The bit is reset by a power-on reset, or by writing a logical zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a power-on reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the reset flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the reset flags.

9.10.2 WDTCSR – Watchdog Timer Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x21 (0x41)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

- **Bit 7 – WDIF: Watchdog Time-out Interrupt Flag**

This bit is set when a time-out occurs in the watchdog timer and the watchdog timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the watchdog time-out interrupt is executed.

- **Bit 6 – WDIE: Watchdog Timeout Interrupt Enable**

When this bit is written to logical one, WDE is cleared, and the I-bit in the status register is set, the watchdog time-out interrupt is enabled. In this mode the corresponding interrupt is executed instead of a reset if a time-out in the watchdog timer occurs.

If WDE is set, WDIE is automatically cleared by hardware when a time-out occurs. This is useful for keeping the watchdog reset security while using the interrupt. After the WDIE bit is cleared, the next time-out will generate a reset. To avoid the watchdog reset, WDIE must be set after each interrupt.

Table 9-3. Watchdog Timer Configuration

WDE	WDIE	Watchdog Timer State	Action on Time-out
0	0	Stopped	None
0	1	Running	Interrupt
1	0	Running	Reset
1	1	Running	Interrupt

- **Bit 4 – WDCE: Watchdog Change Enable**

This bit must be set when the WDE bit is written to logical zero. Otherwise, the watchdog will not be disabled. Once written to logical one, hardware will clear this bit after four clock cycles. See the description of the WDE bit for a watchdog disable procedure. This bit must also be set when changing the prescaler bits. See Section 9.9 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 40.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is written to logical one, the watchdog timer is enabled, and if the WDE is written to logical zero, the watchdog timer function is disabled. WDE can only be cleared if the WDCE bit has logic level one. To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDCE and WDE. A logical one must be written to WDE even though it is set to logical one before the disable operation starts.
2. Within the next four clock cycles, write a logical zero to WDE. This disables the watchdog.

In safety level 2, it is not possible to disable the watchdog timer, even with the algorithm described above. See Section 9.9 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 40.

In safety level 1, WDE is overridden by WDRF in MCUSR. See Section 9.10.1 “MCUSR – MCU Status Register” on page 41 for description of WDRF. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared before disabling the Watchdog with the procedure described above. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

Note: If the watchdog timer is not going to be used in the application, it is important to go through a watchdog disable procedure in the initialization of the device. If the watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset, which in turn will lead to a new watchdog reset. To avoid this situation, the application software should always clear the WDRF flag and the WDE control bit in the initialization routine.

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

The WDP3..0 bits determine the watchdog timer prescaling when the watchdog timer is enabled. The different prescaling values and their corresponding time-out periods are shown in Table 9-4 on page 43.

Table 9-4. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2Kcycles	16ms
0	0	0	1	4Kcycles	32ms
0	0	1	0	8Kcycles	64ms
0	0	1	1	16Kcycles	0.125s
0	1	0	0	32Kcycles	0.25s
0	1	0	1	64Kcycles	0.5s
0	1	1	0	128Kcycles	1.0s
0	1	1	1	256Kcycles	2.0s
1	0	0	0	512Kcycles	4.0s
1	0	0	1	1024Kcycles	8.0s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

The following code example shows one assembly function and one C function for turning off the WDT. The example assumes that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example[1]

```
WDT_off:
WDR
        ; Clear WDRF in MCUSR
        ldi    r16, (0<<WDRF)
        out    MCUSR, r16
        ; Write logical one to WDCE and WDE
        ; Keep old prescaler setting to prevent unintentional Watchdog
Reset
        in      r16, WDTCR
        ori     r16, (1<<WDCE)|(1<<WDE)
        out     WDTCR, r16
        ; Turn off WDT
        ldi     r16, (0<<WDE)
        out     WDTCR, r16
        ret
```

C Code Example[1]

```
void WDT_off(void)
{
        _WDR();
        /* Clear WDRF in MCUSR */
        MCUSR = 0x00
        /* Write logical one to WDCE and WDE */
        WDTCR |= (1<<WDCE) | (1<<WDE);
        /* Turn off WDT */
        WDTCR = 0x00;
}
```

Note: 1. See Section 4. “About Code Examples” on page 8.

10. Interrupts

This section describes the specifics of the interrupt handling as performed in Atmel® ATtiny24/44/84. For a general explanation of the AVR® interrupt handling, see Section 5.8 “Reset and Interrupt Handling” on page 14.

10.1 Interrupt Vectors

Table 10-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset, watchdog reset
2	0x0001	INT0	External interrupt request 0
3	0x0002	PCINT0	Pin change interrupt request 0
4	0x0003	PCINT1	Pin change interrupt request 1
5	0x0004	WDT	Watchdog time-out
6	0x0005	TIMER1 CAPT	Timer/Counter1 capture event
7	0x0006	TIMER1 COMPA	Timer/Counter1 compare match A
8	0x0007	TIMER1 COMPB	Timer/Counter1 compare match B
9	0x0008	TIMER1 OVF	Timer/Counter0 overflow
10	0x0009	TIMER0 COMPA	Timer/Counter0 compare match A
11	0x000A	TIMER0 COMPB	Timer/Counter0 compare match B
12	0x000B	TIMER0 OVF	Timer/Counter0 overflow
13	0x000C	ANA_COMP	Analog comparator
14	0x000D	ADC	ADC conversion complete
15	0x000E	EE_RDY	EEPROM ready
16	0x000F	USI_START	USI START
17	0x0010	USI_OVF	USI overflow

If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The most typical and general program setup for the reset and interrupt vector Addresses in Atmel® ATtiny24/44/84 is:

```
Address      Labels Code                 Comments
0x0000              rjmp   RESET        ; Reset Handler
0x0001              rjmp   EXT_INT0     ; IRQ0 Handler
0x0002              rjmp   PCINT0       ; PCINT0 Handler
0x0003              rjmp   PCINT1       ; PCINT1 Handler
0x0004              rjmp   WATCHDOG     ; Watchdog Interrupt Handler
0x0005              rjmp   TIM1_CAPT    ; Timer1 Capture Handler
0x0006              rjmp   TIM1_COMPA   ; Timer1 Compare A Handler
0x0007              rjmp   TIM1_COMPB   ; Timer1 Compare B Handler
0x0008              rjmp   TIM1_OVF     ; Timer1 Overflow Handler
0x0009              rjmp   TIM0_COMPA   ; Timer0 Compare A Handler
0x000A              rjmp   TIM0_COMPB   ; Timer0 Compare B Handler
0x000B              rjmp   TIM0_OVF     ; Timer0 Overflow Handler
0x000C              rjmp   ANA_COMP     ; Analog Comparator Handler
0x000D              rjmp   ADC          ; ADC Conversion Handler
0x000E              rjmp   EE_RDY       ; EEPROM Ready Handler
0x000F              rjmp   USI_STR      ; USI STart Handler
0x0010              rjmp   USI_OVF      ; USI Overflow Handler
;
0x0011       RESET: ldi    r16, high(RAMEND); Main program start
0x0012              out    SPH,r16      ; Set Stack Pointer to top of RAM
0x0013              ldi    r16, low(RAMEND)
0x0014              out    SPL,r16
0x0015              sei                 ; Enable interrupts
0x0016              <instr> xxx
...    ...    ...    ...
```

11. External Interrupts

The external interrupts are triggered by the INT0 pin or any of the PCINT11..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT11..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. Pin change 0 interrupts (PCI0) will trigger if any enabled PCINT7..0 pin toggles. Pin change 1 interrupts (PCI1) will trigger if any enabled PCINT11..8 pin toggles. The PCMSK0 and PCMSK1 registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT11..0 are detected asynchronously. This implies that these interrupts also can be used for waking the part from sleep modes other than idle mode.

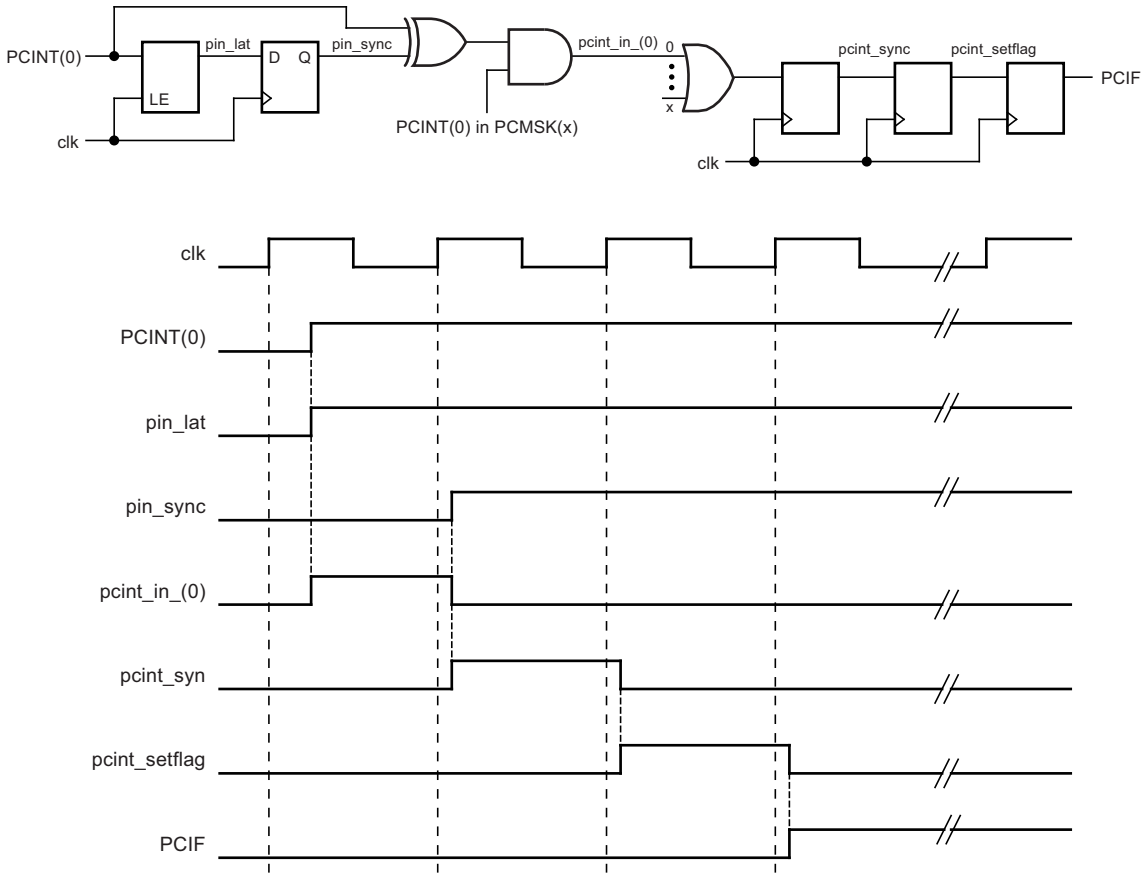
The INT0 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU control register (MCUCR). When the INT0 interrupt is enabled and is configured as level-triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling- or rising-edge interrupts on INT0 requires the presence of an I/O clock, described in Section 7.1 “Clock Systems and their Distribution” on page 24. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt also can be used for waking the part 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 Section 7. “System Clock and Clock Options” on page 24.

11.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 11-1.

Figure 11-1. Timing of Pin Change Interrupts



11.2 Register Description

11.2.1 MCUCR – MCU Control Register

The external interrupt control register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The external interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 11-1. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 11-1. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

11.2.2 GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	–	INT0	PCIE1	PCIE0	–	–	–	–	GIMSK
Read/Write	R	R/W	R/W	R/w	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 3..0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84 and will always read as zero.

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the status register (SREG) is set (one), the external pin interrupt is enabled. The interrupt sense control0 bits 1/0 (ISC01 and ISC00) in the external interrupt control register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of external interrupt request 0 is executed from the INT0 interrupt vector.

- **Bit 5 – PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT11..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT11..8 pins are enabled individually by the PCMSK1 register.

- **Bit 4– PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set and the I-bit in the status register (SREG) is set, pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 interrupt vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

11.2.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A	–	INTF0	PCIF1	PCIF0	–	–	–	–	GIFR
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 3..0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84 and will always read as zero.

- **Bit 6 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (logical one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

- **Bit 5 – PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT11..8 pin triggers an interrupt request, PCIF1 becomes set (logical one). If the I-bit in SREG and the PCIE1 bit in GIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 4– PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF becomes set (logical one). If the I-bit in SREG and the PCIE0 bit in GIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

11.2.4 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	–	–	–	–	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 4– Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny24/44/84 and will always read as zero.

- **Bits 3..0 – PCINT11..8: Pin Change Enable Mask 11..8**

Each PCINT11..8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT11..8 is set (logical one) and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT11..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.2.5 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

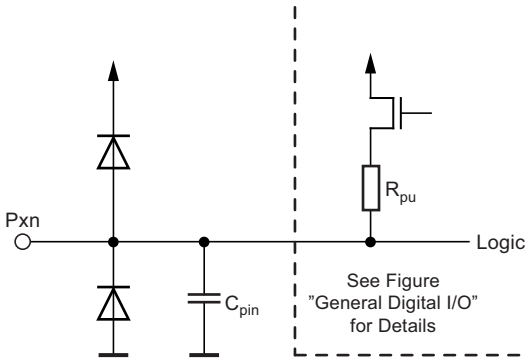
Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set (logical one) and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

12. I/O Ports

12.1 Overview

All Atmel® AVR® ports have true read-modify-write functionality when used as general digital I/O ports. This means that the SBI and CBI instructions can be used to change direction of one port pin without unintentionally changing the direction of any other pin. 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 VCC and ground as indicated in Figure 12-1. See Section 22. “Electrical Characteristics” on page 155 for a complete list of parameters.

Figure 12-1. 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. The physical I/O registers and bit locations are listed in “EXT_CLOCK = external clock is selected as system clock.” on page 61.

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 a logical one 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 the MCUCR disables the pull-up function for all pins in all ports when set.

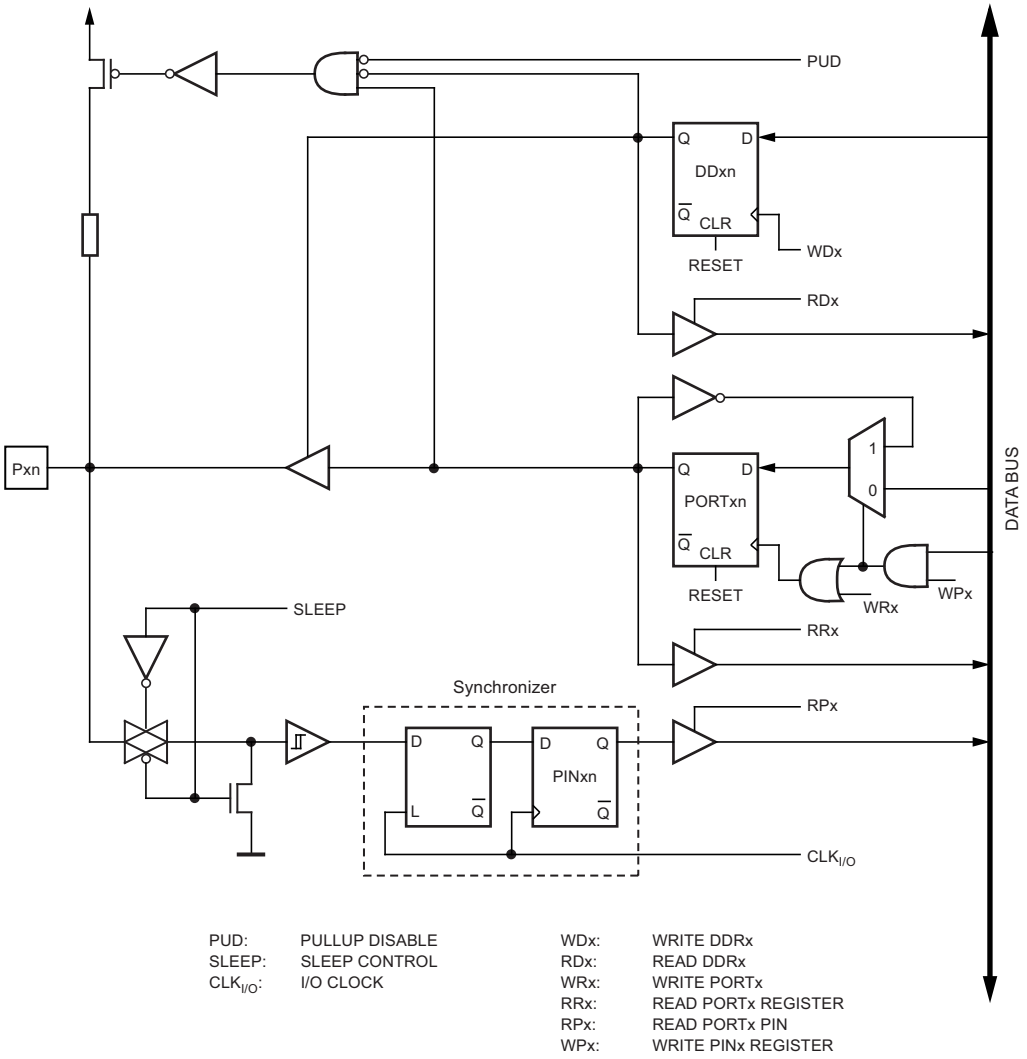
Using the I/O port as general digital I/O is described in Section 12.2 “Ports as General Digital I/O” on page 50. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 12.3 “Alternate Port Functions” on page 54. Refer to the individual module sections for a full description of the alternate functions.

Note that 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.

12.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 12-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 12-2. General Digital I/O(1)



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.

12.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in “EXT_CLOCK = external clock is selected as system clock.” on page 61, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx register selects the direction of this pin. If DDxn is written logical one, Pxn is configured as an output pin. If DDxn is written logical zero, Pxn is configured as an input pin.

If PORTxn is written logical one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logical zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written logical one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logical zero when the pin is configured as an output pin, the port pin is driven low (zero).

12.2.2 Toggling the Pin

Writing a logical one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

12.2.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedance environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step.

Table 12-1 summarizes the control signals for the pin value.

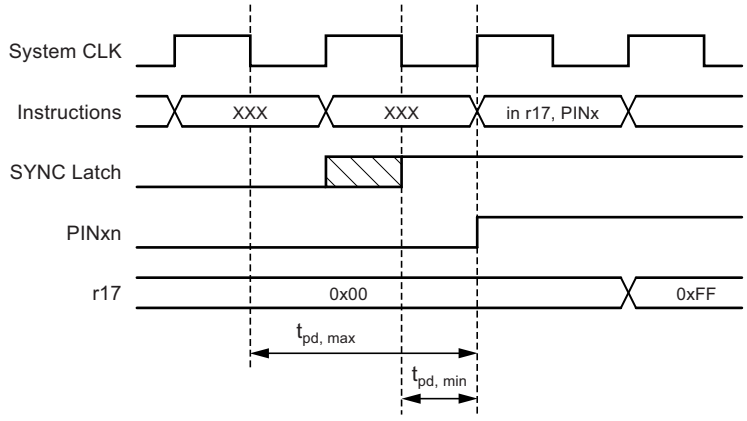
Table 12-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output low (sink)
1	1	X	Output	No	Output high (source)

12.2.4 Reading the Pin Value

Independent of the setting of data direction bit DDxn, the port pin can be read through the PINxn register bit. As shown in Figure 12-2 on page 50, the PINxn register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 12-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

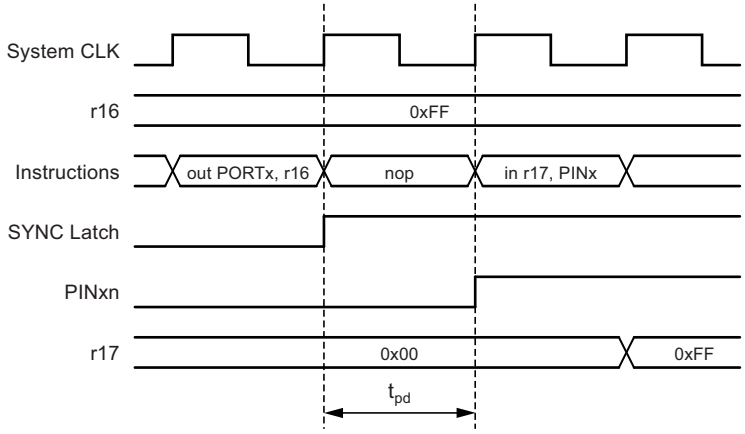
Figure 12-3. Synchronization when Reading an Externally Applied Pin Value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn register at the succeeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 12-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay t_{pd} through the synchronizer is one system clock period.

Figure 12-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port A pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 5 as input with a pull-up assigned to port pin 4. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi    r16,(1<<PA4)|(1<<PA1)|(1<<PA0)
ldi    r17,(1<<DDA3)|(1<<DDA2)|(1<<DDA1)|(1<<DDA0)
out    PORTA,r16
out    DDRA,r17
; Insert nop for synchronization
nop
; Read port pins
in     r16,PINA
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTA = (1<<PA4)|(1<<PA1)|(1<<PA0);
DDRA = (1<<DDA3)|(1<<DDA2)|(1<<DDA1)|(1<<DDA0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINA;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1 and 4, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

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

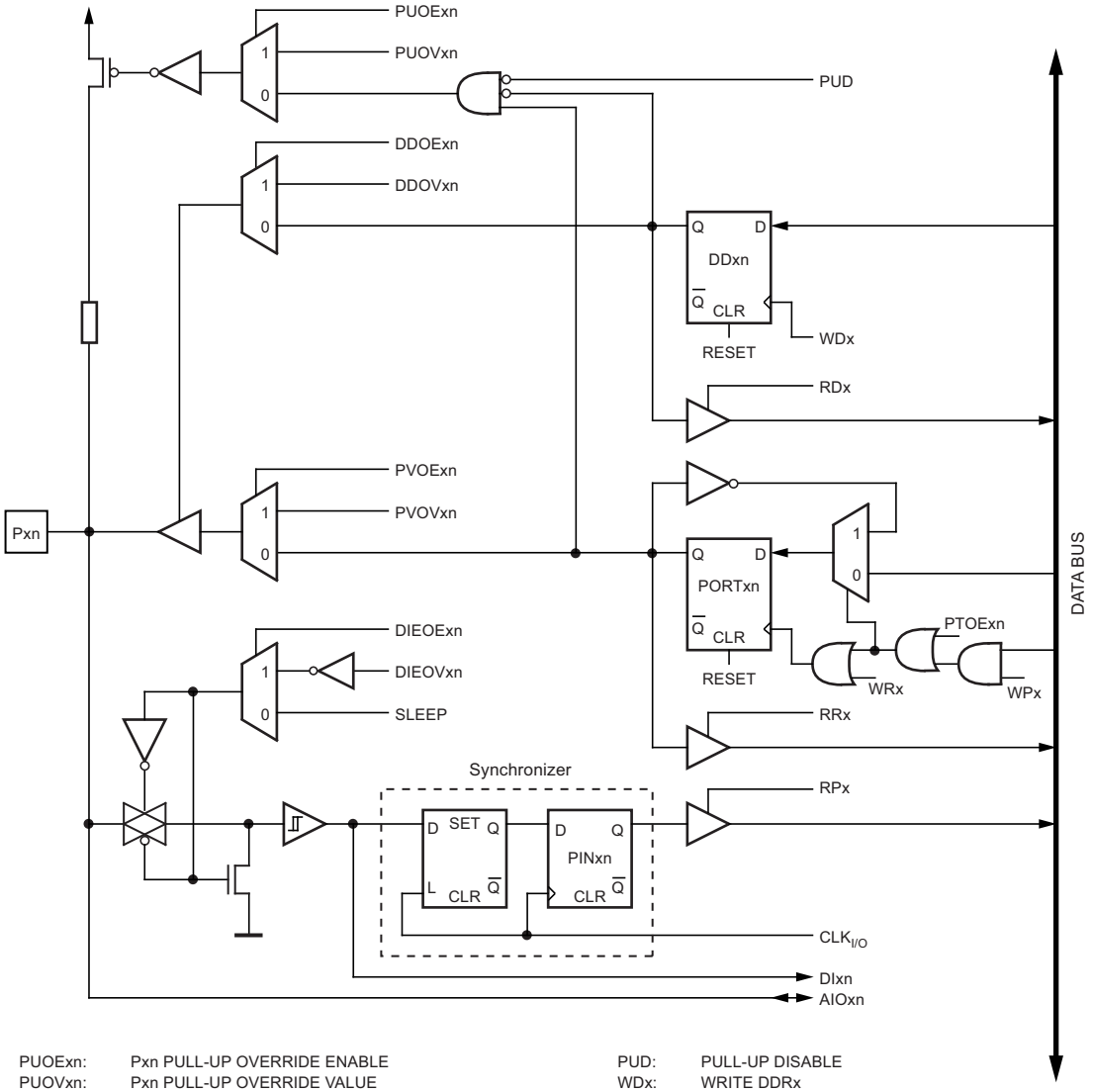
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

12.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 12-5 shows how the port pin control signals from the simplified Figure 12-2 on page 50 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the Atmel® AVR® microcontroller family.

Figure 12-5. Alternate Port Functions(1)



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULL-UP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT ENABLE OVERRIDE VALUE	$CLK_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	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.

Table 12-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 12-5 on page 54 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 12-2. 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.

12.3.1 Alternate Functions of Port A

The port A pins with alternate function are shown in Table 12-7 on page 59.

Table 12-3. Port A Pins Alternate Functions

Port Pin	Alternate Function	
PA0	ADC0:	ADC input channel 0.
	AREF:	External analog reference.
	PCINT0:	Pin change interrupt 0 source 0.
PA1	ADC1:	ADC input channel 1.
	AIN0:	Analog comparator positive input.
	PCINT1:	Pin change interrupt 0 source 1.
PA2	ADC2:	ADC input channel 2.
	AIN1:	Analog comparator negative input.
	PCINT2:	Pin change interrupt 0 source 2.
PA3	ADC3:	ADC input channel 3.
	T0:	Timer/Counter0 counter source.
	PCINT3:	Pin change interrupt 0 source 3.
PA4	ADC4:	ADC input channel 4.
	USCK:	USI Clock three wire mode.
	SCL:	USI Clock two wire mode.
	T1:	Timer/Counter1 counter source.
	PCINT4:	Pin change interrupt 0 source 4.
PA5	ADC5:	ADC input channel 5.
	DO:	USI data output three wire mode.
	OC1B:	Timer/Counter1 compare match B output.
	PCINT5:	Pin change interrupt 0 source 5.
PA6	ADC6:	ADC input channel 6.
	DI:	USI data input three wire mode.
	SDA:	USI data input two wire mode.
	OC1A:	Timer/Counter1 compare match A output.
	PCINT6:	Pin change interrupt 0 source 6.
PA7	ADC7:	ADC input channel 7.
	OC0B:	Timer/Counter0 compare match B output.
	ICP1:	Timer/Counter1 input capture pin.
	PCINT7:	Pin change interrupt 0 source 7.

- **Port A, Bit 0 – ADC0/AREF/PCINT0**

ADC0: Analog to digital converter, channel 0.

AREF: External analog reference for ADC. Pull up and output driver are disabled on PA0 when the pin is used as an external reference or internal voltage reference with external capacitor at the AREF pin by setting (one) the bit REFS0 in the ADC multiplexer selection register (ADMUX).

PCINT0: Pin change interrupt source 0. The PA0 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 1 – ADC1/AIN0/PCINT1**

ADC1: Analog to digital converter, channel 1.

AIN0: Analog comparator positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

PCINT1: Pin change interrupt source 1. The PA1 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 2 – ADC2/AIN1/PCINT2**

ADC2: Analog to digital converter, channel 2.

AIN1: Analog comparator negative input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

PCINT2: Pin change interrupt source 2. The PA2 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 3 – ADC3/T0/PCINT3**

ADC3: Analog to digital converter, channel 3.

T0: Timer/Counter 0 counter source.

PCINT3: Pin change interrupt source 3. The PA3 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 4 – ADC4/USCK/SCL/T1/PCINT4**

ADC4: Analog to digital converter, channel 4.

USCK: Three-wire mode universal serial interface clock.

SCL: Two-wire mode serial clock for USI two-wire mode.

T1: Timer/Counter1 counter source.

PCINT4: Pin change interrupt source 4. The PA4 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 5 – ADC5/DO/OC1B/PCINT5**

ADC5: Analog to digital converter, channel 5.

DO: Data output in USI three-wire mode. Data output (DO) overrides PORTA5 value, and it is driven to the port when the data direction bit DDA5 is set (one). However the PORTA5 bit still controls the pull-up, enabling pull-up if direction is input and PORTA5 is set (one).

OC1B: Output compare match output: The PA5 pin can serve as an external output for the Timer/Counter1 compare match B. The PA5 pin has to be configured as an output (DDA5 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT5: Pin change interrupt source 5. The PA5 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 6 – ADC6/DI/SDA/OC1A/PCINT6**

ADC6: Analog to digital converter, channel 6.

SDA: Two-wire mode serial interface data.

DI: Data input in USI three-wire mode. USI three-wire mode does not override normal port functions, so pin must be configure as an input for DI function.

OC1A: Output compare match output: The PA6 pin can serve as an external output for the Timer/Counter1 compare match A. The PA6 pin has to be configured as an output (DDA6 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

PCINT6: Pin change interrupt source 6. The PA6 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 7 – ADC7/OC0B/ICP1/PCINT7**

ADC7: Analog to digital converter, channel 7.

OC1B: Output compare match output: The PA7 pin can serve as an external output for the Timer/Counter1 compare match B. The PA7 pin has to be configured as an output (DDA7 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

ICP1: Input capture pin: The PA7 pin can act as an input capture pin for Timer/Counter1.

PCINT7: Pin change interrupt source 7. The PA7 pin can serve as an external interrupt source for pin change interrupt 0.

Table 12-4 to Table 12-6 on page 59 relate the alternate functions of Port A to the overriding signals shown in Figure 12-5 on page 54.

Table 12-4. Overriding Signals for Alternate Functions in PA7..PA5

Signal Name	PA7/ADC7/OC0B/ICP1/ PCINT7	PA6/ADC6/DI/SDA/OC1A/ PCINT6	PA5/ADC5/DO/OC1B/ PCINT5
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	USIWM1	0
DDOV	0	($\overline{SDA}$ + $\overline{PORTA6}$) × DDRA6	0
PVOE	OC0B enable	(USIWM1 × DDA6) + OC1A enable	($\overline{USIWM1}$ × USIWM0) + OC1B enable
PVOV	OC0B	$\overline{(USIWM1 \times \overline{DDA6})}$ × OC1A	$\overline{USIWM1}$ × USIWM0 × DO + (~USIWM1 × $\overline{USIWM0}$) × OC1B}
PTOE	0	0	0
DIEOE	PCINT7 × PCIE0 + ADC7D	USISIE + (PCINT6 × PCIE0) + ADC6D	PCINT5 × PCIE + ADC5D
DIEOV	PCINT7 × PCIE0	USISIE + PCINT7 × PCIE0	PCINT5 × PCIE
DI	PCINT7/ICP1 input	DI/SDA/PCINT6 input	PCINT5 input
AIO	ADC7 input	ADC6 input	ADC5 input

Table 12-5. Overriding Signals for Alternate Functions in PA4..PA2

Signal Name	PA4/ADC4/USCK/SCL/T1/PCINT4	PA3/ADC3/T0/PCINT3	PA2/ADC2/AIN1/PCINT2
PUOE	0	0	0
PUOV	0	0	0
DDOE	USIWM1	0	0
DDOV	USI_SCL_HOLD + $\overline{PORTA4}$) × ADC4D	0	0
PVOE	USIWM1 × ADC4D	0	0
PVOV	0	0	0
PTOE	USI_PTOE	0	0
DIEOE	USISIE + (PCINT4 × PCIE0) + ADC4D	(PCINT3 × PCIE0) + ADC3D	PCINT2 × PCIE + ADC2D
DIEOV	USISIE + (PCINT4 × PCIE0)	PCINT3 × PCIE0	PCINT3 × PCIE0
DI	USCK/SCL/T1/PCINT4 input	PCINT1 input	PCINT0 input
AIO	ADC4 input	ADC3 input	ADC2/analog comparator negative input

Table 12-6. Overriding Signals for Alternate Functions in PA1..PA0

Signal Name	PA1/ADC1/AIN0/PCINT1	PA0/ADC0/AREF/PCINT0
PUOE	0	RESET × (REFS1 × REFS0 + REFS1 × REFS0)
PUOV	0	0
DDOE	0	RESET × ($\overline{\text{REFS1}}$ × REFS0 + REFS1 × REFS0)
DDOV	0	0
PVOE	0	RESET × ($\overline{\text{REFS1}}$ × REFS0 + REFS1 × REFS0)
PVOV	0	0
PTOE	0	0
DIEOE	PCINT1 × PCIE0 + ADC1D	PCINT0 × PCIE0 + ADC0D
DIEOV	PCINT1 × PCIE0	PCINT0 × PCIE0
DI	PCINT1 input	PCINT0 input
AIO	ADC1/analog comparator positive input	ADC1 input analog reference

12.3.2 Alternate Functions of Port B

The port B pins with alternate function are shown in Table 12-7.

Table 12-7. Port B Pins Alternate Functions

Port Pin	Alternate Function	
PB0	XTAL1:	Crystal oscillator input.
	PCINT8:	Pin change interrupt 1 source 8.
PB1	XTAL2:	Crystal oscillator output.
	PCINT9:	Pin change interrupt 1 source 9.
PB2	INT0:	External interrupt 0 input.
	OC0A:	Timer/Counter0 compare match A output.
	CKOUT:	System clock output.
	PCINT10:	Pin change interrupt 1 source 10.
PB3	RESET:	Reset pin.
	dW:	debugWire I/O.
	PCINT11:	Pin change interrupt 1 source 11.

- **Port B, Bit 0 – XTAL1/PCINT8**

XTAL1: Chip clock oscillator pin 1. Used for all chip clock sources except the internal calibrated RC oscillator. When used as a clock pin, the pin cannot be used as an I/O pin. When using internal calibrated RC oscillator as a chip clock source, PB0 serves as an ordinary I/O pin.

PCINT8: Pin change interrupt source 8. The PB0 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 1 – XTAL2/PCINT9**

XTAL2: Chip clock oscillator pin 2. Used as clock pin for all chip clock sources except the internal calibrated RC oscillator and external clock. When used as a clock pin, the pin cannot be used as an I/O pin. When using internal calibrated RC oscillator or external clock as a chip clock sources, PB1 serves as an ordinary I/O pin.

PCINT9: Pin change interrupt source 9. The PB1 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 2 – INT0/OC0A/CKOUT/PCINT10**

INT0: External interrupt request 0.

OC0A: Output compare match output: The PB2 pin can serve as an external output for the Timer/Counter 0 compare match A. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.

CKOUT - System clock output: The system clock can be output on the PB2 pin. The system clock will be output if the CKOUT fuse is programmed, regardless of the PORTB2 and DDB2 settings. It will also be output during reset.

PCINT10: Pin change interrupt source 10. The PB2 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 3 – $\overline{\text{RESET}}$/dW/PCINT11**

$\overline{\text{RESET}}$: External $\overline{\text{reset}}$ input is active low and enabled by un-programming (“1”) the RST- DISBL fuse. Pull-up is activated and output driver and digital input are deactivated when the pin is used as the $\overline{\text{RESET}}$ pin.

dW: When the debugWIRE enable (DWEN) fuse is programmed and lock bits are un-programmed, 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.

PCINT11: Pin change interrupt source 11. The PB3 pin can serve as an external interrupt source for pin change interrupt 1.

Table 12-8 and Table 12-9 on page 61 relate the alternate functions of Port B to the overriding signals shown in Figure 12-5 on page 54.

Table 12-8. Overriding Signals for Alternate Functions in PB3..PB2

Signal Name	PB3/$\overline{\text{RESET}}$/dW/PCINT11	PB2/INT0/OC0A/CKOUT/PCINT10
PUOE	$\overline{\text{RSTDISBL}}^{(1)}$+ DEBUGWIRE_ENABLE$^{(2)}$	CKOUT
PUOV	1	0
DDOE	$\overline{\text{RSTDISBL}}^{(1)}$ + DEBUGWIRE_ENABLE$^{(2)}$	CKOUT
DDOV	DEBUGWIRE_ENABLE$^{(2)}$ × $\overline{\text{debugWire Transmit}}$	1'b1
PVOE	$\overline{\text{RSTDISBL}}^{(1)}$ + DEBUGWIRE_ENABLE$^{(2)}$	CKOUT + OC0A enable
PVOV	0	CKOUT × System Clock + $\overline{\text{CKOUT}}$ × OC0A
PTOE	0	0
DIEOE	RSTDISBL$^{(1)}$ + DEBUGWIRE_ENABLE$^{(2)}$ + PCINT11 × PCIE1	PCINT10 × PCIE1 + INT0
DIEOV	DEBUGWIRE_ENABLE$^{(2)}$ + (RSTDISBL$^{(1)}$ × PCINT11 × PCIE1)	PCINT10 × PCIE1 + INT0
DI	dW/PCINT11 Input	INT0/PCINT10 Input
AIO		

Notes: 1. RSTDISBL is 1 when the fuse is “0” (programmed).
2. DebugWIRE is enabled when DWEN fuse is programmed and lock bits are un-programmed.

Table 12-9. Overriding Signals for Alternate Functions in PB1..PB0

Signal Name	PB1/XTAL2/PCINT9	PB0/XTAL1/PCINT8
PUOE	EXT_OSC(1)	EXT_CLOCK(2)+ EXT_OSC(1)
PUOV	0	0
DDOE	EXT_OSC(1)	EXT_CLOCK(2) + EXT_OSC(1)
DDOV	0	0
PVOE	EXT_OSC(1)	EXT_CLOCK(2) + EXT_OSC(1)
PVOV	0	0
PTOE	0	0
DIEOE	EXT_OSC (1)+ PCINT9 × PCIE1	EXT_CLOCK(2) + EXT_OSC(1) + (PCINT8 × PCIE1)
DIEOV	EXT_OSC(1) × PCINT9 × PCIE1	(EXT_CLOCK(2) × $\overline{\text{PWR\_DOWN}}$) + ($\overline{\text{EXT\_CLOCK}}$(2) × $\overline{\text{EXT\_OSC}}$(1) × PCINT8 × PCIE1)
DI	PCINT9 input	CLOCK/PCINT8 input
AIO	XTAL2	XTAL1

Notes: 1. EXT_OSC = crystal oscillator or low frequency crystal oscillator is selected as system clock.
2. EXT_CLOCK = external clock is selected as system clock.

12.4 Register Description

12.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – PUD: Pull-up Disable**

When this bit is written to logical one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See Section 12.2.1 “Configuring the Pin” on page 50 for more details about this feature.

12.4.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

12.4.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRB
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	

12.4.4 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x19 (0x39)	**PINA7**	**PINA6**	**PINA5**	**PINA4**	**PINA3**	**PINA2**	**PINA1**	**PINA0**	**PINB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	N/A	N/A	N/A	N/A	N/A	

12.4.5 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	–	–			**PORTB3**	**PORTB2**	**PORTB1**	**PORTB0**	**PORTB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.4.6 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	–	–			**DDB3**	**DDB2**	**DDB1**	**DDB0**	**DDRB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.4.7 PINB – Port BInput Pins Address

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–			**PINB3**	**PINB2**	**PINB1**	**PINB0**	**PINB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	N/A	N/A	N/A	N/A	N/A	

13. 8-bit Timer/Counter0 with PWM

13.1 Features

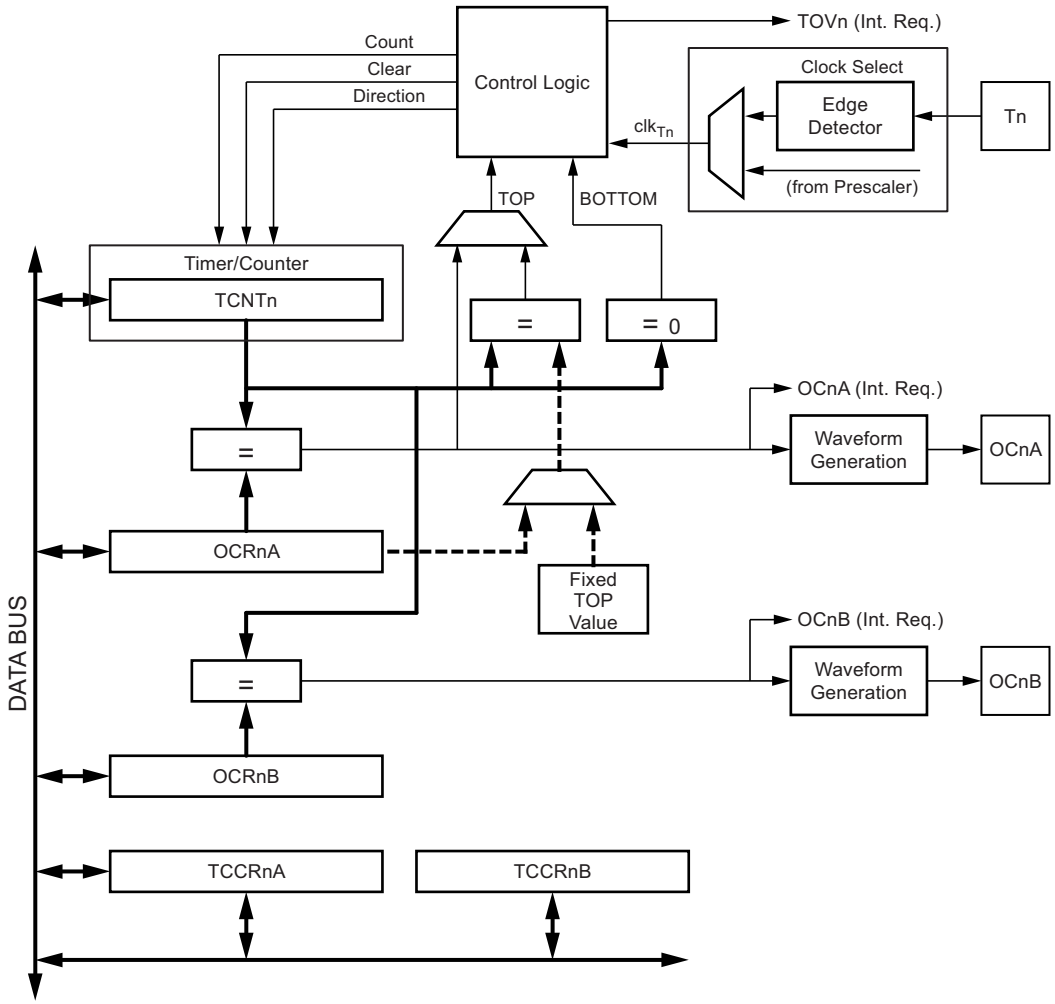
- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

13.2 Overview

Timer/Counter 0 is a general purpose 8-bit Timer/Counter module, with two independent out- put compare units, and with PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 13-1. For the actual placement of I/O pins, refer to Figure 1-1 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 13.9 “Register Description” on page 73.

Figure 13-1. 8-bit Timer/Counter Block Diagram



13.2.1 Registers

The Timer/Counter (TCNT0) and output compare registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer/Counter 0 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 uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T0}).

The double buffered output compare registers (OCR0A and OCR0B) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins (OC0A and OC0B). See Section 13.5 “Output Compare Unit” on page 65 for details. The compare match event will also set the compare flag (OCF0A or OCF0B) which can be used to generate an output compare interrupt request.

13.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the output compare unit, in this case compare unit A or compare unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter 0 counter value and so on.

The definitions in Table are also used extensively throughout the document.

Table 13-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
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 the fixed value 0xFF (MAX) or the value stored in the OCR0A register. The assignment is dependent on the mode of operation.

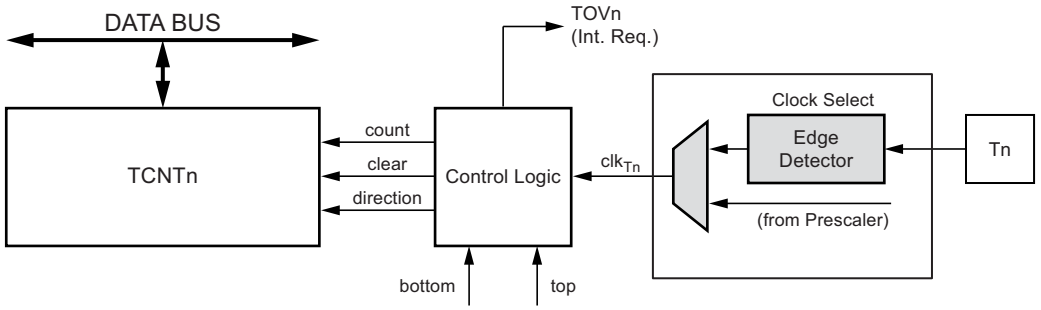
13.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter control register (TCCR0B). For details on clock sources and prescaler, see Section 15. “Timer/Counter Prescaler” on page 103.

13.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 13-2 shows a block diagram of the counter and its surroundings.

Figure 13-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the clock select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter control register A (TCCR0A) and the WGM02 bit located in the Timer/Counter control register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare output (OC0A). For more details about advanced counting sequences and waveform generation, see Section 13.7 “Modes of Operation” on page 67.

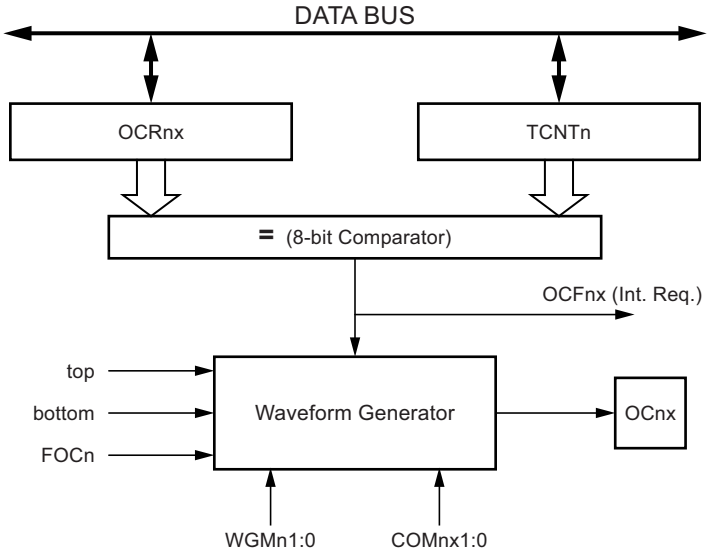
The Timer/Counter overflow flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

13.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the output compare flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to the operating mode set by the WGM02:0 bits and compare output mode (COM0x1:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation. See Section 13.7 “Modes of Operation” on page 67.

Figure 13-3 shows a block diagram of the output compare unit.

Figure 13-3. Output Compare Unit, Block Diagram



The OCR0x registers are double buffered when using any of the pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x compare registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

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

13.5.1 Force Output Compare

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

13.5.2 Compare Match Blocking by TCNT0 Write

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

13.5.3 Using the Output Compare Unit

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

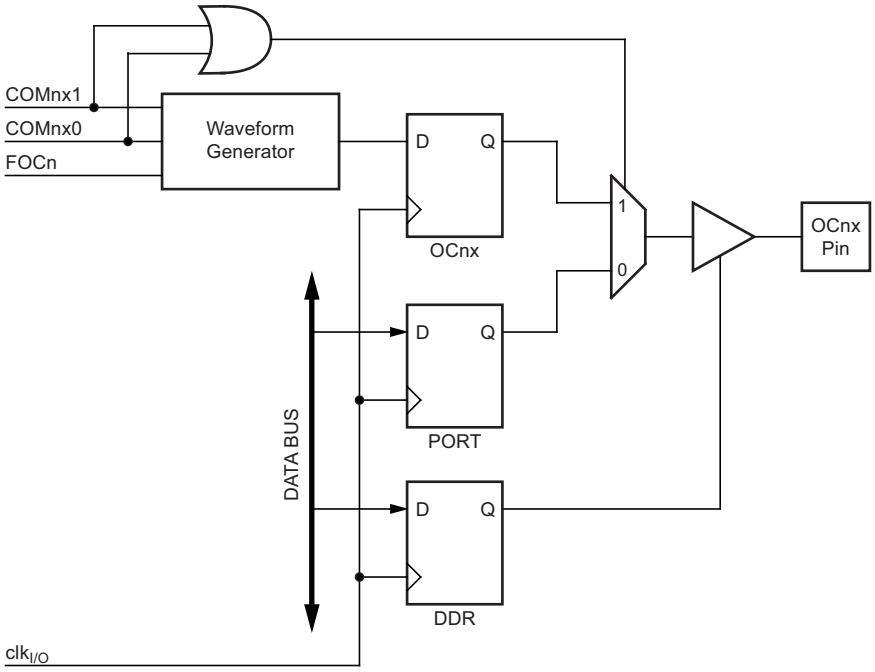
The setup of the OC0x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC0x value is to use the force output compare (0x) strobe bits in normal mode. The OC0x registers keep their values even when changing between waveform generation modes.

Be aware that the COM0x1:0 bits are not double buffered together with the compare value. Changes to the COM0x1:0 bits will take effect immediately.

13.6 Compare Match Output Unit

The compare output mode (COM0x1:0) bits have two functions. The waveform generator uses the COM0x1:0 bits for defining the output compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 13-4 on page 67 shows a simplified schematic of the logic affected by the COM0x1:0 bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x register, not the OC0x pin. If a system reset occurs, the OC0x register is reset to “0”.

Figure 13-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC0x) from the waveform generator if either of the COM0x1:0 bits are set. However, the OC0x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC0x pin (DDR_OC0x) must be set as output before the OC0x value is visible on the pin. The port override function is independent of the waveform generation mode.

The design of the output compare pin logic allows initialization of the OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation, see Section 13.9 “Register Description” on page 73.

13.6.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM0x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0x1:0 = 0 tells the waveform generator that no action on the OC0x register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 13-2 on page 73. For fast PWM mode, refer to Table 13-3 on page 73, and for phase correct PWM refer to Table 13-4 on page 73.

A change of the COM0x1:0 bit states will have 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 0x strobe bits.

13.7 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 (WGM02:0) and compare output mode (COM0x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM0x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (See Section 13.7 “Modes of Operation” on page 67).

For detailed timing information refer to Figure 13-8 on page 71, Figure 13-9 on page 72, Figure 13-10 on page 72 and Figure 13-11 on page 72 in Section 13.8 “Timer/Counter Timing Diagrams” on page 71.

13.7.1 Normal Mode

The simplest mode of operation is the normal mode (WGM02:0 = 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 8-bit value (top = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter overflow flag (TOV0) will be set on the same timer clock cycle on which the TCNT0 becomes zero. The TOV0 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 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.

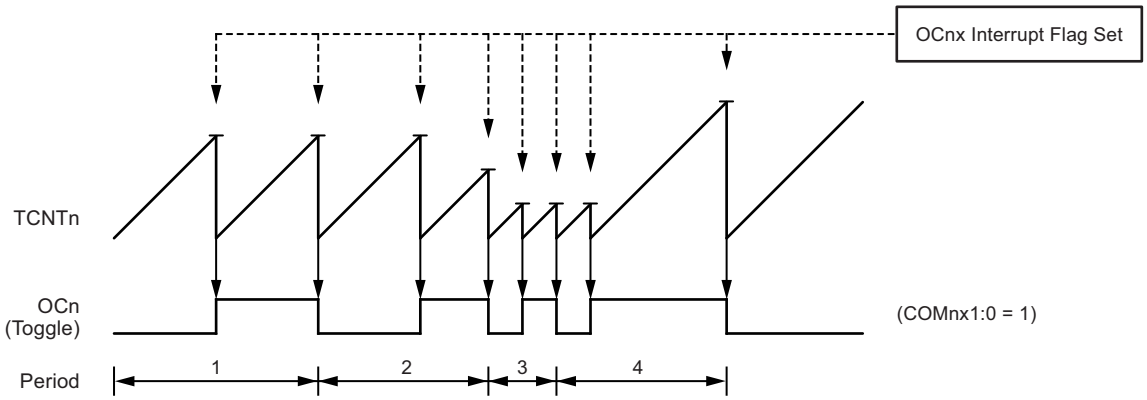
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much CPU time.

13.7.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare, or CTC, mode (WGM02:0 = 2), the OCR0A register is used to manipulate the counter resolution. In CTC mode, the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, 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 13-5. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 13-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the top value by using the OCF0A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the top value. However, changing top to a value close to bottom when the counter is running with no or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_0 = f_{clk\_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \times N \times (1 + OCRnx)}$$

The variable N represents the prescale factor (1, 8, 64, 256, or 1024).

As for the normal mode of operation, the TOV0 flag is set on the same timer clock cycle on which the counter counts from max to 0x00.

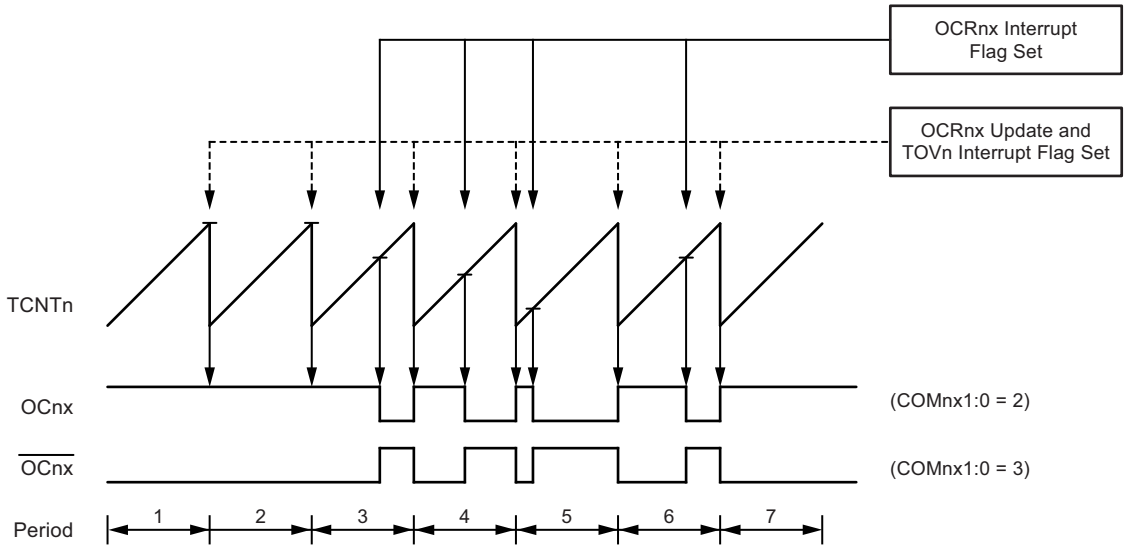
13.7.3 Fast PWM Mode

The fast pulse width modulation, or fast PWM, mode (WGM02:0 = 3 or 7) provides a high-frequency PWM waveform generation option. The fast PWM mode differs from the other PWM option by its single-slope operation. The counter counts from bottom to top then restarts from bottom. Top is defined as 0xFF when WGM2:0 = 3, and as OCR0A when WGM2:0 = 7. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at bottom. In inverting compare output mode, the output is set on compare match and cleared at bottom. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation.

This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows the use of physically smaller external components (coils, capacitors, etc.), and hence reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 13-6. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0x and TCNT0.

Figure 13-6. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM, and an inverted PWM output can be generated by setting the COM0x1:0 bits to three. Setting the COM0A1:0 bits to one allows the AC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 13-3 on page 73). 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 setting (or clearing) the OC0x register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x register at the timer clock cycle when the counter is cleared (changes from top to bottom).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times 256}$$

The variable N represents the prescale 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 fast PWM mode. If the OCR0A is set equal to bottom, the out- put will be a narrow spike for each max+1 timer clock cycle. Setting the OCR0A equal to max will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits).

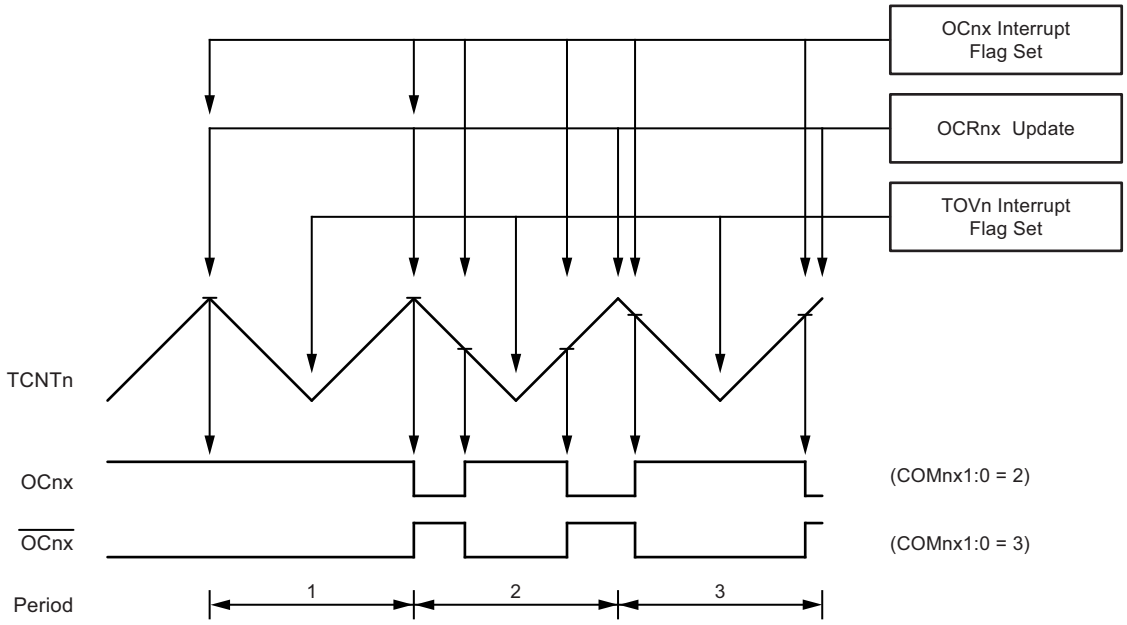
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each compare match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_0 = f_{clk\_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

13.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM02:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from bottom to top and then from top to bottom. Top is defined as 0xFF when WGM2:0 = 1, and as OCR0A when WGM2:0 = 5. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x while up-counting, and set on the compare match while down-counting. In inverting output compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 13-7. The TCNT0 value is in the timing diagram, which is shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 13-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches bottom. The interrupt flag can be used to generate an interrupt each time the counter reaches the bottom value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 bits to three. Setting the COM0A0 bits to one allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 13-4 on page 73). 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 the following equation:

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

The variable N represents the prescale 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 is set equal to bottom, the output will be continuously low, and if set equal 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 Figure 13-7 on page 70 OCn has a transition from high to low even though there is no compare match. The point of this transition is to guarantee symmetry around bottom. There are two cases that give a transition without a compare match.

- OCR0A changes its value from MAX, as in Figure 13-7 on page 70. When the OCR0A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting compare match.
- The timer starts counting from a value higher than the one in OCR0A, and for that reason misses the compare match and, hence, the OCn change that would have happened on the way up.

13.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design, and the timer clock (clkT0) is, therefore, shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set. Figure 13-8 contains timing data for basic Timer/Counter operation.The figure shows the count sequence close to the max value in all modes other than phase correct PWM mode.

Figure 13-8. Timer/Counter Timing Diagram, no Prescaling

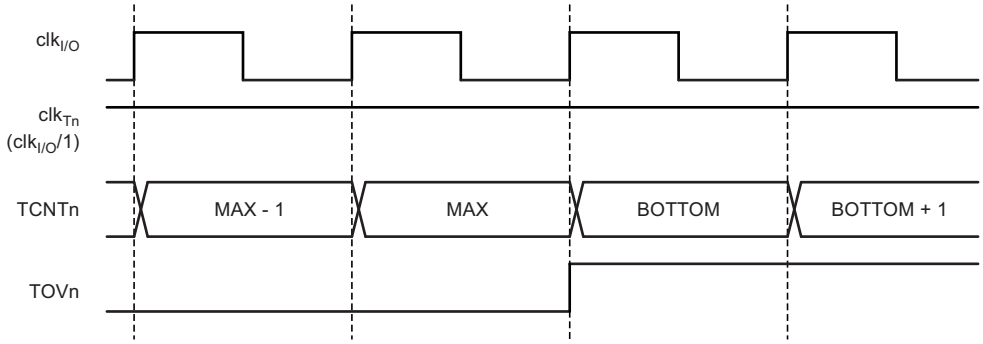


Figure 13-9 shows the same timing data, but with the prescaler enabled.

Figure 13-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

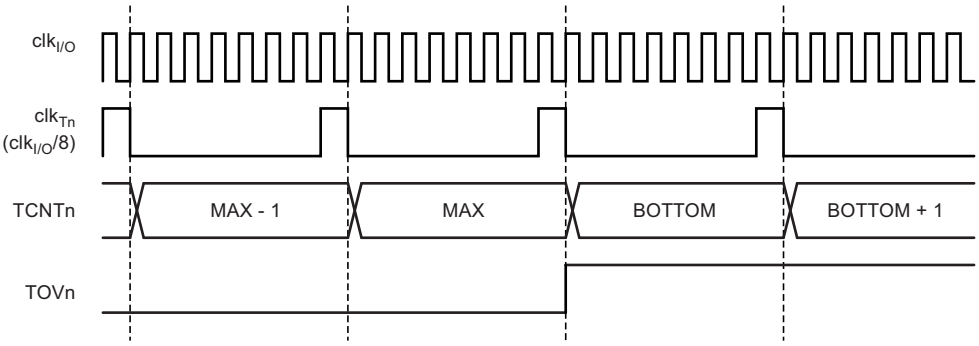


Figure 13-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 13-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

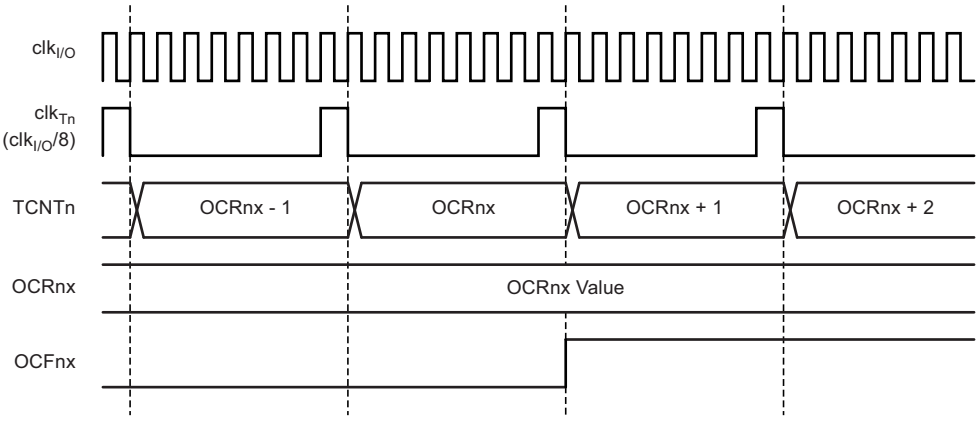
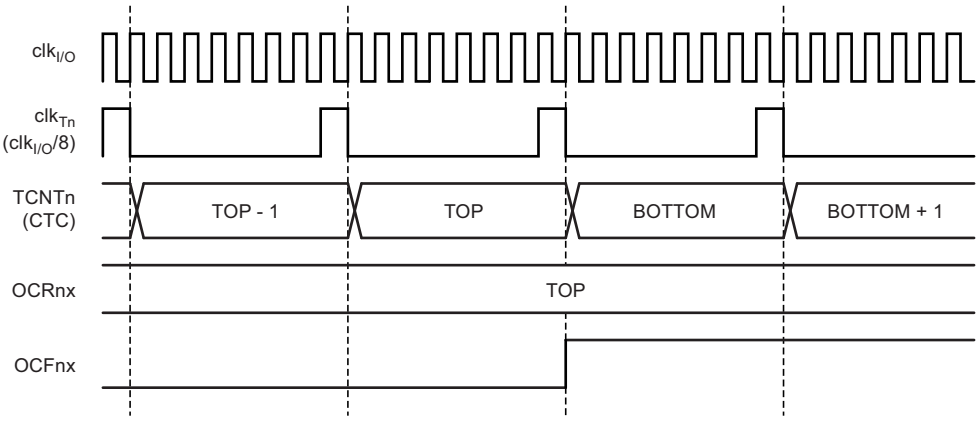


Figure 13-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 13-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}/8$)



13.9 Register Description

13.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM0A1	COM0A**0**	COM0B**1**	COM0B**0**	–	–	**WGM01**	**WGM00**	TCCR0A
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	

- **Bits 7:6 – COM0A1:0: Compare Match Output A 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 WGM02:0 bit setting. Table 13-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 13-2. Compare Output Mode, non-PWM Mode

COM01	COM00	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 13-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 13-3. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
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 Section 13.7.3 “Fast PWM Mode” on page 69 for more details.

Table 13-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 13-4. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match when up-counting. Set OC0A on compare match when down-counting.
1	1	Set OC0A on compare match when up-counting. Clear OC0A on compare match when down-counting.

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 TOP. See Section 13.7.4 “Phase Correct PWM Mode” on page 70 for more details.

- **Bits 5:4 – COM0B1:0: Compare Match Output B Mode**

These bits control the output compare pin (OC0B) behavior. If one or both of the COM0B1:0 bits are set, the OC0B 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 OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting. Table 13-2 on page 73 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 13-5. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

Table 13-3 on page 73 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 13-6. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match, set OC0B at BOTTOM (non-inverting mode)
1	1	Set OC0B on compare match, clear OC0B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See Section 13.7.3 “Fast PWM Mode” on page 69 for more details.

Table 13-4 on page 73 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 13-7. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match when up-counting. Set OC0B on compare match when down-counting.
1	1	Set OC0B on compare match when up-counting. Clear OC0B on compare match when down-counting.

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 13.7.4 “Phase Correct PWM Mode” on page 70 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny24/44/84 and will always read as zero.

- **Bits 1:0 – WGM01:0: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 13-8 on page 75. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 13.7 “Modes of Operation” on page 67).

Table 13-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Note: 1. MAX = 0xFF
BOTTOM = 0x00

13.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to logical zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate compare match is forced on the waveform generation unit. The OC0A output is changed according to its COM0A1:0 bit settings. Note that the FOC0A bit is implemented as a strobe. Therefore, it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bit 6 – FOC0B: Force Output Compare B**

The FOC0B bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to logical zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0B bit, an immediate compare match is forced on the waveform generation unit. The OC0B output is changed according to its COM0B1:0 bit settings. Note that the FOC0B bit is implemented as a strobe. Therefore, it is the value present in the COM0B1:0 bits that determines the effect of the forced compare.

A FOC0B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0B as TOP.

The FOC0B bit is always read as zero.

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84, and will always read as zero.

- **Bit 3 – WGM02: Waveform Generation Mode**

See the description in the Section 13.9.1 “TCCR0A – Timer/Counter Control Register A” on page 73.

- **Bits 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 13-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

13.9.3 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0[7:0]								TCNT0
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 Timer/Counter register gives direct access, for both read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0x registers.

13.9.4 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	OCR0A[7:0]								OCR0A
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 output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0A pin.

13.9.5 OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x3C (0x5C)	OCR0B[7:0]								OCR0B
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 output compare register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0B pin.

13.9.6 TIMSK0 – Timer/Counter 0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny24/44/84 and will always read as zero.

- **Bit 2- OCIE0B: Timer/Counter 0 Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the status register is set, the Timer/Counter 0 compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter 0 interrupt flag register (TIFR0).

- **Bit 1– OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter 0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter 0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register (TIFR0).

- **Bit 0– TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter 0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter 0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register (TIFR0).

13.9.7 TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny24/44/84 and will always read as zero.

- **Bit 2– OCF0B: Output Compare Flag 0 B**

The OCF0B bit is set when a compare match occurs between the Timer/Counter 0 and the data in OCR0B, the output compare register 0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logical one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter compare b match interrupt enable), and OCF0B are set, the Timer/Counter compare match interrupt is executed.

- **Bit 1– OCF0A: Output Compare Flag 0 A**

The OCF0A bit is set when a compare match occurs between the Timer/Counter 0 and the data in OCR0A, the output compare register 0 A. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logical one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter 0 compare match interrupt enable), and OCF0A are set, the Timer/Counter 0 compare match interrupt is executed.

- **Bit 0– TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter 0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logical one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter 0 overflow interrupt enable), and TOV0 are set, the Timer/Counter 0 overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. See Table 13-8 on page 75 and Section 13-8 “Waveform Generation Mode Bit Description” on page 75.

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)

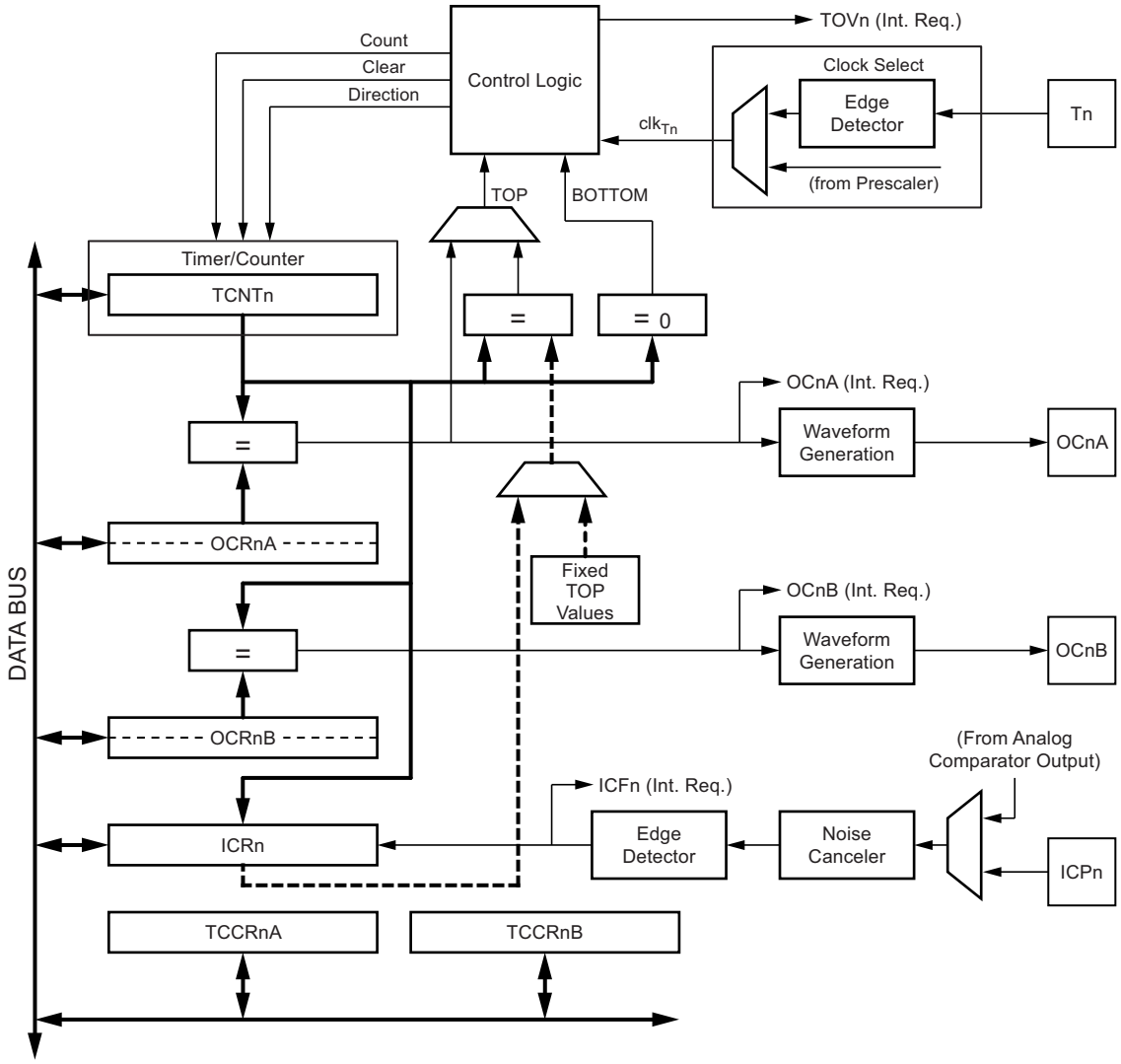
14.2 Overview

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.

Figure 14-1. 16-bit Timer/Counter Block Diagram(1)



Note: 1. See Figure 1-1 on page 3 for Timer/Counter1 pin placement and description.

14.2.1 Registers

The Timer/Counter (TCNT1), output compare registers (OCR1A/B), and input capture register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the Section 14.3 “Accessing 16-bit Registers” on page 80. The Timer/Counter control registers (TCCR1A/B) are 8-bit registers, and have no CPU access restrictions. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the timer interrupt flag register (TIFR). All interrupts are individually masked with the timer interrupt mask register (TIMSK). TIFR and TIMSK are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The clock select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T1}).

The double buffered output compare registers (OCR1A/B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pin (OC1A/B). See Section 14.7 “Output Compare Units” on page 86. The compare match event will also set the compare match flag (OCF1A/B) which can be used to generate an output compare interrupt request.

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.

The following code examples show how to access the 16-bit timer registers, assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 registers. Note that when using C, the compiler handles the 16-bit access.

Assembly Code Examples[1]

```
...
; Set TCNT1 to 0x01FF
ldi    r17,0x01
ldi    r16,0xFF
out    TCNT1H,r17
out    TCNT1L,r16
; Read TCNT1 into r17:r16
in     r16,TCNT1L
in     r17,TCNT1H
...
```

C Code Examples[1]

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...
```

Note: 1. See Section 4. “About Code Examples” on page 8.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register and the interrupt code updates the temporary register by accessing the same or any of the other 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNT1 register contents. Reading any of the OCR1A/B or ICR1 registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_ReadTCNT1:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Read TCNT1 into r17:r16
        in      r16,TCNT1L
        in      r17,TCNT1H
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

C Code Example[1]

```
unsigned int TIM16_ReadTCNT1( void )
{
        unsigned char sreg;
        unsigned int i;
        /* Save global interrupt flag */
        sreg = SREG;
        /* Disable interrupts */
        _CLI();
        /* Read TCNT1 into i */
        i = TCNT1;
        /* Restore global interrupt flag */
        SREG = sreg;
        return i;
}
```

Note: 1. See Section 4. “About Code Examples” on page 8.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_WriteTCNT1:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Set TCNT1 to r17:r16
        out     TCNT1H,r17
        out     TCNT1L,r16
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

C Code Example[1]

```
void TIM16_WriteTCNT1( unsigned int i )
{
        unsigned char sreg;
        unsigned int i;
        /* Save global interrupt flag */
        sreg = SREG;
        /* Disable interrupts */
        _CLI();
        /* Set TCNT1 to i */
        TCNT1 = i;
        /* Restore global interrupt flag */
        SREG = sreg;
}
```

Note: 1. See Section 4. “About Code Examples” on page 8.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

14.3.1 Reusing the Temporary High Byte Register

If when writing to more than one 16-bit register the high byte is the same for all registers writ- ten, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

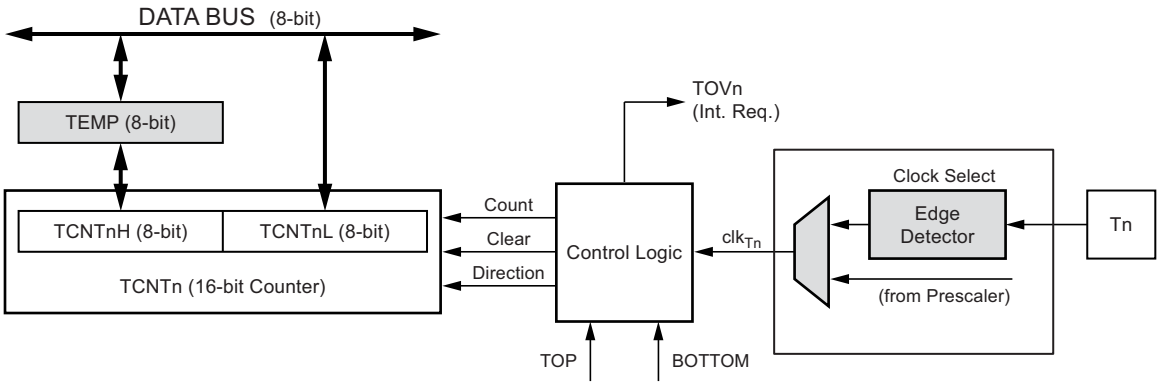
14.4 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS12:0) bits located in the Timer/Counter control register B (TCCR1B). For details on clock sources and prescaler, see Section 15. “Timer/Counter Prescaler” on page 103.

14.5 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 14-2 shows a block diagram of the counter and its surroundings.

Figure 14-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_{T1}	Timer/Counter clock.
TOP	Signal that TCNT1 has reached maximum value.
BOTTOM	Signal that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: counter high (TCNT1H) containing the upper eight bits of the counter, and counter low (TCNT1L) containing the lower eight bits. The TCNT1H register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clkT1). The clkT1 can be generated from an external or internal clock source, selected by the clock select bits (CS12:0). When no clock source is selected (CS12:0 = 0,) the timer is stopped. However, the TCNT1 value can be accessed by the CPU independently of whether clkT1 is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the waveform generation mode bits (WGM13:0) located in Timer/Counter control registers A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs (OC1x). For more details about advanced counting sequences and waveform generation, see Section 14.9 “Modes of Operation” on page 89.

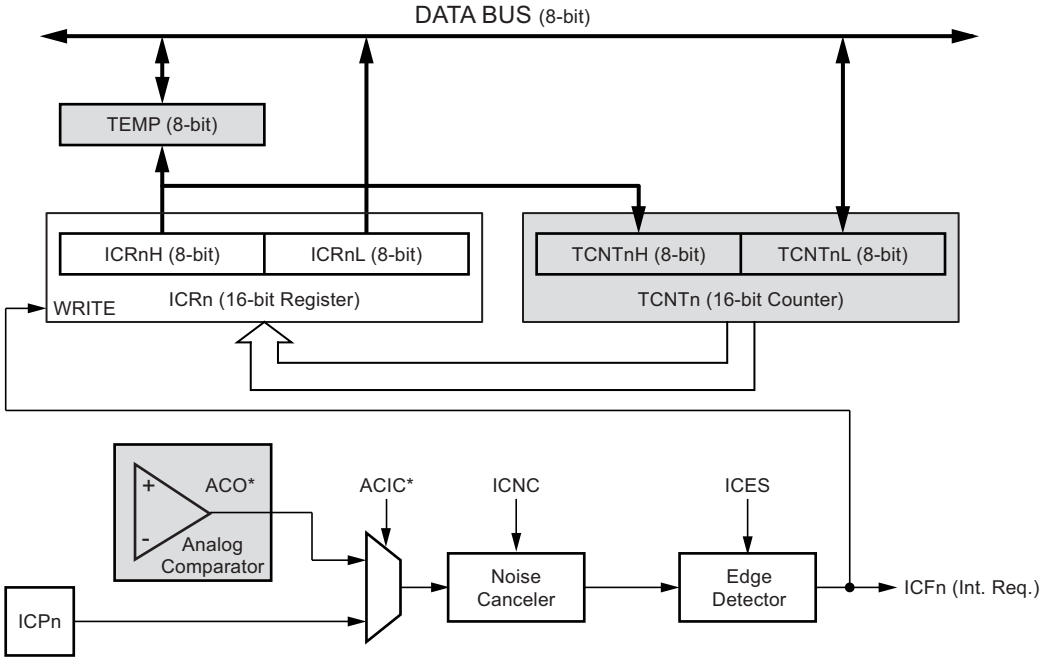
The Timer/Counter overflow flag (TOV1) is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

14.6 Input Capture Unit

The Timer/Counter incorporates an input capture unit that can capture external events and give them a time stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or, alternatively, via the analog comparator unit. The time stamps can then be used to calculate frequency, duty cycle, and other features of the signal applied. Alternatively, the time stamps can be used for creating a log of the events.

The input capture unit is illustrated by the block diagram shown in Figure 14-3 on page 85. The elements of the block diagram that are not directly a part of the input capture unit are shaded gray. The small “n” in register and bit names indicates the Timer/Counter number.

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.

14.6.2 Noise Canceller

The noise canceller improves noise immunity by using a simple digital filtering scheme. The noise canceller input is monitored over four samples, and all four must be equal to change the output, which in turn is used by the edge detector.

The noise canceller is enabled by setting the input capture noise canceller (ICNC1) bit in Timer/Counter control register B (TCCR1B). When enabled, the noise canceller introduces an additional four system clock cycles of delay between a change applied to the input and the update of the ICR1 register. The noise canceller uses the system clock, and is, therefore, not affected by the prescaler.

14.6.3 Using the Input Capture Unit

The main challenge when using the input capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the input capture interrupt, the ICR1 register should be read as early in the interrupt handler routine as possible. Even though the input capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the input capture unit in any mode of operation when the top value (resolution) is actively changed during operation is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge be changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 register has been read. After a change of the edge, the input capture flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 flag is not required (if an interrupt handler is used).

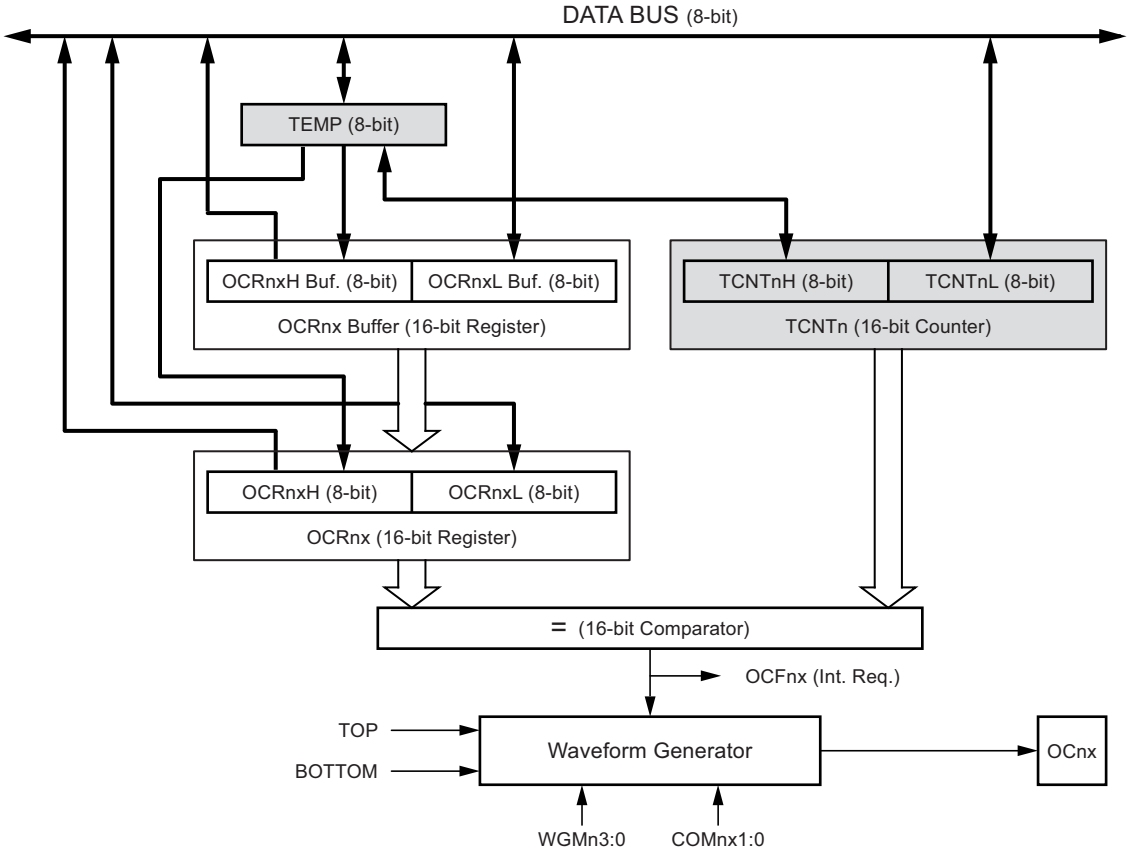
14.7 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the output compare register (OCR1x). If TCNT equals OCR1x, the comparator signals a match. A match will set the output compare flag (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the output compare flag generates an output compare interrupt. The OCF1x flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to the operating mode set by the waveform generation mode (WGM13:0) bits and compare output mode (COM1x1:0) bits. The top and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (Section 14.9 “Modes of Operation” on page 89).

A special feature of output compare unit A allows it to define the Timer/Counter TOP value (i.e., counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the waveform generator.

Figure 14-4 on page 87 shows a block diagram of the output compare unit. The small “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B). The elements of the block diagram that are not directly a part of the output compare unit are shaded gray.

Figure 14-4. Output Compare Unit, Block Diagram



The OCR1x register is double buffered when using any of the twelve pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x compare register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

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

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

14.7.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a logical one to the force output compare (1x) bit. Forcing compare match will not set the OCF1x flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the COM11:0 bit settings define whether the OC1x pin is set, cleared or toggled).

14.7.2 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

14.7.3 Using the Output Compare Unit

Because writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved in changing TCNT1 when using any of the output compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to top in PWM modes with variable top values. The compare match for the top will be ignored, and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to bottom when the counter is down-counting.

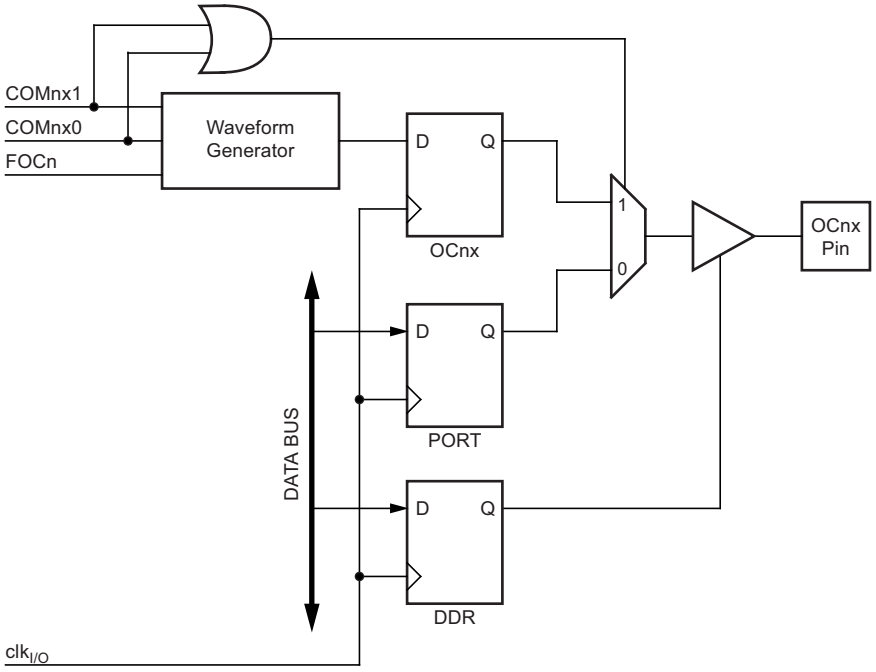
The setup of the OC1x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC1x value is to use the force output compare (1x) strobe bits in normal mode. The OC1x register keeps its value even when changing between waveform generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

14.8 Compare Match Output Unit

The compare output mode (COM1x1:0) bits have two functions. The waveform generator uses the COM1x1:0 bits for defining the output compare (OC1x) state at the next compare match. Secondly the COM1x1:0 bits control the OC1x pin output source. Figure 14-5 shows a simplified schematic of the logic affected by the COM1x1:0 bit settings. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x register, not the OC1x pin. If a system reset occurs, the OC1x register is reset to “0”.

Figure 14-5. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC1x) from the waveform generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the waveform generation mode, but there are some exceptions. See Table 14-2 on page 97, Table 14-3 on page 97 and Table 14-4 on page 98 for details.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation.
See Section 14.11 “Register Description” on page 97.

The COM1x1:0 bits have no effect on the input capture unit.

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.

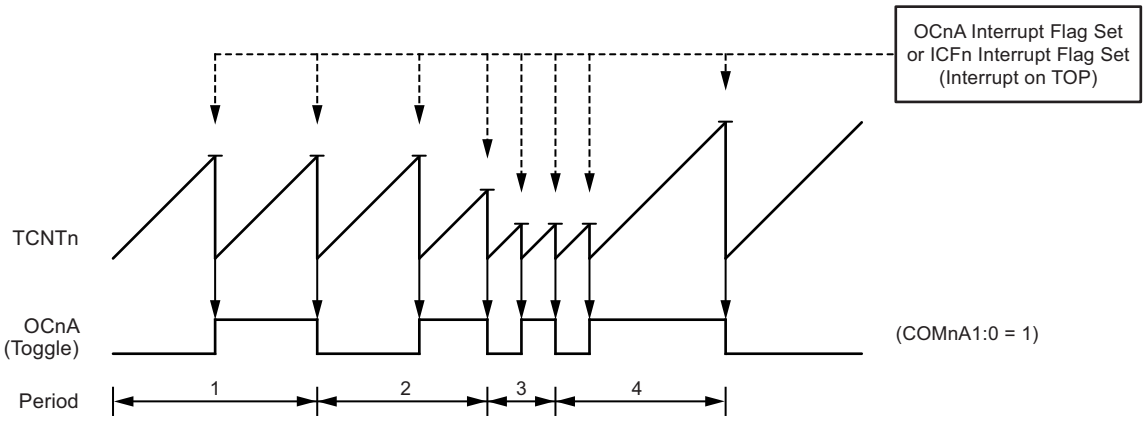
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended because this will occupy too much CPU time.

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



An interrupt can be generated each time the counter value reaches the top value by either using the OCF1A or ICF1 flag according to the register used to define the top value. If the interrupt is enabled, the interrupt handler routine can be used for updating the top value. However, changing the top to a value close to bottom when the counter is running with no or a low prescaler value must be done with care because the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode, using OCR1A for defining top (WGM13:0 = 15) because OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will have a maximum frequency of $_{1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \times N \times (1 + OCRnA)}$$

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

As for the normal mode of operation, the TOV1 flag is set on the same timer clock cycle on which the counter counts from max to 0x0000.

14.9.3 Fast PWM Mode

The fast pulse width modulation, or fast PWM, mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high-frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from bottom to top then restarts from bottom.

In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x, and set at bottom. In inverting compare output mode, output is set on compare match and cleared at bottom.

Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows the use of physically smaller external components (coils, capacitors, etc.), and hence reduces total system cost.

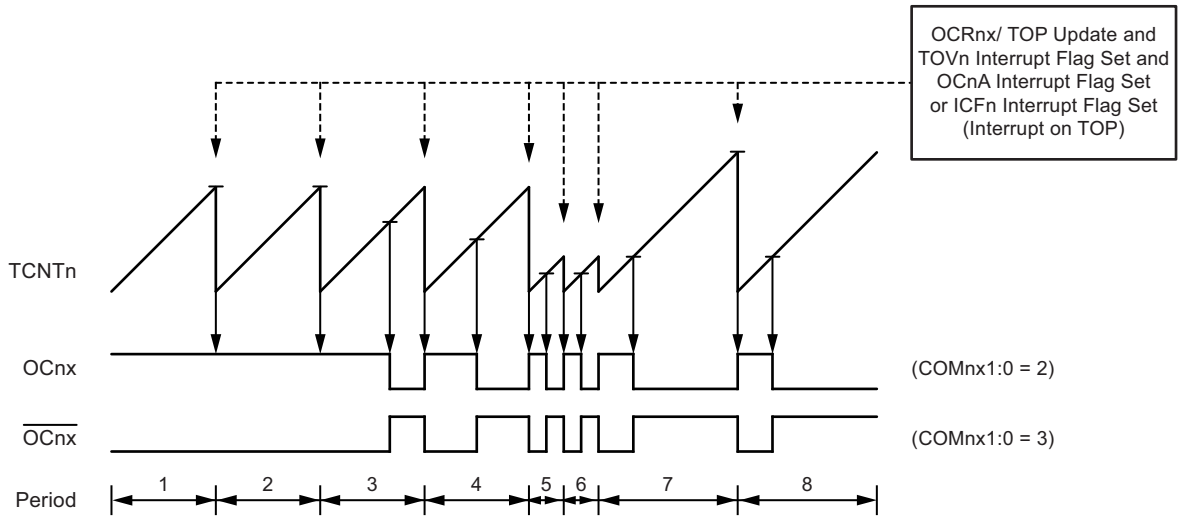
The PWM resolution for fast PWM can be fixed to 8, 9, or 10 bits, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2 bits (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16 bits (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

:

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 14-7 on page 91. The figure shows fast PWM mode when OCR1A or ICR1 is used to define top. The TCNT1 value in the timing diagram is shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 14-7. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set each time the counter reaches top. In addition, the OC1A or ICF1 flag is set on the same timer clock cycle on which TOV1 is set when either OCR1A or ICR1 is used for defining the top value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the top and compare values.

When changing the top value, the program must ensure that the new top value is higher or equal to the value of all of the compare registers. If the top value is lower than any of the compare registers, a compare match will never occur between TCNT1 and OCR1x. Note that when using fixed top values, the unused bits are masked to zero when any of the OCR1x registers are written

The procedure for updating ICR1 differs from that for updating OCR1A when used for defining the top value. The ICR1 register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with no or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the top value. The counter will then have to count to the max value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A register, however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written, the value written will be put into the OCR1A buffer register. The OCR1A compare register will then be updated with the value in the buffer register at the next timer clock cycle when TCNT1 matches top. The update is done on the same timer clock cycle on which TCNT1 is cleared and the TOV1 flag is set.

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

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM, and an inverted PWM out- put can be generated by setting the COM1x1:0 to three (see Table 14-3 on page 97). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x register on the timer clock cycle on which the counter is cleared (changes from top to bottom). The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times (1 + TOP)}$$

The variable N represents the prescaler divider (1, 8, 64, 256, or 1024). The extreme values for the OCR1x register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to bottom (0x0000), the output will be a narrow spike for each top+1 timer clock cycle. Setting OCR1x equal to top will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.) A frequency waveform output (with 50% duty cycle) in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each compare match (COM1A1:0 = 1). The waveform generated will have a maximum frequency of $_{1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

14.9.4 Phase Correct PWM Mode

The phase correct pulse width modulation, or phase correct PWM, mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high-resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from bottom (0x0000) to top and then from top to bottom.

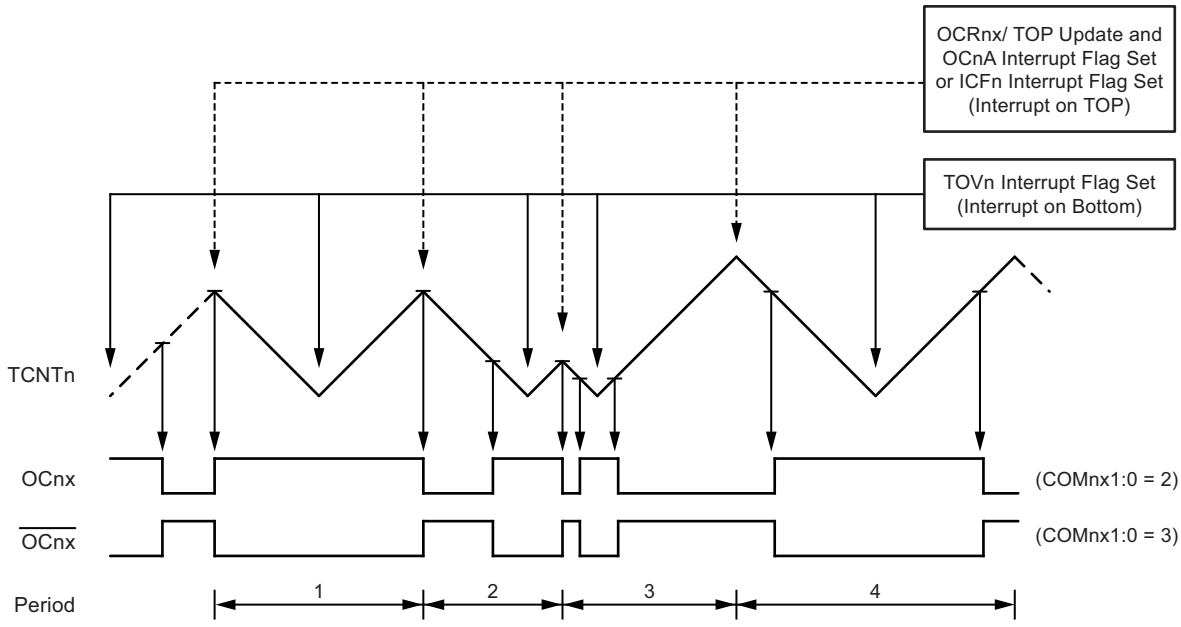
In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting, and set on the compare match while down-counting. In inverting output compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8, 9, or 10 bits, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2 bits (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16 bits (ICR1 or OCR1A set to max). The PWM resolution in bits can be calculated by using the following equation:

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase correct PWM mode, the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the top, and changes the count direction. The TCNT1 value will be equal to top for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 14-8. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define top. The TCNT1 value in the timing diagram is shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 14-8. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set each time the counter reaches bottom. When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag is set accordingly on the same timer clock cycle on which the OCR1x registers are updated with the double buffer value (at top). The interrupt flags can be used to generate an interrupt each time the counter reaches the top or bottom value.

When changing the top value, the program must ensure that the new top value is higher or equal to the value of all of the compare registers. If the top value is lower than any of the compare registers, a compare match will never occur between TCNT1 and OCR1x. Note that when using fixed top values, the unused bits are masked to zero when any of the OCR1x registers are written. As the third period shown in Figure 14-8 on page 92 illustrates, changing the top actively while the Timer/Counter is running in the phase correct mode can result in an asymmetrical output. The reason for this can be found in the time of update of the OCR1x register. Since the OCR1x update occurs at top, the PWM period starts and ends at top. This implies that the length of the falling slope is determined by the previous top value, while the length of the rising slope is determined by the new top value. When these two values differ, the two slopes of the period will differ in length. The difference in length gives the asymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the top value while the Timer/Counter is running. When using a static top value, there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM, and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table 14-4 on page 98). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \times N \times TOP}$$

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

The extreme values for the OCR1x register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to bottom, the output will be continuously low, and if set equal to top, the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

14.9.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct pulse width modulation, or phase and frequency correct PWM, mode (WGM13:0 = 8 or 9) provides a high-resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from bottom (0x0000) to top and then from top to bottom. In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting, and set on the compare match while down counting.

In inverting compare output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

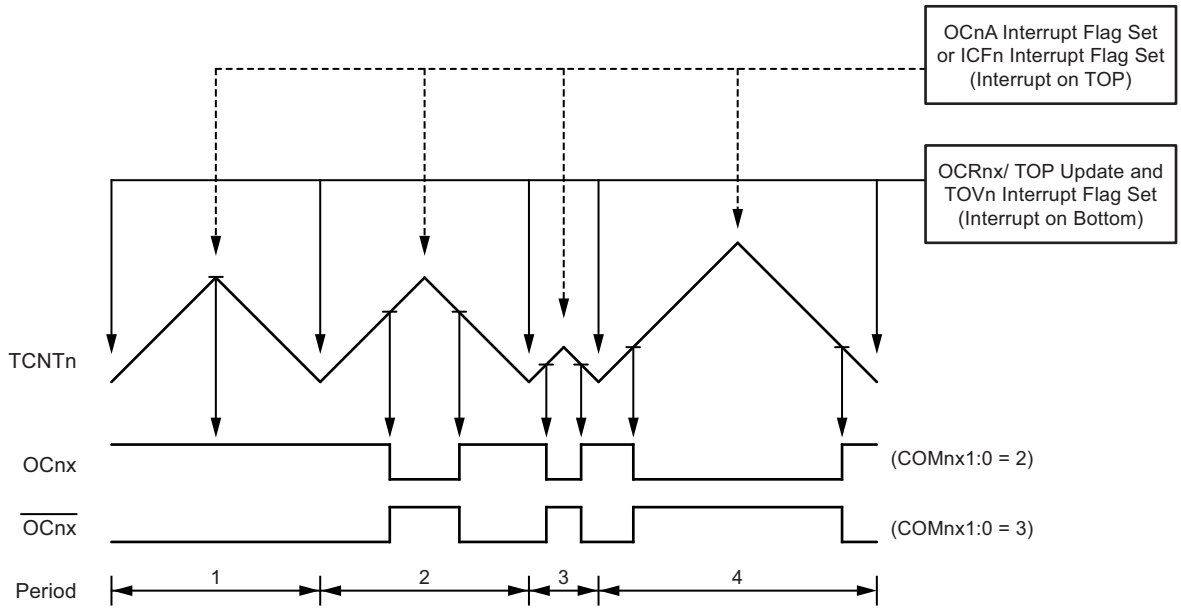
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x register is updated by the OCR1x buffer register (see Figure 14-8 on page 92 and Figure 14-9 on page 94).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2 bits (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16 bits (ICR1 or OCR1A set to max). The PWM resolution in bits can be calculated using the following equation:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode, the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the top, and changes the count direction. The TCNT1 value will be equal to top for one timer clock cycle. The timing diagram for the phase and frequency correct PWM mode is shown on Figure 14-9 on page 94. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define top. The TCNT1 value in the timing diagram is shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 14-9. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set on the same timer clock cycle on which the OCR1x registers are updated with the double buffer value (at bottom). When either OCR1A or ICR1 is used for defining the top value, the OC1A or ICF1 flag is set accordingly when TCNT1 has reached top. The interrupt flags can then be used to generate an interrupt each time the counter reaches the top or bottom value.

When changing the top value, the program must ensure that the new top value is higher or equal to the value of all of the compare registers. If the top value is lower than any of the compare registers, a compare match will never occur between TCNT1 and OCR1x.

As Figure 14-9 shows, the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x registers are updated at bottom, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses, and is, therefore, frequency correct.

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

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM, and an inverted PWM output can be generated by setting the COM1x1:0 to three (see Table 14-4 on page 98). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \times N \times TOP}$$

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

The extreme values for the OCR1x register represent special cases when generating a PWM waveform output in the phase and frequency correct PWM mode. If the OCR1x is set equal to bottom the output will be continuously low, and if set equal to top, the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

14.10 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design, and the timer clock (clkT1) is, therefore, shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR1x register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 14-10 shows a timing diagram for the setting of OCF1x.

Figure 14-10. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling

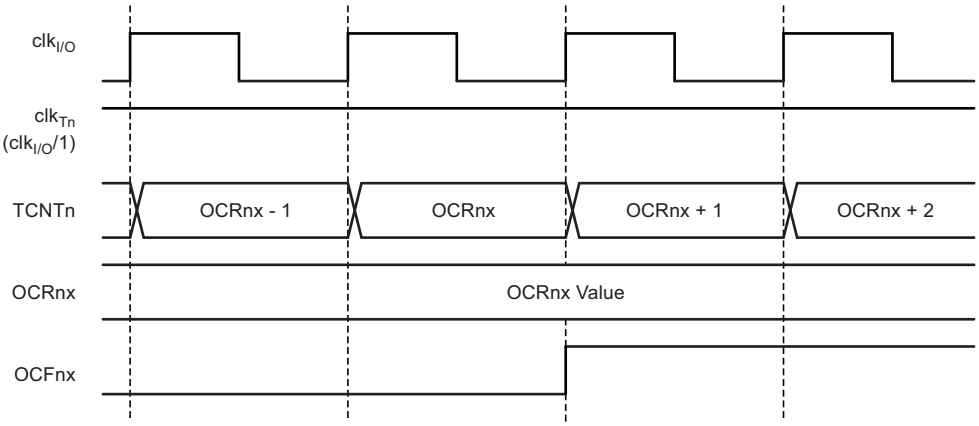


Figure 14-11 shows the same timing data, but with the prescaler enabled.

Figure 14-11. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}$/8)

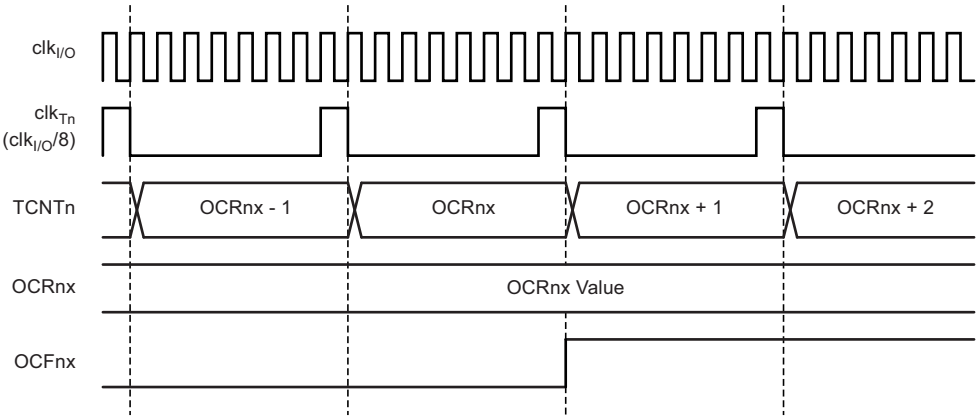


Figure 14-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 flag at BOTTOM.

Figure 14-12. Timer/Counter Timing Diagram, no Prescaling

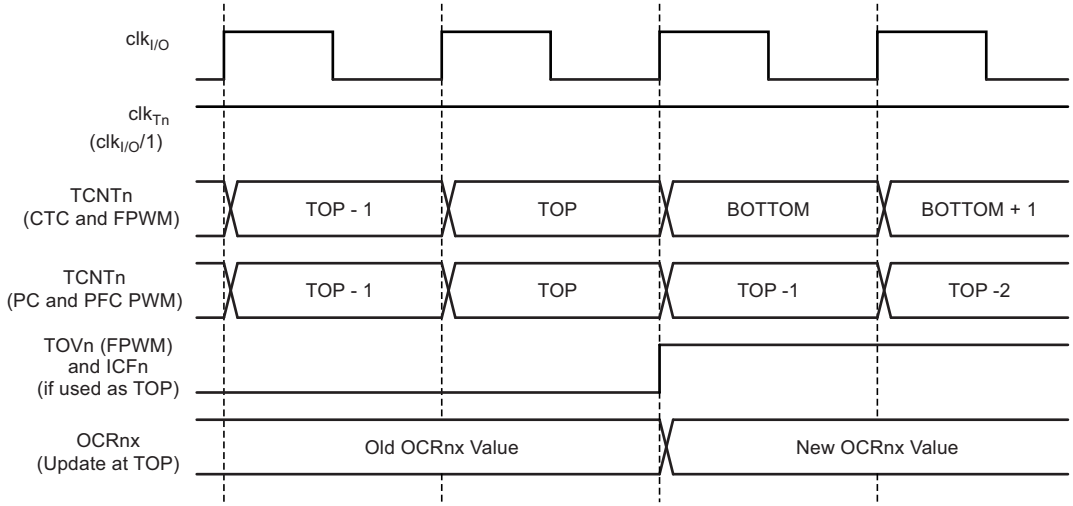
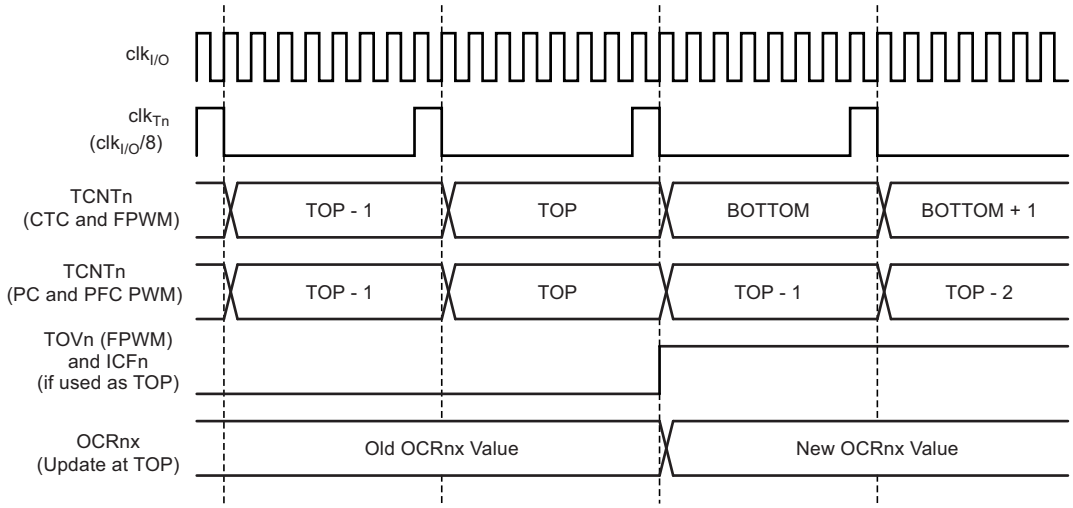


Figure 14-13 shows the same timing data, but with the prescaler enabled.

Figure 14-13. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}$/8)



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.

Table 14-4 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 14-4. Compare Output Mode, Phase Correct and Phase and Frequency Correct 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 when up-counting. Set OC1A/OC1B on compare match when down-counting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when down-counting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See Section 14.9.4 “Phase Correct PWM Mode” on page 92 for more details.

- **Bit 1:0 – WGM11:0: Waveform Generation Mode**

Combined with the WGM13:2 bits found in the TCCR1B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation is to be used, see Table 14-5 on page 98. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and three types of pulse width modulation (PWM) modes (see Section 14.9 “Modes of Operation” on page 89).

Table 14-5. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

14.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: Input Capture Noise Canceller**

Setting this bit (to one) activates the input capture noise canceller. When the noise canceller is activated, the input from the input capture pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The input capture is, therefore, delayed by four oscillator cycles when the noise canceller is enabled.

- **Bit 6 – ICES1: Input Capture Edge Select**

This bit selects which edge on the input capture pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to logical zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to logical one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the input capture register (ICR1). The event will also set the input capture flag (ICF1), and this can be used to cause an input capture interrupt, if this interrupt is enabled.

When the ICR1 is used as top value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B registers), the ICP1 is disconnected, and, consequently, the input capture function is disabled.

- **Bit 5 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to logical zero when TCCR1B is written.

- **Bit 4:3 – WGM13:2: Waveform Generation Mode**

See TCCR1A register description.

- **Bit 2:0 – CS12:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter (see Figure 14-10 on page 95 and Figure 14-11 on page 95.

Table 14-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

14.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
0x22 (0x42)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**

- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specify a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written while operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bit settings. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore, it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt, nor will it clear the timer in clear timer on compare match (CTC) mode using OCR1A as top. The FOC1A/FOC1B bits are always read as zero.

- **Bit 5..0 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to logical zero when the register is written.

14.11.4 TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	TCNT1[15:8]								TCNT1H
0x2C (0x4C)	TCNT1[7:0]								TCNT1L
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 two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access for both read and for write operations to the Timer/Counter unit's 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. See Section 14.3 “Accessing 16-bit Registers” on page 80.

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.

14.11.5 OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	OCR1A[15:8]								OCR1AH
0x2A (0x4A)	OCR1A[7:0]								OCR1AL
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	

14.11.6 OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
0x29 (0x49)	OCR1B[15:8]								OCR1BH
0x28 (0x48)	OCR1B[7:0]								OCR1BL
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 output compare registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC1x pin.

The output compare registers are 16 bits in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to 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. See Section 14.3 “Accessing 16-bit Registers” on page 80.

14.11.7 ICR1H and ICR1L – Input Capture Register 1

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	ICR1[15:8]								ICR1H
0x24 (0x44)	ICR1[7:0]								ICR1L
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 input capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the analog comparator output for Timer/Counter1). The input capture can be used for defining the counter TOP value.

The input capture register is 16 bits in size. To ensure that both the high and low bytes are read 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. Section 14.3 “Accessing 16-bit Registers” on page 80.

14.11.8 TIMSK1 – Timer/Counter Interrupt Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
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 7,6,4,3 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to logical zero when the register is written.

- **Bit 5 – ICIE1: Timer/Counter1, Input Capture Interrupt Enable**

When this bit is written to logical one and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter 1 input capture interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 44) is executed when the ICF1 flag, located in TIFR1, is set.

- **Bit 2– OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable**

When this bit is written to logical one and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter 1 output compare B match interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 44) is executed when the OCF1B flag, located in TIFR1, is set.

- **Bit 1– OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to logical one and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter 1 output compare A match interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 44) is executed when the OCF1A flag, located in TIFR1, is set.

- **Bit 0 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to logical one and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter 1 overflow interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 44) is executed when the TOV1 flag, located in TIFR1, is set.

14.11.9 TIFR1 – Timer/Counter Interrupt Flag Register 1

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	–	–	ICIF1	–	–	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 7,6,4,3 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to logical zero when the register is written.

- **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 logical 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 (1B) 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 logical 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 (1A) 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 logical one to its bit location.

- **Bit 0– TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bit settings. In normal and CTC modes, the TOV1 flag is set when the timer overflows. See Table 14-5 on page 98 for the TOV1 flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter 1 overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logical one to its bit location.

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).

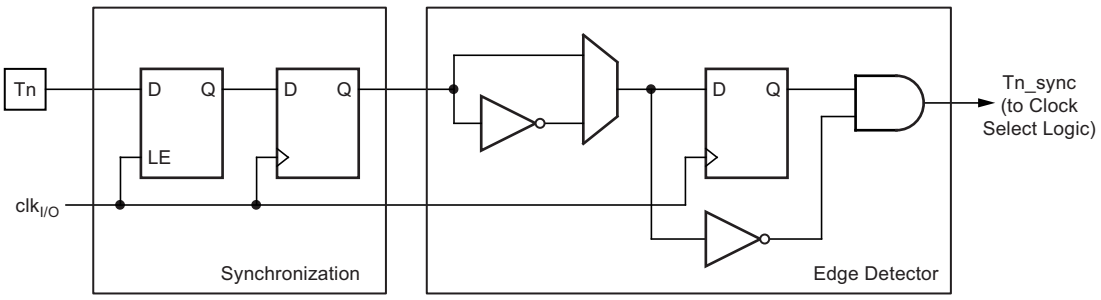
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution.

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



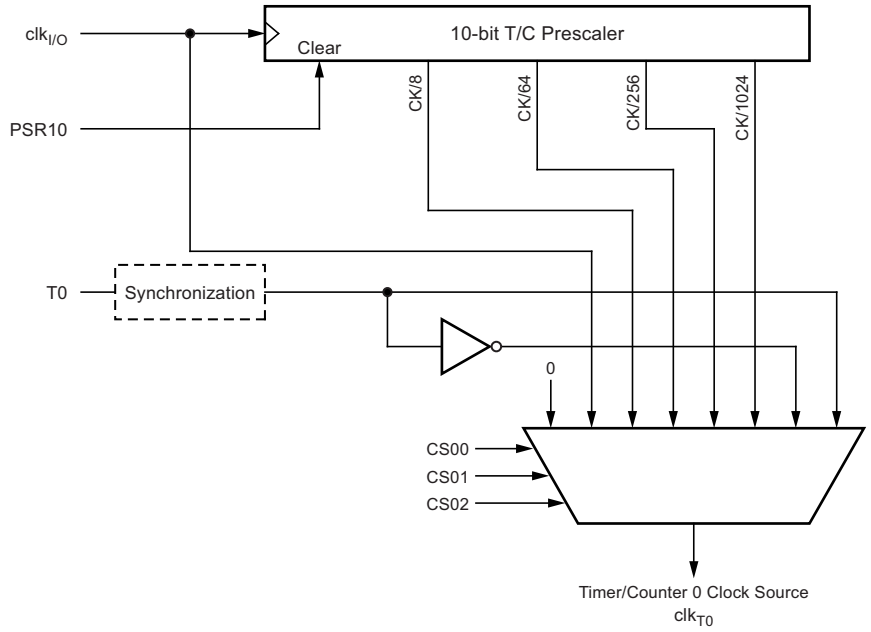
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from when an edge has been applied to the Tn pin to when the counter is updated.

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.

Figure 15-2. Prescaler for Timer/Counter0



Note: The synchronization logic on the input pins (T0) is shown in Figure 15-1 on page 103.

15.3 Register Description

15.3.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	–	–	–	–	–	–	PSR10	GTCCR
Read/Write	R/W	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

Writing the TSM bit to logical one activates the Timer/Counter synchronization mode. In this mode, the value that is written to the PSR10 bit is kept, hence keeping the prescaler reset signal asserted. This ensures that the Timer/Counter is halted and can be configured without the risk of advancing during configuration. When the TSM bit is written to logical zero, the PSR10 bit is cleared by hardware, and the Timer/Counter starts counting.

- **Bit 0 – PSR10: Prescaler 0 Reset Timer/Counter n**

When this bit is set to one, the Timer/Counter n prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

16. USI – Universal Serial Interface

16.1 Features

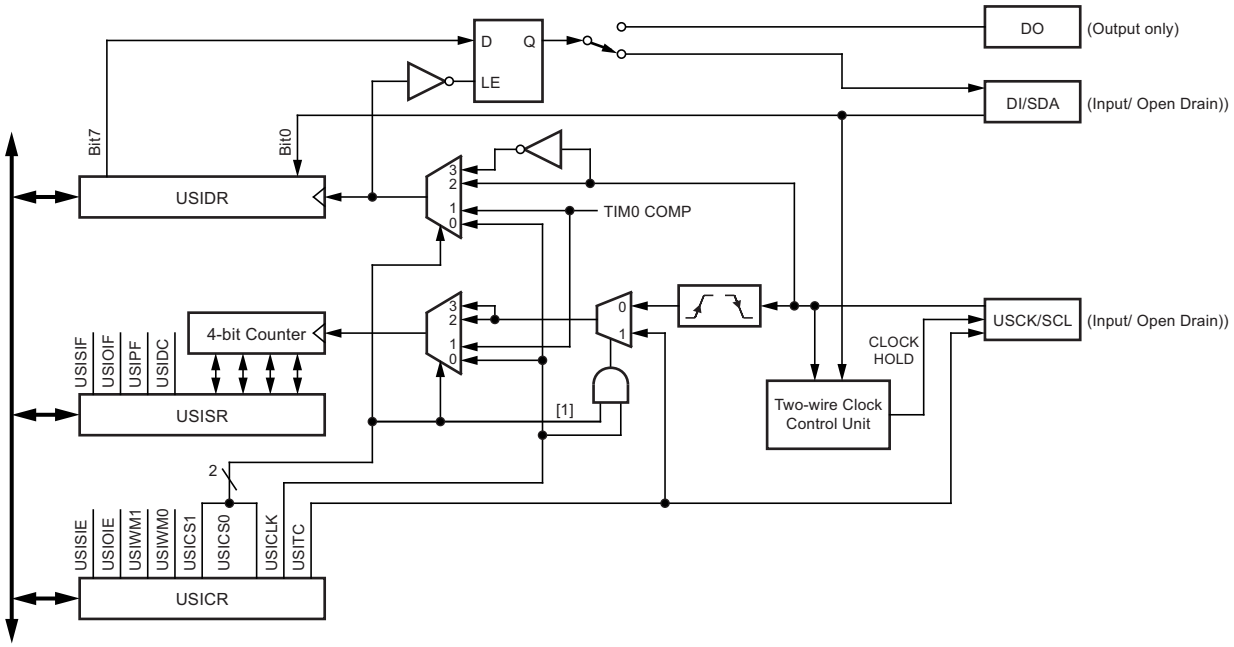
- Two-wire synchronous data transfer (master or slave)
- Three-wire synchronous data transfer (master or slave)
- Data received interrupt
- Wake up from idle mode
- In two-wire mode: Wake up from all sleep modes, including power-down mode
- Two-wire start condition detector with interrupt capability

16.2 Overview

The universal serial interface (USI) provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize the processor load.

A simplified block diagram of the USI is shown in Figure 16-1. 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 16.5 “Register Descriptions” on page 111.

Figure 16-1. Universal Serial Interface, Block Diagram



The 8-bit shift register is directly accessible via the data bus and contains the incoming and outgoing data. The register has no buffering, so the data must be read as quickly as possible to ensure that no data are lost. The most significant bit is connected to one of two output pins, depending on the wire mode configuration. A transparent latch is inserted between the serial register output and output pin, which delays the change of data output to the opposite clock edge of the data input sampling. The serial input is always sampled from the data input (DI) pin independent of the configuration.

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the serial register and the counter are clocked simultaneously by the same clock source. This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected, the counter counts both clock edges. In this case, the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: the USCK pin, Timer/Counter 0 compare match, or from software. The two-wire clock control unit can generate an interrupt when a start condition is detected on the two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

16.3 Functional Descriptions

16.3.1 Three-wire Mode

The USI three-wire mode is compliant with the serial peripheral interface (SPI) mode 0 and 1, but does not have the slave select (SS) pin functionality. However, this feature can be implemented in software if necessary. Pin names used by this mode are: DI, DO, and USCK.

Figure 16-2. Three-wire Mode Operation, Simplified Diagram

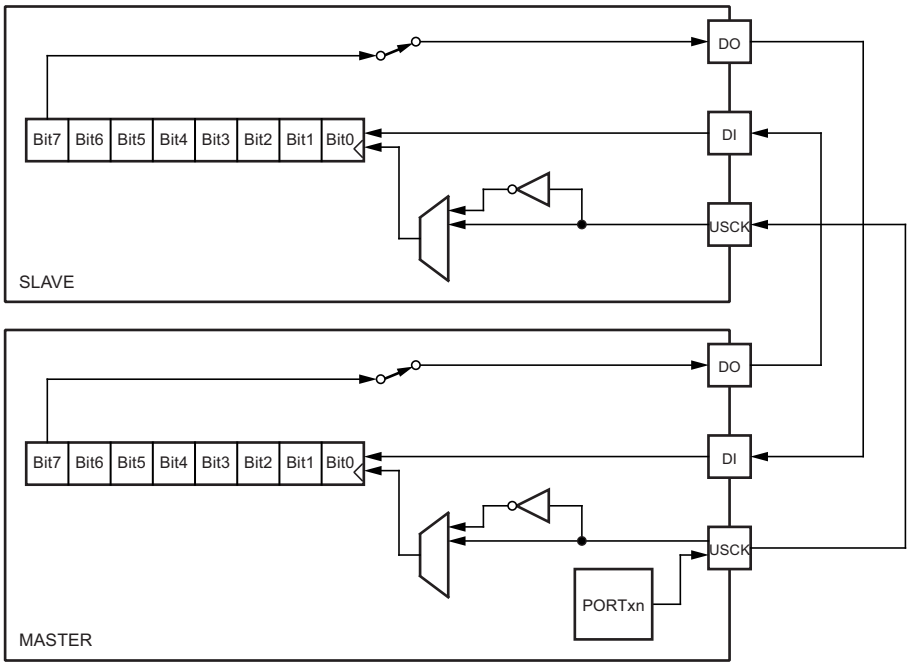
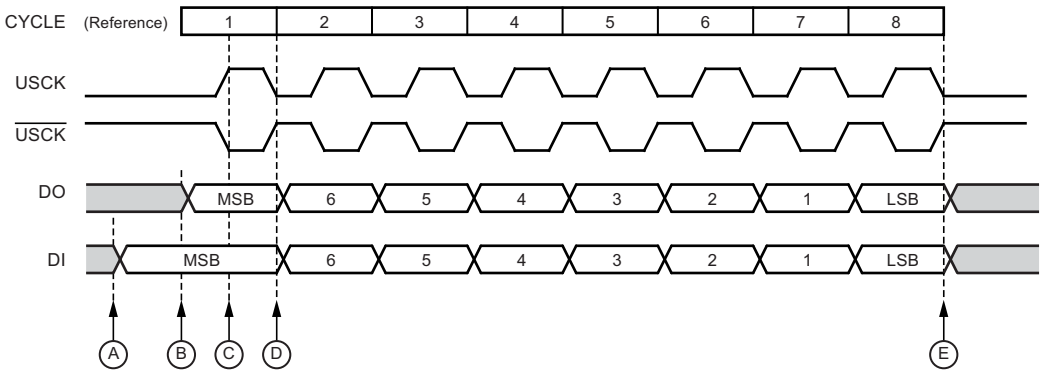


Figure 16-2 shows two USI units operating in three-wire mode, one as master and one as slave. The two shift registers are interconnected in such way that after eight USCK clocks, the data in each register are interchanged. The same clock also increments the USI's 4-bit counter. The counter overflow (interrupt) flag, or USIOIF, can therefore be used to determine when a transfer is completed. The clock is generated by the master device software by toggling the USCK pin via the PORT register, or by writing a logical one to the USITC bit in USICR.

Figure 16-3. Three-wire Mode, Timing Diagram



The three-wire mode timing is shown in Figure 16-3. At the top of the figure is a USCK cycle reference. One bit is shifted into the USI shift register (USIDR) for each of these cycles. The USCK timing is shown for both external clock modes. In external clock mode 0 (USICS0 = 0), DI is sampled at positive edges, and DO is changed (data register is shifted by one) at negative edges. External clock mode 1 (USICS0 = 1) uses the opposite edges versus mode 0, i.e., it samples data at negative edges and changes the output at positive edges. The USI clock modes correspond to the SPI data mode 0 and 1.

Referring to the timing diagram (Figure 16-3 on page 106), a bus transfer involves the following steps:

1. The slave device and master device set up their data output and, depending on the protocol used, enable their output driver (mark A and B). The output is set up by writing the data to be transmitted to the serial data register. Enabling of the output is done by setting the corresponding bit in the port data direction register. Note that points A and B do not have any specific order, but both must be at least one half USCK cycle before point C, where the data are sampled. This must be done to ensure that the data setup requirement is satisfied. The 4-bit counter is reset to zero.
2. The master generates a clock pulse by software toggling the USCK line twice (C and D). The bit value on the slave and master's data input (DI) pin is sampled by the USI on the first edge (C), and the data output is changed on the opposite edge (D). The 4-bit counter will count both edges.
3. Step 2 is repeated eight times for a complete register (byte) transfer.
4. After eight clock pulses (i.e., 16 clock edges), the counter will overflow and indicate that the transfer is completed. The data bytes transferred must now be processed before a new transfer can be initiated. The overflow interrupt will wake up the processor if it is set to idle mode. Depending on the protocol used, the slave device can now set its output to high impedance.

16.3.2 SPI Master Operation Example

The following code demonstrates how to use the USI module as a SPI master:

```
SPITransfer:
       out    USIDR,r16
       ldi    r16,(1<<USIOIF)
       out    USISR,r16
       ldi    r16,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)
SPITransfer_loop:
       out    USICR,r16
       in     r16, USISR
       sbrs   r16, USIOIF
       rjmp   SPITransfer_loop
       in     r16,USIDR
       ret
```

The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO and USCK pins are enabled as output in the DDRE register. The value stored in register r16 prior to the function being called is transferred to the slave device, and when the transfer is completed, the data received from the slave is stored back into the r16 register.

The second and third instructions clear the USI counter overflow flag and the USI counter value. The fourth and fifth instructions set the three-wire mode, positive edge shift register clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI module as an SPI master with maximum speed ($f_{sck} = f_{ck}/4$):

```
SPITransfer_Fast:

       out    USIDR,r16
       ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
       ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

       out    USICR,r16 ; MSB
       out    USICR,r17
       out    USICR,r16
       out    USICR,r17
       out    USICR,r16
       out    USICR,r17
       out    USICR,r16
       out    USICR,r17
       out    USICR,r16
       out    USICR,r17
       out    USICR,r16
       out    USICR,r17
       out    USICR,r16
```

```
        out    USICR,r17
        out    USICR,r16 ; LSB
        out    USICR,r17

        in     r16,USIDR
ret
```

16.3.3 SPI Slave Operation Example

The following code demonstrates how to use the USI module as a SPI slave:

```
init:
        ldi    r16,(1<<USIWM0)|(1<<USICS1)
        out    USICR,r16
...
SlaveSPITransfer:
        out    USIDR,r16
        ldi    r16,(1<<USIOIF)
        out    USISR,r16
SlaveSPITransfer_loop:
        in     r16, USISR
        sbrs   r16, USIOIF
        rjmp   SlaveSPITransfer_loop
        in     r16,USIDR
        ret
```

The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO is configured as output and USCK pin is configured as input in the DDR register. The value stored in register r16 prior to the function being called is transferred to the master device, and when the transfer is completed, the data received from the master is stored back into the r16 register.

Note that the first two instructions are for initialization only and need only to be executed once. These instructions set the three-wire mode and positive edge shift register clock. The loop is repeated until the USI counter overflow flag is set.

16.3.4 Two-wire Mode

The USI two-wire mode is compliant with the inter-IC (I2C or TWI) bus protocol, but without slew rate limiting on outputs and input noise filtering. Pin names used by this mode are SCL and SDA.

Figure 16-4. Two-wire Mode Operation, Simplified Diagram

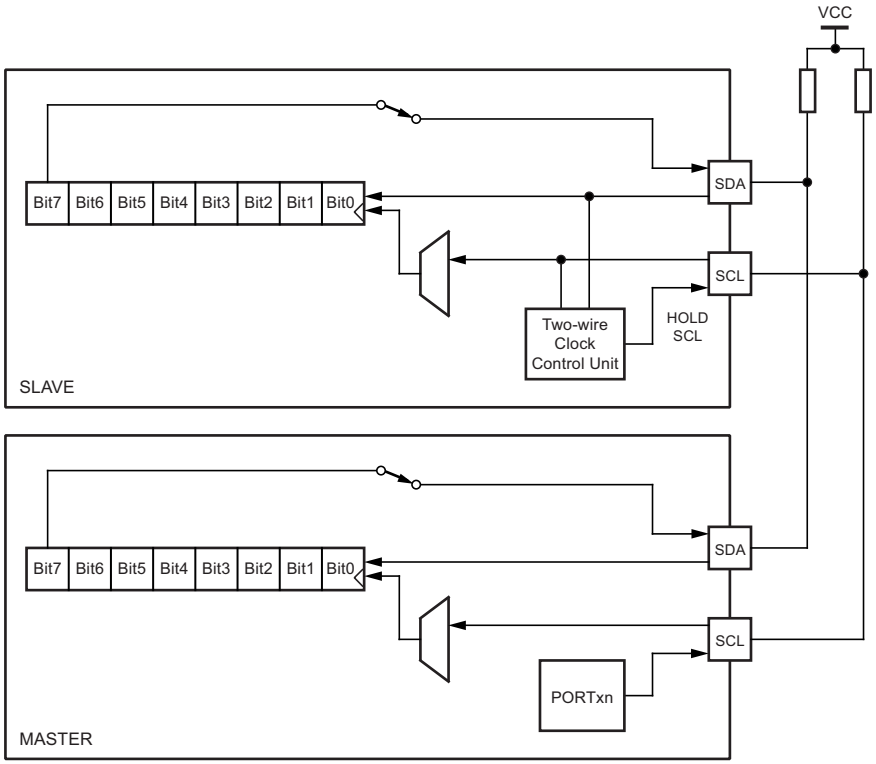
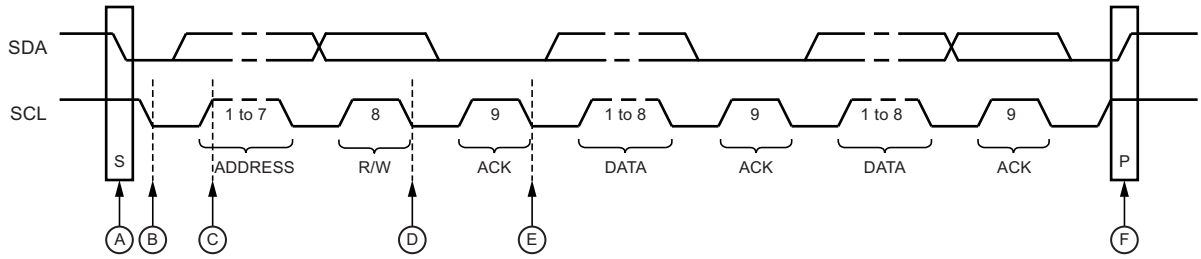


Figure 16-4 shows two USI units operating in two-wire mode, one as master and one as slave. Only the physical layer is shown because the system operation is highly dependent of the communication scheme used. The main differences between the master and slave operation at this level are that the serial clock generation is always done by the master, and only the slave uses the clock control unit. Clock generation must be implemented in software, but the shift operation is done automatically by both devices. Note that only clocking on the negative edge to shift data is practical in this mode. The slave can insert wait states at the start or end of a transfer by forcing the SCL clock low. This means that the master must always check if the SCL line was actually released after it has generated a positive edge.

Because the clock also increments the counter, a counter overflow can be used to indicate that the transfer has completed. The master generates clock by the by toggling the USCK pin via the PORT register.

The data direction is not given by the physical layer. A protocol, like the one used by the TWI-bus, must be implemented to control the data flow.

Figure 16-5. Two-wire Mode, Typical Timing Diagram

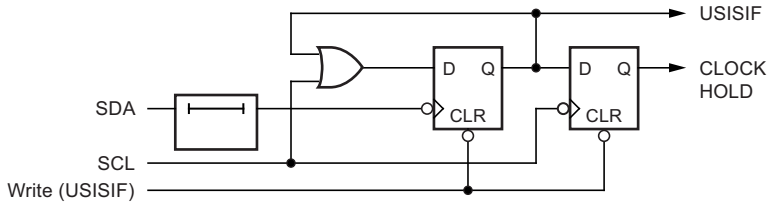


Referring to the timing diagram (Figure 16-5 on page 109), a bus transfer involves the following steps:

1. The start condition is generated by the master by forcing the SDA line low while the SCL line is high (A). SDA can be forced low either by writing a logical zero to bit 7 of the shift register, or by setting the corresponding bit in the PORT register to zero. Note that the data direction register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 16-6) detects the start condition and sets the USISIF flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the master has forced a negative edge on this line (B). This allows the slave to wake up from sleep or complete its other tasks before setting up the shift register to receive the address. This is done by clearing the start condition flag and resetting the counter.
3. The master set the first bit to be transferred and releases the SCL line (C). The Slave samples the data and shift it into the serial register at the positive edge of the SCL clock.
4. After eight bits containing the slave address and data direction (read or write) are transferred, the slave counter overflows and the SCL line is forced low (D). If the slave is not the one the master has addressed, it releases the SCL line and waits for a new start condition.
5. If the slave is addressed, it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the counter register must be set to 14 before releasing SCL at (D)). Depending on the state of the R/W bit, the master or slave enables its output. If the bit is set, a master read operation is in progress (i.e., the slave drives the SDA line). The slave can hold the SCL line low after the acknowledgement cycle (E).
6. Multiple bytes can now be transmitted, all in same direction, until a stop condition is given by the master (F), or a new start condition is given.

If the slave is not able to receive more data, it does not acknowledge the data byte it has last received. When the master does a read operation, it must terminate the operation by forcing the acknowledge bit low after the last byte is transmitted.

Figure 16-6. Start Condition Detector, Logic Diagram



16.3.5 Start Condition Detector

The start condition detector is shown in Figure 16-6. The SDA line is delayed (in the range of 50 to 300ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in two-wire mode.

The start condition detector is working asynchronously, and can, therefore, wake up the processor from the power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case, the oscillator start-up time set by the CKSEL fuses (see Section 7.1 “Clock Systems and their Distribution” on page 24) must also be taken into consideration. See the USISIF bit description in Section 16.5.3 “USISR – USI Status Register” on page 112 for further details.

16.3.6 Clock speed considerations

Maximum frequency for SCL and SCK is f_{CK} /4. This is also the maximum data transmit and receive rate in both two- wire and three-wire mode. In two-wire slave mode the two-wire clock control unit will hold SCL low until the slave is ready to receive more data. This may reduce the actual data rate in two-wire mode.

16.4 Alternative USI Usage

When the USI unit is not used for serial communication, it can be set up to do alternative tasks due to its flexible design.

16.4.1 Half-duplex Asynchronous Data Transfer

By utilizing the shift register in three-wire mode, it is possible to implement a more compact and higher performance UART than by software only.

16.4.2 4-bit Counter

The 4-bit counter can be used as a stand-alone counter with overflow interrupt. Note that if the counter is clocked externally, both clock edges will generate an increment.

16.4.3 12-bit Timer/Counter

Combining the USI 4-bit counter and Timer/Counter0 allows them to be used as a 12-bit counter.

16.4.4 Edge-Triggered External Interrupt

By setting the counter to maximum value (F), it can function as an additional external interrupt. The overflow flag and interrupt enable bit are then used for the external interrupt. This feature is selected by the USICS1 bit.

16.4.5 Software Interrupt

The counter overflow interrupt can be used as a software interrupt triggered by a clock strobe.

16.5 Register Descriptions

16.5.1 USIBR – USI Data Buffer

Bit	7	6	5	4	3	2	1	0	
0x10 (0x30)	MSB							LSB	USIBR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

16.5.2 USIDR – USI Data Register

Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	MSB							LSB	USIDR
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 USI uses no buffering for the serial register, i.e., when accessing the data register (USIDR) the serial register is accessed directly. If a serial clock occurs during the same cycle the register is written, the register will contain the value written and no shift is performed. A (left) shift operation is performed depending on the USICS1..0 bit settings. The shift operation can be controlled by an external clock edge, by a Timer/Counter 0 compare match, or directly by software using the USICLK strobe bit. Note that even when no wire mode is selected (USIWM1..0 = 0), both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the shift register.

The output pin in use - DO or SDA, depending on the wire mode - is connected via the output latch to the most-significant Bit (bit 7) of the data register. The output latch is open (transparent) during the first half of a serial clock cycle when an external clock source is selected (USICS1 = 1), and constantly open when an internal clock source is used (USICS1 = 0). The output will be changed immediately when a new MSB is written as long as the latch is open. The latch ensures that data input is sampled and data output is changed on opposite clock edges.

Note that the corresponding data direction register to the pin must be set to one for enabling data output from the shift register.

16.5.3 USISR – USI Status Register

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	USISR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The status register contains interrupt flags, line status flags and the counter value.

- **Bit 7 – USISIF: Start Condition Interrupt Flag**

When two-wire mode is selected, the USISIF flag is set (one) when a start condition is detected. When output disable mode or three-wire mode is selected and (USICSx = 0b11 and USICLK = 0) or (USICS = 0b10 and USICLK = 0), any edge on the SCK pin sets the flag.

An interrupt will be generated when the flag is set while the USISIE bit in USICR and the global interrupt enable flag are set. The flag will only be cleared by writing a logical one to the USISIF bit. Clearing this bit will release the start detection hold of USCL in two-wire mode. A start condition interrupt will wake up the processor from all sleep modes.

- **Bit 6 – USIOIF: Counter Overflow Interrupt Flag**

This flag is set (one) when the 4-bit counter overflows (i.e., at the transition from 15 to 0). An interrupt will be generated when the flag is set while the USIOIE bit in USICR and the global interrupt enable flag are set. The flag is cleared if a logical one is written to the USIOIF bit or by reading the USIBR register. Clearing this bit will release the counter overflow hold of SCL in two-wire mode.

A counter overflow interrupt will wake up the processor from Idle sleep mode.

- **Bit 5 – USIPF: Stop Condition Flag**

When two-wire mode is selected, the USIPF flag is set (one) when a stop condition is detected. The flag is cleared by writing a logical one to this bit. Note that this is not an interrupt flag. This signal is useful when implementing two-wire bus master arbitration.

- **Bit 4 – USIDC: Data Output Collision**

This bit is logical one when bit 7 in the shift register differs from the physical pin value. The flag is only valid when two-wire mode is used. This signal is useful when implementing two-wire bus master arbitration.

- **Bits 3..0 – USICNT3..0: Counter Value**

These bits reflect the current 4-bit counter value. The 4-bit counter value can be read or written directly by the CPU.

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a Timer/Counter 0 compare match, or by software using USICLK or USITC strobe bits. The clock source depends on the setting of the USICS1..0 bits. For external clock operation, a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by writing a logical one to the USICLK bit while setting an external clock source (USICS1 = 1).

Note that even when no wire mode is selected (USIWM1..0 = 0), the external clock input (USCK/SCL) can still be used by the counter.

16.5.4 USICR – USI Control Register

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	USICR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The control register includes interrupt enable control, wire mode setting, clock select setting, and clock strobe.

- **Bit 7 – USISIE: Start Condition Interrupt Enable**

Setting this bit to one enables the start condition detector interrupt. If there is a pending interrupt when the USISIE and the global interrupt enable flag are set to one, this will immediately be executed. See the USISIF bit description in Section 16.5.3 “USISR – USI Status Register” on page 112 for further details.

- **Bit 6 – USIOIE: Counter Overflow Interrupt Enable**

Setting this bit to one enables the counter overflow interrupt. If there is a pending interrupt when the USIOIE and the global interrupt enable flag are set to one, this will immediately be executed. See the USIOIF bit description in Section 16.5.3 “USISR – USI Status Register” on page 112 for further details.

- **Bit 5..4 – USIWM1..0: Wire Mode**

These bits set the type of wire mode to be used. Basically, only the function of the outputs are affected by these bits. Data and clock inputs are not affected by the mode selected, and will always have the same function. The counter and shift register can, therefore, be clocked externally, and data input sampled, even when outputs are disabled. The relation between USIWM1..0 and USI operation is summarized in Table 16-1.

Table 16-1. Relations between USIWM1..0 and the USI Operation

USIWM1	USIWM0	Description
0	0	Outputs, clock hold, and start detector disabled. Port pins operate as normal.
0	1	Three-wire mode. Uses DO, DI, and USCK pins. The data output (DO) pin overrides the corresponding bit in the PORT register in this mode. However, the corresponding DDR bit still controls the data direction. When the port pin is set as input the pins pull-up is controlled by the PORT bit. The data input (DI) and serial clock (USCK) pins do not affect the normal port operation. When operating as master, clock pulses are software generated by toggling the PORT register, while the data direction is set to output. The USITC bit in the USICR register can be used for this purpose.
1	0	Two-wire mode. Uses SDA (DI) and SCL (USCK) pins(1). The serial data (SDA) and the serial clock (SCL) pins are bi-directional and use open-collector output drivers. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDR register. When the output driver is enabled for the SDA pin, the output driver will force the SDA line low if the output of the shift register or the corresponding bit in the PORT register is zero. Otherwise, the SDA line will not be driven (i.e., it is released). When the SCL pin output driver is enabled, the SCL line will be forced low if the corresponding bit in the PORT register is zero, or by the start detector. Otherwise the SCL line will not be driven. The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the start condition flag (USISIF) releases the line. The SDA and SCL pin inputs are not affected by enabling this mode. Pull-ups on the SDA and SCL port pins are disabled in two-wire mode.
1	1	Two-wire mode. Uses SDA and SCL pins. Same operation as for the Two-wire mode described above, except that the SCL line is also held low when a counter overflow occurs, and is held low until the counter overflow flag (USIOIF) is cleared.

Note: 1. The DI and USCK pins are renamed to serial data (SDA) and serial clock (SCL), respectively, to avoid confusion between the modes of operation.

- **Bit 3..2 – USICS1..0: Clock Source Select**

These bits set the clock source for the shift register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using the external clock source (USCK/SCL). When the software strobe or Timer/Counter 0 compare match clock option is selected, the output latch is transparent and, therefore, the output is changed immediately. Clearing the USICS1..0 bits enables the software strobe option. When using this option, writing a logical one to the USICLK bit clocks both the shift register and the counter. For the external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 16-2 shows the relationship between the USICS1..0 and USICLK settings and the clock source used for the shift register and the 4-bit counter.

Table 16-2. Relations between the USICS1..0 and USICLK Setting

USICS1	USICS0	USICLK	Shift Register Clock Source	4-bit Counter Clock Source
0	0	0	No clock	No Clock
0	0	1	Software clock strobe (USICLK)	Software clock strobe (USICLK)
0	1	X	Timer/Counter0 compare match	Timer/Counter0 compare match
1	0	0	External, positive edge	External, both edges
1	1	0	External, negative edge	External, both edges
1	0	1	External, positive edge	Software clock strobe (USITC)
1	1	1	External, negative edge	Software clock strobe (USITC)

- **Bit 1 – USICLK: Clock Strobe**

Writing a logical one to this bit location strobes the shift register to shift one step, and the counter to increment by one, provided that the USICS1..0 bits are set to zero, and by doing so, the software clock strobe option is selected. The output will change immediately when the clock strobe is executed, i.e., in the same instruction cycle. The value shifted into the shift register is sampled during the previous instruction cycle. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a clock select register. Setting the USICLK bit in this case will select the USITC strobe bit as the clock source for the 4-bit counter (see Table 16-2).

- **Bit 0 – USITC: Toggle Clock Port Pin**

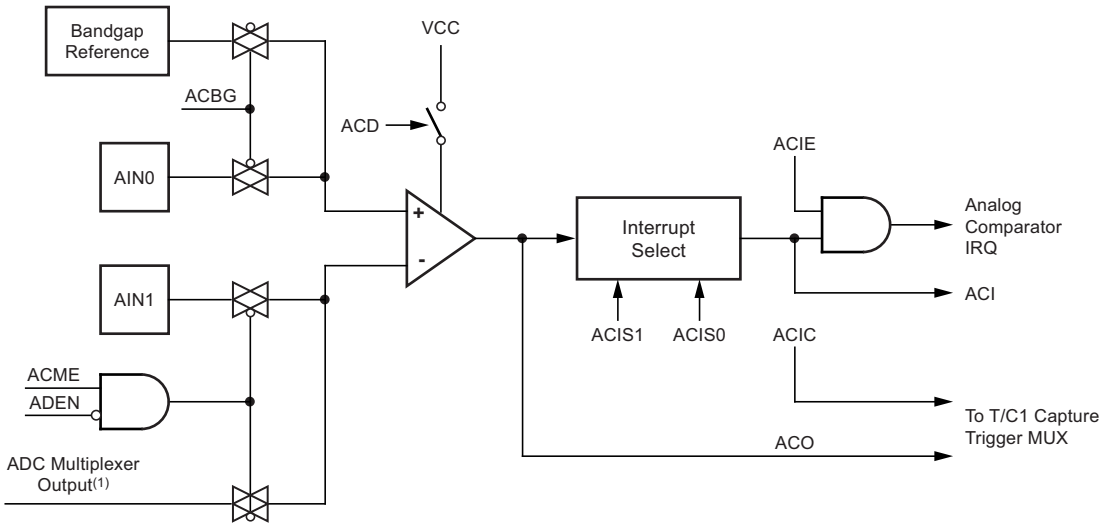
Writing a logical one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the data direction register, but if the PORT value is to be shown on the pin, DDRE4 must be set as output (to one). This feature allows easy clock generation when implementing master devices. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1) and the USICLK bit is set to one, writing to the USITC strobe bit will directly clock the 4-bit counter. This allows an early detection of when the transfer is done when operating as a master device.

17. Analog Comparator

The analog comparator compares the input values on the positive pin (AIN0) and negative pin (AIN1). When the voltage on the positive pin (AIN0) is higher than the voltage on the negative pin (AIN1), the analog comparator output (ACO) is set. The comparator can trigger a separate interrupt, exclusive to the analog comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 17-1 on page 115.

Figure 17-1. Analog Comparator Block Diagram(1)



Notes: 1. See Table 17-1.

2. See Figure 1-1 on page 3 and Table 12-9 on page 61 for analog comparator pin placement.

17.1 Analog Comparator Multiplexed Input

When the analog-to-digital converter (ADC) is configured as a single-ended input channel, it is possible to select any of the ADC7..0 pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input, and, consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX1..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 17-1. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input of the analog comparator.

Table 17-1. Analog Comparator Multiplexed Input

ACME	ADEN	MUX4..0	Analog Comparator Negative Input
0	x	xx	AIN1
1	1	xx	AIN1
1	0	00000	ADC0
1	0	00001	ADC1
1	0	00010	ADC2
1	0	00011	ADC3
1	0	00100	ADC4
1	0	00101	ADC5
1	0	00110	ADC6
1	0	00111	ADC7

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.

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

These bits determine which comparator events trigger the analog comparator interrupt. The different settings are shown in Table 17-2.

Table 17-2. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator interrupt on output toggle.
0	1	Reserved
1	0	Comparator interrupt on falling output edge.
1	1	Comparator interrupt on rising output edge.

When changing the ACIS1/ACIS0 bits, the analog comparator interrupt must be disabled by clearing its interrupt enable bit in the ACSR. Otherwise, an interrupt can occur when the bits are changed.

17.2.3 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1, 0 – ADC0D,ADC1D: ADC 1/0 Digital input buffer disable**

When this bit is written logical one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logical one to reduce power consumption in the digital input buffer.

18. Analog-to-Digital Converter

18.1 Features

- 10-bit resolution
- 1.0 LSB integral non-linearity
- ±2 LSB absolute accuracy
- 65 - 260µs conversion time
- Up to 76kSPS at maximum resolution
- Eight multiplexed single-ended input channels
- Twelve differential input channels with selectable gain (1x, 20x)
- Temperature sensor input channel
- Optional left adjustment for ADC result readout
- 0 to V_{CC} ADC input voltage range
- 1.1V ADC reference voltage
- Free-running or single-conversion mode
- ADC start conversion by auto triggering on interrupt sources
- Interrupt on ADC conversion complete
- Sleep mode noise canceller
- Unipolar / bipolar input mode
- Input polarity reversal channels

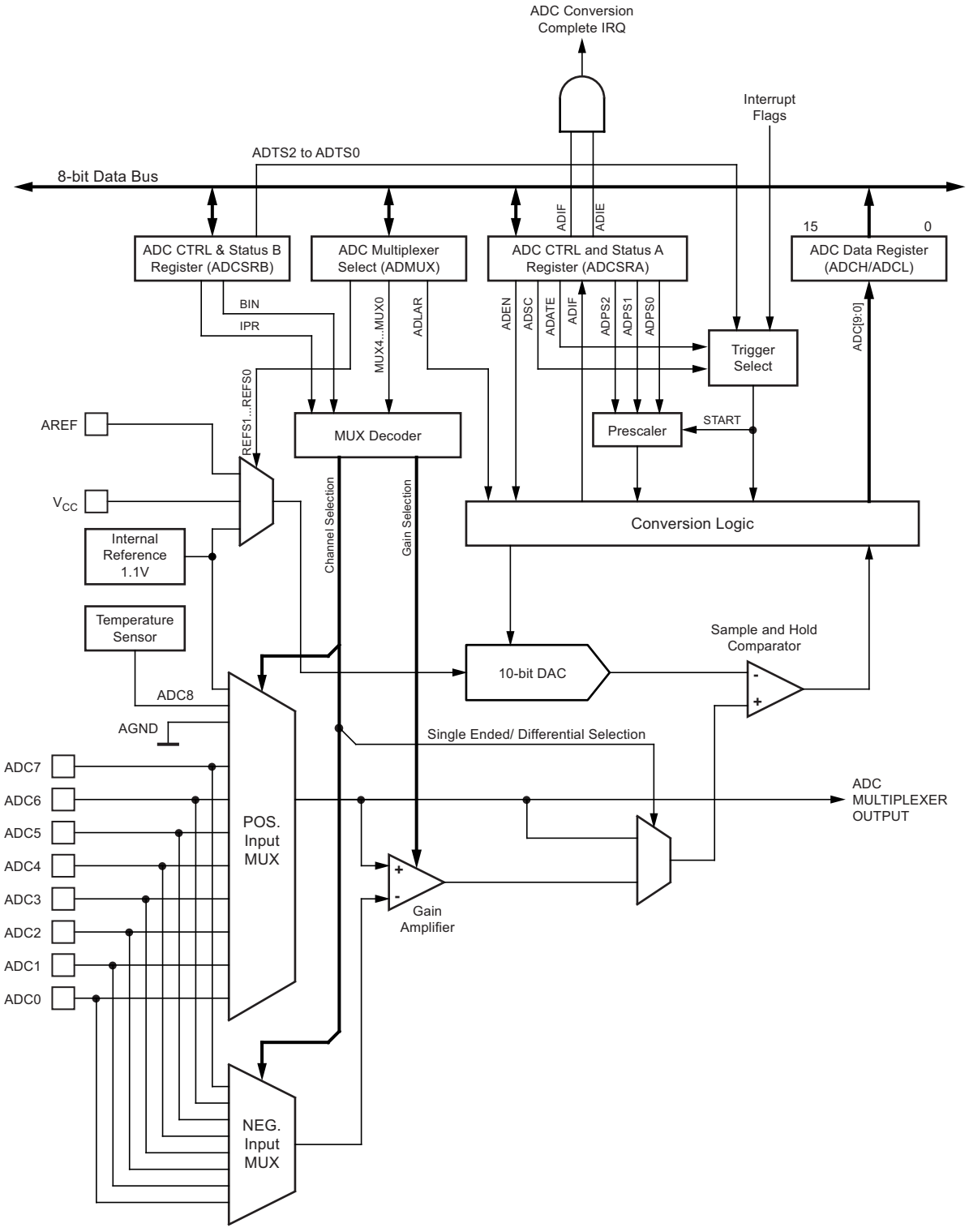
18.2 Overview

The Atmel® ATtiny24/44/84 features a 10-bit successive approximation analog-to-digital converter (ADC). The ADC is connected to 8-pin port A for external sources. In addition to external sources, the internal temperature sensor can be measured by the ADC. The analog multiplexer allows 8 single-ended channels or 12 differential channels from port A. The programmable gain stage provides amplification steps 0dB (1x) and 26dB (20x) for 12 differential ADC channels.

The ADC contains a sample-and-hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 18-1 on page 119.

And internal reference voltage of nominally 1.1V is provided on chip. Alternatively, VCC can be used as reference voltage for single-ended channels. There is also an option to use an external voltage reference and turn off the internal voltage reference.

Figure 18-1. Analog-to-Digital Converter Block Schematic



18.3 ADC Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND, and the maximum value represents the reference voltage. The voltage reference for the ADC may be selected by writing to the REFS1..0 bits in the ADMUX register. The VCC supply, the AREF pin or an internal 1.1V voltage reference may be selected as the ADC voltage reference.

The analog input channel and differential gain are selected by writing to the MUX5..0 bits in the ADMUX register. Any of the eight ADC input pins ADC7..0 can be selected as single-ended inputs to the ADC. For differential measurements, all adjacent analog inputs can be selected as an input pair. Every input can also be measured with ADC3. These pairs of differential inputs are measured by the ADC through the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input pair by the selected gain factor, 1x or 20x, according to the setting of the MUX0 bit in the ADMUX register. This amplified value then becomes the analog input to the ADC. If single-ended channels are used, the gain amplifier is bypassed altogether.

The offset of the differential channels can be measure by selecting the same input for both negative and positive input. Offset calibration can be done for ADC0, ADC3 and ADC7. When ADC0, ADC3, or ADC7 is selected as both the positive and negative input to the differential gain amplifier, the remaining offset in the gain stage and conversion circuitry can be measured directly as the result of the conversion. This figure can be subtracted from subsequent conversions with the same gain setting to reduce offset error to below 1 LSB.

The on-chip temperature sensor is selected by writing “100010” to the MUX5..0 bits in the ADMUX register.

The ADC is enabled by setting the ADC enable bit (ADEN) in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC data registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADCSRB.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the data registers belongs to the same conversion. Once ADCL is read, ADC access to data registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated, and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL registers is re-enabled.

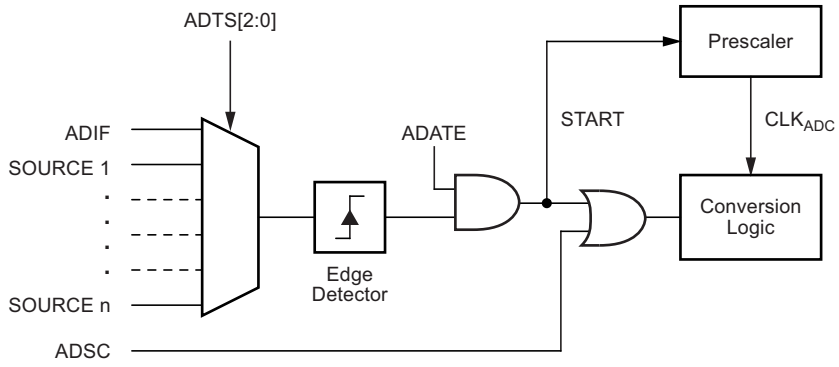
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

18.4 Starting a Conversion

A single conversion is started by writing a logical one to the ADC start conversion bit, ADSC. This bit stays high as long as the conversion is in progress, and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto triggering is enabled by setting the ADC auto trigger enable bit (ADATE) in ADCSRA. The trigger source is selected by setting the ADC trigger select bits (ADTS) in ADCSRB (see the description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the interrupt flag must be cleared in order to trigger a new conversion at the next interrupt event

Figure 18-2. ADC Auto Trigger Logic

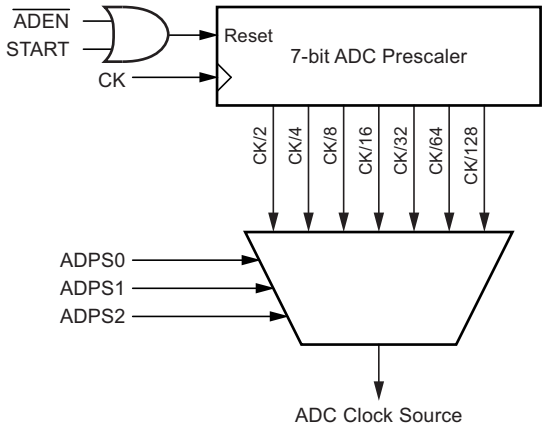


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode, the ADC will perform successive conversions independently of whether the ADC interrupt flag (ADIF) is cleared or not.

If auto triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to logical one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as logical one during a conversion independently of how the conversion was started.

18.5 Prescaling and Conversion Timing

Figure 18-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz.

The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single-ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 14.5 ADC clock cycles after the start of a first conversion. When a conversion is complete, the result is written to the ADC data registers, and ADIF is set. In single-conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When auto triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In free running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 18-1 on page 123.

Figure 18-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

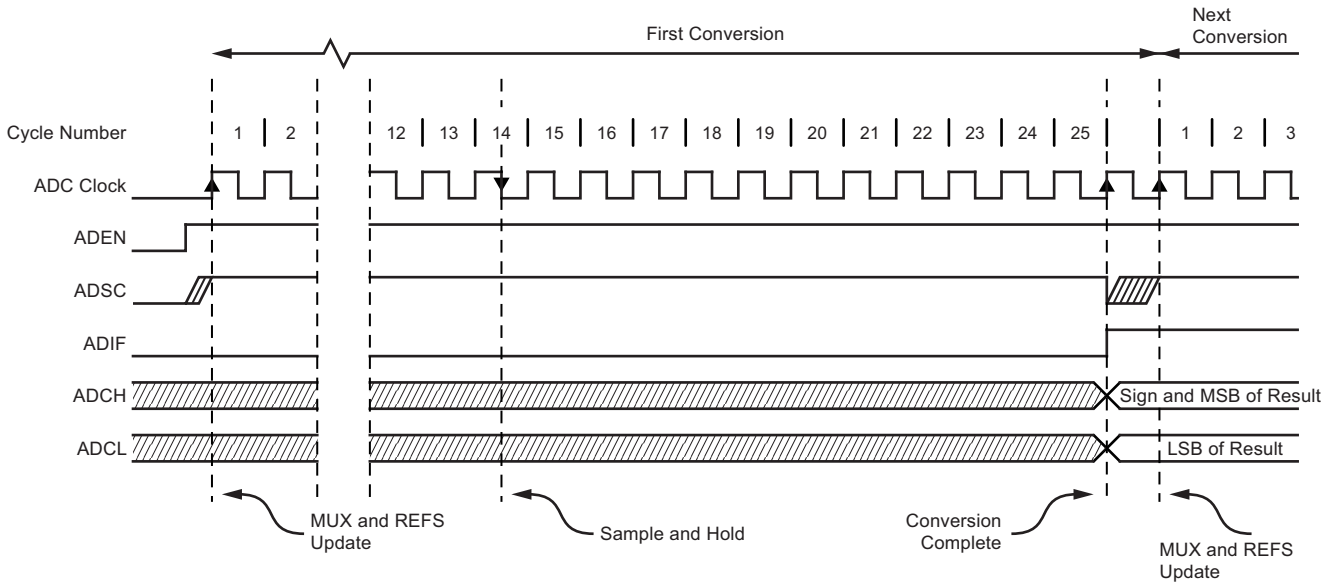


Figure 18-5. ADC Timing Diagram, Single Conversion

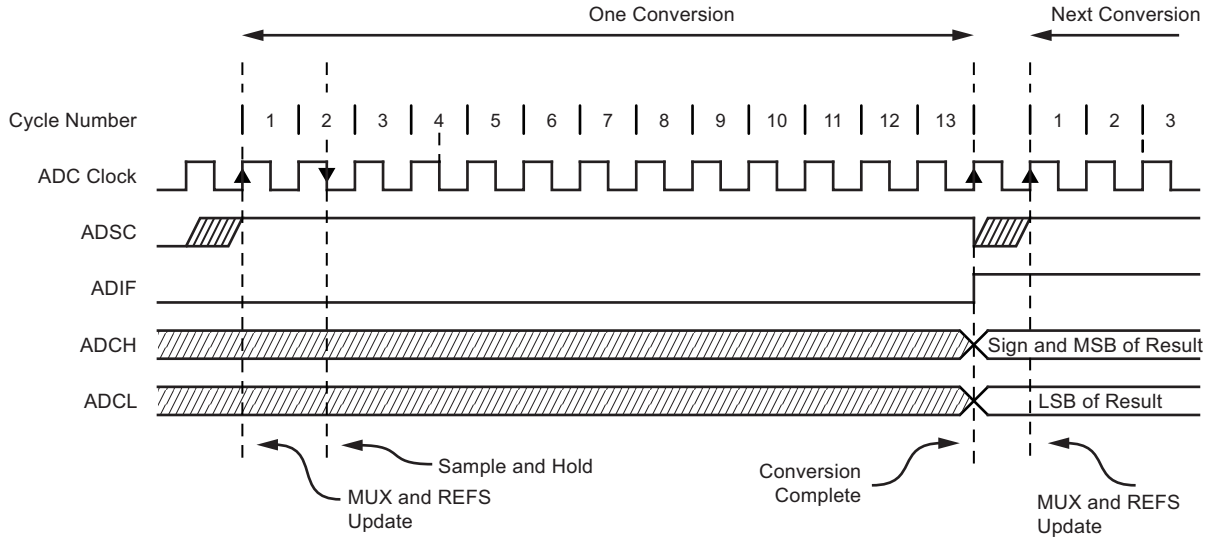


Figure 18-6. ADC Timing Diagram, Auto Triggered Conversion

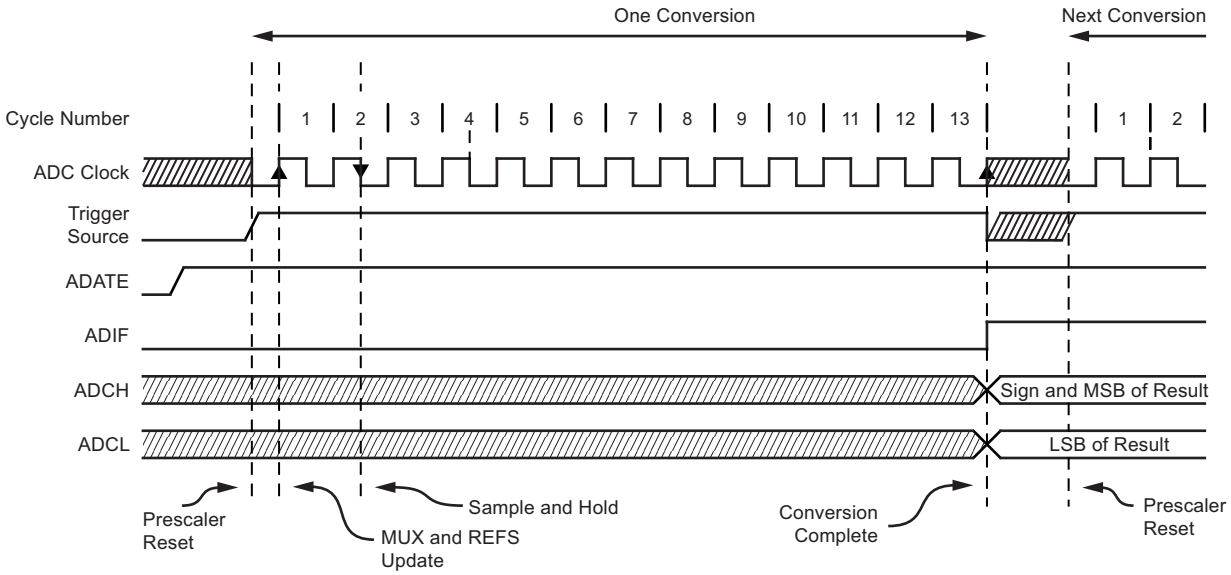


Figure 18-7. ADC Timing Diagram, Free Running Conversion

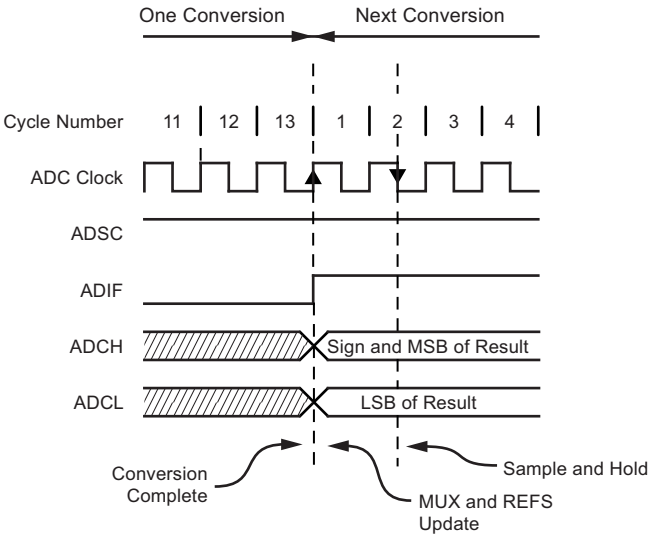


Table 18-1. ADC Conversion Time

Condition	Sample-and-Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	14.5	25
Normal conversions	1.5	13
Auto triggered conversions	2	13.5

18.6 Changing Channel or Reference Selection

The MUX5:0 and REFS1:0 bits in the ADMUX register are single-buffered through a temporary register to which the CPU has random access. This ensures that the channel and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If auto triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX register in order to control which conversion will be affected by the new setting.

If both ADATE and ADEN are written to logical one, an interrupt event can occur at any time. If the ADMUX register is changed in this period, the user cannot tell if the next conversion is based on the old or the new setting. The ADMUX register can be safely updated in the following ways:

a. When ADATE or ADEN is cleared.
b. During conversion, minimum one ADC clock cycle after the trigger event.
c. After a conversion, before the interrupt flag used as the trigger source is cleared.

When updating the ADMUX register in one of these conditions, the new setting will affect the next ADC conversion.

18.6.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In single-conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing logical one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In free running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing logical one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Because the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

18.6.2 ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single-ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either V_{CC}, internal 1.1V reference, or external AREF pin. The first ADC conversion result after switching the reference voltage source may be inaccurate, and the user is advised to discard this result.

18.7 ADC Noise Canceller

The ADC features a noise canceller that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceller can be used with ADC noise reduction and idle modes. To make use of this feature, the following procedure should be used:

a. Make sure that the ADC is enabled and is not busy converting. Single-conversion mode must be selected and the ADC conversion complete interrupt must be enabled.
b. Enter ADC noise reduction mode (or idle mode). The ADC will start a conversion once the CPU has been halted.
c. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC conversion complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC conversion complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering sleep modes other than idle mode or ADC noise reduction mode. The user is advised to write logical zero to ADEN before entering such sleep modes to avoid excessive power consumption.

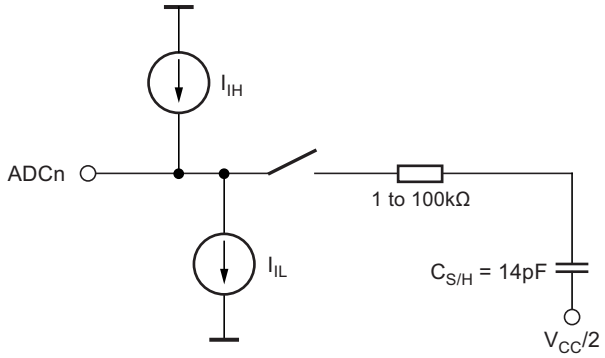
18.7.1 Analog Input Circuitry

The analog input circuitry for single-ended channels is illustrated in Figure 18-8 on page 125. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC or not. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the nyquist frequency ($f_{ADC/2}$) should not be present to avoid distortion from unpredictable signal convolution. The user is advised to remove high-frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 18-8. Analog Input Circuitry



18.7.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. Use the ADC noise canceller function to reduce induced noise from the CPU.

c. If any port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

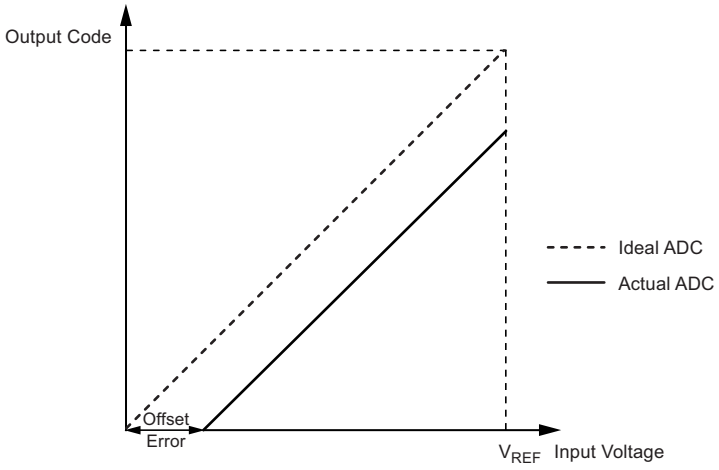
18.7.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n–1.

Several parameters describe the deviation from the ideal behavior:

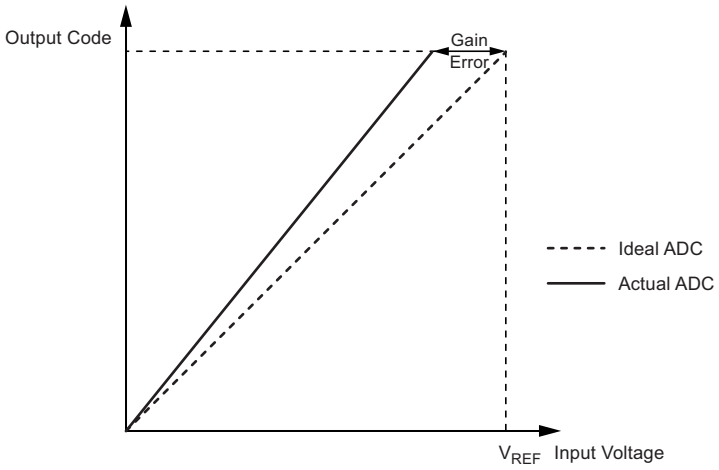
- Offset error: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5LSB). Ideal value: 0LSB.

Figure 18-9. Offset Error



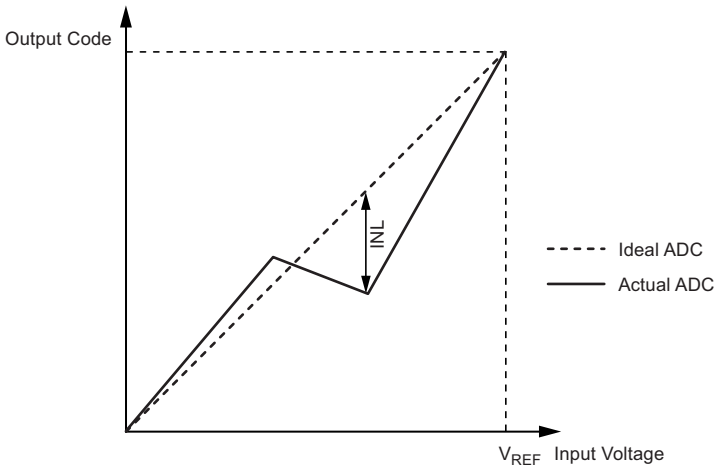
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5LSB below maximum). Ideal value: 0LSB

Figure 18-10. Gain Error



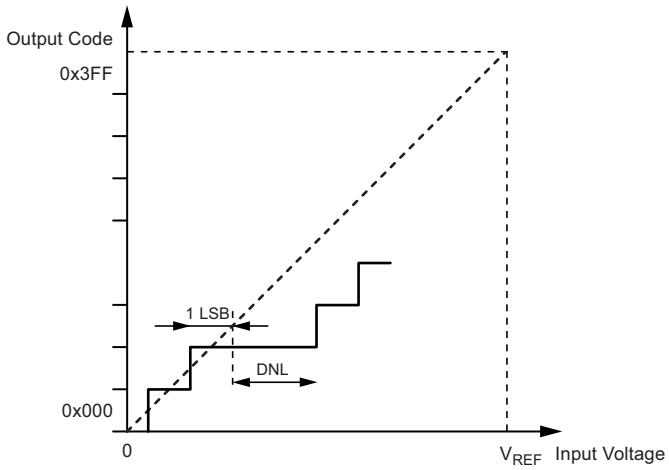
- Integral non-linearity (INL): After adjusting for offset and gain errors, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0LSB.

Figure 18-11. Integral Non-linearity (INL)



- Differential non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1LSB). Ideal value: 0LSB.

Figure 18-12. Differential Non-linearity (DNL)



- Quantization error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1LSB wide) will code to the same value. Always ±0.5LSB.
- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset error, gain error, differential error, non-linearity, and quantization error. Ideal value: ±0.5LSB.

18.8 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversion, as there are three types of conversion: single-ended conversion, unipolar differential conversion and bipolar differential conversion.

18.8.1 Single Ended Conversion

For single ended conversion, the result is:

$$ADC = \frac{V_{IN} \times 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 18-3 on page 129 and Table 18-4 on page 130). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus 1LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

18.8.2 Unipolar Differential Conversion

If differential channels and an unipolar input mode are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \times 1024}{V_{REF}} \times GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference.

The voltage of the positive pin must always be larger than the voltage of the negative pin or otherwise the voltage difference is saturated to zero. The result is presented in one-sided form, from 0x000 (0d) through 0x3FF (+1023d). The GAIN is either 1x or 20x.

18.8.3 Bipolar Differential Conversion

If differential channels and a bipolar input mode are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \times 512}{V_{REF}} \times GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (–512d) through 0x1FF (+511d). The GAIN is either 1x or 20x. Note that if the user wants to perform a quick polarity check of the result, it is sufficient to read the MSB of the result (ADC9 in ADCH). If the bit is one, the result is negative, and if this bit is zero, the result is positive.

As default the ADC converter operates in the unipolar input mode, but the bipolar input mode can be selected by writing the BIN bit in the ADCSRB to one. In the bipolar input mode two-sided voltage differences are allowed and thus the voltage on the negative input pin can also be larger than the voltage on the positive input pin.

18.9 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single-ended ADC8 channel. Selecting the ADC8 channel by writing the MUX5:0 bits in the ADMUX register to "100010" enables the temperature sensor. The internal 1.1V reference must also be selected for the ADC reference source in the temperature sensor measurement. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor. The measured voltage has a linear relationship to the temperature, as described in Table 18-2 on page 129. The voltage sensitivity is approximately 1mV/°C, and the accuracy of the temperature measurement is ±10°C after offset calibration. Bandgap is always calibrated, and its accuracy is only guaranteed between 1.0V and 1.2V.

Table 18-2. Temperature versus Sensor Output Voltage (Typical Case)

Temperature/°C	–40°C	+25°C	+85°C	+125°C
Voltage/mV	243mV	314mv	380mV	424mV

The values described in Table 18-2 are typical values. However, due to the process variation, the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results, the temperature measurement can be calibrated in the application software. The software calibration requires that a calibration value be measured and stored in a register or EEPROM for each chip as a part of the production test. The software calibration can be done utilizing the formula:

```
T = {[(ADCH << 8) | ADCL] - TOS} / k
```

where ADCn are the ADC data registers, k is a fixed coefficient and T_{OS} is the temperature sensor offset value determined and stored into EEPROM as a part of the production test.

To obtain best accuracy the coefficient k should be measured using two temperature calibrations. Using offset calibration, set k = 1.0, where k = (1024 × 1.07mV/°C) /1.1V~1.0 [1/°C].

18.10 Register Description

18.10.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	REFS1	REFS0	MUX5	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:REFS0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 18-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set).

Special care should be taken when changing differential channels. Once a differential channel has been selected, the stage may take as much as 25 ADC clock cycles to stabilize to the new value. Thus conversions should not be started within the first 13 clock cycles after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded.

The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS1:0 bits in the ADMUX register).

For channels where differential gain is used (i.e., the gain stage), using VCC or an optional external AREF higher than (VCC – 1V) is not recommended, as this will affect ADC accuracy. The internal voltage reference may not be connected to the AREF pin if an external voltage is already being applied to it. The internal voltage reference is connected to the AREF pin when REFS1:0 is set to the value “11”.

Table 18-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	V_{CC} used as analog reference, disconnected from PA0 (AREF).
0	1	External voltage reference at PA0 (AREF) pin, internal voltage reference turned off.
1	0	Internal 1.1V voltage reference.
1	1	Reserved.

- **Bits 5:0 – MUX5:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. In the case of differential input, gain selection is also made with these bits. Selections on Table 18-4 on page 130 show values for single-ended channels and where the differential channels and the offset calibration selections are located. Selecting the single-ended channel ADC8 enables the temperature measurement. See Table 18-4 on page 130 ffor details. If these bits are changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSRA is set).

Table 18-4. Single Ended Input Channel Selections.

Single Ended Input	MUX5..0
ADC0 (PA0)	000000
ADC1 (PA1)	000001
ADC2 (PA2)	000010
ADC3 (PA3)	000011
ADC4 (PA4)	000100
ADC5 (PA5)	000101
ADC6 (PA6)	000110
ADC7 (PA7)	000111
Reserved for differential channels(1)	001000 - 011111
0V (AGND)	100000
1.1V (I Ref)	100001
ADC8(2)	100010
Reserved for offset calibration(3)	100011 - 100111
Reserved for reversal differential channels(1)	101000 - 111111

Notes:
1. See Table 18-5 on page 131 for details.
2. Section 18.9 “Temperature Measurement” on page 128
3. For offset calibration only.See Table 18-5 on page 131 and Section 18.3 “ADC Operation” on page 120

See Table 18-5 on page 131 for details of selection of differential input channels as well as selection of offset calibration channels. The MUX0 bit works as a gain selection bit for differential channels, as shown in Table 18-5 on page 131. When the MUX0 bit is cleared (zero) 1x gain is selected, and when it is set (one) 20x gain is selected. For normal differential channel pairs, the MUX5 bit works as a polarity reversal bit. Toggling of the MUX5 bit reverses the positive and negative channel orientation.

For offset calibration purposes, the offset of certain differential channels can be measured by selecting the same input for both negative and positive input. This calibration can be done for ADC0, ADC3, and ADC7. Section 18.3 “ADC Operation” on page 120 describes offset calibration in a more detailed manner.

Table 18-5. Differential Input channel Selections.

Positive Differential Input	Negative Differential Input	MUX5..0	
		Gain 1x	Gain 20x
ADC0 (PA0)	ADC0 (PA0)[1]	N/A	100011
	ADC1 (PA1)	001000	001001
	ADC3 (PA3)	001010	001011
ADC1 (PA1)	ADC0 (PA0)	101000	101001
	ADC2 (PA2)	001100	001101
	ADC3 (PA3)	001110	001111
ADC2 (PA2)	ADC1 (PA1)	101100	101101
	ADC3 (PA3)	010000	010001
ADC3 (PA3)	ADC0 (PA0)	101010	101011
	ADC1 (PA1)	101110	101111
	ADC2 (PA2)	110000	110001
	ADC3 (PA3)[1]	100100	100101
	ADC4 (PA4	010010	010011
	ADC5 (PA5)	010100	010101
	ADC6 (PA6)	010110	010111
	ADC7 (PA7)	011000	011001
ADC4 (PA4	ADC3 (PA3)	110010	110011
	ADC5 (PA5)	011010	011011
ADC5 (PA5)	ADC3 (PA3)	110100	110101
	ADC4 (PA4)	111010	111011
	ADC6 (PA6)	011100	011101
ADC6 (PA6)	ADC3 (PA3)	110110	110111
	ADC5 (PA5)	111100	111101
	ADC7 (PA7)	011110	011111
ADC7 (PA7)	ADC3 (PA3)	111000	111001
	ADC6 (PA6)	111110	111111
	ADC7 (PA7)[1]	100110	100111

Note: 1. For offset calibration only.See Section 18.3 “ADC Operation” on page 120

18.10.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 – ADEN: ADC Enable**

Writing this bit to logical one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress will terminate the conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In single-conversion mode, write this bit to logical one to start each conversion. In free running mode, write this bit to logical one to start the first conversion. The first conversion after ADSC has been written and after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion initializes the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing logical zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to logical one, auto triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC trigger select bits (ADTS in ADCSRB).

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the data registers are updated. The ADC conversion complete interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if performing a read-modify-write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI instruction is used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to logical one and the I-bit in SREG is set, the ADC conversion complete interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 18-6. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

18.10.3 ADCL and ADCH – ADC Data Register

18.10.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	–	–	–	–	–	–	ADC9	ADC8	ADCH
0x04 (0x24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

18.10.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x04 (0x24)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC data register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADCSRB, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in Section 18.8 “ADC Conversion Result” on page 128.

18.10.4 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/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7 – BIN: Bipolar Input Mode**

The gain stage is working in the unipolar mode by default, but the bipolar mode can be selected by writing the BIN bit in the ADCSRB register. In the unipolar mode, only one-sided conversions are supported, and the voltage on the positive input must always be larger than the voltage on the negative input. Otherwise, the result is saturated to the voltage reference. In the bipolar mode, two-sided conversions are supported, and the result is represented in two's complement form. In unipolar mode, the resolution is 10 bits, and in bipolar mode the resolution is 9 bits + 1 sign bit.

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

See Section 17.2.1 “ADCSRB – ADC Control and Status Register B” on page 116.

- **Bit 5 – Res: Reserved Bit**

This bit is reserved bit in the Atmel® ATtiny24/44/84, and will always read as what was written there.

- **Bit 4 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC data register. Write logical 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 Section 18.10.3 “ADCL and ADCH – ADC Data Register” on page 133.

- **Bit 3 – Res: Reserved Bit**

This bit is reserved bit in the Atmel® ATtiny24/44/84, and will always read as what was written there.

- **Bits 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to logical one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected interrupt flag. Note that switching from a trigger source that is cleared to a trigger source that is set will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to free running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC interrupt flag is set.

Table 18-7. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free running mode
0	0	1	Analog comparator
0	1	0	External interrupt request 0
0	1	1	Timer/Counter0 compare match A
1	0	0	Timer/Counter0 overflow
1	0	1	Timer/Counter1 compare match B
1	1	0	Timer/Counter1 overflow
1	1	1	Timer/Counter1 capture event

18.10.5 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – ADC7D..ADC0D: ADC7..0 Digital Input Disable**

When this bit is written logical one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7..0 pin and the digital input from this pin is not needed, this bit should be written logical one to reduce power consumption in the digital input buffer.

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

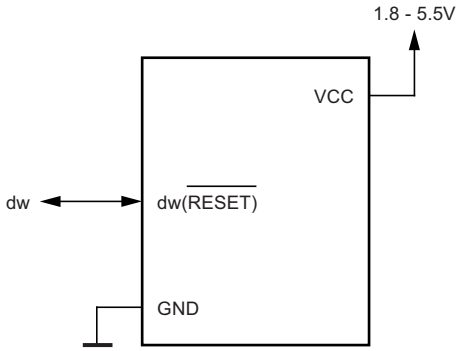


Figure 19-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

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.

20. Self-Programming the Flash

The device provides a self-programming mechanism (SPM) for downloading and uploading program code by the MCU itself. The self programming can use any available data interface and associated protocol to read code and write (program) that code into program memory.

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

Alternative 1 - fill the buffer before a page erase

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

Alternative 2 - fill the buffer after a page erase

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

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be re-written. When using alternative 1, the boot loader provides an effective read-modify-write feature, which allows the user software to first read the page and do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading because the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page.

20.1 Performing Page Erase by SPM

To execute a page erase, set up the address in the Z-pointer, write “00000011” to SPMCSR, and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 are ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

The CPU is halted during the page erase operation.

20.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a page write operation or by writing the CTPB bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM page load operation, all data loaded will be lost.

20.3 Performing a Page Write

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

The CPU is halted during the page write operation.

20.4 Addressing the Flash During Self-Programming

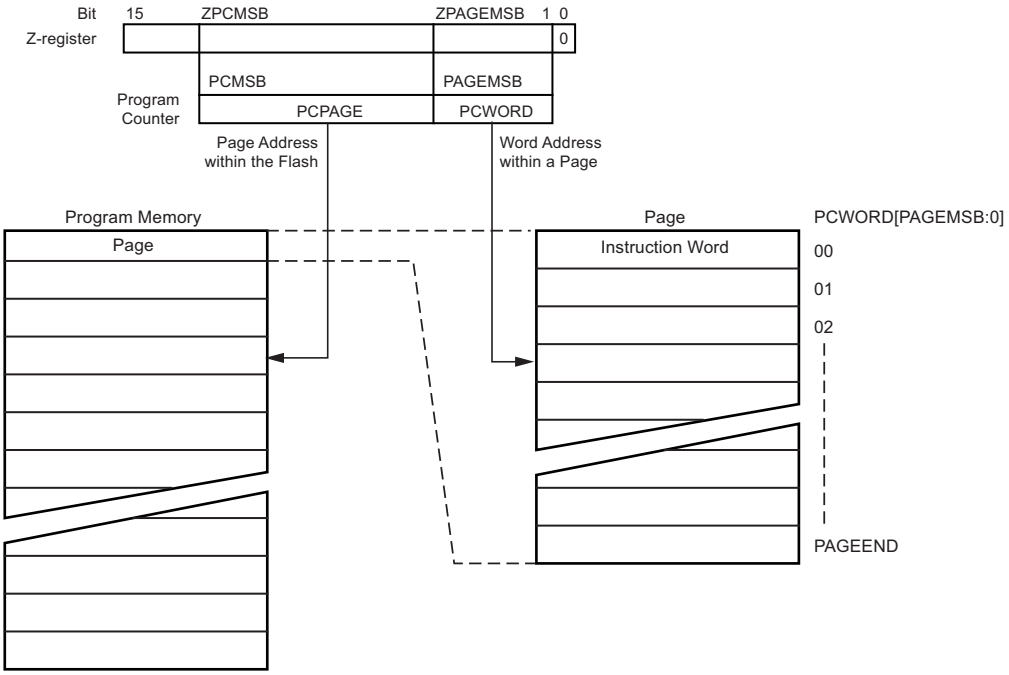
The Z-pointer is used to address the SPM commands.

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

Because the flash is organized in pages (see Table 21-7 on page 144), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 21-1 on page 144. Note that the page erase and page write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the page erase and page write operation.

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

Figure 20-1. Addressing the Flash During SPM(1)



Note: 1. The different variables used in Figure 20-1 are listed in Table 21-7 on page 144.

20.4.1 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to flash. Reading the fuses and lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user check the status bit (EEPE) in the EECR register and verify that the bit is cleared before writing to SPMCSR.

20.4.2 Reading the Lock and Fuse Bits from Software

It is possible to read both the lock and fuse bits from software. To read the lock bits, load the Z-pointer with 0x0001 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the RFLB and SPMEN bits are set in SPMCSR, the value of the lock bits will be loaded in the destination register. The RFLB and SPMEN bits will auto-clear upon completion of reading the lock bits, or if no LPM instruction is executed within three CPU cycles, or if no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared, LPM will work as described in the instruction set summary.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

The algorithm for reading the fuse low byte (FLB) is similar to the one described above for reading the lock bits. To read the fuse low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse low byte will be loaded in the destination register as shown below. See Table 21-5 on page 143 for a detailed description and mapping of the fuse low byte.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the fuse high byte (FHB), load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse high byte will be loaded in the destination register as shown below. See Table 21-4 on page 142 for detailed description and mapping of the fuse high byte.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Lock and fuse bits that are programmed will be read as zero. Lock and fuse bits that are unprogrammed, will be read as one.

20.4.3 Preventing Flash Corruption

During periods of low V_{CC}, the flash program can be corrupted because the supply voltage is too low for the CPU and the flash to operate properly. These issues are the same as for board-level systems using flash, and the same design solutions should be applied.

Flash program corruption can occur for two reasons when the voltage is too low. First, a regular write sequence to the flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly if the supply voltage is too low.

Flash corruption can easily be avoided by following at least one these design recommendations:

1. Keep the Atmel® AVR® RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal brown-out detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided the power supply voltage is sufficient.
2. Keep the AVR core in power-down sleep mode during periods of low VCC. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting SPMCSR and, thus, the flash from unintentional writes.

20.4.4 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 20-1 shows the typical programming time for flash accesses from the CPU.

Table 20-1. SPM Programming Time(1)

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7 ms	4.5 ms

Note: 1. The min and max programming times are per individual operation.

20.5 Register Description

20.5.1 SPMCSR – Store Program Memory Control and Status Register

The store program memory control and status register contains the control bits needed to control program memory operations.

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	–	–	–	CTPB	RFLB	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the Atmel®ATtiny24/44/84 and always read as zero.

- **Bit 4 – CTPB: Clear Temporary Page Buffer**

If the CTPB bit is written while filling the temporary page buffer, the temporary page buffer will be cleared and the data will be lost.

- **Bit 3 – RFLB: Read Fuse and Lock Bits**

An LPM instruction within three cycles after RFLB and SPMEN are set in SPMCSR will read either the lock bits or the fuse bits (depending on Z0 in the Z-pointer) in the destination register. See Section 20.4.1 “EEPROM Write Prevents Writing to SPMCSR” on page 138 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer.

The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to logical one at the same time as SPMEN, the next SPM instruction within four clock cycles executes a page erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page erase operation.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to logical one together with either CTPB, RFLB, PGWRT, or PGERS, the following SPM instruction will have a special meaning (see description above). If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During page erase and page write, the SPMEN bit remains high until the operation is completed.

Writing any combination other than “10001”, “01001”, “00101”, “00011”, or “00001” in the lower five bits will have no effect.

21. Memory Programming

This section describes the different methods for programming the Atmel® ATtiny24/44/84 memories.

21.1 Program And Data Memory Lock Bits

The ATtiny24/44/84 provides two lock bits which can be left unprogrammed (set to one) or can be programmed (set to zero) to obtain the additional security listed in Table 21-2. The lock bits can only be erased to one with the chip erase command.

Program memory can be read via the debugWIRE interface when the DWEN fuse is programmed even if the lock bits are set. Thus, when lock bit security is required, debugWIRE should always be disabled by clearing the DWEN fuse.

Table 21-1. Lock Bit Byte(1)

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
	5	–	1 (unprogrammed)
	4	–	1 (unprogrammed)
	3	–	1 (unprogrammed)
	2	–	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. “1” means unprogrammed, “0” means programmed.

Table 21-2. Lock Bit Protection Modes(1)(2)

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in high-voltage and serial programming mode. The fuse bits are locked in both serial and high-voltage programming mode(1). debugWire is disabled.
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in high-voltage and serial programming mode. The fuse bits are locked in both serial and high-voltage programming mode(1). debugWire is disabled.

Notes: 1. Program the fuse bits before programming LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

21.2 Fuse Bytes

The Atmel ATtiny24/44/84 has three fuse bytes. Table 21-4 to Table 21-5 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero (“0”) if they are programmed.

Table 21-3. Fuse Extended Byte

Fuse High Byte	Bit No	Description	Default Value
	7	-	1 (unprogrammed)
	6	-	1 (unprogrammed)
	5	-	1 (unprogrammed)
	4	-	1 (unprogrammed)
	3	-	1 (unprogrammed)
	2	-	1 (unprogrammed)
	1	-	1 (unprogrammed)
SELFPRGEN	0	Self programming enable	1 (unprogrammed)

Table 21-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL[1]	7	External reset disable	1 (unprogrammed)
DWEN[2]	6	DebugWIRE enable	1 (unprogrammed)
SPIEN[3]	6	Enable serial program and data downloading	0 (programmed, SPI prog. enabled)
WDTON[4]	4	Watchdog timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the chip erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2[5]	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1[5]	1	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL0[5]	0	Brown-out detector trigger level	1 (unprogrammed)

Notes:
1. See Section 12.3.2 “Alternate Functions of Port B” on page 59 for description of RSTDISBL and DWEN fuses. When programming the RSTDISBL fuse, high-voltage Serial programming has to be used to change fuses to perform further programming
2. DWEN must be unprogrammed when lock Bit security is required. See Section 21.1 “Program And Data Memory Lock Bits” on page 141.
3. The SPIEN Fuse is not accessible in SPI Programming mode.
4. See Section 9-2 “WDT Configuration as a Function of the Fuse Settings of WDTON” on page 40 for details.
5. See Table 22-5 on page 158 for BODLEVEL fuse decoding.

Table 21-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8[1]	7	Divide clock by 8	0 (programmed)
CKOUT	6	Clock Output Enable	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[2]
SUT0	4	Select start-up time	0 (programmed)[2]
CKSEL3	3	Select Clock source	0 (programmed)[3]
CKSEL2	2	Select Clock source	0 (programmed)[3]
CKSEL1	1	Select Clock source	1 (unprogrammed)[3]
CKSEL0	0	Select Clock source	0 (programmed)[3]

Notes:
1. See Section 7.9 “System Clock Prescaler” on page 29 for details.
2. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 7-7 on page 27 for details.
3. The default setting of CKSEL3..0 results in internal RC oscillator at 8.0MHz. See Table 7-6 on page 27 for details.

The status of the fuse bits is not affected by chip erase. Note that the fuse bits are locked if lock bit 1 (LB1) is programmed. Program the fuse bits before programming the lock bits.

21.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode, and changes in the fuse values will have no effect until the part leaves programming mode. This does not apply to the EESAVE fuse, which will take effect once it is programmed. The fuses are also latched on power-up in normal mode.

21.3 Signature Bytes

All Atmel® microcontrollers have a three-byte signature code that identifies the device. This code can be read in both serial and high-voltage programming mode, even when the device is locked. The three bytes reside in a separate address space. For the Atmel ATtiny24/44/84, the signature bytes are given in Table 21-6.

Table 21-6. Device ID

Device	Signature Bytes Address		
	0x000	0x001	0x002
ATtiny24	0x1E	0x91	0x0B
ATtiny44	0x1E	0x92	0x07
ATtiny84	0x1E	0x93	0x0C

21.4 Calibration Byte

The signature area of the Atmel ATtiny24/44/84 has one byte of calibration data for the internal RC oscillator. This byte resides in the high byte of address 0x000. During reset, this byte is automatically written into the OSCCAL register to ensure the correct frequency of the calibrated RC oscillator.

21.5 Page Size

Table 21-7. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATtiny24	1Kwords (2Kbytes)	16 words	PC[3:0]	64	PC[9:4]	9
ATtiny44	2Kwords (4Kbytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATtiny84	4Kwords (8Kbytes)	32 words	PC[4:0]	128	PC[11:5]	11

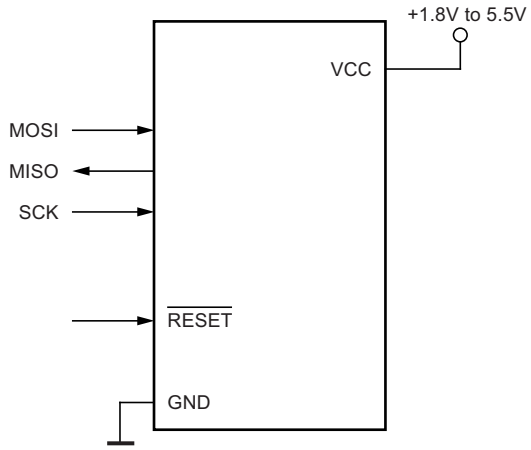
Table 21-8. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATtiny24	128 bytes	4 bytes	EEA[1:0]	32	EEA[6:2]	6
ATtiny44	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7
ATtiny84	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

21.6 Serial Downloading

Both the flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the programming enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 21-9, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 21-1. Serial Programming and Verify(1)



Note: 1. If the device is clocked by the internal oscillator, it is not needed to connect a clock source to the CLKI pin.

Table 21-9. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PA6	I	Serial data in
MISO	PA5	O	Serial data out
SCK	PA4	I	Serial clock

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in serial mode only), and there is no need to first execute the chip erase instruction. The chip erase operation turns the content of every memory location in both the program and EEPROM arrays into 0xFF.

Depending on the CKSEL fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for f_{ck} < 12MHz, 3 CPU clock cycles for f_{ck} ≥ 12MHz

High: > 2 CPU clock cycles for f_{ck} < 12MHz, 3 CPU clock cycles for f_{ck} ≥ 12MHz

21.6.1 Serial Programming Algorithm

When writing serial data to the Atmel® AVR® ATtiny24/44/84, data are clocked on the rising edge of SCK.

When reading data from the Atmel ATtiny24/44/84, data are clocked on the falling edge of SCK. See Figure 22-3 on page 161 and Figure 22-4 on page 161 for timing details.

To program and verify the Atmel ATtiny24/44/84 in the serial programming mode, the following sequence is recommended (see four-byte instruction formats in Table 21-11):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync, the second byte (0x53) will echo back when issuing the third byte of the programming enable instruction. Regardless of whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new programming enable command.
4. The flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the five LSBs of the address and data together with the load program memory page instruction. To ensure correct loading of the page, the data low byte must be loaded before the data high byte is applied for a given address.
   The program memory page is stored by loading the write program memory page instruction with the three MSBs of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least tWD_FLASH before issuing the next page. (See Table 21-10 on page 146.) Accessing the serial programming interface before the flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate write instruction. An EEPROM memory location is first automatically erased before new data are written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 21-10 on page 146.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The memory page is loaded one byte at a time by supplying the two LSBs of the address and data together with the Load EEPROM memory page instruction. The EEPROM memory page is stored by loading the Write EEPROM memory page instruction with the four MSBs of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM memory page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 21-10 on page 146). In a chip-erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

Table 21-10. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	4.0ms
t_{WD_ERASE}	4.0ms
t_{WD_FUSE}	4.5ms

21.6.2 Serial Programming Instruction set

Table 21-11 and Figure 21-2 on page 148 describes the Instruction set.

Table 21-11. Serial Programming Instruction Set

Instruction/Operation[1]	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming enable	$AC	$53	$00	$00
Chip erase (program memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{BSY}$	$F0	$00	$00	data byte out
Load Instructions				
Load extended address byte	$4D	$00	Extended adr	$00
Load program memory page, high byte	$48	adr MSB	adr LSB	high data byte in
Load program memory page, low byte	$40	adr MSB	adr LSB	low data byte in
Load EEPROM memory Page (page access)	$C1	$00	adr LSB	data byte in
Read Instructions				
Read program memory, high byte	$28	adr MSB	adr LSB	high data byte out
Read program memory, low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM memory	$A0	$00	adr LSB	data byte out
Read lock bits	$58	$00	$00	data byte out
Read signature byte	$30	$00	adr LSB	data byte out
Read fuse bits	$50	$00	$00	data byte out
Read fuse high bits	$58	$08	$00	data byte out
Read extended fuse bits	$50	$08	$00	data byte out
Read calibration byte	$38	$00	$00	data byte out

Notes:
1. Not all instructions are applicable for all parts.
2. adr = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused fuses and lock bits should be unprogrammed (‘1’).
5. Refer to the corresponding section for fuse and lock bits, calibration and signature bytes and page size.
6. Instructions accessing program memory use a word address. This address may be random within the page range.
7. See http://www.atmel.com/avr for application notes regarding programming and programmers.

Table 21-11. Serial Programming Instruction Set (Continued)

Instruction/Operation[1]	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Write Instructions[6]				
Write program memory page	$4C	adr MSB	adr LSB	$00
Write EEPROM memory	$C0	$00	adr LSB	data byte in
Write EEPROM memory Page (page access)	$C2	$00	adr LSB	$00
Write lock bits	$AC	$E0	$00	data byte in
Write fuse bits	$AC	$A0	$00	data byte in
Write fuse high bits	$AC	$A8	$00	data byte in
Write extended fuse bits	$AC	$A4	$00	data byte in

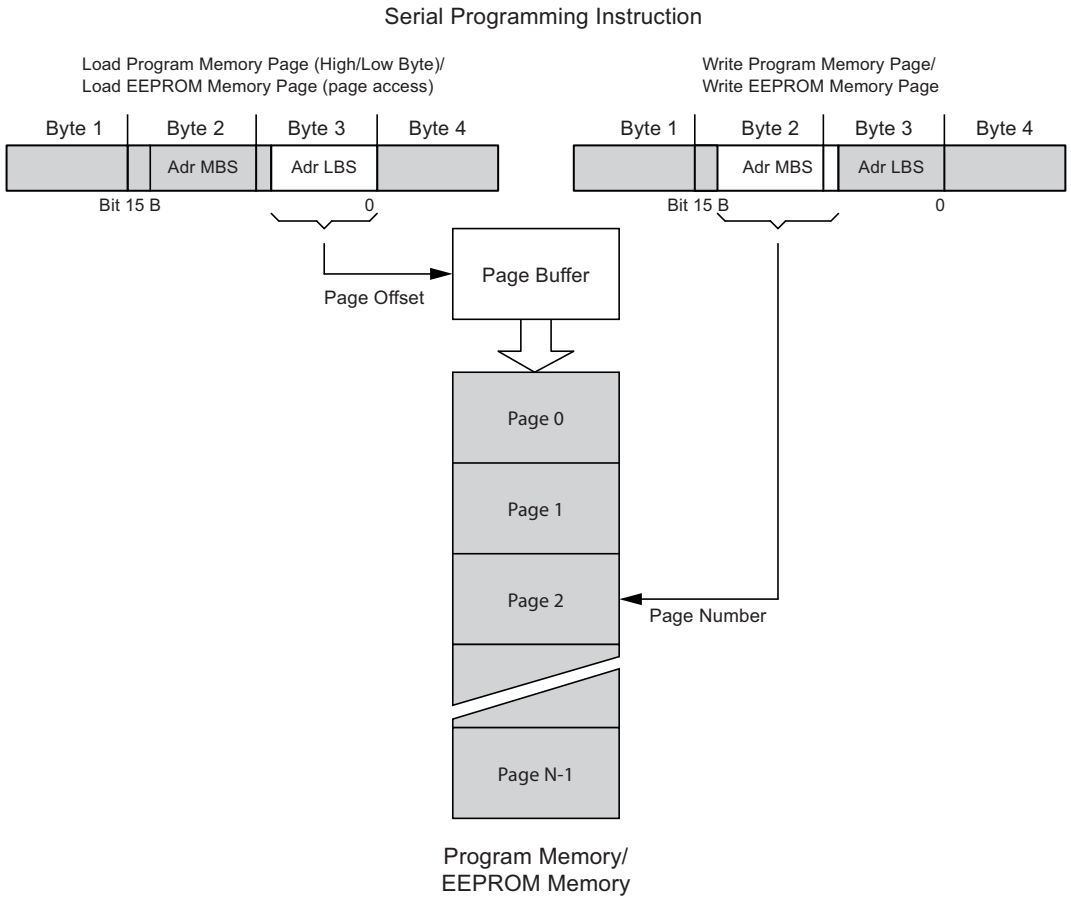
Notes:
1. Not all instructions are applicable for all parts.
2. adr = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused fuses and lock bits should be unprogrammed (‘1’).
5. Refer to the corresponding section for fuse and lock bits, calibration and signature bytes and page size.
6. Instructions accessing program memory use a word address. This address may be random within the page range.
7. See http://www.atmel.com/avr for application notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data are loaded to the page buffer, program the EEPROM page (see Figure 21-2 on page 148).

Figure 21-2. Serial Programming Instruction Example



21.7 High-voltage Serial Programming

This section describes how to program and verify flash program memory, EEPROM data memory, lock bits and fuse bits in the Atmel® ATtiny24/44/84.

Figure 21-3. High-voltage Serial Programming

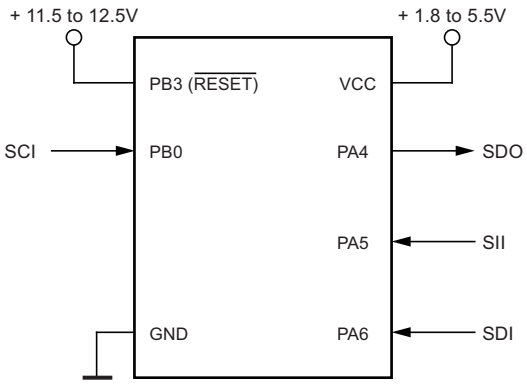


Table 21-12. Pin Name Mapping

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.

21.8.3 Chip Erase

The chip erase will erase the flash and EEPROM[1] memories, as well as 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: 1. The EEPROM memory is preserved during chip erase if the EESAVE fuse is programmed.

1. Load “chip erase” command (see Table 21-15 on page 152).
2. Wait after Instr. 3 until SDO goes high for the “chip erase” cycle to finish.
3. Load “no operation” command.

21.8.4 Programming the Flash

The flash is organized in pages, see Section 21.5 “Page Size” on page 144. When programming the flash, the program data are 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:

1. Load “write flash” command (see Table 21-15 on page 152).
2. Load flash page buffer.
3. Load flash high address and program page. Wait after Instr. 3 until SDO goes high for the “page programming” cycle to finish.
4. Repeat 2 and 3 until the entire flash is programmed, or until all data has been programmed.
5. End page programming by loading “no operation” command.

When writing or reading serial data to the Atmel® ATtiny24/44/84, data are clocked on the rising edge of the serial clock. See Figure 22-5 on page 162, Figure 21-3 on page 148 and Table 22-9 on page 162 for details.

Figure 21-4. Addressing the Flash which is Organized in Pages

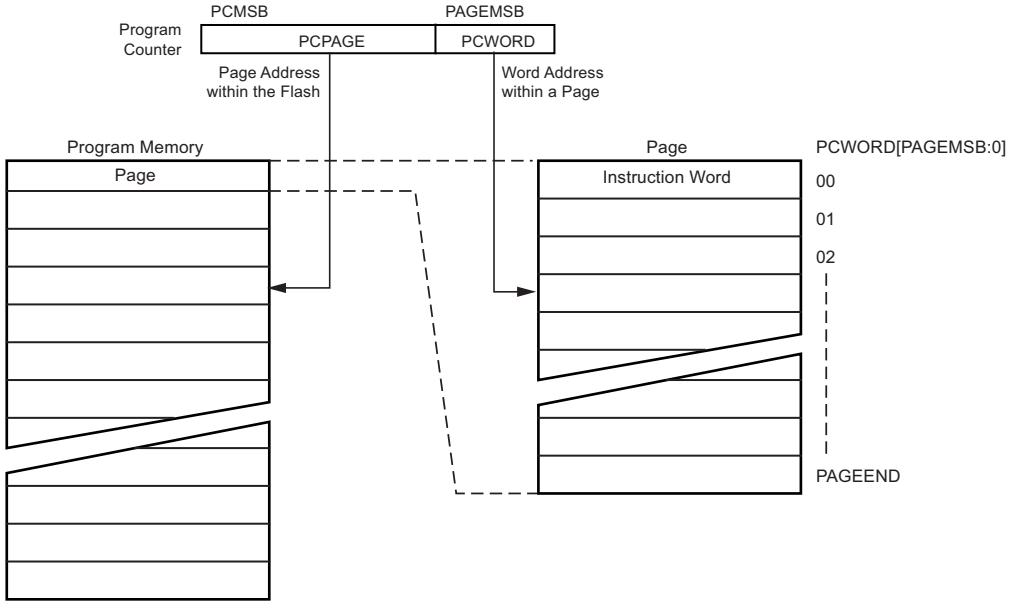
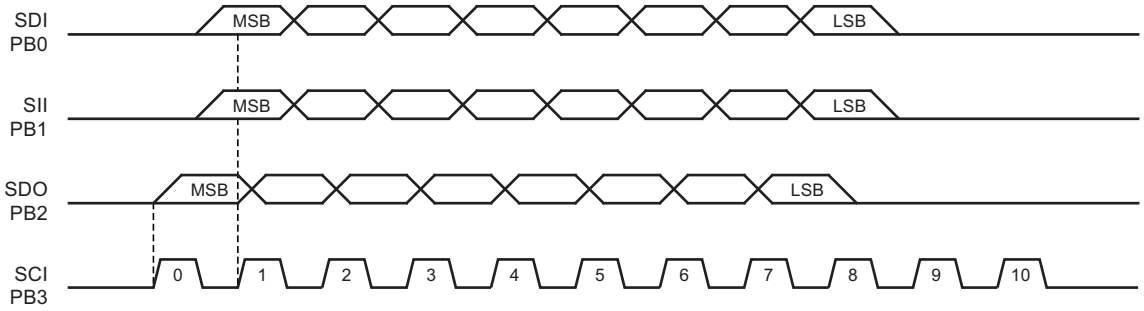


Figure 21-5. High-voltage Serial Programming Waveforms



21.8.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 22-8 on page 161. When programming the EEPROM, the data are latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to Table 21-15 on page 152):

1. Load “Write EEPROM” command.
2. Load EEPROM page buffer.
3. Program EEPROM page. Wait after Instr. 2 until SDO goes high for the page programming cycle to finish.
4. Repeat 2 and 3 until the entire EEPROM is programmed, or until all data has been programmed.
5. End page programming by loading “no operation” command.

21.8.6 Reading the Flash

The algorithm for reading the flash memory is as follows (refer to Table 21-15 on page 152):

1. Load “read flash” command.
2. Read flash low and high bytes. The contents at the selected address are available at serial output SDO.

21.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Table 21-15 on page 152):

1. Load “read EEPROM” command.
2. Read EEPROM byte. The contents at the selected address are available at serial output SDO.

21.8.8 Programming and Reading the Fuse and Lock Bits

The algorithms for programming and reading the fuse low/high bits and lock bits are shown in Table 21-15 on page 152.

21.8.9 Reading the Signature Bytes and Calibration Byte

The algorithms for reading the signature bytes and calibration byte are shown in Table 21-15 on page 152.

21.8.10 Power-off sequence

Set SCI to “0”. Set RESET to “1”. Turn V_{CC} power off.

Table 21-15. High-voltage Serial Programming Instruction Set for ATtiny24/44/84

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3/7	Instr.4	
Chip erase	SDI SII SDO	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		Wait after Instr.3 until SDO goes high for the chip erase cycle to finish.
Load “write flash” Command	SDI SII SDO	0_0001_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter flash programming code.
Load flash Page buffer	SDI SII SDO	0_ **bbbb_bbbb** _00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Repeat after Instr. 1 - 7until the entire page buffer is filled or until all data within the page is filled(1).
	SDI SII SDO	0_**dddd_dddd**_00 0_0011_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx		Instr 5-7.
Load flash High address and program page	SDI SII SDO	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		Wait after Instr 3 until SDO goes high. Repeat Instr. 2 - 3 for each loaded flash page until the entire flash or all data are programmed. Repeat Instr. 1 for a new 256-byte page(1).
Load “read flash” command	SDI SII SDO	0_0000_0010_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter flash read mode.
Read flash Low and high bytes	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	Repeat Instr. 1, 3 - 6 for each new address. Repeat Instr. 2 for a new 256 byte page.
	SDI SII SDO	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx			Instr 5 - 6.
Load “write EEPROM” Command	SDI SII SDO	0_0001_0001_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter EEPROM programming mode.
Load EEPROM Page buffer	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	Repeat Instr. 1 - 5 until the entire page buffer is filled or until all data within the page is filled(2).
	SDI SII SDO	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx				
Program EEPROM page	SDI SII SDO	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx			Wait after Instr. 2 until SDO goes high. Repeat Instr. 1 - 2 for each loaded EEPROM page until the entire EEPROM or all data is programmed.

a = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits, x = don’t care, 1 = Lock Bit1, ***2*** = Lock Bit2, **3** = CKSEL0 Fuse, **4** = CKSEL1 Fuse, **5** = CKSEL2 Fuse, **6** = CKSEL3 Fuse, **7** = SUT0 Fuse, **8** = SUT1 Fuse, **9** = CKDIV8 Fuse, **A** = CKOUT Fuse, **B** = BODLEVEL0 Fuse, **C** = BODLEVEL1 Fuse, **D**= BODLEVEL2 Fuse, **E** = EESAVE Fuse, **F** = WDTON Fuse, **G** = SPIEN Fuse, **H** = DWEN Fuse, **I** = RSTDISBL Fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.
3. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in high-voltage serial programming, only in SPI programming.

Table 21-15. High-voltage Serial Programming Instruction Set for ATtiny24/44/84 (Continued)

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3/7	Instr.4	
Write EEPROM byte	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	Repeat Instr. 1 - 6 for each new address. Wait after Instr. 6 until SDO goes high. (3)
	SDI SII SDO	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx			Instr. 5-6
Load “read EEPROM” Command	SDI SII SDO	0_0000_0011_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter EEPROM read mode.
Read EEPROM byte	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**0_00	Repeat Instr. 1, 3 - 4 for each new address. Repeat Instr. 2 for a new 256-byte page.
Write fuse Low bits	SDI SII SDO	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**A987_6543**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **A - 3** = “0” to program the fuse bit.
Write fuse High bits	SDI SII SDO	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**IHGF_EDCB**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **F - B** = “0” to program the fuse bit.
Write fuse Extended bits	SDI SII SDO	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_000**J**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0110_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1110_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **J** = “0” to program the fuse bit.
Write lock bits	SDI SII SDO	0_0010_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**21**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **2 - 1** = “0” to program the lock bit.
Read fuse Low bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **A_9876_543**x_xx		Reading **A - 3** = “0” means the fuse bit is programmed.
Read fuse High bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1010_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **I_HGFE_DCB**x_xx		Reading **F - B** = “0” means the fuse bit is programmed.
Read fuse Extended bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1010_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1110_00 x_xxxx_xxJx_xx		Reading **J** = “0” means the fuse bit is programmed.
Read lock bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_x**21**x_xx		Reading **2, 1** = “0” means the lock bit is programmed.

a = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits, x = don’t care, 1 = Lock Bit1, ***2*** = Lock Bit2, **3** = CKSEL0 Fuse, **4** = CKSEL1 Fuse, **5** = CKSEL2 Fuse, **6** = CKSEL3 Fuse, **7** = SUT0 Fuse, **8** = SUT1 Fuse, **9** = CKDIV8 Fuse, **A** = CKOUT Fuse, **B** = BODLEVEL0 Fuse, **C** = BODLEVEL1 Fuse, **D**= BODLEVEL2 Fuse, **E** = EESAVE Fuse, **F** = WDTON Fuse, **G** = SPIEN Fuse, **H** = DWEN Fuse, **I** = RSTDISBL Fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.
3. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in high-voltage serial programming, only in SPI programming.

Table 21-15. High-voltage Serial Programming Instruction Set for ATtiny24/44/84 (Continued)

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3/7	Instr.4	
Read signature bytes	SDI SII SDO	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**bb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	Repeats Instr 2 4 for each signature byte address.
Read calibration byte	SDI SII SDO	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx	
Load “no operation” command	SDI SII SDO	0_0000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				

a = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits, x = don’t care, 1 = Lock Bit1, ***2*** = Lock Bit2, **3** = CKSEL0 Fuse, **4** = CKSEL1 Fuse, **5** = CKSEL2 Fuse, **6** = CKSEL3 Fuse, **7** = SUT0 Fuse, **8** = SUT1 Fuse, **9** = CKDIV8 Fuse, **A** = CKOUT Fuse, **B** = BODLEVEL0 Fuse, **C** = BODLEVEL1 Fuse, **D**= BODLEVEL2 Fuse, **E** = EESAVE Fuse, **F** = WDTON Fuse, **G** = SPIEN Fuse, **H** = DWEN Fuse, **I** = RSTDISBL Fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.
3. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in high-voltage serial programming, only in SPI programming.

22. Electrical Characteristics

22.1 Absolute Maximum Ratings

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Min.	Typ.	Max.	Unit
Automotive operating temperature	–40		+125	°C
Storage temperature	–65		+150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	–0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to GND	–0.5		+13.0	V
Voltage on V_{CC} with respect to GND	–0.5	6.0		V
DC current per I/O pin		40.0		mA
DC current V_{CC} and GND pins		200.0		mA
Injection current at V_{CC} = 0V to 5V(2)		±5		mA(1)

Notes: 1. Maximum current per port = ±30mA
2. Functional corruption may occur

Table 22-1. DC Characteristics T_A = –40°C to 125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)(1)

Parameter	Condition	Symbol	Min.	Typ.	Max.	Units
Input low voltage	V_{CC} = 2.4V to 5.5V	V_{IL}	–0.5		$0.3V_{CC}$	V
Input high-voltage Except $\overline{\text{RESET}}$ pin	V_{CC} = 2.4V to 5.5V	V_{IH}	$0.6V_{CC}$(3)		V_{CC} + 0.5(2)	V
Input high-voltage $\overline{\text{RESET}}$ pin		V_{IH2}	$0.9V_{CC}$(3)		V_{CC} + 0.5(2)	V
Output low voltage(4) (port B,PORTA)	I_{OL} = 10mA, V_{CC} = 5V I_{OL} = 5mA, V_{CC} = 3V	V_{OL}			0.8 0.5	V V
Output high-voltage(5) (Port B2:0,PORTA)	I_{OH} = –10mA, V_{CC} = 5V I_{OH} = –5mA, V_{CC} = 3V	V_{OH}	4.3 2.5			V V
Input leakage Current I/O pin	V_{CC} = 5.5V, pin low (absolute value)	$I_{ILPORTA}$			50	nA
Input leakage Current I/O pin	V_{CC} = 5.5V, pin high (absolute value)	$I_{IHPORTA}$			50	nA

Notes: 1. All DC Characteristics contained in this data sheet are based on actual silicon characterization of Atmel® ATtiny24/44/84 AVR® microcontrollers manufactured in corner run process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual Automotive silicon.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. “Min” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
1] The sum of all IOL, for all ports, should not exceed 60mA.
If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
1] The sum of all IOH, for all ports, should not exceed 60mA.
If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition. Pull up driving strength of the PB3 $\overline{\text{RESET}}$ pad is weak.

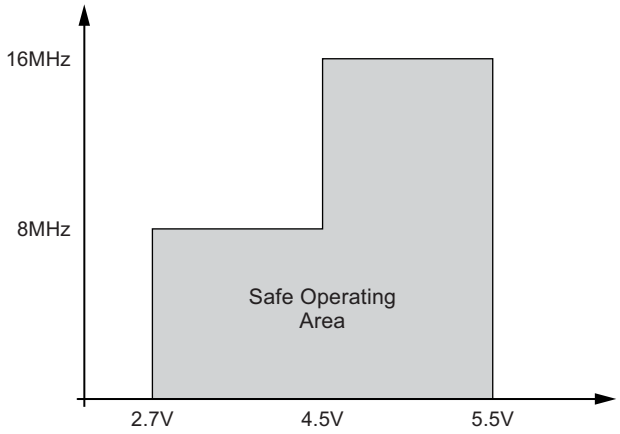
Table 22-1. DC Characteristics T_A = –40°C to 125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)(1) (Continued)

Parameter	Condition	Symbol	Min.	Typ.	Max.	Units
Input leakage Current I/O pin	Vcc = 5.5V, pin low (absolute value)	$I_{IHPORTB}$		< 0.05	1	µA
Input leakage Current I/O pin	Vcc = 5.5V, pin high (absolute value)	$I_{ILPORTB}$		< 0.05	1	µA
Reset pull-up resistor		R_{RST}	30		60	kΩ
I/O pin pull-up resistor		R_{pu}	20		50	kΩ
Power supply current	Active 1MHz, V_{CC} = 3V	I_{CC}		0.4	1.5	mA
	Active 4MHz, V_{CC} = 3V			1.8	3.0	mA
	Active 8MHz, V_{CC} = 5V			5.0	10.0	mA
	Idle 1MHz, V_{CC} = 3V			0.075	0.2	mA
	Idle 4MHz, V_{CC} = 3V			0.3	0.5	mA
	Idle 8MHz, V_{CC} = 5V			1.2	2.5	mA
Power-down mode	WDT enabled, V_{CC} = 3V			5.0	30	µA
	WDT enabled, V_{CC} = 5V			9.0	50	µA
	WDT disabled, V_{CC} = 3V			2.5	24	µA
	WDT disabled, V_{CC} = 5V			4.3	36	µA
Analog comparator input Leakage current	V_{CC} = 5V V_{in} = V_{CC}/2	I_{ACLK}	–50		50	nA

Notes:
1. All DC Characteristics contained in this data sheet are based on actual silicon characterization of Atmel® ATtiny24/44/84 AVR® microcontrollers manufactured in corner run process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual Automotive silicon.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. “Min” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition. Pull up driving strength of the PB3 $\overline{\text{RESET}}$ pad is weak.

22.2 Speed Grades

Figure 22-1. Maximum Frequency versus V_{CC}



22.3 Clock Characterizations

22.3.1 Calibrated Internal RC Oscillator Accuracy

Table 22-2. Calibration Accuracy of Internal RC Oscillator

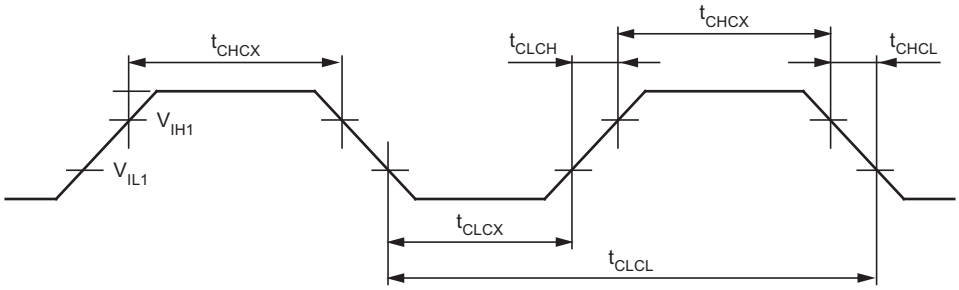
	Frequency	V_{CC}	Temperature	Accuracy
Factory calibration	8.0MHz	3V	25°C	±2%
User calibration	7.3 to 8.1MHz	2.7V to 5.5V	–40°C to 125°C	±20%
Oscillator jitter	8.0MHz	2.7V to 5.5V	–40°C to 125°	Standard deviation 0.4ns(1)

Note: 1. The overall jitter increase proportionally to the divider ratio

Example: With oscillator divided by 32, jitter standard deviation will be 32 × 0.4ns = 12.8ns.

22.3.2 External Clock Drive Waveforms

Figure 22-2. External Clock Drive Waveforms



22.3.3 External Clock Drive

Table 22-3. External Clock Drive

Parameter	Symbol	V_{CC} = 2.7 to 5.5V		V_{CC} = 4.5 to 5.5V		Units
		Min.	Max.	Min.	Max.	
Clock frequency	$1/t_{CLCL}$	0	10	0	20	MHz
Clock period	t_{CLCL}	100		50		ns
High time	t_{CHCX}	40		20		ns
Low time	t_{CLCX}	40		20		ns
Rise time	t_{CLCH}		1.6		0.5	µs
Fall time	t_{CHCL}		1.6		0.5	µs
Change in period from one clock cycle to the next	Δt_{CLCL}		2		2	%

22.4 System and Reset Characterizations

Table 22-4. Reset, Brown-out and Internal Voltage Reference Characteristics(1)

Parameter	Condition	Symbol	Min	Typ	Max	Units
Brown-out detector hysteresis		V_{HYST}		100	250	mV
RAM retention voltage(1)		V_{RAM} 2.	50		mV	
Min pulse width on brown-out reset		t_{BOD}		2		ns
Bandgap reference voltage	$V_{C\,C}$ = 2.7V, T_A = 25°C	V_{BG}	1.0	1.1	1.2	V
Bandgap reference start-up time	$V_{C\,C}$ = 2.7V, T_A = 25°C	t_{BG}		40	70	µs
Bandgap reference current consumption	$V_{C\,C}$ = 2.7V, T_A = 25°C	I_{BG}		10		µA

Notes: 1. Values are guidelines only.
2. This is the limit to which VDD can be lowered without losing RAM data

Table 22-5. BODLEVEL Fuse Coding(1)

BODLEVEL [2..0] Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	BOD disabled			
110		1.8		V
101	2.5	2.7	2.9	
100	4.0	4.3	4.6	
011		2.3		
010		2.2		
001		1.9		
000		2.0		

Note: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to V_{CC} = V_{BOT} during the production test. This guarantees that a brown-out reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed.

22.5 ADC Characteristics – Preliminary Data

Table 22-6. ADC Characteristics, Single Ended Channels. –40°C to +125°C

Parameter	Condition	Symbol	Min	Typ	Max	Units
Resolution	Single-ended conversion			10		Bits
Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	Single-ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz	TUE		2.0	4.0	LSB
	Single-ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz Noise reduction mode			2.0	4.0	LSB
Integral non-linearity (INL)	Single-ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz	INL		0.5	1.5	LSB
Differential non-linearity (DNL)	Single-ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz	DNL		0.3	0.7	LSB
Gain error	Single-ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		–5.0	–3.0	+5.0	LSB
Offset error	Single-ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		–3.5	+1.5	+3.5	LSB
Conversion time	Free running conversion		65		260	µs
Clock frequency			50		200	kHz
External voltage reference		V_{REF}	2.56		AVCC	V
Input voltage		V_{IN}	GND		V_{REF}	V
Internal voltage reference		V_{INT}	1.0	1.1	1.2	V
Analog input resistance		R_{AIN}		100		MΩ

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

Parameter	Condition	Symbol	Min	Typ	Max	Units
Resolution	Gain = 1x			8		Bits
	Gain = 20x			8		Bits
Absolute accuracy	Gain = 1x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz	TUE		2.5	5.0	LSB
	Gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz			3.0	6.0	LSB
Integral non-linearity (INL)	Gain = 1x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz			0.5	2.5	LSB
	Bipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz	INL		0.5	3.0	LSB
	Unipolar – Gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz			1.5	5.0	LSB
Differential non-linearity (DNL)	Gain = 1x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz			0.4	1.0	LSB
	Bipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz	DNL		0.4	1.0	LSB
	Unipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz			0.7	2.0	LSB
Gain error	Bipolar – gain = 1x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–5.0	+2.3	+5.0	LSB
	Unipolar – gain = 1x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–5.0	–2.8	+5.0	LSB
	Bipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–7.0	+2.2	+7.0	LSB
	Unipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–7.0	–1.8	+7.0	LSB
Offset error	Gain = 1x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–5.0	+2.0	+5.0	LSB
	Bipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–5.0	+2.0	+5.0	LSB
	Unipolar – gain = 20x V_{REF} = 4V, V_{CC} = 5V ADC clock = 50 to 200kHz		–6.5	+2.0	+6.5	LSB
Clock frequency			50		200	kHz
Conversion time			65		260	µs

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

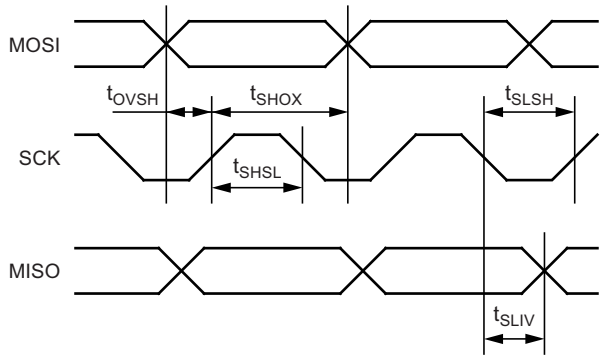


Figure 22-4. Serial Programming Waveforms

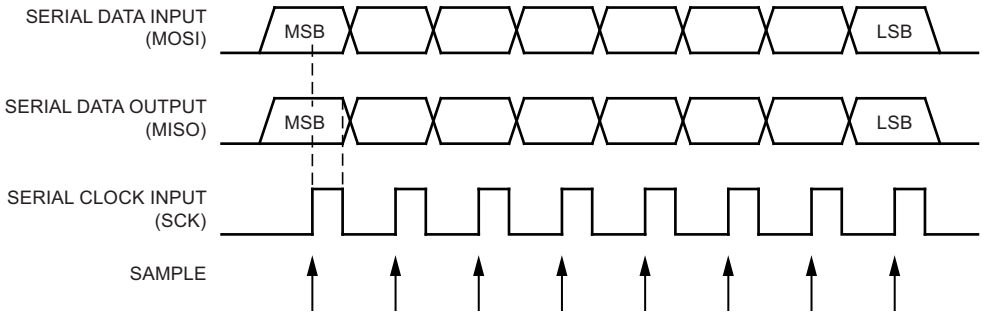


Table 22-8. Serial Programming Characteristics, T_A = –40°C to +125°C, V_{CC} = 2.7 to 5.5V (Unless Otherwise Noted)

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

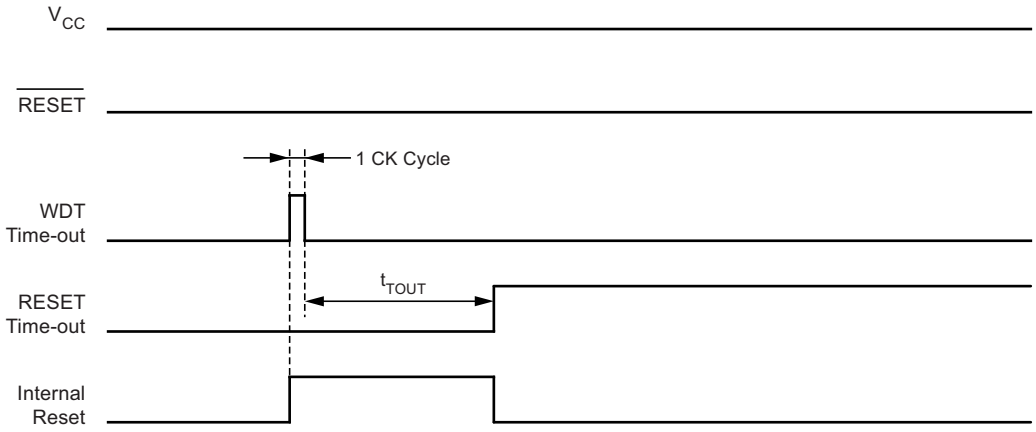


Table 22-9. High-voltage Serial Programming Characteristics
T_A = 25°C ±10%, V_{CC} = 5.0V ±10% (Unless otherwise noted)

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

23. Typical Characteristics – Preliminary Data

The data contained in this section is largely based on simulations and characterization of similar devices in the same process and design methods. Thus, the data should be treated as indications of how the part will behave.

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

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

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

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

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

The difference between current consumption in power-down mode with watchdog timer enabled and power-down mode with watchdog timer disabled represents the differential current drawn by the watchdog timer.

23.1 Active Supply Current

Figure 23-1. Active Supply Current versus Low Frequency (0.1 to 1.0MHz) - Temperature = 25°C

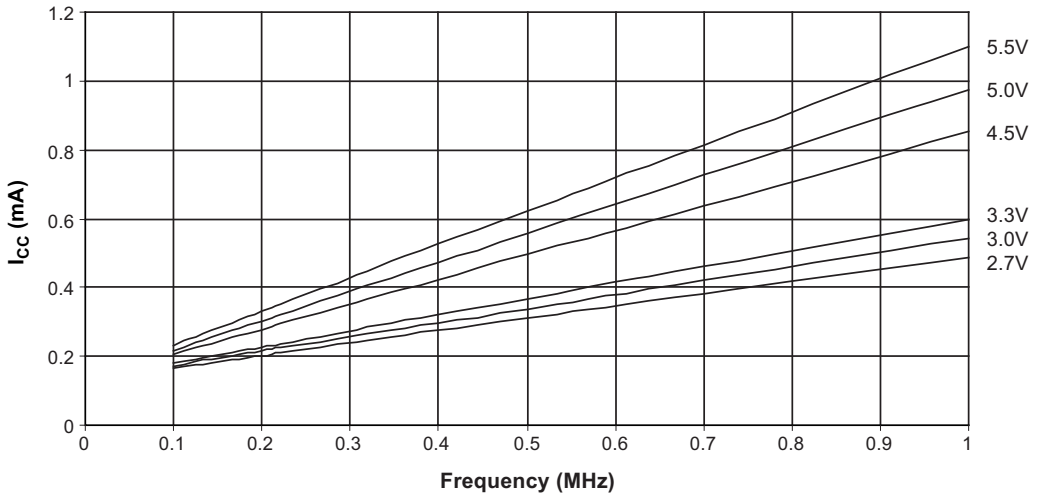


Figure 23-2. Active Supply Current versus Low Frequency (0.1 to 1.0MHz) - Temperature = 125°C

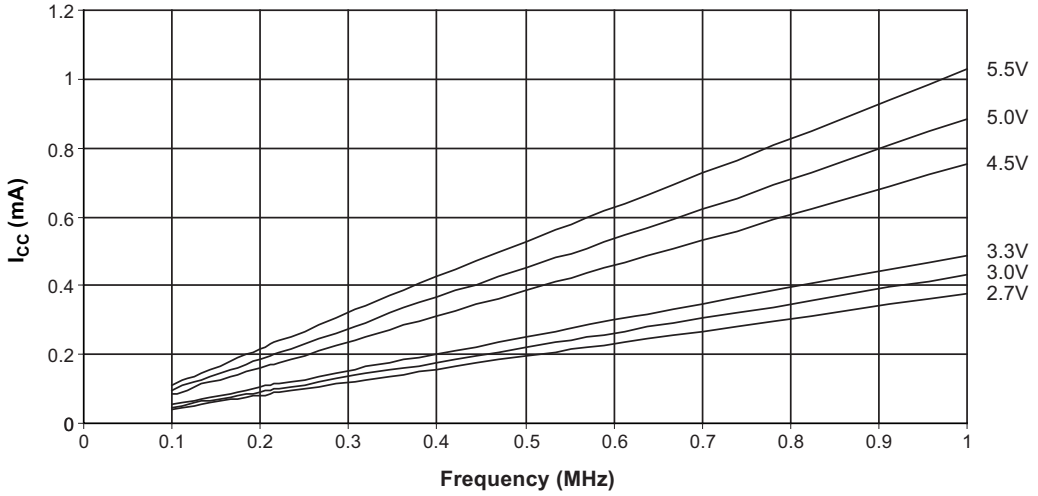


Figure 23-3. Active Supply Current versus Frequency (1 to 20MHz) - Temperature = 25°C

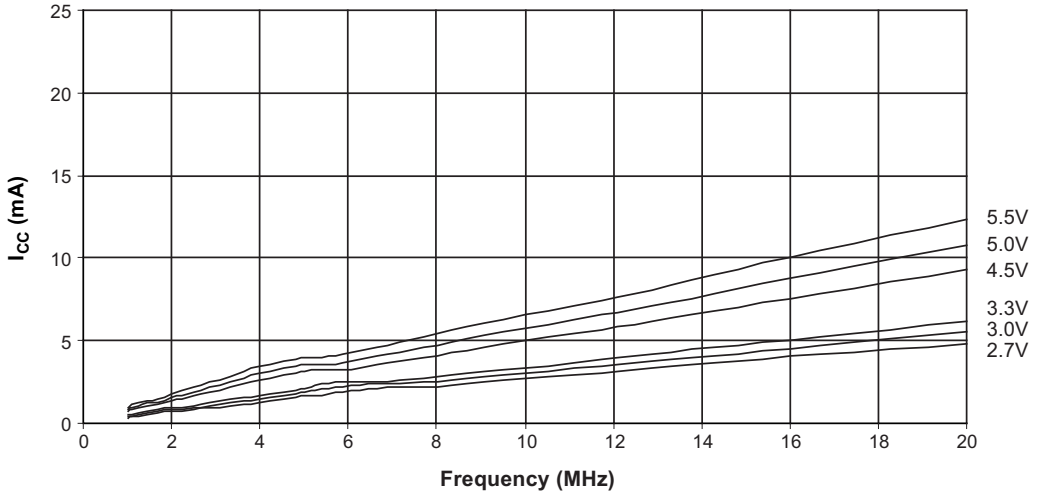


Figure 23-4. Active Supply Current versus Frequency (1 to 20MHz) - Temperature = 125°C

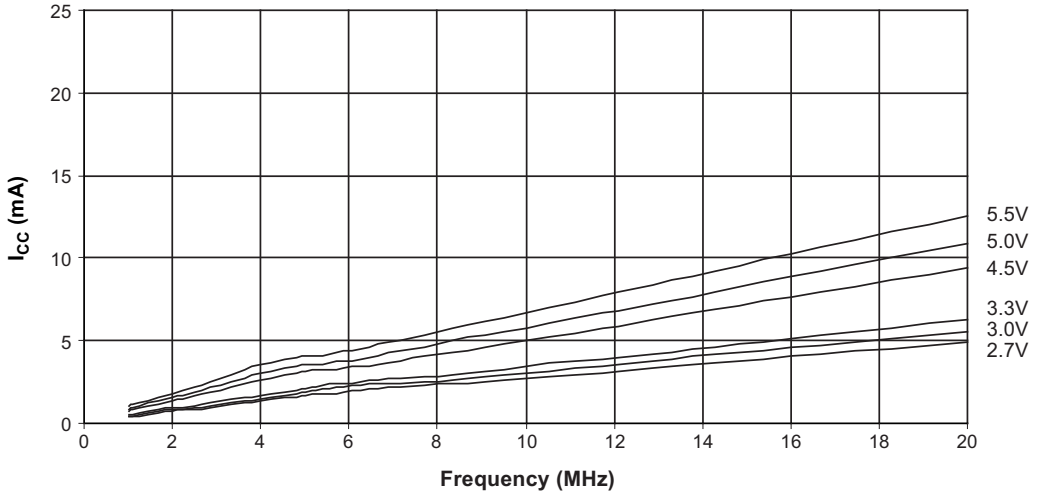


Figure 23-5. Active Supply Current versus V_{CC} (Internal RC Oscillator, 8MHz)

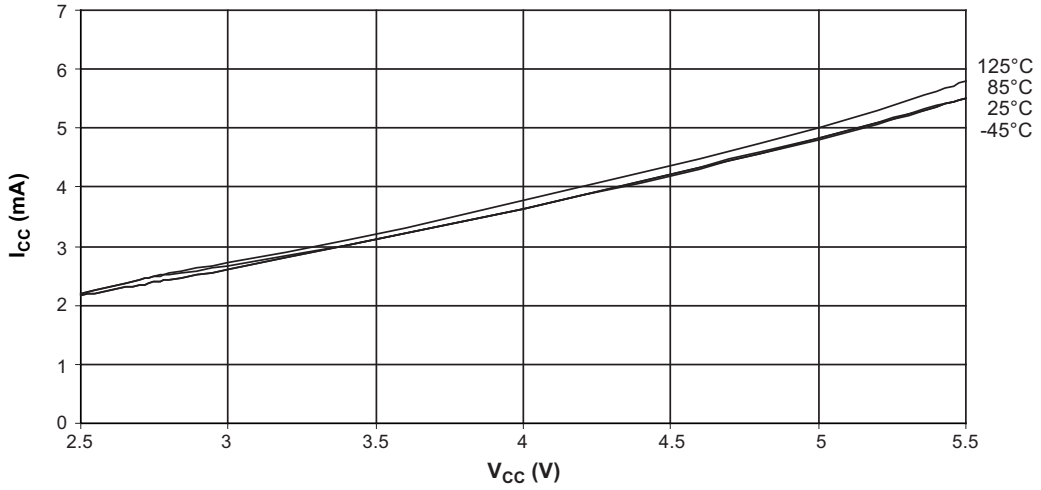


Figure 23-6. Active Supply Current versus V_{CC} (Internal RC Oscillator, 1MHz)

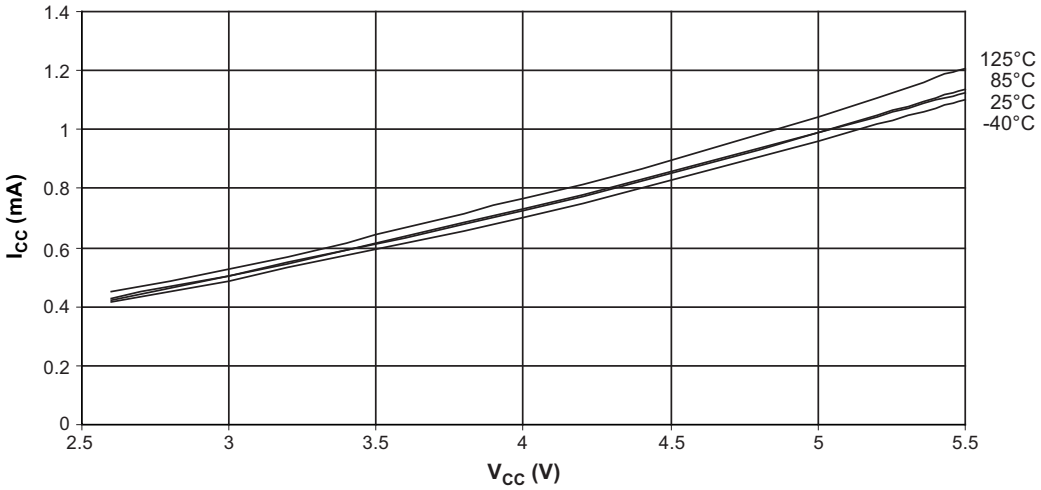
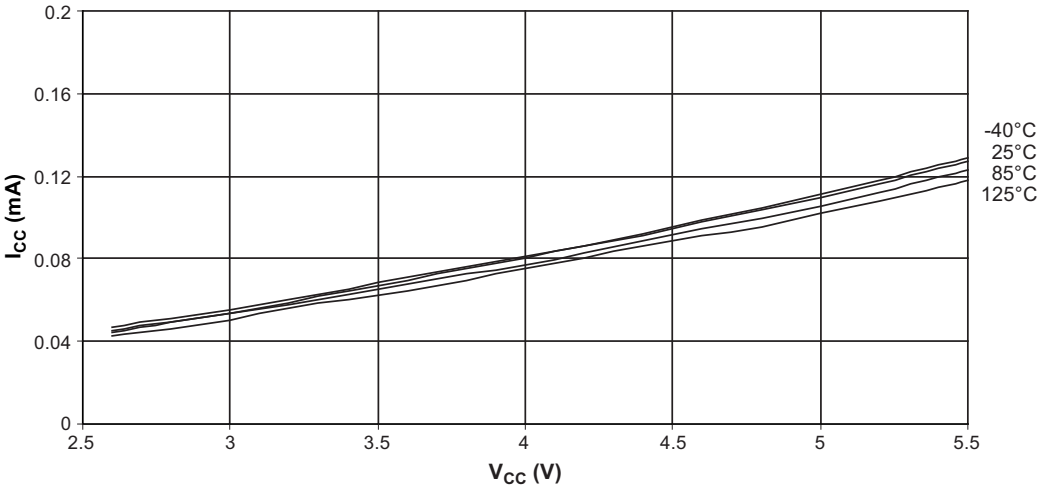


Figure 23-7. Active Supply Current versus V_{CC} (Internal RC Oscillator, 128kHz)



23.2 Idle Supply Current

Figure 23-8. Idle Supply Current versus Low Frequency (0.1 to 1.0MHz)

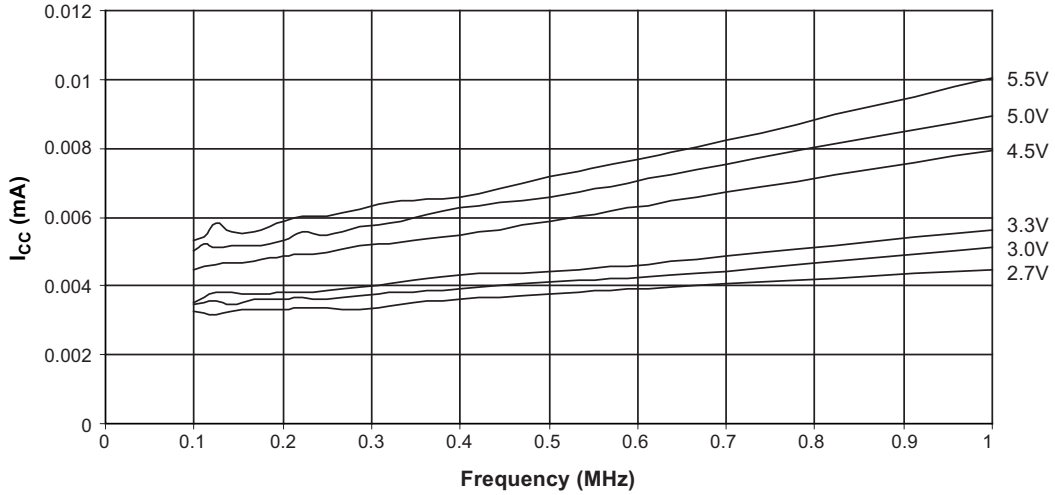


Figure 23-9. Idle Supply Current versus Frequency (1 to 20MHz)

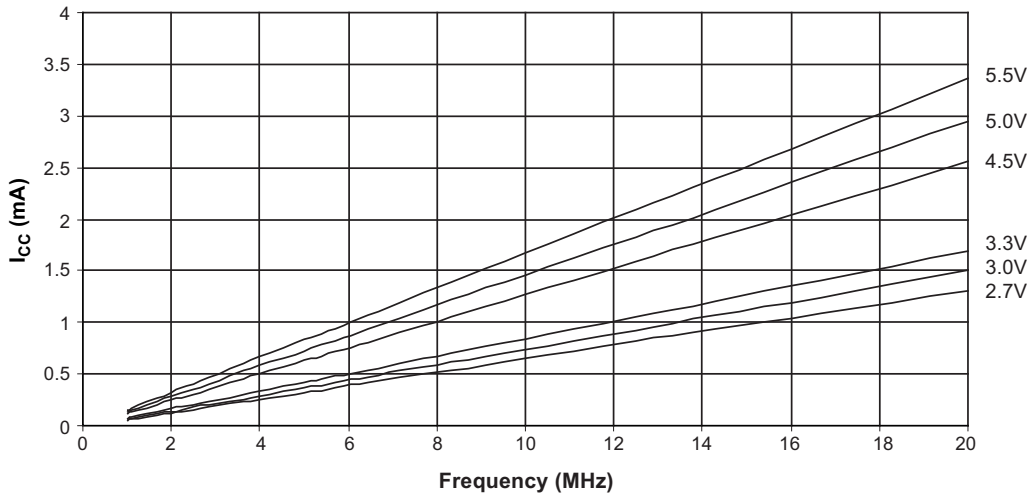


Figure 23-10. Idle Supply Current versus V_{CC} (Internal RC Oscillator, 8MHz)

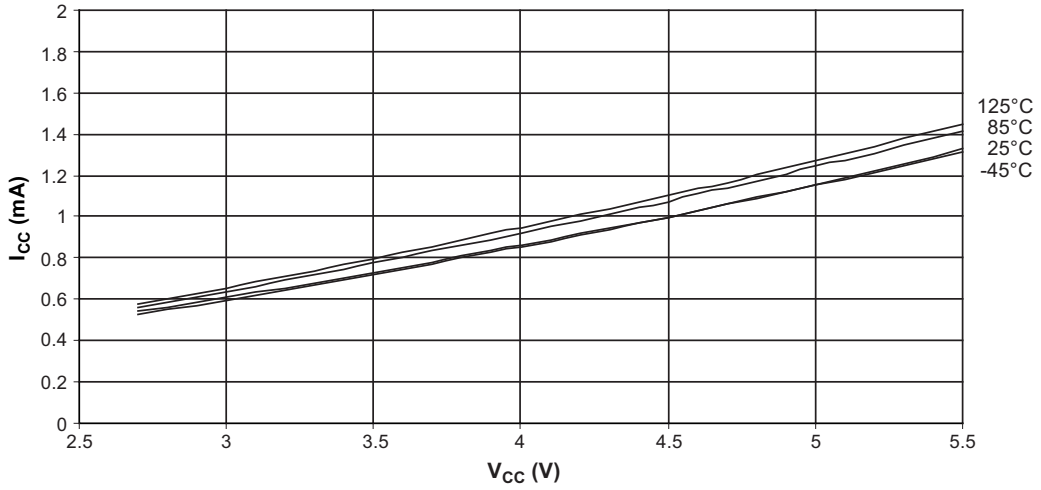


Figure 23-11. Idle Supply Current versus V_{CC} (Internal RC Oscillator, 1MHz)

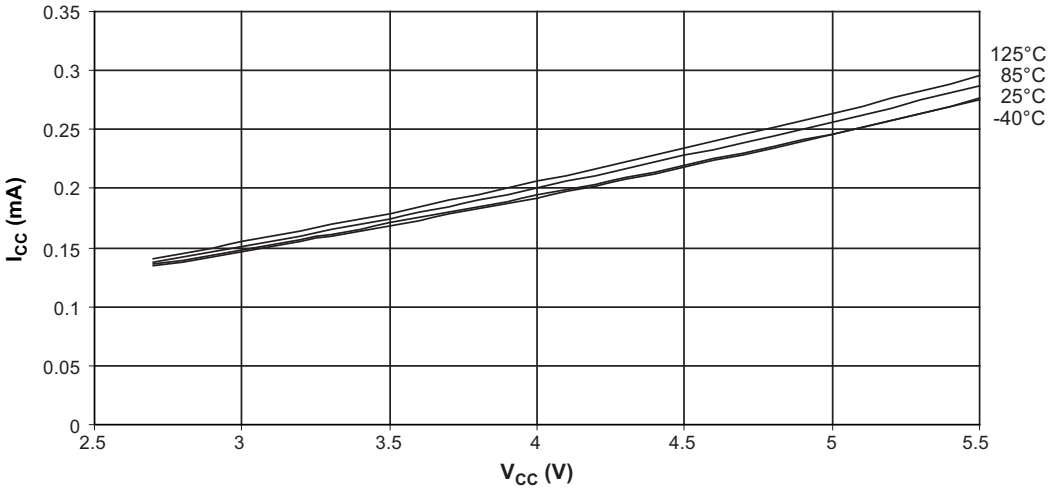
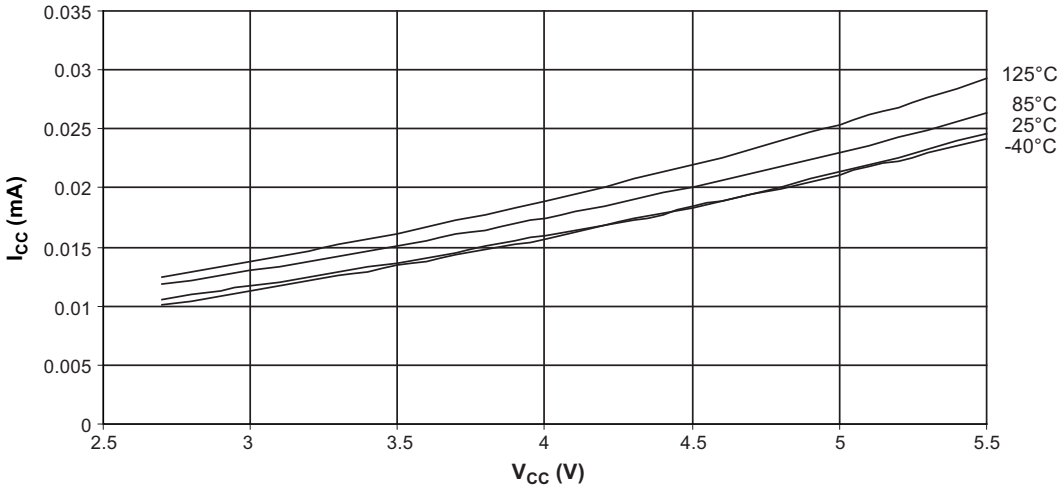


Figure 23-12. Idle Supply Current versus V_{CC} (Internal RC Oscillator, 128kHz)



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)

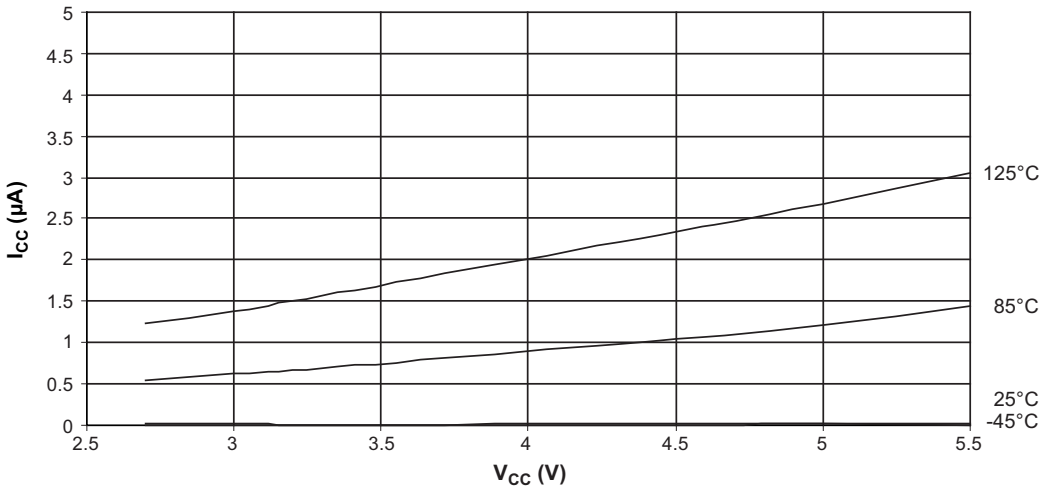
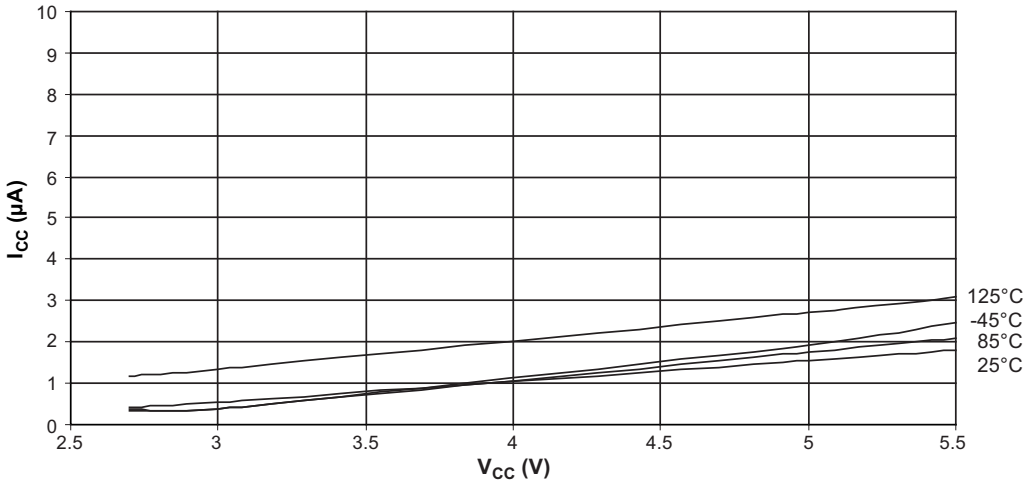


Figure 23-14. Power-down Supply Current versus V_{CC} (Watchdog Timer Enabled)



23.5 Pin Pull-up

Figure 23-15. I/O Pin Pull-up Resistor Current versus input Voltage (V_{CC} = 2.7V)

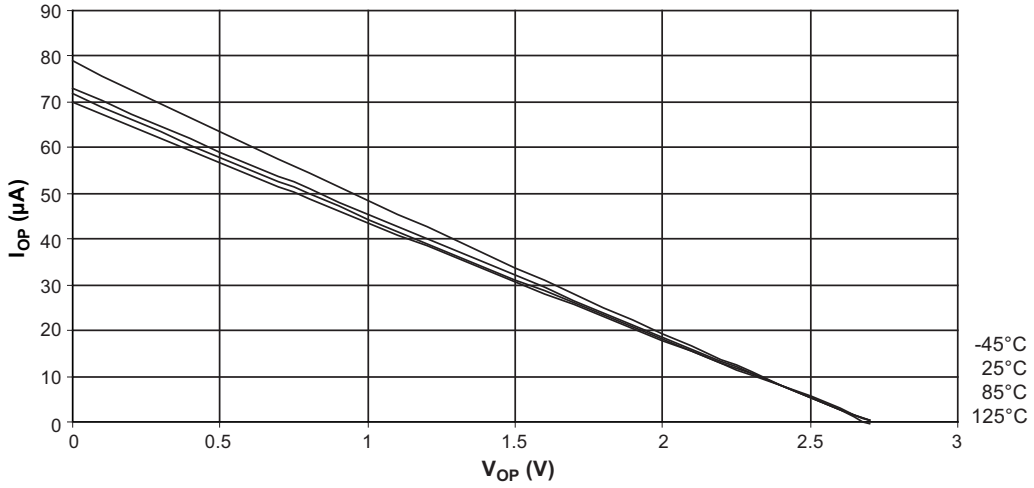


Figure 23-16. I/O pin Pull-up Resistor Current versus Input Voltage (V_{CC} = 5V)

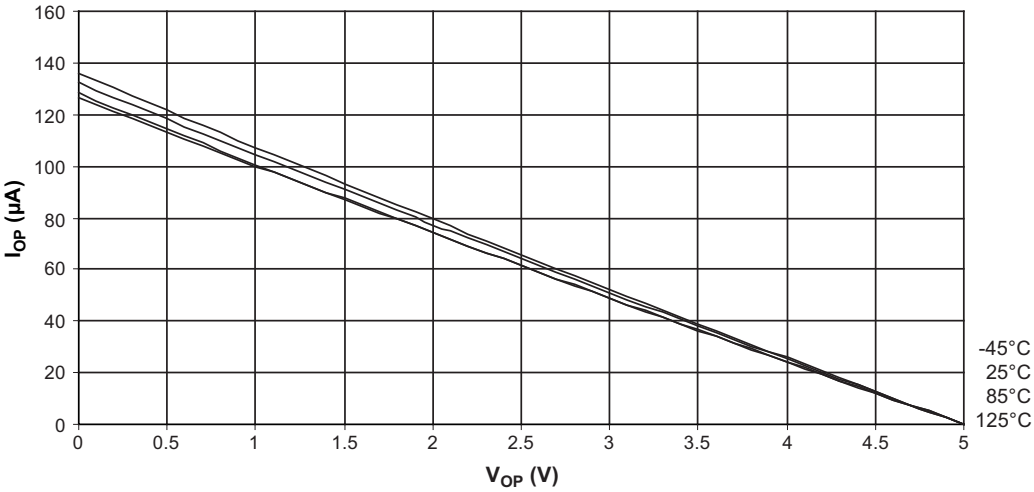


Figure 23-17. Reset Pull-up Resistor Current versus Reset Pin Voltage (V_{CC} = 2.7V)

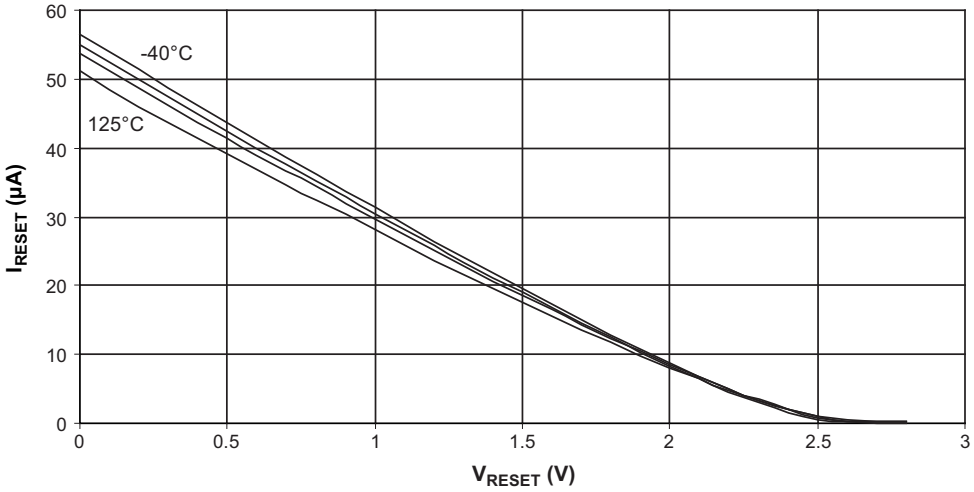
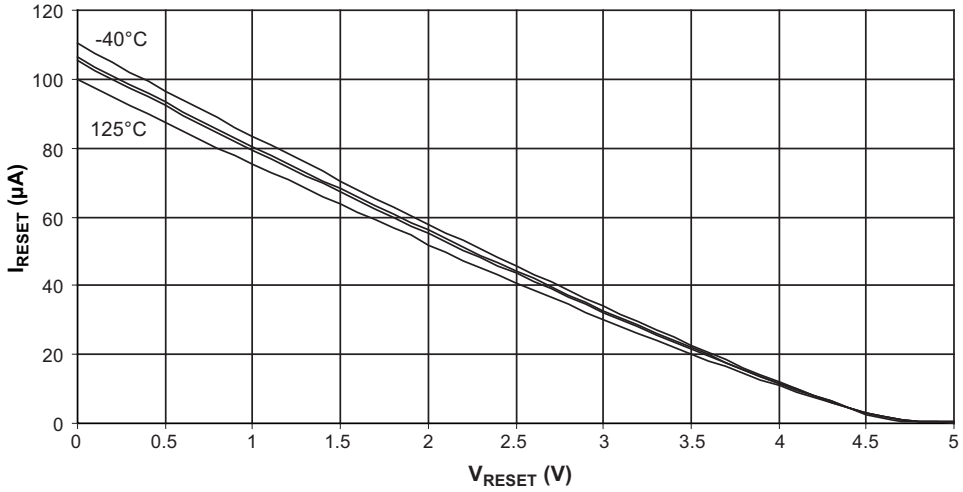


Figure 23-18. Reset Pull-up Resistor Current versus Reset Pin Voltage (V_{CC} = 5V)



23.6 Pin Driver Strength

Figure 23-19. I/O Pin Output Voltage versus Sink Current (V_{CC} = 3V)

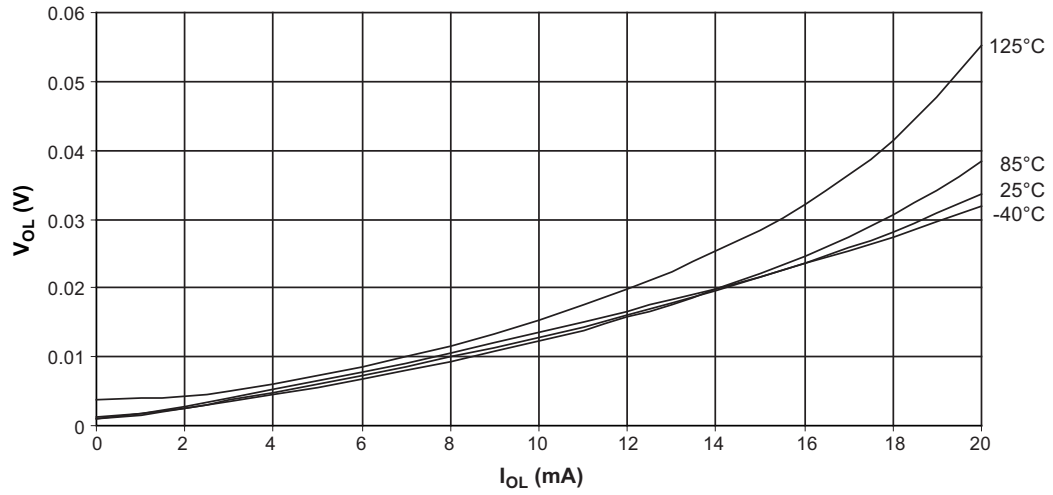


Figure 23-20. I/O pin Output Voltage versus Sink Current (V_{CC} = 5V)

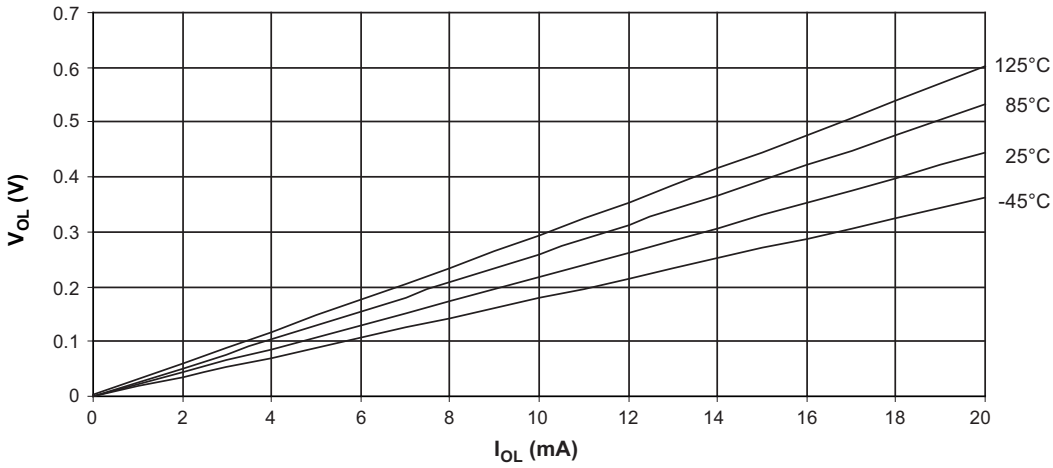


Figure 23-21. I/O Pin Output Voltage versus Source Current (V_{CC} = 3V)

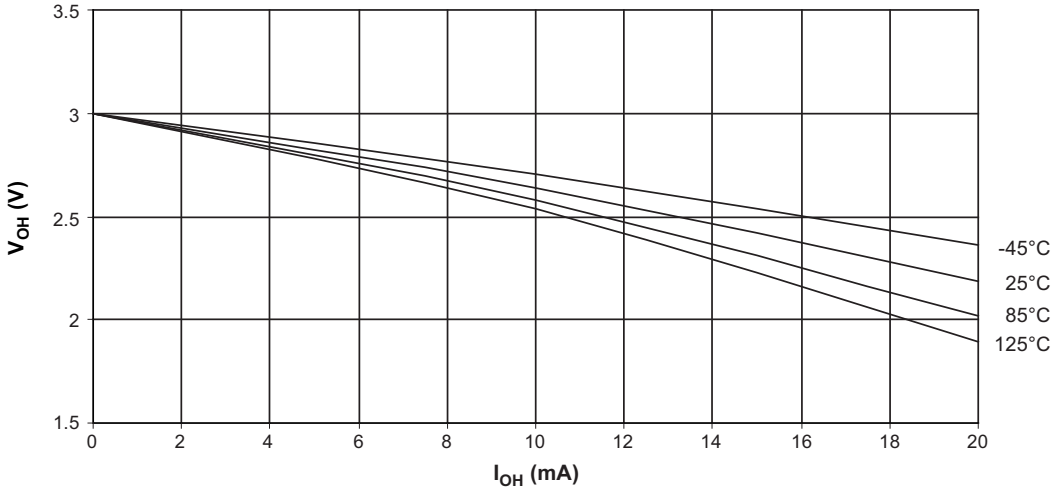
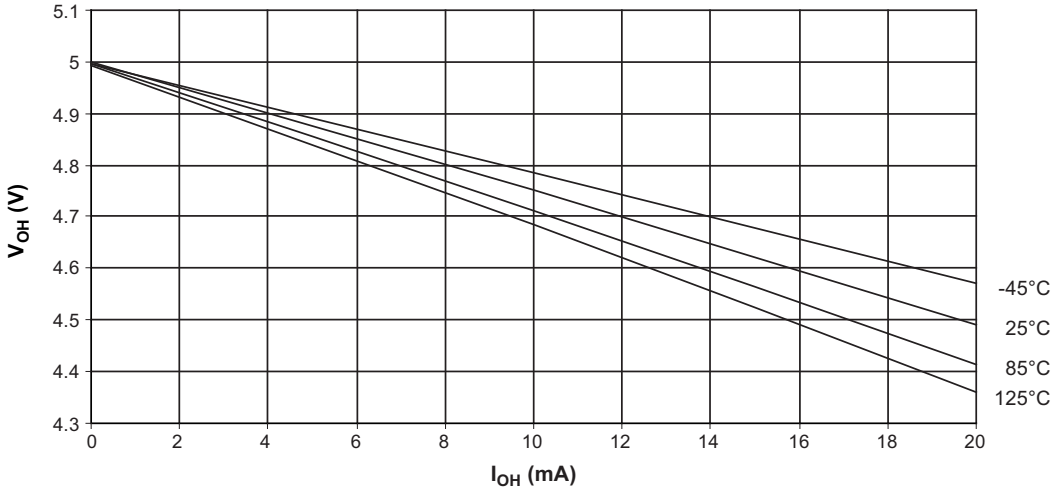


Figure 23-22. I/O Pin output Voltage versus Source Current (V_{CC} = 5V)



23.7 Pin Threshold and Hysteresis

Figure 23-23. I/O Pin Input Threshold Voltage versus V_{CC} (V_{IH}, IO Pin Read as '1')

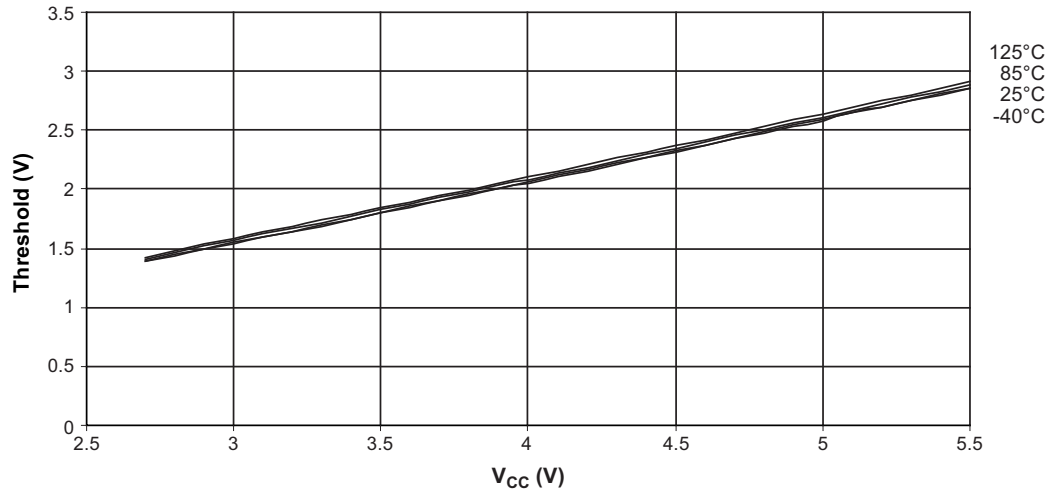


Figure 23-24. I/O Pin Input threshold Voltage versus V_{CC} (V_{IL}, IO Pin Read as '0')

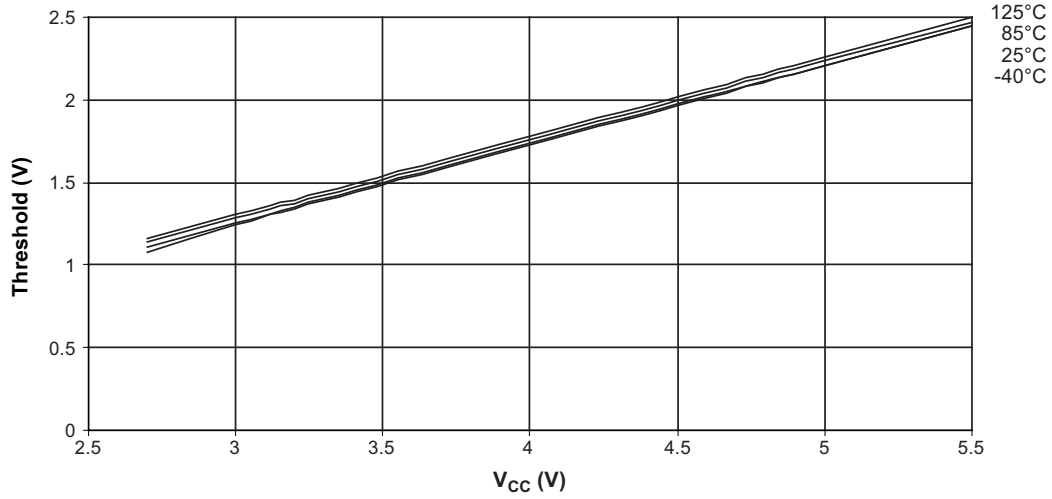


Figure 23-25. I/O Pin Input Hysteresis versus V_{CC}

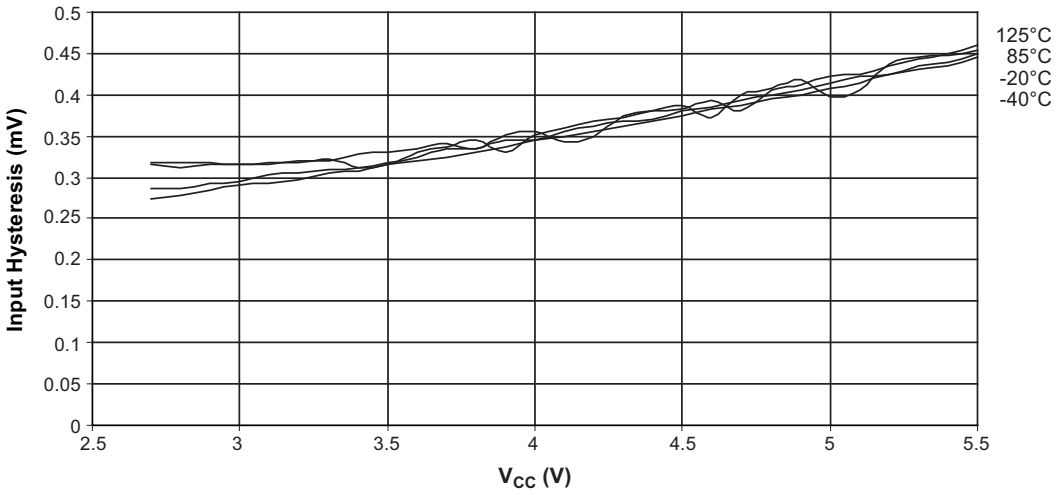


Figure 23-26. Reset Input Threshold Voltage versus V_{CC} (V_{IH}, IO Pin Threshold as '1')

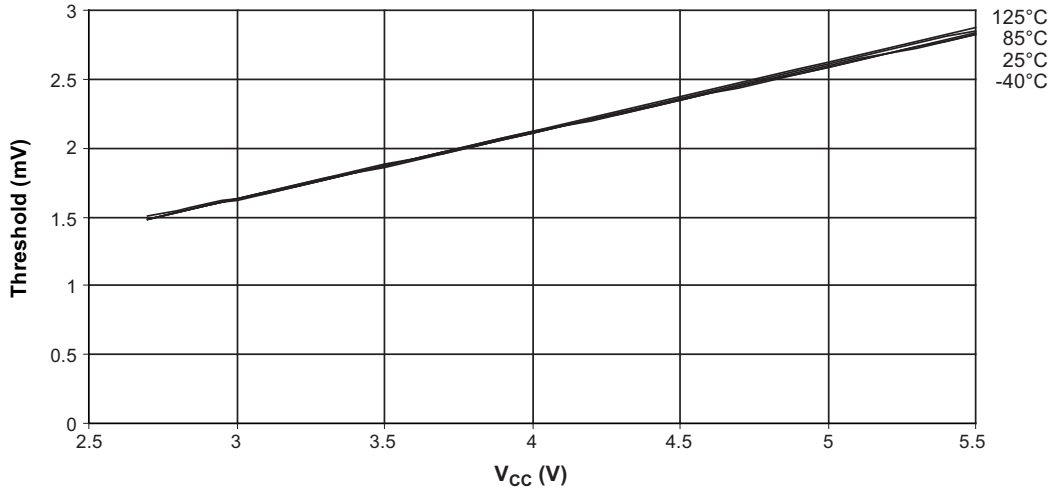


Figure 23-27. Reset Input Threshold Voltage versus V_{CC} (V_{IL}, IO pin Read as '0')

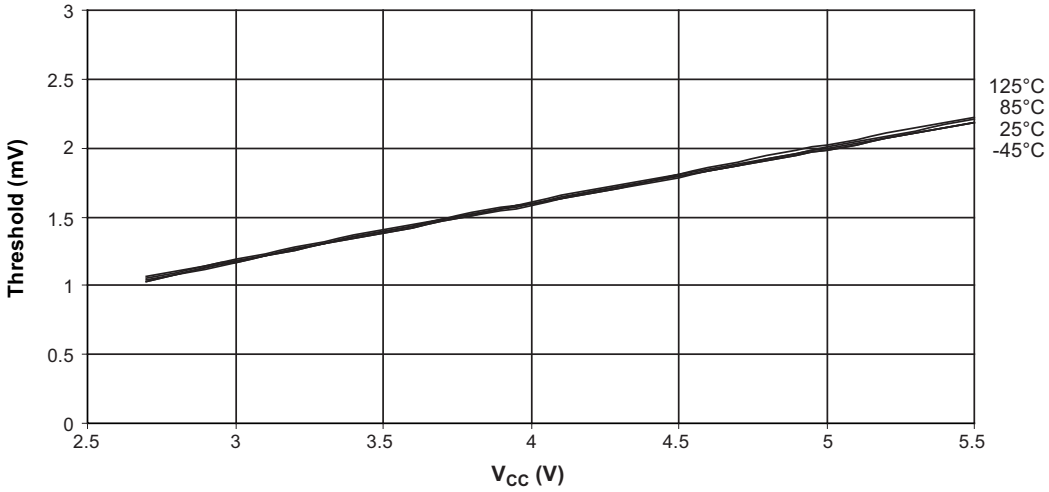
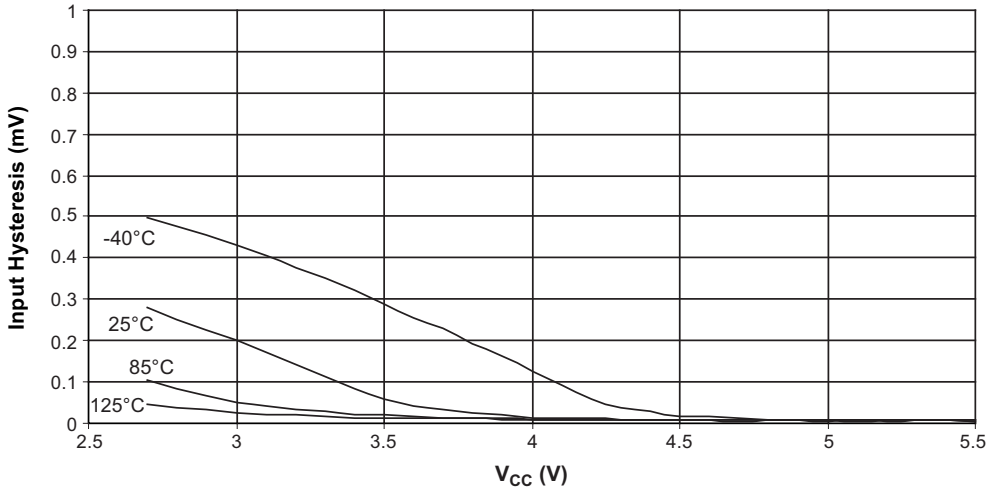


Figure 23-28. Reset Pin Input Hysteresis versus V_{CC}



23.8 BOD Threshold and Analog Comparator Offset

Figure 23-29. BOD Threshold versus Temperature (BODLEVEL is 4.3V)

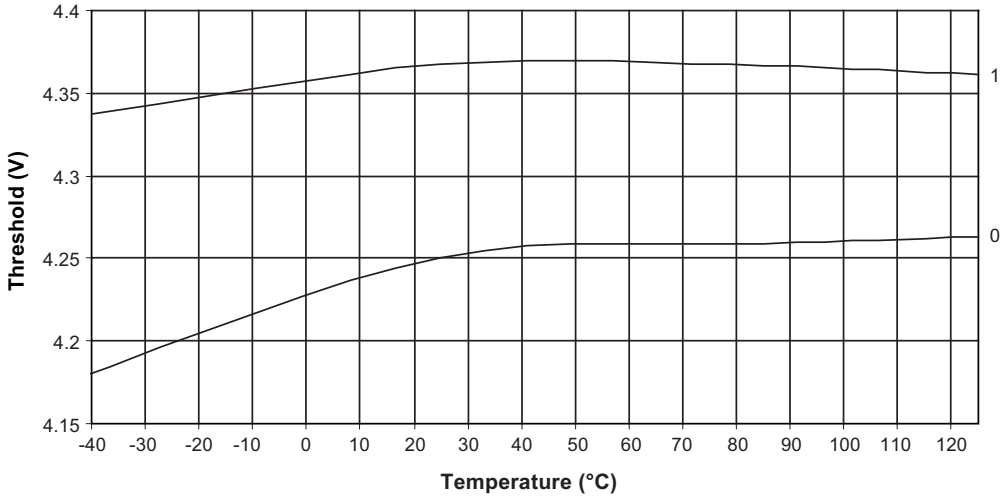


Figure 23-30. BOD Threshold versus Temperature (BODLEVEL is 2.7V)

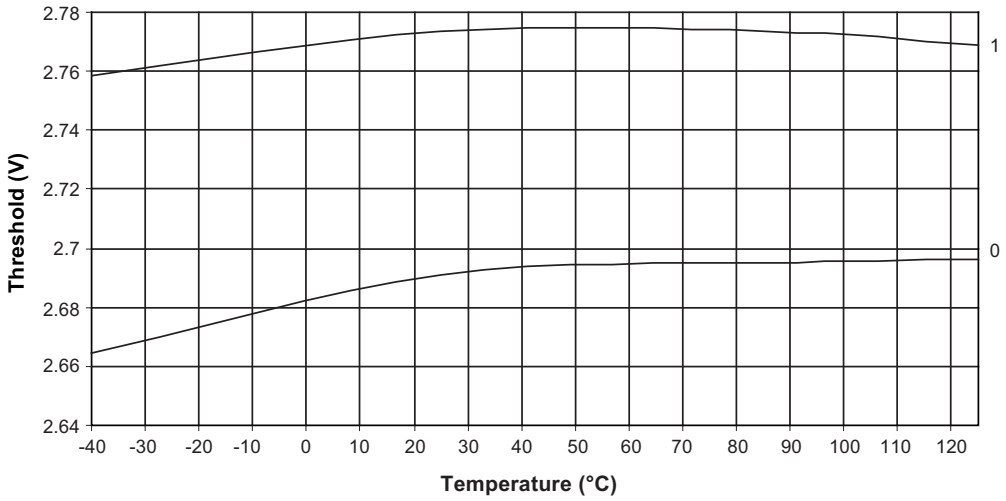
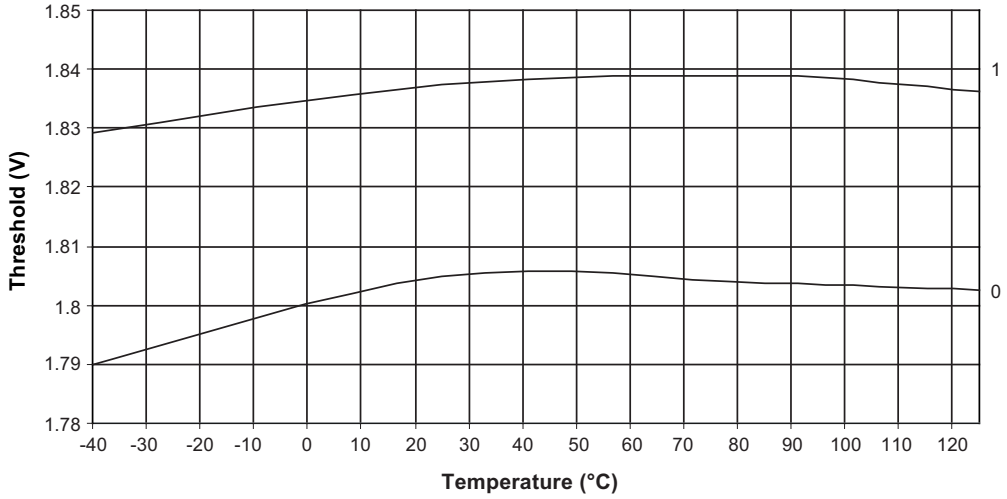


Figure 23-31. BOD Threshold versus Temperature (BODLEVEL is 1.8V)



23.9 Internal Oscillator Speed

Figure 23-32. Watchdog Oscillator Frequency versus V_{CC}

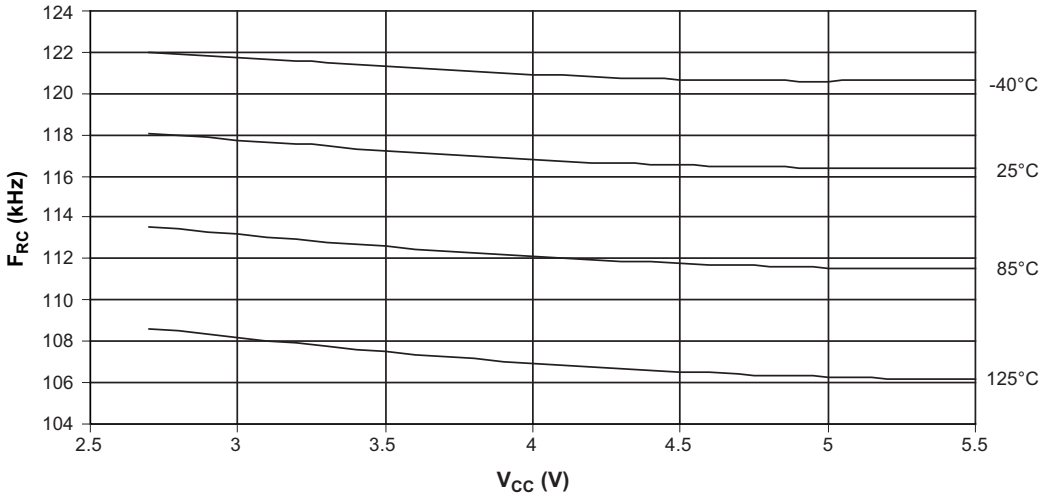


Figure 23-33. Calibrated 8MHz RC Oscillator Frequency versus V_{CC}

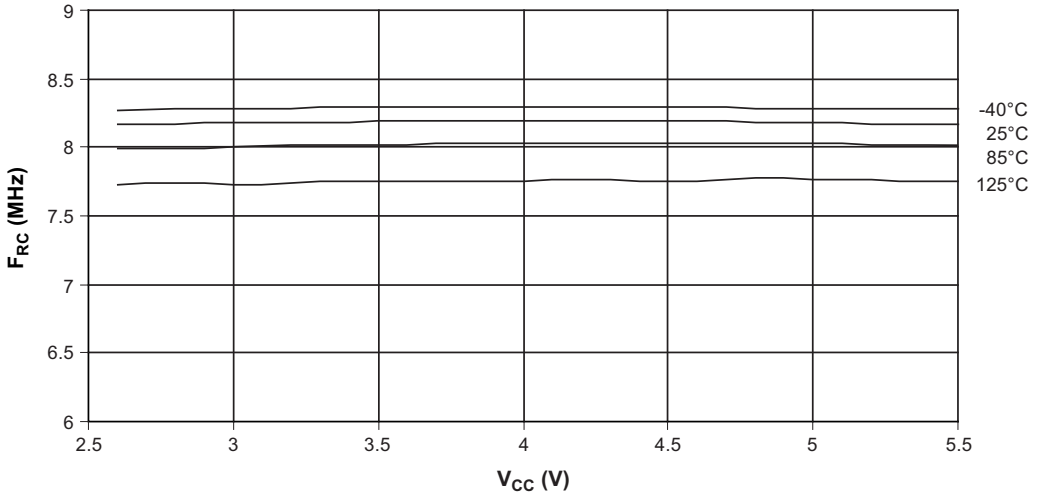


Figure 23-34. Calibrated 8MHz RC Oscillator Frequency versus Temperature

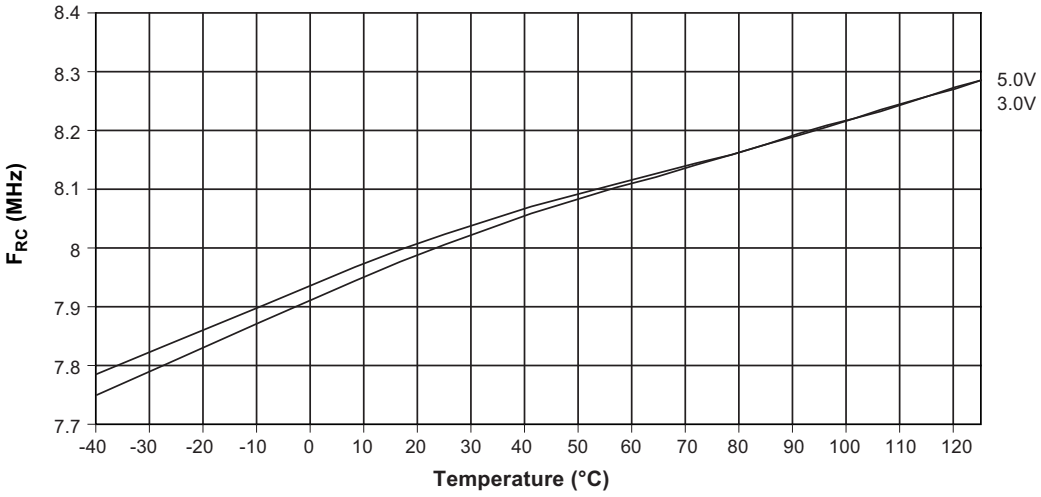
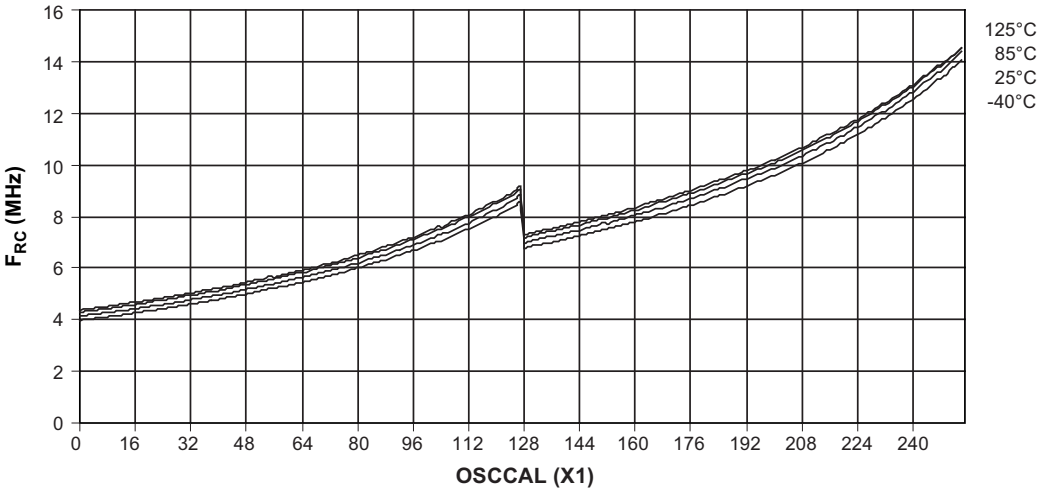


Figure 23-35. Calibrated 8MHz RC Oscillator Frequency versus OSCCAL Value



23.10 Current Consumption of Peripheral Units

Figure 23-36. ADC Current versus V_{CC}

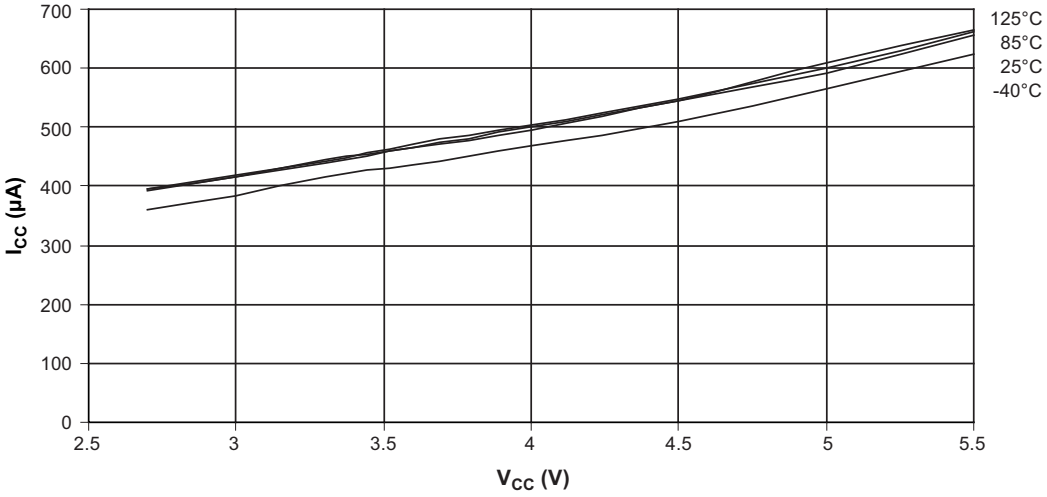


Figure 23-37. AREF External Reference Current versus V_{CC}

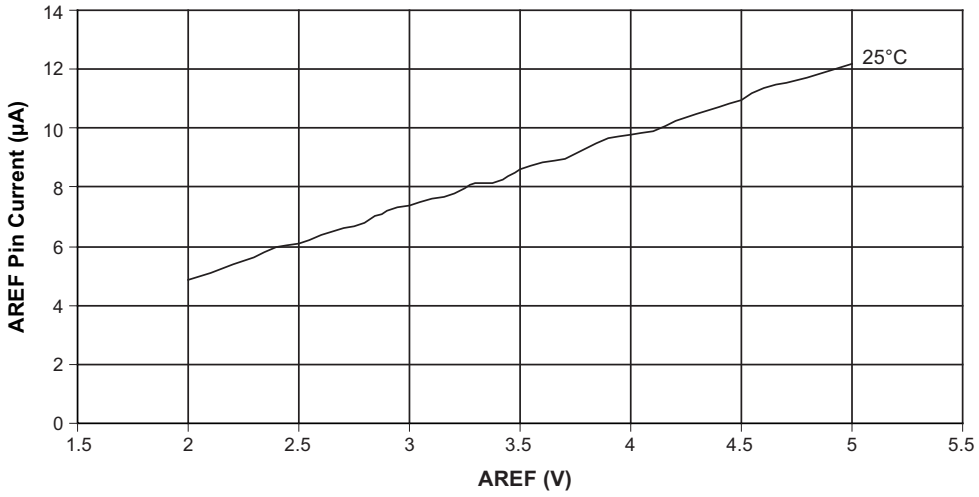


Figure 23-38. Analog Comparator Current versus V_{CC}

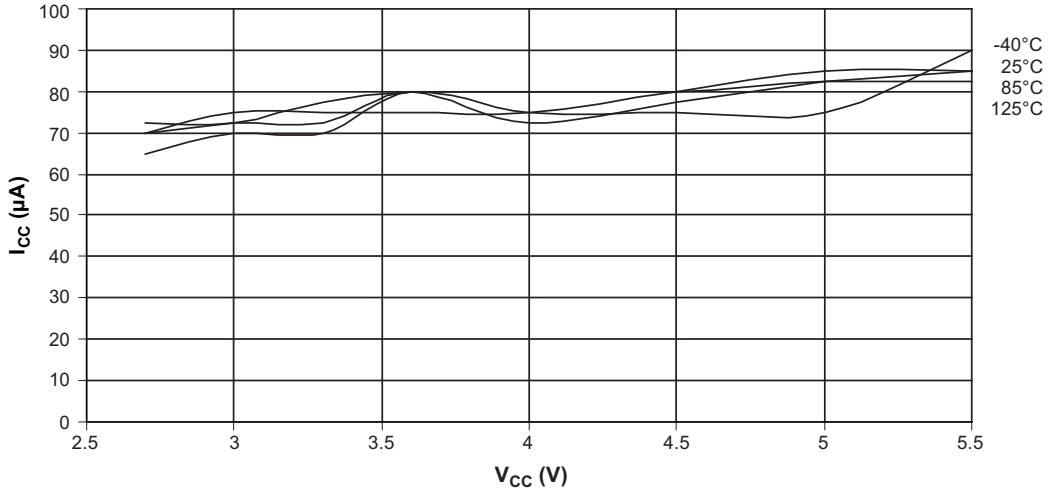


Figure 23-39. Programming Current versus V_{CC}

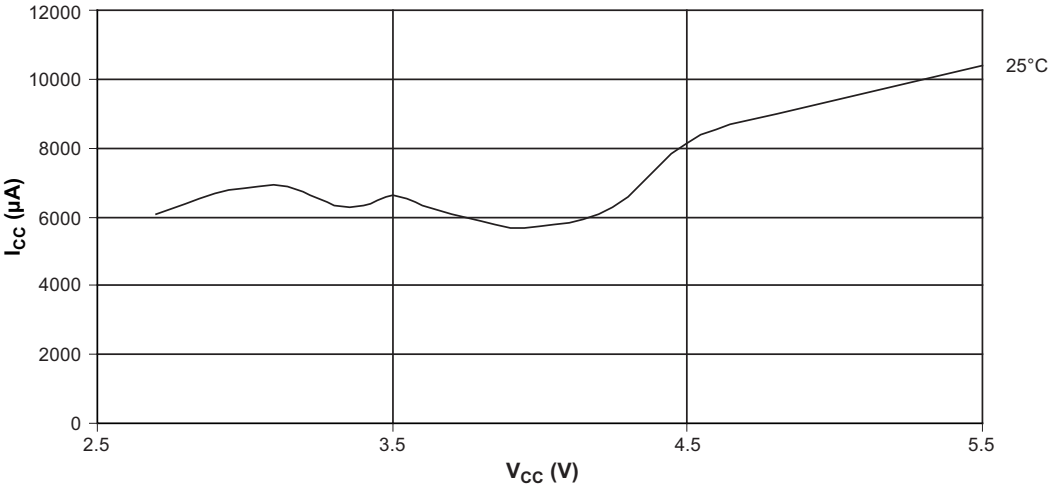
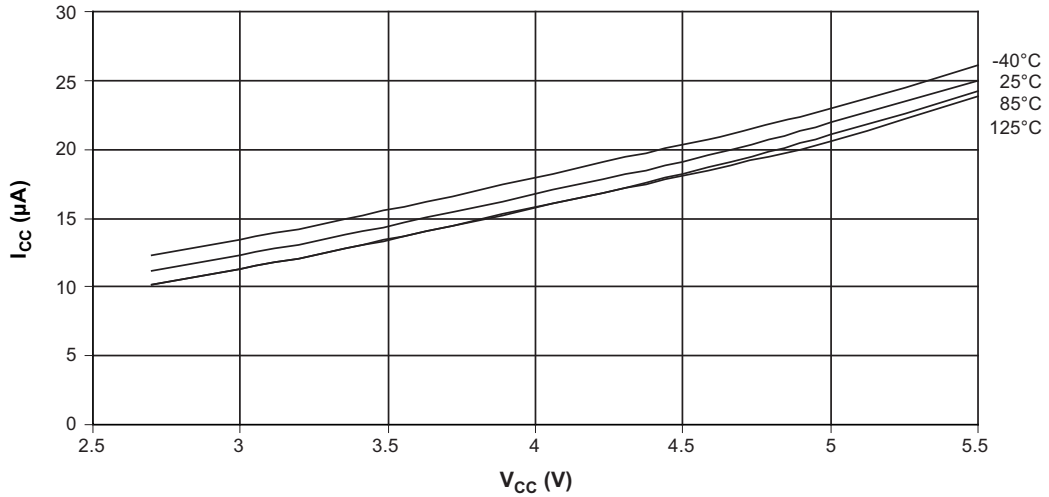


Figure 23-40. Watchdog Timer Current versus V_{CC}



23.11 Current Consumption in Reset and Reset Pulse Width

Figure 23-41. Reset Supply Current versus V_{CC} (0.1 to 1.0MHz, excluding Current Through the Reset Pull-up)

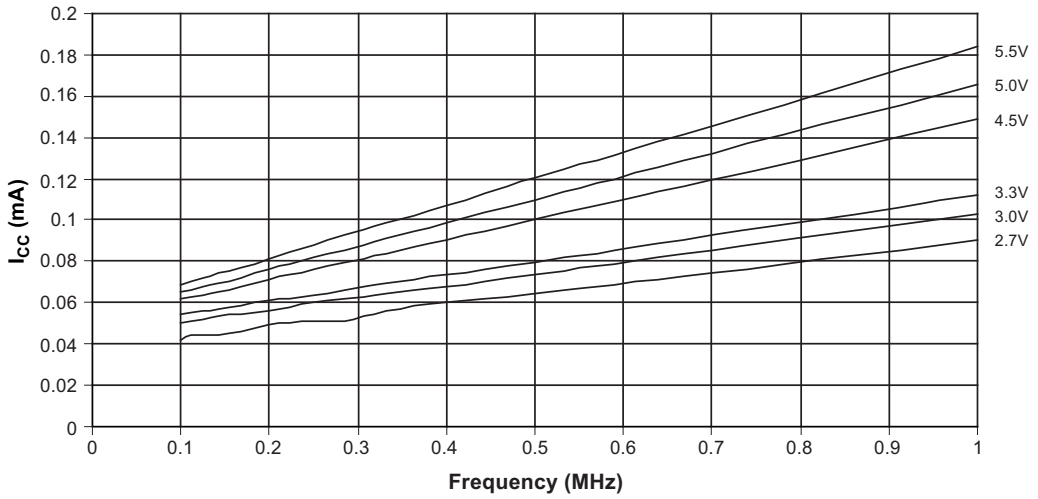


Figure 23-42. Reset Supply Current versus V_{CC} (1 to 20MHz, Excluding Current Through the Reset Pull-up)

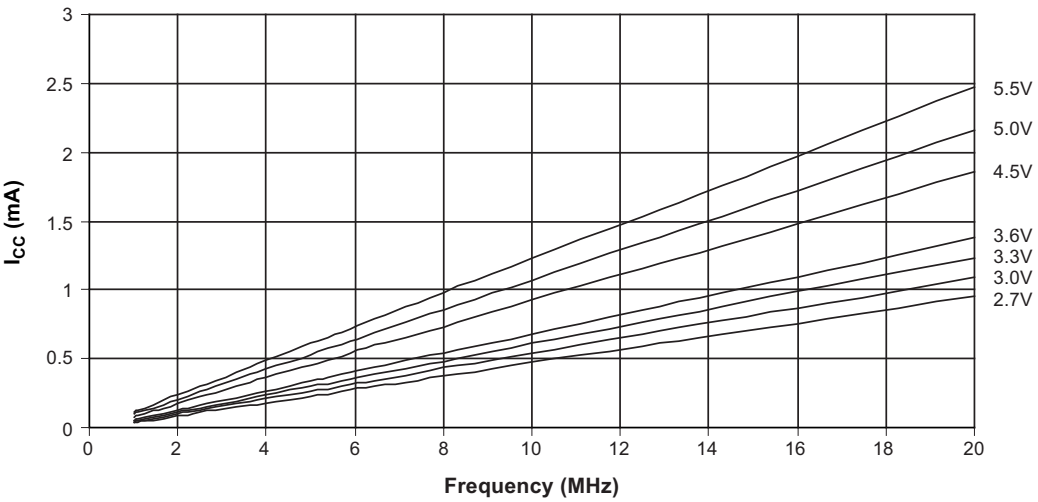
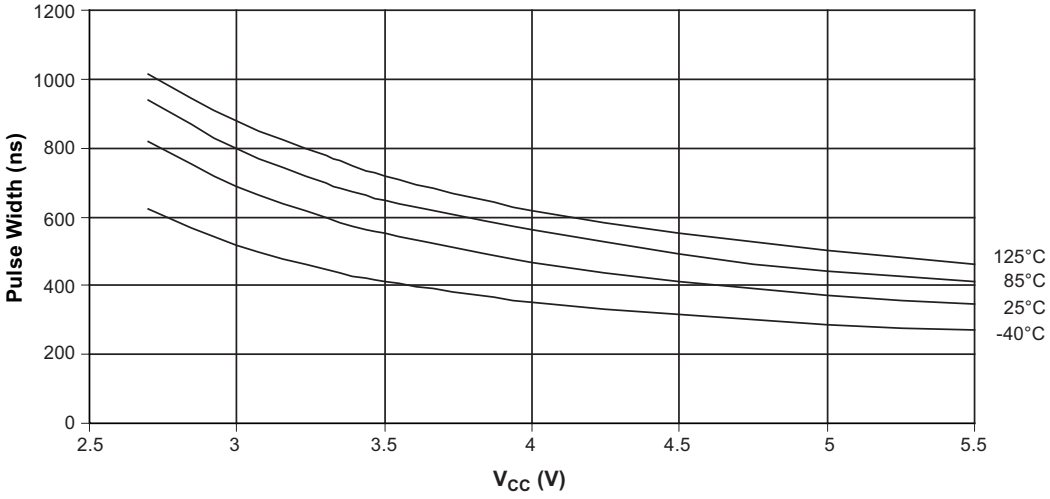


Figure 23-43. Minimum Reset Pulse Width versus V_{CC}



24. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	10
0x3E (0x5E)	SPH	–	–	–	–	–	–	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
0x3C (0x5C)	OCR0B	Timer/Counter0 – Output compare register B								76
0x3B (0x5B)	GIMSK	–	INT0	PCIE1	PCIE0	–	–	–	–	47
0x3A (0x5A	GIFR	–	INTF0	PCIF1	PCIF0	–	–	–	–	48
0x39 (0x59)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	77
0x38 (0x58)	TIFR0		–	–	–	–	OCF0B	OCF0A	TOV0	77
0x37 (0x57)	SPMCSR	–	–	–	CTPB	RFLB	PGWRT	PGERS	SPMEN	140
0x36 (0x56)	OCR0A	Timer/Counter0 – Output compare register A								76
0x35 (0x55)	MCUCR	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	47
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	41
0x33 (0x53)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	75
0x32 (0x52)	TCNT0	Timer/Counter0								76
0x31 (0x51)	OSCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	29
0x30 (0x50)	TCCR0A	COM01	COM0A0	COM0B1	COM0B0	–		WGM01	WGM00	73
0x2F (0x4F)	TCCR1A	COM11	COM1A0	COM1B1	COM1B0	–		WGM11	WGM10	97
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	99
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter register high byte								100
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter register low byte								100
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Compare register A high byte								100
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Compare register A low byte								100
0x29 (0x49)	OCR1BH	Timer/Counter1 – Compare register B high byte								101
0x28 (0x48)	OCR1BL	Timer/Counter1 – Compare register B low byte								101
0x27 (0x47)	DWDR	DWDR[7:0]								136
0x26 (0x46)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	29
0x25 (0x45)	ICR1H	Timer/Counter1 - Input capture register high byte								101
0x24 (0x44)	ICR1L	Timer/Counter1 - Input capture register low byte								101
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	–	PSR10	104
0x22 (0x42)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	100
0x21 (0x41)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	41
0x20 (0x40)	PCMSK1	–	–	–	–	PCINT11	PCINT10	PCINT9	PCINT8	48
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	21
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	21
0x1D (0x3D)	EEDR	EEPROM data register								21
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	22
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	61
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	61
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	62

Notes:
1. For compatibility with future devices, reserved bits should be written to logical zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operation the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

24. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x18 (0x38)	PORTB	–	–	–	–	PORTB3	PORTB2	PORTB1	PORTB0	62
0x17 (0x37)	DDRB	–	–	–	–	DDB3	DDB2	DDB1	DDB0	62
0x16 (0x36)	PINB	–	–	–	–	PINB3	PINB2	PINB1	PINB0	62
0x15 (0x35)	GPIOR2	General purpose I/O register 2								23
0x14 (0x34)	GPIOR1	General purpose I/O register 1								23
0x13 (0x33)	GPIOR0	General purpose I/O register 0								23
0x12 (0x32)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	48
0x11 (0x31))	Reserved	–								
0x10 (0x30)	USIBR	USI buffer register								111
0x0F (0x2F)	USIDR	USI data register								111
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICN3	USICN2	USICN1	USICN0	112
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	112
0x0C (0x2C)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	101
0x0B (0x2B)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	102
0x0A (0x2A)	Reserved	–								
0x09 (0x29)	Reserved	–								
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	116
0x07 (0x27)	ADMUX	REFS1	REFS0	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	129
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	132
0x05 (0x25)	ADCH	ADC data register high byte								133
0x04 (0x24)	ADCL	ADC data register low byte								133
0x03 (0x23)	ADCSRB	BIN	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	133
0x02 (0x22)	Reserved	–								
0x01 (0x21)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	117,134
0x00 (0x20)	PRR	–	–	–	–	PRTIM1	PRTIM0	PRUSI	PRADC	32

Notes:
1. For compatibility with future devices, reserved bits should be written to logical zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operation the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

25. 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
Branch Instructions					
RJMP	k	Relative jump	PC ← PC + k + 1	None	2
IJMP		Indirect jump to (Z)	PC ← Z	None	2
RCALL	k	Relative subroutine call	PC ← PC + k + 1	None	3
ICALL		Indirect call to (Z)	PC ← Z	None	3
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

25. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag is cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
Bit and Bit-Test Instructions					
SBI	P,b	Set bit in I/O register	I/O (P, b) ← 1	None	2
CBI	P,b	Clear bit in I/O register	I/O (P, b) ← 0	None	2
LSL	Rd	Logical shift left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical shift right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate left through carry	Rd(0) ← C,Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate right through carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic shift right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap nibbles	Rd(3..0) ← Rd(7..4),Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit store from register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to register	Rd(b) ← T	None	1
SEC		Set carry	C ← 1	C	1
CLC		Clear carry	C ← 0	C	1
SEN		Set negative flag	N ← 1	N	1
CLN		Clear negative flag	N ← 0	N	1
SEZ		Set zero flag	Z ← 1	Z	1
CLZ		Clear zero flag	Z ← 0	Z	1
SEI		Global interrupt enable	I ← 1	I	1
CLI		Global interrupt disable	I ← 0	I	1

25. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
SES		Set signed test flag	S ← 1	S	1
CLS		Clear signed test flag	S ← 0	S	1
SEV		Set twos complement overflow.	V ← 1	V	1
CLV		Clear twos complement overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half carry flag in SREG	H ← 1	H	1
CLH		Clear half carry flag in SREG	H ← 0	H	1
Data Transfer Instructions					
MOV	Rd, Rr	Move between registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy register word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load immediate	Rd ← K	None	1
LD	Rd, X	Load indirect	Rd ← (X)	None	2
LD	Rd, X+	Load indirect and post-inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load indirect and pre-dec.	X ← X – 1, Rd ← (X)	None	2
LD	Rd, Y	Load indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load indirect and post-inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load indirect and pre-dec.	Y ← Y – 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load indirect with displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load indirect and post-inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load indirect and pre-dec.	Z ← Z – 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load indirect with displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store indirect	(X) ← Rr	None	2
ST	X+, Rr	Store indirect and post-inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store indirect and pre-dec.	X ← X – 1, (X) ← Rr	None	2
ST	Y, Rr	Store indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store indirect and post-inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store indirect and pre-dec.	Y ← Y – 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store indirect with displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store indirect and post-inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store indirect and pre-dec.	Z ← Z – 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store indirect with displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store direct to SRAM	(k) ← Rr	None	2
LPM		Load program memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load program memory and post-inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store program memory	(z) ← R1:R0	None	
IN	Rd, P	In port	Rd ← P	None	1
OUT	P, Rr	Out port	P ← Rr	None	1

25. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
PUSH	Rr	Push register on stack	STACK ← Rr	None	2
POP	Rd	Pop register from stack	Rd ← STACK	None	2
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep	(see specific descr. for sleep function)	None	1
WDR		Watchdog reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For on-chip Debug Only	None	N/A

26. Ordering Information

26.1 ATtiny24/44/84

Ordering Code	Speed (MHz)[3]	Power Supply (V)	Package[1][2]	Operation Range
ATtiny24-15SSZ	16	2.7 to 5.5	TU	Automotive (–40° to +125°C)
ATtiny24-15MZ	16	2.7 to 5.5	PC	Automotive (–40° to +125°C)
ATtiny44-15SSZ	16	2.7 to 5.5	TU	Automotive (–40° to +125°C)
ATtiny44-15MZ	16	2.7 to 5.5	PC	Automotive (–40° to +125°C)
ATtiny84-15MZ	16	2.7 to 5.5	PC	Automotive (–40° to +125°C)

Notes:
1. Green and ROHS packaging
2. Tape and reel with Dry-pack delivery.
3. For Speed versus V_{CC},see Figure 22-1 on page 157.

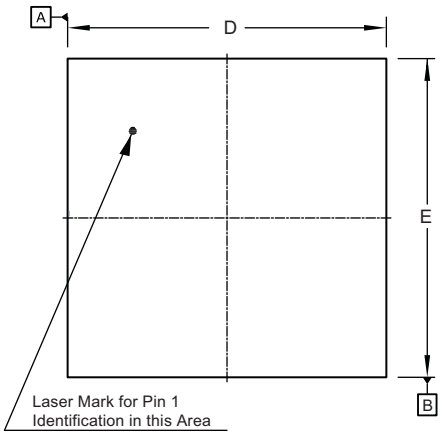
Table 26-1. Package Type

Package Type	
TU	14-lead, 0.150" wide body, plastic gull wing small outline package (SOIC)
PC	20-pad, 4 × 4 × 0.8mm body, quad flat no-lead/micro lead frame package (QFN/MLF)

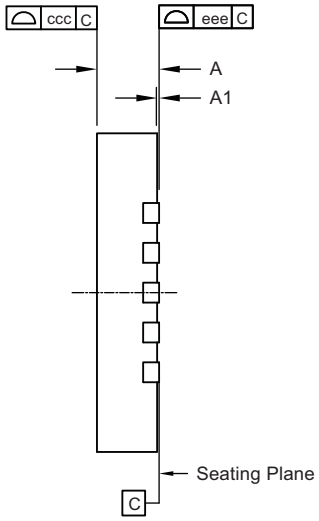
27. Packaging Information

27.1 PC

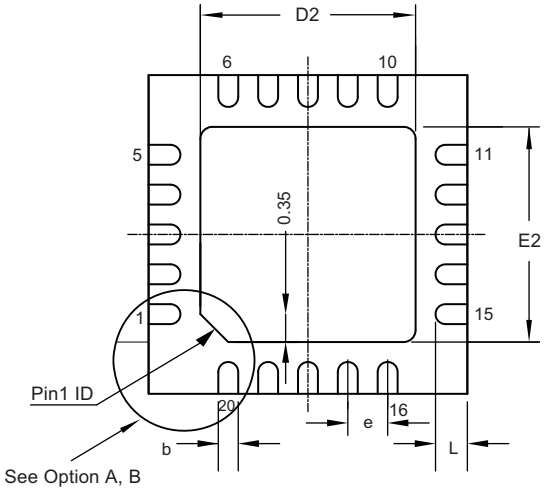
DRAWINGS NOT SCALED



Top View



Side View



Bottom View

COMMON DIMENSIONS

(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.70	0.75	0.80	
A1	0.00	0.02	0.05	
D/E	3.90	4.00	4.10	
D2/E2	2.50	2.60	2.70	
L	0.35	0.45	0.55	
b	0.20	0.25	0.30	
e	0.50 BSC			
Tolerances of Form and Position				
ccc	0.10			
eee	0.08			
n	20			

Option A

Pin 1# Chamfer
(C 0.35 max)

Option B

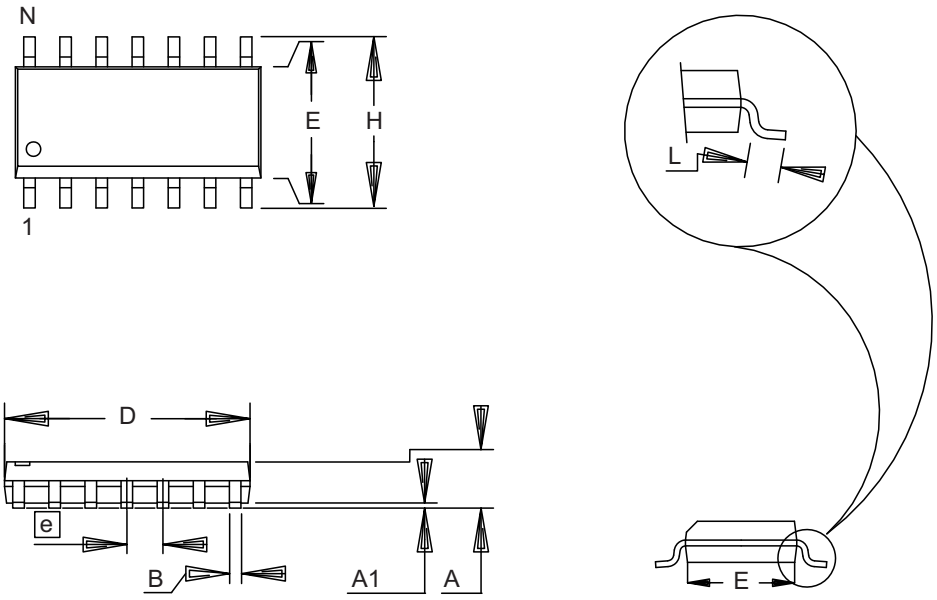
Pin 1# Notch
(0.20 R)

Notes: 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation WGGD-5 for proper dimensions, tolerances, datums, etc. (excepted D2/E2 Min et Nom.)
2. Dimensions b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip. If the terminal has the optical radius on the other end of the terminal, the dimensions should not be measured in that radius area.

08/01/14

Atmel **Package Drawing Contact:** packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	PC, 20-Lds - 0.50mm Pitch, 4x4x0.8mm Body size Very very Thin Quad Flat No Lead Package (WQFN) Sawn	ZVF	PC	J

27.2 TU



	MM			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.35	1.60	1.75	.053	.063	.069
A1	0.10	----	0.25	.004	----	.010
B	0.33	0.41	0.51	.013	.016	.020
D	8.53	8.64	8.74	.336	.340	.344
E	3.80	3.91	3.99	.149	.154	.157
H	5.79	5.99	6.20	.228	.236	.244
L	0.40	0.71	1.27	.016	.028	.050
e	1.27 BSC			.050 BSC		

07/27/07

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

28. Errata

The revision letter in this section refers to the revision of the Atmel® ATtiny24/44/84 Automotive device.

28.1 ATtiny24 Automotive

28.1.1 Rev. E

1. No known errata.

28.2 ATtiny44 Automotive

28.2.1 Rev. D

1. No known errata.

28.3 ATtiny84 Automotive

28.3.1 Rev. B

1. No known errata.

29. Revision History

Please note that the following page numbers referred to in this section refer to the specific revision mentioned, not to this document.

Revision No.	History
7701G-AVR-02/15	• Put datasheet in the latest template • Section 27 “Packaging Information” on pages 188 to 189 updated
7701F-AVR-10/12	• Section 27 “Packaging Information” on page 214 updated.
7701E-AVR-02/11	• MCUCR registers updated
7701D-AVR-09/10	• BOD values updated.
7701C-AVR-11/08	• Internal RC oscillator accuracy update. See Section 7.6 “Calibrated Internal RC Oscillator” on page 27. • ADC characteristics update. See Section 22.5 “ADC Characteristics – Preliminary Data” on page 159 • DC characteristics update. See Table 22-1 on page 155. • Brown-out detector hysteresis See Table 22-4 on page 158. • Calibrated Internal RC Oscillator Accuracy update. See Section 22-2 “Calibration Accuracy of Internal RC Oscillator” on page 157
7701B-AVR-09/07	• DC Characteristics updated. Section 22. “Electrical Characteristics” on page 155. • ADC maximum resolution corrected. Section 18. “Analog-to-Digital Converter” on page 118. • POR value updated. Table 9-1 on page 37.
7701A-AVR-05/07	• Initial automotive version. Started from industrial specification document 8006 rev. E 09/06.

30. Table of Contents

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**