Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 54 Powerful Instructions – Most Single Clock Cycle Execution
  - 16 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 12 MIPS Throughput at 12 MHz
- Non-volatile Program and Data Memories
  - 512/1024 Bytes of In-System Programmable Flash Program Memory
  - 32 Bytes Internal SRAM
  - Flash Write/Erase Cycles: 10,000
  - Data Retention: 20 Years at 85°C / 100 Years at 25°C
- Peripheral Features
  - One 16-bit Timer/Counter with Prescaler and Two PWM Channels
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - 4-channel, 8-bit Analog to Digital Converter (1)
  - On-chip Analog Comparator
- Special Microcontroller Features
  - In-System Programmable (2)
  - External and Internal Interrupt Sources
  - Low Power Idle, ADC Noise Reduction, and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Supply Voltage Level Monitor with Interrupt and Reset
  - Internal Calibrated Oscillator
- I/O and Packages
  - 6-pin SOT: Four Programmable I/O Lines
- Operating Voltage:
  - 1.8 – 5.5V
- Programming Voltage:
  - 5V
- Speed Grade
  - 0 – 4 MHz @ 1.8 – 5.5V
  - 0 – 8 MHz @ 2.7 – 5.5V
  - 0 – 12 MHz @ 4.5 – 5.5V
- Industrial Temperature Range
- Low Power Consumption
  - Active Mode:
    - 200µA at 1MHz and 1.8V
  - Idle Mode:
    - 25µA at 1MHz and 1.8V
  - Power-down Mode:
    - < 0.1µA at 1.8V

Note: 1. The Analog to Digital Converter (ADC) is available in ATtiny5/10, only
2. At 5V, only

8-bit AVR® Microcontroller with 512/1024 Bytes In-System Programmable Flash

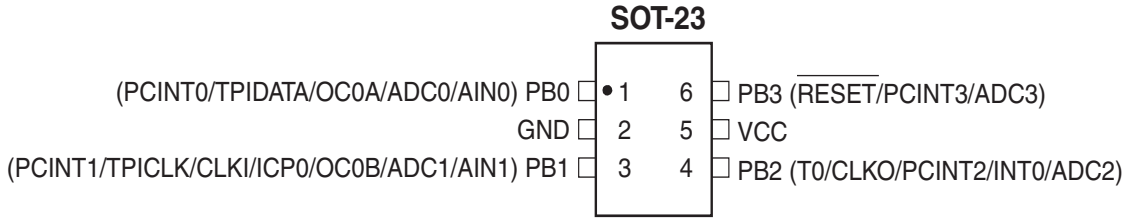
ATtiny4/5/9/10

Preliminary

8127C–AVR–10/09

1. Pin Configurations

Figure 1-1. Pinout of ATtiny4/5/9/10



1.1 Pin Description

1.1.1 VCC

Supply voltage.

1.1.2 GND

Ground.

1.1.3 Port B (PB3..PB0)

This is a 4-bit, bi-directional I/O port with internal pull-up resistors, individually selectable for each bit. The output buffers have symmetrical drive characteristics, with both high sink and source capability. As inputs, the port pins that are externally pulled low will source current if pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

The port also serves the functions of various special features of the ATtiny4/5/9/10, as listed on page 36.

1.1.4 $\overline{\text{RESET}}$

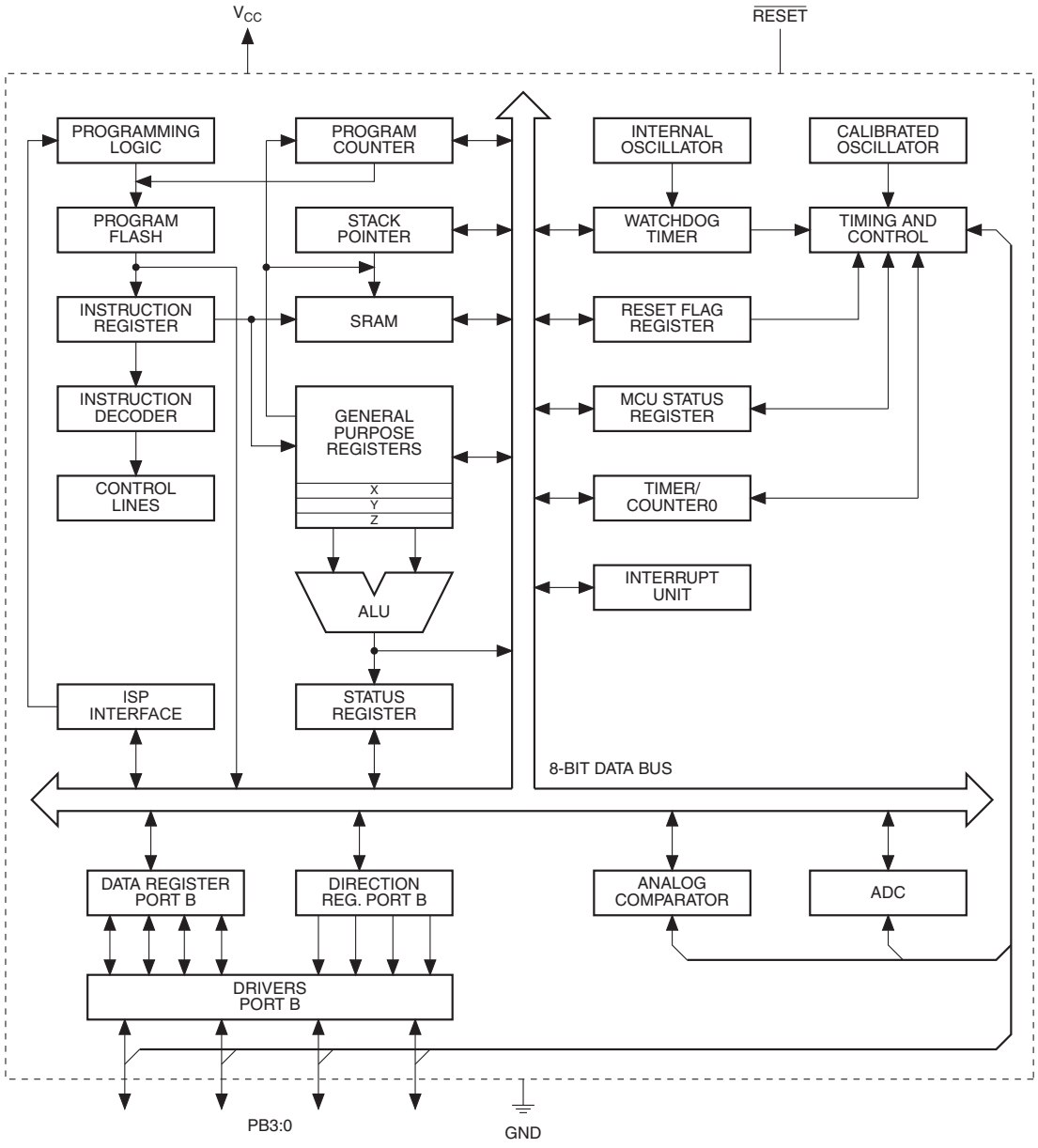
Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running and provided the reset pin has not been disabled. The minimum pulse length is given in Table 16-4 on page 119. Shorter pulses are not guaranteed to generate a reset.

The reset pin can also be used as a (weak) I/O pin.

2. Overview

ATtiny4/5/9/10 are low-power CMOS 8-bit microcontrollers based on the compact AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny4/5/9/10 achieve throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 16 general purpose working registers and system registers. All 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 compact and code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATtiny4/5/9/10 provide the following features: 512/1024 byte of In-System Programmable Flash, 32 bytes of SRAM, four general purpose I/O lines, 16 general purpose working registers, a 16-bit timer/counter with two PWM channels, internal and external interrupts, a programmable watchdog timer with internal oscillator, an internal calibrated oscillator, and four software selectable power saving modes. ATtiny5/10 are also equipped with a four-channel, 8-bit Analog to Digital Converter (ADC).

Idle mode stops the CPU while allowing the SRAM, timer/counter, ADC (ATtiny5/10, only), analog comparator, and interrupt system to continue functioning. ADC Noise Reduction mode minimizes switching noise during ADC conversions by stopping the CPU and all I/O modules except the ADC. In Power-down mode registers keep their contents and all chip functions are disabled until the next interrupt or hardware reset. In Standby mode, the oscillator is running while the rest of the device is sleeping, allowing 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, in-system programmable Flash allows program memory to be re-programmed in-system by a conventional, non-volatile memory programmer.

The ATtiny4/5/9/10 AVR are supported by a suite of program and system development tools, including macro assemblers and evaluation kits.

2.1 Comparison of ATtiny4, ATtiny5, ATtiny9 and ATtiny10

A comparison of the devices is shown in Table 2-1.

Table 2-1. Differences between ATtiny4, ATtiny5, ATtiny9 and ATtiny10

Device	Flash	ADC	Signature
ATtiny4	512 bytes	No	0x1E 0x8F 0x0A
ATtiny5	512 bytes	Yes	0x1E 0x8F 0x09
ATtiny9	1024 bytes	No	0x1E 0x90 0x08
ATtiny10	1024 bytes	Yes	0x1E 0x90 0x03

3. General Information

3.1 Resources

A comprehensive set of drivers, application notes, data sheets and descriptions on development tools are available for download at http://www.atmel.com/avr.

3.2 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.

3.3 Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

3.4 Disclaimer

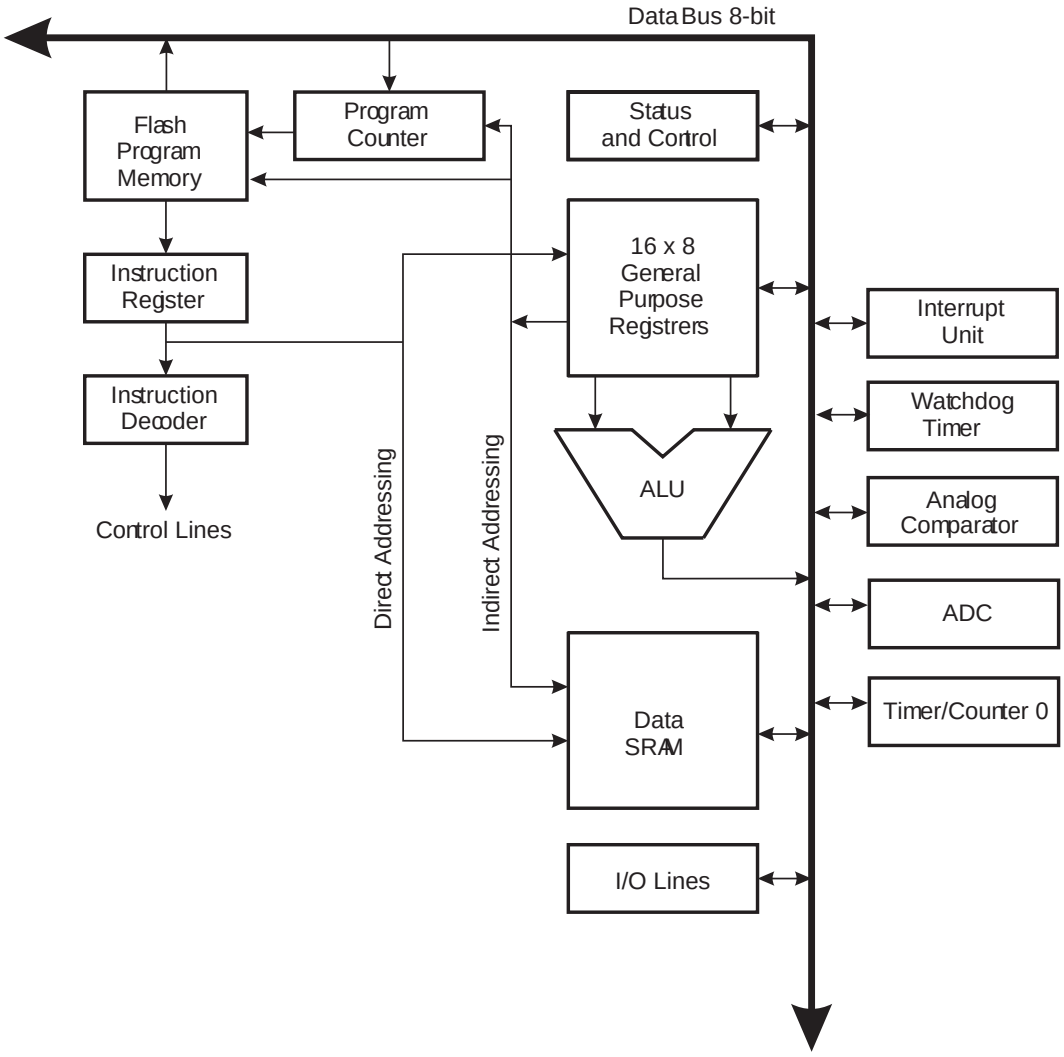
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device has been characterized.

4. CPU Core

This section discusses the 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.

4.1 Architectural Overview

Figure 4-1. 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 16 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 – in one clock cycle.

Six of the 16 registers can be used as three 16-bit indirect address register pointers for data space addressing – enabling efficient address calculations. One of the these 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-register, 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, capable of directly addressing the whole address space. Most AVR instructions have a single 16-bit word format but 32-bit wide instructions also exist. The actual instruction set varies, as some devices only implement a part of the instruction set.

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 SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the four different addressing modes supported in the AVR architecture.

The memory spaces in the 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 as Control Registers, SPI, and other I/O functions. The I/O memory can be accessed as the data space locations, 0x0000 - 0x003F.

4.2 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 16 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 document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for a detailed description.

4.3 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 document “AVR Instruction Set” and section “Instruction Set Summary” on page 151. 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.

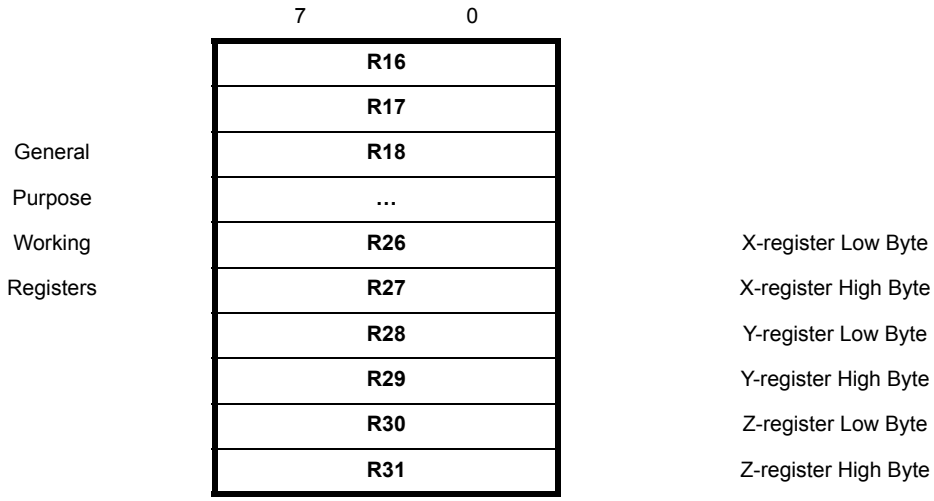
4.4 General Purpose Register File

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

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

Figure 4-2 below shows the structure of the 16 general purpose working registers in the CPU.

Figure 4-2. AVR CPU General Purpose Working Registers



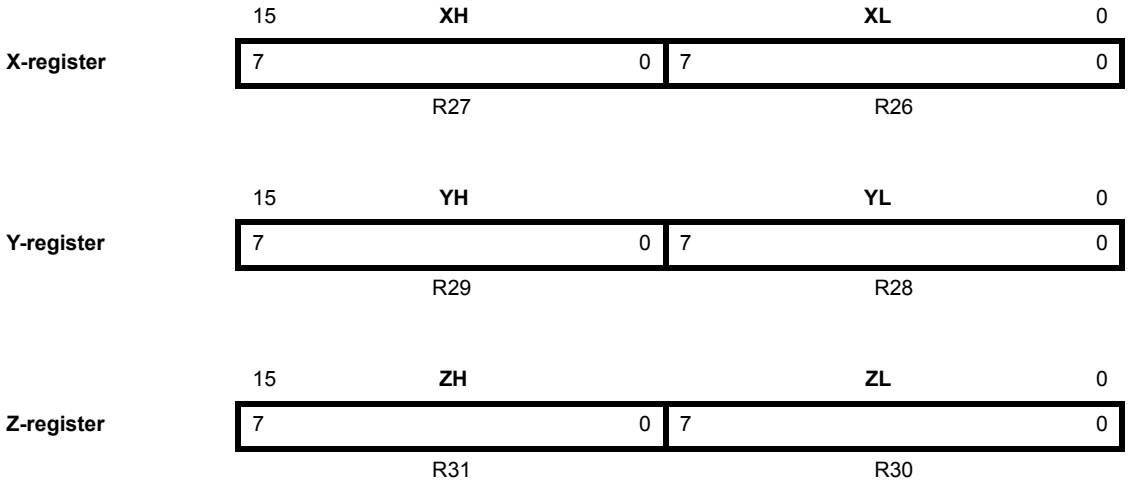
Note: A typical implementation of the AVR register file includes 32 general prupose registers but ATtiny4/5/9/10 implement only 16 registers. For reasons of compatibility the registers are numbered R16...R31, not R0...R15.

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

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

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 4-3.

Figure 4-3. The X-, Y-, and Z-registers



In different addressing modes these address registers function as automatic increment and automatic decrement (see document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for details).

4.5 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 command 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 0x40. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The 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 SPL is needed. In this case, the SPH Register will not be present.

4.6 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The 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 4-4. The Parallel Instruction Fetches and Instruction Executions

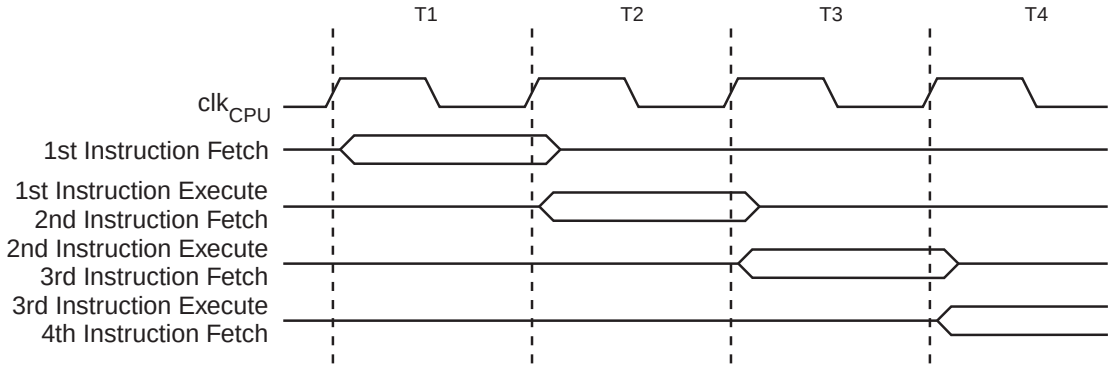
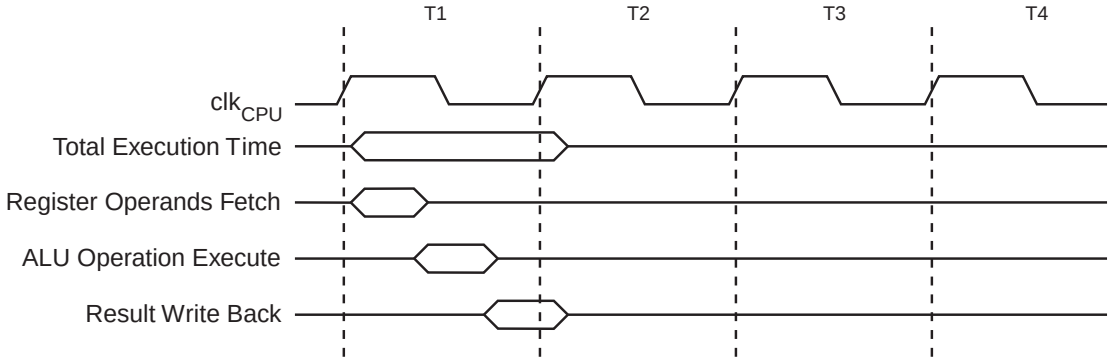


Figure 4-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks,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

Figure 4-5 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 4-5. Single Cycle ALU Operation



4.7 Reset and Interrupt Handling

The 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 logic 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 “Interrupts” on page 35. The list also determines the priority levels of the different interrupts. The lower the address the higher is 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 logic one to the I-bit to enable nested interrupts. All enabled

interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

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

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

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

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

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction.

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

Assembly Code Example

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

Note: See “Code Examples” on page 5.

4.7.1 Interrupt Response Time

The interrupt execution response for all the enabled 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.

4.8 Register Description

4.8.1 CCP – Configuration Change Protection Register

Bit	7	6	5	4	3	2	1	0	
0x3C	CCP[7:0]								CCP
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – CCP[7:0] – Configuration Change Protection**

In order to change the contents of a protected I/O register the CCP register must first be written with the correct signature. After CCP is written the protected I/O registers may be written to during the next four CPU instruction cycles. All interrupts are ignored during these cycles. After these cycles interrupts are automatically handled again by the CPU, and any pending interrupts will be executed according to their priority.

When the protected I/O register signature is written, CCP[0] will read as one as long as the protected feature is enabled, while CCP[7:1] will always read as zero.

Table 4-1 shows the signatures that are in recognised.

Table 4-1. Signatures Recognised by the Configuration Change Protection Register

Signature	Group	Description
0xD8	IOREG: CLKMSR, CLKPSR, WDTCSR	Protected I/O register

4.8.2 SPH and SPL — Stack Pointer Register

Bit	15	14	13	12	11	10	9	8	
0x3E	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
0x3D	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	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

4.8.3 SREG – Status Register

Bit	7	6	5	4	3	2	1	0	
0x3F	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 independent 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 document “AVR Instruction Set” and “Instruction Set Summary” on page 151.

- **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 document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 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 document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for detailed information.

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

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for detailed information.

5. Memories

This section describes the different memories in the ATtiny4/5/9/10. Devices have two main memory areas, the program memory space and the data memory space.

5.1 In-System Re-programmable Flash Program Memory

The ATtiny4/5/9/10 contain 512/1024 bytes of on-chip, in-system reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 256/512 x 16.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATtiny4/5/9/10 Program Counter (PC) is 9 bits wide, thus capable of addressing the 256/512 program memory locations, starting at 0x000. “Memory Programming” on page 107 contains a detailed description on Flash data serial downloading.

Constant tables can be allocated within the entire address space of program memory. Since program memory can not be accessed directly, it has been mapped to the data memory. The mapped program memory begins at byte address 0x4000 in data memory (see Figure 5-1 on page 15). Although programs are executed starting from address 0x000 in program memory it must be addressed starting from 0x4000 when accessed via the data memory.

Internal write operations to Flash program memory have been disabled and program memory therefore appears to firmware as read-only. Flash memory can still be written to externally but internal write operations to the program memory area will not be succesful.

Timing diagrams of instruction fetch and execution are presented in “Instruction Execution Timing” on page 9.

5.2 Data Memory

Data memory locations include the I/O memory, the internal SRAM memory, the non-volatile memory lock bits, and the Flash memory. See Figure 5-1 on page 15 for an illustration on how the ATtiny4/5/9/10 memory space is organized.

The first 64 locations are reserved for I/O memory, while the following 32 data memory locations address the internal data SRAM.

The non-volatile memory lock bits and all the Flash memory sections are mapped to the data memory space. These locations appear as read-only for device firmware.

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

The IN and OUT instructions can access all 64 locations of I/O memory. Direct addressing using the LDS and STS instructions reaches the 128 locations between 0x0040 and 0x00BF.

The indirect addressing reaches the entire data memory space. When using indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

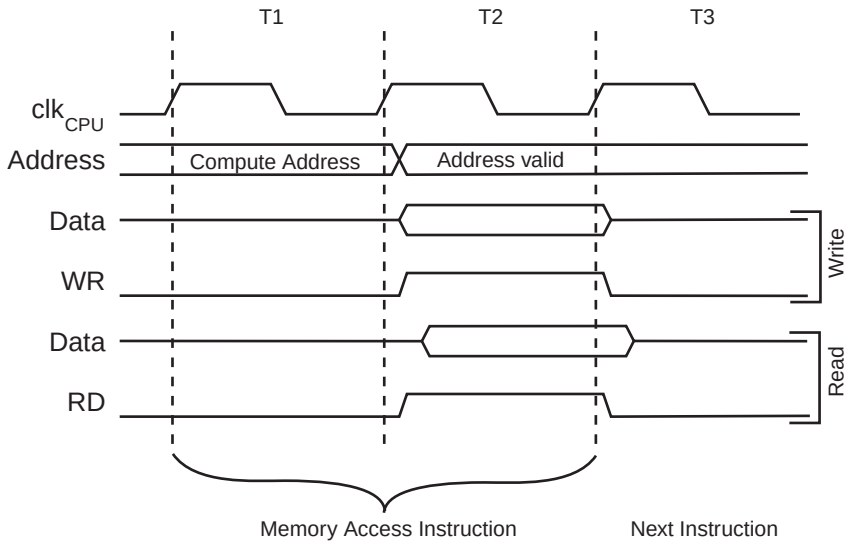
Figure 5-1. Data Memory Map (Byte Addressing)

I/O SPACE	0x0000 ... 0x003F
SRAM DATA MEMORY	0x0040 ... 0x005F
(reserved)	0x0060 ... 0x3EFF
NVM LOCK BITS	0x3F00 ... 0x3F01
(reserved)	0x3F02 ... 0x3F3F
CONFIGURATION BITS	0x3F40 ... 0x3F41
(reserved)	0x3F42 ... 0x3F7F
CALIBRATION BITS	0x3F80 ... 0x3F81
(reserved)	0x3F82 ... 0x3FBF
DEVICE ID BITS	0x3FC0 ... 0x3FC3
(reserved)	0x3FC4 ... 0x3FFF
FLASH PROGRAM MEMORY	0x4000 ... 0x41FF/0x43FF
(reserved)	0x4400 ... 0xFFFF

5.2.1 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 5-2.

Figure 5-2. On-chip Data SRAM Access Cycles



5.3 I/O Memory

The I/O space definition of the ATtiny4/5/9/10 is shown in “Register Summary” on page 149.

All ATtiny4/5/9/10 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed using the LD and ST instructions, enabling data transfer between the 16 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 document “AVR Instruction Set” and section “Instruction Set Summary” on page 151 for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used.

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

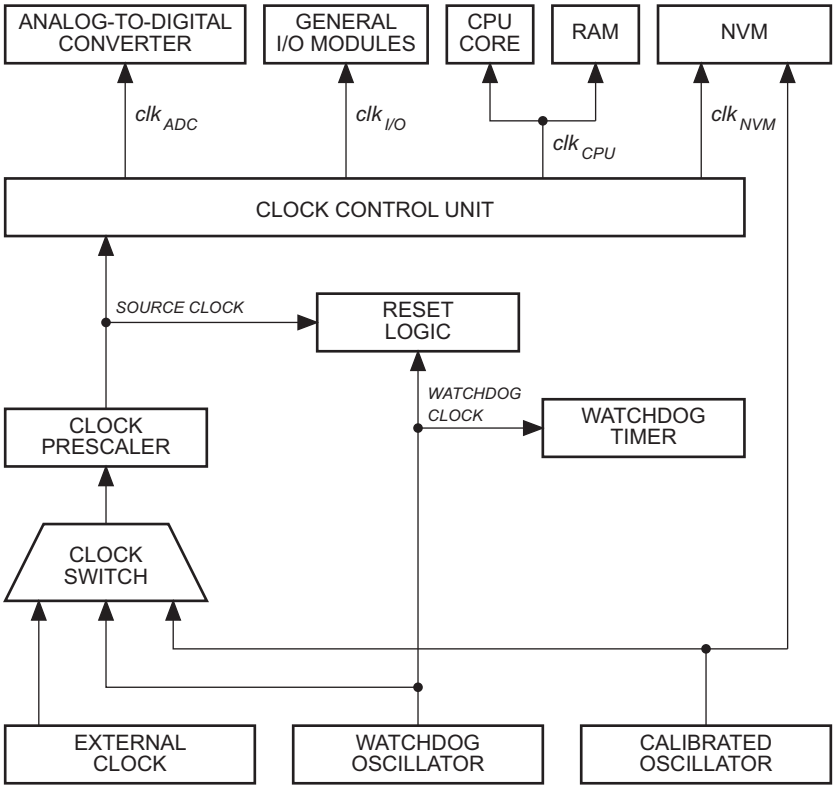
Some of the status flags are cleared by writing a logical one to them. Note that CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work on registers in the address range 0x00 to 0x1F, only.

The I/O and Peripherals Control Registers are explained in later sections.

6. Clock System

Figure 6-1 presents the principal clock systems and their distribution in ATtiny4/5/9/10. 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 and power reduction register bits, as described in “Power Management and Sleep Modes” on page 23. The clock systems is detailed below.

Figure 6-1. Clock Distribution



6.1 Clock Subsystems

The clock subsystems are detailed in the sections below.

6.1.1 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 System Registers and the SRAM data memory. Halting the CPU clock inhibits the core from performing general operations and calculations.

6.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like 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.

6.1.3 NVM clock - clk_{NVM}

The NVM clock controls operation of the Non-Volatile Memory Controller. The NVM clock is usually active simultaneously with the CPU clock.

6.1.4 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 digital circuitry. This gives more accurate ADC conversion results.

The ADC is available in ATtiny5/10, only.

6.2 Clock Sources

All synchronous clock signals are derived from the main clock. The device has three alternative sources for the main clock, as follows:

- Calibrated Internal 8 MHz Oscillator (see page 18)
- External Clock (see page 18)
- Internal 128 kHz Oscillator (see page 19)

See Table 6-3 on page 21 on how to select and change the active clock source.

6.2.1 Calibrated Internal 8 MHz Oscillator

The calibrated internal oscillator provides an approximately 8 MHz clock signal. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. See Table 16-2 on page 118, Figure 17-39 on page 142 and Figure 17-40 on page 142 for more details.

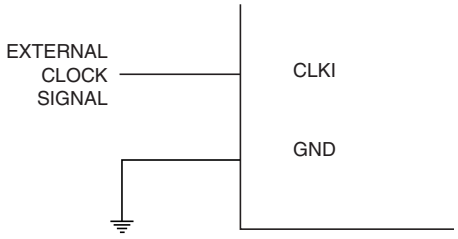
This clock may be selected as the main clock by setting the Clock Main Select bits CLKMS[1:0] in CLKMSR to 0b00. Once enabled, the oscillator will operate with no external components. During reset, hardware loads the calibration byte into the OSCCAL register and thereby automatically calibrates the oscillator. The accuracy of this calibration is shown as Factory calibration in Table 16-2 on page 118.

When this oscillator is used as the main clock, the watchdog oscillator will still be used for the watchdog timer and reset time-out. For more information on the pre-programmed calibration value, see section “Calibration Section” on page 110.

6.2.2 External Clock

To use the device with an external clock source, CLKI should be driven as shown in Figure 6-2. The external clock is selected as the main clock by setting CLKMS[1:0] bits in CLKMSR to 0b10.

Figure 6-2. External Clock Drive Configuration



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

6.2.3 Internal 128 kHz Oscillator

The internal 128 kHz oscillator is a low power oscillator providing a clock of 128 kHz. The frequency depends on supply voltage, temperature and batch variations. This clock may be select as the main clock by setting the CLKMS[1:0] bits in CLKMSR to 0b01.

6.2.4 Switching Clock Source

The main clock source can be switched at run-time using the “CLKMSR – Clock Main Settings Register” on page 21. When switching between any clock sources, the clock system ensures that no glitch occurs in the main clock.

6.2.5 Default Clock Source

The calibrated internal 8 MHz oscillator is always selected as main clock when the device is powered up or has been reset. The synchronous system clock is the main clock divided by 8, controlled by the System Clock Prescaler. The Clock Prescaler Select Bits can be written later to change the system clock frequency. See “System Clock Prescaler”.

6.3 System Clock Prescaler

The system clock is derived from the main clock via the System Clock Prescaler. The system clock can be divided by setting the “CLKPSR – Clock Prescale Register” on page 22. The system clock prescaler can be used to decrease power consumption at times when requirements for processing power is low or to bring the system clock within limits of maximum frequency. The prescaler can be used with all main clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals.

The System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation.

6.3.1 Switching Prescaler Setting

When switching between prescaler settings, the system clock prescaler ensures that no glitch occurs in the system clock and that no intermediate frequency is higher than neither the clock frequency corresponding the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the main 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, two active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

6.4 Starting

6.4.1 Starting from Reset

The internal reset is immediately asserted when a reset source goes active. The internal reset is kept asserted until the reset source is released and the start-up sequence is completed. The start-up sequence includes three steps, as follows.

1. The first step after the reset source has been released consists of the device counting the reset start-up time. The purpose of this reset start-up time is to ensure that supply voltage has reached sufficient levels. The reset start-up time is counted using the internal 128 kHz oscillator. See Table 6-1 for details of reset start-up time.
   Note that the actual supply voltage is not monitored by the start-up logic. The device will count until the reset start-up time has elapsed even if the device has reached sufficient supply voltage levels earlier.
2. The second step is to count the oscillator start-up time, which ensures that the calibrated internal oscillator has reached a stable state before it is used by the other parts of the system. The calibrated internal oscillator needs to oscillate for a minimum number of cycles before it can be considered stable. See Table 6-1 for details of the oscillator start-up time.
3. The last step before releasing the internal reset is to load the calibration and the configuration values from the Non-Volatile Memory to configure the device properly. The configuration time is listed in Table 6-1.

Table 6-1. Start-up Times when Using the Internal Calibrated Oscillator

Reset	Oscillator	Configuration	Total start-up time
64 ms	6 cycles	21 cycles	64 ms + 6 oscillator cycles + 21 system clock cycles (1)

Notes: 1. After powering up the device or after a reset the system clock is automatically set to calibrated internal 8 MHz oscillator, divided by 8

6.4.2 Starting from Power-Down Mode

When waking up from Power-Down sleep mode, the supply voltage is assumed to be at a sufficient level and only the oscillator start-up time is counted to ensure the stable operation of the oscillator. The oscillator start-up time is counted on the selected main clock, and the start-up time depends on the clock selected. See Table 6-2 for details.

Table 6-2. Start-up Time from Power-Down Sleep Mode.

Oscillator start-up time	Total start-up time
6 cycles	6 oscillator cycles (1)

Notes: 1. The start-up time is measured in main clock oscillator cycles.

6.4.3 Starting from Idle / ADC Noise Reduction / Standby Mode

When waking up from Idle, ADC Noise Reduction or Standby Mode, the oscillator is already running and no oscillator start-up time is introduced.

The ADC is available in ATtiny5/10, only.

6.5 Register Description

6.5.1 CLKMSR – Clock Main Settings Register

Bit	7	6	5	4	3	2	1	0	
0x37	–	–	–	–	–	–	CLKMS1	CLKMS0	CLKMSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:2 – Res: Reserved Bits**

These bits are reserved and always read zero.

- **Bit 1:0 – CLKMS[1:0]: Clock Main Select Bits**

These bits select the main clock source of the system. The bits can be written at run-time to switch the source of the main clock. The clock system ensures glitch free switching of the main clock source.

The main clock alternatives are shown in Table 6-3.

Table 6-3. Selection of Main Clock

CLKM1	CLKM0	Main Clock Source
0	0	Calibrated Internal 8 MHzOscillator
0	1	Internal 128 kHz Oscillator (WDT Oscillator)
1	0	External clock
1	1	Reserved

To avoid unintentional switching of main clock source, a protected change sequence must be followed to change the CLKMS bits, as follows:

1. Write the signature for change enable of protected I/O register to register CCP
2. Within four instruction cycles, write the CLKMS bits with the desired value

6.5.2 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
0x39	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	0	0	0	0	0	0	0	0	

- **Bits 7:0 – CAL[7:0]: Oscillator Calibration Value**

The oscillator calibration register is used to trim the calibrated internal oscillator and 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 16-2, “Calibration Accuracy of Internal RC Oscillator,” on page 118.

The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 16-2, “Calibration Accuracy of Internal RC Oscillator,” on page 118. Calibration outside the range given is not guaranteed.

The CAL[7:0] bits are used to tune the frequency of the oscillator. A setting of 0x00 gives the lowest frequency, and a setting of 0xFF gives the highest frequency.

6.5.3 CLKPSR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
0x36	–	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	1	1	

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bits 3:0 – CLKPS[3:0]: 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 and suit the application requirements. As the prescaler divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced accordingly. The division factors are given in Table 6-4.

Table 6-4. 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 (default)
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

To avoid unintentional changes of clock frequency, a protected change sequence must be followed to change the CLKPS bits:

1. Write the signature for change enable of protected I/O register to register CCP
2. Within four instruction cycles, write the desired value to CLKPS bits

At start-up, CLKPS bits are reset to 0b0011 to select the clock division factor of 8. If the selected clock source has a frequency higher than the maximum allowed the application software must make sure a sufficient division factor is used. To make sure the write procedure is not interrupted, interrupts must be disabled when changing prescaler settings.

7. Power Management and Sleep Modes

The high performance and industry leading code efficiency makes the AVR microcontrollers an ideal choise for low power applications. In addition, sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

7.1 Sleep Modes

Figure 6-1 on page 17 presents the different clock systems and their distribution in ATtiny4/5/9/10. The figure is helpful in selecting an appropriate sleep mode. Table 7-1 shows the different sleep modes and their wake up sources.

Table 7-1. Active Clock Domains and Wake-up Sources in Different Sleep Modes

	Active Clock Domains				Oscillators	Wake-up Sources				
Sleep Mode	clk_{CPU}	clk_{NVM}	clk_{IO}	clk_{ADC} (1)	Main Clock Source Enabled	INT0 and Pin Change	ADC (1)	Other I/O	Watchdog Interrupt	VLM Interrupt
Idle			X	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X (2)	X		X	X
Standby					X	X (2)			X	
Power-down						X (2)			X	

Note: 1. The ADC is available in ATtiny5/10, only
2. For INT0, only level interrupt.

To enter any of the four sleep modes, the SE bits in SMCR must be written to logic one and a SLEEP instruction must be executed. The SM2:0 bits in the SMCR register select which sleep mode (Idle, ADC Noise Reduction, Standby or Power-down) will be activated by the SLEEP instruction. See Table 7-2 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.

Note that if a level triggered interrupt is used for wake-up the changed level must be held for some time to wake up the MCU (and for the MCU to enter the interrupt service routine). See "External Interrupts" on page 36 for details.

7.1.1 Idle Mode

When bits SM2:0 are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the analog comparator, timer/counter, watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{NVM}, 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 "ACSR – Analog Comparator Control and Status Register" on page 81. This will reduce power consumption in idle mode. If the ADC is enabled (ATtiny5/10, only), a conversion starts automatically when this mode is entered.

7.1.2 ADC Noise Reduction Mode

When bits SM2:0 are written to 001, 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_{NVM}, while allowing the other clocks to run.

This mode 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.

This mode is available in all devices, although only ATtiny5/10 are equipped with an ADC.

7.1.3 Power-down Mode

When bits SM2:0 are written to 010, 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 a watchdog 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.

7.1.4 Standby Mode

When bits SM2:0 are written to 100, 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. This reduces wake-up time, because the oscillator is already running and doesn't need to be started up.

7.2 Power Reduction Register

The Power Reduction Register (PRR), see "PRR – Power Reduction Register" on page 26, provides a method to reduce power consumption by stopping the clock to individual peripherals. When the clock for a peripheral is stopped then:

- The current state of the peripheral is frozen.
- The associated registers can not be read or written.
- Resources used by the peripheral will remain occupied.

The peripheral should in most cases be disabled before stopping the clock. Clearing the PRR bit wakes up the peripheral and puts it in the same state as before shutdown.

Peripheral shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. See "Supply Current of I/O Modules" on page 122 for examples. In all other sleep modes, the clock is already stopped.

7.3 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR Core controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

7.3.1 Analog Comparator

When entering Idle mode, the analog comparator should be disabled if not used. In the power-down mode, the analog comparator is automatically disabled. See “Analog Comparator” on page 81 for further details.

7.3.2 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 “Analog to Digital Converter” on page 83 for details on ADC operation.

The ADC is available in ATtiny5/10, only.

7.3.3 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. Refer to “Watchdog Timer” on page 30 for details on how to configure the Watchdog Timer.

7.3.4 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 the I/O clock ($clk_{I/O}$) is 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. Refer to the section “Digital Input Enable and Sleep Modes” on page 44 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). Refer to “DIDR0 – Digital Input Disable Register 0” on page 82 for details.

7.4 Register Description

7.4.1 SMCR – Sleep Mode Control Register

The SMCR Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x3A	–	–	–	–	SM2	SM1	SM0	SE	SMCR
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 and will always read zero.

- **Bits 3:1 – SM2..SM0: Sleep Mode Select Bits 2..0**

These bits select between available sleep modes, as shown in Table 7-2.

Table 7-2. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC noise reduction (1)
0	1	0	Power-down
0	1	1	Reserved
1	0	0	Standby
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Note: 1. This mode is available in all devices, although only ATtiny5/10 are equipped with an ADC

- **Bit 0 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

7.4.2 PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
0x35	–	–	–	–	–	–	PRADC	PRTIM0	PRR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 1 – PRADC: Power Reduction ADC**

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

The ADC is available in ATtiny5/10, only.

- **Bit 0 – PRTIM0: Power Reduction Timer/Counter0**

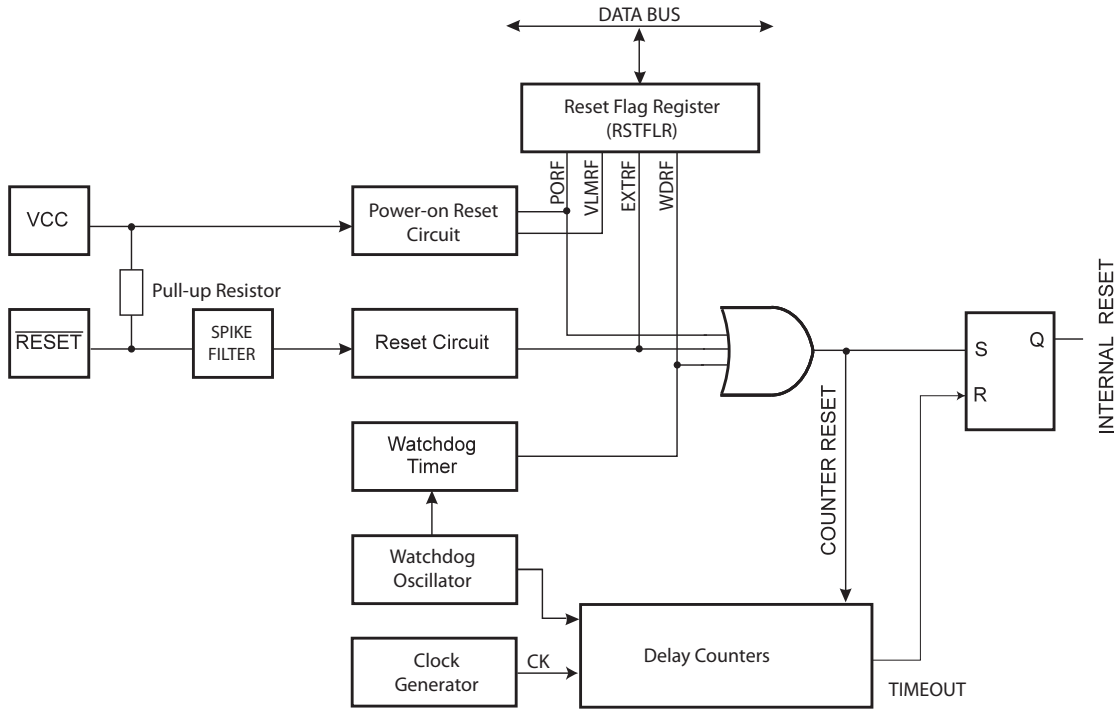
Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

8. System Control and Reset

8.1 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 8-1 shows the reset logic. Electrical parameters of the reset circuitry are defined in section "System and Reset Characteristics" on page 119.

Figure 8-1. Reset Logic



The I/O ports of the 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 start up sequence is described in "Starting from Reset" on page 20.

8.2 Reset Sources

The ATtiny4/5/9/10 have three 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
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled

8.2.1 Power-on Reset

A Power-on Reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in section "System and Reset Characteristics" on page 119. 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 Power-on Reset (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 8-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

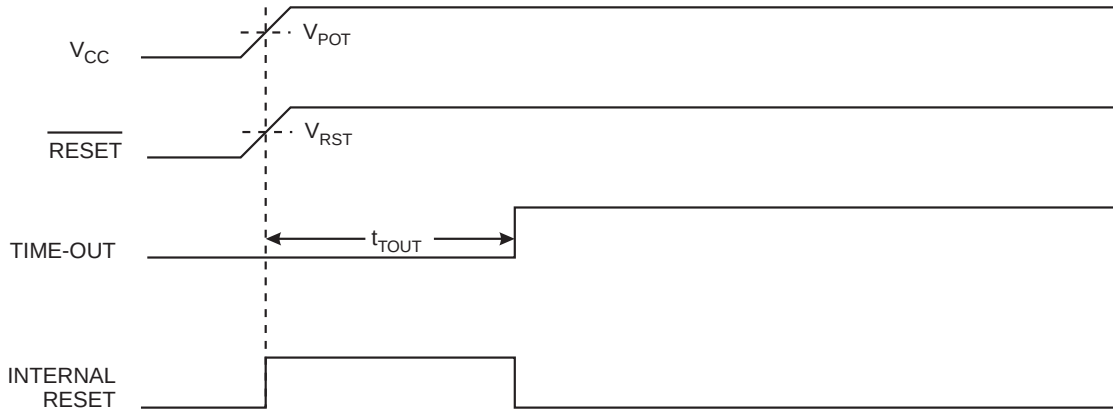
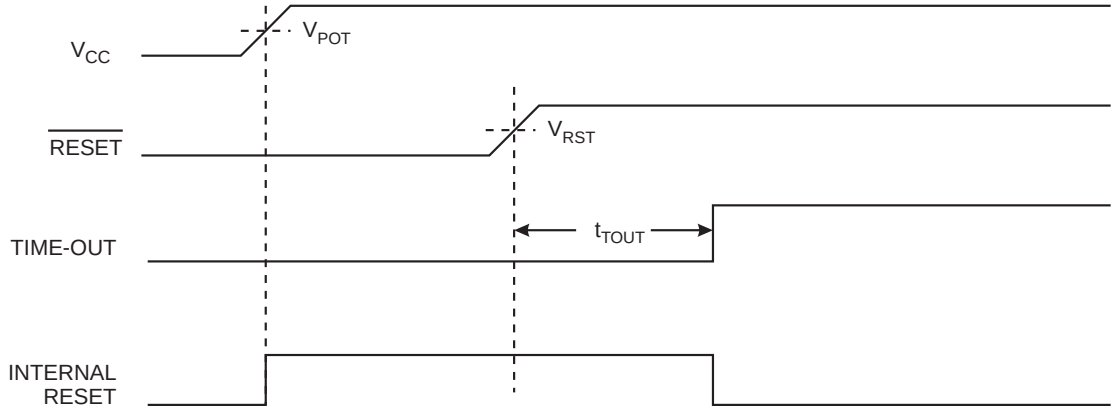


Figure 8-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



8.2.2 V_{CC} Level Monitoring

ATtiny4/5/9/10 have a V_{CC} Level Monitoring (VLM) circuit that compares the voltage level at the V_{CC} pin against fixed trigger levels. The trigger levels are set with VLM2:0 bits, see "VLMCSR – VCC Level Monitoring Control and Status register" on page 33.

The VLM circuit provides a status flag, VLMF, that indicates if voltage on the V_{CC} pin is below the selected trigger level. The flag can be read from VLMCSR, but it is also possible to have an interrupt generated when the VLMF status flag is set. This interrupt is enabled by the VLMIE bit in the VLMCSR register. The flag can be cleared by changing the trigger level or by writing it to zero. The flag is automatically cleared when the voltage at V_{CC} rises back above the selected trigger level.

The VLM can also be used to improve reset characteristics at falling supply. Without VLM, the Power-On Reset (POR) does not activate before supply voltage has dropped to a level where the MCU is not necessarily functional any more. With VLM, it is possible to generate a reset at supply voltages where the MCU is still functional.

When active, the VLM circuit consumes some power, as illustrated in Figure 17-48 on page 146. To save power the VLM circuit can be turned off completely, or it can be switched on and off at regular intervals. However, detection takes some time and it is therefore recommended to leave the circuitry on long enough for signals to settle. See "VCC Level Monitor" on page 119.

When VLM is active and voltage at V_{CC} is above the selected trigger level operation will be as normal and the VLM can be shut down for a short period of time. If voltage at V_{CC} drops below the selected threshold the VLM will either flag an interrupt or generate a reset, depending on the configuration.

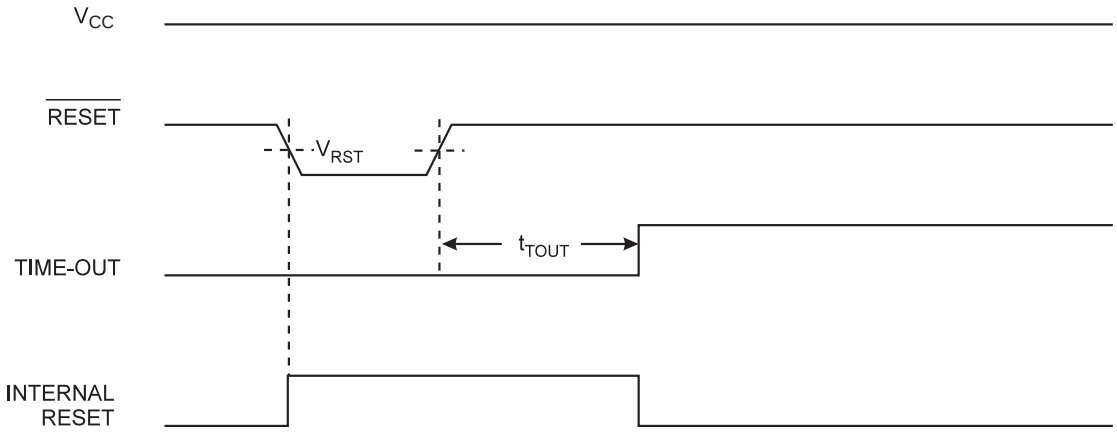
When the VLM has been configured to generate a reset at low supply voltage it will keep the device in reset as long as V_{CC} is below the reset level. See Table 8-4 on page 34 for reset level details. If supply voltage rises above the reset level the condition is removed and the MCU will come out of reset, and initiate the power-up start-up sequence.

If supply voltage drops enough to trigger the POR then PORF is set after supply voltage has been restored.

8.2.3 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 "System and Reset Characteristics" on page 119) 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.

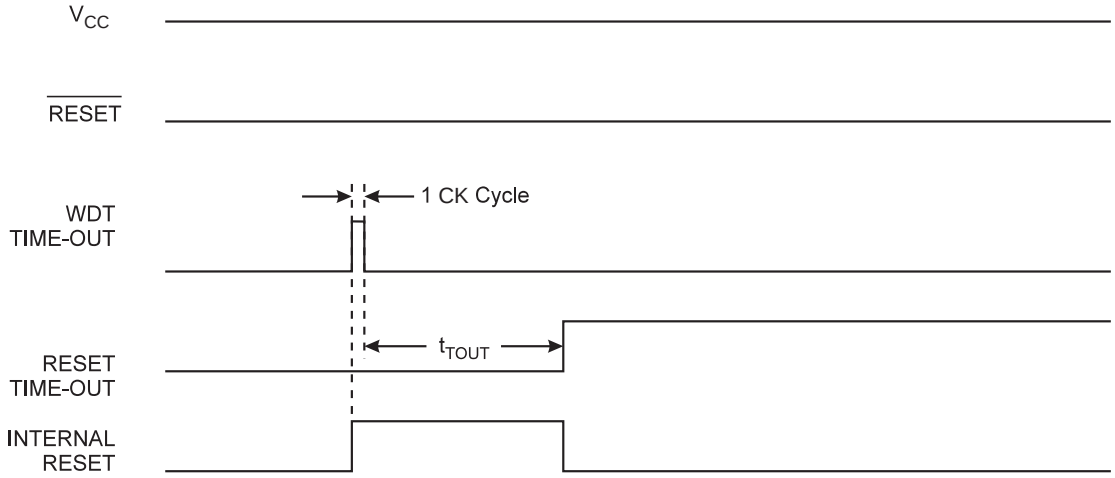
Figure 8-4. External Reset During Operation



8.2.4 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 page 30 for details on operation of the Watchdog Timer and Table 16-4 on page 119 for details on reset time-out.

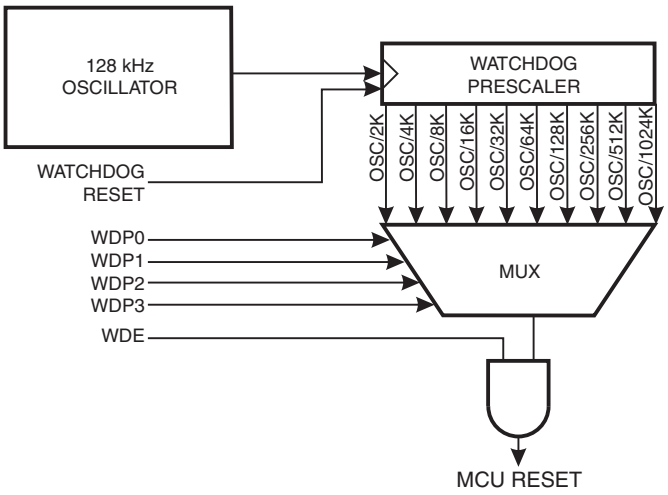
Figure 8-5. Watchdog Reset During Operation



8.3 Watchdog Timer

The Watchdog Timer is clocked from an on-chip oscillator, which runs at 128 kHz. See Figure 8-6. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 8-2 on page 32. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a device 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 ATtiny4/5/9/10 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to Table 8-3 on page 33.

Figure 8-6. Watchdog Timer



The Wathdog 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 fuse WDTON as shown in Table 8-1 on page 31. See "Procedure for Changing the Watchdog Timer Configuration" on page 31 for details.

Table 8-1. 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	Protected change sequence	No limitations
Programmed	2	Enabled	Always enabled	Protected change sequence

8.3.1 Procedure for Changing the Watchdog Timer Configuration

The sequence for changing configuration differs between the two safety levels, as follows:

8.3.1.1 Safety Level 1

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

1. Write the signature for change enable of protected I/O registers to register CCP
2. Within four instruction cycles, in the same operation, write WDE and WDP bits

8.3.1.2 Safety Level 2

In this mode, the Watchdog Timer is always enabled, and the WDE bit will always read as one. A protected change is needed when changing the Watchdog Time-out period. To change the Watchdog Time-out, the following procedure must be followed:

1. Write the signature for change enable of protected I/O registers to register CCP
2. Within four instruction cycles, write the WDP bit. The value written to WDE is irrelevant

8.3.2 Code Examples

The following code example shows how to turn 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

```
WDT_off:
  wdr
  ; Clear WDRF in RSTFLR
  in   r16, RSTFLR
  andi  r16, ~(1<<WDRF)
  out  RSTFLR, r16
  ; Write signature for change enable of protected I/O register
  ldi  r16, 0xD8
  out  CCP, r16
  ; Within four instruction cycles, turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCSR, r16
  ret
```

Note: See “Code Examples” on page 5.

8.4 Register Description

8.4.1 WDTCSR – Watchdog Timer Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x31	WDIF	WDIE	WDP3	–	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

- **Bit 7 – WDIF: Watchdog Timer 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 WDIE is set, the Watchdog Time-out Interrupt is requested.

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

When this bit is written to one, the Watchdog interrupt request is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is requested if time-out in the Watchdog Timer occurs.

If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode). This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety-function of the Watchdog System Reset mode. If the interrupt is not executed before the next time-out, a System Reset will be applied.

Table 8-2. Watchdog Timer Configuration

WDTON[1]	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	x	x	System Reset Mode	Reset

Note: 1. WDTON configuration bit set to “0“ means programmed and “1“ means unprogrammed.

- **Bit 4 – Res: Reserved Bit**

This bit is reserved and will always read zero.

- **Bit 3 – WDE: Watchdog System Reset Enable**

WDE is overridden by WDRF in MCUSR. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared first. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

- **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 running. The different prescaling values and their corresponding time-out periods are shown in Table 8-3 on page 33.

Table 8-3. 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	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s
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		

8.4.2 VLMCSR – V_{CC} Level Monitoring Control and Status register

Bit	7	6	5	4	3	2	1	0	
0x34	VLMF	VLMIE	–	–	–	VLM2	VLM1	VLM0	VLMCSR
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – VLMF: VLM Flag**

This bit is set by the VLM circuit to indicate that a voltage level condition has been triggered (see Table 8-4). The bit is cleared when the trigger level selection is set to “Disabled”, or when voltage at V_{CC} rises above the selected trigger level.

- **Bit 6 – VLMIE: VLM Interrupt Enable**

When this bit is set the VLM interrupt is enabled. A VLM interrupt is generated every time the VLMF flag is set.

- **Bits 5:3 – Res: Reserved Bits**

These bits are reserved. For ensuring compatibility with future devices, these bits must be written to zero, when the register is written.

- **Bits 2:0 – VLM2:0: Trigger Level of Voltage Level Monitor**

These bits set the trigger level for the voltage level monitor, as described in Table 8-4 below.

Table 8-4. Setting the Trigger Level of Voltage Level Monitor.

VLM2:0	Label	Description
000	VLM0	Voltage Level Monitor disabled
001	VLM1L	Triggering generates a regular Power-On Reset (POR). The VLM flag is not set
010	VLM1H	
011	VLM2	Triggering sets the VLM Flag (VLMF) and generates a VLM interrupt, if enabled
100	VLM3	
101	Not allowed	
110		
111		

For VLM voltage levels, see TBD, TBD and TBD.

8.4.3 RSTFLR – Reset Flag Register

The Reset Flag Register provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
0x3B	–	–	–	–	WDRF	–	EXTRF	PORF	RSTFLR
Read/Write	R	R	R	R	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	X	0	X	X	

- **Bits 7:4, 2– Res: Reserved Bits**

These bits are reserved bits in ATtiny4/5/9/10 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 logic 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 logic 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. Interrupts

This section describes the specifics of the interrupt handling in ATtiny4/5/9/10. For a general explanation of the AVR interrupt handling, see “Reset and Interrupt Handling” on page 10.

9.1 Interrupt Vectors

Interrupt vectors of ATtiny4/5/9/10 are described in Table 9-1 below.

Table 9-1. Reset and Interrupt Vectors

Vector No.	Program Address	Label	Interrupt Source
1	0x0000	RESET	External Pin, Power-on Reset, VLM Reset, Watchdog Reset
2	0x0001	INT0	External Interrupt Request 0
3	0x0002	PCINT0	Pin Change Interrupt Request 0
4	0x0003	TIM0_CAPT	Timer/Counter0 Input Capture
5	0x0004	TIM0_OVF	Timer/Counter0 Overflow
6	0x0005	TIM0_COMPA	Timer/Counter0 Compare Match A
7	0x0006	TIM0_COMPB	Timer/Counter0 Compare Match B
8	0x0007	ANA_COMP	Analog Comparator
9	0x0008	WDT	Watchdog Time-out
10	0x0009	VLM	V_{CC} Voltage Level Monitor
11	0x000A	ADC	ADC Conversion Complete (1)

Note: 1. The ADC is available in ATtiny5/10, only.

In case the program never enables an interrupt source, the Interrupt Vectors will not be used and, consequently, regular program code can be placed at these locations.

The most typical and general setup for interrupt vector addresses in ATtiny4/5/9/10 is shown in the program example below.

```
Address  Labels Code                 Comments
0x0000          rjmp   RESET          ; Reset Handler
0x0001          rjmp   INT0           ; IRQ0 Handler
0x0002          rjmp   PCINT0         ; PCINT0 Handler
0x0003          rjmp   TIM0_CAPT      ; Timer0 Capture Handler
0x0004          rjmp   TIM0_OVF       ; Timer0 Overflow Handler
0x0005          rjmp   TIM0_COMPA     ; Timer0 Compare A Handler
0x0006          rjmp   TIM0_COMPB     ; Timer0 Compare B Handler
0x0007          rjmp   ANA_COMP       ; Analog Comparator Handler
0x0008          rjmp   WDT            ; Watchdog Interrupt Handler
0x0009          rjmp   VLM            ; Voltage Level Monitor Handler
0x000A          rjmp   ADC            ; ADC Conversion Handler

<continues>
```

```
<continued>

0x000B  RESET: ldi    r16, high(RAMEND); Main program start
0x000C         out    SPH,r16          ; Set Stack Pointer
0x000D         ldi    r16, low(RAMEND) ; to top of RAM
0x000E         out    SPL,r16
0x000F         sei                     ; Enable interrupts
0x0010         <instr>
...            ...
```

9.2 External Interrupts

External Interrupts are triggered by the INT0 pin or any of the PCINT3..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT3..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 PCINT3..0 pin toggles. The PCMSK Register controls which pins contribute to the pin change interrupts. Pin change interrupts on PCINT3..0 are detected asynchronously, which means that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 interrupt can be triggered by a falling or rising edge or a low level. This is set up as shown in “EICRA – External Interrupt Control Register A” on page 37. When the INT0 interrupt is enabled and 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, as described in “Clock System” on page 17.

9.2.1 Low Level Interrupt

A low level interrupt on INT0 is detected asynchronously. This means that the interrupt source can be used for waking the part also from sleep modes other than Idle (the I/O clock is halted in all sleep modes except Idle).

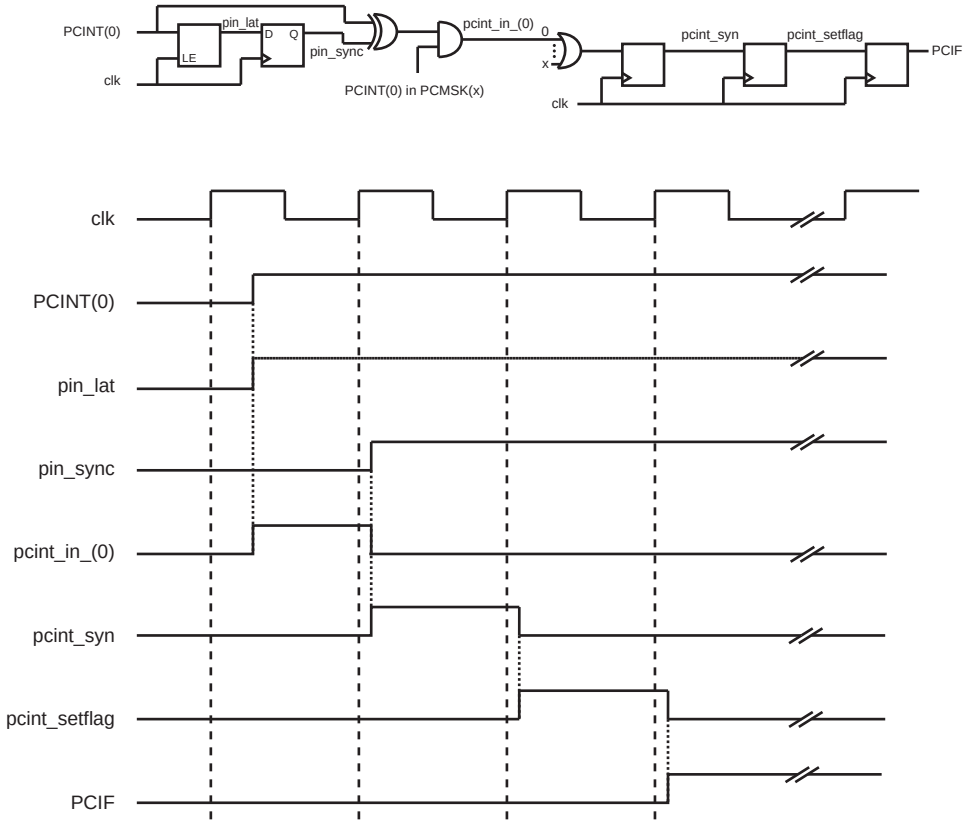
Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined as described in “Clock System” on page 17.

If the low level on the interrupt pin is removed before the device has woken up then program execution will not be diverted to the interrupt service routine but continue from the instruction following the SLEEP command.

9.2.2 Pin Change Interrupt Timing

A timing example of a pin change interrupt is shown in Figure 9-1.

Figure 9-1. Timing of pin change interrupts



9.3 Register Description

9.3.1 EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
0x15	–	–	–	–	–	–	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **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 9-2. 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 9-2. 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.

9.3.2 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x13	–	–	–	–	–	–	–	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 0 – 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 Control bits (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.

9.3.3 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x14	–	–	–	–	–	–	–	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved and will always read zero.

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

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK 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 constantly zero when INT0 is configured as a level interrupt.

9.3.4 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
0x12	–	–	–	–	–	–	–	PCIE0	PCICR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved and will always read zero.

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

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT3..0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT3..0 pins are enabled individually by the PCMSK Register.

9.3.5 PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x11	–	–	–	–	–	–	–	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved and will always read zero.

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

When a logic change on any PCINT3..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR 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.

9.3.6 PCMSK – Pin Change Mask Register

Bit	7	6	5	4	3	2	1	0	
0x10	–	–	–	–	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK
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 and will always read zero.

- **Bits 3:0 – PCINT3..0: Pin Change Enable Mask 3..0**

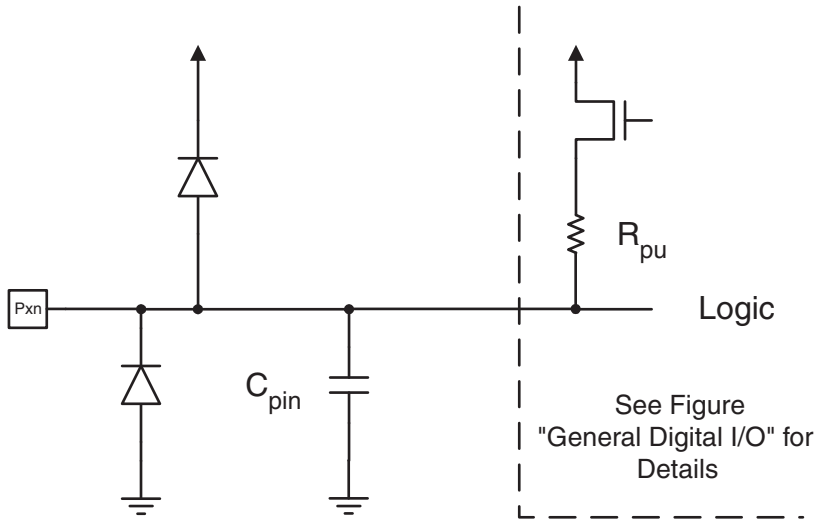
Each PCINT3..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT3..0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT3..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

10. I/O Ports

10.1 Overview

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors. Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 10-1 on page 40. See "Electrical Characteristics" on page 116 for a complete list of parameters.

Figure 10-1. 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 "Register Description" on page 50.

Four I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, Pull-up Enable Register – PUEx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register, the Data Direction Register, and the Pull-up Enable Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register.

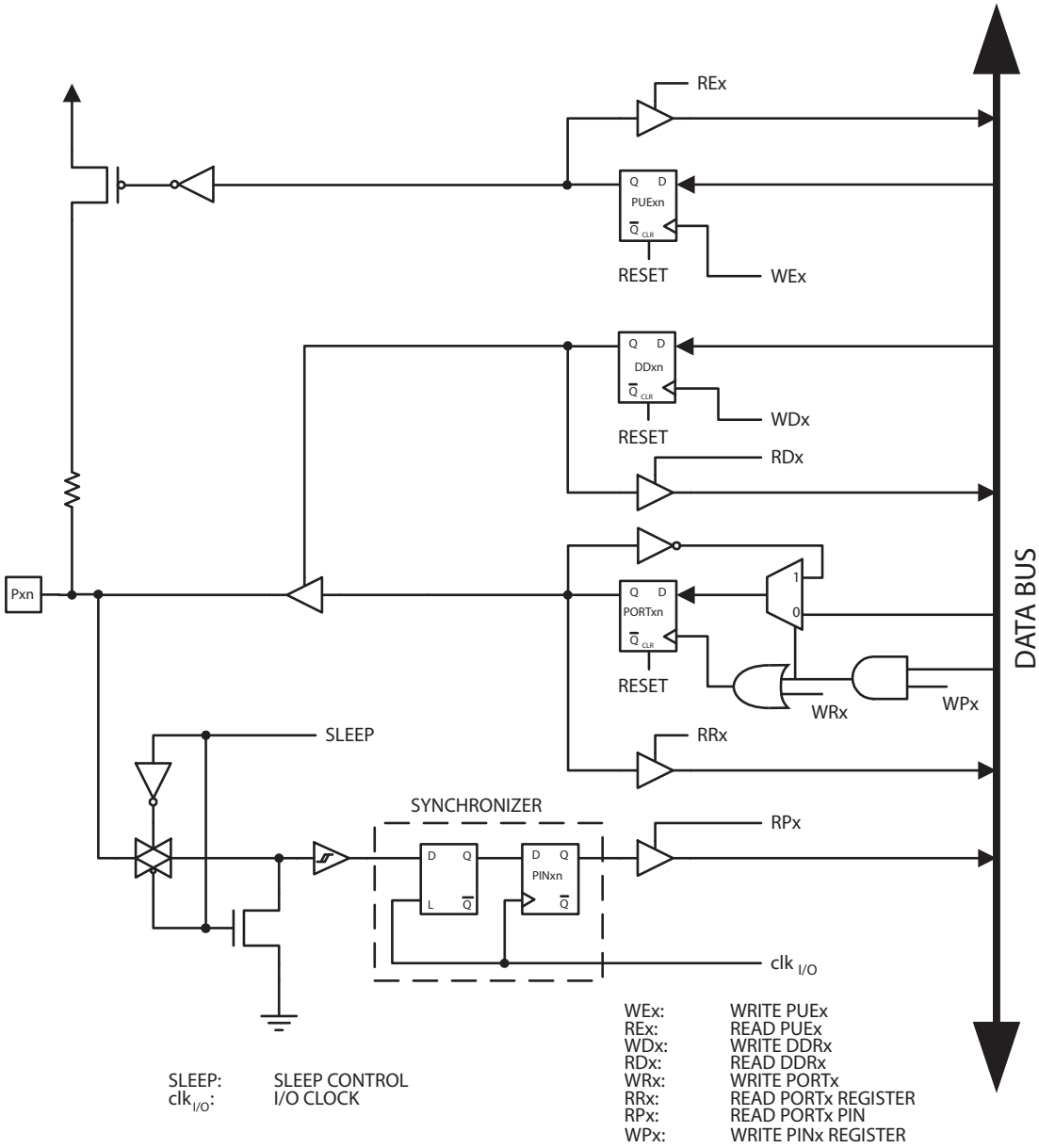
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 41. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 45. 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.

10.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 10-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 10-2. General Digital I/O[1]



Note: 1. WEx, WRx, WPx, WDx, REx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, and SLEEP are common to all ports.

10.2.1 Configuring the Pin

Each port pin consists of four register bits: DDxn, PORTxn, PUExn, and PINxn. As shown in "Register Description" on page 50, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, the PUExn bits at the PUEx 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 logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

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

The pull-up resistor is activated, if the PUExn is written logic one. To switch the pull-up resistor off, PUExn has to be written logic zero.

Table 10-1 summarizes the control signals for the pin value.

Table 10-1. Port Pin Configurations

DDxn	PORTxn	PUExn	I/O	Pull-up	Comment
0	X	0	Input	No	Tri-state (hi-Z)
0	X	1	Input	Yes	Sources current if pulled low externally
1	0	0	Output	No	Output low (sink)
1	0	1	Output	Yes	NOT RECOMMENDED. Output low (sink) and internal pull-up active. Sources current through the internal pull-up resistor and consumes power constantly
1	1	0	Output	No	Output high (source)
1	1	1	Output	Yes	Output high (source) and internal pull-up active

Port pins are tri-stated when a reset condition becomes active, even when no clocks are running.

10.2.2 Toggling the Pin

Writing a logic 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.

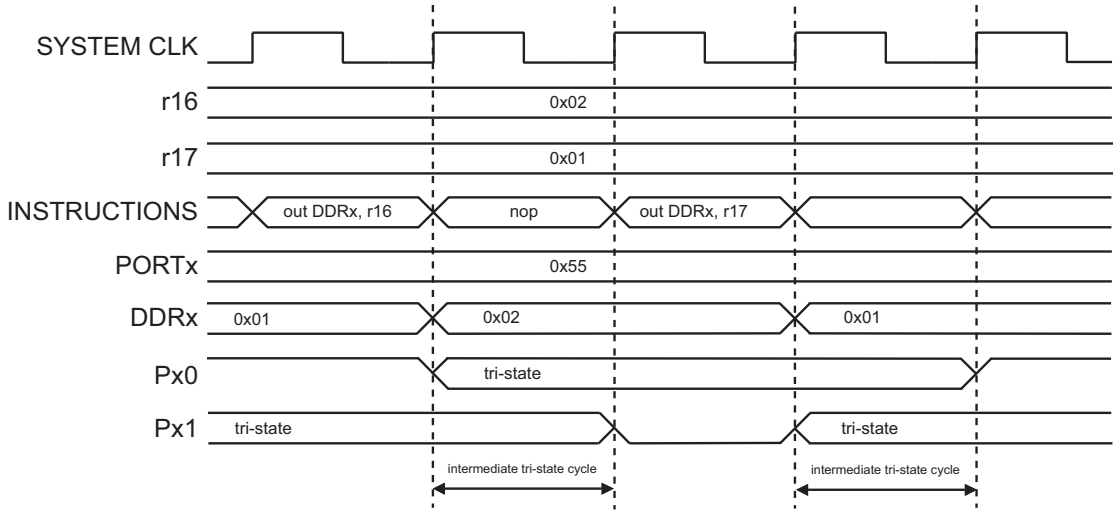
10.2.3 Break-Before-Make Switching

In Break-Before-Make mode, switching the DDRxn bit from input to output introduces an immediate tri-state period lasting one system clock cycle, as indicated in Figure 10-3. For example, if the system clock is 4 MHz and the DDRxn is written to make an output, an immediate tri-state period of 250 ns is introduced before the value of PORTxn is seen on the port pin.

To avoid glitches it is recommended that the maximum DDRxn toggle frequency is two system clock cycles. The Break-Before-Make mode applies to the entire port and it is activated by the BBMx bit. For more details, see “PORTCR – Port Control Register” on page 50.

When switching the DDRxn bit from output to input no immediate tri-state period is introduced.

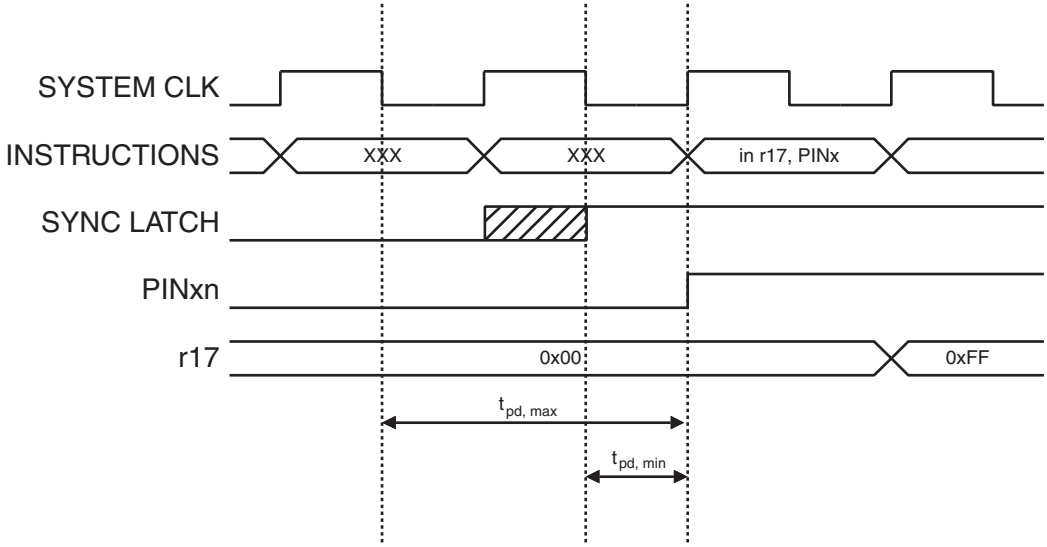
Figure 10-3. Switching Between Input and Output in Break-Before-Make-Mode



10.2.4 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 10-2 on page 41, 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 10-4 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.

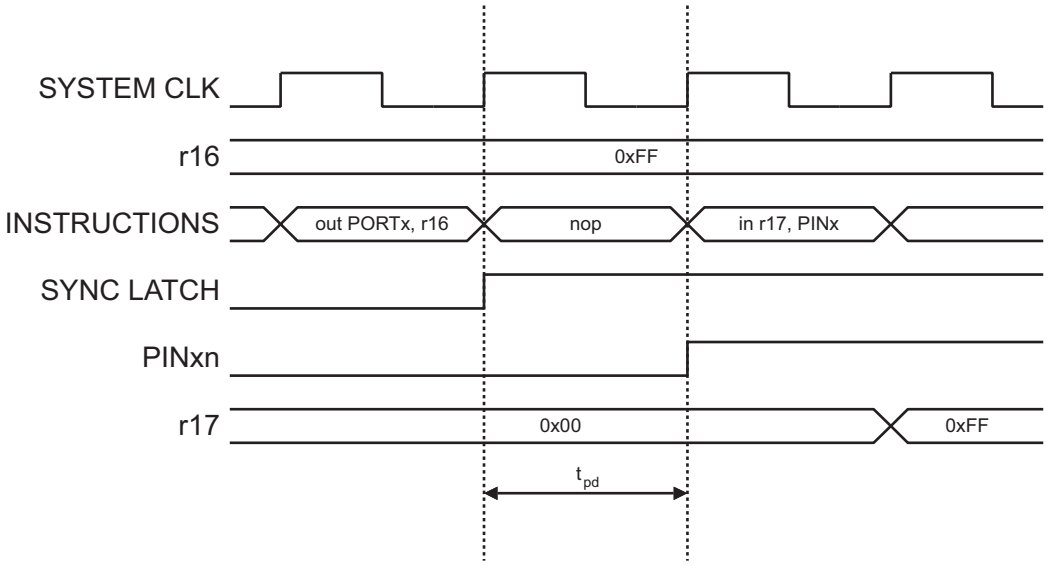
Figure 10-4. Synchronization when Reading an Externally Applied Pin value



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 tpd,max and tpd,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 10-5 on page 44. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is one system clock period.

Figure 10-5. Synchronization when Reading a Software Assigned Pin Value



10.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 10-2 on page 41, 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 and Standby modes 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 “Alternate Port Functions” on page 45.

If a logic high level (“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.

10.2.6 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 pulldown. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

10.2.7 Program Example

The following code example shows how to set port B pin 0 high, pin 1 low, and define the port pins from 2 to 3 as input with a pull-up assigned to port pin 2. 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

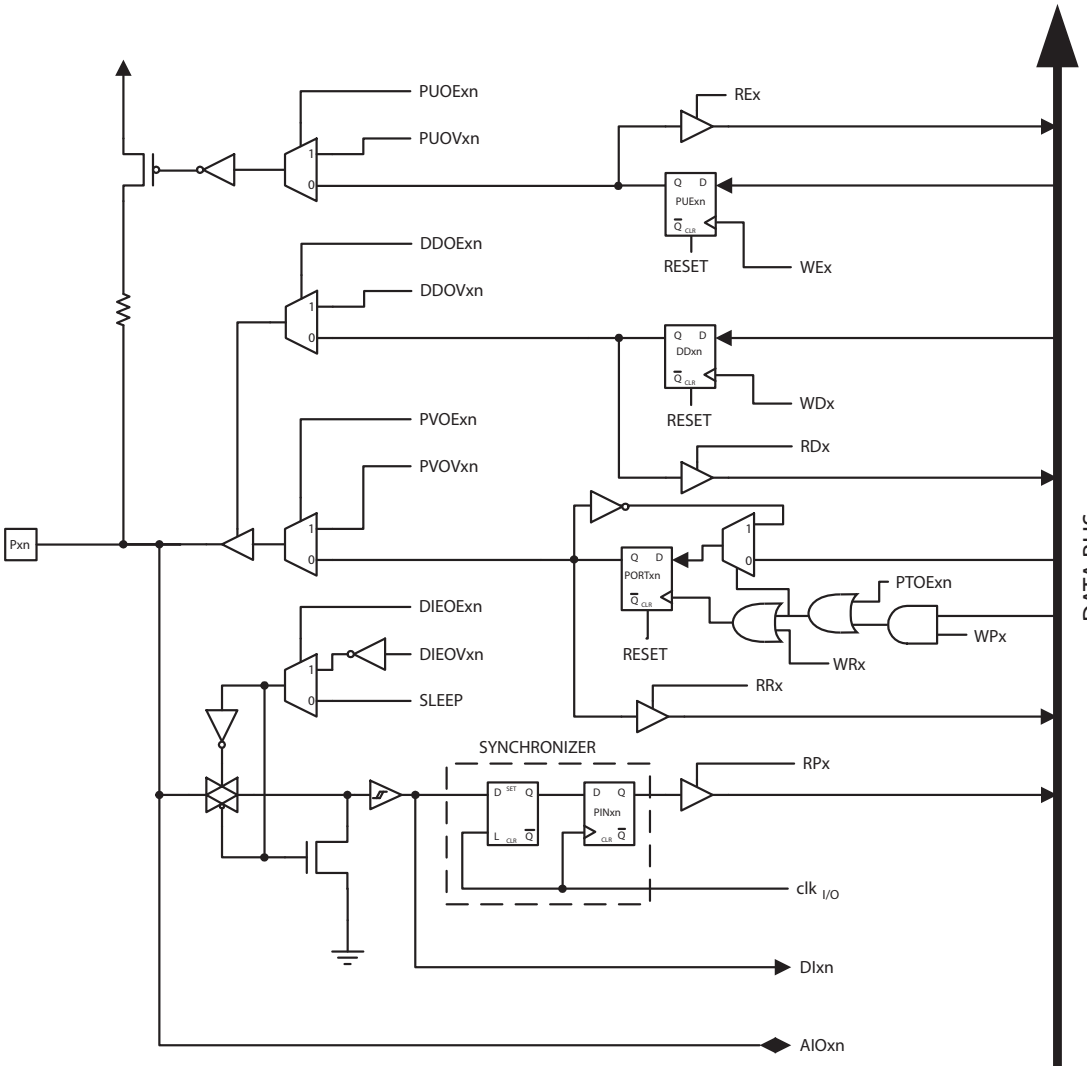
```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r16,(1<<PUEB2)
ldi  r17,(1<<PB0)
ldi  r18,(1<<DDB1)|(1<<DDB0)
out  PUEB,r16
out  PORTB,r17
out  DDRB,r18
; Insert nop for synchronization
nop
; Read port pins
in   r16,PINB
...
```

Note: See “Code Examples” on page 5.

10.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. In Figure 10-6 below is shown how the port pin control signals from the simplified Figure 10-2 on page 41 can be overridden by alternate functions.

Figure 10-6. Alternate Port Functions[1]



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

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

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

The illustration in the figure above serves as a generic description applicable to all port pins in the AVR microcontroller family. Some overriding signals may not be present in all port pins.

Table 10-2 on page 47 summarizes the function of the overriding signals. The pin and port indexes from Figure 10-6 on page 46 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 10-2. 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 PUExn = 0b1.
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 PUExn Register bit.
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.

10.3.1 Alternate Functions of Port B

The Port B pins with alternate function are shown in Table 10-3 on page 48.

Table 10-3. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB0	ADC0: ADC Input Channel 0 AIN0: Analog Comparator, Positive Input OC0A: Timer/Counter0 Compare Match A Output PCINT0: Pin Change Interrupt 0, Source 0 TPIDATA:Serial Programming Data
PB1	ADC1: ADC Input Channel 1 AIN1: Analog Comparator, Negative Input CLKI: External Clock ICP0: Timer/Counter0 Input Capture Input OC0B: Timer/Counter0 Compare Match B Output PCINT1:Pin Change Interrupt 0, Source 1 TPICLK: Serial Programming Clock
PB2	ADC2: ADC Input Channel 2 CLKO: System Clock Output INT0: External Interrupt 0 Source PCINT2: Pin Change Interrupt 0, Source 2 T0: Timer/Counter0 Clock Source
PB3	ADC3: ADC Input Channel 3 PCINT3: Pin Change Interrupt 0, Source 3 $\overline{\text{RESET}}$: Reset Pin

- **Port B, Bit 0 – ADC0/AIN0/OC0A/PCINT0/TPIDATA**
  - ADC0: Analog to Digital Converter, Channel 0 (ATtiny5/10, only)
  - 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.
  - OC0A, Output Compare Match output: The PB0 pin can serve as an external output for the Timer/Counter0 Compare Match A. The pin has to be configured as an output (DDB0 set (one)) to serve this function. This is also the output pin for the PWM mode timer function.
  - PCINT0: Pin Change Interrupt source 0. The PB0 pin can serve as an external interrupt source for pin change interrupt 0.
  - TPIDATA: Serial Programming Data.

- **Port B, Bit 1 – ADC1/AIN1/CLKI/ICP0/OC0B/PCINT1/TPICLK**
  - ADC1: Analog to Digital Converter, Channel 1 (ATtiny5/10, only)
  - 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.
  - CLKI: External Clock.
  - ICP0: Input Capture Pin. The PB1 pin can act as an Input Capture pin for Timer/Counter0.

- OC0B: Output Compare Match output: The PB1 pin can serve as an external output for the Timer/Counter0 Compare Match B. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
- PCINT1: Pin Change Interrupt source 1. The PB1 pin can serve as an external interrupt source for pin change interrupt 0.
- TPICLK: Serial Programming Clock.

- **Port B, Bit 2 – ADC2/CLKO/INT0/PCINT2/T0**
  - ADC2: Analog to Digital Converter, Channel 2 (ATtiny5/10, only)
  - CLKO: System Clock Output. The system clock can be output on pin PB2. The system clock will be output if CKOUT bit is programmed, regardless of the PORTB2 and DDB2 settings.
  - INT0: External Interrupt Request 0
  - PCINT2: Pin Change Interrupt source 2. The PB2 pin can serve as an external interrupt source for pin change interrupt 0.
  - T0: Timer/Counter0 counter source.

- **Port B, Bit 3 – ADC3/PCINT3/$\overline{\text{RESET}}$**
  - ADC3: Analog to Digital Converter, Channel 3 (ATtiny5/10, only)
  - PCINT3: Pin Change Interrupt source 3. The PB3 pin can serve as an external interrupt source for pin change interrupt 0.
  - $\overline{\text{RESET}}$:

Table 10-4 and Table 10-5 on page 50 relate the alternate functions of Port B to the overriding signals shown in Figure 10-6 on page 46.

Table 10-4. Overriding Signals for Alternate Functions in PB3..PB2

Signal Name	PB3/ADC3/$\overline{\text{RESET}}$/PCINT3	PB2/ADC2/INT0/T0/CLKO/PCINT2
PUOE	$\overline{\text{RSTDISBL}}$[1]	CKOUT[2]
PUOV	1	0
DDOE	$\overline{\text{RSTDISBL}}$[1]	CKOUT[2]
DDOV	0	1
PVOE	0	CKOUT[2]
PVOV	0	(system clock)
PTOE	0	0
DIEOE	$\overline{\text{RSTDISBL}}$[1] + (PCINT3 • PCIE0) + ADC3D	(PCINT2 • PCIE0) + ADC2D + INT0
DIEOV	RSTDISBL • PCINT3 • PCIE0	(PCINT2 • PCIE0) + INT0
DI	PCINT3 Input	INT0/T0/PCINT2 Input
AIO	ADC3 Input	ADC2 Input

Notes: 1. RSTDISBL is 1 when the configuration bit is “0” (Programmed).
2. CKOUT is 1 when the configuration bit is “0” (Programmed).

Table 10-5. Overriding Signals for Alternate Functions in PB1..PB0

Signal Name	PB1/ADC1/AIN1/OC0B/CLKI/ICP0/PCINT1	PB0/ADC0/AIN0/OC0A/PCINT0
PUOE	EXT_CLOCK(1)	0
PUOV	0	0
DDOE	EXT_CLOCK(1)	0
DDOV	0	0
PVOE	EXT_CLOCK(1) + OC0B Enable	OC0A Enable
PVOV	$\overline{\text{EXT_CLOCK}}$(1) • OC0B	OC0A
PTOE	0	0
DIEOE	EXT_CLOCK(1) + (PCINT1 • PCIE0) + ADC1D	(PCINT0 • PCIE0) + ADC0D
DIEOV	(EXT_CLOCK(1) • $\overline{\text{PWR_DOWN}}$) + ($\overline{\text{EXT_CLOCK}}$(1) • PCINT1 • PCIE0)	PCINT0 • PCIE0
DI	CLOCK/ICP0/PCINT1 Input	PCINT0 Input
AIO	ADC1/Analog Comparator Negative Input	ADC0/Analog Comparator Positive Input

Notes: 1. EXT_CLOCK is 1 when external clock is selected as main clock.

10.4 Register Description

10.4.1 PORTCR – Port Control Register

Bit	7	6	5	4	3	2	1	0	
0x03	–	–	–	–	–	–	BBMB	–	PORTCR
Read/Write	R	R	R	R	R	R	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2, 0 – Reserved**

These bits are reserved and will always read zero.

- **Bit 1 – BBMB: Break-Before-Make Mode Enable**

When this bit is set the Break-Before-Make mode is activated for the entire Port B. The intermediate tri-state cycle is then inserted when writing DDRxn to make an output. For further information, see “Break-Before-Make Switching” on page 42.

10.4.2 PUEB – Port B Pull-up Enable Control Register

Bit	7	6	5	4	3	2	1	0	
0x03	–	–	–	–	PUEB3	PUEB2	PUEB1	PUEB0	PUEB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.4.3 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x02	–	–	–	–	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	

10.4.4 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01	–	–	–	–	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	

10.4.5 PINB – Port B Input Pins

Bit	7	6	5	4	3	2	1	0	
0x00	–	–	–	–	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	N/A	N/A	N/A	N/A	

11. 16-bit Timer/Counter0

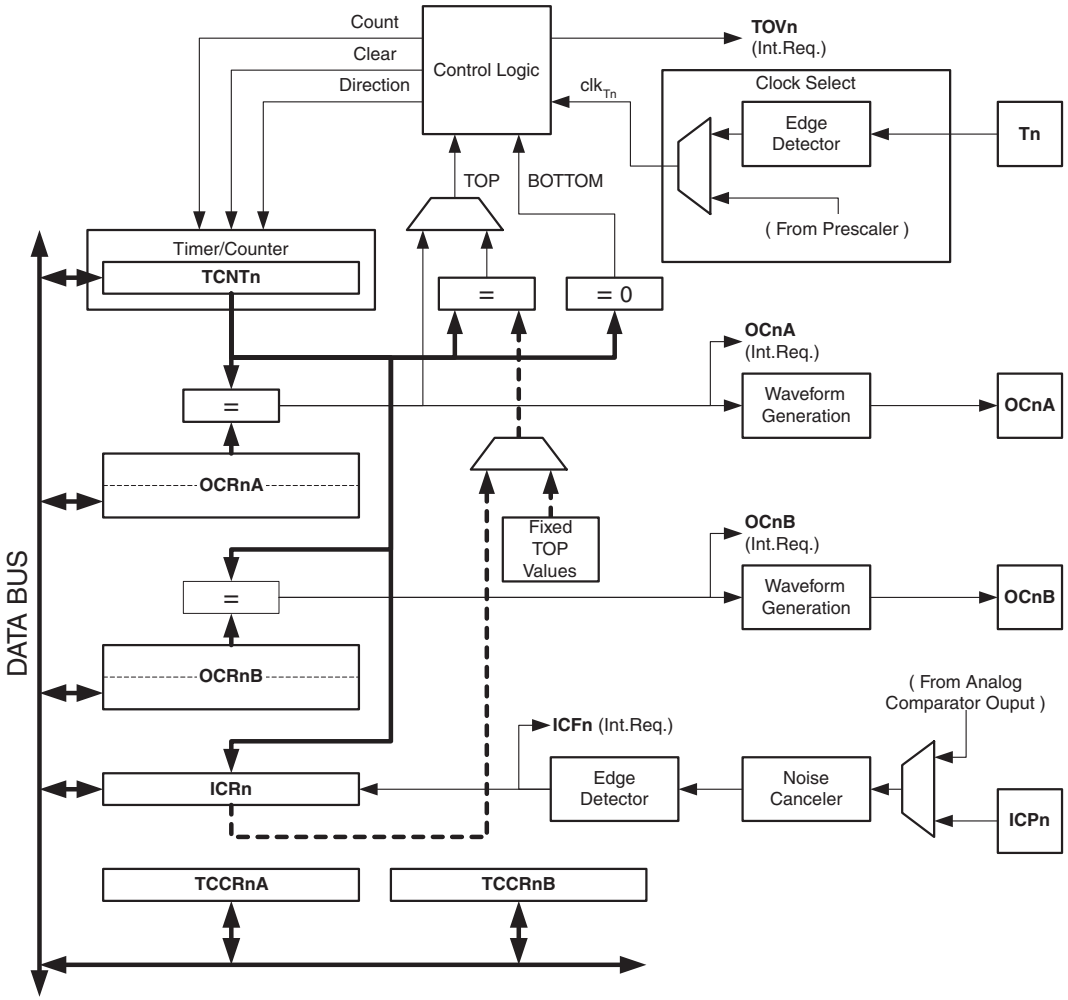
11.1 Features

- **True 16-bit Design, Including 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 (TOV0, OCF0A, OCF0B, and ICF0)**

11.2 Overview

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

Figure 11-1. 16-bit Timer/Counter Block Diagram



A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 11-1 on page 52. For actual placement of I/O pins, refer to "Pinout of ATtiny4/5/9/10" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "Register Description" on page 73.

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., TCNT0 for accessing Timer/Counter0 counter value and so on.

11.2.1 Registers

The Timer/Counter (TCNT0), Output Compare Registers (OCR0A/B), and Input Capture Register (ICR0) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 71. The Timer/Counter Control Registers (TCCR0A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (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 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 (clkT0).

The double buffered Output Compare Registers (OCR0A/B) are compared with the Timer/Counter value at all time. 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 (OC0A/B). See "Output Compare Units" on page 59. The compare match event will also set the Compare Match Flag (OCF0A/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 (ICP0) or on the Analog Comparator pins (See "Analog Comparator" on page 81). The Input Capture unit includes a digital filtering unit (Noise Canceler) 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 OCR0A Register, the ICR0 Register, or by a set of fixed values. When using OCR0A as TOP value in a PWM mode, the OCR0A Register can not 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 in run time. If a fixed TOP value is required, the ICR0 Register can be used as an alternative, freeing the OCR0A to be used as PWM output.

11.2.2 Definitions

The following definitions are used extensively throughout the section:

Table 11-1. Definitions

Constant	Description
BOTTOM	The counter reaches 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 depends on the mode of operation

11.3 Clock Sources

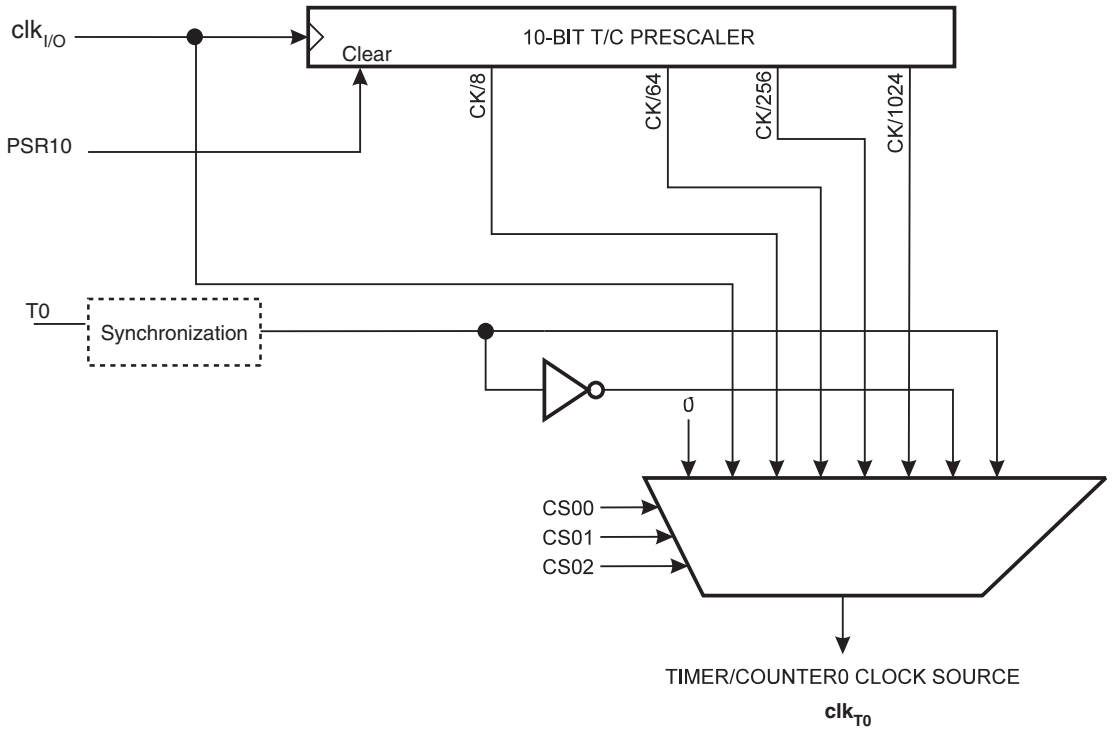
The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter control Register B (TCCR0B). For details on clock sources and prescaler, see section “Prescaler”.

11.3.1 Prescaler

The Timer/Counter can be clocked directly by the system clock (by setting the CS2:0 = 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.

See Figure 11-2 for an illustration of the prescaler unit.

Figure 11-2. Prescaler for Timer/Counter0



Note: 1. The synchronization logic on the input pins (T0) is shown in Figure 11-3 on page 55.

The prescaled clock has a frequency of $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$. See Table 11-6 on page 76 for details.

11.3.1.1 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/CounterCounter, and it is shared by the Timer/Counter Tn. Since 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 (CS2:0 = 2, 3, 4, or 5). 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).

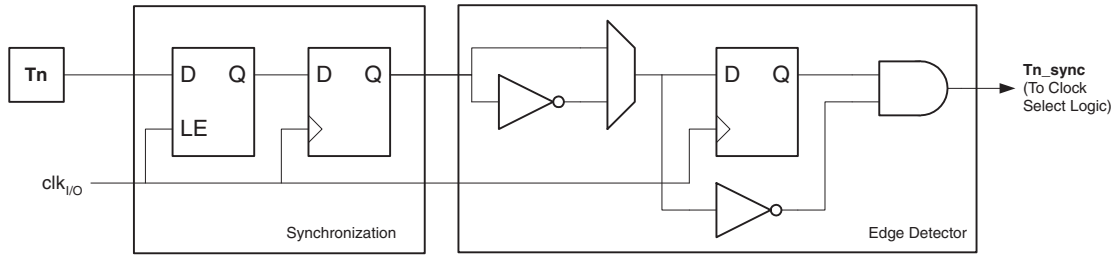
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution.

11.3.2 External Clock Source

An external clock source applied to the T0 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 11-3 on page 55 shows a functional equivalent block diagram of the T0 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 (CS2:0 = 7) or negative (CS2:0 = 6) edge it detects.

Figure 11-3. T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

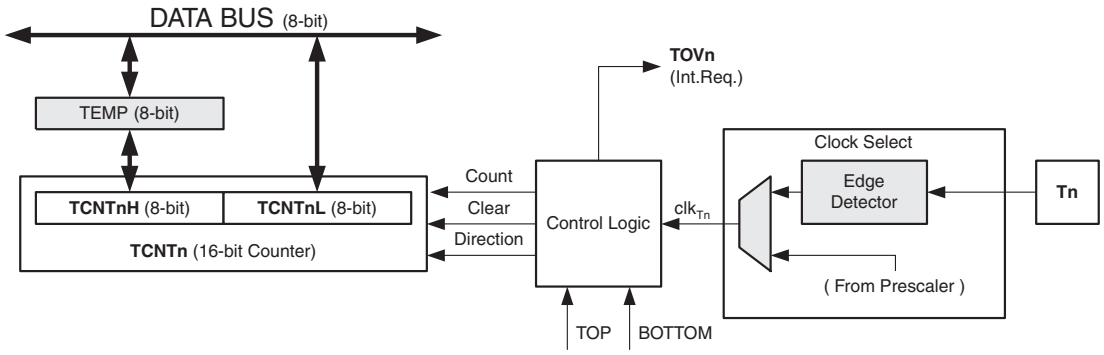
Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, 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 capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk_I/O}/2.5$.

An external clock source can not be prescaled.

11.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 11-4 on page 56 shows a block diagram of the counter and its surroundings.

Figure 11-4. 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_{T0}	Timer/Counter clock.
TOP	Signalize that TCNT0 has reached maximum value.
BOTTOM	Signalize that TCNT0 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT0H) containing the upper eight bits of the counter, and Counter Low (TCNT0L) containing the lower eight bits. The TCNT0H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT0H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT0H value when the TCNT0L is read, and TCNT0H is updated with the temporary register value when TCNT0L 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 TCNT0 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 (clk_{T0}). The 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, independent 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 Waveform Generation mode bits (WGM03:0) located in the Timer/Counter Control Registers A and B (TCCR0A and TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0x. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 62.

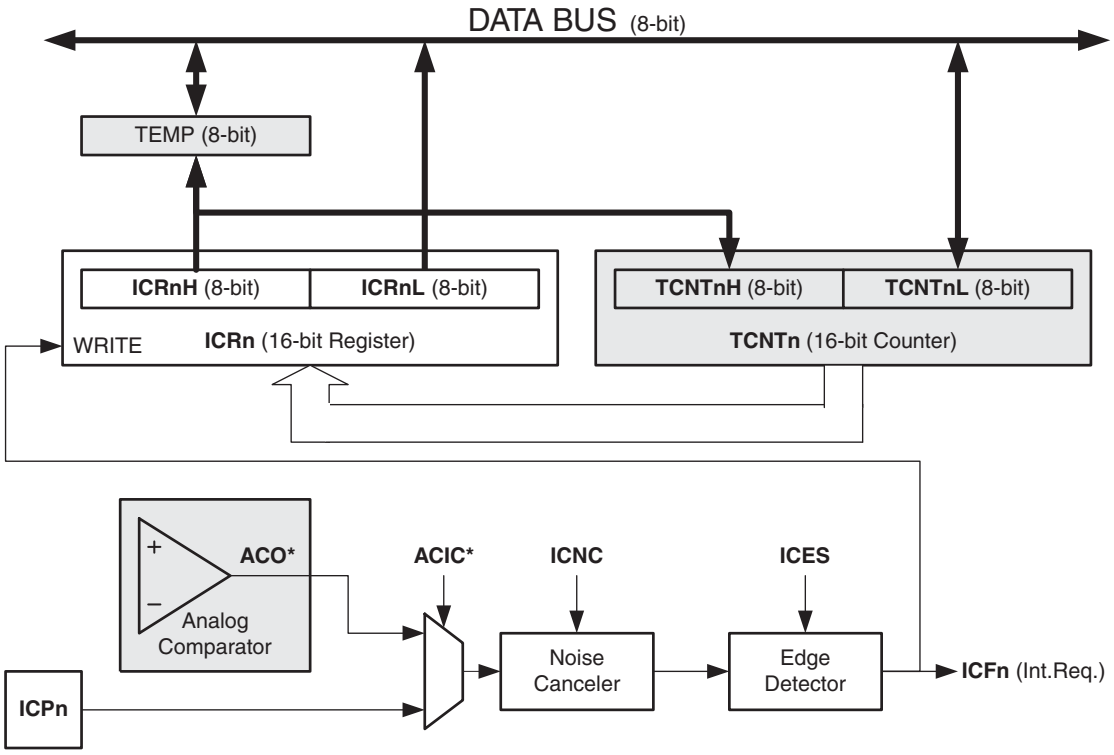
The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM03:0 bits. TOV0 can be used for generating a CPU interrupt.

11.5 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 ICP0 pin. 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 11-5 on page 57. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The lower case “n” in register and bit names indicates the Timer/Counter number.

Figure 11-5. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the Input Capture pin (ICP0), alternatively on the Analog Comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT0) is written to the Input Capture Register (ICR0). The Input Capture Flag (ICF0) is set at the same system clock as the TCNT0 value is copied into ICR0 Register. If enabled (ICIE0 = 1), the Input Capture Flag generates an Input Capture interrupt. The ICF0 flag is automatically cleared when the interrupt is executed. Alternatively the ICF0 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 (ICR0) is done by first reading the low byte (ICR0L) and then the high byte (ICR0H). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR0H I/O location it will access the TEMP Register.

The ICR0 Register can only be written when using a Waveform Generation mode that utilizes the ICR0 Register for defining the counter’s TOP value. In these cases the Waveform Genera-

tion mode (WGM03:0) bits must be set before the TOP value can be written to the ICR0 Register. When writing the ICR0 Register the high byte must be written to the ICR0H I/O location before the low byte is written to ICR0L.

For more information on how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 71.

11.5.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the Input Capture pin (ICP0). Timer/Counter0 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 “ACSR – Analog Comparator Control and Status Register”. Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the Input Capture pin (ICP0) and the Analog Comparator output (ACO) inputs are sampled using the same technique as for the T0 pin (Figure 11-3 on page 55). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR0 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP0 pin.

11.5.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the Input Capture Noise Canceler (ICNC0) bit in Timer/Counter Control Register B (TCCR0B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR0 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

11.5.3 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 ICR0 Register before the next event occurs, the ICR0 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 ICR0 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 is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR0 Register has been read. After a change of the edge, the Input Capture Flag (ICF0) must be

cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF0 flag is not required (if an interrupt handler is used).

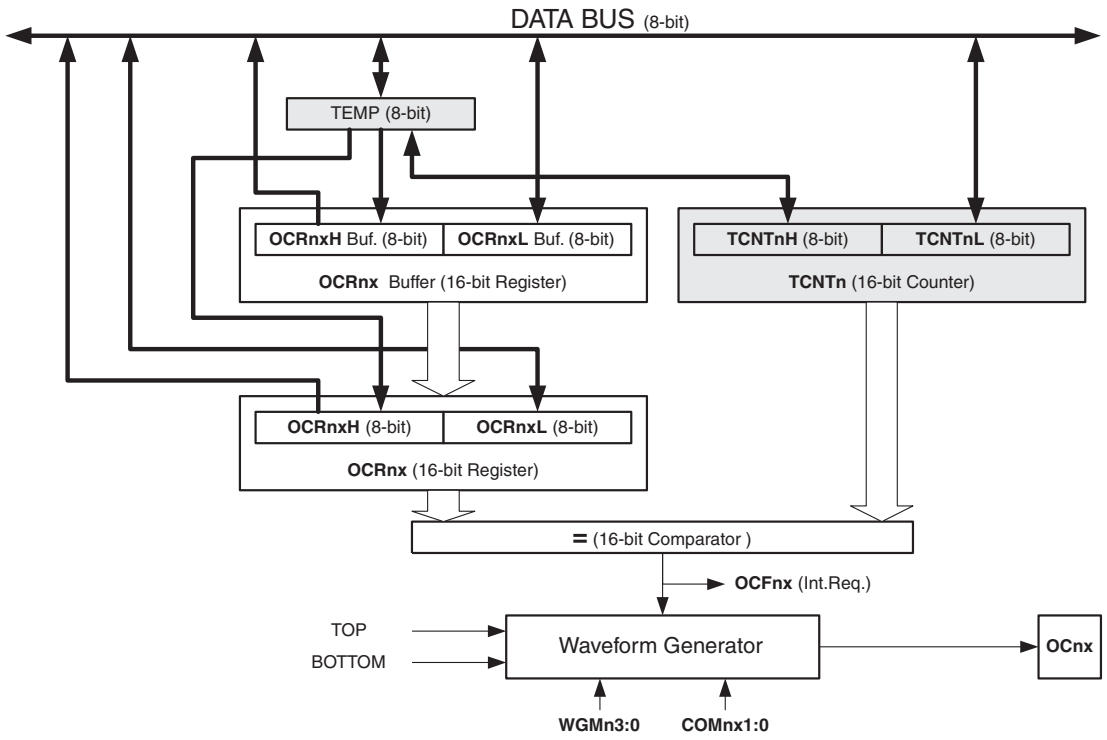
11.6 Output Compare Units

The 16-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0x). If TCNT equals OCR0x the comparator signals a match. A match will set the Output Compare Flag (OCF0x) at the next timer clock cycle. If enabled (OCIE0x = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF0x flag is automatically cleared when the interrupt is executed. Alternatively the OCF0x 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 operating mode set by the Waveform Generation mode (WGM03:0) bits and Compare Output mode (COM0x1:0) 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 (“Modes of Operation” on page 62).

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 11-6 on page 59 shows a block diagram of the Output Compare unit. The small “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B). The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

Figure 11-6. Output Compare Unit, Block Diagram



The OCR0x 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 OCR0x 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 OCR0x Register access may seem complex, but this is not 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. The content of the OCR0x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT0 and ICR0 Register). Therefore OCR0x 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 OCR0x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR0xH) 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 (OCR0xL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR0x buffer or OCR0x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 71.

11.6.1 Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a 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 COM01:0 bits settings define whether the OC0x pin is set, cleared or toggled).

11.6.2 Compare Match Blocking by TCNT0 Write

All CPU writes to the TCNT0 Register will block any compare match that occurs 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.

11.6.3 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 when changing TCNT0 when using any of the Output Compare channels, independent 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. Do not write the TCNT0 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 TCNT0 value equal to BOTTOM when the counter is downcounting.

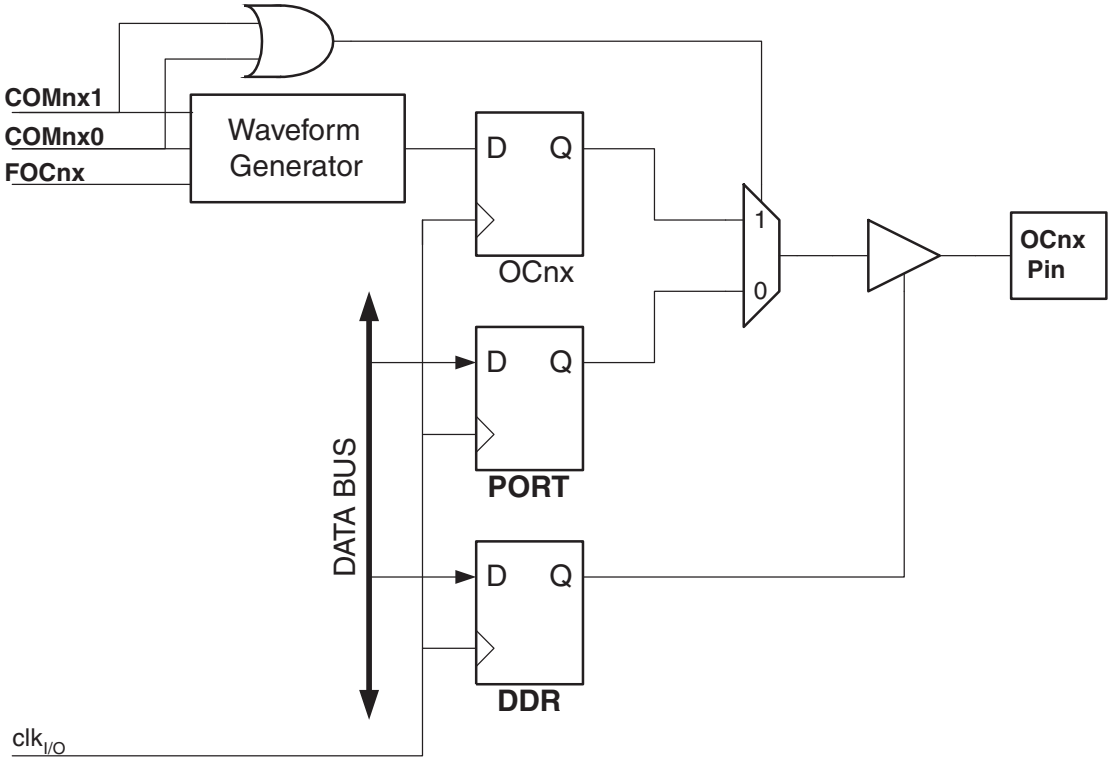
The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (0x) strobe bits in Normal mode. The OC0x Register keeps its value even when changing between Waveform Generation modes.

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

11.7 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. Secondly the COM0x1:0 bits control the OC0x pin output source. Figure 11-7 on page 61 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 occur, the OC0x Register is reset to “0”.

Figure 11-7. Compare Match Output Unit, Schematic (non-PWM Mode)



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 generally independent of the Waveform Generation mode, but there are some exceptions. See Table 11-2 on page 74, Table 11-3 on page 74 and Table 11-4 on page 74 for details.

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 “Register Description” on page 73

The COM0x1:0 bits have no effect on the Input Capture unit.

11.7.1 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 11-2 on page 74. For fast PWM mode refer to Table 11-3 on page 74, and for phase correct and phase and frequency correct PWM refer to Table 11-4 on page 74.

A change of the COM0x1:0 bits state 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.

11.8 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 (WGM03:0) 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 toggle at a compare match (“Compare Match Output Unit” on page 61)

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 69.

11.8.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM03:0 = 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 (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 flag in this case behaves like a 17th 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 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 are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

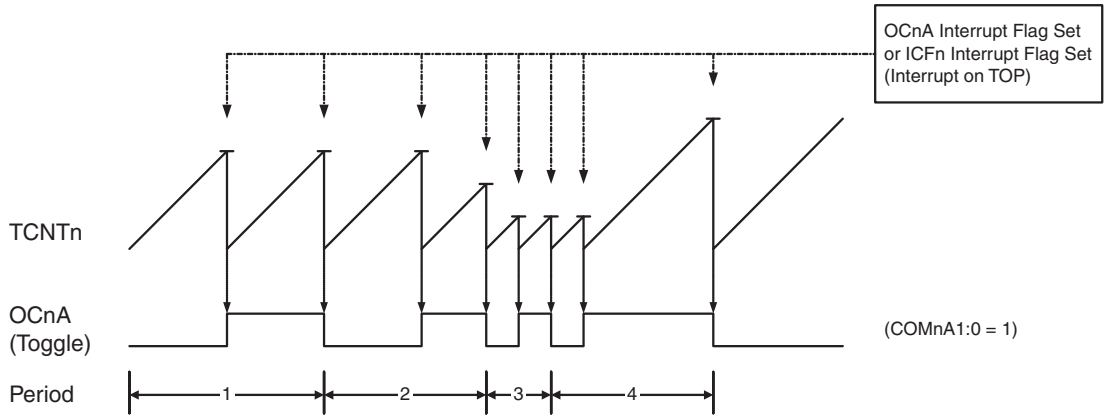
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

11.8.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM03:0 = 4 or 12), the OCR0A or ICR0 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches either the OCR0A (WGM03:0 = 4) or the ICR0 (WGM03:0 = 12). The OCR0A or ICR0 define 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 11-8 on page 63. The counter value (TCNT0) increases until a compare match occurs with either OCR0A or ICR0, and then counter (TCNT0) is cleared.

Figure 11-8. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF0A or ICF0 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 none 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 or ICR0 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 (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 OCR0A for defining TOP (WGM03:0 = 15) since the OCR0A then will be double buffered.

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 (DDR_OC0A = 1). The waveform generated will have a maximum frequency of $_{0A}$ = $f_{clk_I/O}/2$ when OCR0A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

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

As for the Normal mode of operation, the TOV0 flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

11.8.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM03:0 = 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 (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope

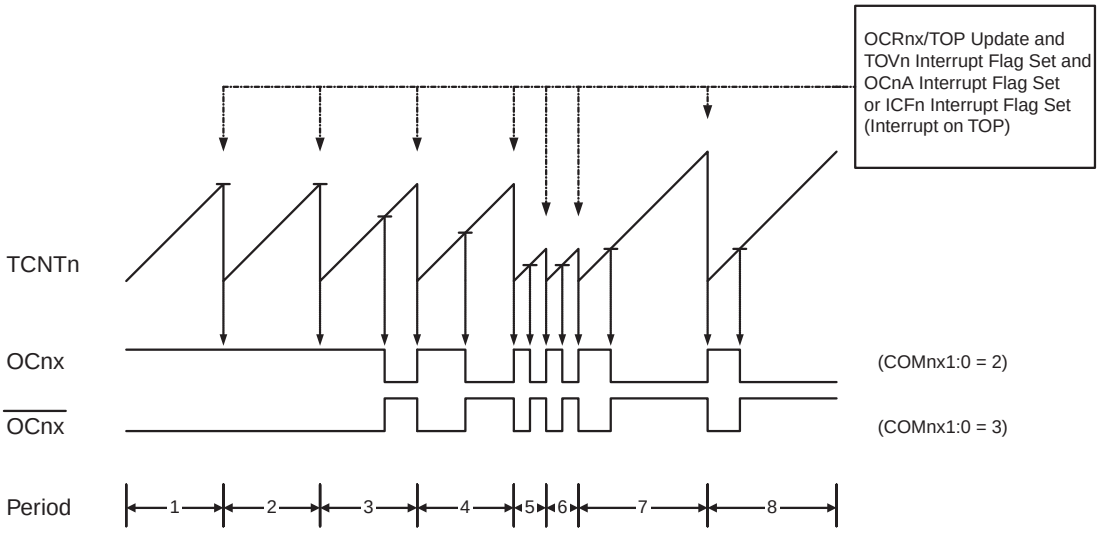
operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency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

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR0 or OCR0A. The minimum resolution allowed is 2-bit (ICR0 or OCR0A set to 0x0003), and the maximum resolution is 16-bit (ICR0 or OCR0A 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 (WGM03:0 = 5, 6, or 7), the value in ICR0 (WGM03:0 = 14), or the value in OCR0A (WGM03:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 11-9 on page 64. The figure shows fast PWM mode when OCR0A or ICR0 is used to define TOP. 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. The OC0x interrupt flag will be set when a compare match occurs.

Figure 11-9. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches TOP. In addition the OC0A or ICF0 flag is set at the same timer clock cycle as TOV0 is set when either OCR0A or ICR0 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 the TCNT0 and the OCR0x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR0x Registers are written.

The procedure for updating ICR0 differs from updating OCR0A when used for defining the TOP value. The ICR0 Register is not double buffered. This means that if ICR0 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR0 value written is lower than the current value of TCNT0. 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 OCR0A Register however, is double buffered. This feature allows the OCR0A I/O location to be written anytime. When the OCR0A I/O location is written the value written will be put into the OCR0A Buffer Register. The OCR0A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT0 matches TOP. The update is done at the same timer clock cycle as the TCNT0 is cleared and the TOV0 flag is set.

Using the ICR0 Register for defining TOP works well when using fixed TOP values. By using ICR0, the OCR0A Register is free to be used for generating a PWM output on OC0A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR0A 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 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 to three (see Table 11-3 on page 74). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC0x). 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 the counter is cleared (changes from TOP to BOTTOM).

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

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot (1 + TOP)}$$

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

The extreme values for the OCR0x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR0x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM0x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0A to toggle its logical level on each compare match (COM0A1:0 = 1). The waveform generated will have a maximum frequency of $f_{0A} = f_{clk_I/O}/2$ when OCR0A is set to zero (0x0000). 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.

11.8.4 Phase Correct PWM Mode

The phase correct Pulse Width Modulation or phase correct PWM mode (WGM03:0 = 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 (OC0x) is cleared on the compare match between TCNT0 and OCR0x while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope

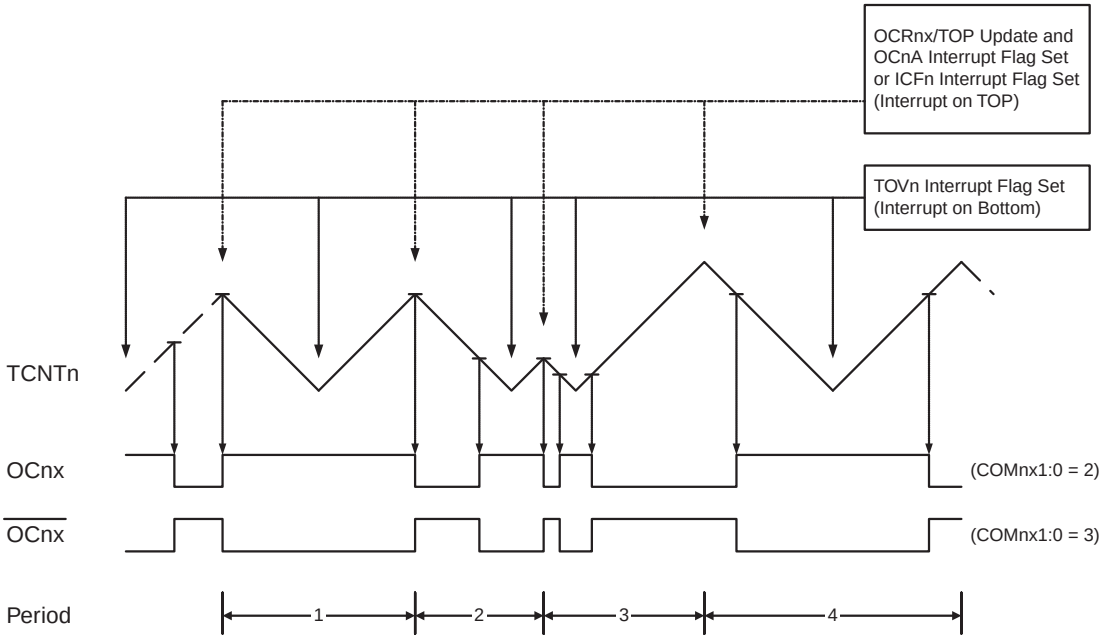
operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR0 or OCR0A. The minimum resolution allowed is 2-bit (ICR0 or OCR0A set to 0x0003), and the maximum resolution is 16-bit (ICR0 or OCR0A 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 (WGM03:0 = 1, 2, or 3), the value in ICR0 (WGM03:0 = 10), or the value in OCR0A (WGM03:0 = 11). The counter has then reached the TOP and 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 11-10 on page 66. The figure shows phase correct PWM mode when OCR0A or ICR0 is used to define TOP. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-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. The OC0x interrupt flag will be set when a compare match occurs.

Figure 11-10. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. When either OCR0A or ICR0 is used for defining the TOP value, the OC0A or ICF0 flag is set accordingly at the same timer clock cycle as the OCR0x 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 the TCNT0 and the OCR0x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR0x Registers are written. As the third period shown in Figure 11-10 on page 66 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR0x Register. Since the OCR0x 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 unsymmetrical 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 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 to three (See Table 11-4 on page 74). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC0x). The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and clearing (or setting) 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:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

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

The extreme values for the OCR0x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0x 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.

11.8.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation, or phase and frequency correct PWM mode (WGM03:0 = 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 (OC0x) is cleared on the compare match between TCNT0 and OCR0x while upcounting, and set on the compare match while downcounting. 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 OCR0x Register is updated by the OCR0x Buffer Register, (see Figure 11-10 on page 66 and Figure 11-11 on page 68).

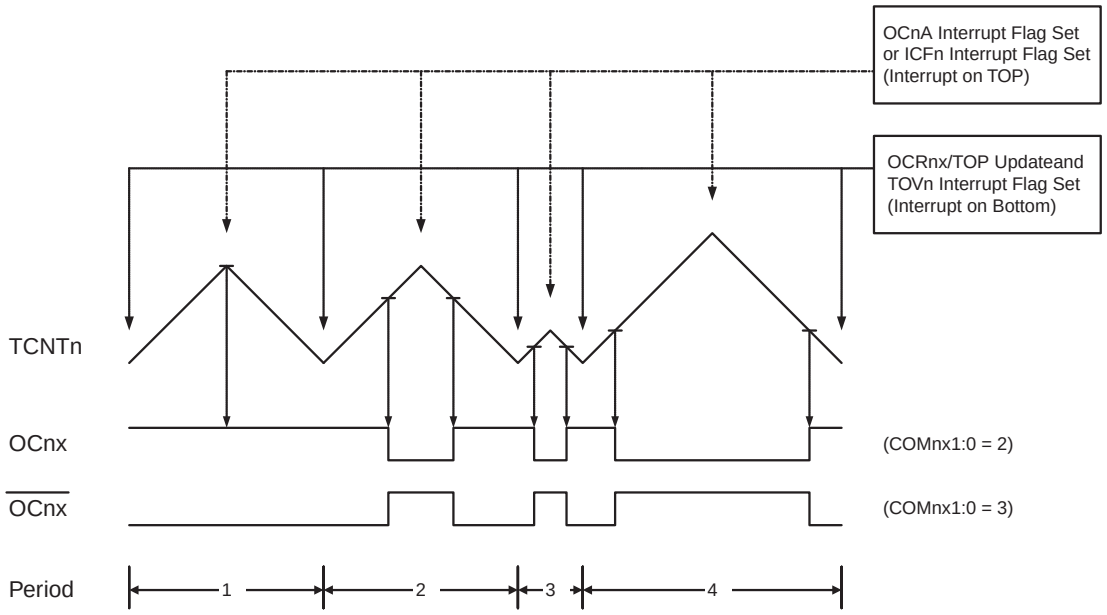
The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR0 or OCR0A. The minimum resolution allowed is 2-bit (ICR0 or OCR0A set to 0x0003), and

the maximum resolution is 16-bit (ICR0 or OCR0A 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 ICR0 (WGM03:0 = 8), or the value in OCR0A (WGM03:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 11-11 on page 68. The figure shows phase and frequency correct PWM mode when OCR0A or ICR0 is used to define TOP. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-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. The OC0x interrupt flag will be set when a compare match occurs.

Figure 11-11. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set at the same timer clock cycle as the OCR0x Registers are updated with the double buffer value (at BOTTOM). When either OCR0A or ICR0 is used for defining the TOP value, the OC0A or ICF0 flag set when TCNT0 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 the TCNT0 and the OCR0x.

As Figure 11-11 on page 68 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR0x 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 ICR0 Register for defining TOP works well when using fixed TOP values. By using ICR0, the OCR0A Register is free to be used for generating a PWM output on OC0A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR0A 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 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 to three (See Table 11-4 on page 74). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC0x). The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and clearing (or setting) the OC0x Register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

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

The extreme values for the OCR0x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0x 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.

11.9 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR0x Register is updated with the OCR0x buffer value (only for modes utilizing double buffering). Figure 11-12 on page 69 shows a timing diagram for the setting of OCF0x.

Figure 11-12. Timer/Counter Timing Diagram, Setting of OCF0x, no Prescaling

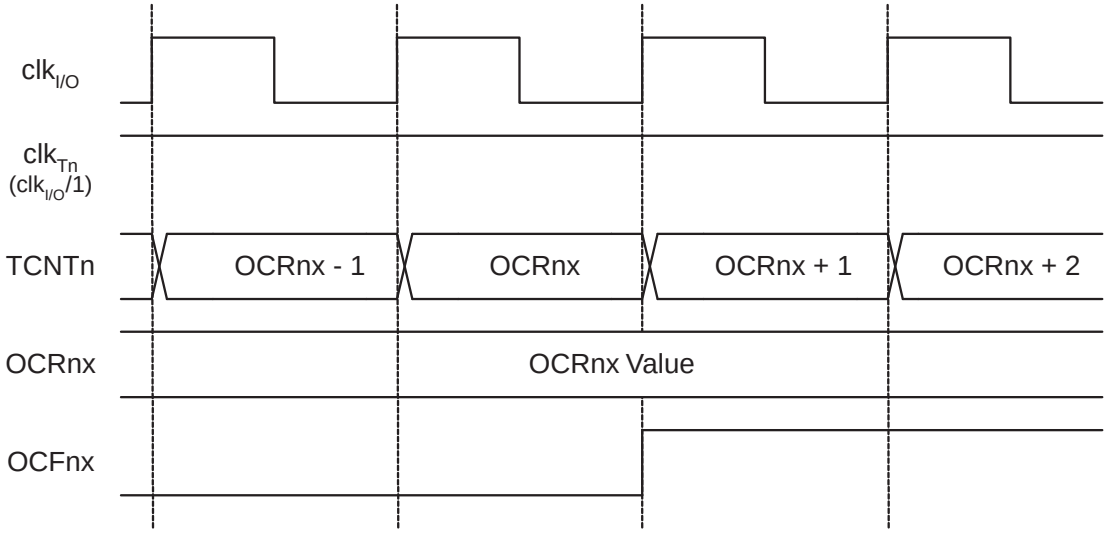


Figure 11-13 on page 70 shows the same timing data, but with the prescaler enabled.

Figure 11-13. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk_I/O}/8$)

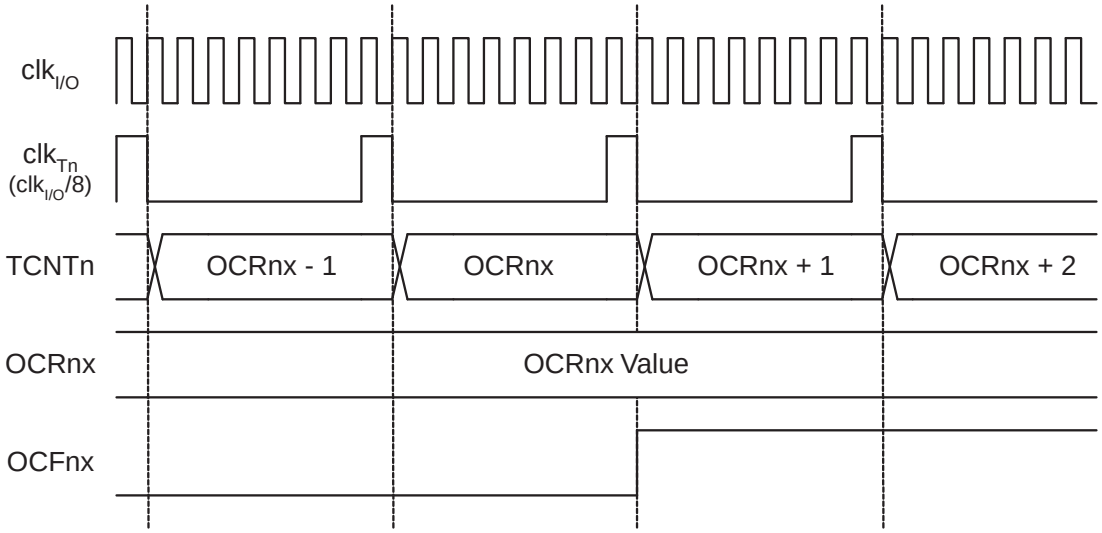


Figure 11-14 on page 70 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR0x Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV0 flag at BOTTOM.

Figure 11-14. Timer/Counter Timing Diagram, no Prescaling

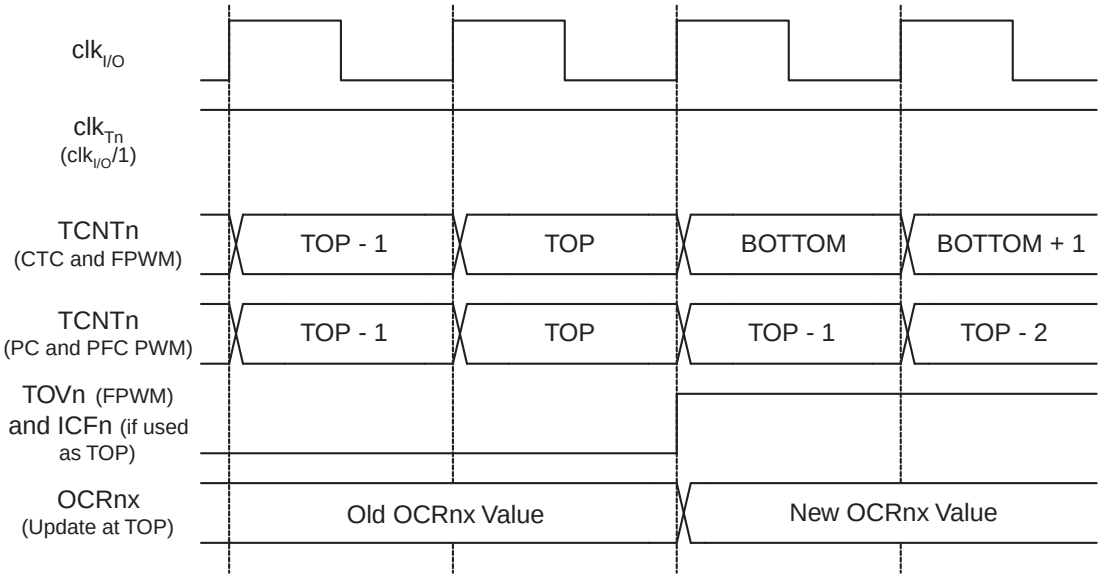
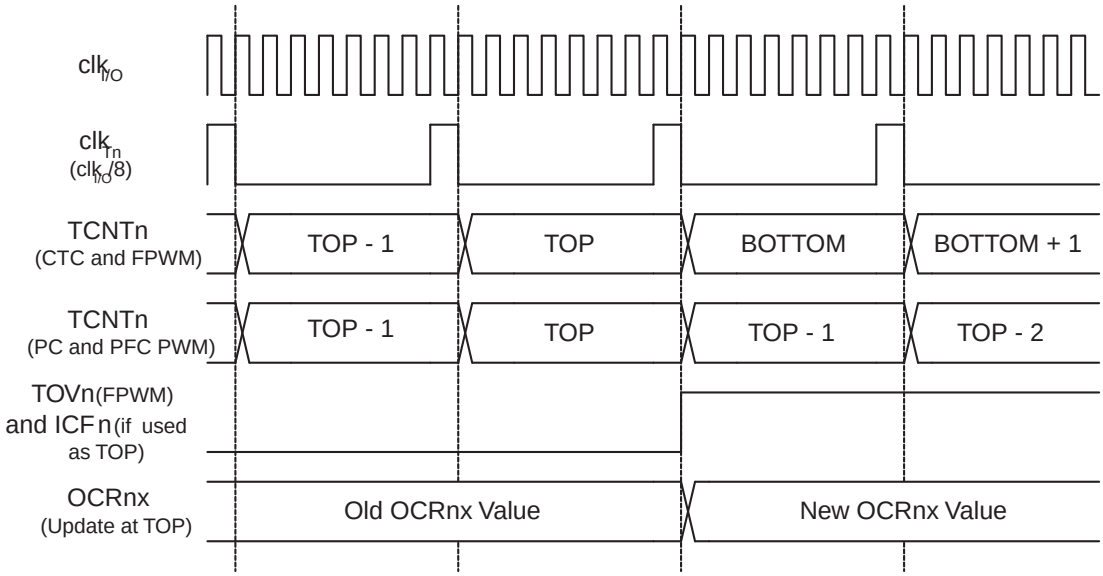


Figure 11-15 on page 71 shows the same timing data, but with the prescaler enabled.

Figure 11-15. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)



11.10 Accessing 16-bit Registers

The TCNT0, OCR0A/B, and ICR0 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing 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 uses the temporary register for the high byte. Reading the OCR0A/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 example shows 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 OCR0A/B and ICR0 Registers.

Assembly Code Example

```
...
; Set TCNT0 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT0H,r17
out TCNT0L,r16
; Read TCNT0 into r17:r16
in  r16,TCNT0L
in  r17,TCNT0H
...
```

Note: See “Code Examples” on page 5.

The code example returns the TCNT0 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 other of the 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 example shows how to do an atomic read of the TCNT0 Register contents. Reading any of the OCR0A/B or ICR0 Registers can be done by using the same principle.

Assembly Code Example

```
TIM16_ReadTCNT0:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNT0 into r17:r16
  in  r16,TCNT0L
  in  r17,TCNT0H
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

Note: See “Code Examples” on page 5.

The code example returns the TCNT0 value in the r17:r16 register pair.

The following code example shows how to do an atomic write of the TCNT0 Register contents. Writing any of the OCR0A/B or ICR0 Registers can be done by using the same principle.

Assembly Code Example

```
TIM16_WriteTCNT0:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNT0 to r17:r16
  out TCNT0H,r17
  out TCNT0L,r16
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

Note: See “Code Examples” on page 5.

The code example requires that the r17:r16 register pair contains the value to be written to TCNT0.

11.10.1 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, 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.

11.11 Register Description

11.11.1 TCCR0A – Timer/Counter0 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2E	COM0A1	COM0A0	COM0B1	COM0B0	–	–	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 Output Mode for Channel A**

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

The COM0A1:0 and COM0B1:0 control the behaviour of Output Compare pins OC0A and OC0B, respectively. If one or both COM0A1:0 bits are written to one, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. Similarly, if one or both COM0B1:0 bit are written to one, the OC0B output overrides the normal port functionality of the I/O pin it is connected to.

Note, however, that the Data Direction Register (DDR) bit corresponding to the OC0A or OC0B pin must be set in order to enable the output driver.

When OC0A or OC0B is connected to the pin, the function of COM0x1:0 bits depends on the WGM03:0 bits. Table 11-2 shows the COM0x1:0 bit functionality when the WGM03:0 bits are set to a Normal or CTC (non-PWM) Mode.

Table 11-2. Compare Output in Non-PWM Modes

COM0A1/ COM0B1	COM0A0 COM0B0	Description
0	0	Normal port operation: OC0A/OC0B disconnected
	1	Toggle OC0A/OC0B on compare match
1	0	Clear (set low) OC0A/OC0B on compare match
	1	Set (high) OC0A/OC0B on compare match

Table 11-3 shows the COM0x1:0 bit functionality when the WGM03:0 bits are set to one of the Fast PWM Modes.

Table 11-3. Compare Output in Fast PWM Modes

COM0A1/ COM0B1	COM0A0/ COM0B0	Description
0	0	Normal port operation: OC0A/OC0B disconnected
	1	WGM03 = 0: Normal port operation, OC0A/OC0B disconnected WGM03 = 1: Toggle OC0A on compare match, OC0B reserved
1 (1)	0	Clear OC0A/OC0B on compare match Set OC0A/OC0B at BOTTOM (non-inverting mode)
	1	Set OC0A/OC0B on compare match Clear OC0A/OC0B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0A/OCR0B equals TOP and COM0A1/COM0B1 is set. In this case the compare match is ignored, but set or clear is done at BOTTOM. See "Fast PWM Mode" on page 63 for more details.

Table 11-4 shows the COM0x1:0 bit functionality when the WGM03:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 11-4. Compare Output in Phase Correct and Phase & Frequency Correct PWM Modes

COM0A1/ COM0B1	COM0A0/ COM0B0	Description
0	0	Normal port operation: OC0A/OC0B disconnected.
	1	WGM03 = 0: Normal port operation, OC0A/OC0B disconnected WGM03 = 1: Toggle OC0A on compare match, OC0B reserved
1 (1)	0	Counting up: Clear OC0A/OC0B on compare match Counting down: Set OC0A/OC0B on compare match
	1	Counting up: Set OC0A/OC0B on compare match Counting down: Clear OC0A/OC0B on compare match

Note: 1. A special case occurs when OCR0A/OCR0B equals TOP and COM0A1/COM0B1 is set. "Phase Correct PWM Mode" on page 65 for more details.

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

Combined with WGM03:2 bits of TCCR0B, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform to generate. See Table 11-5. 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. ("Modes of Operation" on page 62).

Table 11-5. Waveform Generation Modes

Mode	WGM0 3:0	Mode of Operation	TOP	Update of OCR0x at	TOV0 Flag Set on
0	0000	Normal	0xFFFF	Immediate	MAX
1	0001	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0010	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0011	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0100	CTC (Clear Timer on Compare)	OCR0A	Immediate	MAX
5	0101	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0110	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0111	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1000	PWM, Phase & Freq. Correct	ICR0	BOTTOM	BOTTOM
9	1001	PWM, Phase & Freq. Correct	OCR0A	BOTTOM	BOTTOM
10	1010	PWM, Phase Correct	ICR0	TOP	BOTTOM
11	1011	PWM, Phase Correct	OCR0A	TOP	BOTTOM
12	1100	CTC (Clear Timer on Compare)	ICR0	Immediate	MAX
13	1101	(Reserved)	–	–	–
14	1110	Fast PWM	ICR0	TOP	TOP
15	1111	Fast PWM	OCR0A	TOP	TOP

11.11.2 TCCR0B – Timer/Counter0 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2D	ICNC0	ICES0	–	WGM03	WGM02	CS02	CS01	CS00	TCCR0B
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 – ICNC0: Input Capture Noise Canceler**

Setting this bit (to one) activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture pin (ICP0) is filtered. The filter function requires four successive equal valued samples of the ICP0 pin for changing its output. The Input Capture is therefore delayed by four oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICES0: Input Capture Edge Select**

This bit selects which edge on the Input Capture pin (ICP0) that is used to trigger a capture event. When the ICES0 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES0 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES0 setting, the counter value is copied into the Input Capture Register (ICR0). The event will also set the Input Capture Flag (ICF0), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR0 is used as TOP value (see description of the WGM03:0 bits located in the TCCR0A and the TCCR0B Register), the ICP0 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 zero when TCCR0B is written.

- **Bits 4:3 – WGM03:2: Waveform Generation Mode**

See “TCCR0A – Timer/Counter0 Control Register A” on page 73.

- **Bits 2:0 – CS02:0: Clock Select**

The three Clock Select bits set the clock source to be used by the Timer/Counter, see Figure 11-12 and Figure 11-13.

Table 11-6. Clock Select Bit Description

CS02	CS01	CS00	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 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.

11.11.3 TCCR0C – Timer/Counter0 Control Register C

Bit	7	6	5	4	3	2	1	0	
0x2C	FOC0A	FOC0B	–	–	–	–	–	–	TCCR0C
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

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

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

The FOC0A/FOC0B bits are only active when the WGM03:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR0A is written when operating in a PWM mode. When writing a logical one to the FOC0A/FOC0B bit, an immediate compare match is forced on the Waveform Generation unit.

The OC0A/OC0B output is changed according to its COM0x1:0 bits setting. Note that the FOC0A/FOC0B bits are implemented as strobes. Therefore it is the value present in the COM0x1:0 bits that determine the effect of the forced compare.

A FOC0A/FOC0B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR0A as TOP.

The FOC0A/FOC0B bits are always read as zero.

- **Bits 5:0 – Reserved Bits**

These bits are reserved for future use. For ensuring compatibility with future devices, these bits must be written to zero when the register is written.

11.11.4 TCNT0H and TCNT0L – Timer/Counter0

Bit	7	6	5	4	3	2	1	0	
0x29	TCNT0[15:8]								TCNT0H
0x28	TCNT0[7:0]								TCNT0L
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 (TCNT0H and TCNT0L, combined TCNT0) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 71.

Modifying the counter (TCNT0) while the counter is running introduces a risk of missing a compare match between TCNT0 and one of the OCR0x Registers.

Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock for all compare units.

11.11.5 OCR0AH and OCR0AL – Output Compare Register 0 A

Bit	7	6	5	4	3	2	1	0	
0x27	OCR1A[15:8]								OCR0AH
0x26	OCR1A[7:0]								OCR0AL
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	

11.11.6 OCR0BH and OCR0BL – Output Compare Register 0 B

Bit	7	6	5	4	3	2	1	0	
0x25	OCR0B[15:8]								OCR0BH
0x24	OCR0B[7:0]								OCR0BL
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 (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0x pin.

The Output Compare Registers are 16-bit 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 “Accessing 16-bit Registers” on page 71.

11.11.7 ICR0H and ICR0L – Input Capture Register 0

Bit	7	6	5	4	3	2	1	0	
0x23	ICR0[15:8]								ICR0H
0x22	ICR0[7:0]								ICR0L
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 (TCNT0) value each time an event occurs on the ICP0 pin (or optionally on the Analog Comparator output for Timer/Counter0). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit 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. “Accessing 16-bit Registers” on page 71.

11.11.8 TIMSK0 – Timer/Counter Interrupt Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x2B	–	–	ICIE0	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6, 4:3 – Reserved Bits**

These bits are reserved for future use. For ensuring compatibility with future devices, these bits must be written to zero when the register is written.

- **Bit 5 – ICIE0: Timer/Counter0, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Input Capture interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 66.) is executed when the ICF0 Flag, located in TIFR0, is set.

- **Bit 2 – OCIE0B: Timer/Counter0, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 35) is executed when the OCF0B flag, located in TIFR0, is set.

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

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 35) is executed when the OCF0A flag, located in TIFR0, is set.

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

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Overflow interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 35) is executed when the TOV0 flag, located in TIFR0, is set.

11.11.9 TIFR0 – Timer/Counter Interrupt Flag Register 0

Bit	7	6	5	4	3	2	1	0	
0x2A	–	–	ICF0	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6, 4:3 – Reserved Bits**

These bits are reserved for future use. For ensuring compatibility with future devices, these bits must be written to zero when the register is written.

- **Bit 5 – ICF0: Timer/Counter0, Input Capture Flag**

This flag is set when a capture event occurs on the ICP0 pin. When the Input Capture Register (ICR0) is set by the WGM03:0 to be used as the TOP value, the ICF0 flag is set when the counter reaches the TOP value.

ICF0 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF0 can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF1B: Timer/Counter0, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT0) value matches the Output Compare Register B (OCR0B).

Note that a Forced Output Compare (0B) strobe will not set the OCF0B flag.

OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF0A: Timer/Counter0, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT0) value matches the Output Compare Register A (OCR0A).

Note that a Forced Output Compare (1A) strobe will not set the OCF0A flag.

OCF0A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF0A can be cleared by writing a logic one to its bit location.

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

The setting of this flag is dependent of the WGM03:0 bits setting. In Normal and CTC modes, the TOV0 flag is set when the timer overflows. See Table 11-5 on page 75 for the TOV0 flag behavior when using another WGM03:0 bit setting.

TOV0 is automatically cleared when the Timer/Counter0 Overflow Interrupt Vector is executed. Alternatively, TOV0 can be cleared by writing a logic one to its bit location.

11.11.10 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x2F	TSM	–	–	–	–	–	–	PSR	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 one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR 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 zero, the PSR bit is cleared by hardware, and the Timer/Counter start counting.

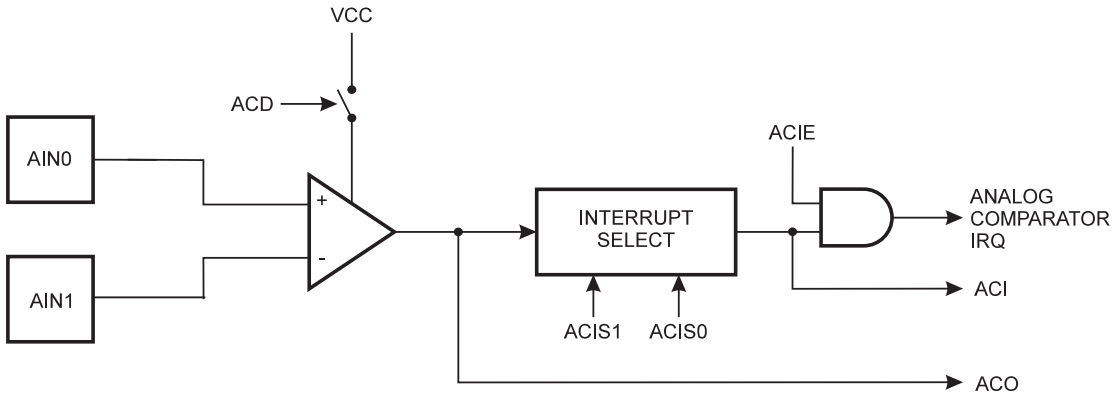
- **Bit 0 – PSR: Prescaler 0 Reset Timer/Counter 0**

When this bit is one, the Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

12. 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 12-1.

Figure 12-1. Analog Comparator Block Diagram.



See Figure 1-1 on page 2 for pin use of analog comparator, and Table 10-4 on page 49 and Table 10-5 on page 50 for alternate pin usage.

12.1 Register Description

12.1.1 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x1F	**ACD**	**–**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

When this bit is written logic one, the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator, thus reducing 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.

- **Bits 6 – Res: Reserved Bit**

This bit is reserved and will always read zero.

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

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

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

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

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

When the ACIE bit is written logic one, the Analog Comparator interrupt request is enabled. When written logic zero, the interrupt request is disabled.

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

When set, this bit enables the input capture function in Timer/Counter0 to be triggered by the analog comparator. In this case, the comparator output is directly connected to the input capture front-end logic, using the noise canceler and edge select features of the Timer/Counter0 input capture interrupt. To make the comparator trigger the Timer/Counter0 input capture interrupt, the ICIE1 bit in “TIMSK0 – Timer/Counter Interrupt Mask Register 0” must be set.

When this bit is cleared, no connection between the analog comparator and the input capture function exists.

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

These bits determine which comparator events that trigger the analog comparator interrupt. The different settings are shown in Table 12-1.

Table 12-1. Selecting Source for Analog Comparator Interrupt.

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 “ACSR – Analog Comparator Control and Status Register”. Otherwise an interrupt can occur when the bits are changed.

12.1.2 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
0x17	–	–	–	–	–	–	ADC1D	ADC0D	DIDR0
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1:0 – ADC1D, ADC0D: Digital Input Disable**

When this bit is set, the digital input buffer on pin AIN1 (ADC1) / AIN0 (ADC0) is disabled and the corresponding PIN register bit will read as zero. When used as an analog input but not required as a digital input the power consumption in the digital input buffer can be reduced by writing this bit to logic one.

13. Analog to Digital Converter

13.1 Features

- **8-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **± 1 LSB Absolute Accuracy**
- **65µs Conversion Time**
- **15 kSPS at Full Resolution**
- **Four Multiplexed Single Ended Input Channels**
- **Input Voltage Range: 0 – V_{CC}**
- **Supply Voltage Range: 2.5V – 5.5V**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

13.2 Overview

ATtiny5/10 feature an 8-bit, successive approximation ADC. The ADC is connected to a 4-channel analog multiplexer which allows four single-ended voltage inputs constructed from the pins of port B. The single-ended voltage inputs refer to 0V (GND).

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 13-1 on page 84.

Internal reference voltage of V_{CC} is provided on-chip.

The ADCis not available in ATtiny4/9.

13.3 Operation

In order to be able to use the ADC the Power Reduction bit, PRADC, in the Power Reduction Register must be disabled. This is done by clearing the PRADC bit. See “PRR – Power Reduction Register” on page 26 for more details.

The ADC is enabled by setting the ADC Enable bit, ADEN in “ADCSRA – ADC Control and Status Register A”. 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.

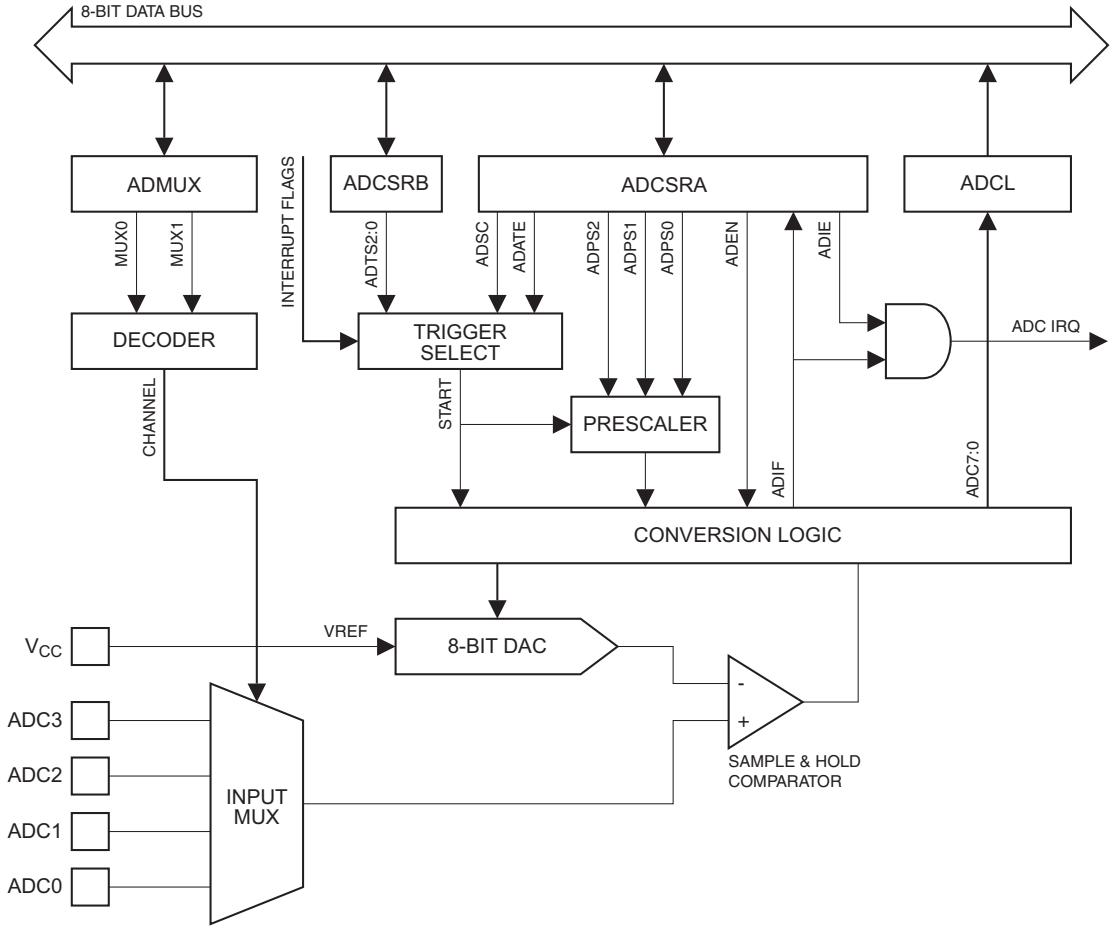
The ADC converts an analog input voltage to an 8-bit digital value using successive approximation. The minimum value represents GND and the maximum value represents the voltage on V_{CC}.

The analog input channel is selected by writing MUX1:0 bits. See “ADMUX – ADC Multiplexer Selection Register” on page 93. Any of the ADC input pins can be selected as single ended inputs to the ADC.

The ADC generates an 8-bit result which is presented in the ADC data register. See “ADCL – ADC Data Register” on page 95.

The ADC has its own interrupt request which can be triggered when a conversion completes.

Figure 13-1. Analog to Digital Converter Block Schematic



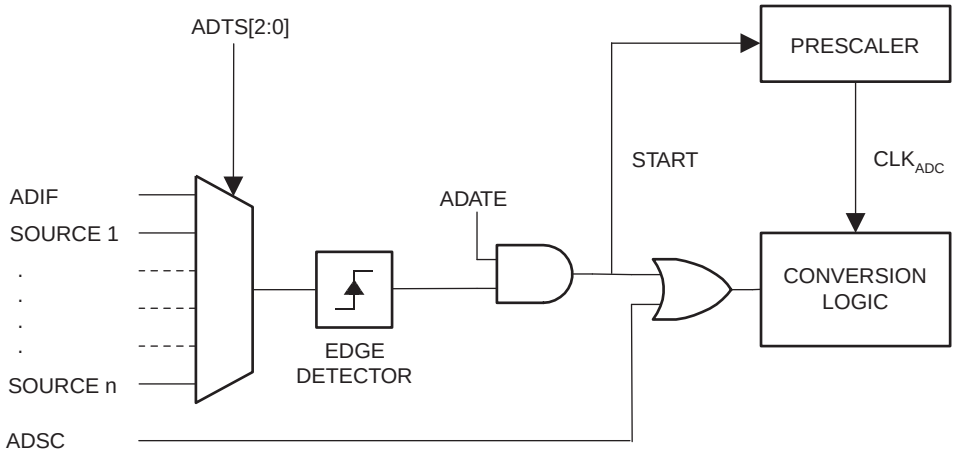
13.4 Starting a Conversion

Make sure the ADC is powered by clearing the ADC Power Reduction bit, PRADC, in the Power Reduction Register, PRR (see “PRR – Power Reduction Register” on page 26).

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 – ADC Control and Status Register B”. See Table 13-4 on page 95 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. 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 13-2. ADC Auto Trigger Logic



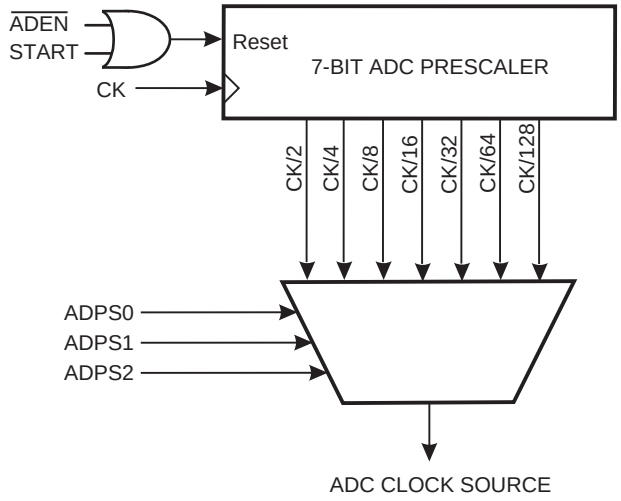
Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to bit 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 one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

13.5 Prescaling and Conversion Timing

By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution.

Figure 13-3. ADC Prescaler



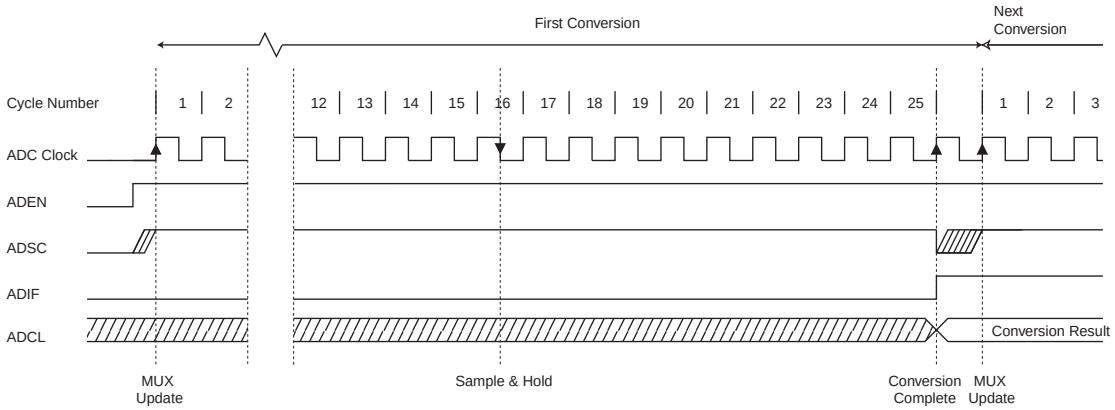
The ADC module contains a prescaler, as illustrated in Figure 13-3 on page 85, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is

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.

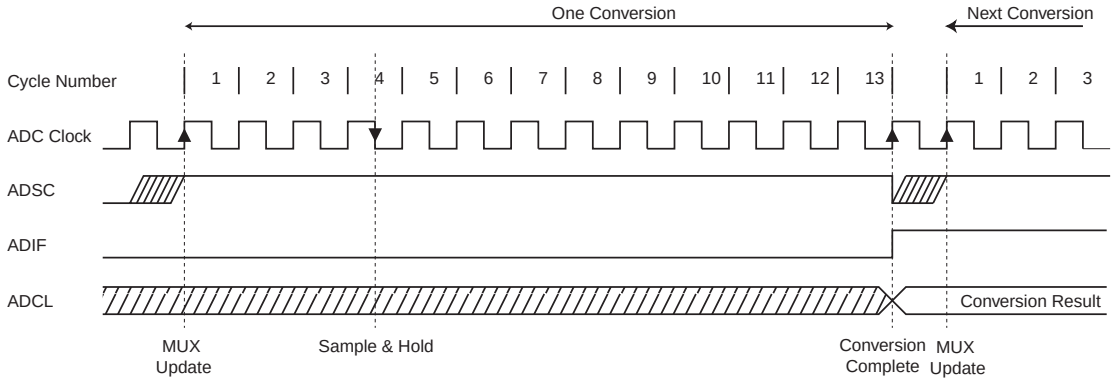
A normal conversion takes 13 ADC clock cycles, as summarised in Table 13-1 on page 87. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry. See Figure 13-4.

Figure 13-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)



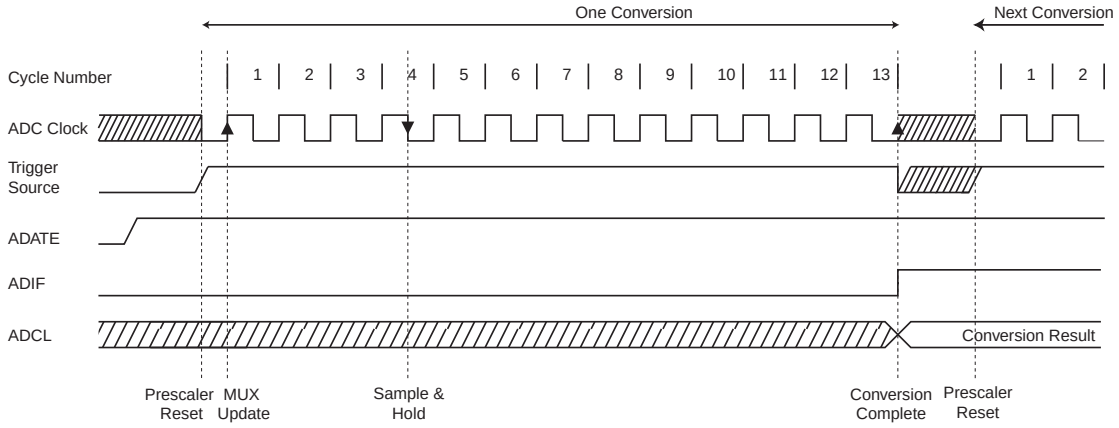
The actual sample-and-hold takes place 3 ADC clock cycles after the start of a normal conversion and 16 ADC clock cycles after the start of a first conversion. See Figure 13-5. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, 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.

Figure 13-5. ADC Timing Diagram, Single Conversion



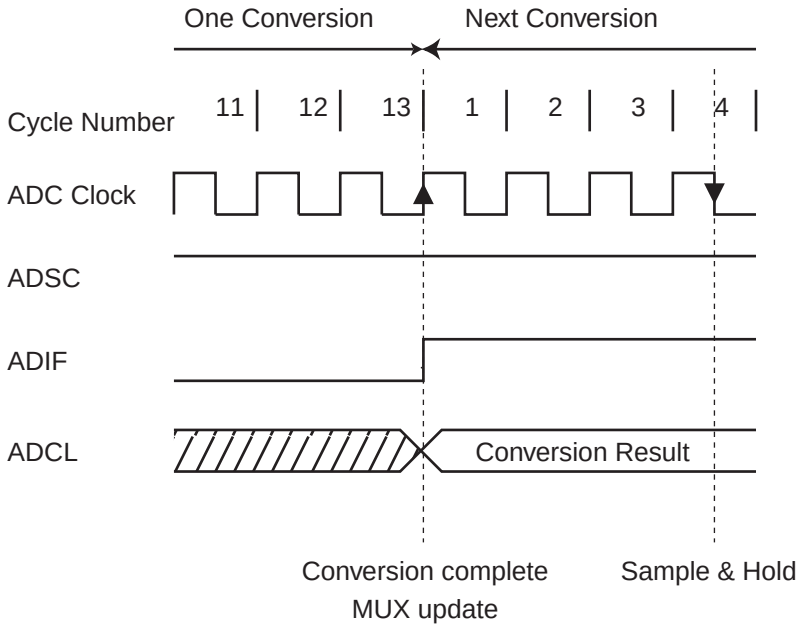
When Auto Triggering is used, the prescaler is reset when the trigger event occurs. See Figure 13-6. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

Figure 13-6. ADC Timing Diagram, Auto Triggered Conversion



In Free Running mode (see Figure 13-7), a new conversion will be started immediately after the conversion completes, while ADSC remains high.

Figure 13-7. ADC Timing Diagram, Free Running Conversion



For a summary of conversion times, see Table 13-1.

Table 13-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	16.5	25
Normal conversions	3.5	13
Auto Triggered conversions	4	13.5

13.6 Changing Channel

The MUXn bits in the ADMUX Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channel selection only takes place at a safe point during the conversion. The channel is continuously updated until a conversion is started. Once the conversion starts, the channel 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 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 settings.

If both ADATE and ADEN is written to 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 settings. ADMUX can be safely updated in the following ways:

- When ADATE or ADEN is cleared.
- During conversion, minimum one ADC clock cycle after the trigger event.
- After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

13.6.1 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 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 one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

13.6.2 ADC Voltage Reference

The reference voltage of the ADC determines the conversion range, which in this case is limited to 0V (V_{GND}) and $V_{REF} = V_{cc}$. Channels that exceed V_{REF} will result in codes saturated at 0xFF.

13.7 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

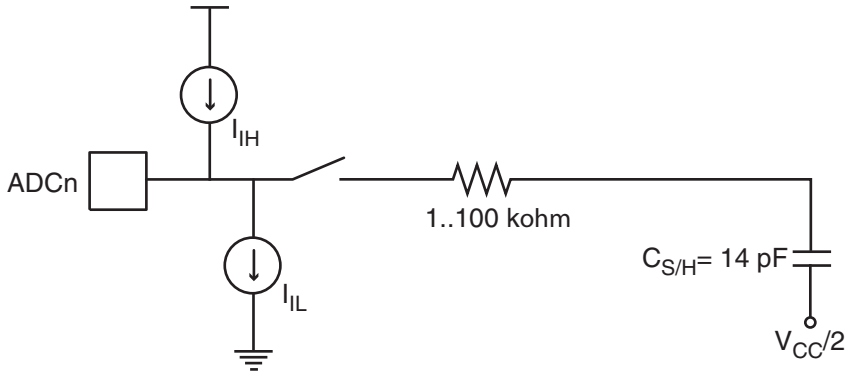
- Make sure that the ADC is enabled and is not busy converting. Single Conversion mode must be selected and the ADC conversion complete interrupt must be enabled.
- Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
- If no other interrupts occur 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 other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption.

13.8 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 13-8 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. When the channel is selected, the source must drive the S/H (sample and hold) capacitor through the series resistance (combined resistance in the input path).

Figure 13-8. Analog Input Circuitry



The capacitor in Figure 13-8 depicts the total capacitance, including the sample/hold capacitor and any stray or parasitic capacitance inside the device. The value given is worst case.

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ, or less. With such sources, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor. This can vary widely. The user is recommended to only use low impedance 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.

13.9 Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. When conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

- Keep analog signal paths as short as possible.
- Make sure analog tracks run over the analog ground plane.
- Keep analog tracks well away from high-speed switching digital tracks.
- If any port pin is used as a digital output, it mustn't switch while a conversion is in progress.
- Place bypass capacitors as close to V_{CC} and GND pins as possible.

Where high ADC accuracy is required it is recommended to use ADC Noise Reduction Mode, as described in Section 13.7 on page 88. A good system design with properly placed, external bypass capacitors does reduce the need for using ADC Noise Reduction Mode

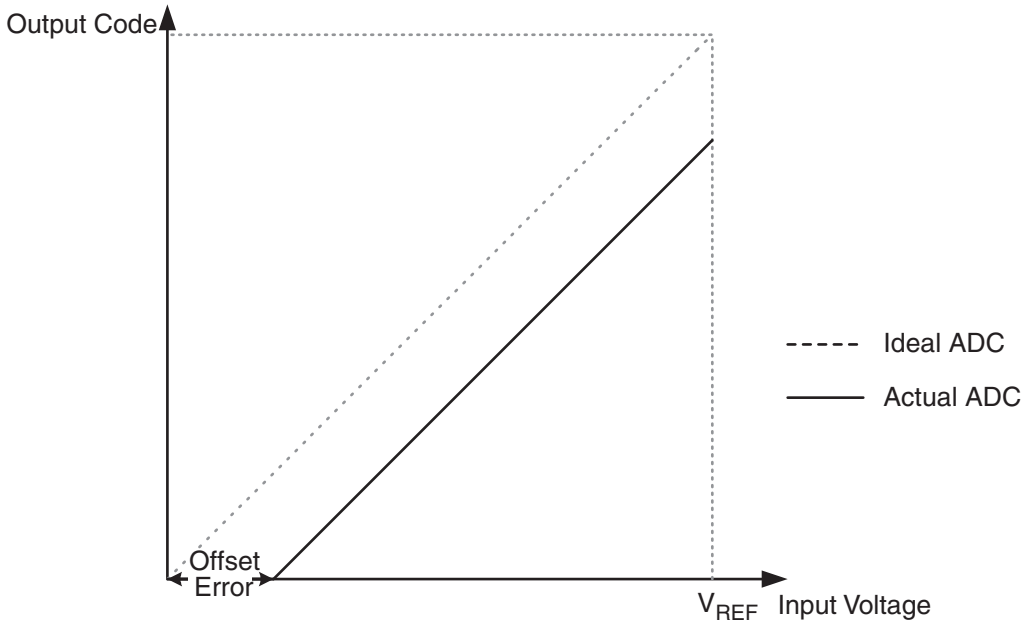
13.10 ADC Accuracy Definitions

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:

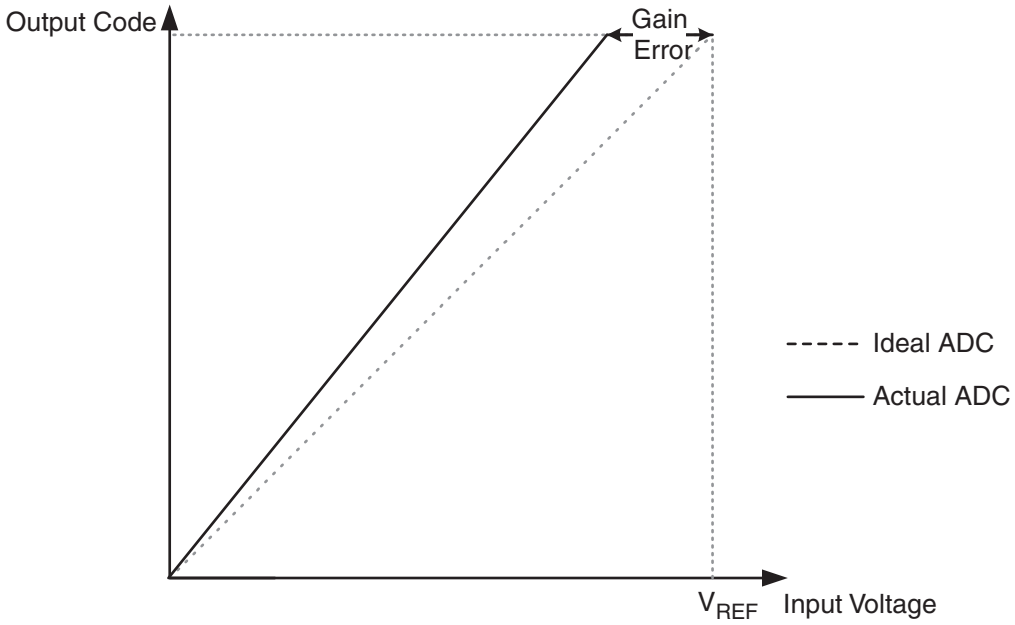
- Offset: The deviation of the first transition (0x00 to 0x01) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 13-9. Offset Error



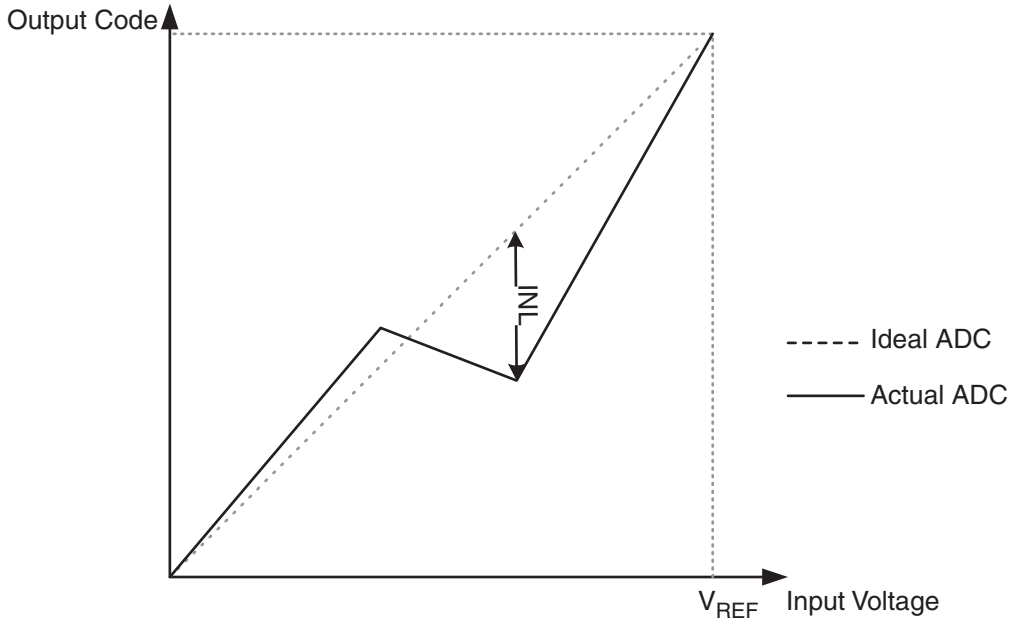
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0xFE to 0xFF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 13-10. Gain Error



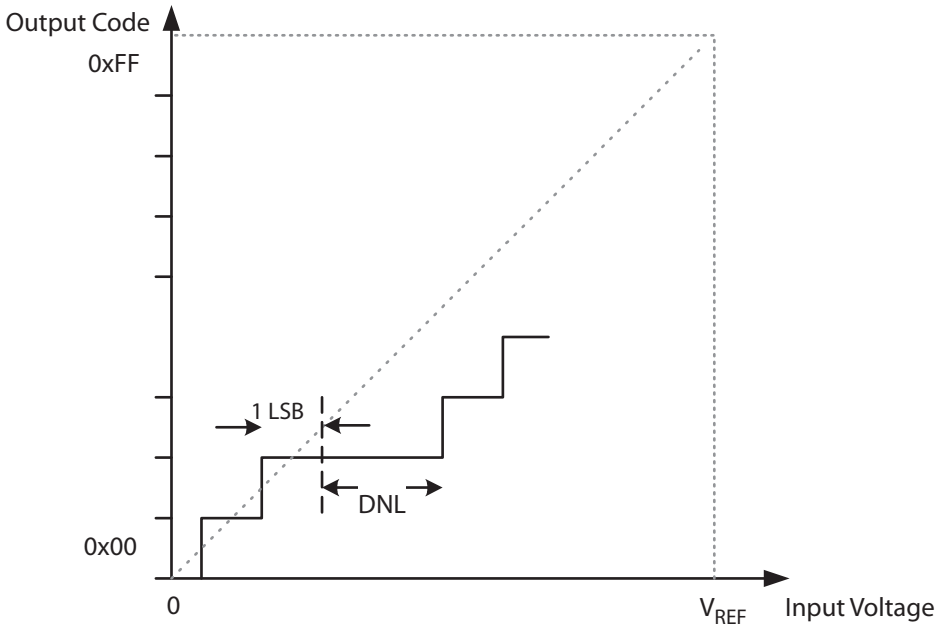
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 13-11. 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 (1 LSB). Ideal value: 0 LSB.

Figure 13-12. 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 (1 LSB wide) will code to the same value. Always ± 0.5 LSB.

- 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, gain error, differential error, non-linearity, and quantization error. Ideal value: ± 0.5 LSB.

13.11 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Data Register (ADCL). For single ended conversion, the result is

$$ADCL = \frac{V_{IN} \cdot 256}{V_{CC}}$$

where V_{IN} (see Table 13-2 on page 93) is the voltage on the selected input pin and V_{CC} is the voltage reference. 0x00 represents analog ground, and 0xFF represents the selected reference voltage minus one LSB.

13.12 Register Description

13.12.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x1B	–	–	–	–	–	–	MUX1	MUX0	ADMUX
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bits 1:0 – MUX1:0: Analog Channel Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 13-2 for details. If these bits are changed during a conversion, the change will not go in effect until the conversion is complete (ADIF in ADCSRA is set).

Table 13-2. Input Channel Selections

MUX1	MUX0	Single Ended Input
0	0	ADC0 (PB0)
	1	ADC1 (PB1)
1	0	ADC2 (PB2)
	1	ADC3 (PB3)

13.12.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x1D	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 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 this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written 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 performs initialization of 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 zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to 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 requested if the ADIE bit is 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.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one, the ADC Conversion Complete Interrupt request is enabled.

- **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 13-3. 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

13.12.3 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x1C	–	–	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
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 and will always read zero.

- **Bits 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to 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 13-4. 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 Flag 0
0	1	1	Timer/Counter 0 Compare Match A
1	0	0	Timer/Counter 0 Overflow
1	0	1	Timer/Counter 0 Compare Match B
1	1	0	Pin Change Interrupt 0 Request
1	1	1	Timer/Counter 0 Capture Event

13.12.4 ADCL – ADC Data Register

Bit	7	6	5	4	3	2	1	0	
0x19	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in the ADC register.

- **Bits 7:0 – ADC7:0: ADC Conversion Result**

These bits represent the result from the conversion.

13.12.5 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
0x17	–	–	–	–	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
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 Bit**

These bits are reserved and will always read zero.

- **Bits 3:0 – ADC3D..ADC0D: ADC3..0 Digital Input Disable**

When this bit is written logic 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 ADC3..0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

14. Programming interface

14.1 Features

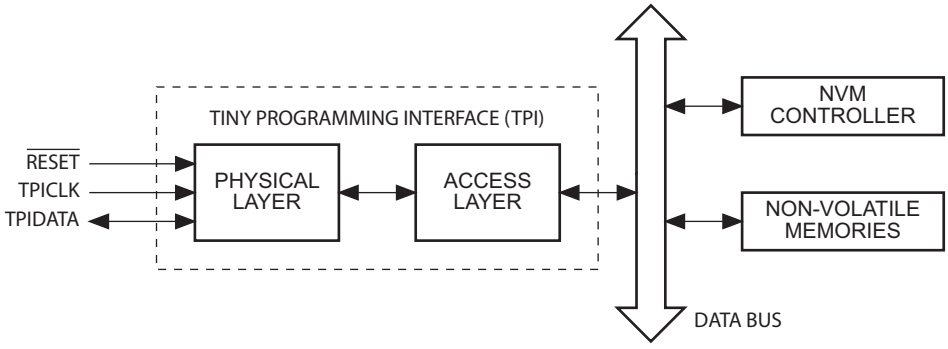
- **Physical Layer:**
  - **Synchronous Data Transfer**
  - **Bi-directional, Half-duplex Receiver And Transmitter**
  - **Fixed Frame Format With One Start Bit, 8 Data Bits, One Parity Bit And 2 Stop Bits**
  - **Parity Error Detection, Frame Error Detection And Break Character Detection**
  - **Parity Generation And Collision Detection**
  - **Automatic Guard Time Insertion Between Data Reception And Transmission**
- **Access Layer:**
  - **Communication Based On Messages**
  - **Automatic Exception Handling Mechanism**
  - **Compact Instruction Set**
  - **NVM Programming Access Control**
  - **Tiny Programming Interface Control And Status Space Access Control**
  - **Data Space Access Control**

14.2 Overview

The Tiny Programming Interface (TPI) supports external programming of all Non-Volatile Memories (NVM). Memory programming is done via the NVM Controller, by executing NVM controller commands as described in “Memory Programming” on page 107.

The Tiny Programming Interface (TPI) provides access to the programming facilities. The interface consists of two layers: the access layer and the physical layer. The layers are illustrated in Figure 14-1.

Figure 14-1. The Tiny Programming Interface and Related Internal Interfaces



Programming is done via the physical interface. This is a 3-pin interface, which uses the $\overline{\text{RESET}}$ pin as enable, the TPICLK pin as the clock input, and the TPIDATA pin as data input and output.

NVM can be programmed at 5V, only.

14.3 Physical Layer of Tiny Programming Interface

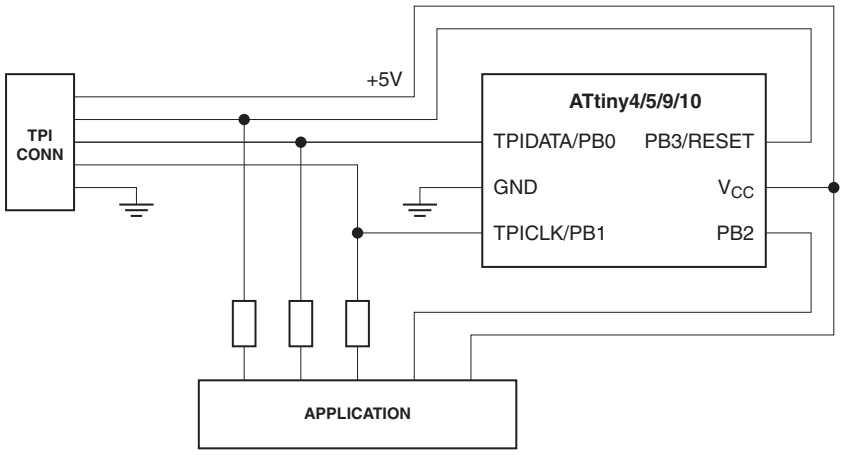
The TPI physical layer handles the basic low-level serial communication. The TPI physical layer uses a bi-directional, half-duplex serial receiver and transmitter. The physical layer includes serial-to-parallel and parallel-to-serial data conversion, start-of-frame detection, frame error detection, parity error detection, parity generation and collision detection.

The TPI is accessed via three pins, as follows:

$\overline{\text{RESET}}$:	Tiny Programming Interface enable input
TPICLK:	Tiny Programming Interface clock input
TPIDATA:	Tiny Programming Interface data input/output

In addition, the V_{CC} and GND pins must be connected between the external programmer and the device. See Figure 14-2.

Figure 14-2. Using an External Programmer for In-System Programming via TPI



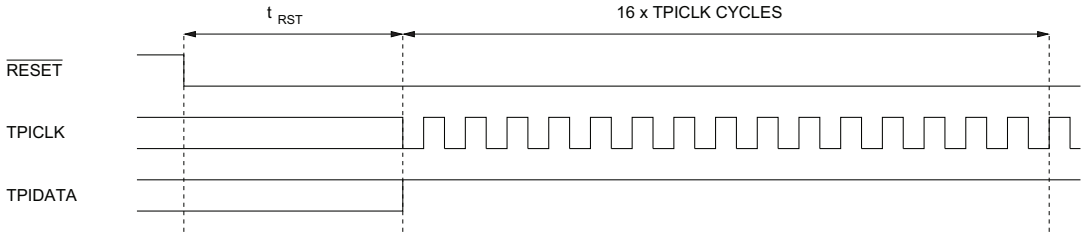
NVM can be programmed at 5V, only. In some designs it may be necessary to protect components that can not tolerate 5V with, for example, series resistors.

14.3.1 Enabling

The following sequence enables the Tiny Programming Interface (see Figure 14-3 for guidance):

- Apply 5V between V_{CC} and GND
- Depending on the method of reset to be used:
  - Either: wait t_{TOUT} (see Table 16-4 on page 119) and then set the $\overline{\text{RESET}}$ pin low. This will reset the device and enable the TPI physical layer. The $\overline{\text{RESET}}$ pin must then be kept low for the entire programming session
  - Or: if the RSTDISBL configuration bit has been programmed, apply 12V to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin must be kept at 12V for the entire programming session
- Wait t_{RST} (see Table 16-4 on page 119)
- Keep the TPIDATA pin high for 16 TPICLK cycles

Figure 14-3. Sequence for enabling the Tiny Programming Interface



14.3.2 Disabling

Provided that the NVM enable bit has been cleared, the TPI is automatically disabled if the $\overline{\text{RESET}}$ pin is released to inactive high state or, alternatively, if V_{HV} is no longer applied to the $\overline{\text{RESET}}$ pin.

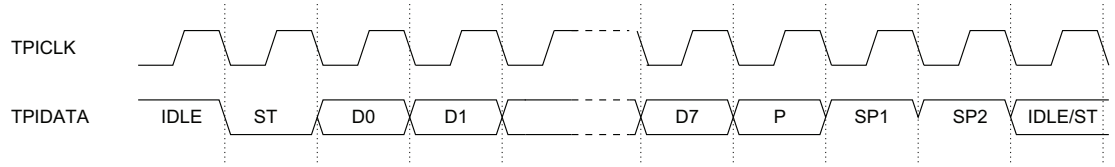
If the NVM enable bit is not cleared a power down is required to exit TPI programming mode.

See NVMEN bit in "TPISR – Tiny Programming Interface Status Register" on page 106.

14.3.3 Frame Format

The TPI physical layer supports a fixed frame format. A frame consists of one character, eight bits in length, and one start bit, a parity bit and two stop bits. Data is transferred with the least significant bit first.

Figure 14-4. Serial frame format.



Symbols used in Figure 14-4:

ST: Start bit (always low)
D0-D7: Data bits (least significant bit sent first)
P: Parity bit (using even parity)
SP1: Stop bit 1 (always high)
SP2: Stop bit 2 (always high)

14.3.4 Parity Bit Calculation

The parity bit is always calculated using even parity. The value of the bit is calculated by doing an exclusive-or of all the data bits, as follows:

$$P = D0 \otimes D1 \otimes D2 \otimes D3 \otimes D4 \otimes D5 \otimes D6 \otimes D7 \otimes 0$$

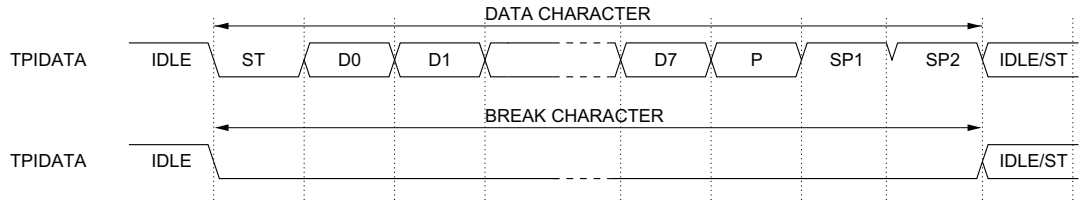
where:

P: Parity bit using even parity
D0-D7: Data bits of the character

14.3.5 Supported Characters

The BREAK character is equal to a 12 bit long low level. It can be extended beyond a bit-length of 12.

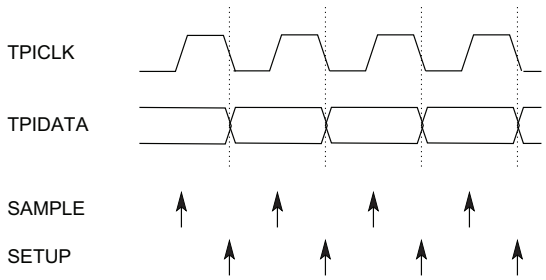
Figure 14-5. Supported characters.



14.3.6 Operation

The TPI physical layer operates synchronously on the TPICLK provided by the external programmer. The dependency between the clock edges and data sampling or data change is shown in Figure 14-6. Data is changed at falling edges and sampled at rising edges.

Figure 14-6. Data changing and Data sampling.



The TPI physical layer supports two modes of operation: Transmit and Receive. By default, the layer is in Receive mode, waiting for a start bit. The mode of operation is controlled by the access layer.

14.3.7 Serial Data Reception

When the TPI physical layer is in receive mode, data reception is started as soon as a start bit has been detected. Each bit that follows the start bit will be sampled at the rising edge of the TPICLK and shifted into the shift register until the second stop bit has been received. When the complete frame is present in the shift register the received data will be available for the TPI access layer.

There are three possible exceptions in the receive mode: frame error, parity error and break detection. All these exceptions are signalized to the TPI access layer, which then enters the error state and puts the TPI physical layer into receive mode, waiting for a BREAK character.

- Frame Error Exception. The frame error exception indicates the state of the stop bit. The frame error exception is set if the stop bit was read as zero.
- Parity Error Exception. The parity of the data bits is calculated during the frame reception. After the frame is received completely, the result is compared with the parity bit of the frame. If the comparison fails the parity error exception is signalized.
- Break Detection Exception. The Break detection exception is given when a complete frame of all zeros has been received.

14.3.8 Serial Data Transmission

When the TPI physical layer is ready to send a new frame it initiates data transmission by loading the shift register with the data to be transmitted. When the shift register has been loaded with new data, the transmitter shifts one complete frame out on the TPIDATA line at the transfer rate given by TPICLK.

If a collision is detected during transmission, the output driver is disabled. The TPI access layer enters the error state and the TPI physical layer is put into receive mode, waiting for a BREAK character.

14.3.9 Collision Detection Exception

The TPI physical layer uses one bi-directional data line for both data reception and transmission. A possible drive contention may occur, if the external programmer and the TPI physical layer drive the TPIDATA line simultaneously. In order to reduce the effect of the drive contention, a collision detection mechanism is supported. The collision detection is based on the way the TPI physical layer drives the TPIDATA line.

The TPIDATA line is driven by a tri-state, push-pull driver with internal pull-up. The output driver is always enabled when a logical zero is sent. When sending successive logical ones, the output is only driven actively during the first clock cycle. After this, the output driver is automatically tri-stated and the TPIDATA line is kept high by the internal pull-up. The output is re-enabled, when the next logical zero is sent.

The collision detection is enabled in transmit mode, when the output driver has been disabled. The data line should now be kept high by the internal pull-up and it is monitored to see, if it is driven low by the external programmer. If the output is read low, a collision has been detected.

There are some potential pit-falls related to the way collision detection is performed. For example, collisions cannot be detected when the TPI physical layer transmits a bit-stream of successive logical zeros, or bit-stream of alternating logical ones and zeros. This is because the output driver is active all the time, preventing polling of the TPIDATA line. However, within a single frame the two stop bits should always be transmitted as logical ones, enabling collision detection at least once per frame (as long as the frame format is not violated regarding the stop bits).

The TPI physical layer will cease transmission when it detects a collision on the TPIDATA line. The collision is signalized to the TPI access layer, which immediately changes the physical layer to receive mode and goes to the error state. The TPI access layer can be recovered from the error state only by sending a BREAK character.

14.3.10 Direction Change

In order to ensure correct timing of the half-duplex operation, a simple guard time mechanism has been added to the physical layer. When the TPI physical layer changes from receive to transmit mode, a configurable number of additional IDLE bits are inserted before the start bit is transmitted. The minimum transition time between receive and transmit mode is two IDLE bits. The total IDLE time is the specified guard time plus two IDLE bits.

The guard time is configured by dedicated bits in the TPIPCR register. The default guard time value after the physical layer is initialized is 128 bits.

The external programmer looses control of the TPIDATA line when the TPI target changes from receive mode to transmit. The guard time feature relaxes this critical phase of the communication. When the external programmer changes from receive mode to transmit, a minimum of one IDLE bit should be inserted before the start bit is transmitted.

14.4 Access Layer of Tiny Programming Interface

The TPI access layer is responsible for handling the communication with the external programmer. The communication is based on message format, where each message comprises an instruction followed by one or more byte-sized operands. The instruction is always sent by the external programmer but operands are sent either by the external programmer or by the TPI access layer, depending on the type of instruction issued.

The TPI access layer controls the character transfer direction on the TPI physical layer. It also handles the recovery from the error state after exception.

The Control and Status Space (CSS) of the Tiny Programming Interface is allocated for control and status registers in the TPI access Layer. The CSS consist of registers directly involved in the operation of the TPI itself. These register are accessible using the SLDCS and SSTCS instructions.

The access layer can also access the data space, either directly or indirectly using the Pointer Register (PR) as the address pointer. The data space is accessible using the SLD, SST, SIN and SOUT instructions. The address pointer can be stored in the Pointer Register using the SLDPR instruction.

14.4.1 Message format

Each message comprises an instruction followed by one or more byte operands. The instruction is always sent by the external programmer. Depending on the instruction all the following operands are sent either by the external programmer or by the TPI.

The messages can be categorized in two types based on the instruction, as follows:

- Write messages. A write message is a request to write data. The write message is sent entirely by the external programmer. This message type is used with the SSTCS, SST, STPR, SOUT and SKEY instructions.
- Read messages. A read message is a request to read data. The TPI reacts to the request by sending the byte operands. This message type is used with the SLDCS, SLD and SIN instructions.

All the instructions except the SKEY instruction require the instruction to be followed by one byte operand. The SKEY instruction requires 8 byte operands. For more information, see the TPI instruction set on page 102.

14.4.2 Exception Handling and Synchronisation

Several situations are considered exceptions from normal operation of the TPI. When the TPI physical layer is in receive mode, these exceptions are:

- The TPI physical layer detects a parity error.
- The TPI physical layer detects a frame error.
- The TPI physical layer recognizes a BREAK character.

When the TPI physical layer is in transmit mode, the possible exceptions are:

- The TPI physical layer detects a data collision.

All these exceptions are signalized to the TPI access layer. The access layer responds to an exception by aborting any on-going operation and enters the error state. The access layer will stay in the error state until a BREAK character has been received, after which it is taken back to its default state. As a consequence, the external programmer can always synchronize the protocol by simply transmitting two successive BREAK characters.

14.5 Instruction Set

The TPI has a compact instruction set that is used to access the TPI Control and Status Space (CSS) and the data space. The instructions allow the external programmer to access the TPI, the NVM Controller and the NVM memories. All instructions except SKEY require one byte operand following the instruction. The SKEY instruction is followed by 8 data bytes. All instructions are byte-sized.

The TPI instruction set is summarised in Table 14-1.

Table 14-1. Instruction Set Summary

Mnemonic	Operand	Description	Operation
SLD	data, PR	Serial LoaD from data space using indirect addressing	data ← DS[PR]
SLD	data, PR+	Serial LoaD from data space using indirect addressing and post-increment	data ← DS[PR] PR ← PR+1
SST	PR, data	Serial STore to data space using indirect addressing	DS[PR] ← data
SST	PR+, data	Serial STore to data space using indirect addressing and post-increment	DS[PR] ← data PR ← PR+1
SSTPR	PR, a	Serial STore to Pointer Register using direct addressing	PR[a] ← data
SIN	data, a	Serial IN from data space	data ← I/O[a]
SOUT	a, data	Serial OUT to data space	I/O[a] ← data
SLDCS	data, a	Serial LoaD from Control and Status space using direct addressing	data ← CSS[a]
SSTCS	a, data	Serial STore to Control and Status space using direct addressing	CSS[a] ← data
SKEY	Key, {8{data}}	Serial KEY	Key ← {8{data}}

14.5.1 SLD - Serial LoaD from data space using indirect addressing

The SLD instruction uses indirect addressing to load data from the data space to the TPI physical layer shift-register for serial read-out. The data space location is pointed by the Pointer Register (PR), where the address must have been stored before data is accessed. The Pointer Register is either left unchanged by the operation, or post-incremented, as shown in Table 14-2.

Table 14-2. The Serial Load from Data Space (SLD) Instruction

Operation	Opcode	Remarks	Register
data ← DS[PR]	0010 0000	PR ← PR	Unchanged
data ← DS[PR]	0010 0100	PR ← PR + 1	Post increment

14.5.2 SST - Serial STore to data space using indirect addressing

The SST instruction uses indirect addressing to store into data space the byte that is shifted into the physical layer shift register. The data space location is pointed by the Pointer Register (PR), where the address must have been stored before the operation. The Pointer Register can be either left unchanged by the operation, or it can be post-incremented, as shown in Table 14-3.

Table 14-3. The Serial Store to Data Space (SLD) Instruction

Operation	Opcode	Remarks	Register
DS[PR] ← data	0110 0000	PR ← PR	Unchanged
DS[PR] ← data	0110 0100	PR ← PR + 1	Post increment

14.5.3 SSTPR - Serial STore to Pointer Register

The SSTPR instruction stores the data byte that is shifted into the physical layer shift register to the Pointer Register (PR). The address bit of the instruction specifies which byte of the Pointer Register is accessed, as shown in Table 14-4.

Table 14-4. The Serial Store to Pointer Register (SSTPR) Instruction

Operation	Opcode	Remarks
PR[a] ← data	0110 100a	Bit ‘a’ addresses Pointer Register byte

14.5.4 SIN - Serial IN from i/o space using direct addressing

The SIN instruction loads data byte from the I/O space to the shift register of the physical layer for serial read-out. The instuction uses direct addressing, the address consisting of the 6 address bits of the instruction, as shown in Table 14-5.

Table 14-5. The Serial IN from i/o space (SIN) Instruction

Operation	Opcode	Remarks
data ← I/O[a]	0aa1 aaaa	Bits marked ‘a’ form the direct, 6-bit addres

14.5.5 SOUT - Serial OUT to i/o space using direct addressing

The SOUT instruction stores the data byte that is shifted into the physical layer shift register to the I/O space. The instruction uses direct addressing, the address consisting of the 6 address bits of the instruction, as shown in Table 14-6.

Table 14-6. The Serial OUT to i/o space (SOUT) Instruction

Operation	Opcode	Remarks
I/O[a] ← data	1aa1 aaaa	Bits marked ‘a’ form the direct, 6-bit addres

14.5.6 SLDCS - Serial LoaD data from Control and Status space using direct addressing

The SLDCS instruction loads data byte from the TPI Control and Status Space to the TPI physical layer shift register for serial read-out. The SLDCS instruction uses direct addressing, the direct address consisting of the 4 address bits of the instruction, as shown in Table 14-7.

Table 14-7. The Serial Load Data from Control and Status space (SLDCS) Instruction

Operation	Opcode	Remarks
data ← CSS[a]	1000 aaaa	Bits marked ‘a’ form the direct, 4-bit addres

14.5.7 SSTCS - Serial STore data to Control and Status space using direct addressing

The SSTCS instruction stores the data byte that is shifted into the TPI physical layer shift register to the TPI Control and Status Space. The SSTCS instruction uses direct addressing, the direct address consisting of the 4 address bits of the instruction, as shown in Table 14-8.

Table 14-8. The Serial STore data to Control and Status space (SSTCS) Instruction

Operation	Opcode	Remarks
CSS[a] ← data	1100 aaaa	Bits marked ‘a’ form the direct, 4-bit addres

14.5.8 SKEY - Serial KEY signaling

The SKEY instruction is used to signal the activation key that enables NVM programming. The SKEY instruction is followed by the 8 data bytes that includes the activation key, as shown in Table 14-9.

Table 14-9. The Serial KEY signaling (SKEY) Instruction

Operation	Opcode	Remarks
Key ← {8[data]}	1110 0000	Data bytes follow after instruction

14.6 Accessing the Non-Volatile Memory Controller

By default, NVM programming is not enabled. In order to access the NVM Controller and be able to program the non-volatile memories, a unique key must be sent using the SKEY instruction. The 64-bit key that will enable NVM programming is given in Table 14-10.

Table 14-10. Enable Key for Non-Volatile Memory Programming

Key	Value
NVM Program Enable	0x1289AB45CDD888FF

After the key has been given, the Non-Volatile Memory Enable (NVMEN) bit in the TPI Status Register (TPISR) must be polled until the Non-Volatile memory has been enabled.

NVM programming is disabled by writing a logical zero to the NVMEN bit in TPISR.

14.7 Control and Status Space Register Descriptions

The control and status registers of the Tiny Programming Interface are mapped in the Control and Status Space (CSS) of the interface. These registers are not part of the I/O register map and are accessible via SLDCS and SSTCS instructions, only. The control and status registers are directly involved in configuration and status monitoring of the TPI.

A summary of CSS registers is shown in Table 14-11.

Table 14-11. Summary of Control and Status Registers

Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0F	TPIIR	Tiny Programming Interface Identification Code							
0x0E ... 0x03	Reserved	–	–	–	–	–	–	–	–
0x02	TPIPCR	–	–	–	–	–	GT2	GT1	GT0
0x01	Reserved	–	–	–	–	–	–	–	–
0x00	TPISR	–	–	–	–	–	–	NVMEN	–

14.7.1 TPIIR – Tiny Programming Interface Identification Register

Bit	7	6	5	4	3	2	1	0	
CSS: 0x0F	Programming Interface Identification Code								TPIIR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – TPIIC: Tiny Programming Interface Identification Code**

These bits give the identification code for the Tiny Programming Interface. The code can be used be the external programmer to identify the TPI. The identification code of the Tiny Programming Interface is shown in Table 14-12..

Table 14-12. Identification Code for Tiny Programming Interface

Code	Value
Interface Identification	0x80

14.7.2 TPIPCR – Tiny Programming Interface Physical Layer Control Register

Bit	7	6	5	4	3	2	1	0	
CSS: 0x02	–	–	–	–	–	GT2	GT1	GT0	TPIPCR
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 and will always read zero.

- **Bits 2:0 – GT[2:0]: Guard Time**

These bits specify the number of additional IDLE bits that are inserted to the idle time when changing from reception mode to transmission mode. Additional delays are not inserted when changing from transmission mode to reception.

The total idle time when changing from reception to transmission mode is Guard Time plus two IDLE bits. Table 14-13 shows the available Guard Time settings.

Table 14-13. Guard Time Settings

GT2	GT1	GT0	Guard Time (Number of IDLE bits)
0	0	0	+128 (default value)
0	0	1	+64
0	1	0	+32
0	1	1	+16
1	0	0	+8
1	0	1	+4
1	1	0	+2
1	1	1	+0

The default Guard Time is 128 IDLE bits. To speed up the communication, the Guard Time should be set to the shortest safe value.

14.7.3 TPISR – Tiny Programming Interface Status Register

Bit	7	6	5	4	3	2	1	0	
CSS: 0x00	–	–	–	–	–	–	NVMEN	–	TPIPCR
Read/Write	R	R	R	R	R	R	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2, 0 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 1 – NVMEN: Non-Volatile Memory Programming Enabled**

NVM programming is enabled when this bit is set. The external programmer can poll this bit to verify the interface has been successfully enabled.

NVM programming is disabled by writing this bit to zero.

15. Memory Programming

15.1 Features

- **Two Embedded Non-Volatile Memories:**
  - **Non-Volatile Memory Lock bits (NVM Lock bits)**
  - **Flash Memory**
- **Four Separate Sections Inside Flash Memory:**
  - **Code Section (Program Memory)**
  - **Signature Section**
  - **Configuration Section**
  - **Calibration Section**
- **Read Access to All Non-Volatile Memories from Application Software**
- **Read and Write Access to Non-Volatile Memories from External programmer:**
  - **Read Access to All Non-Volatile Memories**
  - **Write Access to NVM Lock Bits, Flash Code Section and Flash Configuration Section**
- **External Programming:**
  - **Support for In-System and Mass Production Programming**
  - **Programming Through the Tiny Programming Interface (TPI)**
- **High Security with NVM Lock Bits**

15.2 Overview

The Non-Volatile Memory (NVM) Controller manages all access to the Non-Volatile Memories. The NVM Controller controls all NVM timing and access privileges, and holds the status of the NVM.

During normal execution the CPU will execute code from the code section of the Flash memory (program memory). When entering sleep and no programming operations are active, the Flash memory is disabled to minimize power consumption.

All NVM are mapped to the data memory. Application software can read the NVM from the mapped locations of data memory using load instruction with indirect addressing.

The NVM has only one read port and, therefore, the next instruction and the data can not be read simultaneously. When the application reads data from NVM locations mapped to the data space, the data is read first before the next instruction is fetched. The CPU execution is here delayed by one system clock cycle.

Internal programming operations to NVM have been disabled and the NVM therefore appears to the application software as read-only. Internal write or erase operations of the NVM will not be successful.

The method used by the external programmer for writing the Non-Volatile Memories is referred to as external programming. External programming can be done both in-system or in mass production. See Figure 14-2 on page 97. The external programmer can read and program the NVM via the Tiny Programming Interface (TPI).

In the external programming mode all NVM can be read and programmed, except the signature and the calibration sections which are read-only.

NVM can be programmed at 5V, only.

15.3 Non-Volatile Memories

The ATtiny4/5/9/10 have the following, embedded NVM:

- Non-Volatile Memory Lock Bits
- Flash memory with four separate sections

15.3.1 Non-Volatile Memory Lock Bits

The ATtiny4/5/9/10 provide two Lock Bits, as shown in Table 15-1.

Table 15-1. Lock Bit Byte

Lock Bit	Bit No	Description	Default Value
	7		1 (unprogrammed)
	6		1 (unprogrammed)
	5		1 (unprogrammed)
	4		1 (unprogrammed)
	3		1 (unprogrammed)
	2		1 (unprogrammed)
NVLB2	1	Non-Volatile Lock Bit	1 (unprogrammed)
NVLB1	0	Non-Volatile Lock Bit	1 (unprogrammed)

The Lock Bits can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional security shown in Table 15-2. Lock Bits can be erased to "1" with the Chip Erase command, only.

Table 15-2. Lock Bit Protection Modes

Memory Lock Bits [1]			
Lock Mode	NVLB2 [2]	NVLB1 [2]	Protection Type
1	1	1	No Memory Lock feature Enabled
2	1	0	Further Programming of the Flash memory is disabled in the external programming mode. The configuration section bits are locked in the external programming mode
3	0	0	Further programming and verification of the flash is disabled in the external programming mode. The configuration section bits are locked in the external programming mode

Notes: 1. Program the configuration section bits before programming NVLB1 and NVLB2.
2. "1" means unprogrammed, "0" means programmed

15.3.2 Flash Memory

The embedded Flash memory of ATtiny4/5/9/10 has four separate sections, as shown in Table 15-3 and Table 15-3.

Table 15-3. Number of Words and Pages in the Flash (ATtiny9/10)

Section	Size (Bytes)	Page Size (Words)	Pages	WADDR	PADDR
Code (program memory)	1024	8	64	[3:1]	[9:4]
Configuration	8	8	1	[3:1]	–
Signature (1)	16	8	2	[3:1]	[4:4]
Calibration (1)	8	8	1	[3:1]	–

Notes: 1. This section is read-only.

Table 15-4. Number of Words and Pages in the Flash (ATtiny4/5)

Section	Size (Bytes)	Page Size (Words)	Pages	WADDR	PADDR
Code (program memory)	512	8	32	[3:1]	[9:4]
Configuration	8	8	1	[3:1]	–
Signature (1)	16	8	2	[3:1]	[4:4]
Calibration (1)	8	8	1	[3:1]	–

Notes: 1. This section is read-only.

15.3.3 Configuration Section

ATtiny4/5/9/10 have one configuration byte, which resides in the configuration section. See Table 15-5.

Table 15-5. Configuration bytes

Configuration word address	Configuration word data	
	High byte	Low byte
0x00	Reserved	Configuration Byte 0
0x01 ... 0x07	Reserved	Reserved

Table 15-6 briefly describes the functionality of all configuration bits and how they are mapped into the configuration byte.

Table 15-6. Configuration Byte 0

Bit	Bit Name	Description	Default Value
7:3	–	Reserved	1 (unprogrammed)
2	CKOUT	System Clock Output	1 (unprogrammed)
1	WDTON	Watchdog Timer always on	1 (unprogrammed)
0	RSTDISBL	External Reset disable	1 (unprogrammed)

Configuration bits are not affected by a chip erase but they can be cleared using the configuration section erase command (see “Erasing the Configuration Section” on page 113). Note that configuration bits are locked if Non-Volatile Lock Bit 1 (NVLB1) is programmed.

15.3.3.1 *Latching of Configuration Bits*

All configuration bits are latched either when the device is reset or when the device exits the external programming mode. Changes to configuration bit values have no effect until the device leaves the external programming mode.

15.3.4 Signature Section

The signature section is a dedicated memory area used for storing miscellaneous device information, such as the device signature. Most of this memory section is reserved for internal use, as shown in Table 15-7.

Table 15-7. Signature bytes

Signature word address	Signature word data	
	High byte	**Low byte**
0x00	Device ID 1	Manufacturer ID
0x01	Reserved for internal use	Device ID 2
0x02 ... 0x0F	Reserved for internal use	Reserved for internal use

ATtiny4/5/9/10 have a three-byte signature code, which can be used to identify the device. The three bytes reside in the signature section, as shown in Table 15-7. The signature data for ATtiny4/5/9/10 is given in Table 15-8.

Table 15-8. Signature codes

Part	Signature Bytes		
	Manufacturer ID	**Device ID 1**	**Device ID 2**
ATtiny4	0x1E	0x8F	0x0A
ATtiny5	0x1E	0x8F	0x09
ATtiny9	0x1E	0x90	0x08
ATtiny10	0x1E	0x90	0x03

15.3.5 Calibration Section

ATtiny4/5/9/10 have one calibration byte. The calibration byte contains the calibration data for the internal oscillator and resides in the calibration section, as shown in Table 15-9. During reset, the calibration byte is automatically written into the OSCCAL register to ensure correct frequency of the calibrated internal oscillator.

Table 15-9. Calibration byte

Calibration word address	Calibration word data	
	High byte	**Low byte**
0x00	Reserved	Internal oscillator calibration value
0x01 ... 0x07	Reserved	Reserved

15.3.5.1 *Latching of Calibration Value*

To ensure correct frequency of the calibrated internal oscillator the calibration value is automatically written into the OSCCAL register during reset.

15.4 Accessing the NVM

NVM lock bits, and all Flash memory sections are mapped to the data space as shown in Figure 5-1 on page 15. The NVM can be accessed for read and programming via the locations mapped in the data space.

The NVM Controller recognises a set of commands that can be used to instruct the controller what type of programming task to perform on the NVM. Commands to the NVM Controller are issued via the NVM Command Register. See "NVMCMD - Non-Volatile Memory Command Register" on page 115. After the selected command has been loaded, the operation is started by writing data to the NVM locations mapped to the data space.

When the NVM Controller is busy performing an operation it will signal this via the NVM Busy Flag in the NVM Control and Status Register. See "NVMCSR - Non-Volatile Memory Control and Status Register" on page 115. The NVM Command Register is blocked for write access as long as the busy flag is active. This is to ensure that the current command is fully executed before a new command can start.

Programming any part of the NVM will automatically inhibit the following operations:

- All programming to any other part of the NVM
- All reading from any NVM location

ATtiny4/5/9/10 support only external programming. Internal programming operations to NVM have been disabled, which means any internal attempt to write or erase NVM locations will fail.

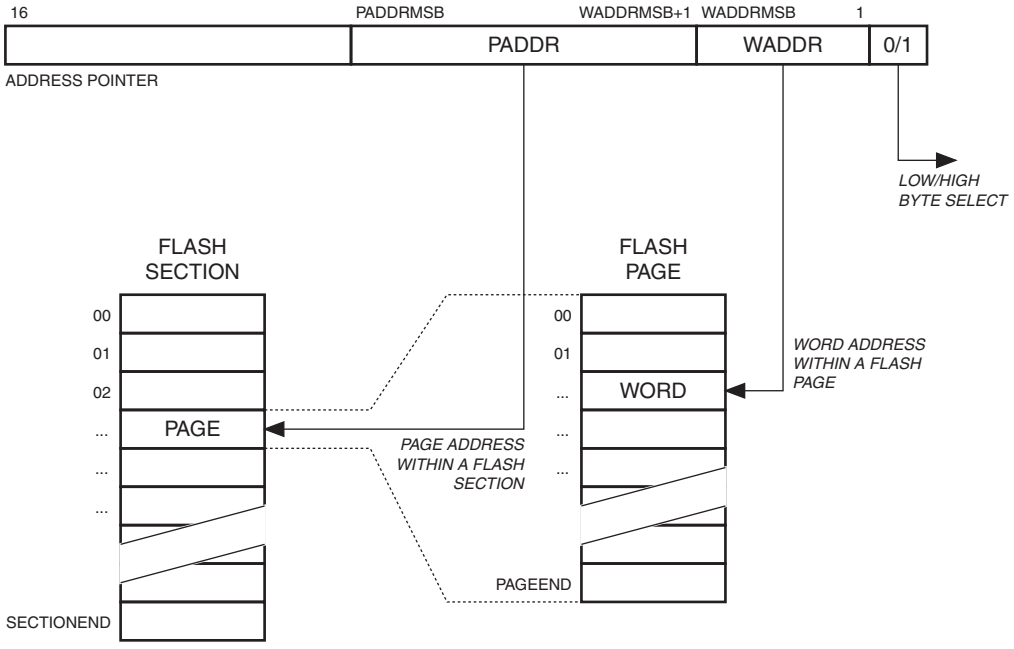
15.4.1 Addressing the Flash

The data space uses byte accessing but since the Flash sections are accessed as words and organized in pages, the byte-address of the data space must be converted to the word-address of the Flash section. This is illustrated in Figure 15-1. Also, see Table 15-3 on page 109.

The most significant bits of the data space address select the NVM Lock bits or the Flash section mapped to the data memory. The word address within a page (WADDR) is held by bits [WADDRMSB:1], and the page address (PADDR) by bits [PADDRMSB:WADDRMSB+1]. Together, PADDR and WADDR form the absolute address of a word in the Flash section.

The least significant bit of the Flash section address is used to select the low or high byte of the word.

Figure 15-1. Addressing the Flash Memory



15.4.2 Reading the Flash

The Flash can be read from the data memory mapped locations one byte at a time. For read operations, the least significant bit (bit 0) is used to select the low or high byte in the word address. If this bit is zero, the low byte is read, and if it is one, the high byte is read.

15.4.3 Programming the Flash

The Flash can be written word-by-word. Before writing a Flash word, the Flash target location must be erased. Writing to an un-erased Flash word will corrupt its content.

The Flash is word-accessed for writing, and the data space uses byte-addressing to access Flash that has been mapped to data memory. It is therefore important to write the word in the correct order to the Flash, namely low bytes before high bytes. First, the low byte is written to the temporary buffer. Then, writing the high byte latches both the high byte and the low byte into the Flash word buffer, starting the write operation to Flash.

The Flash erase operations can only performed for the entire Flash sections.

The Flash programming sequence is as follows:

1. Perform a Flash section erase or perform a Chip erase
2. Write the Flash section word by word

15.4.3.1 Chip Erase

The Chip Erase command will erase the entire code section of the Flash memory and the NVM Lock Bits. For security reasons, the NVM Lock Bits are not reset before the code section has been completely erased. Configuration, Signature and Calibration sections are not changed.

Before starting the Chip erase, the NVMCMD register must be loaded with the CHIP_ERASE command. To start the erase operation a dummy byte must be written into the high byte of a word location that resides inside the Flash code section. The NVMBSY bit remains set until erasing has been completed. While the Flash is being erased neither Flash buffer loading nor Flash reading can be performed.

The Chip Erase can be carried out as follows:

1. Write the CHIP_ERASE command to the NVMCMD register
2. Start the erase operation by writing a dummy byte to the high byte of any word location inside the code section
3. Wait until the NVMBSY bit has been cleared

15.4.3.2 *Erasing the Code Section*

The algorithm for erasing all pages of the Flash code section is as follows:

1. Write the SECTION_ERASE command to the NVMCMD register
2. Start the erase operation by writing a dummy byte to the high byte of any word location inside the code section
3. Wait until the NVMBSY bit has been cleared

15.4.3.3 *Writing a Code Word*

The algorithm for writing a word to the code section is as follows:

1. Write the WORD_WRITE command to the NVMCMD register
2. Write the low byte of the data into the low byte of a word location
3. Write the high byte of the data into the high byte of the same word location. This will start the Flash write operation
4. Wait until the NVMBSY bit has been cleared

15.4.3.4 *Erasing the Configuration Section*

The algorithm for erasing the Configuration section is as follows:

1. Write the SECTION_ERASE command to the NVMCMD register
2. Start the erase operation by writing a dummy byte to the high byte of any word location inside the configuration section
3. Wait until the NVMBSY bit has been cleared

15.4.3.5 *Writing a Configuration Word*

The algorithm for writing a Configuration word is as follows.

1. Write the WORD_WRITE command to the NVMCMD register
2. Write the low byte of the data to the low byte of a configuration word location
3. Write the high byte of the data to the high byte of the same configuration word location. This will start the Flash write operation.
4. Wait until the NVMBSY bit has been cleared

15.4.4 Reading NVM Lock Bits

The Non-Volatile Memory Lock Byte can be read from the mapped location in data memory.

15.4.5 Writing NVM Lock Bits

The algorithm for writing the Lock bits is as follows.

1. Write the WORD_WRITE command to the NVMCMD register.
2. Write the lock bits value to the Non-Volatile Memory Lock Byte location. This is the low byte of the Non-Volatile Memory Lock Word.
3. Start the NVM Lock Bit write operation by writing a dummy byte to the high byte of the NVM Lock Word location.
4. Wait until the NVMBSY bit has been cleared.

15.5 Self programming

The ATtiny4/5/9/10 don't support internal programming.

15.6 External Programming

The method for programming the Non-Volatile Memories by means of an external programmer is referred to as external programming. External programming can be done both in-system or in mass production.

The Non-Volatile Memories can be externally programmed via the Tiny Programming Interface (TPI). For details on the TPI, see “Programming interface” on page 96. Using the TPI, the external programmer can access the NVM control and status registers mapped to I/O space and the NVM memory mapped to data memory space.

15.6.1 Entering External Programming Mode

The TPI must be enabled before external programming mode can be entered. The following procedure describes, how to enter the external programming mode after the TPI has been enabled:

1. Make a request for enabling NVM programming by sending the NVM memory access key with the SKEY instruction.
2. Poll the status of the NVMEN bit in TPISR until it has been set.

Refer to the Tiny Programming Interface description on page 96 for more detailed information of enabling the TPI and programming the NVM.

15.6.2 Exiting External Programming Mode

Clear the NVM enable bit to disable NVM programming, then release the $\overline{\text{RESET}}$ pin.

See NVMEN bit in “TPISR – Tiny Programming Interface Status Register” on page 106.

15.7 Register Description

15.7.1 NVMCSR - Non-Volatile Memory Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x32	NVMBSY	–	–	–	–	–	–	–	NVMCSR
Read/Write	R/W	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - NVMBSY: Non-Volatile Memory Busy**

This bit indicates the NVM memory (Flash memory and Lock Bits) is busy, being programmed. This bit is set when a program operation is started, and it remains set until the operation has been completed.

- **Bit 6:0 - Res: Reserved Bits**

These bits are reserved and will always be read as zero.

15.7.2 NVMCMD - Non-Volatile Memory Command Register

Bit	7	6	5	4	3	2	1	0	
0x33	–	–	NVMCMD[5:0]						NVMCMD
Read/Write	R	R	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 - Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 5:0 - NVMCMD[5:0]: Non-Volatile Memory Command**

These bits define the programming commands for the flash, as shown in Table 15-10.

Table 15-10. NVM Programming commands

Operation Type	NVMCMD		Mnemonic	Description
	Binary	Hex		
General	0b000000	0x00	NO_OPERATION	No operation
	0b010000	0x10	CHIP_ERASE	Chip erase
Section	0b010100	0x14	SECTION_ERASE	Section erase
Word	0b011101	0x1D	WORD_WRITE	Word write

16. Electrical Characteristics

16.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: 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.

16.2 DC Characteristics

Table 16-1. DC Characteristics. T_A = -40°C to +85°C

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5		$0.2V_{CC}$ $0.3V_{CC}$	V
V_{IH}	Input High-voltage Except $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.7V_{CC}$(1) $0.6V_{CC}$(1)		V_{CC} +0.5(2)	V
	Input High-voltage $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V to 5.5V	$0.9V_{CC}$(1)		V_{CC} +0.5(2)	V
V_{OL}	Output Low Voltage(3) Except $\overline{\text{RESET}}$ pin(5)	I_{OL} = 10 mA, V_{CC} = 5V I_{OL} = 5 mA, V_{CC} = 3V			0.6 0.5	V
V_{OH}	Output High-voltage(4) Except $\overline{\text{RESET}}$ pin(5)	I_{OH} = -10 mA, V_{CC} = 5V I_{OH} = -5 mA, V_{CC} = 3V	4.3 2.5			V
I_{LIL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)		<0.05	1	µA
I_{LIH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)		<0.05	1	µA
R_{RST}	Reset Pull-up Resistor	Vcc = 5.5V, input low	30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor	Vcc = 5.5V, input low	20		50	kΩ
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2	-50		50	nA

Table 16-1. DC Characteristics. T_A = -40°C to +85°C (Continued)

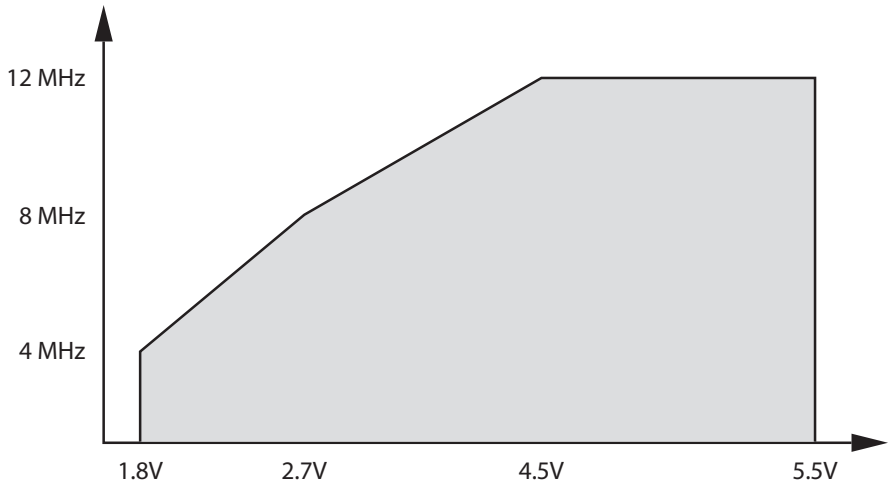
Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{CC}	Power Supply Current[6]	Active 1MHz, V_{CC} = 2V		0.2	0.5	mA
		Active 4MHz, V_{CC} = 3V		0.8	1.5	mA
		Active 8MHz, V_{CC} = 5V		2.7	5	mA
		Idle 1MHz, V_{CC} = 2V		0.02	0.2	mA
		Idle 4MHz, V_{CC} = 3V		0.13	0.5	mA
		Idle 8MHz, V_{CC} = 5V		0.6	1.5	mA
	Power-down mode[7]	WDT enabled, V_{CC} = 3V		4.5	10	µA
		WDT disabled, V_{CC} = 3V		0.15	2	µA

Notes:
1. "Min" means the lowest value where the pin is guaranteed to be read as high.
2. "Max" means the highest value where the pin is guaranteed to be read as low.
3. Although each I/O port can sink more than the test conditions (10 mA at V_{CC} = 5V, 5 mA at V_{CC} = 3V) under steady state conditions (non-transient), the sum of all I_{OL} (for all ports) should not exceed 60 mA. If I_{OL} exceeds the test conditions, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (10 mA at V_{CC} = 5V, 5 mA at V_{CC} = 3V) under steady state conditions (non-transient), the sum of all I_{OH} (for all ports) should not exceed 60 mA. If I_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. The $\overline{\text{RESET}}$ pin must tolerate high voltages when entering and operating in programming modes and, as a consequence, has a weak drive strength as compared to regular I/O pins. See Figure 17-25 on page 135, and Figure 17-26 on page 135.
6. Values are with external clock using methods described in "Minimizing Power Consumption" on page 24. Power Reduction is enabled (PRR = 0xFF) and there is no I/O drive.
7. BOD Disabled.

16.3 Speed Grades

The maximum operating frequency of the device depends on V_{CC} . As shown in Figure 16-1, the relationship between maximum frequency vs. V_{CC} is linear between 1.8V < V_{CC} < 4.5V.

Figure 16-1. Maximum Frequency vs. V_{CC}



16.4 Clock Characteristics

16.4.1 Accuracy of Calibrated Internal Oscillator

It is possible to manually calibrate the internal oscillator to be more accurate than default factory calibration. Note that the oscillator frequency depends on temperature and voltage. Voltage and temperature characteristics can be found in Figure 17-39 on page 142 and Figure 17-40 on page 142.

Table 16-2. Calibration Accuracy of Internal RC Oscillator

Calibration Method	Target Frequency	V_{CC}	Temperature	Accuracy at given Voltage & Temperature[1]
Factory Calibration	8.0 MHz	3V	25°C	±10%
User Calibration	Fixed frequency within: 7.3 – 8.1 MHz	Fixed voltage within: 1.8V – 5.5V	Fixed temp. within: -40°C – 85°C	±1%

Notes: 1. Accuracy of oscillator frequency at calibration point (fixed temperature and fixed voltage).

16.4.2 External Clock Drive

Figure 16-2. External Clock Drive Waveform

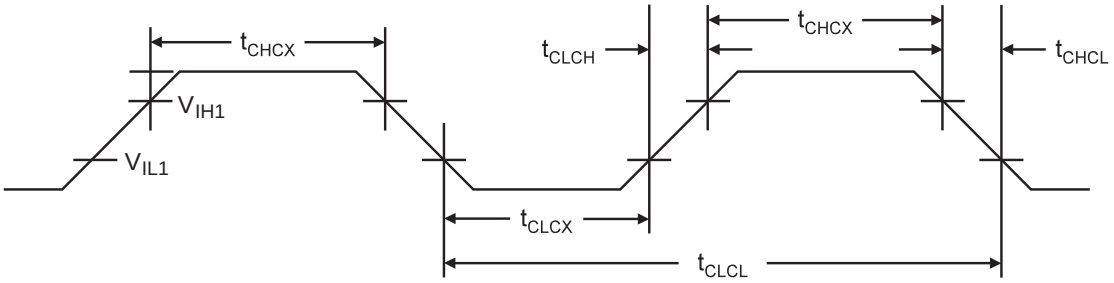


Table 16-3. External Clock Drive Characteristics

Symbol	Parameter	V_{CC} = 1.8 - 5.5V		V_{CC} = 2.7 - 5.5V		V_{CC} = 4.5 - 5.5V		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$1/t_{CLCL}$	Clock Frequency	0	4	0	8	0	12	MHz
t_{CLCL}	Clock Period	250		125		83		ns
t_{CHCX}	High Time	100		50		33		ns
t_{CLCX}	Low Time	100		50		33		ns
t_{CLCH}	Rise Time		2.0		1		0.6	µs
t_{CHCL}	Fall Time		2.0		1		0.6	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2		2	%

16.5 System and Reset Characteristics

Table 16-4. Reset, VLM, and Internal Voltage Characteristics

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.2 V_{CC}		0.9V_{CC}	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin	V_{CC} = 1.8V V_{CC} = 3V V_{CC} = 5V		2000 700 400		ns
t_{TOUT}	Time-out after reset			64	128	ms
V_{HYST}	VLM Hysteresis			50		mV

Note: 1. Values are guidelines, only

16.5.1 Power-On Reset

Table 16-5. Characteristics of Enhanced Power-On Reset. T_A = -40 - 85°C

Symbol	Parameter	Min(1)	Typ(1)	Max(1)	Units
V_{POR}	Release threshold of power-on reset (2)	1.1	1.4	1.6	V
V_{POA}	Activation threshold of power-on reset (3)	0.6	1.3	1.6	V
SR_{ON}	Power-On Slope Rate	0.01			V/ms

Note: 1. Values are guidelines, only
2. Threshold where device is released from reset when voltage is rising
3. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

16.5.2 V_{CC} Level Monitor

Table 16-6. Voltage Level Monitor Thresholds

Parameter	Min	Typ (1)	Max	Units
Trigger level VLM1L	1.1	1.4	1.6	V
Trigger level VLM1H	1.4	1.6	1.8	
Trigger level VLM2	2.0	2.5	2.7	
Trigger level VLM3	3.2	3.7	4.5	
Settling time VMLM2,VLM3 (VLM1H,VLM1L)		5 (50)		µs

Note: 1. Typical values at room temperature

16.6 Analog Comparator Characteristics

Table 16-7. Analog Comparator Characteristics, T_A = -40°C - 85°C

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{AIO}	Input Offset Voltage	V_{CC} = 5V, V_{IN} = V_{CC} / 2		< 10	40	mV
I_{LAC}	Input Leakage Current	V_{CC} = 5V, V_{IN} = V_{CC} / 2	-50		50	nA
t_{APD}	Analog Propagation Delay (from saturation to slight overdrive)	V_{CC} = 2.7V		750		ns
		V_{CC} = 4.0V		500		
	Analog Propagation Delay (large step change)	V_{CC} = 2.7V		100		
		V_{CC} = 4.0V		75		
t_{DPD}	Digital Propagation Delay	V_{CC} = 1.8V - 5.5		1	2	CLK

Note: All parameters are based on simulation results. None are tested in production

16.7 ADC Characteristics (ATtiny5/10, only)

Table 16-8. ADC Characteristics. T = -40°C – 85°C. V_{CC} = 2.5V – 5.5V

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution				8	Bits
	Absolute accuracy (Including INL, DNL, and Quantization, Gain and Offset Errors)	V_{REF} = V_{CC} = 4V, ADC clock = 200 kHz		1.0		LSB
		V_{REF} = V_{CC} = 4V, ADC clock = 200 kHz Noise Reduction Mode		1.0		LSB
	Integral Non-Linearity (INL) (Accuracy after Offset and Gain Calibration)	V_{REF} = V_{CC} = 4V, ADC clock = 200 kHz		1.0		LSB
	Differential Non-linearity (DNL)	V_{REF} = V_{CC} = 4V, ADC clock = 200 kHz		0.5		LSB
	Gain Error	V_{REF} = V_{CC} = 4V, ADC clock = 200 kHz		1.0		LSB
	Offset Error	V_{REF} = V_{CC} = 4V, ADC clock = 200 kHz		1.0		LSB
	Conversion Time	Free Running Conversion	65		260	µs
	Clock Frequency		50		200	kHz
V_{IN}	Input Voltage		GND		V_{REF}	V
	Input Bandwidth			7.7		kHz
R_{AIN}	Analog Input Resistance			100		MΩ
	ADC Conversion Output		0		255	LSB

16.8 Serial Programming Characteristics

Figure 16-3. Serial Programming Timing

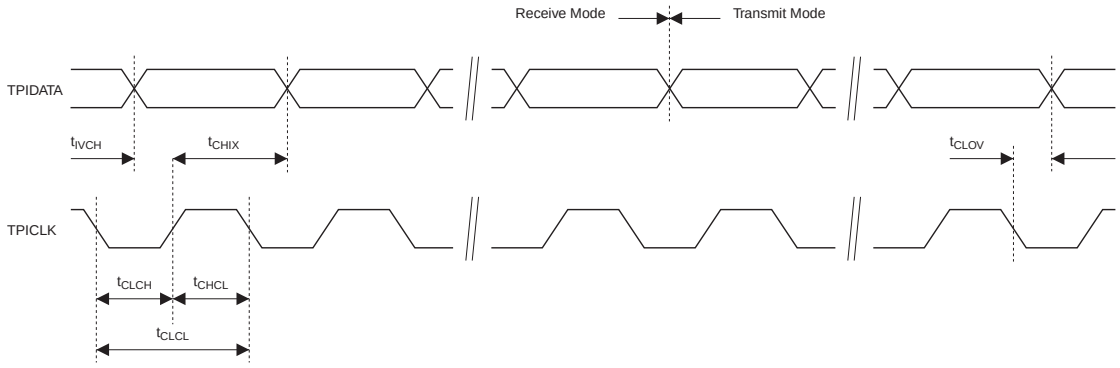


Table 16-9. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Clock Frequency			2	MHz
t_{CLCL}	Clock Period	500			ns
t_{CLCH}	Clock Low Pulse Width	200			ns
t_{CHCH}	Clock High Pulse Width	200			ns
t_{IVCH}	Data Input to Clock High Setup Time	50			ns
t_{CHIX}	Data Input Hold Time After Clock High	100			ns
t_{CLOV}	Data Output Valid After Clock Low Time			200	ns

17. Typical Characteristics

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. During characterisation devices are operated at frequencies higher than test limits but they are not guaranteed to function properly at frequencies higher than the ordering code indicates.

All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. 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.

A sine wave generator with rail-to-rail output is used as clock source but current consumption in Power-Down mode is independent of clock selection. 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.

The current drawn from pins with a capacitive load may be estimated (for one pin) as follows:

$$I_{CP} \approx V_{CC} \times C_L \times f_{SW}$$

where V_{CC} = operating voltage, C_L = load capacitance and f_{SW} = average switching frequency of I/O pin.

17.1 Supply Current of I/O Modules

Tables and formulas below can be used to calculate additional current consumption for the different I/O modules in Active and Idle mode. Enabling and disabling of I/O modules is controlled by the Power Reduction Register. See “Power Reduction Register” on page 24 for details.

Table 17-1. 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
PRTIM0	6.6 uA	40.0 uA	153.0 uA
PRADC (1)	29.6 uA	88.3 uA	333.3 uA

Note: 1. The ADC is available in ATtiny5/10, only

Table 17-2 below can be used for calculating typical current consumption for other supply voltages and frequencies than those mentioned in the Table 17-1 above.

Table 17-2. Additional Current Consumption (percentage) in Active and Idle mode

PRR bit	Current consumption additional to active mode with external clock (see Figure 17-1 and Figure 17-2)	Current consumption additional to idle mode with external clock (see Figure 17-7 and Figure 17-8)
PRTIM0	2.3 %	10.4 %
PRADC (1)	6.7 %	28.8 %

Note: 1. The ADC is available in ATtiny5/10, only

17.2 ATtiny4/5/9/10

17.2.1 Active Supply Current

Figure 17-1. Active Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

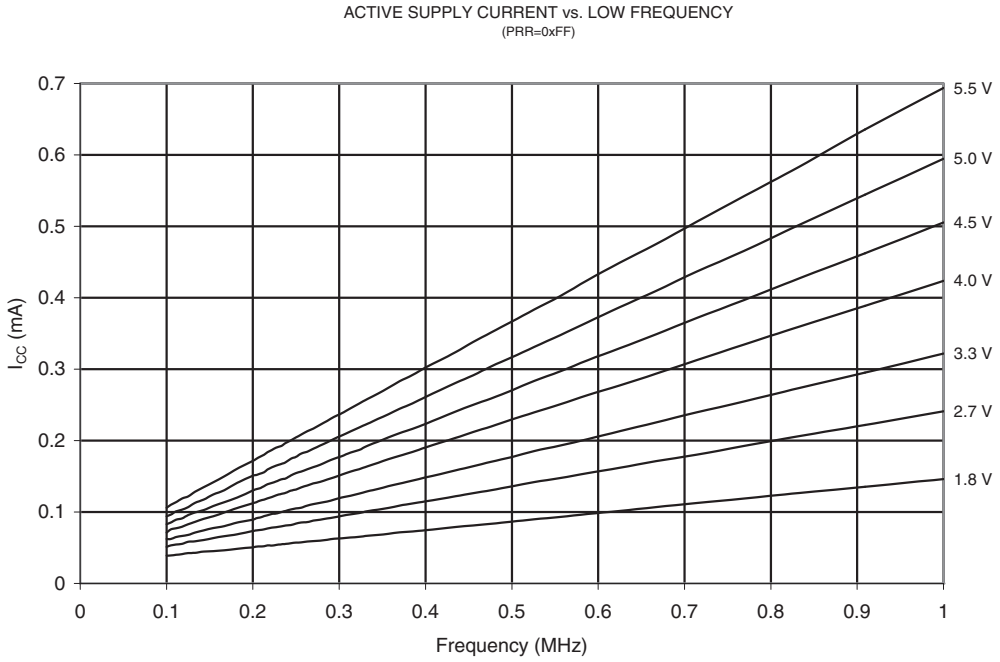


Figure 17-2. Active Supply Current vs. frequency (1 - 12 MHz)

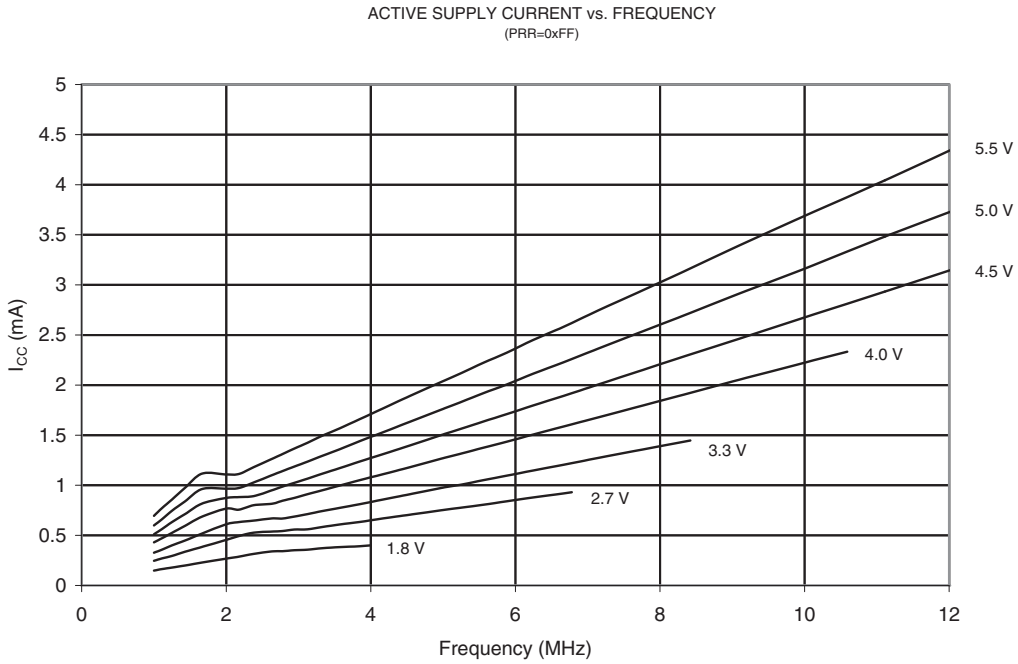


Figure 17-3. Active Supply Current vs. V_{CC} (Internal Oscillator, 8 MHz)

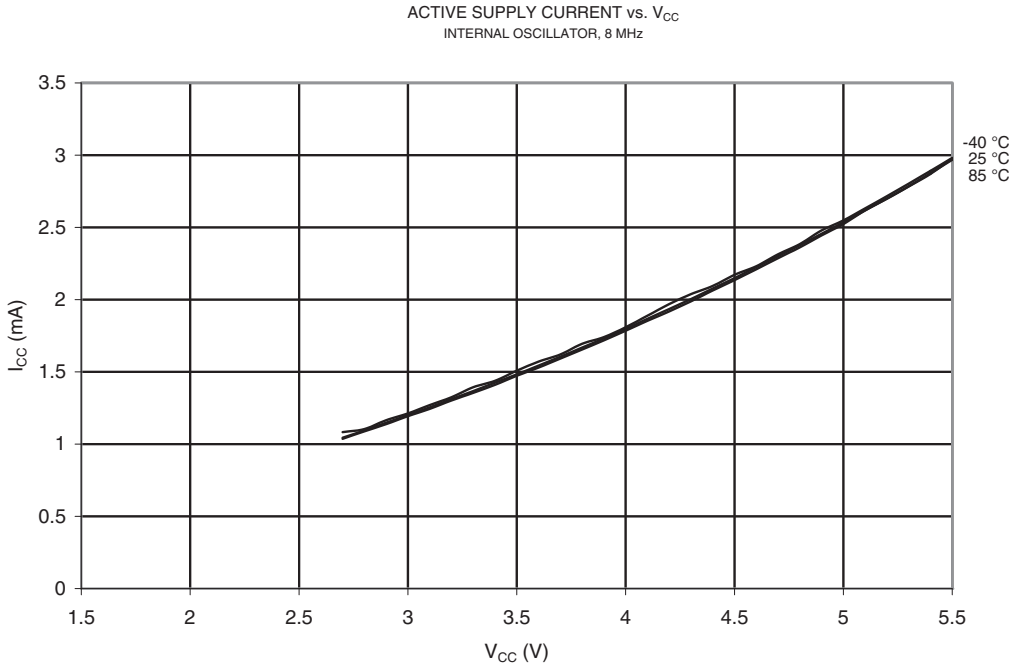


Figure 17-4. Active Supply Current vs. V_{CC} (Internal Oscillator, 1 MHz)

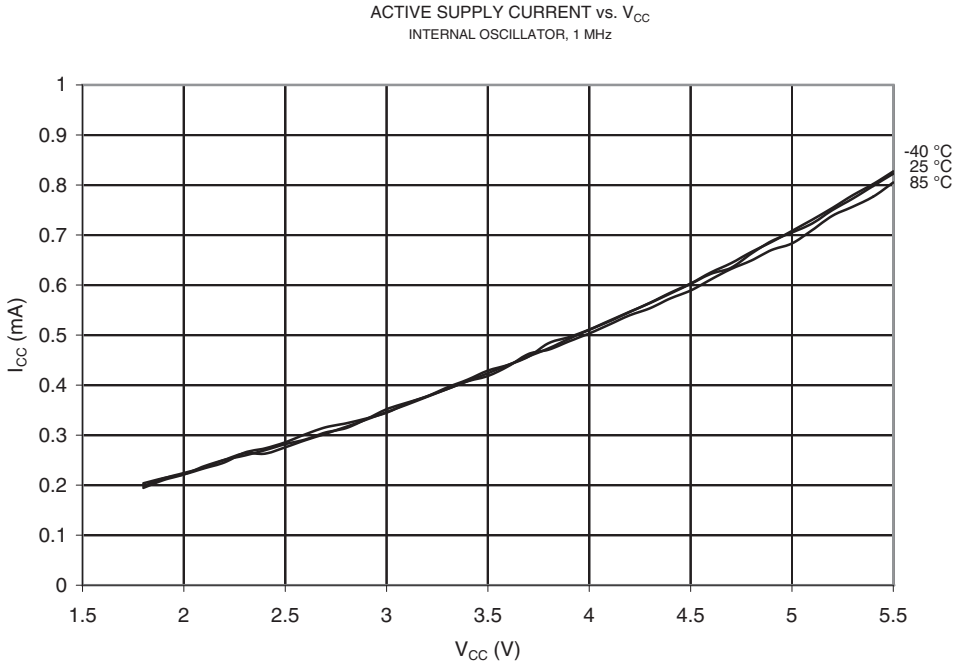


Figure 17-5. Active Supply Current vs. V_{CC} (Internal Oscillator, 128 kHz)

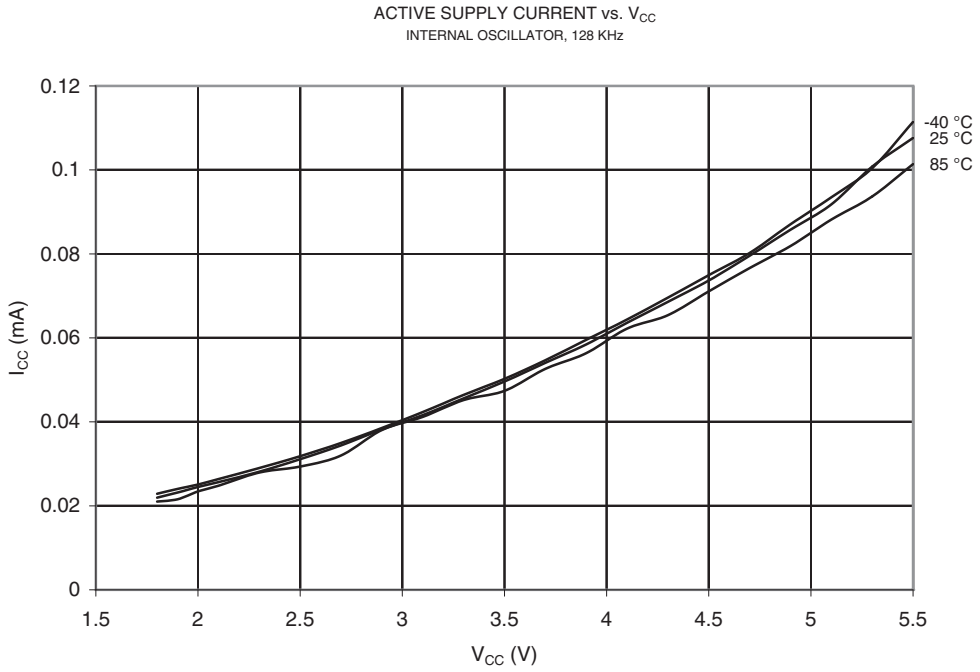
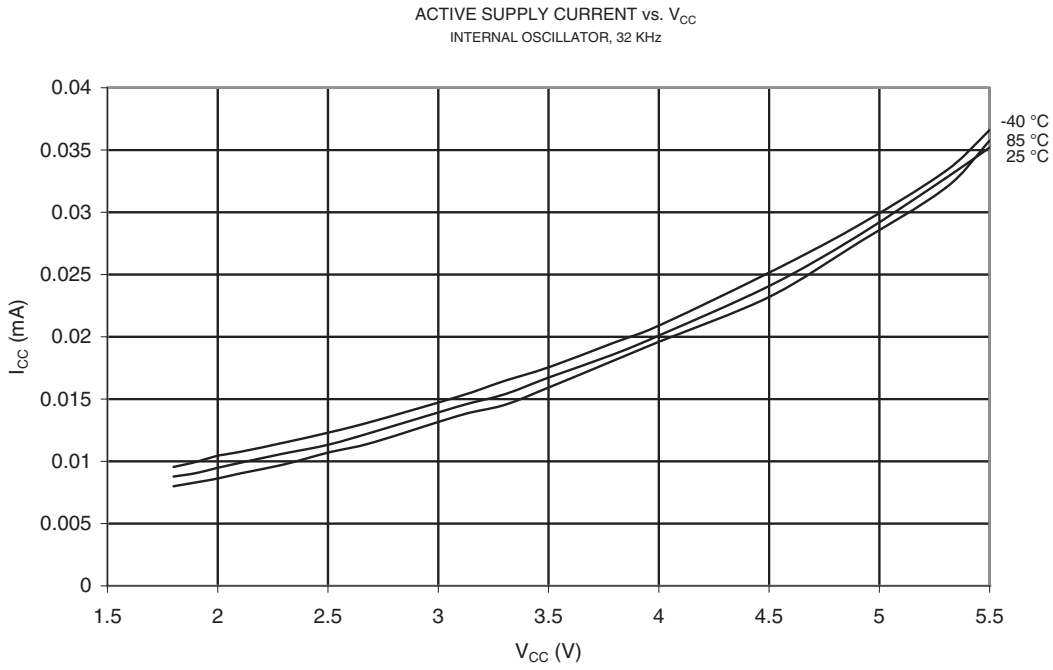


Figure 17-6. Active Supply Current vs. V_{CC} (External Clock, 32 kHz)



17.2.2 Idle Supply Current

Figure 17-7. Idle Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

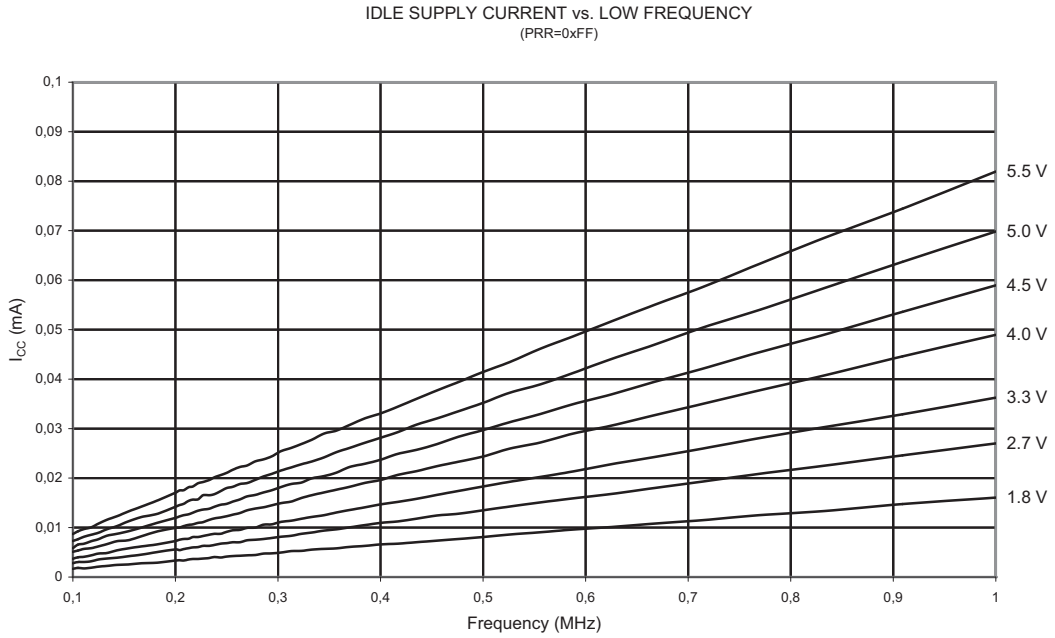


Figure 17-8. Idle Supply Current vs. Frequency (1 - 12 MHz)

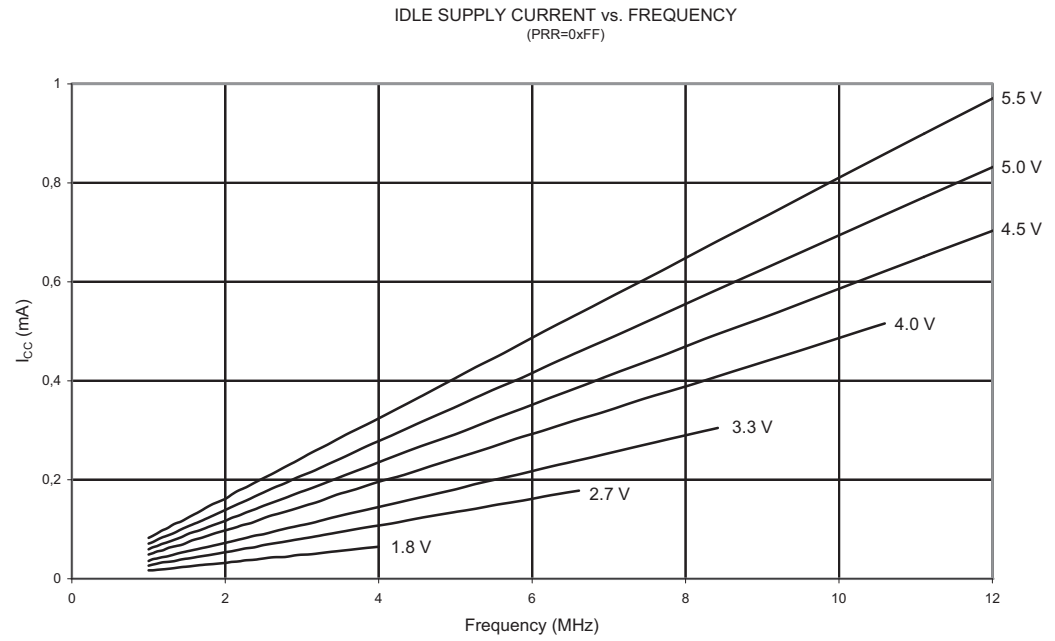


Figure 17-9. Idle Supply Current vs. V_{CC} (Internal Oscillator, 8 MHz)

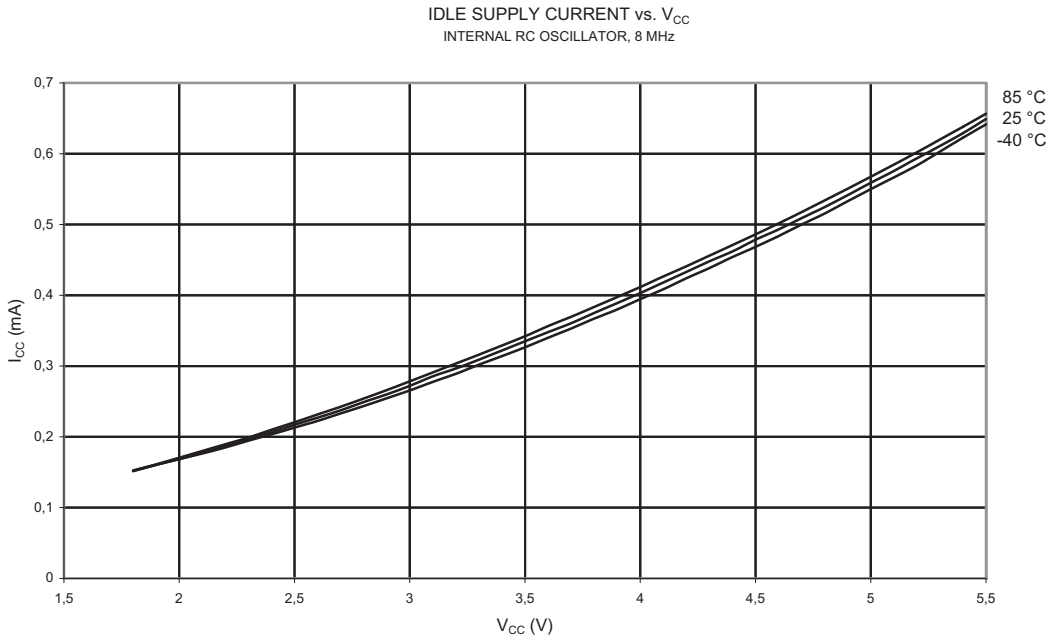
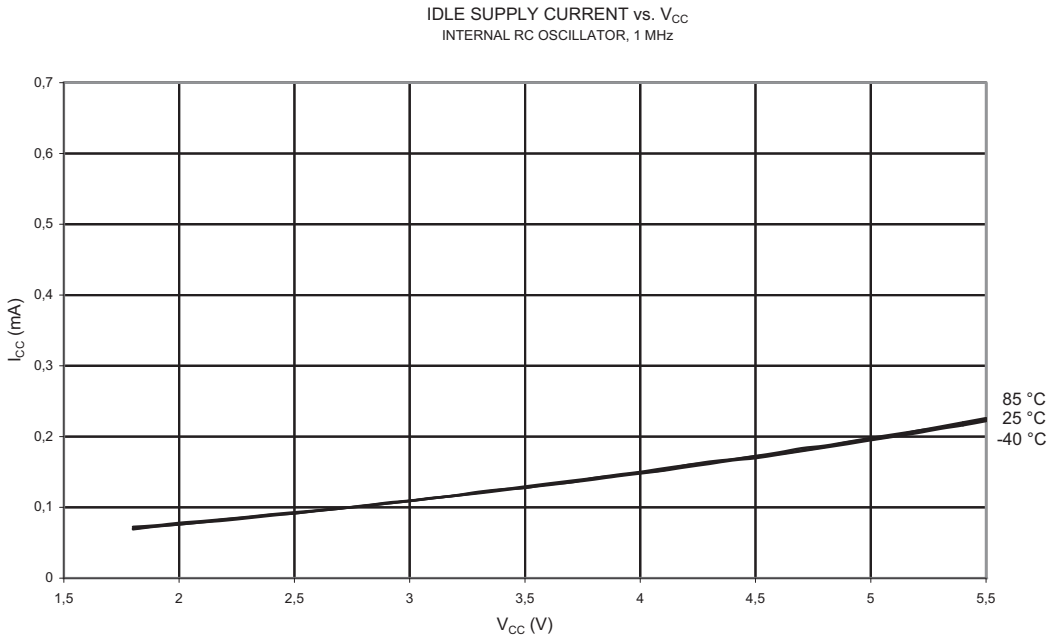


Figure 17-10. Idle Supply Current vs. V_{CC} (Internal Oscillator, 1 MHz)



17.2.3 Power-down Supply Current

Figure 17-11. Power-down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

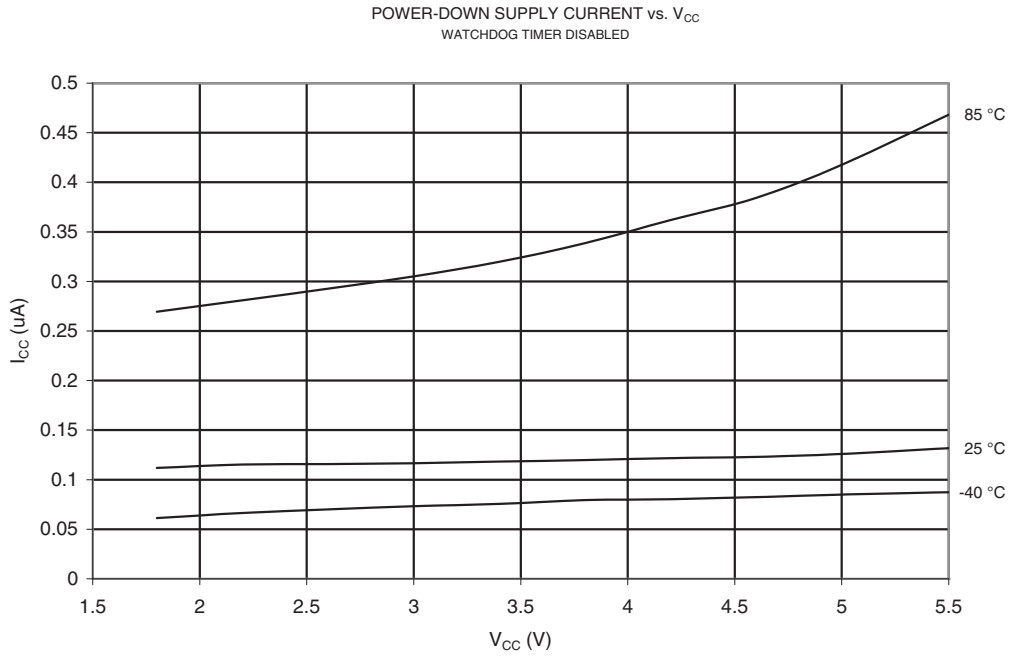
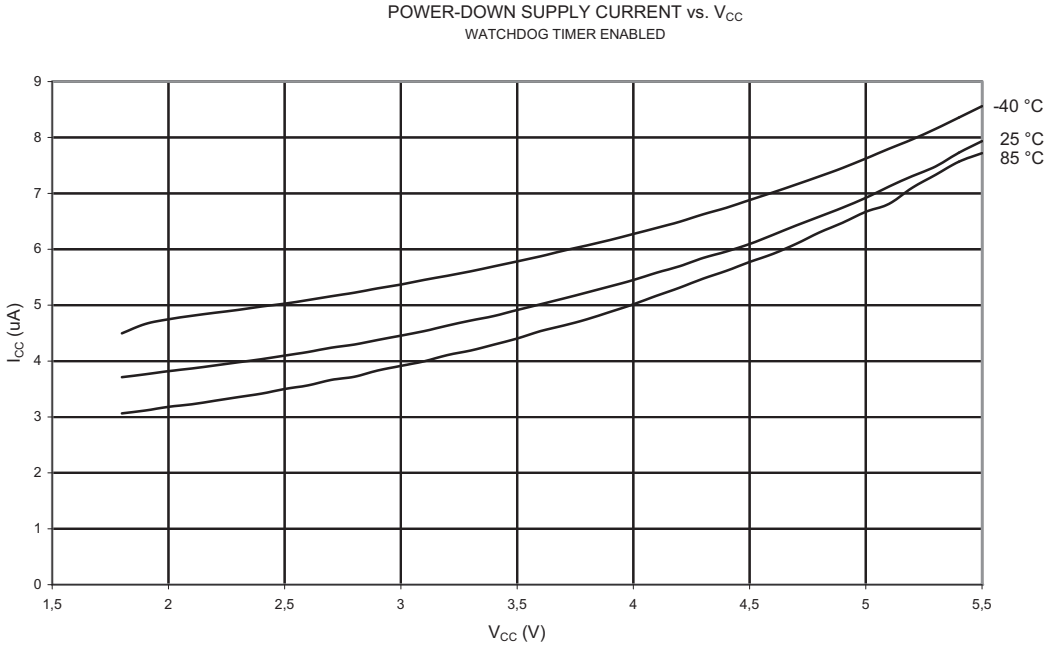


Figure 17-12. Power-down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



17.2.4 Pin Pull-up

Figure 17-13. I/O pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 1.8V)

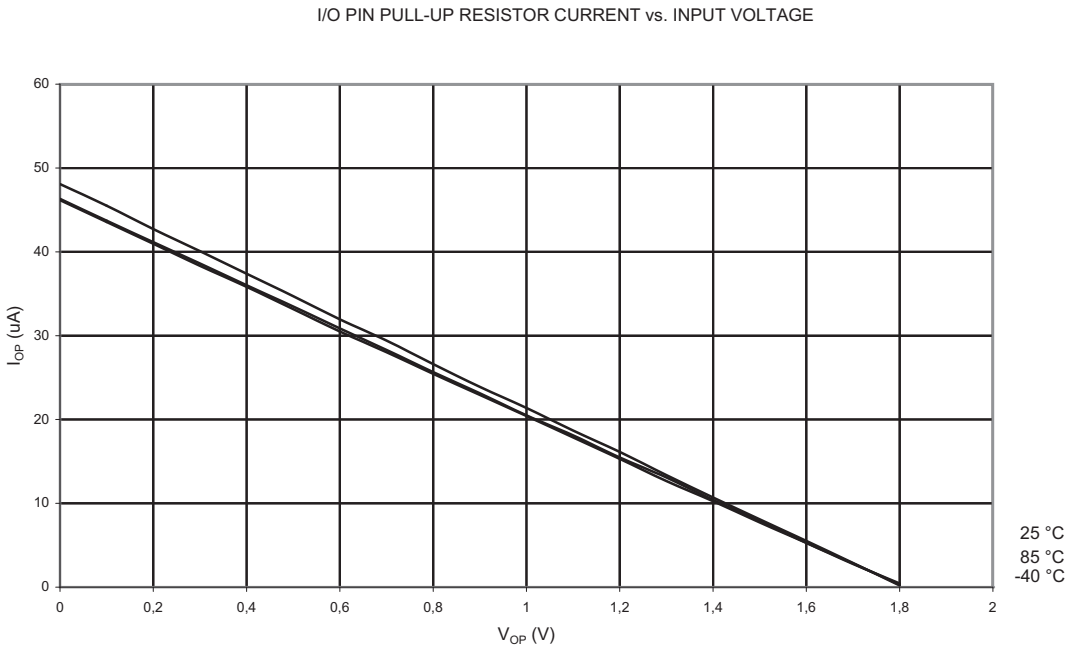


Figure 17-14. I/O Pin Pull-up Resistor Current vs. input Voltage (V_{CC} = 2.7V)

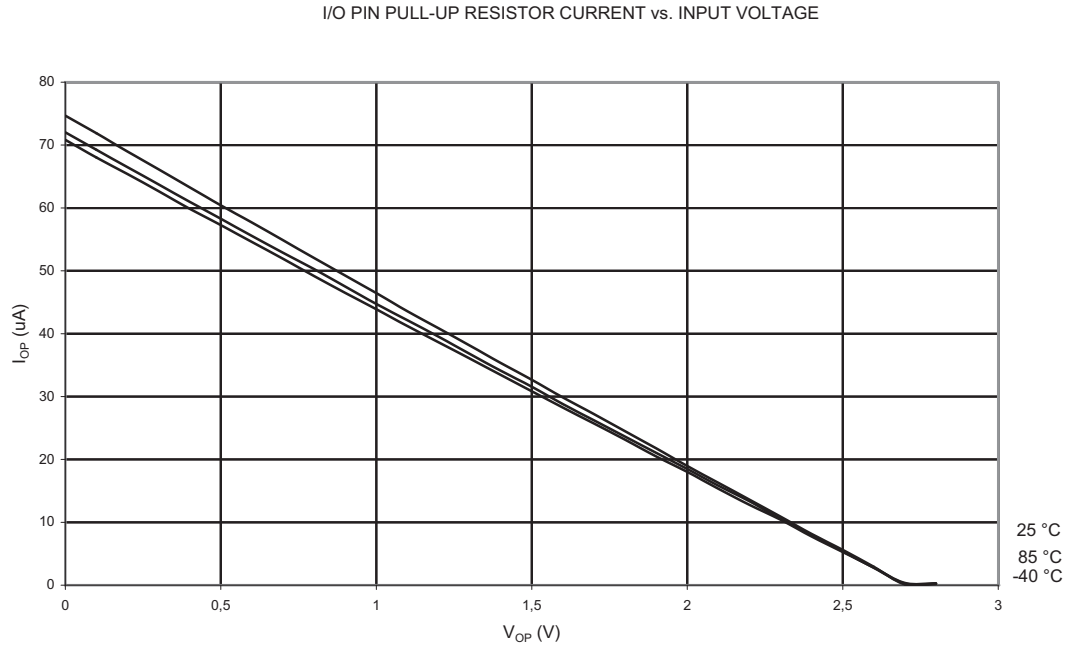


Figure 17-15. I/O pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

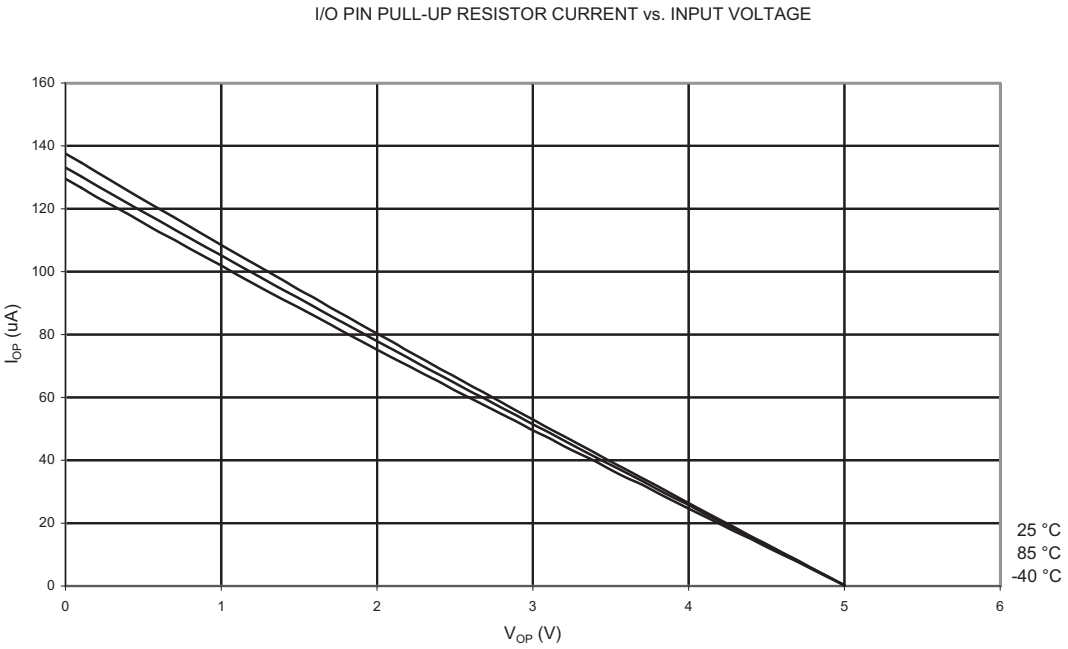


Figure 17-16. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 1.8V)

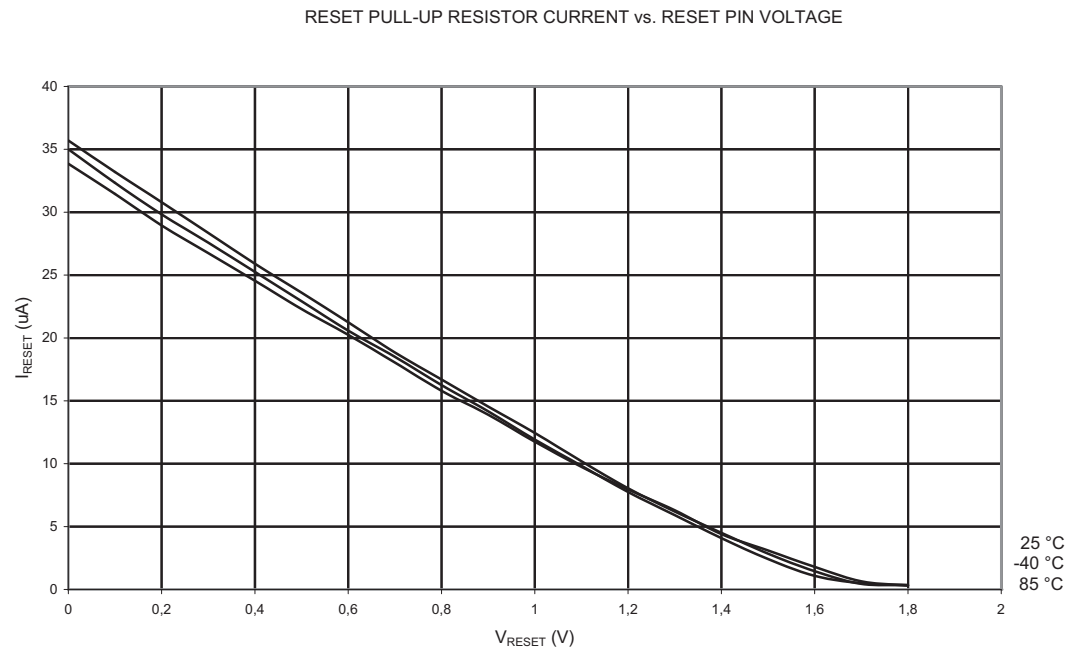


Figure 17-17. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)

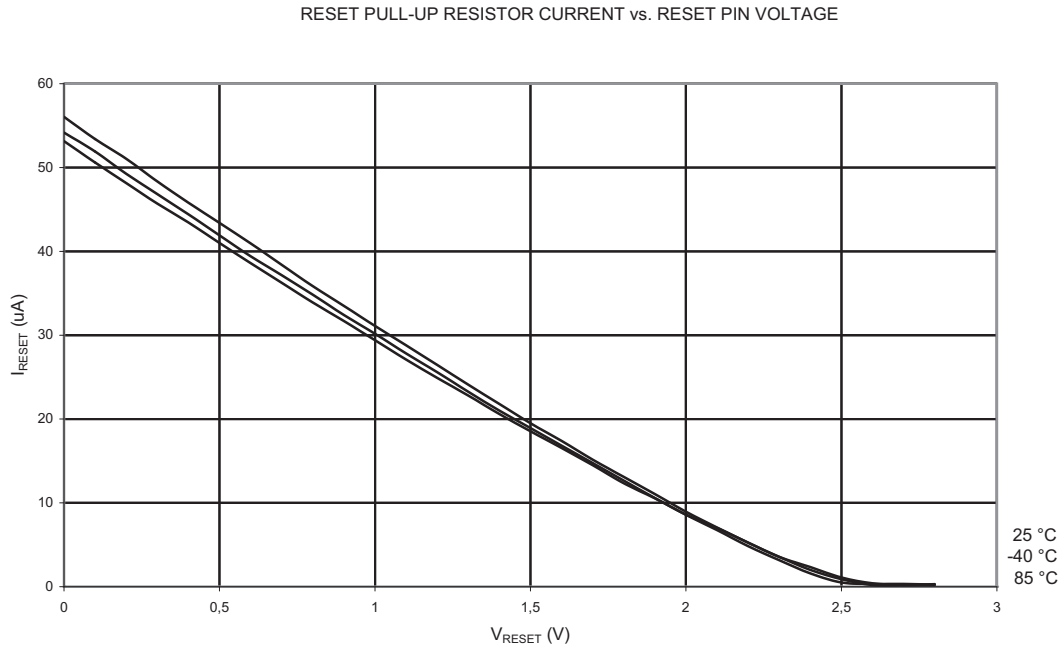
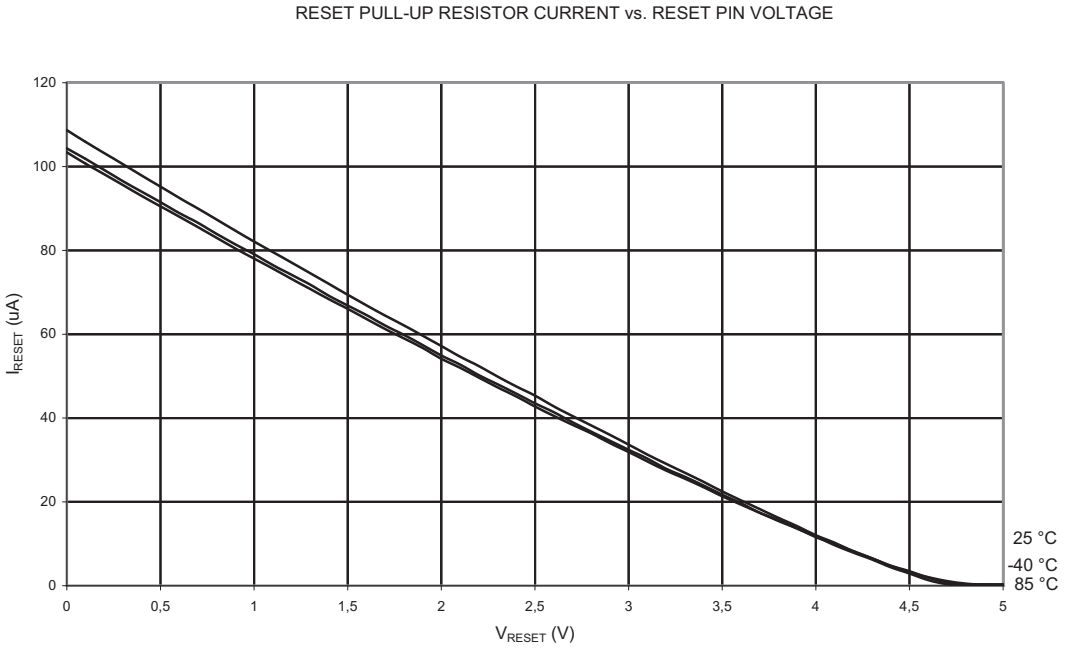


Figure 17-18. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)



17.2.5 Pin Driver Strength

Figure 17-19. I/O Pin Output Voltage vs. Sink Current (V_{CC} = 1.8V)

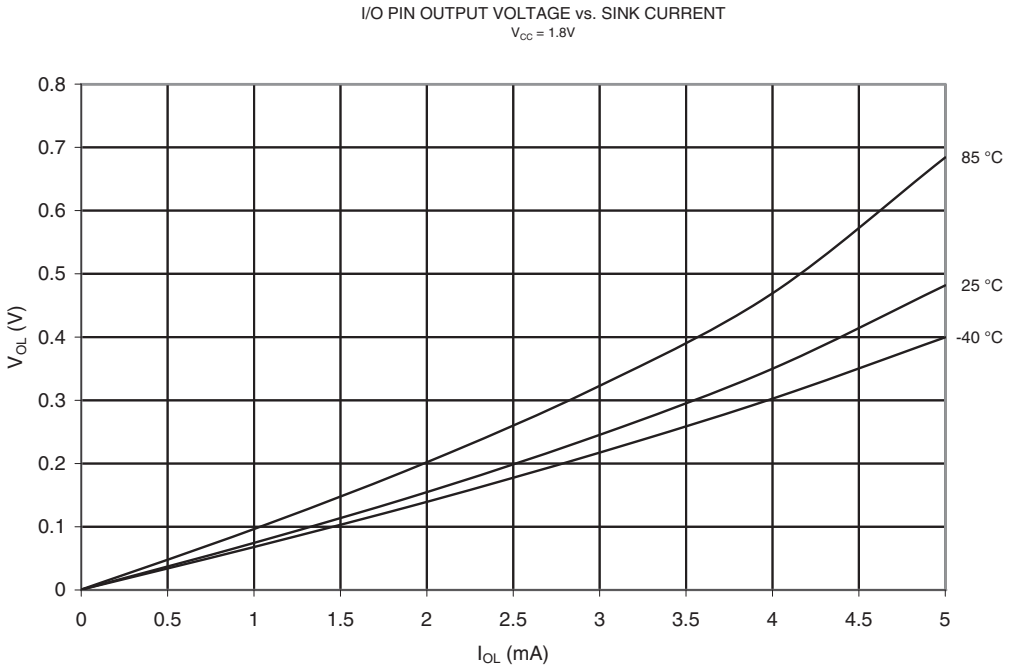


Figure 17-20. I/O Pin Output Voltage vs. Sink Current (V_{CC} = 3V)

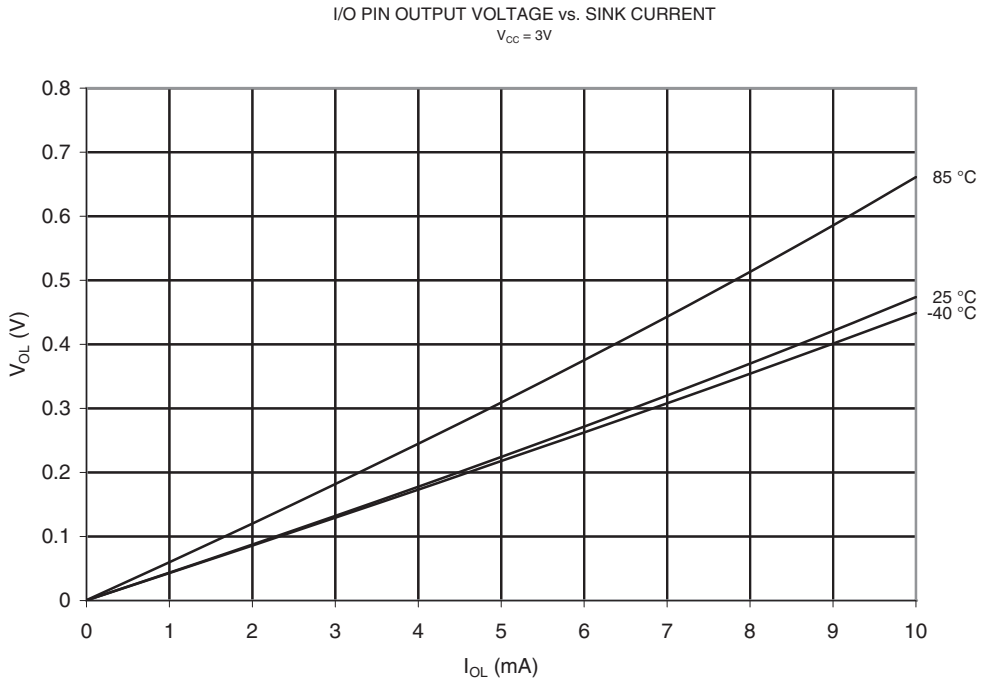


Figure 17-21. I/O pin Output Voltage vs. Sink Current (V_{CC} = 5V)

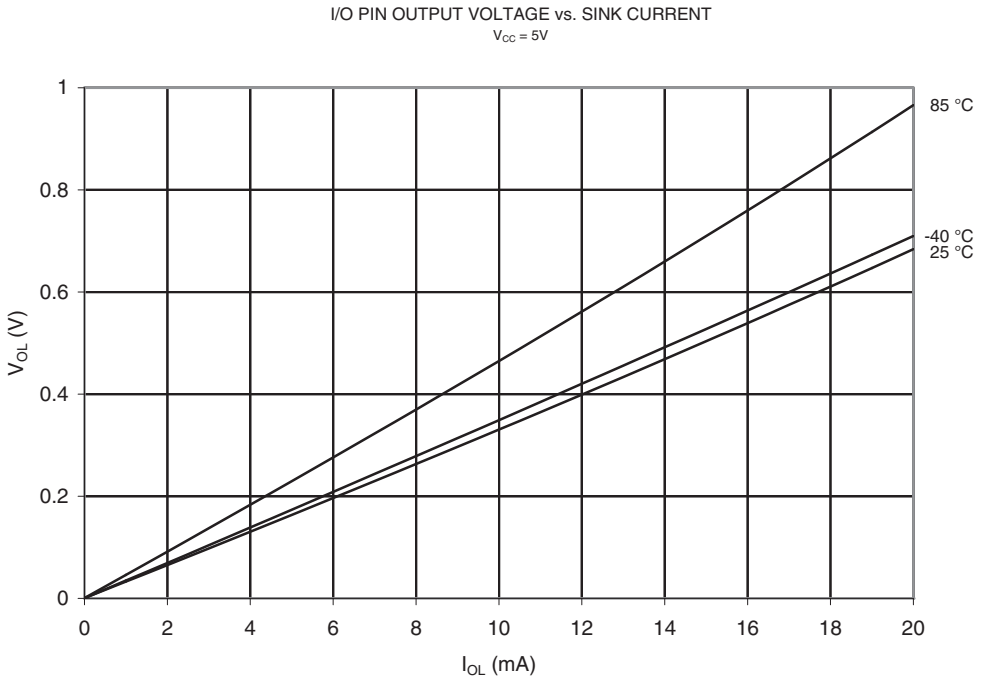


Figure 17-22. I/O Pin Output Voltage vs. Source Current (V_{CC} = 1.8V)

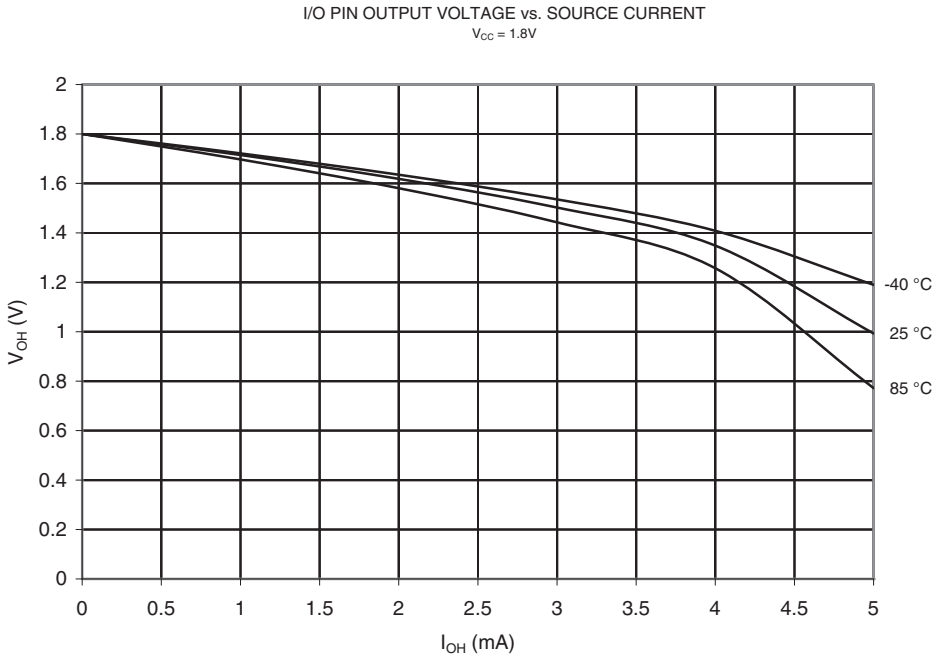


Figure 17-23. I/O Pin Output Voltage vs. Source Current (V_{CC} = 3V)

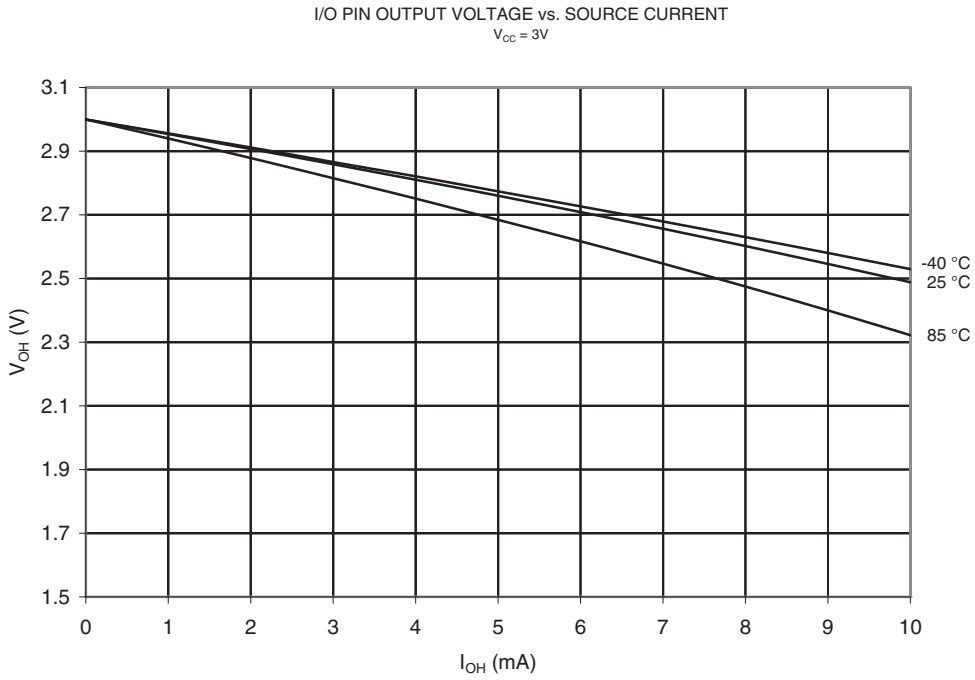


Figure 17-24. I/O Pin output Voltage vs. Source Current (V_{CC} = 5V)

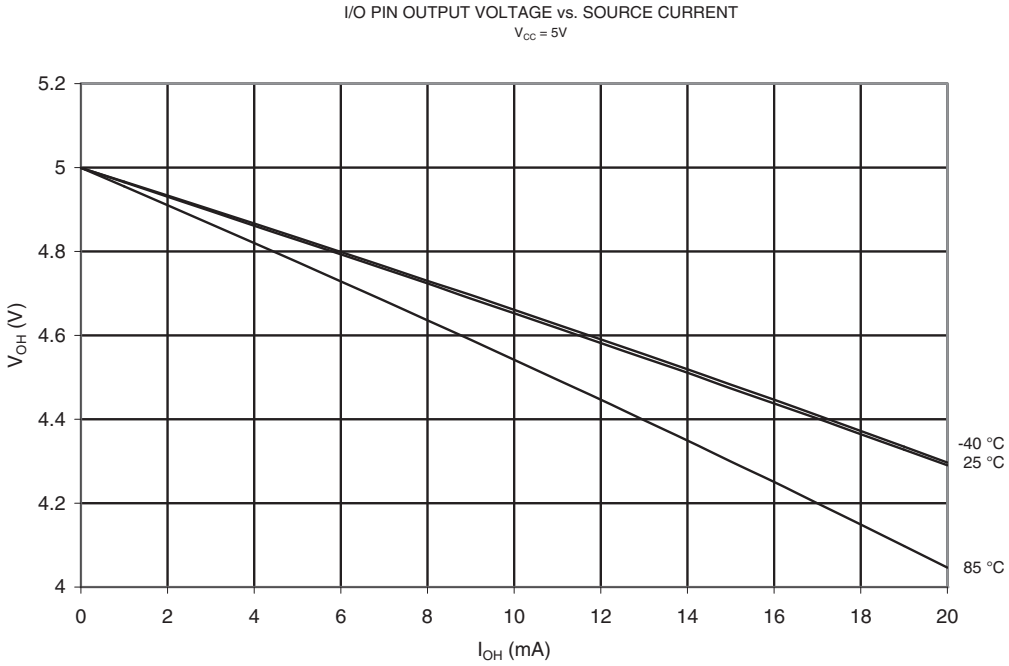


Figure 17-25. Reset Pin as I/O, Output Voltage vs. Sink Current

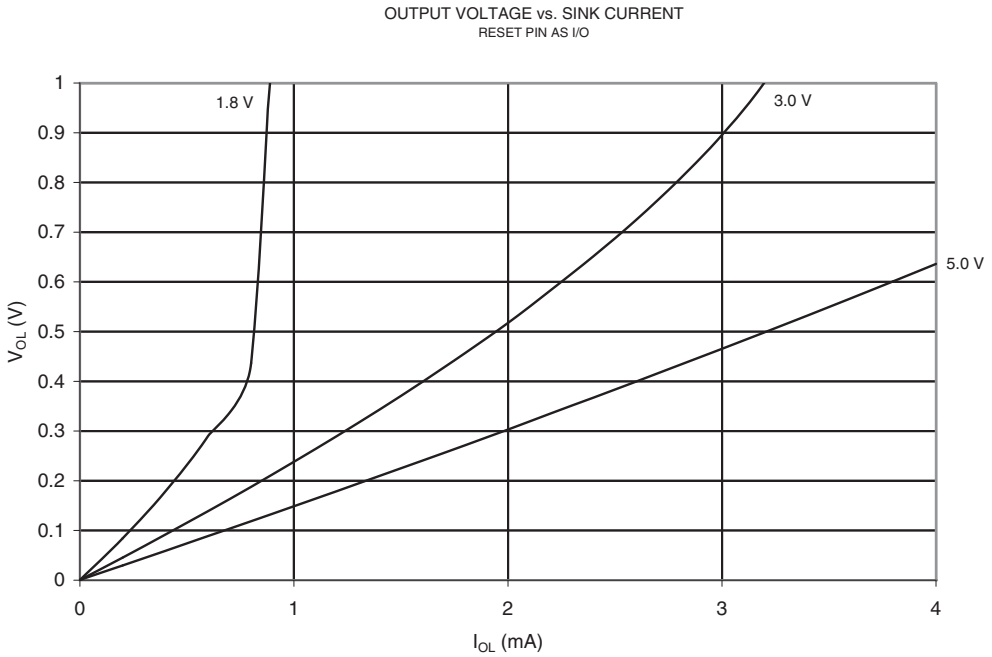
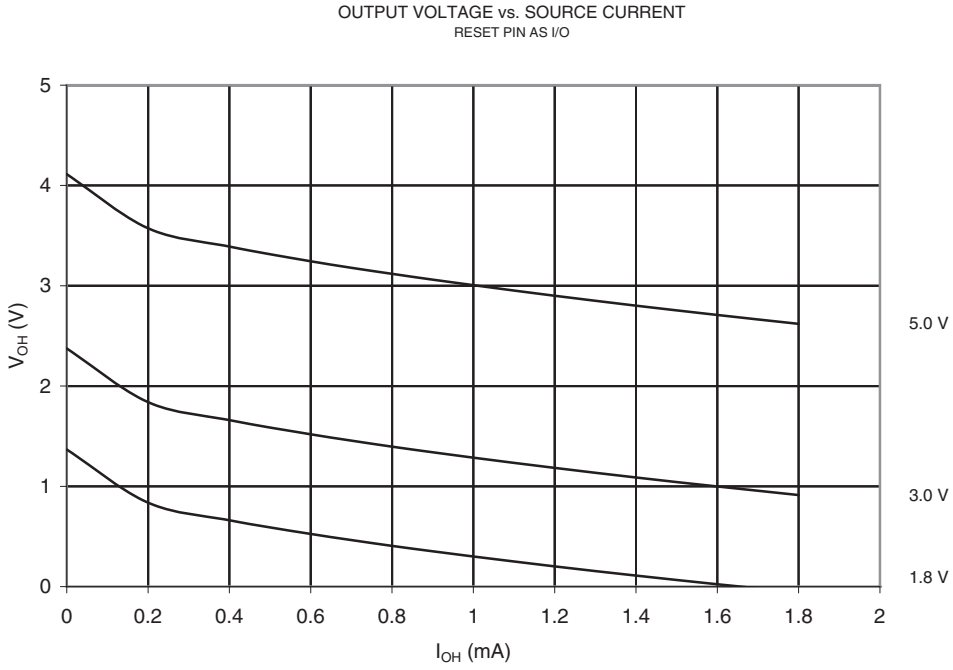


Figure 17-26. Reset Pin as I/O, Output Voltage vs. Source Current



17.2.6 Pin Threshold and Hysteresis

Figure 17-27. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, IO Pin Read as ‘1’)

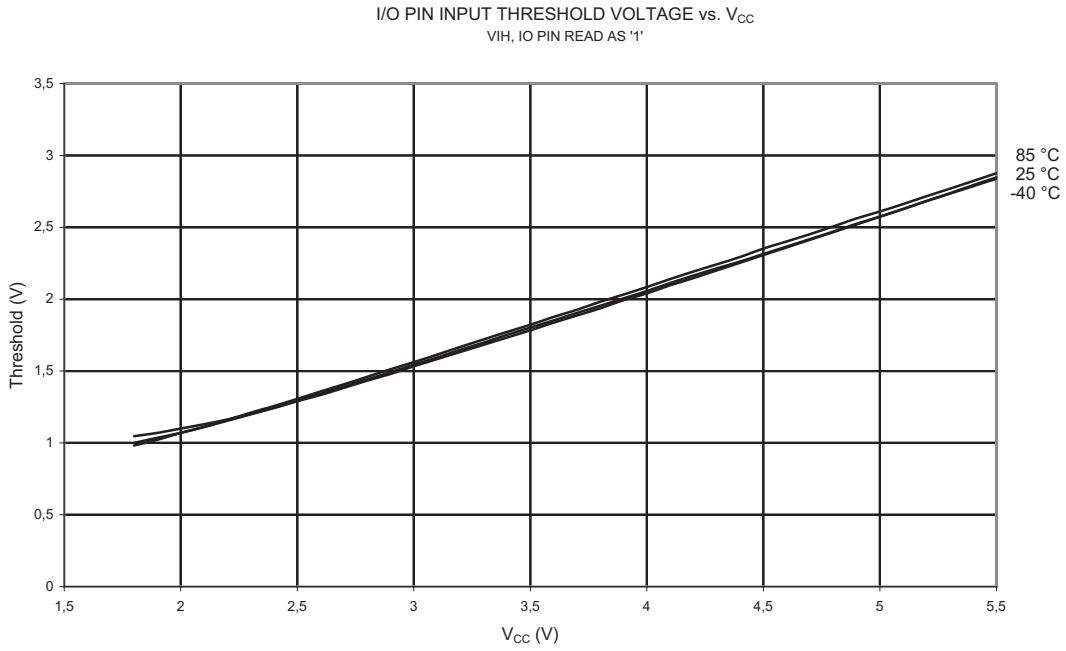


Figure 17-28. I/O Pin Input threshold Voltage vs. V_{CC} (V_{IL}, IO Pin Read as ‘0’)

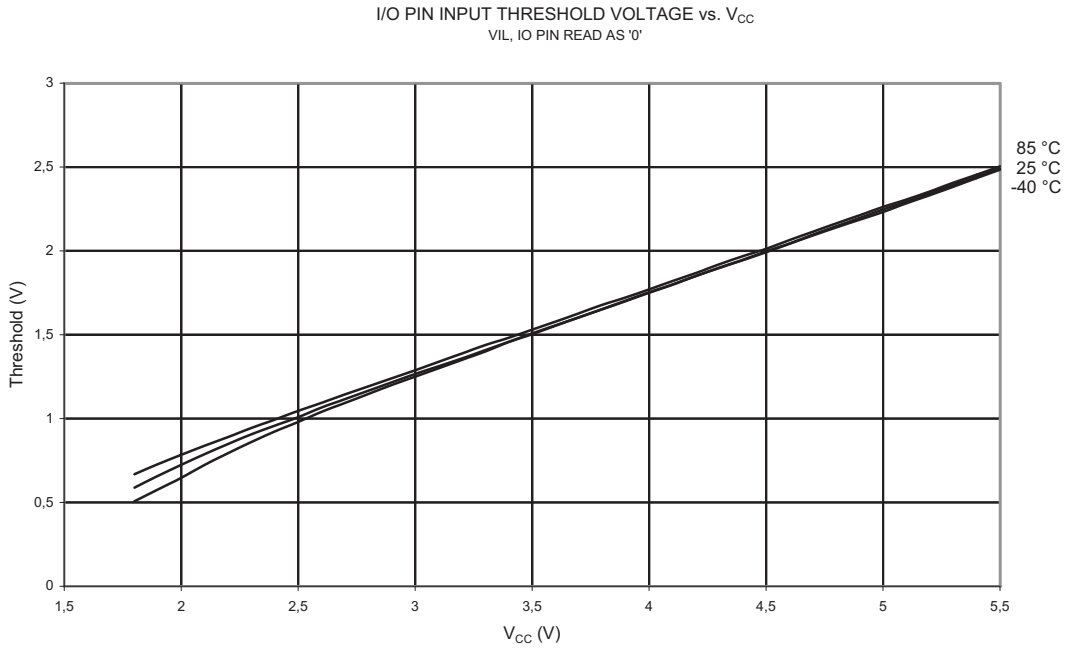


Figure 17-29. I/O Pin Input Hysteresis vs. V_{CC}

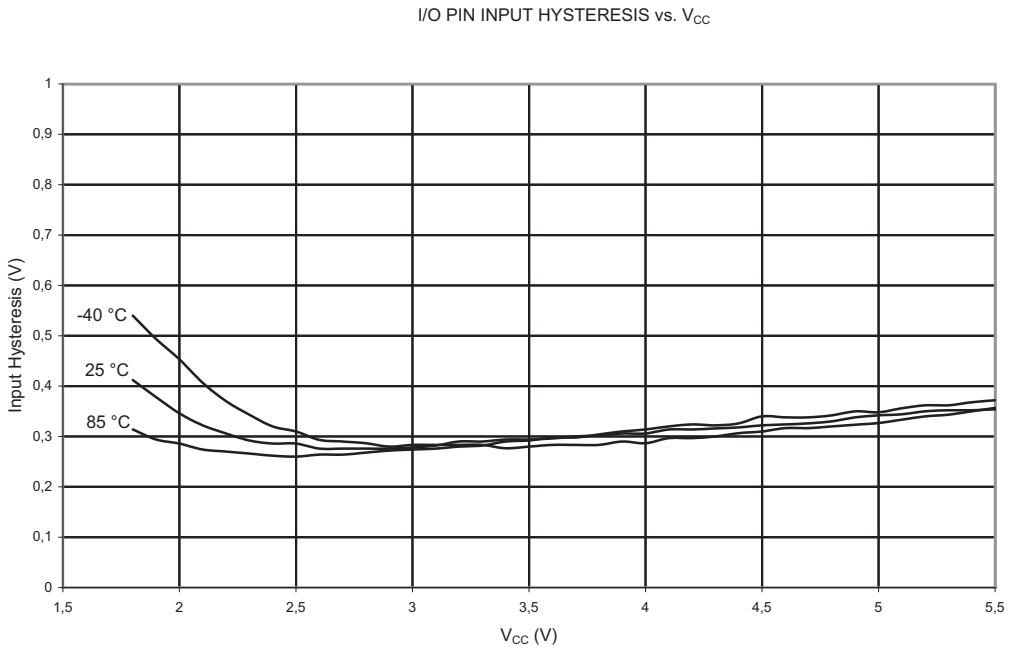


Figure 17-30. Reset Pin as I/O, Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as '1')

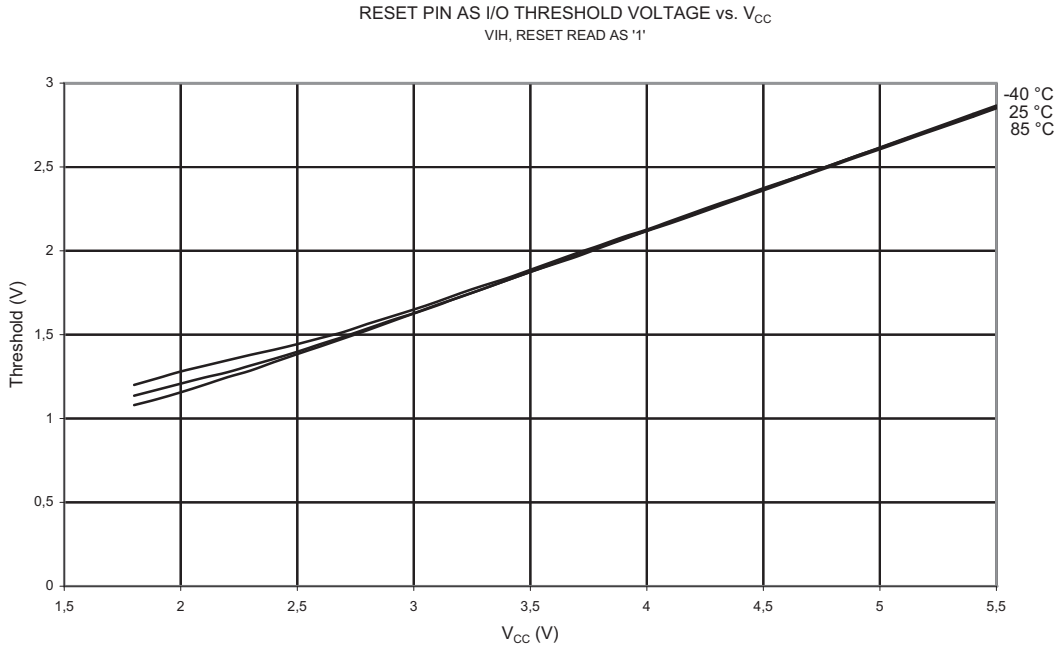


Figure 17-31. Reset Pin as I/O, Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O pin Read as '0')

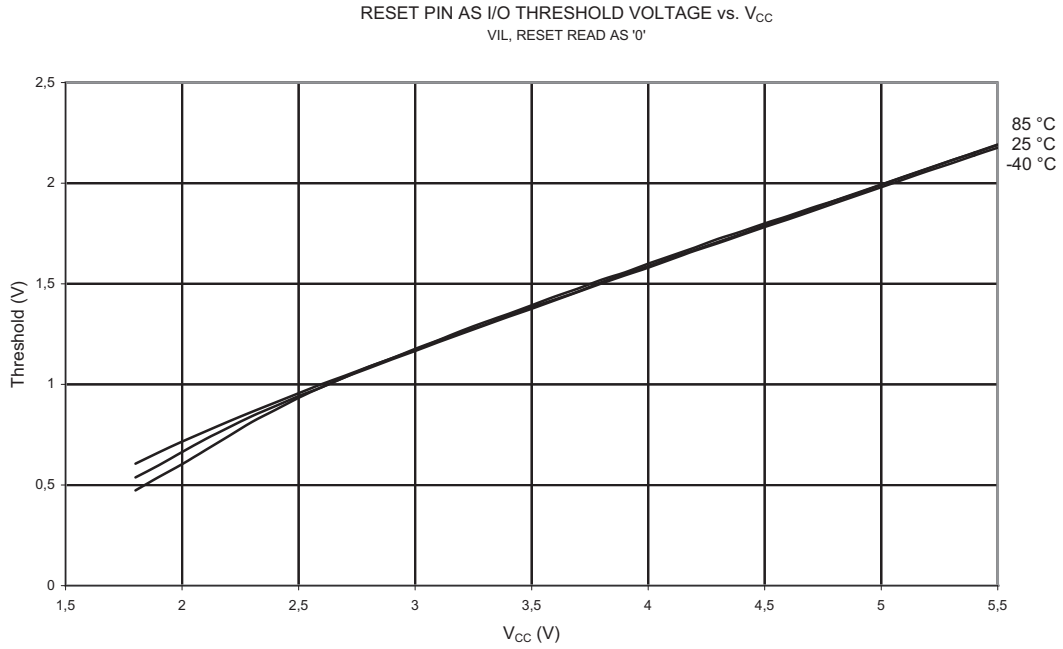


Figure 17-32. Reset Input Hysteresis vs. V_{CC} (Reset Pin Used as I/O)

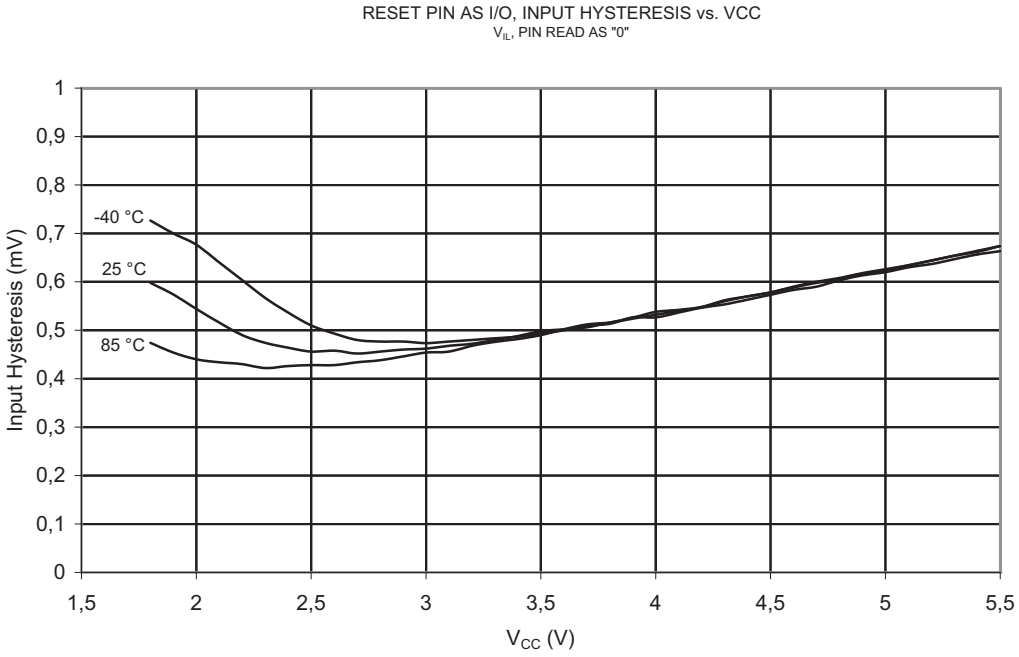


Figure 17-33. Reset Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as '1')

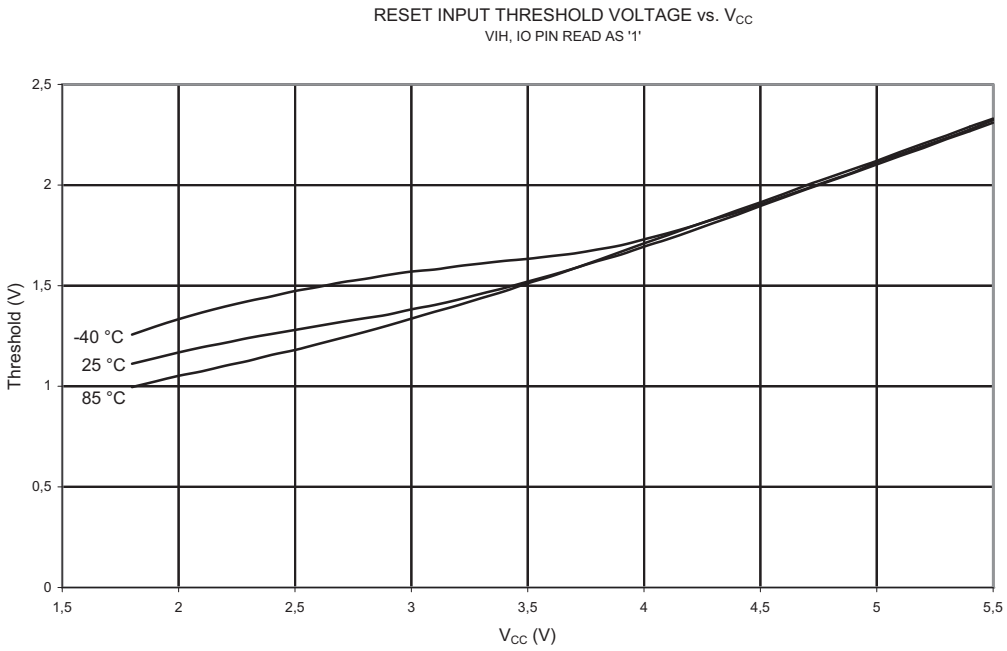


Figure 17-34. Reset Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O pin Read as '0')

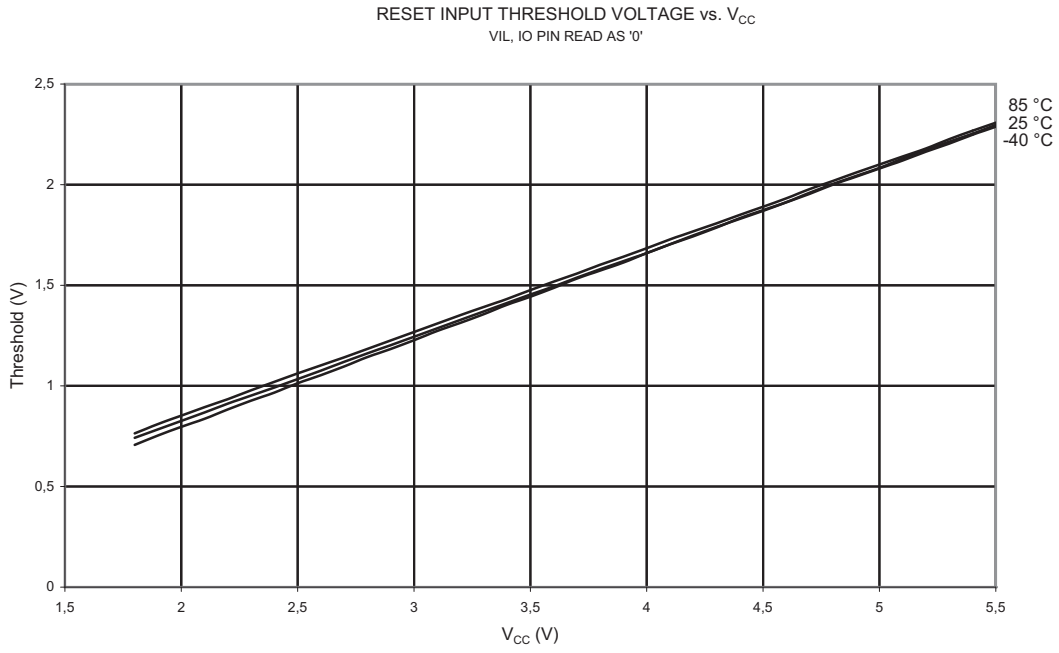
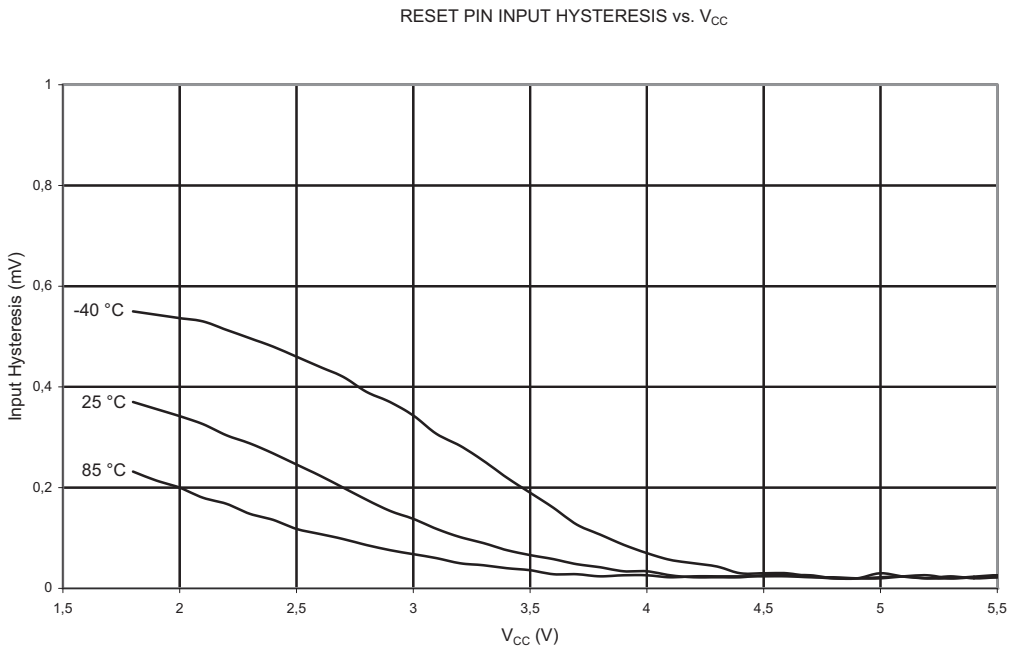
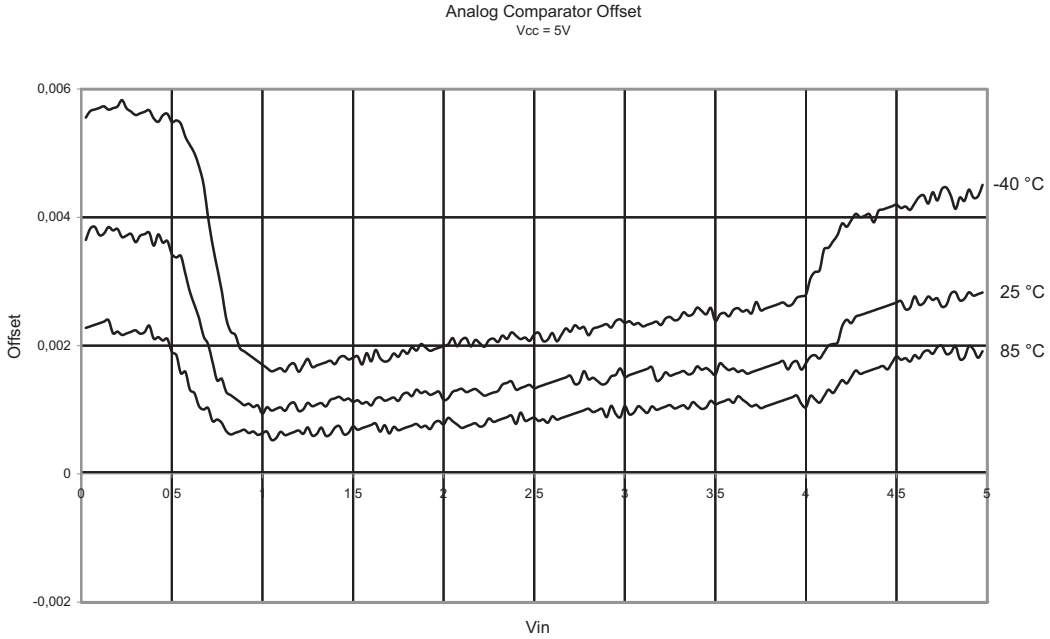


Figure 17-35. Reset Pin, Input Hysteresis vs. V_{CC}



17.2.7 Analog Comparator Offset

Figure 17-36. Analog Comparator Offset



17.2.8 Internal Oscillator Speed

Figure 17-37. Watchdog Oscillator Frequency vs. V_{CC}

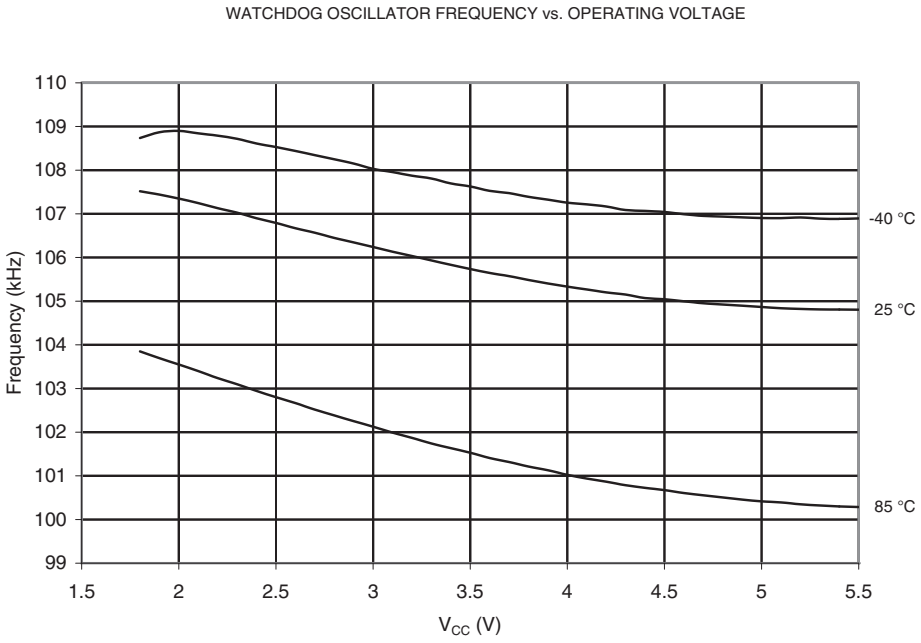


Figure 17-38. Watchdog Oscillator Frequency vs. Temperature

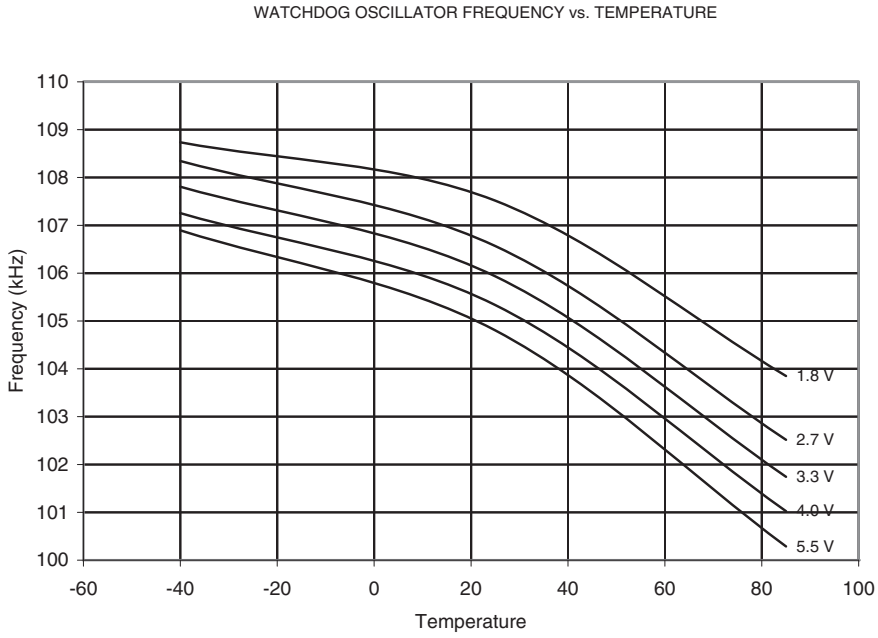


Figure 17-39. Calibrated Oscillator Frequency vs. V_{CC}

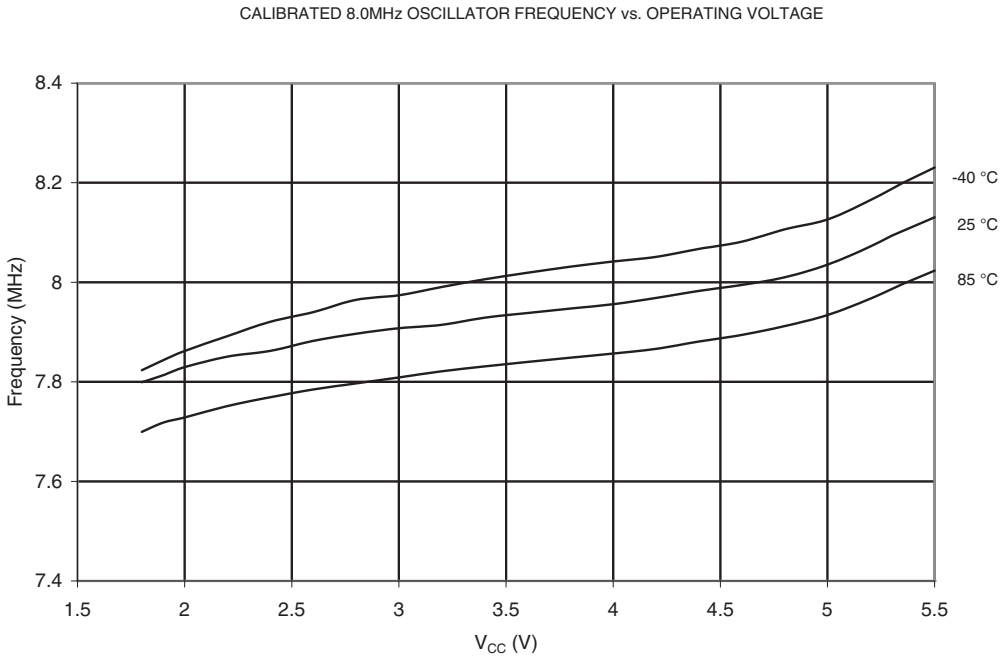


Figure 17-40. Calibrated Oscillator Frequency vs. Temperature

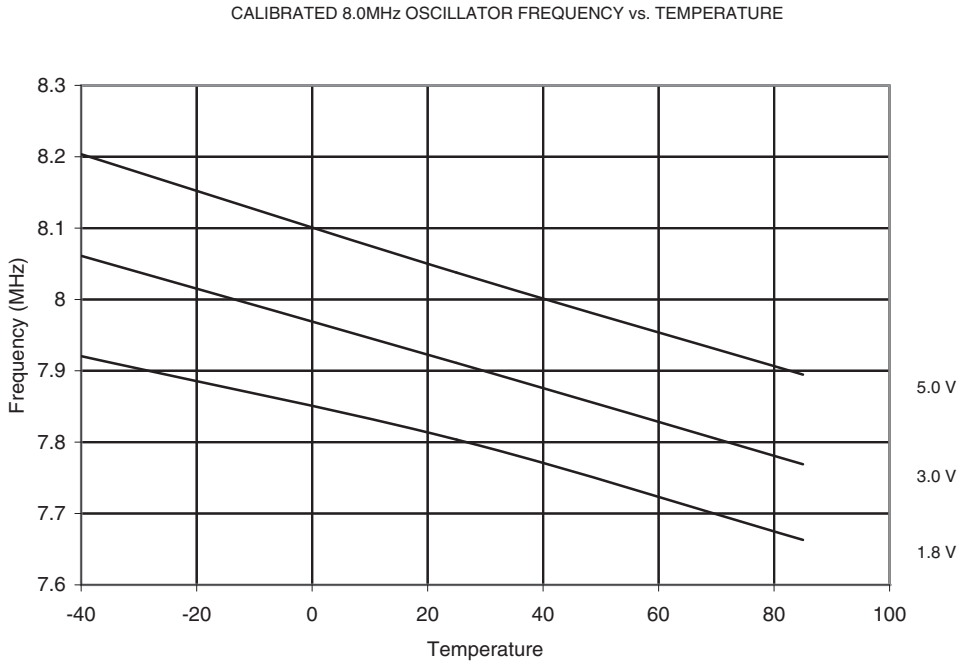
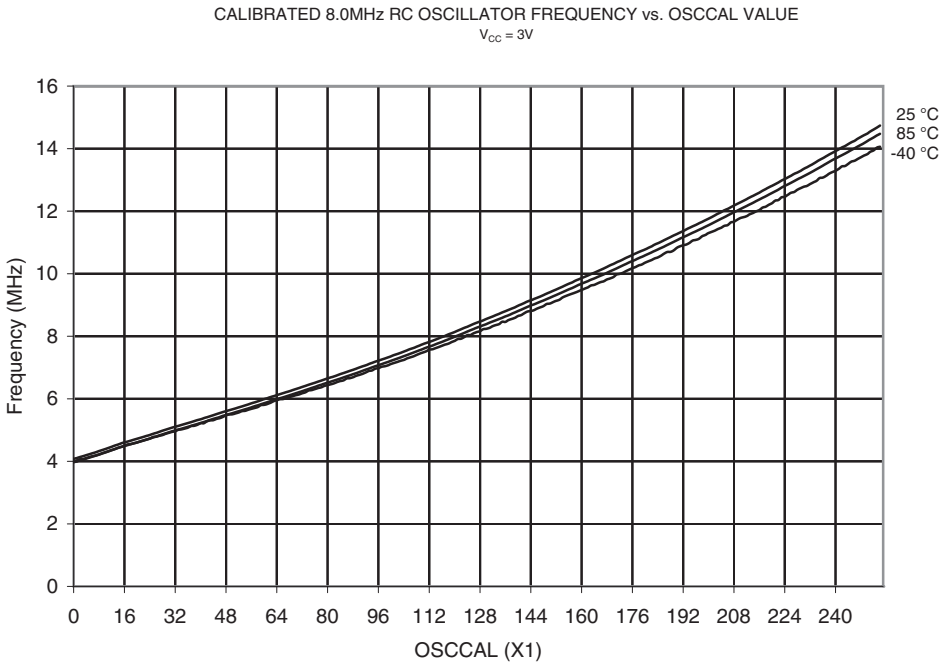


Figure 17-41. Calibrated Oscillator Frequency vs, OSCCAL Value



17.2.9 VLM Thresholds

Figure 17-42. VLM1L Threshold of V_{CC} Level Monitor

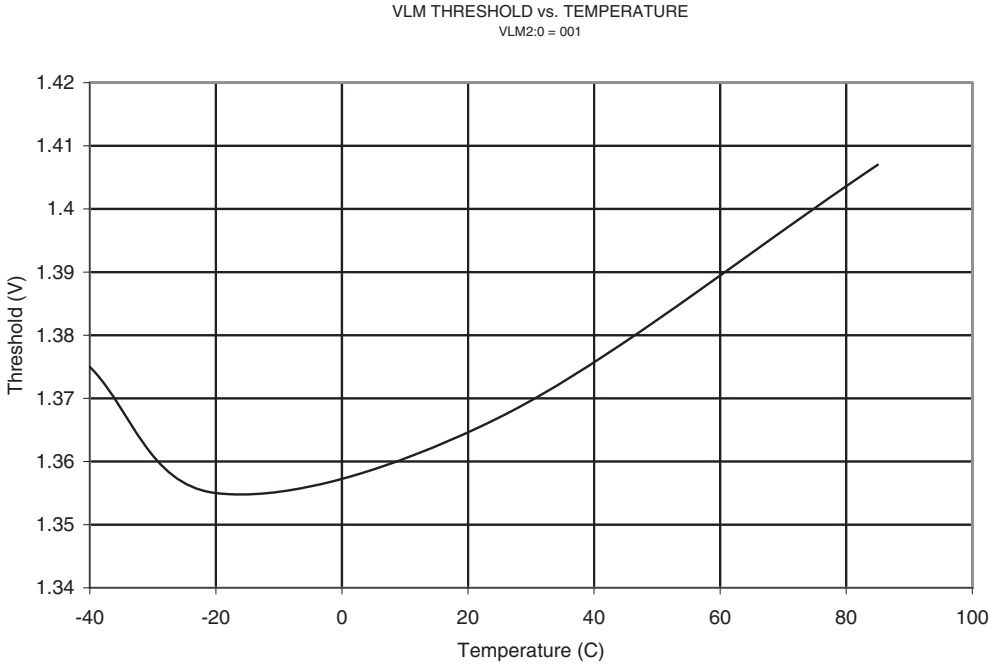


Figure 17-43. VLM1H Threshold of V_{CC} Level Monitor

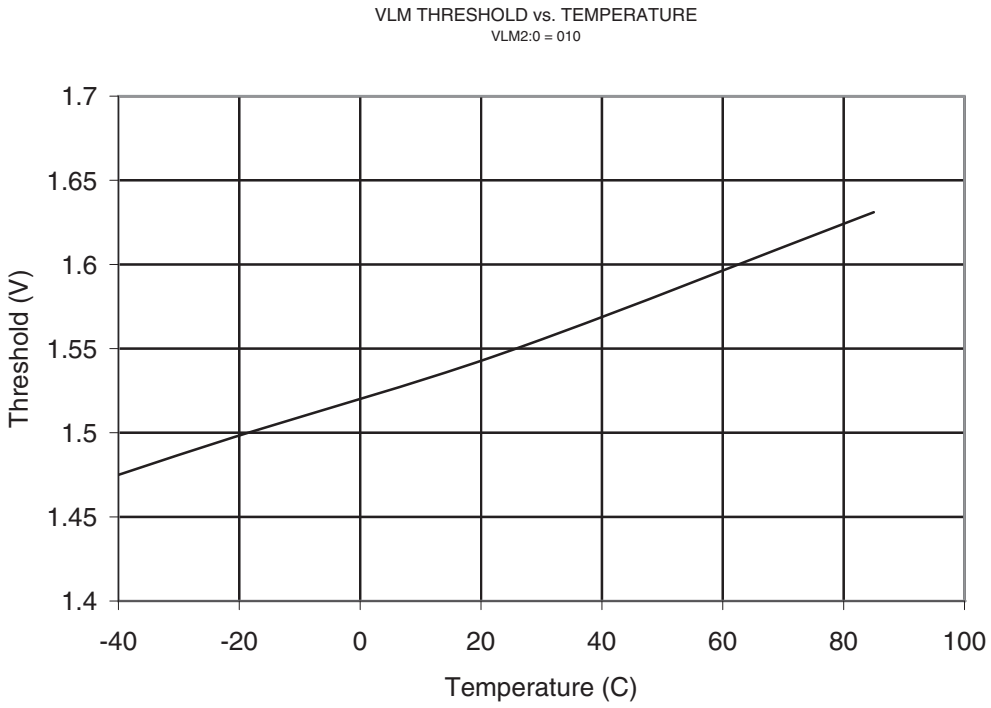


Figure 17-44. VLM2 Threshold of V_{CC} Level Monitor

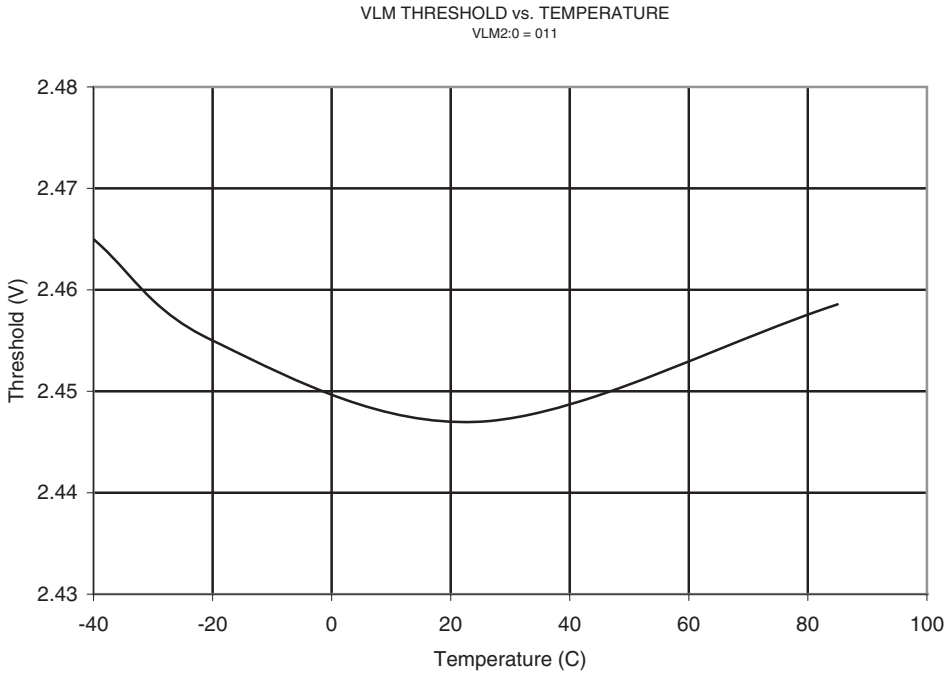
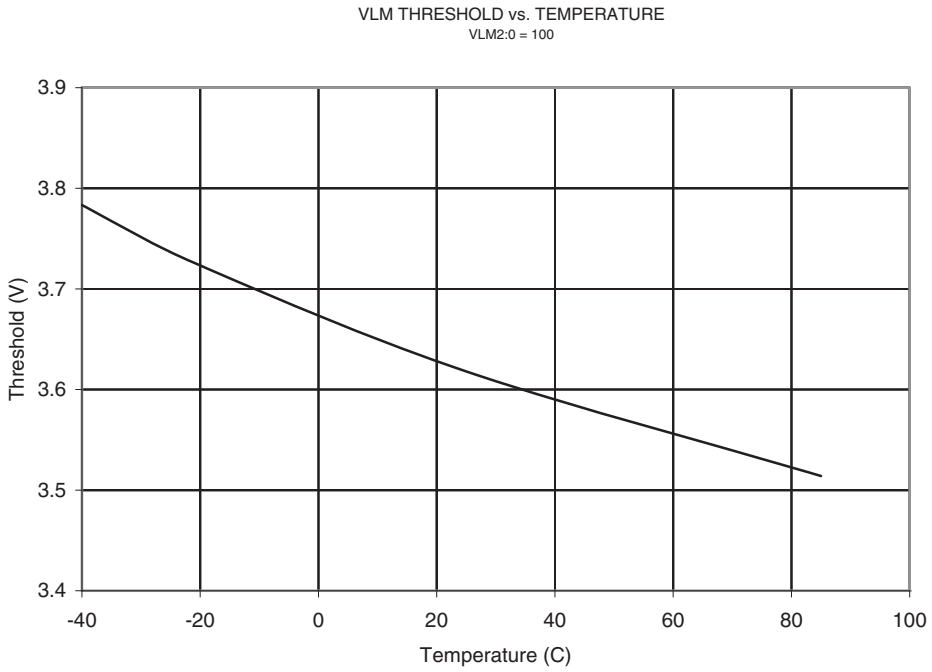


Figure 17-45. VLM3 Threshold of V_{CC} Level Monitorr2



17.2.10 Current Consumption of Peripheral Units

Figure 17-46. ADC Current vs. V_{CC} (ATtiny5/10, only)

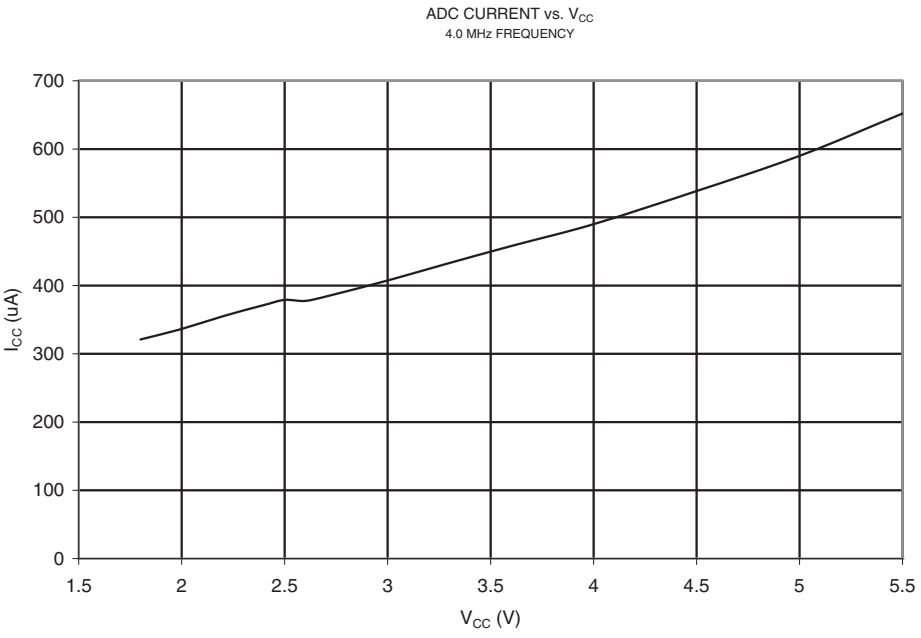


Figure 17-47. Analog Comparator Current vs. V_{CC}

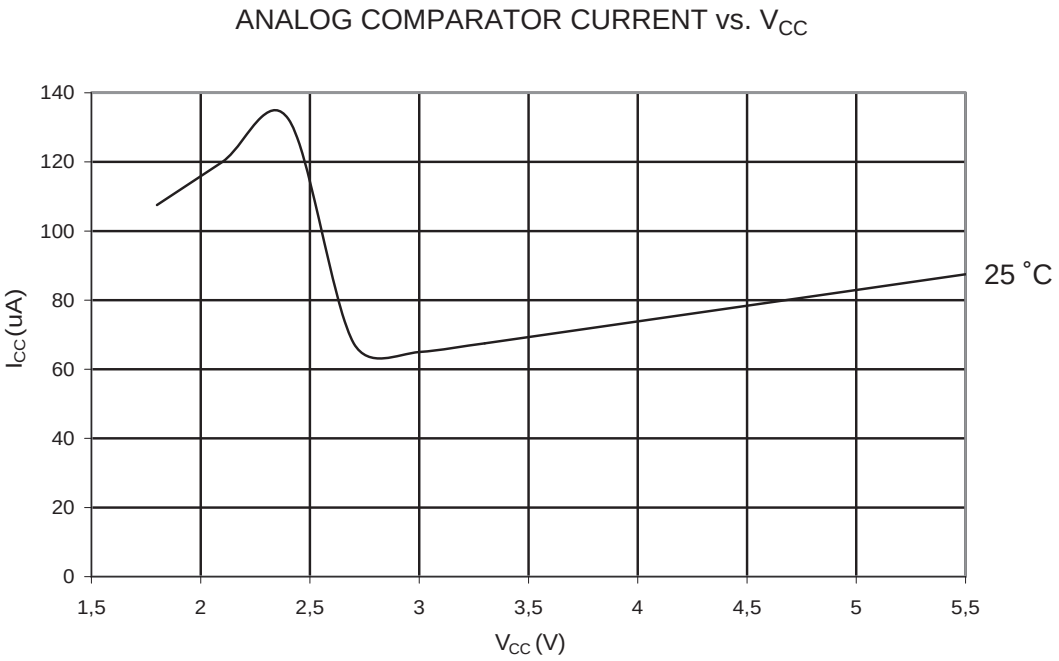


Figure 17-48. V_{CC} Level Monitor Current vs. V_{CC}

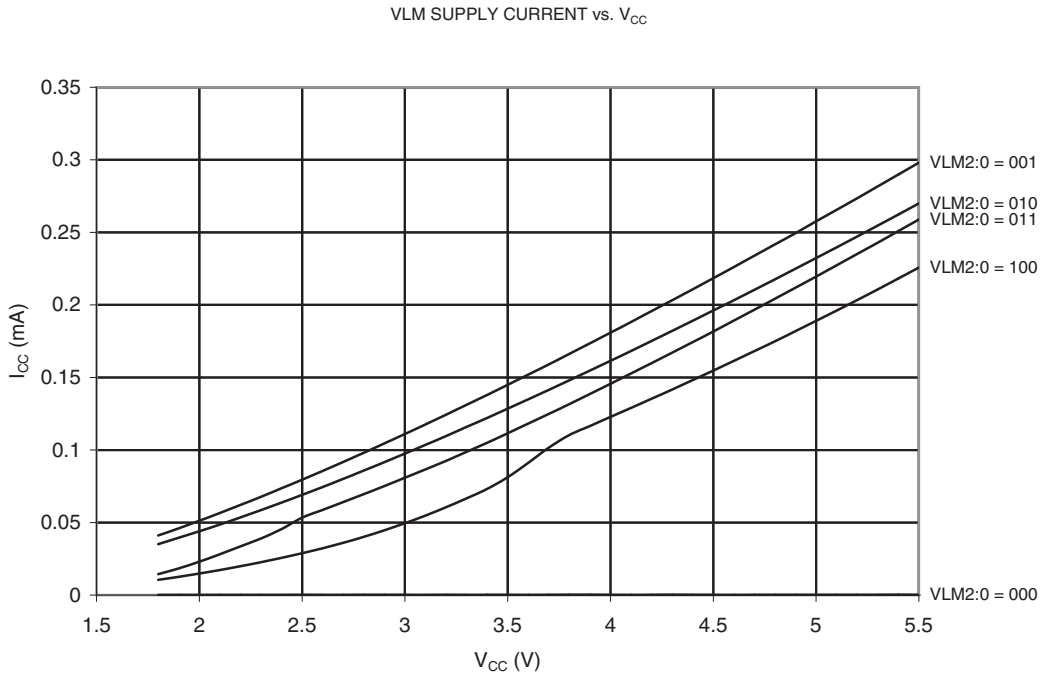


Figure 17-49. Temperature Dependence of VLM Current vs. V_{CC}

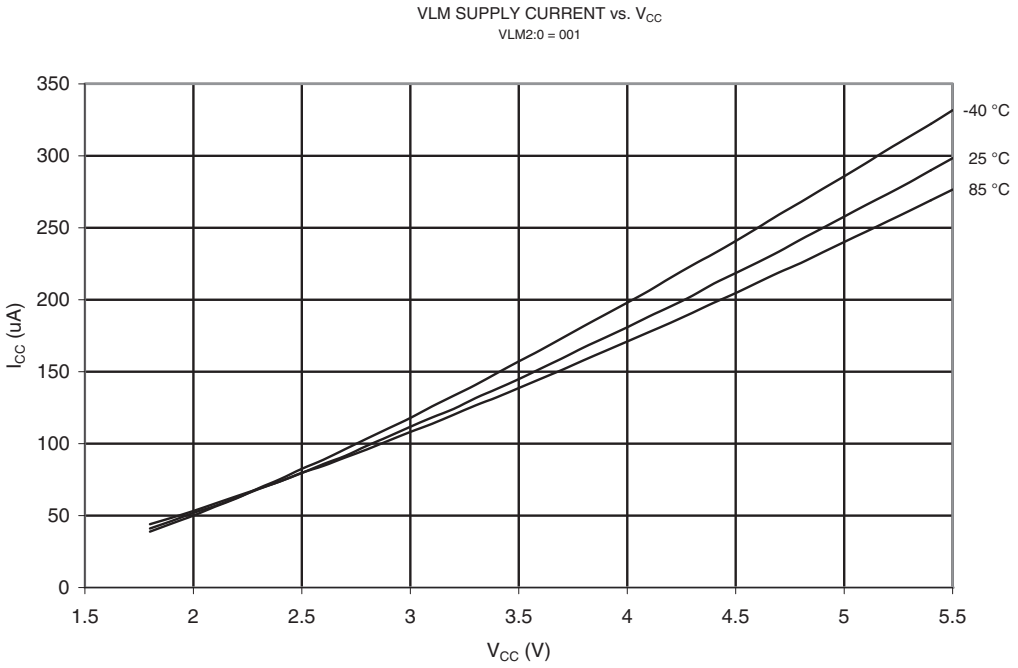
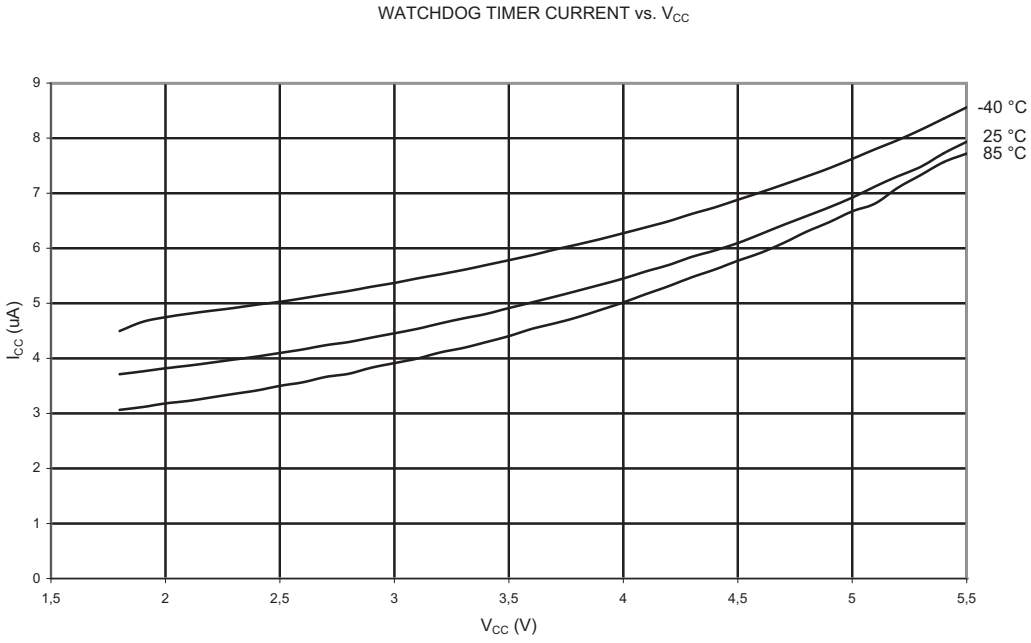


Figure 17-50. Watchdog Timer Current vs. V_{CC}



17.2.11 Current Consumption in Reset and Reset Pulsewidth

Figure 17-51. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, excluding Current Through the Reset Pull-up)

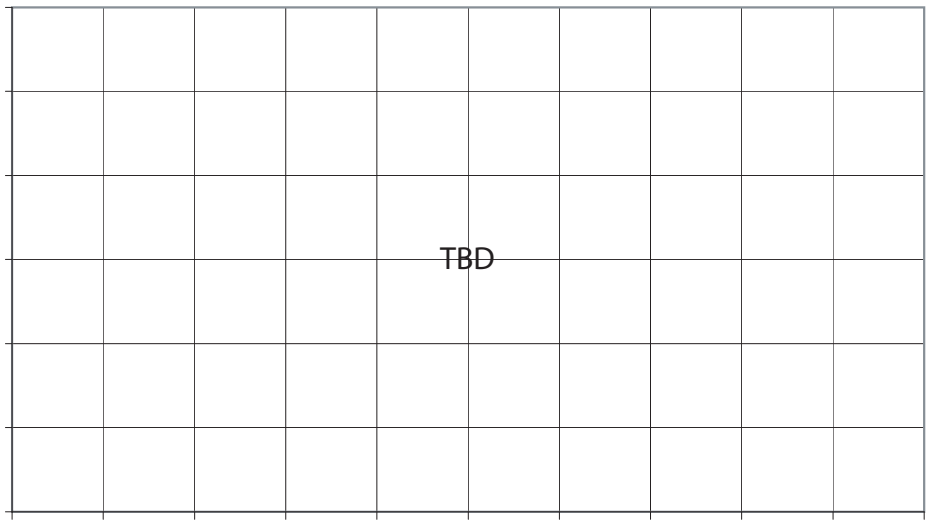
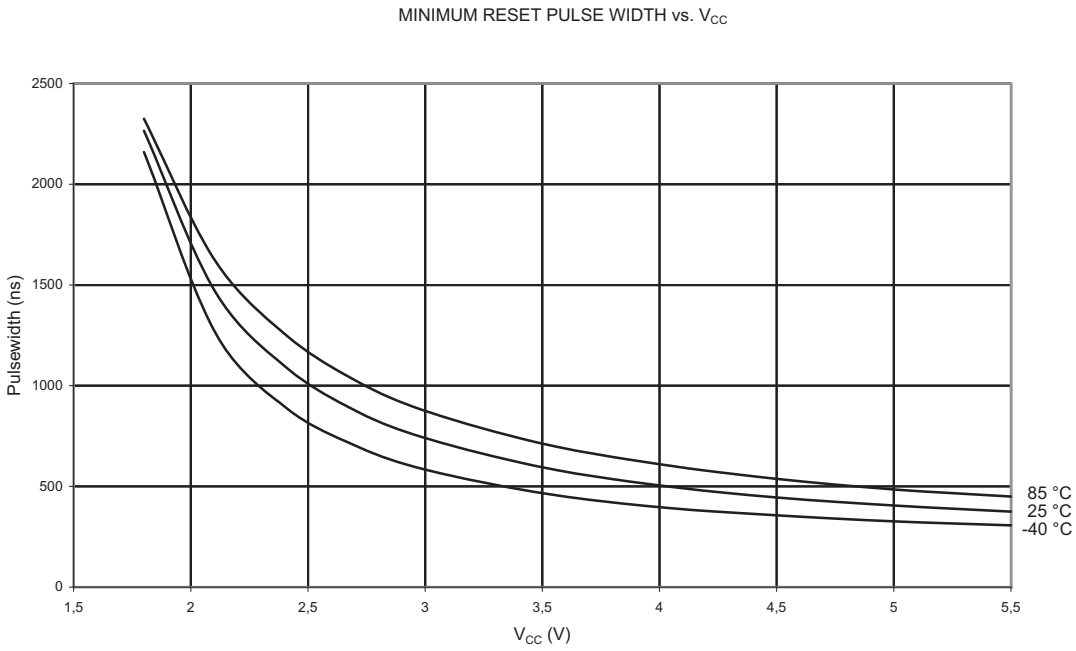


Figure 17-52. Minimum Reset Pulse Width vs. V_{CC}



18. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F	SREG	I	T	H	S	V	N	Z	C	Page 12
0x3E	SPH	Stack Pointer High Byte								Page 12
0x3D	SPL	Stack Pointer Low Byte								Page 12
0x3C	CCP	CPU Change Protection Byte								Page 12
0x3B	RSTFLR	–	–	–	–	WDRF	–	EXTRF	PORF	Page 34
0x3A	SMCR	–	–	–	–	SM2	SM1	SM0	SE	Page 25
0x39	OSCCAL	Oscillator Calibration Byte								Page 21
0x38	Reserved	–	–	–	–	–	–	–	–	
0x37	CLKMSR	–	–	–	–	–	–	CLKMS1	CLKMS0	Page 21
0x36	CLKPSR	–	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	Page 22
0x35	PRR	–	–	–	–	–	–	PRADC	PRTIM0	Page 26
0x34	VLMCSR	VLMF	VLMIE	–	–	–	VLM2	VLM1	VLM0	Page 33
0x33	NVMCMD	–	–	NVM Comman						Page 115
0x32	NVMCSR	NVMBSY	–	–	–	–	–	–	–	Page 115
0x31	WDTCSR	WDIF	WDIE	WDP3	–	WDE	WDP2	WDP1	WDP0	Page 32
0x30	Reserved	–	–	–	–	–	–	–	–	
0x2F	GTCCR	TSM	–	–	–	–	–	–	PSR	Page 79
0x2E	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	Page 73
0x2D	TCCR0B	ICNC0	ICES0	–	WGM03	WGM02	CS02	CS01	CS00	Page 75
0x2C	TCCR0C	FOC0A	FOC0B	–	–	–	–	–	–	Page 76
0x2B	TIMSK0	–	–	ICIE0	–	–	OCIE0B	OCIE0A	TOIE0	Page 78
0x2A	TIFR0	–	–	ICF0	–	–	OCF0B	OCF0A	TOV0	Page 79
0x29	TCNT0H	Timer/Counter0 – Counter Register High Byte								Page 77
0x28	TCNT0L	Timer/Counter0 – Counter Register Low Byte								Page 77
0x27	OCR0AH	Timer/Counter0 – Compare Register A High Byte								Page 77
0x26	OCR0AL	Timer/Counter0 – Compare Register A Low Byte								Page 77
0x25	OCR0BH	Timer/Counter0 – Compare Register B High Byte								Page 77
0x24	OCR0BL	Timer/Counter0 – Compare Register B Low Byte								Page 77
0x23	ICR0H	Timer/Counter0 - Input Capture Register High Byte								Page 78
0x22	ICR0L	Timer/Counter0 - Input Capture Register Low Byte								Page 78
0x21	Reserved	–	–	–	–	–	–	–	–	
0x20	Reserved	–	–	–	–	–	–	–	–	
0x1F	ACSR	ACD	–	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	Page 81
0x1E	Reserved	–	–	–	–	–	–	–	–	
0x1D	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	Page 93
0x1C	ADCSRB	–	–	–	–	–	ADTS2	ADTS1	ADTS0	Page 94
0x1B	ADMUX	–	–	–	–	–	–	MUX1	MUX0	Page 93
0x1A	Reserved	–	–	–	–	–	–	–	–	
0x19	ADCL	ADC Conversion Result								Page 95
0x18	Reserved	–	–	–	–	–	–	–	–	
0x17	DIDR0	–	–	–	–	ADC3D	ADC2D	ADC1D	ADC0D	Page 82, Page 95
0x16	Reserved	–	–	–	–	–	–	–	–	
0x15	EICRA	–	–	–	–	–	–	ISC01	ISC00	Page 37
0x14	EIFR	–	–	–	–	–	–	–	INTF0	Page 38
0x13	EIMSK	–	–	–	–	–	–	–	INT0	Page 38
0x12	PCICR	–	–	–	–	–	–	–	PCIE0	Page 39
0x11	PCIFR	–	–	–	–	–	–	–	PCIF0	Page 39
0x10	PCMSK	–	–	–	–	PCINT3	PCINT2	PCINT1	PCINT0	Page 39
0x0F	Reserved	–	–	–	–	–	–	–	–	
0x0E	Reserved	–	–	–	–	–	–	–	–	
0x0D	Reserved	–	–	–	–	–	–	–	–	
0x0C	PORTCR	–	–	–	–	–	–	BBMB	–	Page 50
0x0B	Reserved	–	–	–	–	–	–	–	–	
0x0A	Reserved	–	–	–	–	–	–	–	–	
0x09	Reserved	–	–	–	–	–	–	–	–	
0x08	Reserved	–	–	–	–	–	–	–	–	
0x07	Reserved	–	–	–	–	–	–	–	–	
0x06	Reserved	–	–	–	–	–	–	–	–	
0x05	Reserved	–	–	–	–	–	–	–	–	
0x04	Reserved	–	–	–	–	–	–	–	–	
0x03	PUEB	–	–	–	–	PUEB3	PUEB2	PUEB1	PUEB0	Page 50
0x02	PORTB	–	–	–	–	PORTB3	PORTB2	PORTB1	PORTB0	Page 51
0x01	DDRB	–	–	–	–	DDRB3	DDRB2	DDRB1	DDRB0	Page 51
0x00	PINB	–	–	–	–	PINB3	PINB2	PINB1	PINB0	Page 51

Note:
1. For compatibility with future devices, reserved bits should be written to 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 AVRs, 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.
4. The ADC is available in ATtiny5/10, only.

19. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd ← $FF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3/4
ICALL		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	3/4
RET		Subroutine Return	PC ← STACK	None	4/5
RETI		Interrupt Return	PC ← STACK	I	4/5
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, C,N,V,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd – K	Z, C,N,V,S,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	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	A, b	Skip if Bit in I/O Register is Set	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
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					
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V,H	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,H	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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	1
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	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
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow.	V ← 1	V	1
CLV		Clear Two's 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	Copy Register	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	1/2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2/3
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	1/2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2/3
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	1/2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2/3
LDS	Rd, k	Store Direct from SRAM	Rd ← (k)	None	1
ST	X, Rr	Store Indirect	(X) ← Rr	None	1
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	1
ST	- X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	1
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	1
ST	- Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	1
ST	Z+, Rr	Store Indirect and Post-Increment.	(Z) ← Rr, Z ← Z + 1	None	1
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	1
IN	Rd, A	In from I/O Location	Rd ← I/O (A)	None	1
OUT	A, Rr	Out to I/O Location	I/O (A) ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
BREAK		Break	(see specific descr. for Break)	None	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1

20. Ordering Information

20.1 ATtiny4

Speed (MHz)	Power Supply	Ordering Code[2]	Package[1]	Operational Range
12	1.8 - 5.5V	ATtiny4-TSHR[3][4]	6ST1	Industrial (-40°C to 85°C)[4]

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Topside marking for ATtiny4: T4**x** (**x** stands for “die revision”).
4. Bottomside marking for ATtiny4: z**H**zzz [**H** stands for (-40°C to 85°C)].

Package Type	
6ST1	6-lead, 2.90 x 1.60 mm Plastic Small Outline Package (SOT23)

20.2 ATtiny5

Speed (MHz)	Power Supply	Ordering Code[2]	Package[1]	Operational Range
12	1.8 - 5.5V	ATtiny5-TSHR[3][4]	6ST1	Industrial (-40°C to 85°C)[4]

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Topside marking for ATtiny5: T5**x** (**x** stands for “die revision”).
4. Bottomside marking for ATtiny5: z**H**zzz [**H** stands for (-40°C to 85°C)].

Package Type	
6ST1	6-lead, 2.90 x 1.60 mm Plastic Small Outline Package (SOT23)

20.3 ATtiny9

Speed (MHz)	Power Supply	Ordering Code[2]	Package[1]	Operational Range
12	1.8 - 5.5V	ATtiny9-TSHR[3][4]	6ST1	Industrial (-40°C to 85°C)[4]

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Topside marking for ATtiny9: T9**x** (**x** stands for “die revision”).
4. Bottomside marking for ATtiny9: z**H**zzz [**H** stands for (-40°C to 85°C)].

Package Type	
6ST1	6-lead, 2.90 x 1.60 mm Plastic Small Outline Package (SOT23)

20.4 ATtiny10

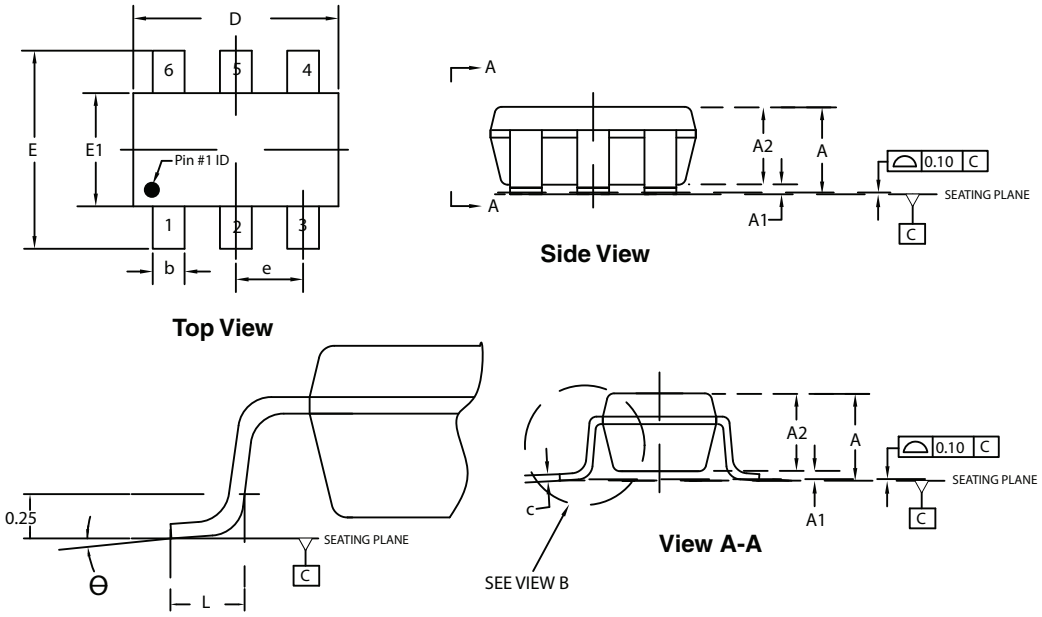
Speed (MHz)	Power Supply	Ordering Code[2]	Package[1]	Operational Range
12	1.8 - 5.5V	ATtiny10-TSHR[3][4]	6ST1	Industrial (-40°C to 85°C)[4]

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Topside marking for ATtiny10: T10**x** (**x** stands for “die revision”).
4. Bottomside marking for ATtiny10: z**H**zzz [**H** stands for (-40°C to 85°C)].

Package Type	
6ST1	6-lead, 2.90 x 1.60 mm Plastic Small Outline Package (SOT23)

21. Packaging Information

21.1 6ST1



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.45	
A1	0	–	0.15	
A2	0.90	–	1.30	
D	2.80	2.90	3.00	2
E	2.60	2.80	3.00	
E1	1.50	1.60	1.75	
L	0.30	0.45	0.55	
e	0.95 BSC			
b	0.30	–	0.50	3
c	0.09	–	0.20	
θ	0°	–	8°	

Notes:
1. This package is compliant with JEDEC specification MO-178 Variation AB
2. Dimension D does not include mold Flash, protrusions or gate burrs. Mold Flash, protrustion or gate burrs shall not exceed 0.25 mm per end.
3. Dimension b does not include dambar protrusion. Allowable dambar protrusion shall not cause the lead width to exceed the maximum b dimension by more than 0.08 mm
4. Die is facing down after trim/form.

6/30/08

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	**6ST1,** 6-lead, 2.90 x 1.60 mm Plastic Small Outline Package (SOT23)	TAQ	6ST1	A

22. Errata

The revision letters in this section refer to the revision of the corresponding ATtiny4/5/9/10 device.

22.1 ATtiny4

22.1.1 Rev. D

- **ESD HBM (ESD STM 5.1) level ±1000V**
- **Lock bits re-programming**

1. ESD HBM (ESD STM 5.1) level ±1000V

The device meets ESD HBM (ESD STM 5.1) level ±1000V.

Problem Fix / Workaround

Always use proper ESD protection measures (Class 1C) when handling integrated circuits before and during assembly.

2. Lock bits re-programming

Attempt to re-program Lock bits to present, or lower protection level (tampering attempt), causes erroneously one, random line of Flash program memory to get erased. The Lock bits will not get changed, as they should not.

Problem Fix / Workaround

Do not attempt to re-program Lock bits to present, or lower protection level.

22.1.2 Rev. A – C

Not sampled.

22.2 ATtiny5

22.2.1 Rev. D

- **ESD HBM (ESD STM 5.1) level ±1000V**
- **Lock bits re-programming**

1. ESD HBM (ESD STM 5.1) level ±1000V

The device meets ESD HBM (ESD STM 5.1) level ±1000V.

Problem Fix / Workaround

Always use proper ESD protection measures (Class 1C) when handling integrated circuits before and during assembly.

2. Lock bits re-programming

Attempt to re-program Lock bits to present, or lower protection level (tampering attempt), causes erroneously one, random line of Flash program memory to get erased. The Lock bits will not get changed, as they should not.

Problem Fix / Workaround

Do not attempt to re-program Lock bits to present, or lower protection level.

22.2.2 Rev. A – C

Not sampled.

22.3 ATtiny9

22.3.1 Rev. D

- **ESD HBM (ESD STM 5.1) level ±1000V**
- **Lock bits re-programming**

1. ESD HBM (ESD STM 5.1) level ±1000V

The device meets ESD HBM (ESD STM 5.1) level ±1000V.

Problem Fix / Workaround

Always use proper ESD protection measures (Class 1C) when handling integrated circuits before and during assembly.

2. Lock bits re-programming

Attempt to re-program Lock bits to present, or lower protection level (tampering attempt), causes erroneously one, random line of Flash program memory to get erased. The Lock bits will not get changed, as they should not.

Problem Fix / Workaround

Do not attempt to re-program Lock bits to present, or lower protection level.

22.3.2 Rev. A – C

Not sampled.

22.4 ATtiny10

22.4.1 Rev. C – D

- **ESD HBM (ESD STM 5.1) level ±1000V**
- **Lock bits re-programming**

1. ESD HBM (ESD STM 5.1) level ±1000V

The device meets ESD HBM (ESD STM 5.1) level ±1000V.

Problem Fix / Workaround

Always use proper ESD protection measures (Class 1C) when handling integrated circuits before and during assembly.

2. Lock bits re-programming

Attempt to re-program Lock bits to present, or lower protection level (tampering attempt), causes erroneously one, random line of Flash program memory to get erased. The Lock bits will not get changed, as they should not.

Problem Fix / Workaround

Do not attempt to re-program Lock bits to present, or lower protection level.

22.4.2 Rev. A – B

Not sampled.

23. Datasheet Revision History

23.1 Rev. 8127C – 10/09

1. Updated values and notes:
   - Table 16-1 in Section 16.2 “DC Characteristics” on page 116
   - Table 16-3 in Section 16.4 “Clock Characteristics” on page 118
   - Table 16-6 in Section 16.5.2 “VCC Level Monitor” on page 119
   - Table 16-9 in Section 16.8 “Serial Programming Characteristics” on page 121
2. Updated Figure 16-1 in Section 16.3 “Speed Grades” on page 117
3. Added Typical Characteristics Figure 17-36 in Section 17.2.7 “Analog Comparator Offset” on page 140. Also, updated some other plots in Typical Characteristics.
4. Added topside and bottomside marking notes in Section 20. “Ordering Information” on page 153, up to page 156
5. Added ESD errata, see Section 22. “Errata” on page 158
6. Added Lock bits re-programming errata, see Section 22. “Errata” on page 158

23.2 Rev. 8127B – 08/09

1. Updated document template
2. Expanded document to also cover devices ATtiny4, ATtiny5 and ATtiny9
3. Added section:
   - “Comparison of ATtiny4, ATtiny5, ATtiny9 and ATtiny10” on page 4
4. Updated sections:
   - “ADC Clock – clkADC” on page 18
   - “Starting from Idle / ADC Noise Reduction / Standby Mode” on page 20
   - “ADC Noise Reduction Mode” on page 24
   - “Analog to Digital Converter” on page 25
   - “SMCR – Sleep Mode Control Register” on page 25
   - “PRR – Power Reduction Register” on page 26
   - “Alternate Functions of Port B” on page 48
   - “Overview” on page 83
   - “Physical Layer of Tiny Programming Interface” on page 96
   - “Overview” on page 107
   - “ADC Characteristics (ATtiny5/10, only)” on page 120
   - “Supply Current of I/O Modules” on page 122
   - “Register Summary” on page 149
   - “Ordering Information” on page 153
5. Added figure:
   - “Using an External Programmer for In-System Programming via TPI” on page 97
6. Updated figure:
   - “Data Memory Map (Byte Addressing)” on page 15
7. Added table:
   - “Number of Words and Pages in the Flash (ATtiny4/5)” on page 109

8. Updated tables:
   - “Active Clock Domains and Wake-up Sources in Different Sleep Modes” on page 23
   - “Reset and Interrupt Vectors” on page 35
   - “Number of Words and Pages in the Flash (ATtiny9/10)” on page 109
   - “Signature codes” on page 110

23.3 Rev. 8127A – 04/09

1. Initial revision

Table of Contents

Features *1*

1 Pin Configurations *2*

1.1 Pin Description 2

2 Overview *3*

2.1 Comparison of ATtiny4, ATtiny5, ATtiny9 and ATtiny10 4

3 General Information *5*

3.1 Resources 5

3.2 Code Examples 5

3.3 Data Retention 5

3.4 Disclaimer 5

4 CPU Core *6*

4.1 Architectural Overview 6

4.2 ALU – Arithmetic Logic Unit 7

4.3 Status Register 7

4.4 General Purpose Register File 8

4.5 Stack Pointer 9

4.6 Instruction Execution Timing 9

4.7 Reset and Interrupt Handling 10

4.8 Register Description 12

5 Memories *14*

5.1 In-System Re-programmable Flash Program Memory 14

5.2 Data Memory 14

5.3 I/O Memory 16

6 Clock System *17*

6.1 Clock Subsystems 17

6.2 Clock Sources 18

6.3 System Clock Prescaler 19

6.4 Starting 20

6.5 Register Description 21

7 Power Management and Sleep Modes *23*

7.1 Sleep Modes 23

7.2 Power Reduction Register 24

7.3 Minimizing Power Consumption 24
7.4 Register Description 25

8 System Control and Reset 27

8.1 Resetting the AVR 27
8.2 Reset Sources 27
8.3 Watchdog Timer 30
8.4 Register Description 32

9 Interrupts 35

9.1 Interrupt Vectors 35
9.2 External Interrupts 36
9.3 Register Description 37

10 I/O Ports 40

10.1 Overview 40
10.2 Ports as General Digital I/O 41
10.3 Alternate Port Functions 45
10.4 Register Description 50

11 16-bit Timer/Counter0 52

11.1 Features 52
11.2 Overview 52
11.3 Clock Sources 54
11.4 Counter Unit 55
11.5 Input Capture Unit 57
11.6 Output Compare Units 59
11.7 Compare Match Output Unit 61
11.8 Modes of Operation 62
11.9 Timer/Counter Timing Diagrams 69
11.10 Accessing 16-bit Registers 71
11.11 Register Description 73

12 Analog Comparator 81

12.1 Register Description 81

13 Analog to Digital Converter 83

13.1 Features 83
13.2 Overview 83
13.3 Operation 83

13.4 Starting a Conversion 84
13.5 Prescaling and Conversion Timing 85
13.6 Changing Channel 88
13.7 ADC Noise Canceler 88
13.8 Analog Input Circuitry 89
13.9 Noise Canceling Techniques 90
13.10 ADC Accuracy Definitions 90
13.11 ADC Conversion Result 92
13.12 Register Description 93

14 Programming interface 96

14.1 Features 96
14.2 Overview 96
14.3 Physical Layer of Tiny Programming Interface 96
14.4 Access Layer of Tiny Programming Interface 100
14.5 Instruction Set 102
14.6 Accessing the Non-Volatile Memory Controller 104
14.7 Control and Status Space Register Descriptions 105

15 Memory Programming 107

15.1 Features 107
15.2 Overview 107
15.3 Non-Volatile Memories 108
15.4 Accessing the NVM 111
15.5 Self programming 114
15.6 External Programming 114
15.7 Register Description 115

16 Electrical Characteristics 116

16.1 Absolute Maximum Ratings* 116
16.2 DC Characteristics 116
16.3 Speed Grades 117
16.4 Clock Characteristics 118
16.5 System and Reset Characteristics 119
16.6 Analog Comparator Characteristics 120
16.7 ADC Characteristics (ATtiny5/10, only) 120
16.8 Serial Programming Characteristics 121

17 Typical Characteristics 122

17.1 Supply Current of I/O Modules 122
17.2 ATtiny4/5/9/10 123

18 Register Summary 149

19 Instruction Set Summary 151

20 Ordering Information 153

20.1 ATtiny4 153
20.2 ATtiny5 154
20.3 ATtiny9 155
20.4 ATtiny10 156

21 Packaging Information 157

21.1 6ST1 157

22 Errata 158

22.1 ATtiny4 158
22.2 ATtiny5 158
22.3 ATtiny9 159
22.4 ATtiny10 159

23 Datasheet Revision History 160

23.1 Rev. 8127C – 10/09 160
23.2 Rev. 8127B – 08/09 160
23.3 Rev. 8127A – 04/09 161

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2009 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.