ATtiny25/45/85 Automotive

8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash

DATASHEET

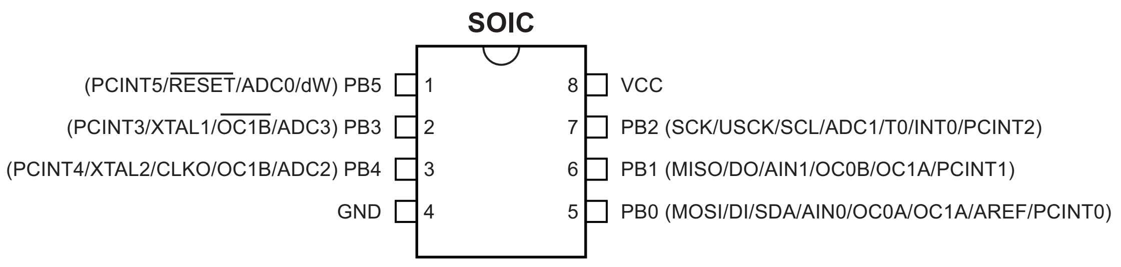
Features

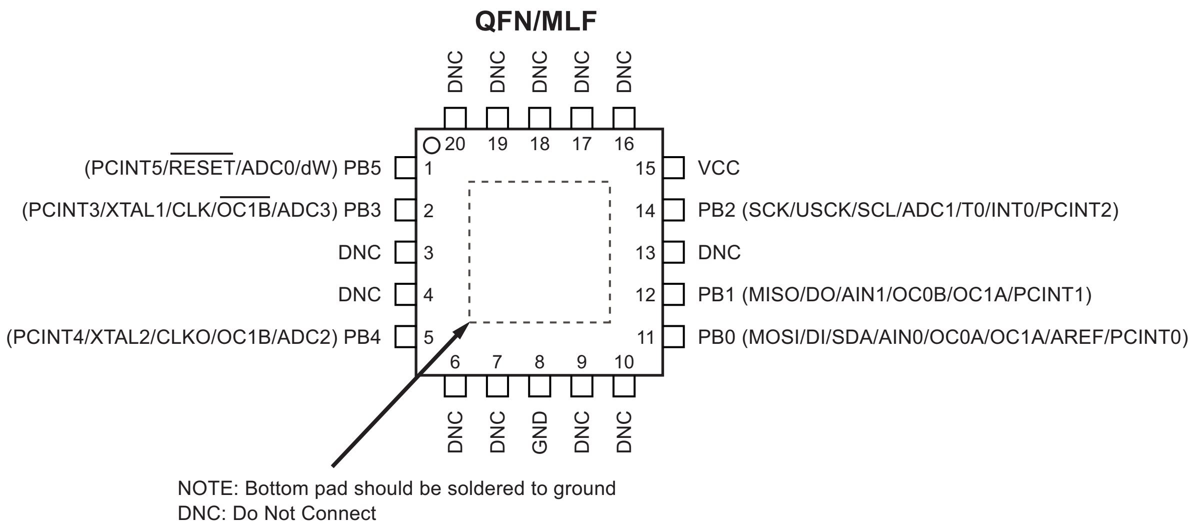
- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 120 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
- Non-volatile program and data memories
  - 2/4/8Kbyte of in-system programmable program memory flash (ATtiny25/45/85)
    - Endurance: 10,000 write/erase cycles
  - 128/256/512 bytes in-system programmable EEPROM (Atmel® ATtiny25/45/85)
    - Endurance: 100,000 write/erase cycles
  - 128/256/512 bytes internal SRAM (ATtiny25/45/85)
  - Programming lock for self-programming flash program and EEPROM data security
- Peripheral features
  - 8-bit Timer/Counter with prescaler and Two PWM channels
  - 8-bit high speed Timer/Counter with separate prescaler
    - 2 High frequency PWM outputs with separate output compare registers
    - Programmable dead time generator
  - Universal serial interface with start condition detector
  - 10-bit ADC
    - 4 Single ended channels
    - 2 Differential ADC channel pairs with programmable gain (1x, 20x)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
- Special microcontroller features
  - debugWIRE on-chip debug system
  - In-system programmable via SPI port
  - External and internal interrupt sources
  - Low power idle, ADC noise reduction, and power-down modes
  - Enhanced power-on reset circuit
  - Programmable brown-out detection circuit
  - Internal calibrated oscillator

- I/O and packages
  - Six programmable I/O lines
  - 8-pin SOIC
  - 20-pin QFN
- Operating voltage
  - 2.7 – 5.5V for Atmel® ATtiny25/45/85
- Speed grade
  - ATtiny25/45/85: 0 to 8MHz at 2.7 to 5.5V, 0 – 16MHz at 4.5 to 5.5V
- Automotive temperature range
  - –40°C to +125°C
- Low Power Consumption
  - Active mode:
    - 1MHz, 2.7V: 300µA
  - Power-down mode:
    - 0.2µA at 2.7V

Pin Configurations

Figure 1. Pinout ATtiny25/45/85


SOIC
(PCINT5/RESET/ADC0/dW) PB5 1
(PCINT3/XTAL1/OC1B/ADC3) PB3 2
(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 3
GND 4
8 VCC
7 PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
6 PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
5 PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)

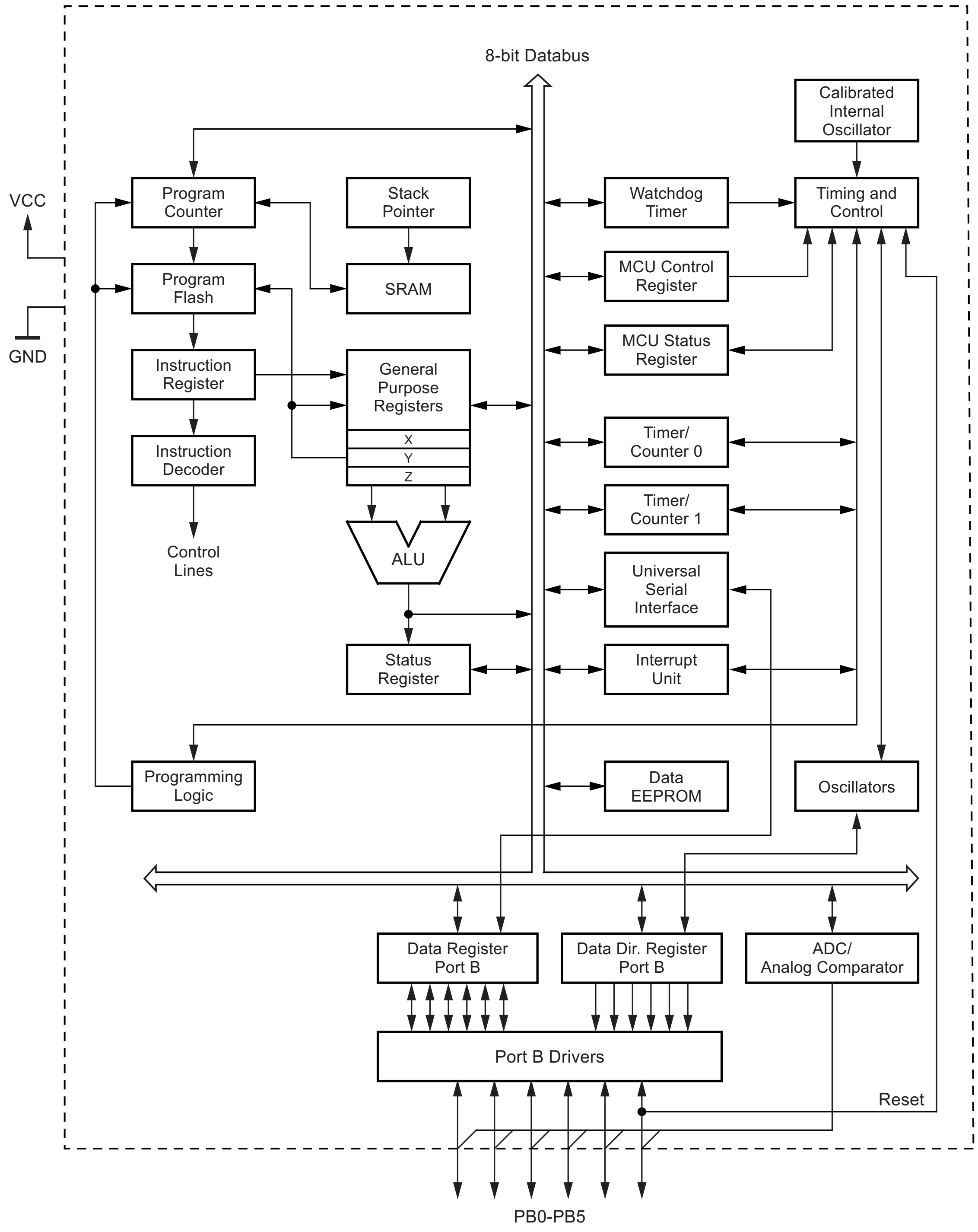

QFN/MLF
DNC 20
DNC 19
DNC 18
DNC 17
DNC 16
(PCINT5/RESET/ADC0/dW) PB5 1
(PCINT3/XTAL1/CLK/OC1B/ADC3) PB3 2
DNC 3
DNC 4
(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 5
15 VCC
14 PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
13 DNC
12 PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
11 PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)
6 DNC
7 DNC
8 GND
9 DNC
10 DNC
NOTE: Bottom pad should be soldered to ground
DNC: Do Not Connect

1. Overview

The Atmel® ATtiny25/45/85 is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny25/45/85 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

1.1 Block Diagram

Figure 1-1. Block Diagram



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

The Atmel® ATtiny25/45/85 provides the following features: 2/4/8K byte of in-system programmable flash, 128/256/512 bytes EEPROM, 128/256/256 bytes SRAM, 6 general purpose I/O lines, 32 general purpose working registers, one 8-bit Timer/Counter with compare modes, one 8-bit high speed Timer/Counter, universal serial interface, internal and external interrupts, a 4-channel, 10-bit ADC, a programmable watchdog timer with internal oscillator, and three software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, analog comparator, and interrupt system to continue functioning. The power-down mode saves the register contents, disabling all chip functions until the next interrupt or hardware reset. The ADC noise reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions.

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

The ATtiny25/45/85 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

1.2 Automotive Quality Grade

The ATtiny25/45/85 have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the ATtiny25/45/85 have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, the products are available in three different temperature grades, but with equivalent quality and reliability objectives. Different temperature identifiers have been defined as listed in Table 1-1.

Table 1-1. Temperature Grade Identification for Automotive Products

Temperature	Temperature Identifier	Comments
–40; +85	T	Similar to industrial temperature grade but with automotive quality
–40; +105	T1	Reduced automotive temperature range
–40; +125	Z	Full automotive temperature range

1.3 Pin Descriptions

1.3.1 VCC

Supply voltage.

1.3.2 GND

Ground.

1.3.3 Port B (PB5..PB0)

Port B is a 6-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the Atmel ATtiny25/45/85 as listed on Section 9.3.2 "Alternate Functions of Port B" on page 50.

1.3.4 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table on page 34. Shorter pulses are not guaranteed to generate a reset.

2. About Code Examples

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

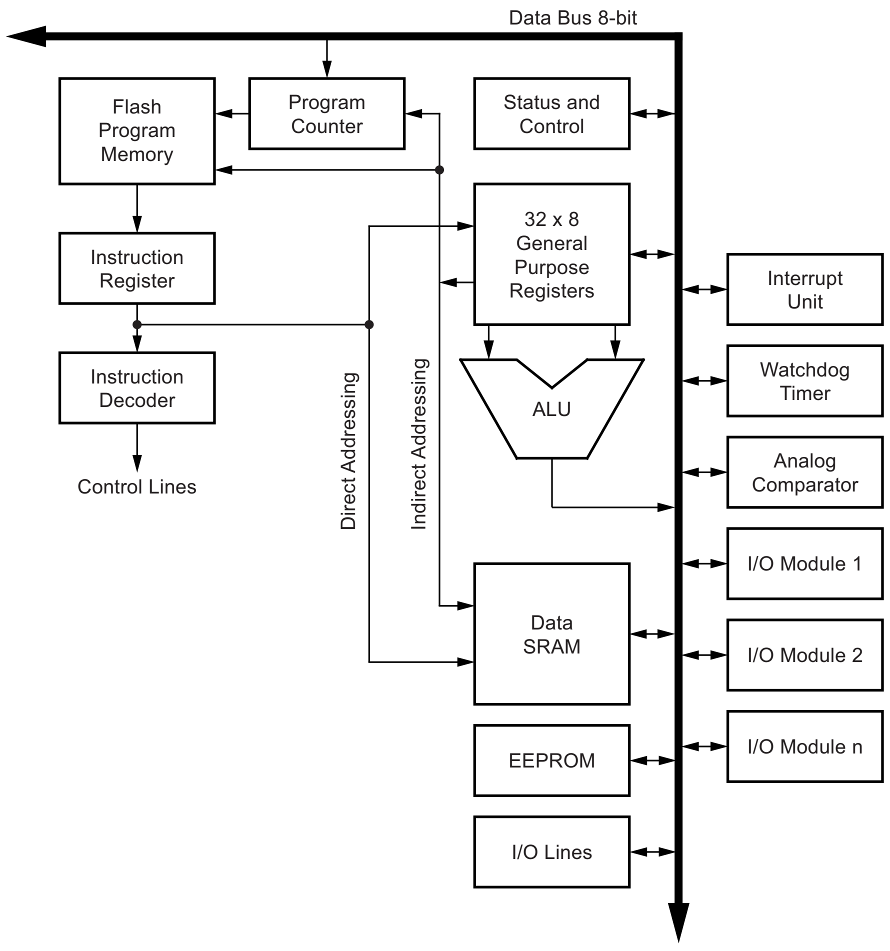
3. AVR CPU Core

3.1 Introduction

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.

3.2 Architectural Overview

Figure 3-1. Block Diagram of the AVR Architecture



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

The fast-access register file contains 32 × 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 32 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, able to directly address the whole address space. Most AVR instructions are 16-bits wide. There are also 32-bit instructions.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the 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 five 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 directly, or as the data space locations following those of the register file, 0x20 – 0x5F.

3.3 ALU – Arithmetic Logic Unit

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

3.4 Status Register

The status register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the status register is updated after all ALU operations, as specified in the instruction set reference. 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.

The AVR status register – SREG – is defined as

Bit	7	6	5	4	3	2	1	0	
	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 instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

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

- **Bit 5 – H: Half Carry Flag**

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

- **Bit 4 – S: Sign Bit, S = N $\oplus$ V**

The S-bit is always an exclusive or between the negative flag N and the two’s complement overflow flag V. See the “instruction set description” for detailed information.

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

The two’s complement overflow flag V supports two’s complement arithmetic. See the “instruction set description” for detailed information.

- **Bit 2 – N: Negative Flag**

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

- **Bit 1 – Z: Zero Flag**

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

- **Bit 0 – C: Carry Flag**

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

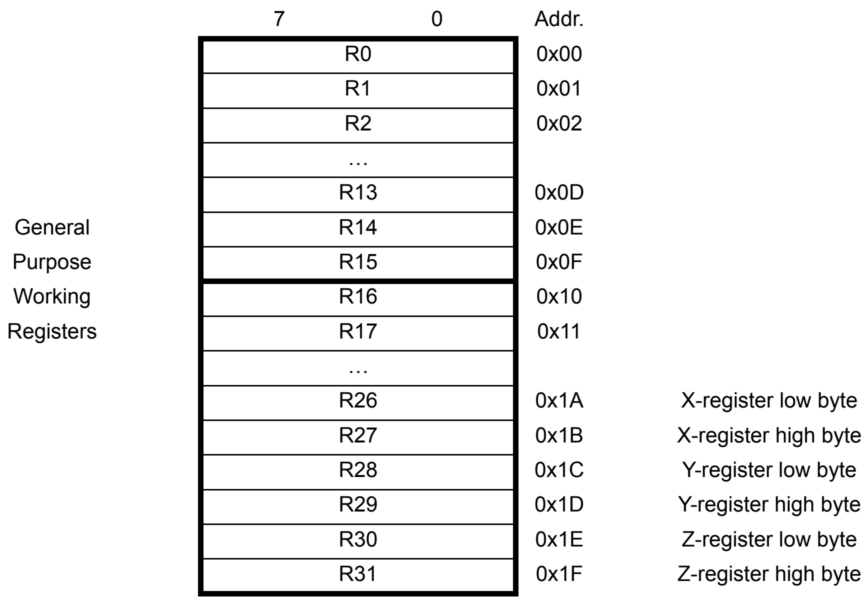
3.5 General Purpose Register File

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

Figure 3-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 3-2. AVR CPU General Purpose Working Registers


7
0
Addr.
General Purpose Working Registers
R0 0x00
R1 0x01
R2 0x02
…
R13 0x0D
R14 0x0E
R15 0x0F
R16 0x10
R17 0x11
…
R26 0x1A X-register low byte
R27 0x1B X-register high byte
R28 0x1C Y-register low byte
R29 0x1D Y-register high byte
R30 0x1E Z-register low byte
R31 0x1F Z-register high byte

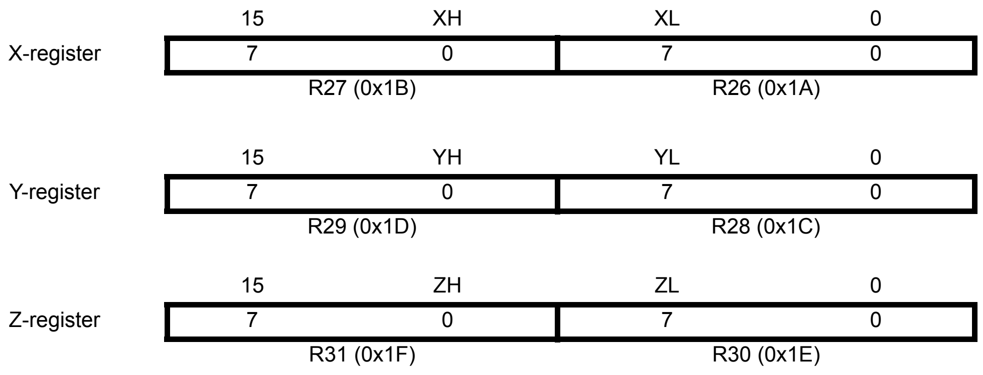
Most of the instructions operating on the register file have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 3-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

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

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

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



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

3.6 Stack Pointer

The stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The stack pointer register always points to the top of the stack. Note that the stack is implemented as growing from higher memory locations to lower memory locations. This implies that a stack PUSH 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 0x60. 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.

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

3.7 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 3-4 shows the parallel instruction fetches and instruction executions enabled by the harvard architecture and the fast access register file concept. This is the basic pipelining concept to obtain up to 1MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 3-4. The Parallel Instruction Fetches and Instruction Executions

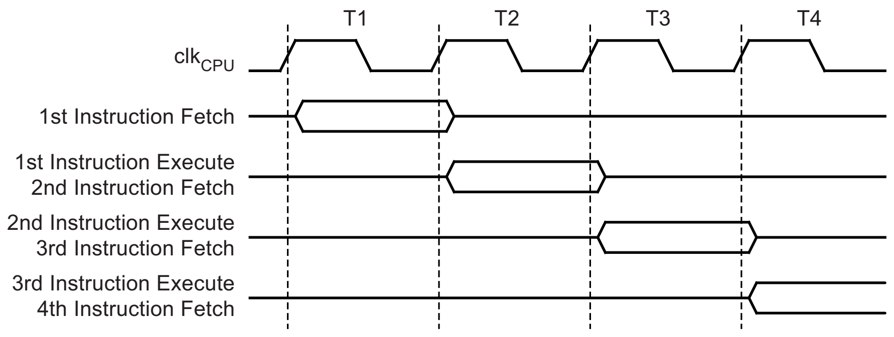
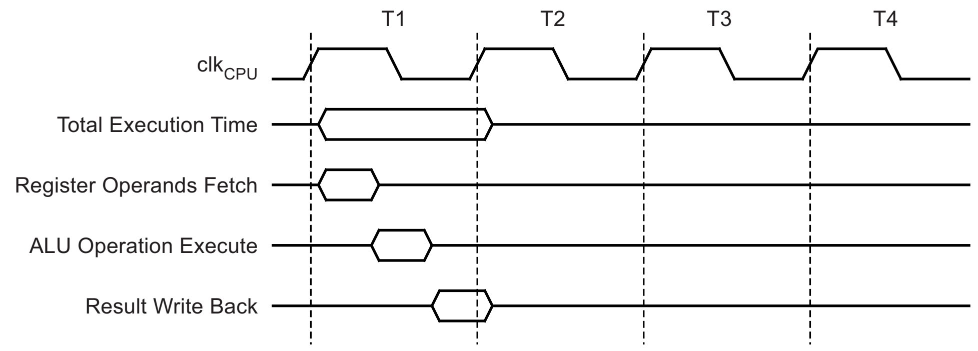


Figure 3-5 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 3-5. Single Cycle ALU Operation



3.8 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 Section 8. "Interrupts" on page 42. 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. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

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

C Code Example

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

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

Assembly Code Example

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

C Code Example

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

3.8.1 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. AVR ATtiny25/45/85 Memories

This section describes the different memories in the ATtiny25/45/85. The AVR architecture has two main memory spaces, the data memory and the program memory space. In addition, the ATtiny25/45/85 features an EEPROM memory for data storage. All three memory spaces are linear and regular.

4.1 In-System Re-programmable Flash Program Memory

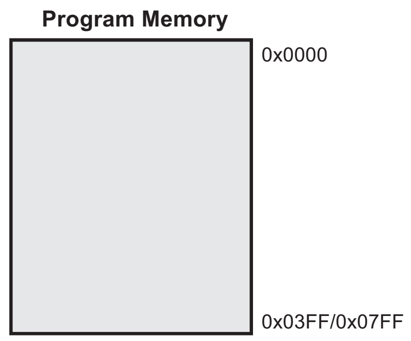
The Atmel® ATtiny25/45/85 contains 2/4/8K byte on-chip in-system reprogrammable flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the flash is organized as 1024/2048/4096 × 16.

The flash memory has an endurance of at least 10,000 write/erase cycles. The ATtiny25/45/85 program counter (PC) is 10/11/12 bits wide, thus addressing the 1024/2048/4096 program memory locations. Section 20. "Memory Programming" on page 123 contains a detailed description on flash data serial downloading using the SPI pins.

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

Timing diagrams for instruction fetch and execution are presented in Section 3.7 "Instruction Execution Timing" on page 9.

Figure 4-1. Program Memory Map



4.2 SRAM Data Memory

Figure 4-2 on page 13 shows how the ATtiny25/45/85 SRAM memory is organized.

The lower 224/352/607 data memory locations address both the register file, the I/O memory and the internal data SRAM. The first 32 locations address the register file, the next 64 locations the standard I/O memory, and the last 128/256/512 locations address the internal data SRAM.

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

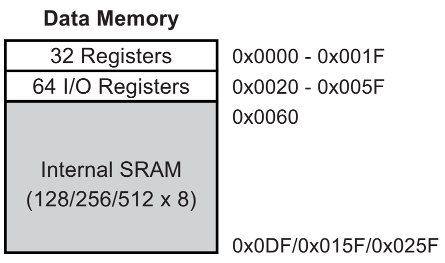
The direct addressing reaches the entire data space.

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

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

The 32 general purpose working registers, 64 I/O registers, and the 128/256/512 bytes of internal data SRAM in the ATtiny25/45/85 are all accessible through all these addressing modes. The register file is described in Section 3.5 "General Purpose Register File" on page 7.

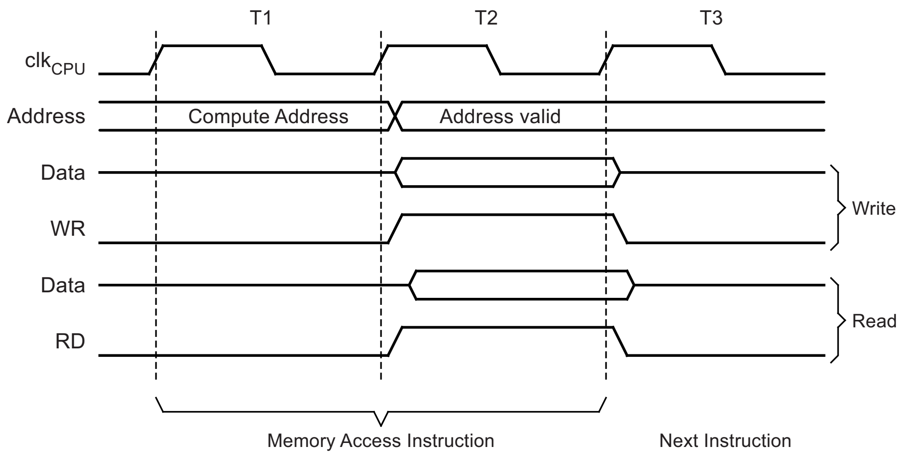
Figure 4-2. Data Memory Map



4.2.1 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 4-3.

Figure 4-3. On-chip Data SRAM Access Cycles



4.3 EEPROM Data Memory

The Atmel® ATtiny25/45/85 contains 128/256/512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register. For a detailed description of serial data downloading to the EEPROM, see Section 20.6 “Serial Downloading” on page 126.

4.3.1 EEPROM Read/Write Access

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

The write access times for the EEPROM are given in Table 4-1 on page 15. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See Section 4.3.10 “Preventing EEPROM Corruption” on page 17 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to Section 4.3.6 “Atomic Byte Programming” on page 15 and Section 4.3.7 “Split Byte Programming” on page 15 for details on this.

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

4.3.2 EEPROM Address Register High – EEARH

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	EEAR8	EEARH
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	X	X	X	X	X	X	X	X	

- **Bit 7..1 – Res6..0: Reserved Bits**

These bits are reserved for future use and will always read as 0 in ATtiny25/45/85.

- **Bits 0 – EEAR8: EEPROM Address**

The EEPROM address register – EEARH – specifies the high EEPROM address in the 128/256/512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 127/255/511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

4.3.3 EEPROM Address Register – EEARL

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

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

The EEPROM address register – EEARL – specifies the low EEPROM address in the 128/256/512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 127/255/511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

4.3.4 EEPROM Data Register – EEDR

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

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

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

4.3.5 EEPROM Control Register – EECR

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

- **Bit 7 – Res: Reserved Bit**

This bit is reserved for future use and will always read as 0 in ATtiny25/45/85. For compatibility with future AVR® devices, always write this bit to zero. After reading, mask out this bit.

- **Bit 6 – Res: Reserved Bit**

This bit is reserved in the Atmel® ATtiny25/45/85 and will always read as zero.

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

The EEPROM programming mode bits setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the erase and Write operations in two different operations. The programming times for the different modes are shown in Table 4-1. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 4-1. EEPROM Mode Bits

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

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

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

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

The EEMPE bit determines whether writing EEPE to one will have effect or not.

When EEMPE is set, setting EEPE within four clock cycles will program the EEPROM at the selected address. If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles.

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

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

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

The EEPROM read enable signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

4.3.6 Atomic Byte Programming

Using atomic byte programming is the simplest mode. When writing a byte to the EEPROM, the user must write the address into the EEAR register and data into EEDR register. If the EEPMn bits are zero, writing EEPE (within four cycles after EEMPE is written) will trigger the erase/write operation. Both the erase and write cycle are done in one operation and the total programming time is given in Table 19-1 on page 122. The EEPE bit remains set until the erase and write operations are completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

4.3.7 Split Byte Programming

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

4.3.8 Erase

To erase a byte, the address must be written to EEAR. If the EEPMn bits are 0b01, writing the EEPE (within four cycles after EEMPE is written) will trigger the erase operation only (programming time is given in Table 19-1 on page 122). The EEPE bit remains set until the erase operation completes. While the device is busy programming, it is not possible to do any other EEPROM operations.

4.3.9 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEPMn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in Table 19-1 on page 122). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated oscillator is used to time the EEPROM accesses. Make sure the oscillator frequency is within the requirements described in Section 5.6.1 "Oscillator Calibration Register – OSCCAL" on page 24.

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

Assembly Code Example

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

C Code Example

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

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

Assembly Code Example

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

C Code Example

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

4.3.10 Preventing EEPROM Corruption

During periods of low V_{CC}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

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

4.4 I/O Memory

The I/O space definition of the ATtiny25/45/85 is shown in Section “” on page 164.

All ATtiny25/45/85 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range 0x00 – 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written 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 the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

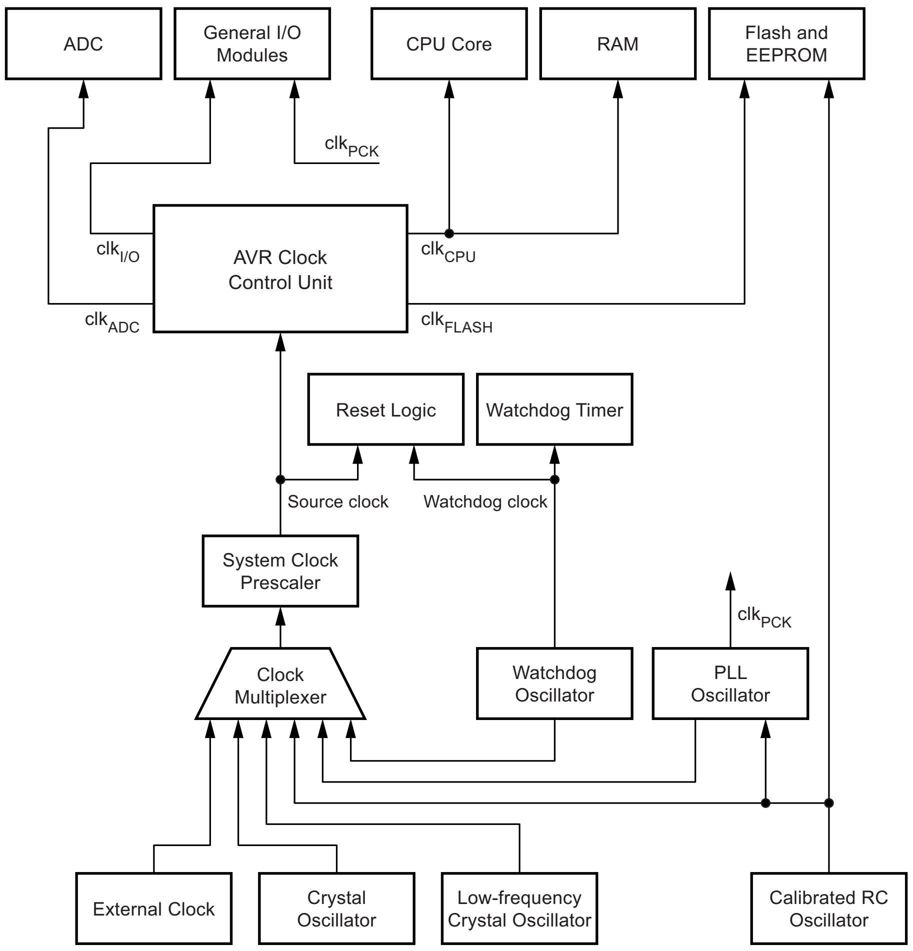
The I/O and peripherals control registers are explained in later sections.

5. System Clock and Clock Options

5.1 Clock Systems and their Distribution

Figure 5-1 presents the principal clock systems in the AVR® and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in Section 6. "Power Management and Sleep Modes" on page 28. The clock systems are detailed below.

Figure 5-1. Clock Distribution



5.1.1 CPU Clock – clk_{CPU}

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

5.1.2 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.

5.1.3 Flash Clock – clk_{FLASH}

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

5.1.4 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.

5.1.5 Internal PLL for Fast Peripheral Clock Generation - clk_{PCK}

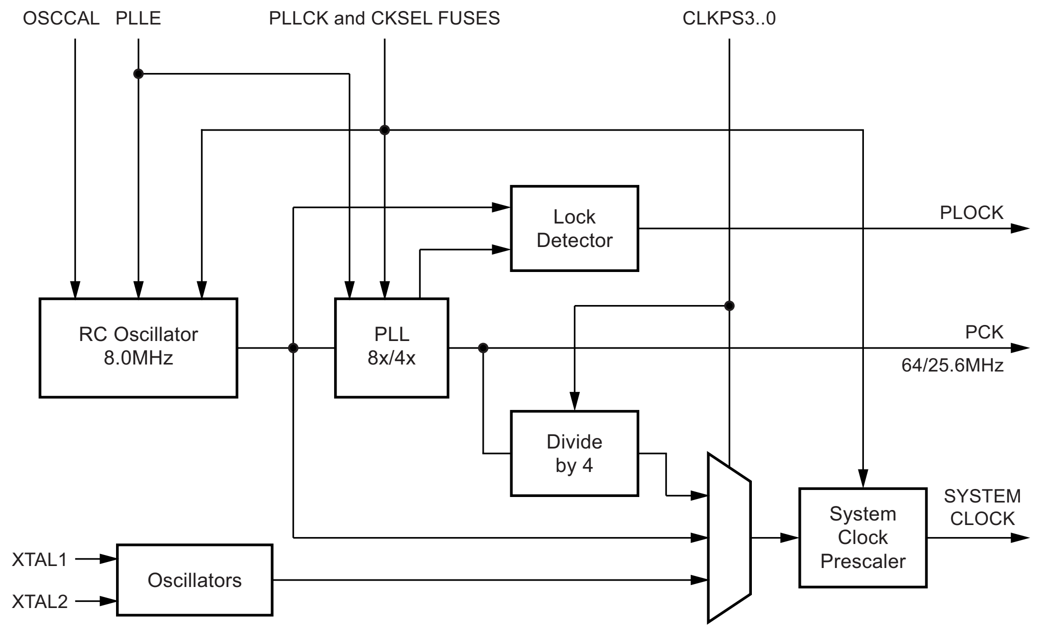
The internal PLL in ATtiny25/45/85 generates a clock frequency that is 8x multiplied from a source input. The source of the PLL input clock is the output of the internal RC oscillator having a frequency of 8.0MHz. Thus the output of the PLL, the fast peripheral clock is 64MHz. The fast peripheral clock, or a clock prescaled from that, can be selected as the clock source for Timer/Counter1. See the Figure 5-2.

The PLL is locked on the RC oscillator and adjusting the RC oscillator via OSCCAL register will adjust the fast peripheral clock at the same time. However, even if the RC oscillator is taken to a higher frequency than 8MHz, the fast peripheral clock frequency saturates at 85MHz (worst case) and remains oscillating at the maximum frequency. It should be noted that the PLL in this case is not locked any longer with the RC oscillator clock.

Therefore, it is recommended not to take the OSCCAL adjustments to a higher frequency than 8MHz in order to keep the PLL in the correct operating range. The internal PLL is enabled only when the PLLE bit in PLLCSR is set or the PLLCK fuse is programmed (‘0’). The bit PLOCK from PLLCSR is set when PLL is locked.

Both internal RC oscillator and PLL are switched off in power down and stand-by sleep modes.

Figure 5-2. PCK Clocking System



5.2 Clock Sources

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

Table 5-1. Device Clocking Options Select[(1)]

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

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

The various choices for each clocking option is given in the following sections. When the CPU wakes up from power-down or power-save, the selected clock source is used to time the start-up, ensuring stable oscillator operation before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before commencing normal operation. The watchdog oscillator is used for timing this real-time part of the start-up time. The number of WDT oscillator cycles used for each time-out is shown in Table 5-2.

Table 5-2. Number of Watchdog Oscillator Cycles

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

5.3 Default Clock Source

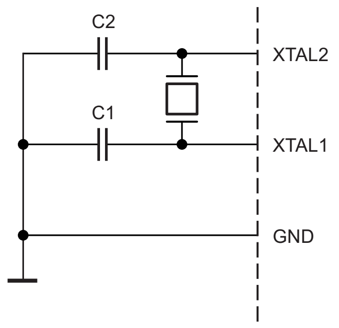
The device is shipped with CKSEL = “0010”, SUT = “10”, and CKDIV8 programmed. The default clock source setting is therefore the internal RC oscillator running at 8MHz with longest start-up time and an initial system clock prescaling of 8. This default setting ensures that all users can make their desired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

5.4 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 5-3. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 5-3. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 5-3. Crystal Oscillator Connections



The oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 5-3.

Table 5-3. Crystal Oscillator Operating Modes

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

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

The CKSEL0 fuse together with the SUT1..0 fuses select the start-up times as shown in Table 5-4.

Table 5-4. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[1]	14CK + 4ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	14CK + 64ms	Ceramic resonator, slowly rising power
0	10	1KCK[2]	14CK	Ceramic resonator, BOD enabled
0	11	1KCK[2]	14CK + 4ms	Ceramic resonator, fast rising power
1	00	1KCK[2]	14CK + 64ms	Ceramic resonator, slowly rising power
1	01	16KCK	14CK	Crystal oscillator, BOD enabled
1	10	16KCK	14CK + 4ms	Crystal oscillator, fast rising power
1	11	16KCK	14CK + 64ms	Crystal oscillator, slowly rising power

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

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

5.5 Low-frequency Crystal Oscillator

To use a 32.768kHz watch crystal as the clock source for the device, the low-frequency crystal oscillator must be selected by setting CKSEL fuses to ‘0110’. The crystal should be connected as shown in Figure 5-3 on page 22. Refer to the 32kHz crystal oscillator application note for details on oscillator operation and how to choose appropriate values for C1 and C2.

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

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

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

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

5.6 Calibrated Internal RC Oscillator

The calibrated internal RC oscillator provides an 8.0MHz clock. The frequency is the nominal value at 3V and 25°C. If the frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 fuse must be programmed in order to divide the internal frequency by 8 during start-up. See Section 5.10 “System Clock Prescaler” on page 26 for more details. This clock may be selected as the system clock by programming the CKSEL fuses as shown in Table 5-6. If selected, it will operate with no external components. During reset, hardware loads the calibration byte into the OSCCAL register and thereby automatically calibrates the RC oscillator. At 3V and 25°C, this calibration gives a frequency within ±1% of the nominal frequency. When this oscillator is used as the chip clock, the watchdog oscillator will still be used for the watchdog timer and for the reset time-out. For more information on the pre-programmed calibration value, see Section 20.4 “Calibration Byte” on page 125.

Table 5-6. Internal Calibrated RC Oscillator Operating Modes

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

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

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

Table 5-7. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

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

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

5.6.1 Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **Bits 7..0 – CAL7..0: Oscillator Calibration Value**

Writing the calibration byte to this address will trim the internal oscillator to remove process variations from the oscillator frequency. This is done automatically during chip reset. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the internal oscillator. Writing 0xFF to the register gives the highest available frequency. The calibrated oscillator is used to time EEPROM and flash access. If EEPROM or flash is written, do not calibrate to more than 8.8MHz frequency. Otherwise, the EEPROM or flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80.

The CAL6..0 bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range. Incrementing CAL6..0 by 1 will give a frequency increment of less than 2% in the frequency range 7.3 to 8.1MHz.

Avoid changing the calibration value in large steps when calibrating the calibrated internal RC oscillator to ensure stable operation of the MCU. A variation in frequency of more than 2% from one cycle to the next can lead to unpredictable behavior. Changes in OSCCAL should not exceed 0x20 for each calibration. It is required to ensure that the MCU is kept in reset during such changes in the clock frequency

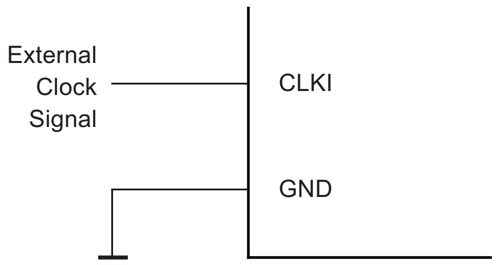
Table 5-8. Internal RC Oscillator Frequency Range

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency	Max Frequency in Percentage of Nominal Frequency
0x00	50%	100%
0x3F	75%	150%
0x7F	100%	200%

5.7 External Clock

To drive the device from an external clock source, CLKI should be driven as shown in Figure 5-4. To run the device on an external clock, the CKSEL fuses must be programmed to “00”.

Figure 5-4. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 5-9.

Table 5-9. Start-up Times for the External Clock Selection

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

Note that the system clock prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to Section 5.10 "System Clock Prescaler" on page 26 for details.

5.7.1 High Frequency PLL Clock - PLL_{CLK}

There is an internal PLL that provides nominally 64MHz clock rate locked to the RC oscillator for the use of the peripheral Timer/Counter1 and for the system clock source. When selected as a system clock source, by programming the CKSEL fuses to '0001', it is divided by four like shown in Table 5-10. When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 5-11. See also Section 5-2 "PCK Clocking System" on page 20.

Table 5-10. PLLCK Operating Modes

CKSEL3..0	Nominal Frequency
0001	16MHz

Table 5-11. Start-up Times for the PLLCK

SUT1..0	Start-up Time from Power Down and Power Save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended usage
00	1KCK	14CK + 8ms	BOD enabled
01	16KCK	14CK + 8ms	Fast rising power
10	1KCK	14CK + 68ms	Slowly rising power
11	16KCK	14CK + 68ms	Slowly rising power

5.8 128 kHz Internal Oscillator

The 128kHz internal oscillator is a low power oscillator providing a clock of 128kHz. The frequency is nominal at 3V and 25°C. This clock may be select as the system clock by programming the CKSEL fuses to "11".

When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 5-12.

Table 5-12. Start-up Times for the 128kHz Internal Oscillator

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

5.9 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT fuse has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. Note that the clock will not be output during reset and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal RC oscillator, can be selected when the clock is output on CLKO. If the system clock prescaler is used, it is the divided system clock that is output.

5.10 System Clock Prescaler

The Atmel® ATtiny25/45/85 system clock can be divided by setting the clock prescale register – CLKPR. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 5-13 on page 27.

5.10.1 Clock Prescale Register – CLKPR

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

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

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

- **Bits 6..4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny25/45/85 and will always read as zero.

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

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 5-13 on page 27.

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

1. Write the clock prescaler change enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

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

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

Table 5-13. Clock Prescaler Select

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

5.10.2 Switching Time

When switching between prescaler settings, the system clock prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than neither the clock frequency corresponding to 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 undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler – even if it were readable, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

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

6. Power Management and Sleep Modes

The high performance and industry leading code efficiency makes the AVR® microcontrollers an ideal choice for low power applications.

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.

To enter any of the three sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM1..0 bits in the MCUCR register select which sleep mode (idle, ADC noise reduction, or power-down) will be activated by the SLEEP instruction. See Table 6-1 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.

Figure 5-1 on page 19 presents the different clock systems in the Atmel ATtiny25/45/85, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

6.1 MCU Control Register – MCUCR

The MCU control register contains control bits for power management.

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

- **Bit 7 – BODS: BOD Sleep**

BOD disable functionality is available in some devices, only. See Section 6.5 "Limitations" on page 29.

In order to disable BOD during sleep (see Table 6-2 on page 29) the BODS bit must be written to logic one. This is controlled by a timed sequence and the enable bit, BODSE in MCUCR. First both BODS and BODSE must be set to one. Second, within four clock cycles, BODS must be set to one and BODSE must be set to zero. The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

In devices where sleeping BOD has not been implemented this bit is unused and will always read zero.

- **Bit 5 – 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.

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

These bits select between the three available sleep modes as shown in Table 6-1.

Table 6-1. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	ADC noise reduction
1	0	Power-down
1	1	Stand-by mode

- **Bit 2 – BODSE: BOD Sleep Enable**

BOD disable functionality is available in some devices, only. See Section 6.5 "Limitations" on page 29.

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

This bit is unused in devices where software BOD disable has not been implemented and will read as zero in those devices.

6.2 Idle Mode

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

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

6.3 ADC Noise Reduction Mode

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

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

6.4 Power-down Mode

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

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 10. "External Interrupts" on page 54 for details.

Table 6-2. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators	Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{PCK}	Main Clock Source Enabled	INT0 and Pin Change	SPM/ EEPROM Ready	USI Start Condition	ADC	Other I/O	Watchdog Interrupt
Idle			X	X	X	X	X	X	X	X	X	X
ADC Noise Reduction				X		X	X[1]	X	X	X		X
Power-down							X[1]		X			X

Note: 1. For INT0, only level interrupt.

6.5 Limitations

BOD disable functionality has been implemented in the following devices, only:

- ATtiny25, revision D, and newer
- ATtiny45, revision D, and newer
- ATtiny85, revision C, and newer

6.6 Power Reduction Register

The power reduction register, PRR, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in idle mode and active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

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

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

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

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

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2- 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.

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

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

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

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

6.7 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR® controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few 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.

6.7.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to Section 17. "Analog to Digital Converter" on page 101 for details on ADC operation.

6.7.2 Analog Comparator

When entering Idle mode, the analog comparator should be disabled if not used. When entering ADC noise reduction mode, the analog comparator should be disabled. In the other sleep modes, the analog comparator is automatically disabled. However, if the analog comparator is set up to use the Internal voltage reference as input, the analog comparator should be disabled in all sleep modes. Otherwise, the internal voltage reference will be enabled, independent of sleep mode. Refer to Section 16. "Analog Comparator" on page 98 for details on how to configure the analog comparator.

6.7.3 Brown-out Detector

If the brown-out detector is not needed in the application, this module should be turned off. If the brown-out detector is enabled by the BODLEVEL fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to Section 7.5 “Brown-out Detection” on page 35 for details on how to configure the brown-out detector.

6.7.4 Internal Voltage Reference

The internal voltage reference will be enabled when needed by the brown-out detection, the analog comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to Section 7.8 “Internal Voltage Reference” on page 37 for details on the start-up time.

6.7.5 Watchdog Timer

If the watchdog timer is not needed in the application, this module should be turned off. If the watchdog timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to Section 7.9 “Watchdog Timer” on page 38 for details on how to configure the watchdog timer.

6.7.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the Section 9.2.5 “Digital Input Enable and Sleep Modes” on page 47 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 Section 16.3.1 “Digital Input Disable Register 0 – DIDR0” on page 100 for details.

7. System Control and Reset

7.1 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 7-1 on page 33 shows the reset logic. Table on page 34 defines the electrical parameters of the reset circuitry.

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.

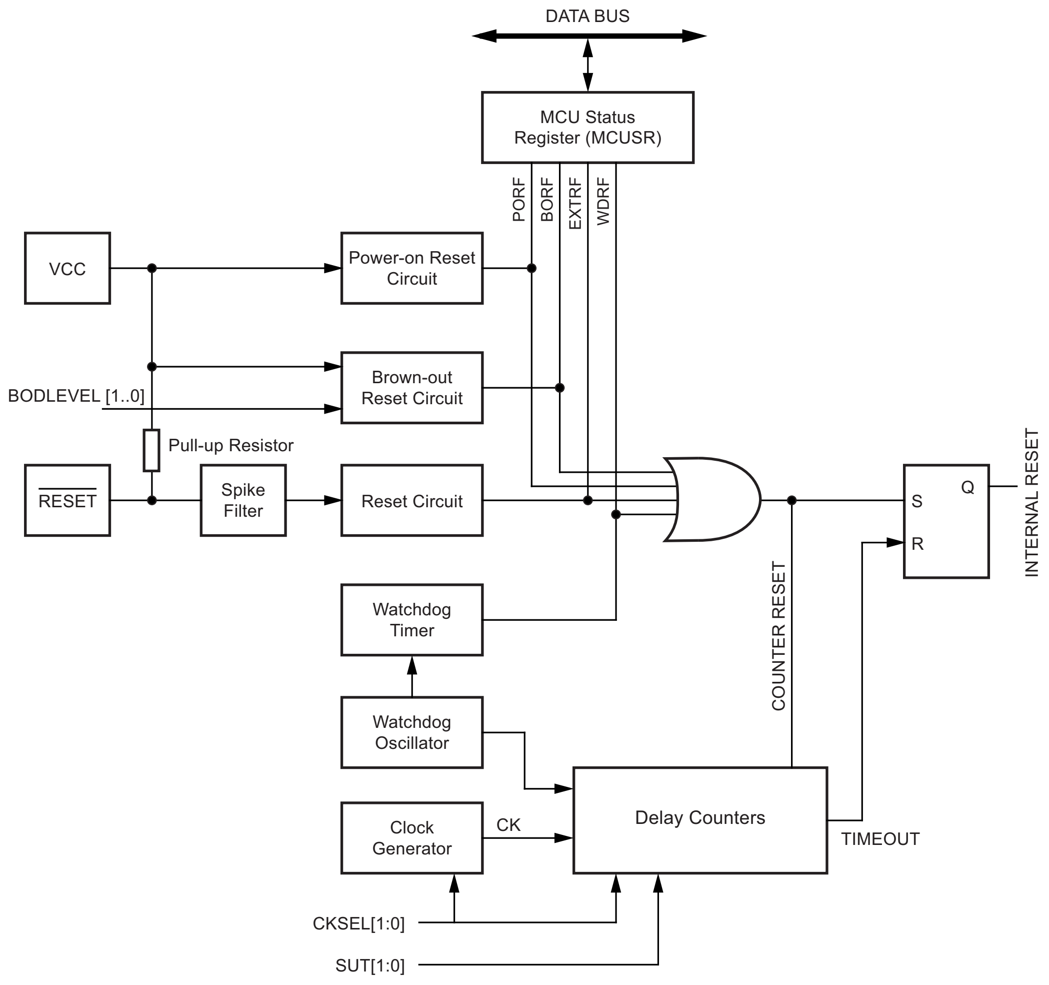
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 5.2 “Clock Sources” on page 21.

7.2 Reset Sources

The Atmel® ATtiny25/45/85 has four sources of reset:

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

Figure 7-1. Reset Logic



7.3 Power-on Reset

A power-on reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in Table . 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 7-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

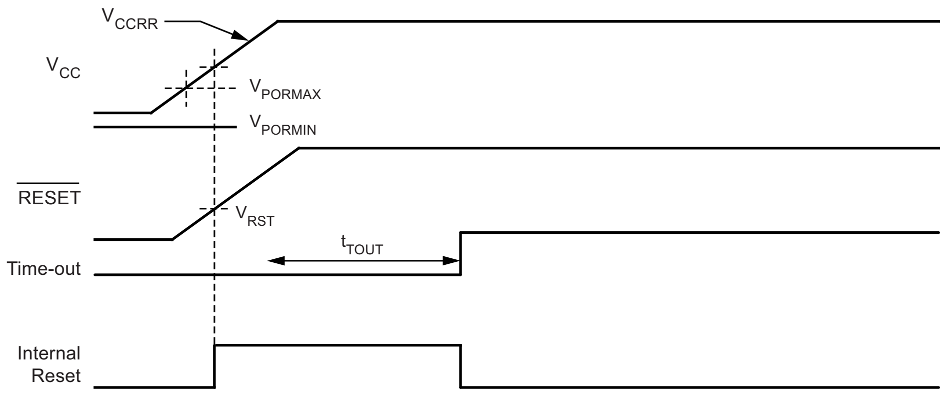


Figure 7-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally

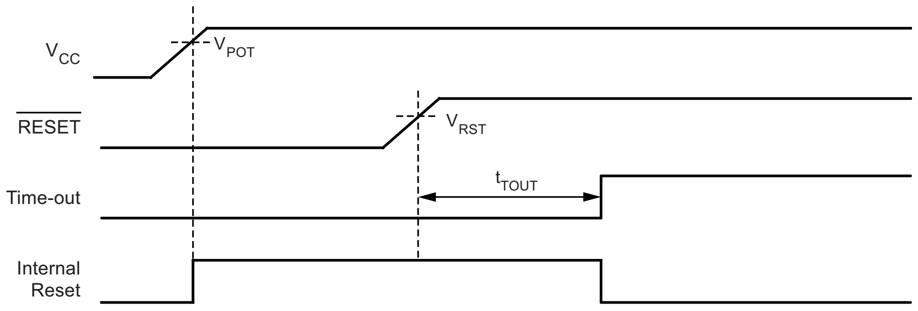


Table 7-1. Power On Reset Specifications

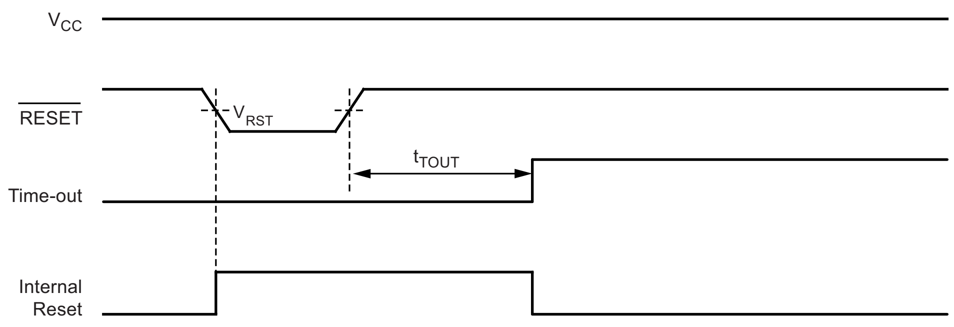
Parameter	Symbol	Min	Typ	Max	Units
Power-on reset threshold voltage (rising)	V_{POT}	1.1	1.4	1.7	V
Power-on reset threshold voltage (falling)(1)		0.8	1.3	1.6	V
VCC Max. start voltage to ensure internal power-on reset signal	V_{PORMAX}			0.4	V
VCC Min. start voltage to ensure internal power-on reset signal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

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

7.4 External Reset

An external reset is generated by a low level on the $\overline{RESET}$ pin if enabled. Reset pulses longer than the minimum pulse width (see Table on page 34) 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 7-4. External Reset During Operation



7.5 Brown-out Detection

ATtiny25/45/85 has an on-chip brown-out detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL fuses. The trigger level has a hysteresis to ensure spike free brown-out detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

Table 7-2. BODLEVEL Fuse Coding[(1)]

BODLEVEL [2..0] Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.0	4.3	4.6	
011		2.3[(2)]		
010		2.2[(2)]		
001		1.9[(2)]		
000		2.0[(2)]		

Notes: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a brown-out reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed.

2. Centered value, not tested.

Table 7-3. Brown-out Characteristics

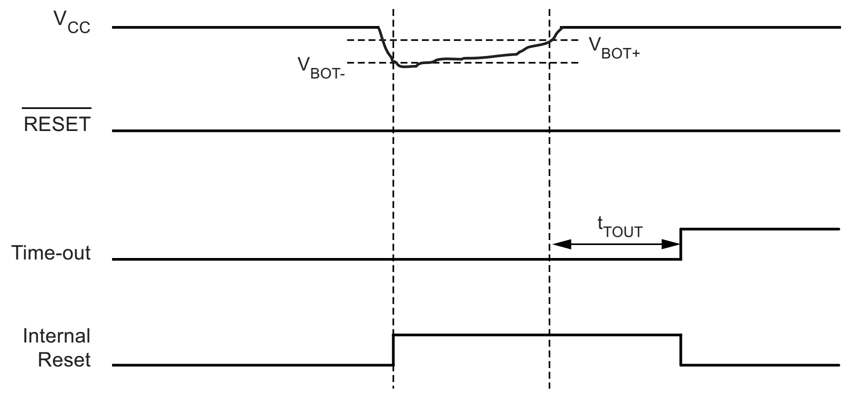
Parameter	Symbol	Min	Typ	Max	Units
RAM retention voltage[(1)]	V_{RAM}		50		mV
Brown-out detector hysteresis	V_{HYST}		50		mV
Min pulse width on brown-out reset	t_{BOD}		2		µs

Note: 1. This is the limit to which VDD can be lowered without losing RAM data

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 7-5), the brown-out reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 7-5), the delay counter starts the MCU after the time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 7-1 on page 34.

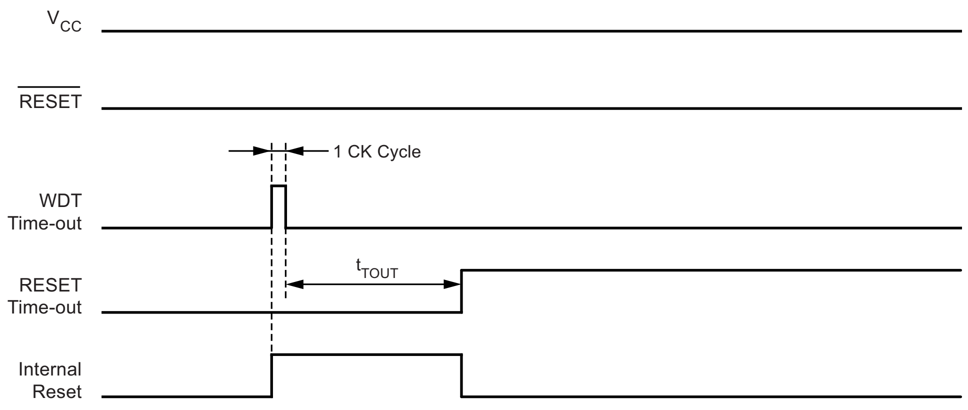
Figure 7-5. Brown-out Reset During Operation



7.6 Watchdog Reset

When the watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the time-out period t_{TOUT}. Refer to Section 7.9 “Watchdog Timer” on page 38 for details on operation of the watchdog timer.

Figure 7-6. Watchdog Reset During Operation



7.7 MCU Status Register – MCUSR

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

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

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 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 2 – BORF: Brown-out Reset Flag**

This bit is set if a brown-out reset occurs. The bit is reset by a power-on reset, or by writing a 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.

7.8 Internal Voltage Reference

Atmel ATtiny25/45/85 features an internal bandgap reference. This reference is used for brown-out detection, and it can be used as an input to the analog comparator or the ADC.

7.8.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 7-4. To save power, the reference is not always turned on. The reference is on during the following situations:

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

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

Table 7-4. Internal Voltage Reference Characteristics

Parameter	Condition	Symbol	Min	Typ	Max	Units
Bandgap reference voltage	V_{CC} = 1.1V/2.7V, T_A = 25°C	V_{BG}	1.0	1.1	1.2	V
Bandgap reference start-up time	V_{CC} = 2.7V, T_A = 25°C	t_{BG}		40	70	µs
Bandgap reference current consumption	V_{CC} = 2.7V, T_A = 25°C	I_{BG}		15		µA

7.9 Watchdog Timer

The watchdog timer is clocked from an on-chip oscillator which runs at 128kHz. By controlling the watchdog timer prescaler, the watchdog reset interval can be adjusted as shown in Table 7-7 on page 40. The WDR – watchdog reset – instruction resets the watchdog timer. The watchdog timer is also reset when it is disabled and when a chip reset occurs. Ten different clock cycle periods can be selected to determine the reset period. If the reset period expires without another watchdog reset, the Atmel® ATtiny25/45/85 resets and executes from the reset vector. For timing details on the watchdog reset, refer to Table 7-7 on page 40.

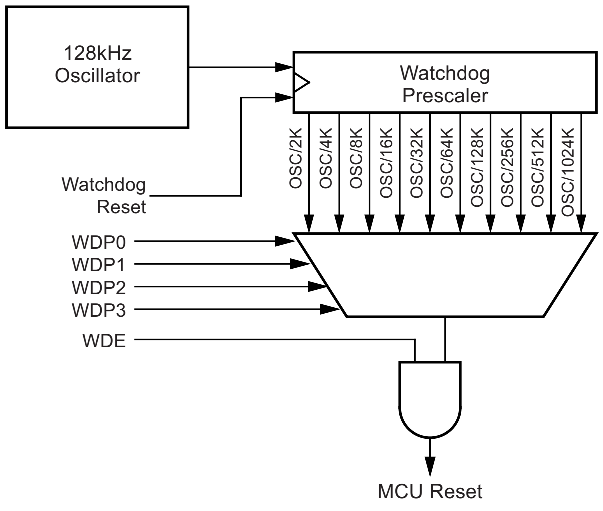
The watchdog timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the watchdog to wake-up from power-down.

To prevent unintentional disabling of the watchdog or unintentional change of time-out period, two different safety levels are selected by the fuse WDTON as shown in Table 7-5 Refer to Section 7.10 "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 41 for details.

Table 7-5. WDT Configuration as a Function of the Fuse Settings of WDTON

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

Figure 7-7. Watchdog Timer



7.9.1 Watchdog Timer Control Register – WDTCR

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

- **Bit 7 – WDIF: Watchdog Timeout Interrupt Flag**

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

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

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

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

Table 7-6. Watchdog Timer Configuration

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

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

This bit must be set when the WDE bit is written to logic zero. Otherwise, the watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a watchdog disable procedure. This bit must also be set when changing the prescaler bits. See Section 7.10 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 41

- **Bit 3 – WDE: Watchdog Enable**

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

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logic 0 to WDE. This disables the watchdog.

In safety level 2, it is not possible to disable the watchdog timer, even with the algorithm described above. See Section 7.10 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 41

In safety level 1, WDE is overridden by WDRF in MCUSR. See Section 7.7 “MCU Status Register – MCUSR” on page 37 for description of WDRF. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared before disabling the Watchdog with the procedure described above. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

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

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

The WDP3..0 bits determine the watchdog timer prescaling when the watchdog timer is enabled. The different prescaling values and their corresponding time-out periods are shown in Table 7-7.

Table 7-7. Watchdog Timer Prescale Select

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

Note: 1. If selected, one of the valid settings below 0b1010 will be used.

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

Assembly Code Example[1]

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

C Code Example[1]

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

Note: 1. The example code assumes that the part specific header file is included.

7.10 Timed Sequences for Changing the Configuration of the Watchdog Timer

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

7.10.1 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 timed sequence is needed when disabling an enabled watchdog timer. To disable an enabled watchdog timer, the following procedure must be followed:

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

7.10.2 Safety Level 2

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

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

8. Interrupts

This section describes the specifics of the interrupt handling as performed in Atmel® ATtiny25/45/85. For a general explanation of the AVR® interrupt handling, refer to Section 3.8 “Reset and Interrupt Handling” on page 10.

8.1 Interrupt Vectors in ATtiny25/45/85

Table 8-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset, watchdog reset
2	0x0001	INT0	External interrupt request 0
3	0x0002	PCINT0	Pin change interrupt request 0
4	0x0003	TIM1_COMPA	Timer/Counter1 compare match A
5	0x0004	TIM1_OVF	Timer/Counter1 overflow
6	0x0005	TIM0_OVF	Timer/Counter0 overflow
7	0x0006	EE_RDY	EEPROM ready
8	0x0007	ANA_COMP	Analog comparator
9	0x0008	ADC	ADC conversion complete
10	0x0009	TIM1_COMPB	Timer/Counter1 compare match B
11	0x000A	TIM0_COMPA	Timer/Counter0 compare match A
12	0x000B	TIM0_COMPB	Timer/Counter0 compare match B
13	0x000C	WDT	Watchdog time-out
14	0x000D	USI_START	USI START
15	0x000E	USI_OVF	USI overflow

If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The most typical and general program setup for the reset and interrupt vector addresses in Atmel ATtiny25/45/85 is:

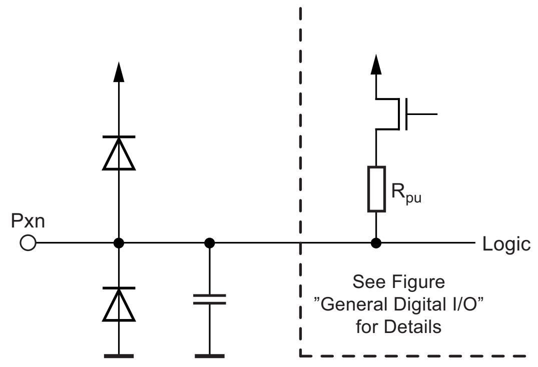
```
Address      Labels Code               Comments
0x0000              rjmp   RESET        ; Reset Handler
0x0001              rjmp   EXT_INT0     ; IRQ0 Handler
0x0002              rjmp   PCINT0       ; PCINT0 Handler
0x0003              rjmp   TIM1_COMPA   ; Timer1 CompareA Handler
0x0004              rjmp   TIM1_OVF     ; Timer1 Overflow Handler
0x0005              rjmp   TIM0_OVF     ; Timer0 Overflow Handler
0x0006              rjmp   EE_RDY       ; EEPROM Ready Handler
0x0007              rjmp   ANA_COMP     ; Analog Comparator Handler
0x0008              rjmp   ADC          ; ADC Conversion Handler
0x0009              rjmp   TIM1_COMPB   ; Timer1 CompareB Handler
0x000A              rjmp   TIM0_COMPA   ;
0x000B              rjmp   TIM0_COMPB   ;
0x000C              rjmp   WDT          ;
0x000D              rjmp   USI_START    ;
0x000E              rjmp   USI_OVF      ;
0x000F       RESET: ldi    r16, low(RAMEND); Main program start
0x0010              ldi    r17, high(RAMEND); Tiny85 has also SPH
0x0011              out    SPL, r16     ; Set Stack Pointer to top of RAM
0x0012              out    SPH, r17     ; Tiny85 has also SPH
0x0013              sei                 ; Enable interrupts
0x0014       <instr> xxx
...    ...   ...    ...
```

9. I/O Ports

9.1 Introduction

All AVR® ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and ground as indicated in Figure 9-1. Refer to Section 21. "Electrical Characteristics" on page 137 for a complete list of parameters.

Figure 9-1. 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 Section 9.4 "Register Description for I/O-Ports" on page 53.

Three I/O memory address locations are allocated for each port, one each for the data register – PORTx, data direction register – DDRx, and the port input Pins – PINx. The port input pins I/O location is read only, while the data register and the data direction register are read/write. However, writing a logic one to a bit in the PINx register, will result in a toggle in the corresponding bit in the data register. In addition, the pull-up disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

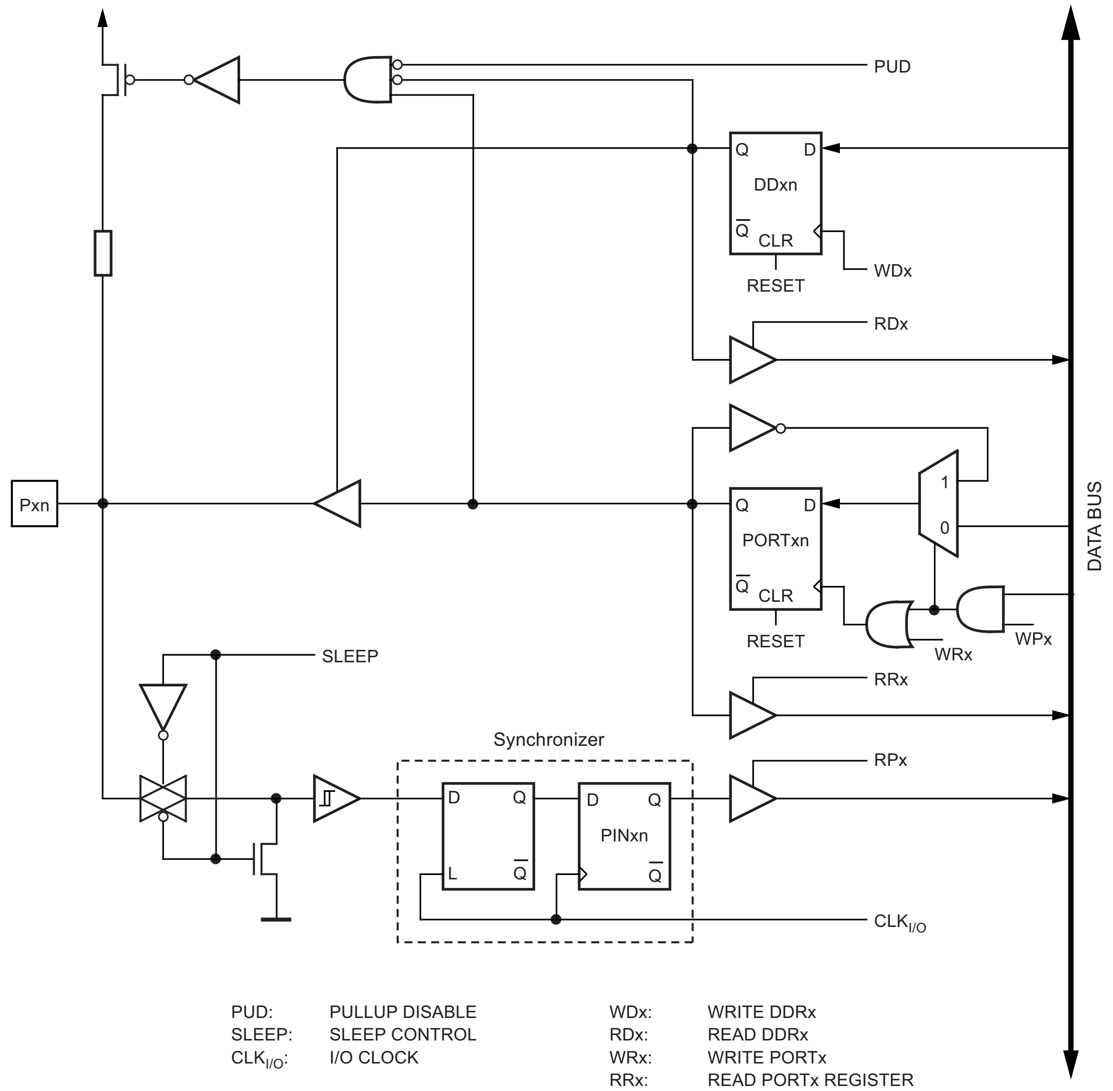
Using the I/O port as general digital I/O is described in Section 9.2 "Ports as General Digital I/O" on page 44. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 9.3 "Alternate Port Functions" on page 48. 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.

9.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 9-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 9-2. General Digital I/O[1]



PUD:	PULLUP DISABLE	WDx:	WRITE DDRx
SLEEP:	SLEEP CONTROL	RDx:	READ DDRx
CLK$_{I/O}$:	I/O CLOCK	WRx:	WRITE PORTx
		RRx:	READ PORTx REGISTER
		RPx:	READ PORTx PIN
		WPx:	WRITE PINx REGISTER

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

9.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in Section 9.4 “Register Description for I/O-Ports” on page 53, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx register selects the direction of this pin. If DDxn is written 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 input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written 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).

9.2.2 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.

9.2.3 Switching Between Input and Output

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

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

Table 9-1 summarizes the control signals for the pin value.

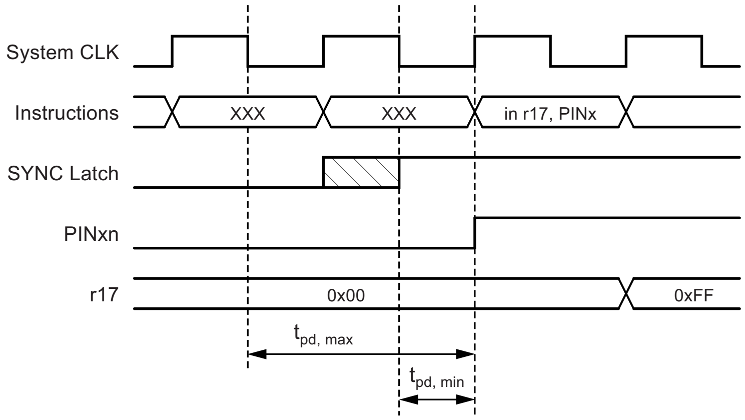
Table 9-1. Port Pin Configurations

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

9.2.4 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 9-2 on page 44, 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 9-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

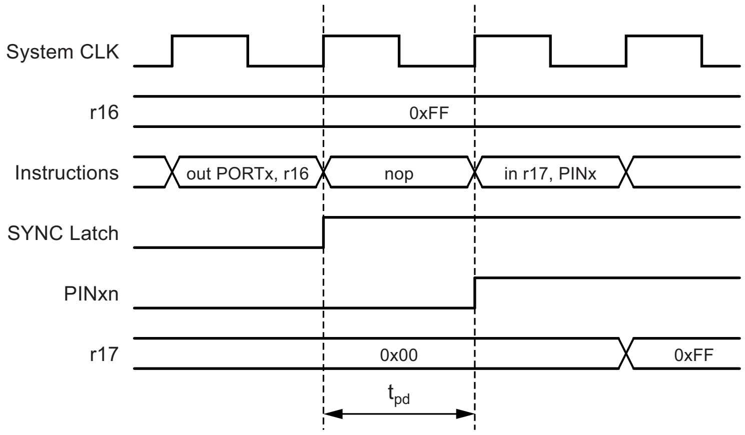
Figure 9-3. Synchronization when Reading an Externally Applied Pin Value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn register at the succeeding positive clock edge. As indicated by the two arrows 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 9-4. 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 9-4. Synchronization when Reading a Software Assigned Pin Value



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

Assembly Code Example[1]

```
              ...
              ; Define pull-ups and set outputs high
              ; Define directions for port pins
              ldi     r16,(1<<PB4)|(1<<PB1)|(1<<PB0)
              ldi     r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
              out     PORTB,r16
              out     DDRB,r17
              ; Insert nop for synchronization
              nop
              ; Read port pins
              in      r16,PINB
              ...
```

C Code Example

```
       unsigned char i;
              ...
              /* Define pull-ups and set outputs high */
              /* Define directions for port pins */
              PORTB = (1<<PB4)|(1<<PB1)|(1<<PB0);
              DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
              /* Insert nop for synchronization*/
              _NOP();
              /* Read port pins */
              i = PINB;
              ...
```

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

9.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 9-2, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 9.3 "Alternate Port Functions" on page 48.

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.

9.2.6 Unconnected Pins

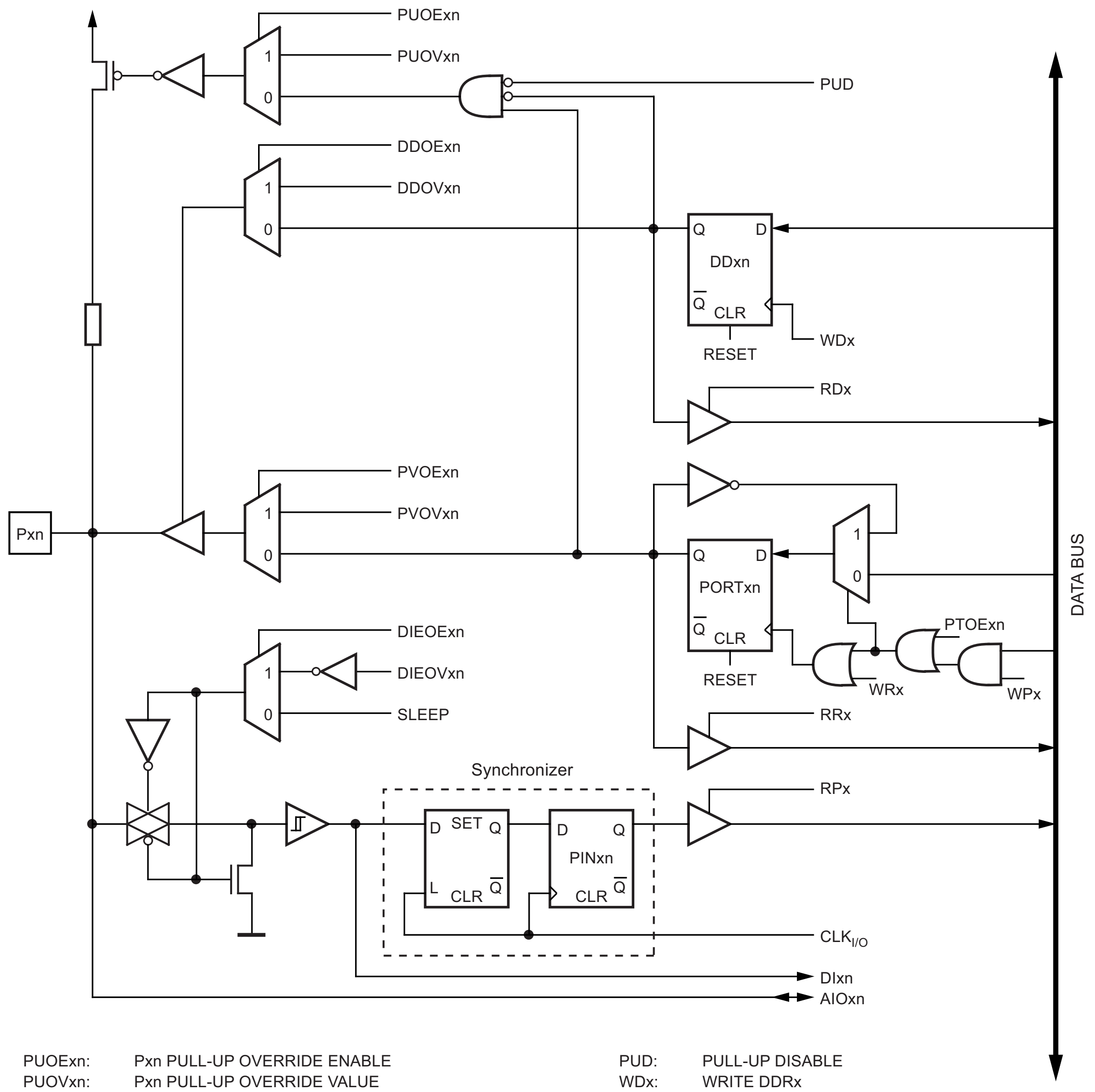
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-5 shows how the port pin control signals from the simplified Figure 9-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR® microcontroller family.

Figure 9-5. Alternate Port Functions(1)



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

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

Table 9-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 9-5 on page 48 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 9-2. Generic Description of Overriding Signals for Alternate Functions

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

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

9.3.1 MCU Control Register – MCUCR

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

- **Bit 6 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See Section 9.2.1 "Configuring the Pin" on page 45 for more details about this feature.

9.3.2 Alternate Functions of Port B

The port B pins with alternate function are shown in Table 9-3.

Table 9-3. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB5	$\overline{\text{RESET}}$ / dW / ADC0 / PCINT5[1]
PB4	XTAL2 / CLKO / ADC2 / OC1B / PCINT4[2]
PB3	XTAL1 / ADC3 / $\overline{\text{OC1B}}$ / PCINT3[3]
PB2	SCK / ADC1 / T0 / USCK / SCL / INT0 / PCINT[4]
PB1	MISO / AIN1 / OC0B / OC1A / DO / PCINT1[4]
PB0	MOSI / AIN0 / OC0A / $\overline{\text{OC1A}}$ / DI / SDA / AREF / PCINT0[6]

Notes:
1. Reset Pin, debugWIRE I/O, ADC input channel or pin change interrupt.
2. XOSC output, divided system clock output, ADC input channel, Timer/Counter1 output compare and PWM output B, or pin change interrupt.
3. XOSC input / external clock input, ADC input channel, Timer/Counter1 inverted output compare and PWM output B, or pin change interrupt.
4. Serial clock input, ADC input channel, Timer/Counter clock input, USI clock (three-wire mode), USI clock (two-wire mode), external interrupt, or pin change interrupt.
5. Serial data input, analog comparator negative input, Timer/Counter0 output compare and PWM Output B, Timer/Counter1 output compare and PWM Output A, USI data output (three-wire mode), or pin change interrupt.
6. Serial data output, analog comparator positive input, Timer/Counter0 output compare and PWM output A, Timer/Counter1 inverted output compare and PWM output A, USI data Input (three-wire mode), USI Data (two-wire mode), Voltage Ref., or pin change interrupt.

- **Port B, Bit 5 - $\overline{\text{RESET}}$/dW/ADC0/PCINT5**

$\overline{\text{RESET}}$: External reset input is active low and enabled by unprogramming ("1") the RSTDISBL fuse. Pull up is activated and output driver and digital input are deactivated when the pin is used as the $\overline{\text{RESET}}$ pin.

dW: When the debugWIRE enable (DWEN) fuse is programmed and lock bits are unprogrammed, the debugWIRE system within the target device is activated. The $\overline{\text{RESET}}$ port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

ADC0: Analog to digital converter, channel 0.

PCINT5: Pin change interrupt source 5.

- **Port B, Bit 4- XTAL2/CLKO/ADC2/OC1B/PCINT4**

XTAL2: Chip clock oscillator pin 2. Used as clock pin for all chip clock sources except internal calibrated RC oscillator and external clock. When used as a clock pin, the pin can not be used as an I/O pin. When using internal calibrated RC oscillator or external clock as a chip clock sources, PB4 serves as an ordinary I/O pin.

CLKO: The divided system clock can be output on the pin PB4. The divided system clock will be output if the CKOUT fuse is programmed, regardless of the PORTB4 and DDB4 settings. It will also be output during reset.

ADC2: Analog to digital converter, channel 2.

OC1B: Output compare match output: The PB4 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB4 set). The OC1B pin is also the output pin for the PWM mode timer function.

PCINT4: Pin change interrupt source 4.

- **Port B, Bit 3 - XTAL1/ADC3/$\overline{\textbf{OC1B}}$/PCINT3**

XTAL1: Chip clock oscillator pin 1. Used for all chip clock sources except internal calibrated RC oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

ADC3: Analog to digital converter, channel 3.

$\overline{\text{OC1B}}$: Inverted output compare match output: The PB3 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB3 set). The $\overline{\text{OC1B}}$ pin is also the inverted output pin for the PWM mode timer function.

PCINT3: Pin change interrupt source 3.

- **Port B, Bit 2 - SCK/ADC1/T0/USCK/SCL/INT0/PCINT2**

SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB2. When the SPI is enabled as a master, the data direction of this pin is controlled by DDPB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

ADC1: Analog to digital converter, channel 1.

T0: Timer/Counter0 counter source.

USCK: Three-wire mode universal serial interface clock.

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

INT0: External interrupt source 0.

PCINT2: Pin change interrupt source 2.

- **Port B, Bit 1 - MISO/AIN1/OC0B/OC1A/DO/PCINT1**

MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB1. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB1. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB1 bit.

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.

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.

OC1A: Output compare match output: The PB1 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB1 set). The OC1A pin is also the output pin for the PWM mode timer function.

DO: Three-wire mode universal serial interface data output. Three-wire mode data output overrides PORTB1 value and it is driven to the port when data direction bit DDB1 is set (one). PORTB1 still enables the pull-up, if the direction is input and PORTB1 is set (one).

PCINT1: Pin change interrupt source 1.

- **Port B, Bit 0 - MOSI/AIN0/OC0A/$\overline{\textbf{OC1A}}$/DI/SDA/AREF/PCINT0**

MOSI: SPI master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB0. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB0. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB0 bit.

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 when configured as an output (DDB0 set (one)). The OC0A pin is also the output pin for the PWM mode timer function.

$\overline{\text{OC1A}}$: Inverted output compare match output: The PB0 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB0 set). The $\overline{\text{OC1A}}$ pin is also the inverted output pin for the PWM mode timer function.

SDA: Two-wire mode serial interface data.

AREF: External analog reference for ADC. Pull up and output driver are disabled on PB0 when the pin is used as an external reference or internal voltage reference with external capacitor at the AREF pin.

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

PCINT0: Pin change interrupt source 0.

Table 9-4 and Table 9-5 relate the alternate functions of port B to the overriding signals shown in Figure 9-5 on page 48.

Table 9-4. Overriding Signals for Alternate Functions in PB5..PB3

Signal Name	PB5/RESET/ADC0/PCINT5	PB4/ADC2/XTAL2/OC1B/PCINT4	PB3/ADC3/XTAL1/_OC1B/PCINT3
PUOE	$\overline{\text{RSTDISBL}}$(1) × DWEN(1)	0	0
PUOV	1	0	0
DDOE	RSTDISBL(1) × DWEN(1)	0	0
DDOV	debugWire transmit	0	0
PVOE	0	OC1B enable	_OC1B enable
PVOV	0	OC1B	_OC1B
PTOE	0	0	0
DIEOE	$\overline{\text{RSTDISBL}}$(1) + (PCINT5 × PCIE + ADC0D)	PCINT4 × PCIE + ADC2D	PCINT3 × PCIE + ADC3D
DIEOV	ADC0D	ADC2D	ADC3D
DI	PCINT5 input	PCINT4 input	PCINT3 input
AIO	RESET input, ADC0 input	ADC2 input	ADC3 input

Note: 1. 1 when the fuse is “0” (programmed).

Table 9-5. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB2/SCK/ADC1/T0/USCK/SCL/INT0/PCINT2	PB1/MISO/DO/AIN1/OC1A/OC0B/PCINT1	PB0/MOSI/DI/SDA/AIN0/AREF/_OC1A/OC0A/PCINT0
PUOE	0	0	0
PUOV	0	0	0
DDOE	USI_TWO_WIRE	0	USI_TWO_WIRE
DDOV	(USI_SCL_HOLD + $\overline{\text{PORTB2}}$) × DDB2	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB0}}$) × DDB0
PVOE	USI_TWO_WIRE × DDB2	OC0B enable + OC1A enable + USI_THREE_WIRE	OC0A enable + _OC1A enable + (USI_TWO_WIRE × DDB0)
PVOV	0	OC0B + OC1A + DO	OC0A + _OC1A
PTOE	USITC	0	0
DIEOE	PCINT2 × PCIE + ADC1D + USISIE	PCINT1 × PCIE + AIN1D	PCINT0 × PCIE + AIN0D + USISIE
DIEOV	ADC1D	AIN1D	AIN0D
DI	T0/USCK/SCL/INT0/PCINT2 input	PCINT1 input	DI/SDA/PCINT0 input
AIO	ADC1 input	Analog comparator negative input	Analog comparator positive input

9.4 Register Description for I/O-Ports

9.4.1 Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	–	–	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

9.4.2 Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	–	–	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

9.4.3 Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	–	–	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	N/A	N/A	N/A	N/A	N/A	

10. External Interrupts

The external interrupts are triggered by the INT0 pin or any of the PCINT5..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT5..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. Pin change interrupts PCI will trigger if any enabled PCINT5..0 pin toggles. The PCMSK register control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT5..0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than idle mode.

The INT0 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU control register – MCUCR. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 requires the presence of an I/O clock, described in Section 5.1 “Clock Systems and their Distribution” on page 19. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than idle mode. The I/O clock is halted in all sleep modes except idle mode.

Note that if a level triggered interrupt is used for wake-up from power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL fuses as described in Section 5. “System Clock and Clock Options” on page 19.

10.1 MCU Control Register – MCUCR

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

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

- **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 10-1. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 10-1. Interrupt 0 Sense Control

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

10.2 General Interrupt Mask Register – GIMSK

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

- **Bits 7, 4..0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

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

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

- **Bit 5 – PCIE: Pin Change Interrupt Enable**

When the PCIE bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt is enabled. Any change on any enabled PCINT5..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI interrupt vector. PCINT5..0 pins are enabled individually by the PCMSK0 register.

10.3 General Interrupt Flag Register – GIFR

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

- **Bits 7, 4..0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

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

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

- **Bit 5 – PCIF: Pin Change Interrupt Flag**

When a logic change on any PCINT5..0 pin triggers an interrupt request, PCIF becomes set (one). If the I-bit in SREG and the PCIE bit in GIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

10.4 Pin Change Mask Register – PCMSK

Bit	7	6	5	4	3	2	1	0	
	–	–	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	1	1	1	1	1	1	

- **Bits 7, 6 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny25/45/85 and will always read as zero.

- **Bits 5..0 – PCINT5..0: Pin Change Enable Mask 5..0**

Each PCINT5..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT5..0 is set and the PCIE bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT5..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11. 8-bit Timer/Counter0 with PWM

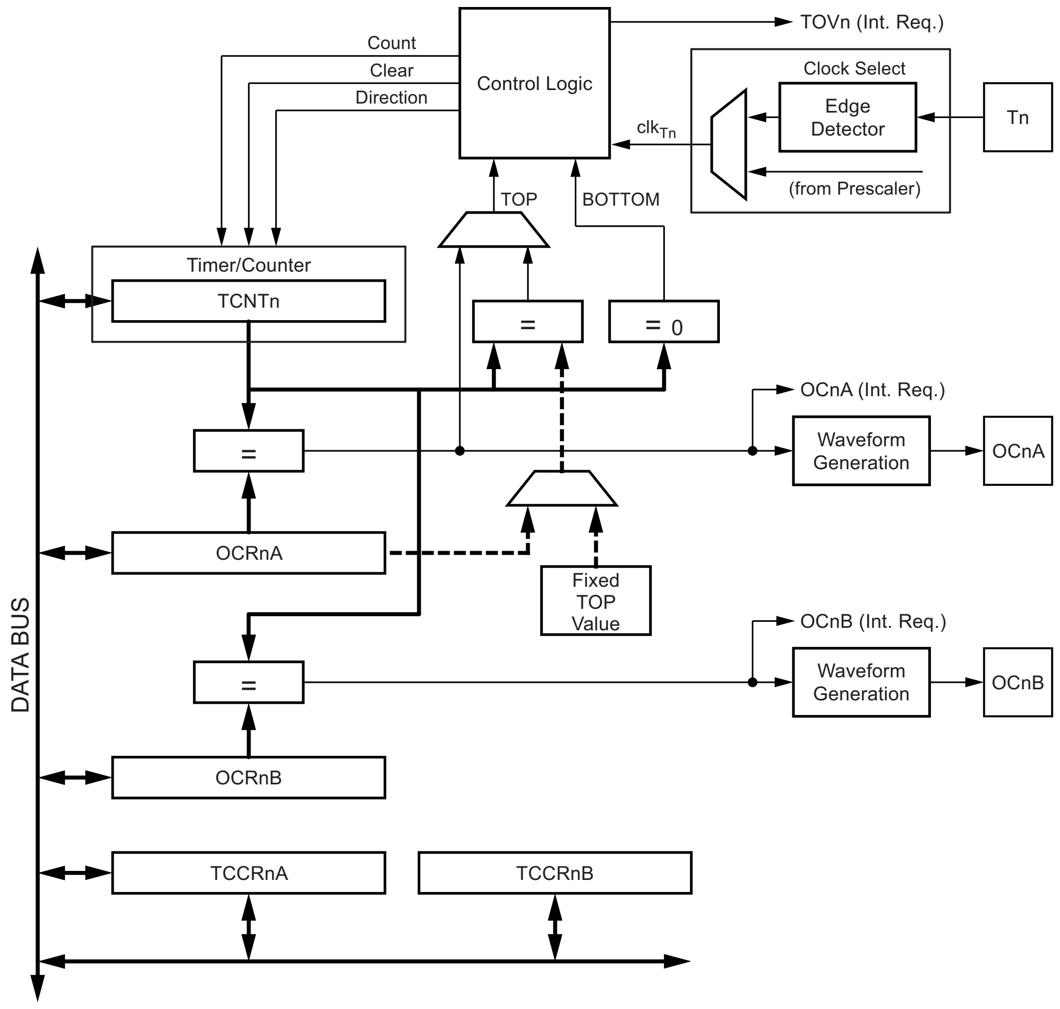
Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent output compare units, and with PWM support. It allows accurate program execution timing (event management) and wave generation. The main features are:

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

11.1 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 11-1. For the actual placement of I/O pins, refer to Figure 1 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 Section 11.8 "8-bit Timer/Counter Register Description" on page 66.

Figure 11-1. 8-bit Timer/Counter Block Diagram



11.1.1 Registers

The Timer/Counter (TCNT0) and output compare registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the timer 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 (clk_{T0}).

The double buffered output compare registers (OCR0A and OCR0B) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins (OC0A and OC0B). See Section 11.4 "Output Compare Unit" on page 58 for details. The compare match event will also set the compare flag (OCF0A or OCF0B) which can be used to generate an output compare interrupt request.

11.1.2 Definitions

Many register and bit references in this section are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 0. A lower case "x" replaces the output compare unit, in this case compare unit A or compare unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions below are also used extensively throughout the document.

Table 11-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A register. The assignment is dependent on the mode of operation.

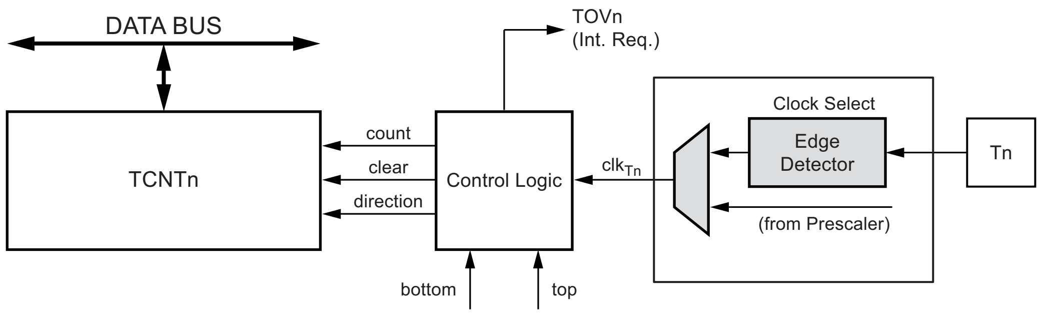
11.2 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS02:0) bits located in the Timer/Counter control register (TCCR0B). For details on clock sources and prescaler, see Section 12. "Timer/Counter Prescaler" on page 72.

11.3 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 11-2 shows a block diagram of the counter and its surroundings.

Figure 11-2. Counter Unit Block Diagram



Signal description (internal signals):

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

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

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter control register (TCCR0A) and the WGM02 bit located in the Timer/Counter control register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare output OC0A. For more details about advanced counting sequences and waveform generation, see Section 11.6 "Modes of Operation" on page 61.

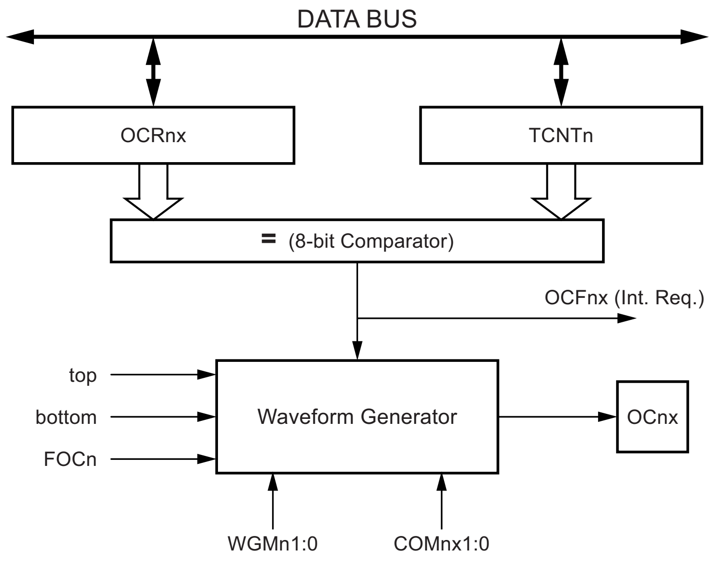
The Timer/Counter overflow flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

11.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the output compare flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM02:0 bits and compare output mode (COM0x1:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (see Section 11.6 "Modes of Operation" on page 61).

Figure 11-3 shows a block diagram of the output compare unit.

Figure 11-3. Output Compare Unit, Block Diagram



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

The OCR0x register access may seem complex, but this is not 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.

11.4.1 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 (FOC0x) bit. Forcing compare match will not set the OCF0x flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the COM0x1:0 bits settings define whether the OC0x pin is set, cleared or toggled).

11.4.2 Compare Match Blocking by TCNT0 Write

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

11.4.3 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 the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down-counting.

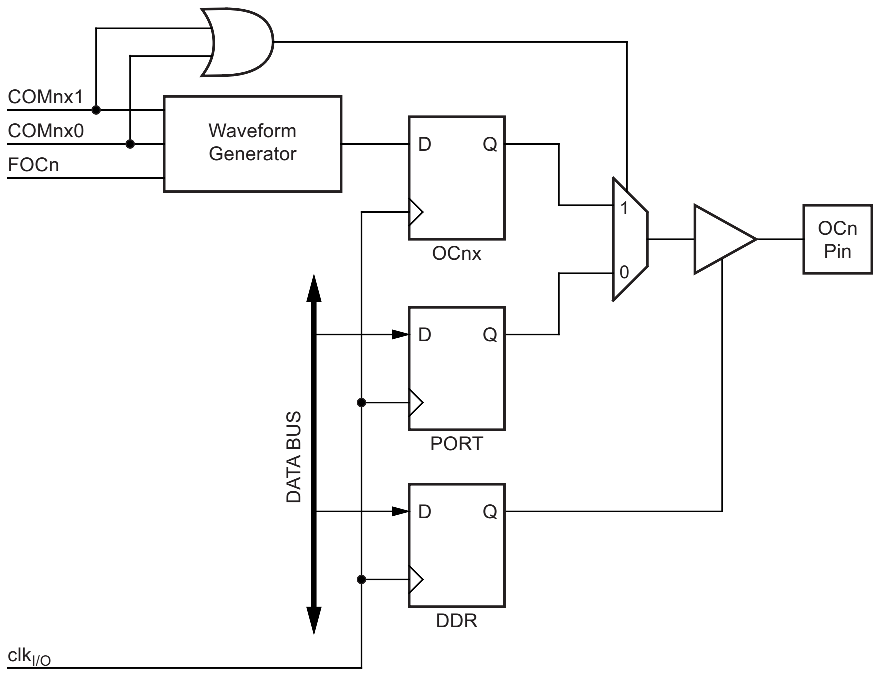
The setup of the OC0x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC0x value is to use the force output compare (FOC0x) strobe bits in normal mode. The OC0x registers keep their values even when changing between waveform generation modes.

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

11.5 Compare Match Output Unit

The compare output mode (COM0x1:0) bits have two functions. The waveform generator uses the COM0x1:0 bits for defining the output compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 11-4 on page 60 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 11-4. Compare Match Output Unit, Schematic



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

The design of the output compare pin logic allows initialization of the OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation. See Section 11.8 “8-bit Timer/Counter Register Description” on page 66

11.5.1 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 66. For fast PWM mode, refer to Table 11-3 on page 67, and for phase correct PWM refer to Table 11-4 on page 67.

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 FOC0x strobe bits.

11.6 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM02:0) and compare output mode (COM0x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM0x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (see Section 11.5 “Compare Match Output Unit” on page 60).

For detailed timing information refer to Figure 11-8 on page 65, Figure 11-9 on page 65, Figure 11-10 on page 65 and Figure 11-11 on page 66 in Section 11.7 “Timer/Counter Timing Diagrams” on page 65.

11.6.1 Normal Mode

The simplest mode of operation is the normal mode (WGM02:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter overflow flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

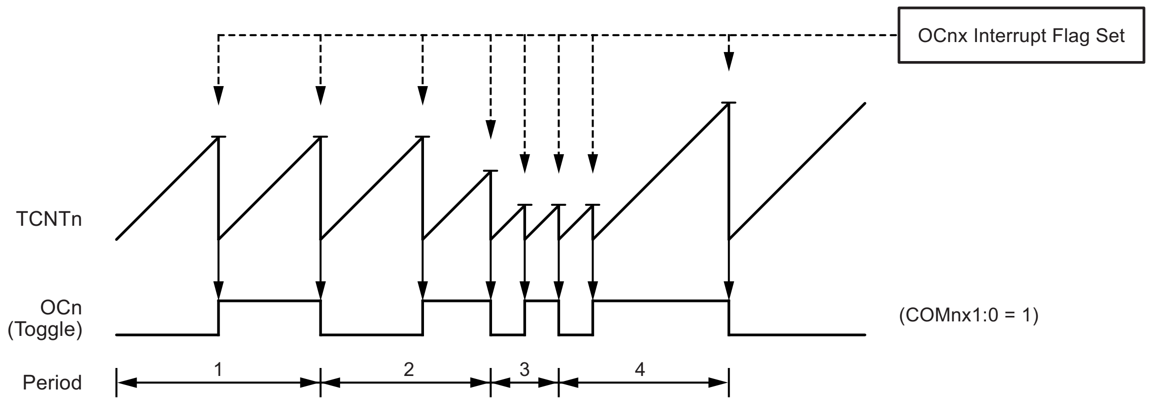
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

11.6.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM02:0 = 2), the OCR0A register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 11-5 on page 61. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 11-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with 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 is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

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

The *N* variable represents the prescale 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 0x00.

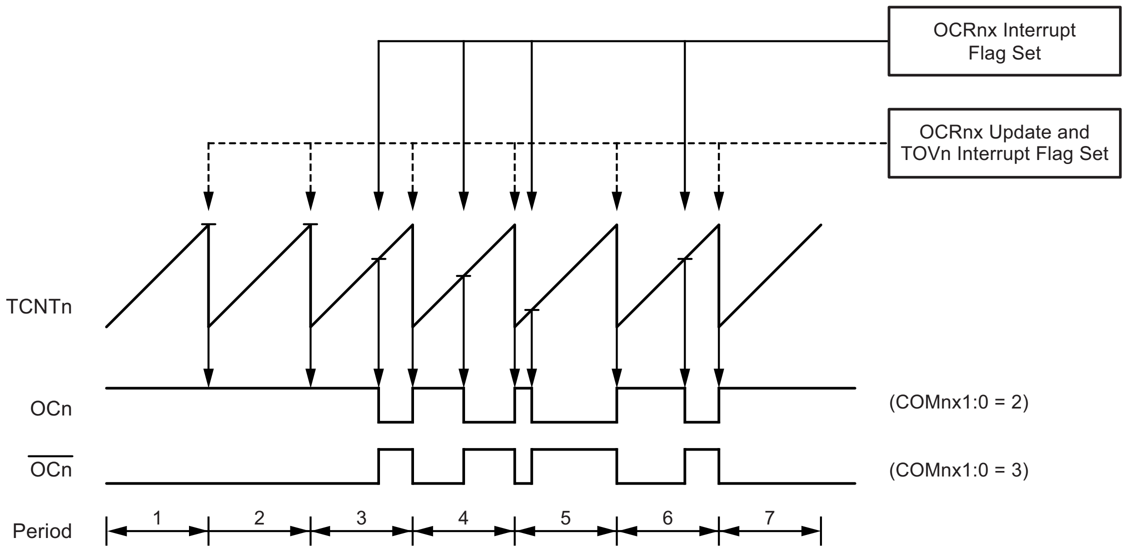
11.6.3 Fast PWM Mode

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

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 11-6 on page 62. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 11-6. Fast PWM Mode, Timing Diagram



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

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A1:0 bits to one allows the AC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 11-3 on page 67). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

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

The extreme values for the OCR0A register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle.

Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

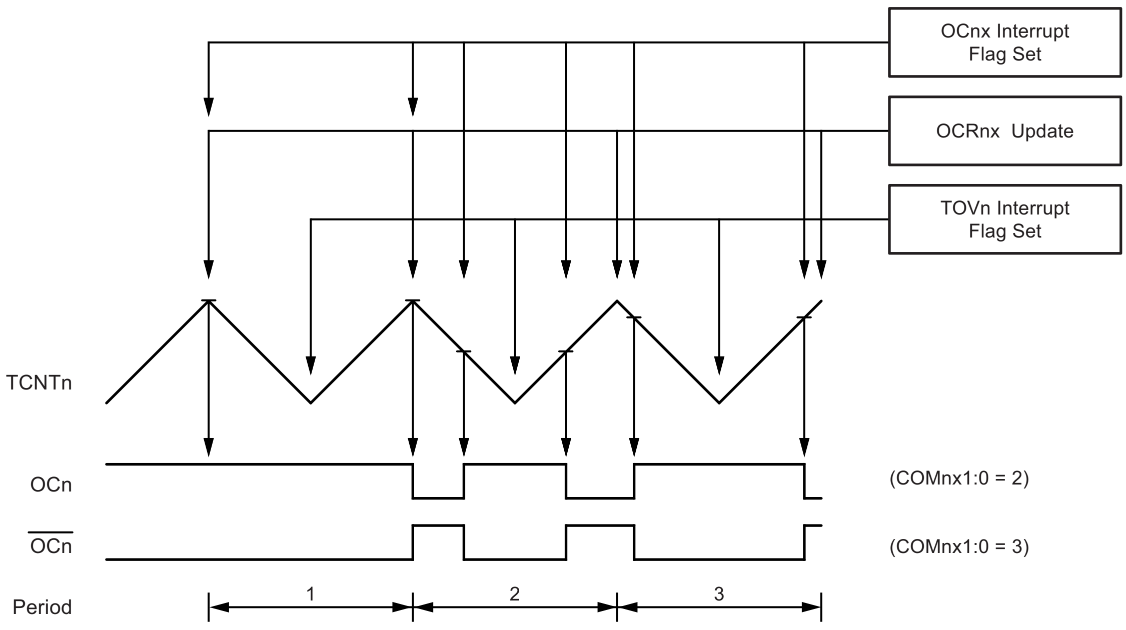
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each compare match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

11.6.4 Phase Correct PWM Mode

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

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 11-7 on page 64. 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.

Figure 11-7. Phase Correct PWM Mode, Timing Diagram



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

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A0 bits to one allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 11-4 on page 67). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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

The extreme values for the OCR0A register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 11-7 on page 64 OCn has a transition from high to low even though there is no compare match. The point of this transition is to guaratee symmetry around BOTTOM. There are two cases that give a transition without compare match.

- OCR0A changes its value from MAX, like in Figure 11-7 on page 64. When the OCR0A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting compare match.
- The timer starts counting from a value higher than the one in OCR0A, and for that reason misses the compare match and hence the OCn change that would have happened on the way up.

11.7 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. Figure 11-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 11-8. Timer/Counter Timing Diagram, no Prescaling

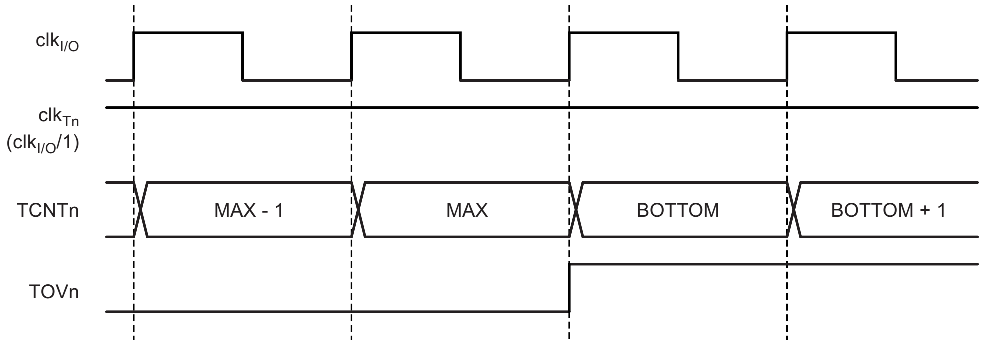


Figure 11-9 shows the same timing data, but with the prescaler enabled.

Figure 11-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

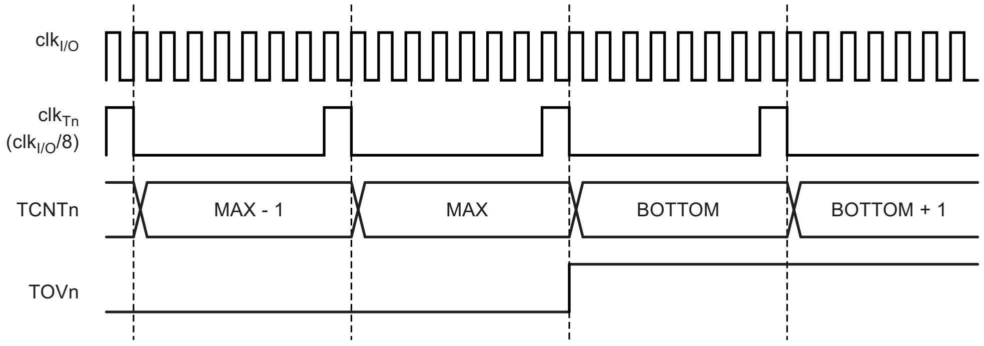


Figure 11-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 11-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

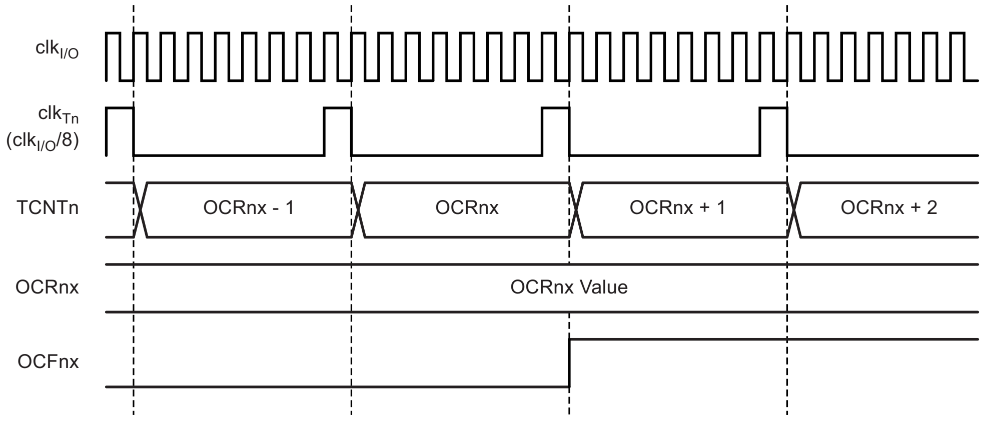
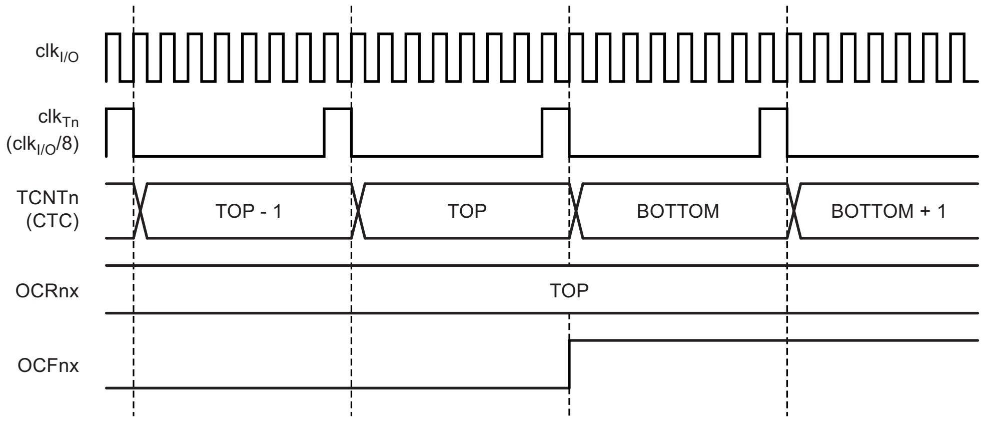


Figure 11-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

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



11.8 8-bit Timer/Counter Register Description

11.8.1 Timer/Counter Control Register A – TCCR0A

Bit	7	6	5	4	3	2	1	0	
	COM0A**1**	COM0A**0**	COM0B**1**	COM0B**0**	–	–	**WGM01**	**WGM00**	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM01A:0: Compare Match Output A Mode**

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

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 11-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 11-2. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 11-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 11-3. Compare Output Mode, Fast PWM Mode[(1)]

COM01	COM00	Description
0	0	Normal port operation, OC0A disconnected.

0	1	WGM02 = 0: Normal Port Operation, OC0A disconnected.
		WGM02 = 1: Toggle OC0A on compare match.

1	0	Clear OC0A on compare match, set OC0A at TOP
1	1	Set OC0A on compare match, clear OC0A at TOP

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 11.6.3 “Fast PWM Mode” on page 62 for more details.

Table 11-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 11-4. Compare Output Mode, Phase Correct PWM Mode[(1)]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.

0	1	WGM02 = 0: Normal port operation, OC0A disconnected.
		WGM02 = 1: Toggle OC0A on compare match.

1	0	Clear OC0A on compare match when up-counting. Set OC0A on compare match when down-counting.
1	1	Set OC0A on compare match when up-counting. Clear OC0A on compare match when down-counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 11.6.4 “Phase Correct PWM Mode” on page 63 for more details.

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

These bits control the output compare pin (OC0B) behavior. If one or both of the COM0B1:0 bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting. Table 11-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 11-5. Compare Output Mode, non-PWM Mode

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

Table 11-3 on page 67 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 11-6. Compare Output Mode, Fast PWM Mode[1]

COM01	COM00	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match, set OC0B at TOP
1	1	Set OC0B on compare match, clear OC0B at TOP

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 11.6.3 "Fast PWM Mode" on page 62 for more details.

Table 11-4 on page 67 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 11-7. Compare Output Mode, Phase Correct PWM Mode[1]

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

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 11.6.4 "Phase Correct PWM Mode" on page 63 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

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

Combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 11-8 on page 68. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 11.6 "Modes of Operation" on page 61).

Table 11-8. Waveform Generation Mode Bit Description

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

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

11.8.2 Timer/Counter Control Register B – TCCR0B

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

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

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

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

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

The FOC0A bit is always read as zero.

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

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

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

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

The FOC0B bit is always read as zero.

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

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

See the description in the Section 11.8.1 "Timer/Counter Control Register A – TCCR0A" on page 66.

- **Bits 2:0 – CS02:0: Clock Select**

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

Table 11-9. Clock Select Bit Description

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

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

11.8.3 Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. writing to the TCNT0 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0x registers.

11.8.4 Output Compare Register A – OCR0A

Bit	7	6	5	4	3	2	1	0	
	OCR0A[7:0]								OCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0A pin.

11.8.5 Output Compare Register B – OCR0B

Bit	7	6	5	4	3	2	1	0	
	OCR0B[7:0]								OCR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The output compare register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0B pin.

11.8.6 Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	–	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	–	**TIMSK**
Read/Write	R	R	R	R	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..4, 0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

- **Bit 3 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

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

- **Bit 2 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

- **Bit 1 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

11.8.7 Timer/Counter 0 Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	–	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	–	**TIFR**
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny25/45/85 and will always read as zero.

- **Bit 4– OCF0A: Output Compare Flag 0 A**

The OCF0A bit is set when a compare match occurs between the Timer/Counter0 and the data in OCR0A – output compare register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 compare match interrupt enable), and OCF0A are set, the Timer/Counter0 compare match interrupt is executed.

- **Bit 3 – OCF0B: Output Compare Flag 0 B**

The OCF0B bit is set when a compare match occurs between the Timer/Counter and the data in OCR0B – output compare register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter compare B match interrupt enable), and OCF0B are set, the Timer/Counter compare match interrupt is executed.

- **Bit 1 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 overflow interrupt enable), and TOV0 are set, the Timer/Counter0 overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to Table 11-8 on page 68, Section 11-8 "Waveform Generation Mode Bit Description" on page 68.

12. Timer/Counter Prescaler

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

12.1 Prescaler Reset

The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter. 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 (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

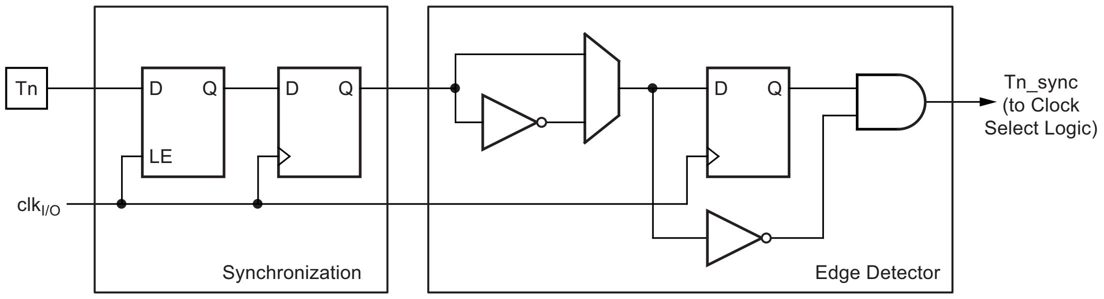
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution.

12.2 External Clock Source

An external clock source applied to the T0 pin can be used as Timer/Counter clock (clk_{T0}). The T0 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 12-1 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 (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 12-1. T0 Pin Sampling



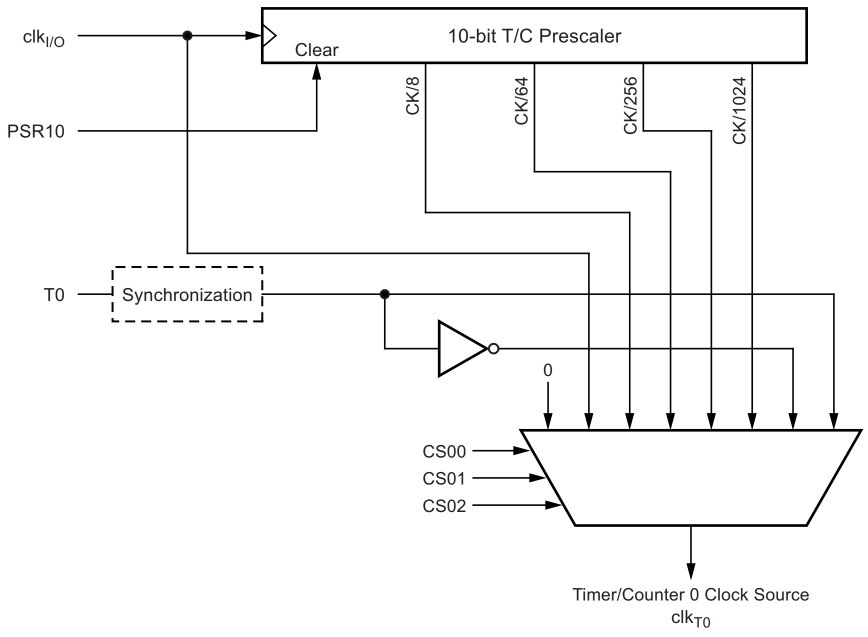
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T0 pin to the counter is updated.

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.

Figure 12-2. Prescaler for Timer/Counter0



Note: 1. The synchronization logic on the input pins (T0) is shown in Figure 12-1.

12.2.1 General Timer/Counter Control Register – GTCCR

Bit	7	6	5	4	3	2	1	0	
	TSM	PWM1B	COM1B1	COM1B0	FOC1B	FOC1A	PSR1	PSR0	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 PSR0 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 PSR0 bit is cleared by hardware, and the Timer/Counter start counting.

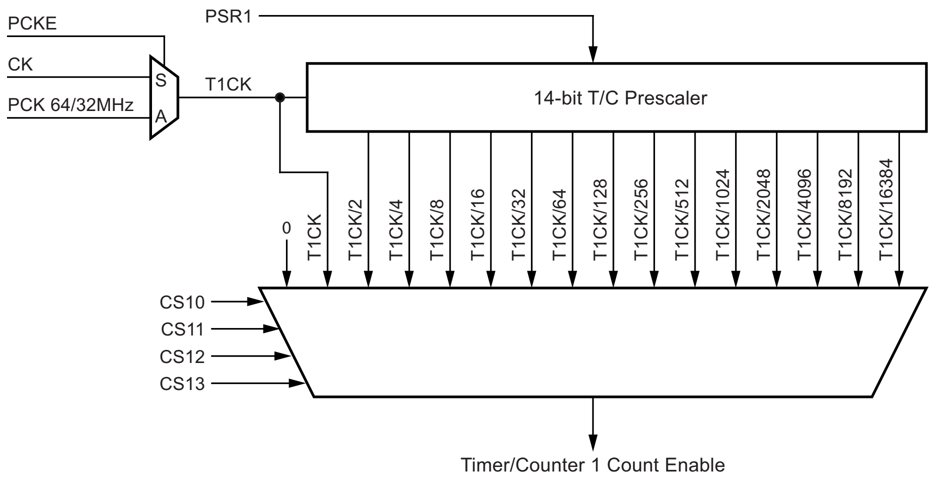
- **Bit 0 – PSR0: Prescaler Reset Timer/Counter0**

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.

13. Counter and Compare Units

Figure 13-1 shows the Timer/Counter1 prescaler that supports two clocking modes, a synchronous clocking mode and an asynchronous clocking mode. The synchronous clocking mode uses the system clock (CK) as the clock timebase and asynchronous mode uses the fast peripheral clock (PCK) as the clock time base. The PCKE bit from the PLLCSR register enables the asynchronous mode when it is set (‘1’).

Figure 13-1. Timer/Counter1 Prescaler



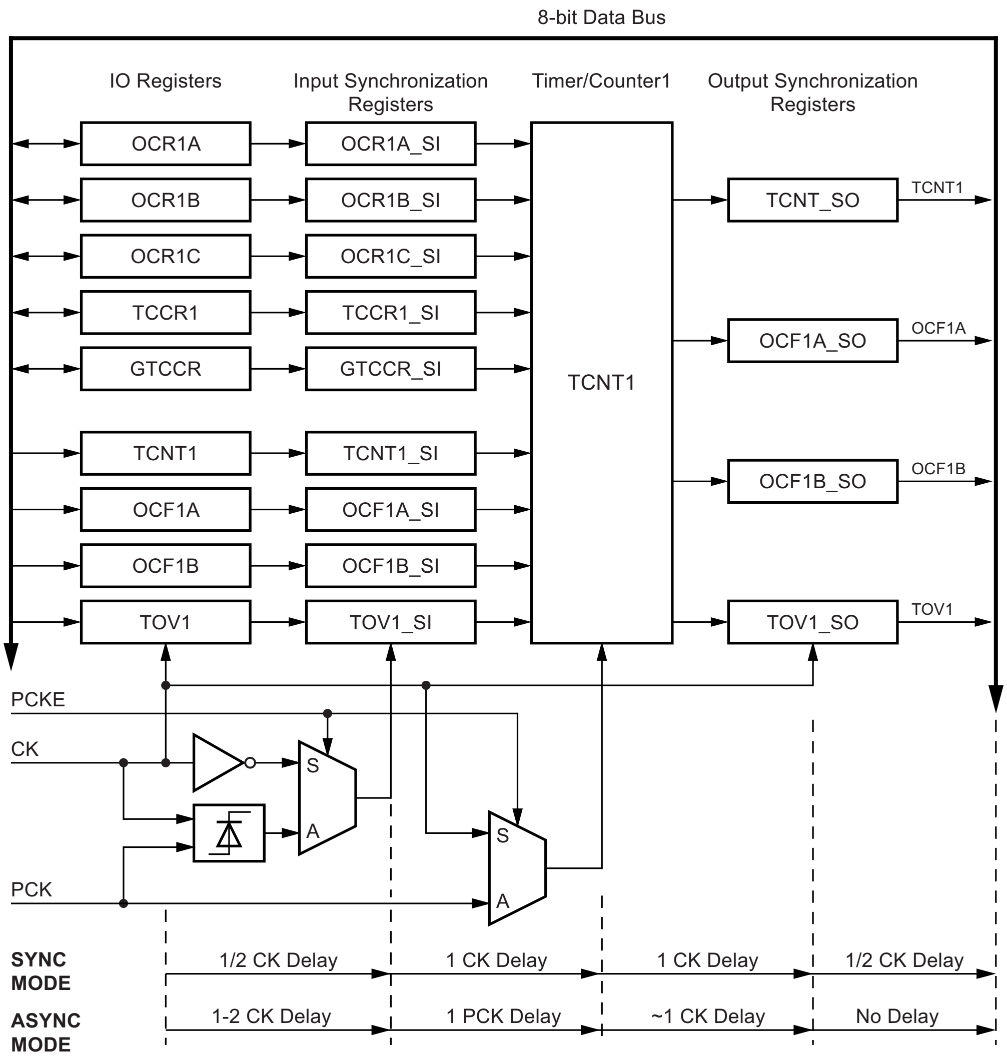
In the asynchronous clocking mode the clock selections are from PCK to PCK/16384 and stop, and in the synchronous clocking mode the clock selections are from CK to CK/16384 and stop. The clock options are described in Table 13-2 on page 78 and the Timer/Counter1 control register, TCCR1. Setting the PSR1 bit in GTCCR register resets the prescaler. The PCKE bit in the PLLCSR register enables the asynchronous mode. The frequency of the fast peripheral clock is 64MHz (or 32MHz in low speed mode).

13.1 Timer/Counter1

The Timer/Counter1 general operation is described in the asynchronous mode and the operation in the synchronous mode is mentioned only if there are differences between these two modes. Figure 13-2 shows Timer/Counter 1 synchronization register block diagram and synchronization delays in between registers. Note that all clock gating details are not shown in the figure. The Timer/Counter1 register values go through the internal synchronization registers, which cause the input synchronization delay, before affecting the counter operation. The registers TCCR1, GTCCR, OCR1A, OCR1B, and OCR1C can be read back right after writing the register. The read back values are delayed for the Timer/Counter1 (TCNT1) register and flags (OCF1A, OCF1B, and TOV1), because of the input and output synchronization.

The Timer/Counter1 features a high resolution and a high accuracy usage with the lower prescaling opportunities. It can also support two accurate, high speed, 8-bit pulse width modulators using clock speeds up to 64MHz (or 32MHz in low speed mode). In this mode, Timer/Counter1 and the output compare registers serve as dual stand-alone PWMs with non-overlapping non-inverted and inverted outputs. Refer to Section 13.1.11 “Timer/Counter1 in PWM Mode” on page 82 for a detailed description on this function. Similarly, the high prescaling opportunities make this unit useful for lower speed functions or exact timing functions with infrequent actions.

Figure 13-2. Timer/Counter 1 Synchronization Register Block Diagram

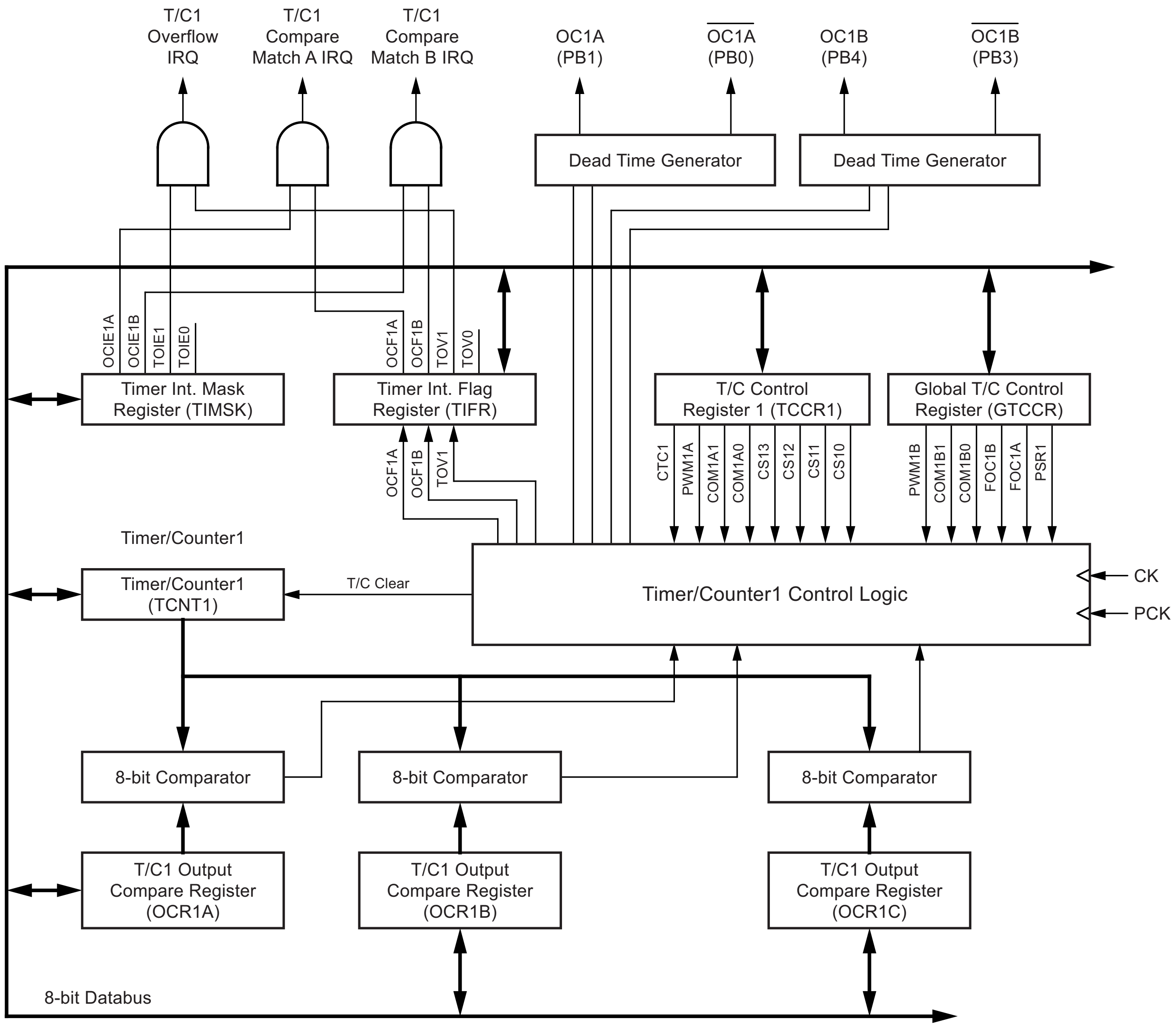


Timer/Counter1 and the prescaler allow running the CPU from any clock source while the prescaler is operating on the fast 64MHz (or 32MHz in low speed mode) PCK clock in the asynchronous mode.

Note that the system clock frequency must be lower than one third of the PCK frequency. The synchronization mechanism of the asynchronous Timer/Counter1 needs at least two edges of the PCK when the system clock is high. If the frequency of the system clock is too high, it is a risk that data or control values are lost.

The following Figure 13-3 on page 76 shows the block diagram for Timer/Counter1.

Figure 13-3. Timer/Counter1 Block Diagram



Three status flags (overflow and compare matches) are found in the Timer/Counter interrupt flag register - TIFR. control signals are found in the Timer/Counter control registers TCCR1 and GTCCR. The interrupt enable/disable settings are found in the Timer/Counter interrupt mask register - TIMSK.

The Timer/Counter1 contains three output compare registers, OCR1A, OCR1B, and OCR1C as the data source to be compared with the Timer/Counter1 contents. In normal mode the output compare functions are operational with all three output compare registers. OCR1A determines action on the OC1A pin (PB1), and it can generate timer1 OC1A interrupt in normal mode and in PWM mode. Likewise, OCR1B determines action on the OC1B pin (PB3) and it can generate timer1 OC1B interrupt in normal mode and in PWM mode. OCR1C holds the Timer/Counter maximum value, i.e. the clear on compare match value. In the normal mode an overflow interrupt (TOV1) is generated when Timer/Counter1 counts from $FF to $00, while in the PWM mode the overflow interrupt is generated when Timer/Counter1 counts either from $FF to $00 or from OCR1C to $00. The inverted PWM outputs $\overline{\text{OC1A}}$ and $\overline{\text{OC1B}}$ are not connected in normal mode.

In PWM mode, OCR1A and OCR1B provide the data values against which the timer counter value is compared. Upon compare match the PWM outputs (OC1A, $\overline{\text{OC1A}}$, OC1B, $\overline{\text{OC1B}}$) are generated. In PWM mode, the timer counter counts up to the value specified in the output compare register OCR1C and starts again from $00. This feature allows limiting the counter "full" value to a specified value, lower than $FF. Together with the many prescaler options, flexible PWM frequency selection is provided. Table 13-6 on page 84 lists clock selection and OCR1C values to obtain PWM frequencies from 20kHz to 250kHz in 10kHz steps and from 250kHz to 500kHz in 50kHz steps. Higher PWM frequencies can be obtained at the expense of resolution.

13.1.1 Timer/Counter1 Control Register - TCCR1

Bit	7	6	5	4	3	2	1	0	
\$30 (\$50)	**CTC1**	**PWM1A**	**COM1A1**	**COM1A0**	**CS13**	**CS12**	**CS11**	**CS10**	**TCCR1**
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- CTC1: Clear Timer/Counter on Compare Match**

When the CTC1 control bit is set (one), Timer/Counter1 is reset to \$00 in the CPU clock cycle after a compare match with OCR1C register value. If the control bit is cleared, Timer/Counter1 continues counting and is unaffected by a compare match.

- **Bit 6- PWM1A: Pulse Width Modulator A Enable**

When set (one) this bit enables PWM mode based on comparator OCR1A in Timer/Counter1 and the counter value is reset to \$00 in the CPU clock cycle after a compare match with OCR1C register value.

- **Bits 5,4 - COM1A1, COM1A0: Comparator A Output Mode, Bits 1 and 0**

The COM1A1 and COM1A0 control bits determine any output pin action following a compare match with compare register A in Timer/Counter1. Output pin actions affect pin PB1 (OC1A). Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) in order to control an output pin. Note that $\overline{\text{OC1A}}$ is not connected in normal mode.

Table 13-1. Comparator A Mode Select

COM1A1	COM1A0	Description
0	0	Timer/Counter comparator A disconnected from output pin OC1A.
0	1	Toggle the OC1A output line.
1	0	Clear the OC1A output line.
1	1	Set the OC1A output line

In PWM mode, these bits have different functions. Refer to Table 13-4 on page 83 for a detailed description.

- **Bits 3..0 - CS13, CS12, CS11, CS10: Clock Select Bits 3, 2, 1, and 0**

The clock select bits 3, 2, 1, and 0 define the prescaling source of Timer/Counter1.

Table 13-2. Timer/Counter1 Prescale Select

CS13	CS12	CS11	CS10	Asynchronous Clocking Mode	Synchronous Clocking Mode
0	0	0	0	T/C1 stopped	T/C1 stopped
0	0	0	1	PCK	CK
0	0	1	0	PCK/2	CK/2
0	0	1	1	PCK/4	CK/4
0	1	0	0	PCK/8	CK/8
0	1	0	1	PCK/16	CK/16
0	1	1	0	PCK/32	CK/32
0	1	1	1	PCK/64	CK/64
1	0	0	0	PCK/128	CK/128
1	0	0	1	PCK/256	CK/256
1	0	1	0	PCK/512	CK/512
1	0	1	1	PCK/1024	CK/1024
1	1	0	0	PCK/2048	CK/2048
1	1	0	1	PCK/4096	CK/4096
1	1	1	0	PCK/8192	CK/8192
1	1	1	1	PCK/16384	CK/16384

The Stop condition provides a timer enable/disable function.

13.1.2 General Timer/Counter1 Control Register - GTCCR

Bit	7	6	5	4	3	2	1	0	
$2C ($4C)	TSM	PWM1B	COM1B1	COM1B0	FOC1B	FOC1A	PSR1	PSR0	GTCCR
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 6- PWM1B: Pulse Width Modulator B Enable**

When set (one) this bit enables PWM mode based on comparator OCR1B in Timer/Counter1 and the counter value is reset to $00 in the CPU clock cycle after a compare match with OCR1C register value.

- **Bits 5,4 - COM1B1, COM1B0: Comparator B Output Mode, Bits 1 and 0**

The COM1B1 and COM1B0 control bits determine any output pin action following a compare match with compare register B in Timer/Counter1. Output pin actions affect pin PB3 (OC1B). Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) in order to control an output pin. Note that $\overline{\text{OC1B}}$ is not connected in normal mode.

Table 13-3. Comparator B Mode Select

COM1B1	COM1B0	Description
0	0	Timer/Counter comparator B disconnected from output pin OC1B.
0	1	Toggle the OC1B output line.
1	0	Clear the OC1B output line.
1	1	Set the OC1B output line

In PWM mode, these bits have different functions. Refer to Table 13-4 on page 83 for a detailed description.

- **Bit 3- FOC1B: Force Output Compare Match 1B**

Writing a logical one to this bit forces a change in the compare match output pin PB3 (OC1B) according to the values already set in COM1B1 and COM1B0. If COM1B1 and COM1B0 written in the same cycle as FOC1B, the new settings will be used. The force output compare bit can be used to change the output pin value regardless of the timer value. The automatic action programmed in COM1B1 and COM1B0 takes place as if a compare match had occurred, but no interrupt is generated. The FOC1B bit always reads as zero. FOC1B is not in use if PWM1B bit is set.

- **Bit 2- FOC1A: Force Output Compare Match 1A**

Writing a logical one to this bit forces a change in the compare match output pin PB1 (OC1A) according to the values already set in COM1A1 and COM1A0. If COM1A1 and COM1A0 written in the same cycle as FOC1A, the new settings will be used. The force output compare bit can be used to change the output pin value regardless of the timer value. The automatic action programmed in COM1A1 and COM1A0 takes place as if a compare match had occurred, but no interrupt is generated. The FOC1A bit always reads as zero. FOC1A is not in use if PWM1A bit is set.

- **Bit 1- PSR1: Prescaler Reset Timer/Counter1**

When this bit is set (one), the Timer/Counter prescaler (TCNT1 is unaffected) will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always read as zero.

13.1.3 Timer/Counter1 - TCNT1

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	**MSB**							**LSB**	**TCNT1**
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	

This 8-bit register contains the value of Timer/Counter1.

Timer/Counter1 is realized as an up counter with read and write access. Due to synchronization of the CPU, Timer/Counter1 data written into Timer/Counter1 is delayed by one and half CPU clock cycles in synchronous mode and at most one CPU clock cycles for asynchronous mode.

13.1.4 Timer/Counter1 Output Compare RegisterA - OCR1A

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	**MSB**							**LSB**	**OCR1A**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The output compare register A is an 8-bit read/write register.

The Timer/Counter output compare register A contains data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in TCCR1. A compare match does only occur if Timer/Counter1 counts to the OCR1A value. A software write that sets TCNT1 and OCR1A to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF1A after a synchronization delay following the compare event.

13.1.5 Timer/Counter1 Output Compare RegisterB - OCR1B

Bit	7	6	5	4	3	2	1	0	
$2D ($4D)	**MSB**							**LSB**	**OCR1B**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The output compare register B is an 8-bit read/write register.

The Timer/Counter output compare register B contains data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in TCCR1. A compare match does only occur if Timer/Counter1 counts to the OCR1B value. A software write that sets TCNT1 and OCR1B to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF1B after a synchronization delay following the compare event.

13.1.6 Timer/Counter1 Output Compare RegisterC - OCR1C

Bit	7	6	5	4	3	2	1	0	
$2B ($4B)	**MSB**							**LSB**	**OCR1C**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	1	1	1	1	1	1	1	1	

The output compare register C is an 8-bit read/write register.

The Timer/Counter output compare register C contains data to be continuously compared with Timer/Counter1. A compare match does only occur if Timer/Counter1 counts to the OCR1C value. A software write that sets TCNT1 and OCR1C to the same value does not generate a compare match. If the CTC1 bit in TCCR1 is set, a compare match will clear TCNT1.

This register has the same function in normal mode and PWM mode.

13.1.7 Timer/Counter Interrupt Mask Register - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	**-**	**OCIE1A**	**OCIE1B**	**OCIE0A**	**OCIE0B**	**TOIE1**	**TOIE0**	**-**	**TIMSK**
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny25/45/85 and always reads as zero.

- **Bit 6 - OCIE1A: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 compare match A, interrupt is enabled. The corresponding interrupt at vector $003 is executed if a compare match A occurs. The compare flag in Timer/Counter1 is set (one) in the Timer/Counter interrupt flag register.

- **Bit 5 - OCIE1B: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1B bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 compare match B, interrupt is enabled. The corresponding interrupt at vector $009 is executed if a compare match B occurs. The compare flag in Timer/Counter1 is set (one) in the Timer/Counter interrupt flag register.

- **Bit 4– OCIE0A: Timer/Counter Output Compare Match A Interrupt Enable**

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

- **Bit 3 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

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

- **Bit 2 - TOIE1: Timer/Counter1 Overflow Interrupt Enable**

When the TOIE1 bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 overflow interrupt is enabled. The corresponding interrupt (at vector $004) is executed if an overflow in Timer/Counter1 occurs. The overflow flag (timer1) is set (one) in the Timer/Counter interrupt flag register - TIFR.

- **Bit 0 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel ATtiny25/45/85 and always reads as zero.

13.1.8 Timer/Counter Interrupt Flag Register - TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	-	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	-	TIFR
Read/Write	R	R/W	R/W	R	R	R/W	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny25/45/85 and always reads as zero.

- **Bit 6 - OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1A - output compare register 1A. OCF1A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared, after synchronization clock cycle, by writing a logic one to the flag. When the I-bit in SREG, OCIE1A, and OCF1A are set (one), the Timer/Counter1 A compare match interrupt is executed.

- **Bit 5 - OCF1B: Output Compare Flag 1B**

The OCF1B bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1B - output compare register 1A. OCF1B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1B is cleared, after synchronization clock cycle, by writing a logic one to the flag. When the I-bit in SREG, OCIE1B, and OCF1B are set (one), the Timer/Counter1 B compare match interrupt is executed.

- **Bit 2 - TOV1: Timer/Counter1 Overflow Flag**

In normal mode (PWM1A=0 and PWM1B=0) the bit TOV1 is set (one) when an overflow occurs in Timer/Counter1. The bit TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared, after synchronization clock cycle, by writing a logical one to the flag.

In PWM mode (either PWM1A=1 or PWM1B=1) the bit TOV1 is set (one) when compare match occurs between Timer/Counter1 and data value in OCR1C - output compare register 1C. Clearing the Timer/Counter1 with the bit CTC1 does not generate an overflow.

When the SREG I-bit, and TOIE1 (Timer/Counter1 overflow interrupt enable), and TOV1 are set (one), the Timer/Counter1 overflow interrupt is executed.

- **Bit 0 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel ATtiny25/45/85 and always reads as zero.

13.1.9 PLL Control and Status Register - PLLCSR

Bit	7	6	5	4	3	2	1	0	
$27 ($27)	LSM	-	-	-	-	PCKE	PLLE	PLOCK	PLLCSR
Read/Write	R/W	R	R	R	R	R/W	R/W	R	
Initial value	0	0	0	0	0	0	0/1	0	

- **Bit 7- LSM: Low Speed Mode**

The high speed mode is enabled as default and the fast peripheral clock is 64MHz, but the low speed mode can be set by writing the LSM bit to one. Then the fast peripheral clock is scaled down to 32MHz. The low speed mode must be set, if the supply voltage is below 2.7 volts, because the Timer/Counter1 is not running fast enough on low voltage levels. It is highly recommended that Timer/Counter1 is stopped whenever the LSM bit is changed.

- **Bit 6.. 3- Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny25/45/85 and always read as zero.

- **Bit 2- PCKE: PCK Enable**

The PCKE bit change the Timer/Counter1 clock source. When it is set, the asynchronous clock mode is enabled and fast 64MHz (or 32MHz in low speed mode) PCK clock is used as Timer/Counter1 clock source. If this bit is cleared, the synchronous clock mode is enabled, and system clock CK is used as Timer/Counter1 clock source. This bit can be set only if PLLE bit is set. It is safe to set this bit only when the PLL is locked i.e the PLOCK bit is 1. The bit PCKE can only be set, if the PLL has been enabled earlier.

- **Bit 1- PLLE: PLL Enable**

When the PLLE is set, the PLL is started and if needed internal RC-oscillator is started as a PLL reference clock. If PLL is selected as a system clock source the value for this bit is always 1.

- **Bit 0- PLOCK: PLL Lock Detector**

When the PLOCK bit is set, the PLL is locked to the reference clock, and it is safe to enable PCK for Timer/Counter1. After the PLL is enabled, it takes about 100 micro seconds for the PLL to lock.

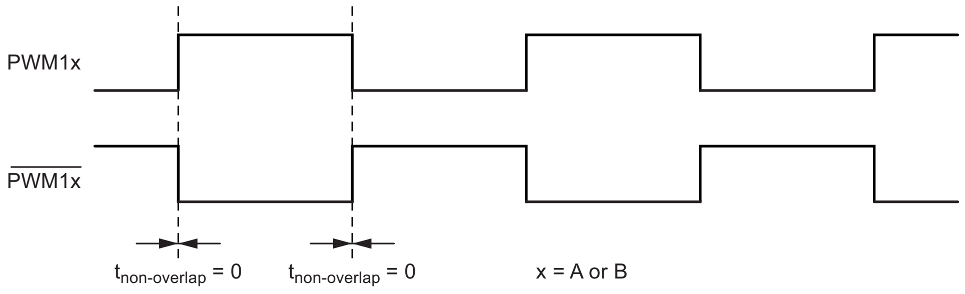
13.1.10 Timer/Counter1 Initialization for Asynchronous Mode

To change Timer/Counter1 to the asynchronous mode, first enable PLL, wait 100µs before polling the PLOCK bit until it is set, and then set the PCKE bit.

13.1.11 Timer/Counter1 in PWM Mode

When the PWM mode is selected, Timer/Counter1 and the output compare register C - OCR1C form a dual 8-bit, free-running and glitch-free PWM generator with outputs on the PB1(OC1A) and PB3(OC1B) pins and inverted outputs on pins PB0($\overline{\text{OC1A}}$) and PB2($\overline{\text{OC1B}}$). As default non-overlapping times for complementary output pairs are zero, but they can be inserted using a dead time generator (see Section 14. "Dead Time Generator" on page 85).

Figure 13-4. The PWM Output Pair



When the counter value match the contents of OCR1A or OCR1B, the OC1A and OC1B outputs are set or cleared according to the COM1A1/COM1A0 or COM1B1/COM1B0 bits in the Timer/Counter1 control register A - TCCR1, as shown in Table 13-4 on page 83.

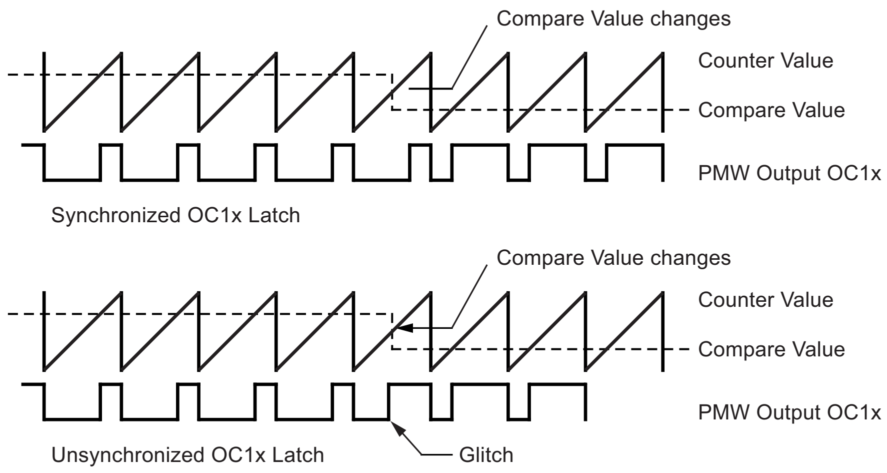
Timer/Counter1 acts as an up-counter, counting from $00 up to the value specified in the output compare register OCR1C, and starting from $00 up again. A compare match with OC1C will set an overflow interrupt flag (TOV1) after a synchronization delay following the compare event.

Table 13-4. Compare Mode Select in PWM Mode

COM11	COM10	Effect on Output Compare Pins
0	0	OC1x not connected. $\overline{\text{OC1x}}$ not connected.
0	1	OC1x cleared on compare match. Set whenTCNT1 = $01. $\overline{\text{OC1x}}$ set on compare match. Cleared when TCNT1 = $00.
1	0	OC1x cleared on compare match. Set when TCNT1 = $01. $\overline{\text{OC1x}}$ not connected.
1	1	OC1x Set on compare match. Cleared when TCNT1= $01. $\overline{\text{OC1x}}$ not connected.

Note that in PWM mode, writing to the output compare registers OCR1A or OCR1B, the data value is first transferred to a temporary location. The value is latched into OCR1A or OCR1B when the Timer/Counter reaches OCR1C. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A or OCR1B. See Figure 13-5 for an example.

Figure 13-5. Effects of Unsynchronized OCR Latching



During the time between the write and the latch operation, a read from OCR1A or OCR1B will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A or OCR1B.

When OCR1A or OCR1B contain $00 or the top value, as specified in OCR1C register, the output PB1(OC1A) or PB3(OC1B) is held low or high according to the settings of COM1A1/COM1A0. This is shown in Table 13-5 on page 83.

Table 13-5. PWM Outputs OCR1x = $00 or OCR1C, x = A or B

COM1x1	COM1x0	OCR1x	Output OC1x	Output $\overline{\text{OC1x}}$
0	1	$00	L	H
0	1	OCR1C	H	L
1	0	$00	L	Not connected.
1	0	OCR1C	H	Not connected.
1	1	$00	H	Not connected.
1	1	OCR1C	L	Not connected.

In PWM mode, the timer overflow flag - TOV1 is set when the TCNT1 counts to the OCR1C value and the TCNT1 is reset to $00. The timer overflow interrupt1 is executed when TOV1 is set provided that timer overflow interrupt and global interrupts are enabled. This also applies to the timer output compare flags and interrupts.

The frequency of the PWM will be timer clock 1 frequency divided by (OCR1C value + 1). See the following equation:

$$f_{\mathrm{PWM}} = \frac{f_{\mathrm{TCK1}}}{(\mathrm{OCR1C} + 1)}$$

Resolution shows how many bit is required to express the value in the OCR1C register. It is calculated by following equation

$Resolution_{\mathrm{PWM}} = \log_2(\mathrm{OCR1C} + 1)$.

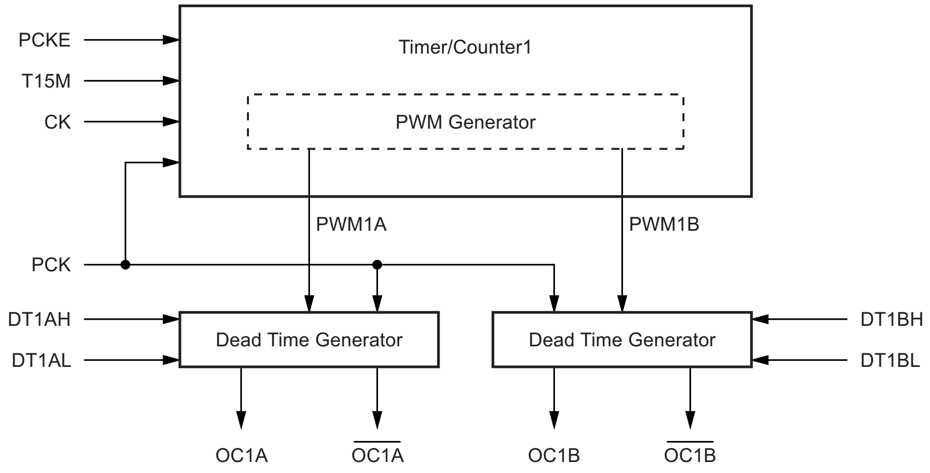
Table 13-6. Timer/Counter1 Clock Prescale Select in the Asynchronous Mode

PWM Frequency	Clock Selection	CS13..CS10	OCR1C	RESOLUTION
20kHz	PCK/16	0101	199	7.6
30kHz	PCK/16	0101	132	7.1
40kHz	PCK/8	0100	199	7.6
50kHz	PCK/8	0100	159	7.3
60kHz	PCK/8	0100	132	7.1
70kHz	PCK/4	0011	228	7.8
80kHz	PCK/4	0011	199	7.6
90kHz	PCK/4	0011	177	7.5
100kHz	PCK/4	0011	159	7.3
110kHz	PCK/4	0011	144	7.2
120kHz	PCK/4	0011	132	7.1
130kHz	PCK/2	0010	245	7.9
140kHz	PCK/2	0010	228	7.8
150kHz	PCK/2	0010	212	7.7
160kHz	PCK/2	0010	199	7.6
170kHz	PCK/2	0010	187	7.6
180kHz	PCK/2	0010	177	7.5
190kHz	PCK/2	0010	167	7.4
200kHz	PCK/2	0010	159	7.3
250kHz	PCK	0001	255	8.0
300kHz	PCK	0001	212	7.7
350kHz	PCK	0001	182	7.5
400kHz	PCK	0001	159	7.3
450kHz	PCK	0001	141	7.1
500kHz	PCK	0001	127	7.0

14. Dead Time Generator

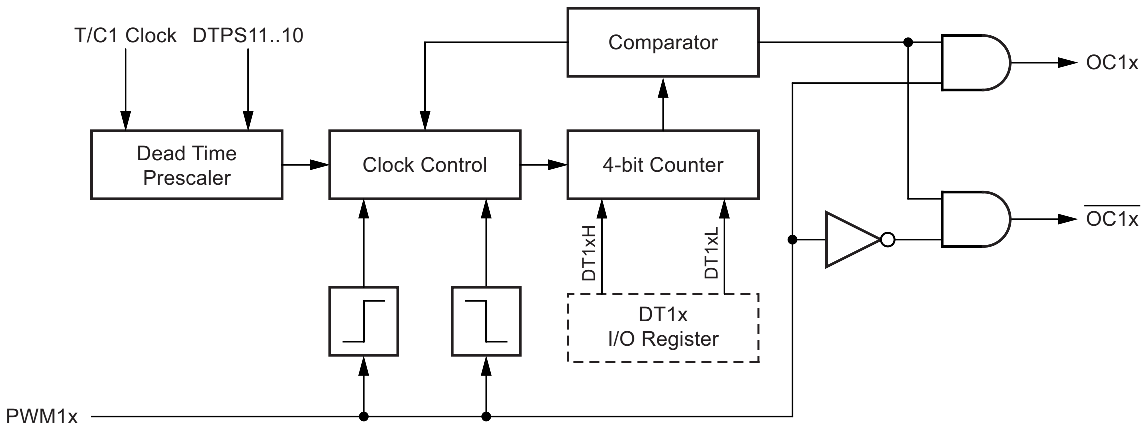
The dead time generator is provided for the Timer/Counter1 PWM output pairs to allow driving external power control switches safely. The dead time generator is a separate block that can be connected to Timer/Counter1 and it is used to insert dead times (non-overlapping times) for the Timer/Counter1 complementary output pairs (OC1A-$\overline{\text{OC1A}}$ and OC1B-$\overline{\text{OC1B}}$). The sharing of tasks is as follows: the Timer/Counter generates the PWM output and the dead time generator generates the non-overlapping PWM output pair from the Timer/Counter PWM signal. Two dead time generators are provided, one for each PWM output. The non-overlap time is adjustable and the PWM output and it's complementary output are adjusted separately, and independently for both PWM outputs.

Figure 14-1. Timer/Counter1 and Dead Time Generators



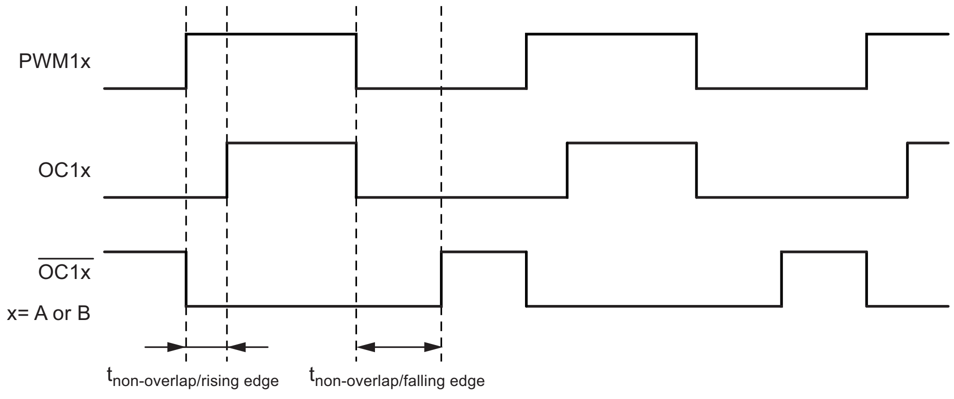
The dead time generation is based on the 4-bit down counters that count the dead time, as shown in Figure 14-1 There is a dedicated prescaler in front of the dead time generator that can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8. This provides for large range of dead times that can be generated. The prescaler is controlled by two control bits DTPS11..10 from the I/O register at address 0x23. The block has also a rising and falling edge detector that is used to start the dead time counting period. Depending on the edge, one of the transitions on the rising edges, OC1x or $\overline{\text{OC1x}}$ is delayed until the counter has counted to zero. The comparator is used to compare the counter with zero and stop the dead time insertion when zero has been reached. The counter is loaded with a 4-bit DT1xH or DT1xL value from DT1x I/O register, depending on the edge of the PWM generator output when the dead time insertion is started.

Figure 14-2. Dead Time Generator



The length of the counting period is user adjustable by selecting the dead time prescaler setting in 0x23 register, and selecting then the dead time value in I/O register DT1x. The DT1x register consists of two 4-bit fields, DT1xH and DT1xL that control the dead time periods of the PWM output and its' complementary output separately. Thus the rising edge of OC1x and $\overline{\text{OC1x}}$ can have different dead time periods. The dead time is adjusted as the number of prescaled dead time generator clock cycles.

Figure 14-3. The Complementary Output Pair


PWM1x
OC1x
$\overline{\text{OC1x}}$
x= A or B
$t_{\text{non-overlap/rising edge}}$
$t_{\text{non-overlap/falling edge}}$

14.1 Timer/Counter1 Dead Time Prescaler register 1 - DTPS1

Bit	7	6	5	4	3	2	1	0	
$23 ($43)							DTPS11	DTPS10	**DTPS1**
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The dead time prescaler register, DTPS1 is a 2-bit read/write register.

Bits 1 - 0 - DTPS1: Timer/Counter1 dead time prescaler register 1

The dedicated dead time prescaler in front of the dead time generator can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8 providing a large range of dead times that can be generated. The dead time prescaler is controlled by two bits DTPS11..10 from the dead time prescaler register. These bits define the division factor of the dead time prescaler. The division factors are given in Table 14-1.

Table 14-1. Division Factors of the Dead Time Prescaler

DTPS11	DTPS10	Prescaler divides the T/C1 clock by
0	0	1x (no division)
0	1	2x
1	0	4x
1	1	8x

14.2 Timer/Counter1 Dead Time A - DT1A

Bit	7	6	5	4	3	2	1	0	
$25 ($45)	**DT1AH3**	**DT1AH2**	**DT1AH1**	**DT1AH0**	**DT1AL3**	**DT1AL2**	**DT1AL1**	**DT1AL0**	**DT1A**
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 dead time value register A is an 8-bit read/write register.

The dead time delay of is adjusted by the dead time value register, DT1A. The register consists of two fields, DT1AH3..0 and DT1AL3..0, one for each complementary output. Therefore a different dead time delay can be adjusted for the rising edge of OC1A and the rising edge of $\overline{\text{OC1A}}$.

- **Bits 7..4- DT1AH3..DT1AH0: Dead Time Value for OC1A Output**

The dead time value for the OC1A output. The dead time delay is set as a number of the prescaled Timer/Counter clocks. The minimum dead time is zero and the maximum dead time is the prescaled time/counter clock period multiplied by 15.

- **Bits 3..0- DT1AL3..DT1AL0: Dead Time Value for $\overline{\text{OC1A}}$ Output**

The dead time value for the $\overline{\text{OC1A}}$ output. The dead time delay is set as a number of the prescaled Timer/Counter clocks. The minimum dead time is zero and the maximum dead time is the prescaled time/counter clock period multiplied by 15.

14.3 Timer/Counter1 Dead Time B - DT1B

Bit	7	6	5	4	3	2	1	0	
$25 ($45)	**DT1BH3**	**DT1BH2**	**DT1BH1**	**DT1BH0**	**DT1BL3**	**DT1BL2**	**DT1BL1**	**DT1BL0**	**DT1B**
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 dead time value register bit an 8-bit read/write register.

The dead time delay of is adjusted by the dead time value register, DT1B. The register consists of two fields, DT1BH3..0 and DT1BL3..0, one for each complementary output. Therefore a different dead time delay can be adjusted for the rising edge of OC1A and the rising edge of $\overline{\text{OC1A}}$.

- **Bits 7..4- DT1BH3..DT1BH0: Dead Time Value for OC1B Output**

The dead time value for the OC1B output. The dead time delay is set as a number of the prescaled Timer/Counter clocks. The minimum dead time is zero and the maximum dead time is the prescaled time/counter clock period multiplied by 15.

- **Bits 3..0- DT1BL3..DT1BL0: Dead Time Value for $\overline{\text{OC1B}}$ Output**

The dead time value for the $\overline{\text{OC1B}}$ output. The dead time delay is set as a number of the prescaled Timer/Counter clocks. The minimum dead time is zero and the maximum dead time is the prescaled time/counter clock period multiplied by 15.

15. Universal Serial Interface – USI

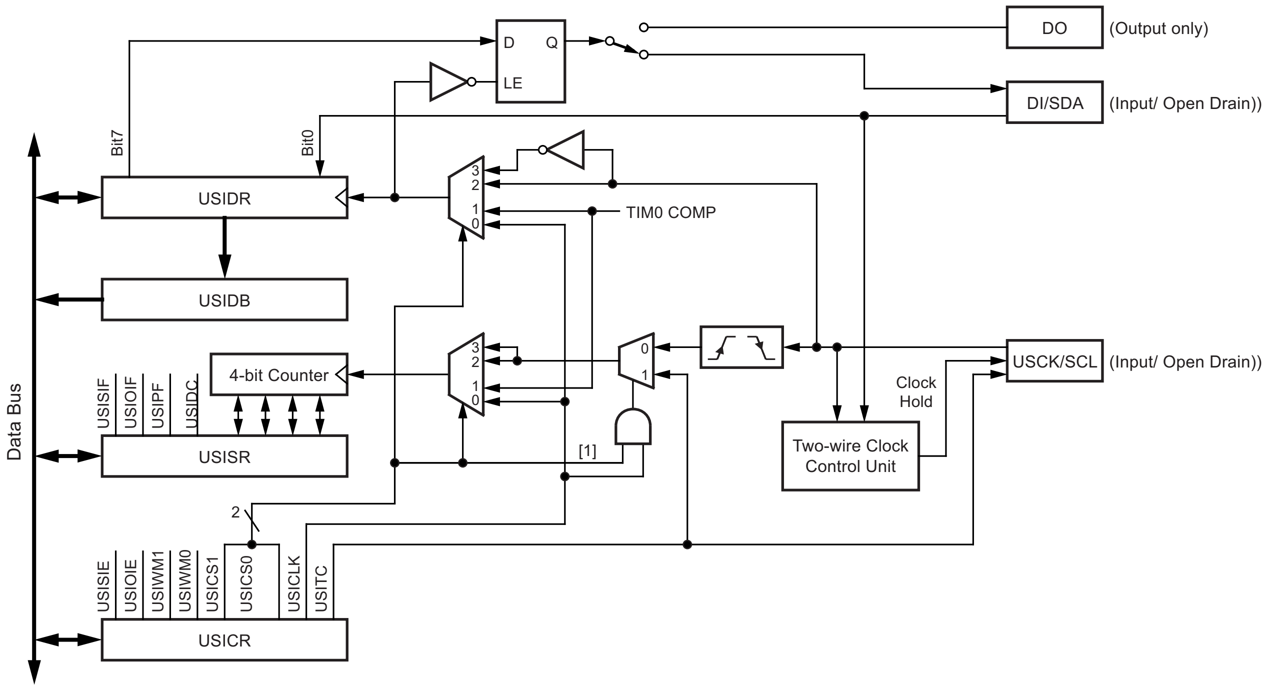
The universal serial interface, or USI, provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize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

- Two-wire synchronous data transfer (master or slave, $f_{SCLmax} = f_{CK}/16$)
- Three-wire synchronous data transfer (master or slave $f_{SCKmax} = f_{CK}/4$)
- Data received interrupt
- Wakeup from idle mode
- In two-wire mode: Wake-up from all sleep modes, including power-down mode
- Two-wire start condition detector with interrupt capability

15.1 Overview

A simplified block diagram of the USI is shown on Figure 15-1 For the actual placement of I/O pins, refer to Figure 1 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 Section 15.4 "USI Register Descriptions" on page 94.

Figure 15-1. Universal Serial Interface, Block Diagram



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

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the serial register and the counter are clocked simultaneously by the same clock source.

This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected the counter counts both clock edges. In this case the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: The USCK pin, Timer/Counter0 compare match or from software.

The two-wire clock control unit can generate an interrupt when a start condition is detected on the two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

15.2 Functional Descriptions

15.2.1 Three-wire Mode

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

Figure 15-2. Three-wire Mode Operation, Simplified Diagram

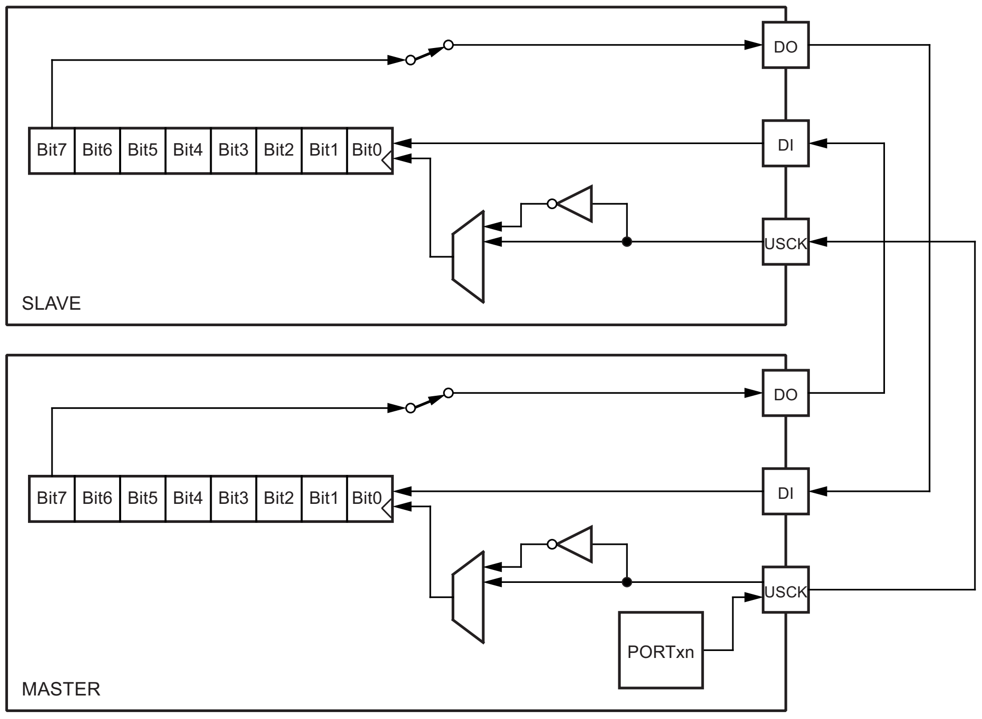
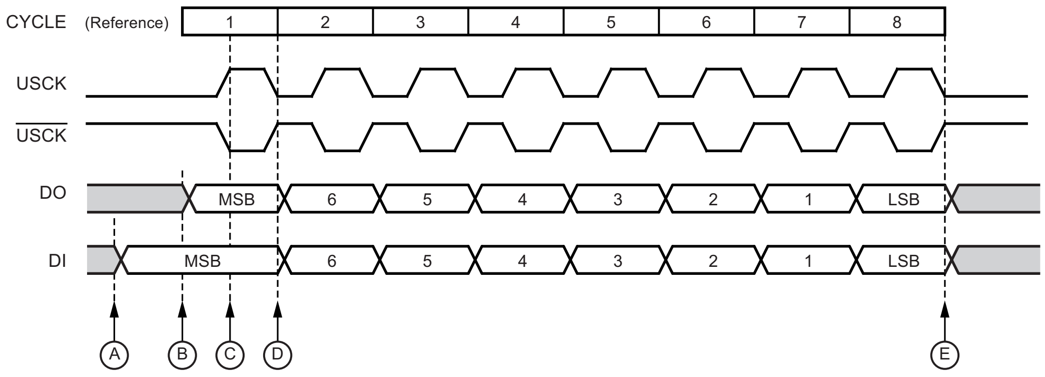


Figure 15-2 shows two USI units operating in three-wire mode, one as master and one as slave. The two shift registers are interconnected in such way that after eight USCK clocks, the data in each register are interchanged. The same clock also increments the USI's 4-bit counter. The counter overflow (interrupt) flag, or USIOIF, can therefore be used to determine when a transfer is completed. The clock is generated by the master device software by toggling the USCK pin via the PORT register or by writing a one to the USITC bit in USICR.

Figure 15-3. Three-wire Mode, Timing Diagram



The three-wire mode timing is shown in Figure 15-3 on page 89 At the top of the figure is a USCK cycle reference. One bit is shifted into the USI shift register (USIDR) for each of these cycles. The USCK timing is shown for both external clock modes. In external clock mode 0 (USICS0 = 0), DI is sampled at positive edges, and DO is changed (data register is shifted by one) at negative edges. External clock mode 1 (USICS0 = 1) uses the opposite edges versus mode 0, i.e., samples data at negative and changes the output at positive edges. The USI clock modes corresponds to the SPI data mode 0 and 1.

Referring to the timing diagram (Figure 15-3 on page 89), a bus transfer involves the following steps:

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

15.2.2 SPI Master Operation Example

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

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

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

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

The following code demonstrates how to use the USI module as a SPI master with maximum speed (fsck = fck/4):

```
SPITransfer_Fast:

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

        sts    USICR,r16; MSB
        sts    USICR,r17
        sts    USICR,r16
        sts    USICR,r17
        sts    USICR,r16
        sts    USICR,r17
        sts    USICR,r16
        sts    USICR,r17
        sts    USICR,r16
        sts    USICR,r17
        sts    USICR,r16
        sts    USICR,r17
        sts    USICR,r16
        sts    USICR,r17
        sts    USICR,r16; LSB
        sts    USICR,r17

        lds    r16,USIDR
ret
```

15.2.3 SPI Slave Operation Example

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

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

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

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

15.2.4 Two-wire Mode

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

Figure 15-4. Two-wire Mode Operation, Simplified Diagram

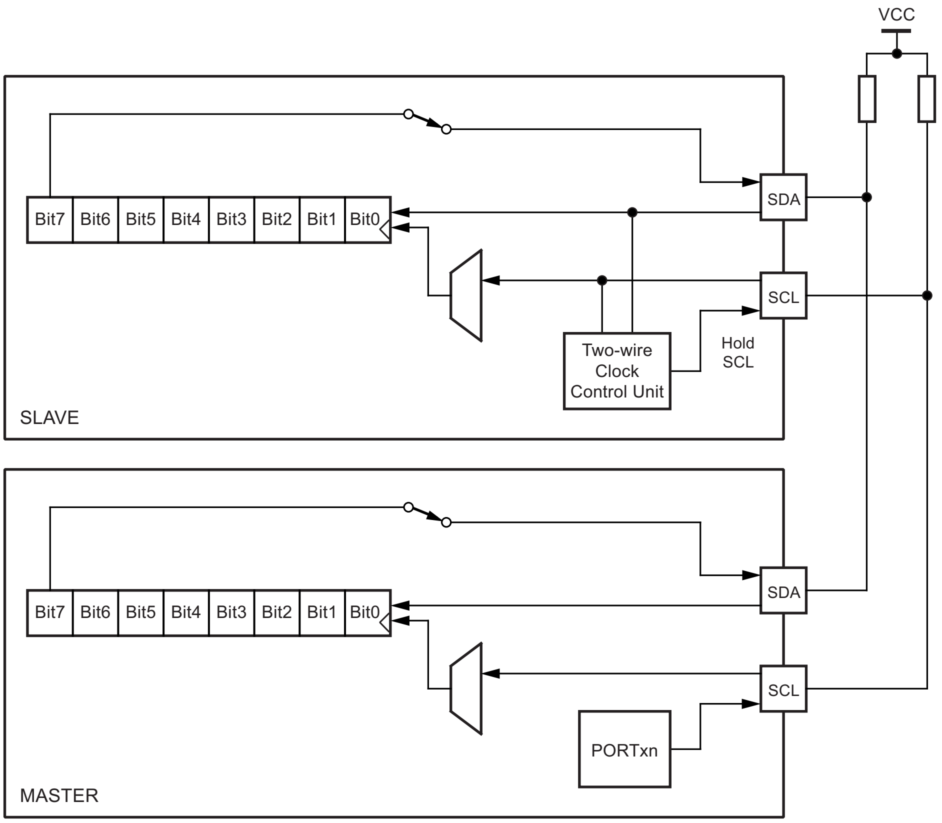
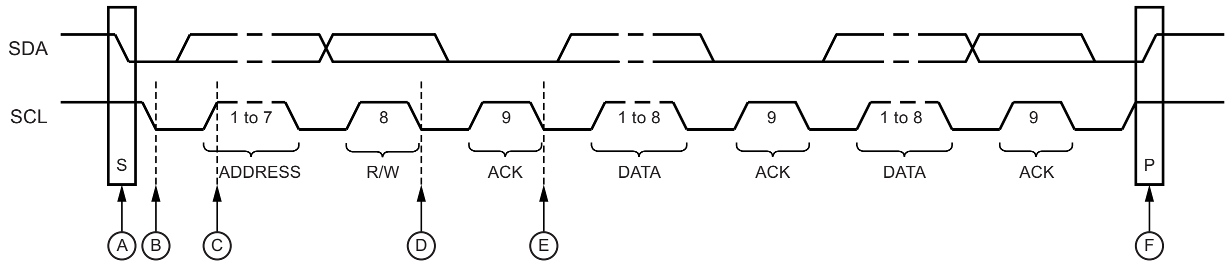


Figure 15-4 shows two USI units operating in two-wire mode, one as master and one as slave. It is only the physical layer that is shown since the system operation is highly dependent of the communication scheme used. The main differences between the master and slave operation at this level, is the serial clock generation which is always done by the master, and only the slave uses the clock control unit. Clock generation must be implemented in software, but the shift operation is done automatically by both devices. Note that only clocking on negative edge for shifting data is of practical use in this mode. The slave can insert wait states at start or end of transfer by forcing the SCL clock low. This means that the master must always check if the SCL line was actually released after it has generated a positive edge.

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

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

Figure 15-5. Two-wire Mode, Typical Timing Diagram

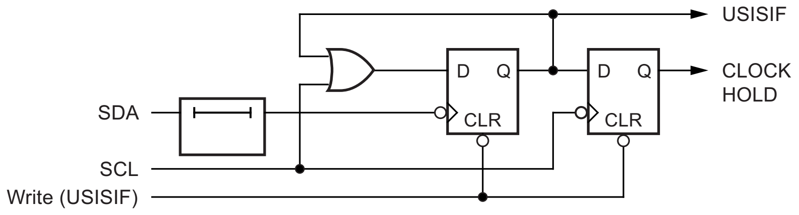


Referring to the timing diagram (Figure 15-5 on page 92), a bus transfer involves the following steps:

1. The a start condition is generated by the master by forcing the SDA low line while the SCL line is high (A). SDA can be forced low either by writing a zero to bit 7 of the shift register, or by setting the corresponding bit in the PORT register to zero. Note that the data direction register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 15-5) detects the start condition and sets the USISIF flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the master has forced an negative edge on this line (B). This allows the Slave to wake up from sleep or complete its other tasks before setting up the shift register to receive the address. This is done by clearing the start condition flag and reset the counter.
3. The master set the first bit to be transferred and releases the SCL line (C). The slave samples the data and shift it into the serial register at the positive edge of the SCL clock.
4. After eight bits are transferred containing slave address and data direction (read or write), the slave counter overflows and the SCL line is forced low (D). If the slave is not the one the master has addressed, it releases the SCL line and waits for a new start condition.
5. If the slave is addressed it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the counter register must be set to 14 before releasing SCL at (D)). Depending of the R/W bit the master or slave enables its output. If the bit is set, a master read operation is in progress (i.e., the slave drives the SDA line) The slave can hold the SCL line low after the acknowledge (E).
6. Multiple bytes can now be transmitted, all in same direction, until a stop condition is given by the master (F). Or a new start condition is given.

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

Figure 15-6. Start Condition Detector, Logic Diagram



15.2.5 Start Condition Detector

The start condition detector is shown in Figure 15-6 The SDA line is delayed (in the range of 50 to 300 ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in two-wire mode.

The start condition detector is working asynchronously and can therefore wake up the processor from the power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case the oscillator start-up time set by the CKSEL fuses (see Section 5.1 “Clock Systems and their Distribution” on page 19) must also be taken into the consideration. Refer to the USISIF bit description on page 95 for further details.

15.3 Alternative USI Usage

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

15.3.1 Half-duplex Asynchronous Data Transfer

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

15.3.2 4-bit Counter

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

15.3.3 12-bit Timer/Counter

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

15.3.4 Edge Triggered External Interrupt

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

15.3.5 Software Interrupt

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

15.4 USI Register Descriptions

15.4.1 USI Data Register – USIDR

Bit	7	6	5	4	3	2	1	0	
	MSB							**LSB**	**USIDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

When accessing the USI data register (USIDR) the serial register can be accessed directly. If a serial clock occurs at the same cycle the register is written, the register will contain the value written and no shift is performed. A (left) shift operation is performed depending of the USICS1..0 bits setting. The shift operation can be controlled by an external clock edge, by a Timer/Counter0 compare match, or directly by software using the USICLK strobe bit. Note that even when no wire mode is selected (USIWM1..0 = 0) both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the shift register.

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

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

15.4.2 USI Buffer Register – USIBR

Bit	7	6	5	4	3	2	1	0	
	MSB							**LSB**	**USIBR**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The content of the serial register is loaded to the USI buffer register when the trasfer is completed, and instead of accessing the USI data register (the serial register) the USI data buffer can be accessed when the CPU reads the received data. This gives the CPU time to handle other program tasks too as the controlling of the USI is not so timing critical. The USI flags as set same as when reading the USIDR register.

15.4.3 USI Status Register – USISR

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

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

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

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

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

A start condition interrupt will wakeup the processor from all sleep modes.

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

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

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

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

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

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

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

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

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

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a Timer/Counter0 compare match, or by software using USICLK or USITC strobe bits. The clock source depends of the setting of the USICS1..0 bits. For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by write a one to the USICLK bit while setting an external clock source (USICS1 = 1).

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

15.4.4 USI Control Register – USICR

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

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

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

Setting this bit to one enables the start condition detector interrupt. If there is a pending interrupt when the USISIE and the global interrupt enable flag is set to one, this will immediately be executed. Refer to the USISIF bit description on page 95 for further details.

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

Setting this bit to one enables the counter overflow interrupt. If there is a pending interrupt when the USIOIE and the global interrupt enable flag is set to one, this will immediately be executed. Refer to the USIOIF bit description on page 95 for further details.

- **Bit 5..4 – USIWM1..0: Wire Mode**

These bits set the type of wire mode to be used. Basically only the function of the outputs are affected by these bits. Data and clock inputs are not affected by the mode selected and will always have the same function. The counter and shift register can therefore be clocked externally, and data input sampled, even when outputs are disabled. The relations between USIWM1..0 and the USI operation is summarized in Table 15-1 on page 96.

Table 15-1. Relations between USIWM1..0 and the USI Operation

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

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

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

These bits set the clock source for the shift register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or Timer/Counter0 compare match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS1..0 bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the shift register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 15-2 shows the relationship between the USICS1..0 and USICLK setting and clock source used for the shift register and the 4-bit counter.

Table 15-2. Relations between the USICS1..0 and USICLK Setting

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

- **Bit 1 – USICLK: Clock Strobe**

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

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a clock select register. Setting the USICLK bit in this case will select the USITC strobe bit as clock source for the 4-bit counter (see Table 15-2 on page 97).

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

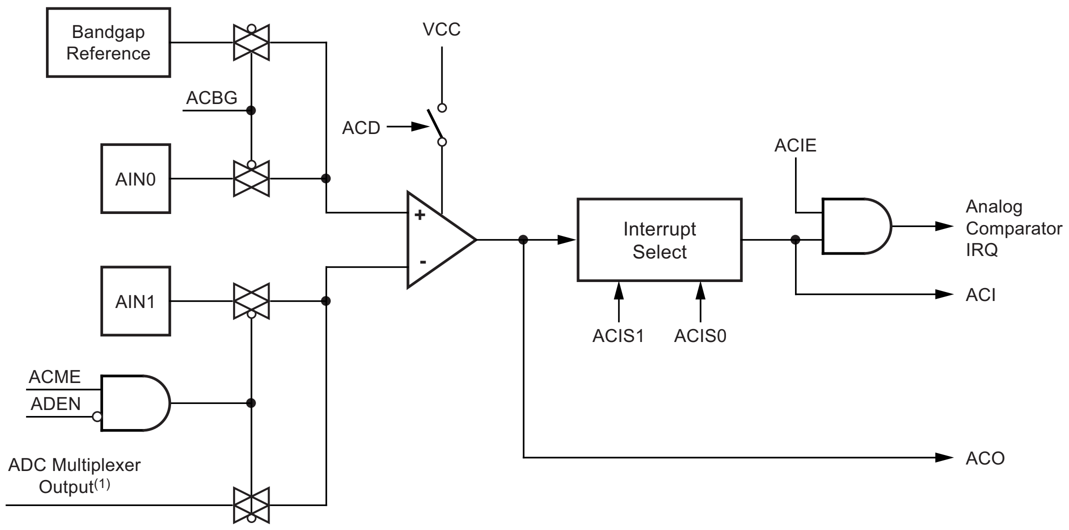
Writing a one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the data direction register, but if the PORT value is to be shown on the pin the DDRE4 must be set as output (to one). This feature allows easy clock generation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

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

16. 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 16-1.

Figure 16-1. Analog Comparator Block Diagram[2]



Notes: 1. See Table 16-2 on page 100.
2. Refer to Figure 1 on page 2 and Table 9-5 on page 52 for analog comparator pin placement.

16.1 ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	BIN	ACME	IPR	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the analog comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the analog comparator. For a detailed description of this bit, see Section 16.3 "Analog Comparator Multiplexed Input" on page 100.

16.2 Analog Comparator Control and Status Register – ACSR

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

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

When this bit is written 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.

This will reduce power consumption in Active and idle mode. When changing the ACD bit, the analog comparator interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set an internal 1.1V/2.56V reference voltage replaces the positive input to the analog comparator. The selection of the internal voltage reference is done by writing the REFS2..0 bits in ADMUX register. When this bit is cleared, AIN0 is applied to the positive input of the analog comparator.

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

The output of the analog comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 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 and the I-bit in the status register is set, the analog comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny25/45/85 and will always read as zero.

- **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 16-1.

Table 16-1. ACIS1/ACIS0 Settings

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

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

16.3 Analog Comparator Multiplexed Input

It is possible to select any of the ADC3..0 pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX1..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 16-2. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the analog comparator.

Table 16-2. Analog Comparator Multiplexed Input

ACME	ADEN	MUX1..0	Analog Comparator Negative Input
0	x	xx	AIN1
1	1	xx	AIN1
1	0	00	ADC0
1	0	01	ADC1
1	0	10	ADC2
1	0	11	ADC3

16.3.1 Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	–	–	ADC0D	ADC2D	ADC3D	ADC1D	AIN1D	AIN0D	DIDR0
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	

- **Bits 1, 0 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

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

17. Analog to Digital Converter

17.1 Features

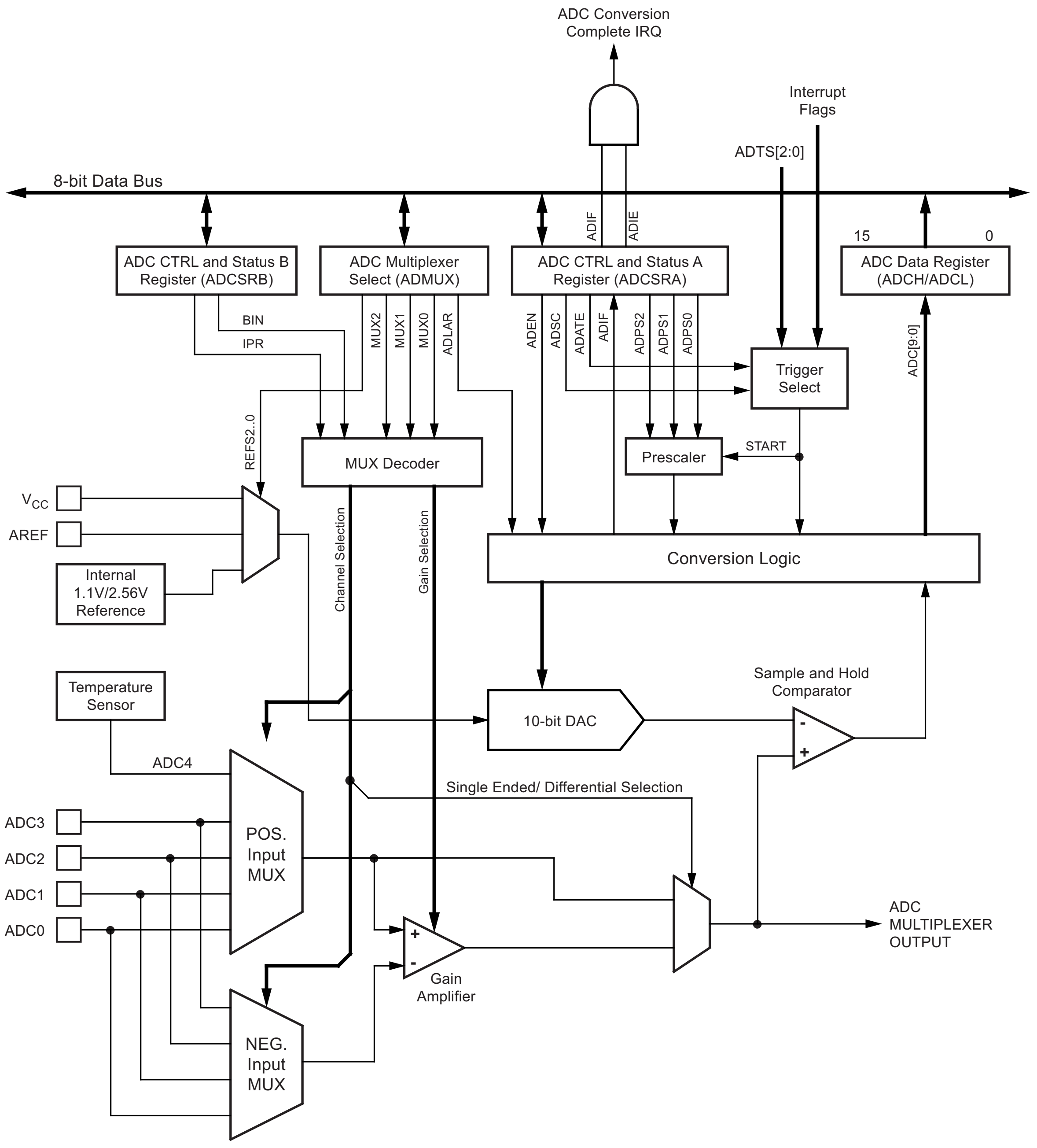
- 10-bit resolution
- 0.5 LSB integral non-linearity
- ±2 LSB absolute accuracy
- 65 - 260µs conversion time
- Up to 15kSPS at maximum resolution
- Four multiplexed single ended input channels
- Two differential input channels with selectable gain
- Temperature sensor input channel
- Optional left adjustment for ADC result readout
- 0 - V_{CC} ADC input voltage range
- Selectable 1.1V / 2.56V ADC voltage reference
- Free running or single conversion mode
- ADC start conversion by auto triggering on interrupt sources
- Interrupt on ADC conversion complete
- Sleep mode noise canceler
- Unipolar/bibilar input mode
- Input polarity reversal mode

The Atmel® ATtiny25/45/85 features a 10-bit successive approximation ADC. The ADC is connected to a 4-channel analog multiplexer which allows one differential voltage input and four single-ended voltage inputs constructed from the pins of port B. The differential input (PB3, PB4 or PB2, PB5) is equipped with a programmable gain stage, providing amplification step of 26dB (20x) on the differential input voltage before the A/D conversion. 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 17-1 on page 102.

Internal voltage references of nominally 1.1V or 2.56V are provided on-chip and these voltage references can optionally be externally decoupled at the AREF (PB0) pin by a capacitor, for better noise performance. Alternatively, V_{CC} can be used as voltage reference for single ended channels. There is also an option to use an external voltage reference and turn-off the internal voltage reference. These options are selected using the REFS2..0 bits of the ADMUX control register.

Figure 17-1. Analog to Digital Converter Block Schematic



17.2 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on V_{CC}, the voltage on the AREF pin or an internal 1.1V/2.56V voltage reference.

The voltage reference for the ADC may be selected by writing to the REFS2..0 bits in ADMUX. The VCC supply, the AREF pin or an internal 1.1V/2.56V voltage reference may be selected as the ADC voltage reference. Optionally the internal 1.1V/2.56V voltage reference may be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX3..0 bits in ADMUX. Any of the four ADC input pins ADC3..0 can be selected as single ended inputs to the ADC. ADC2 or ADC0 can be selected as positive input and ADC0, ADC1, ADC2 or ADC3 can be selected as negative input to the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input pair by the selected gain factor, 1x or 20x, according to the setting of the MUX3..0 bits in ADMUX. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

If ADC0 or ADC2 is selected as both the positive and negative input to the differential gain amplifier (ADC0-ADC0 or ADC2-ADC2), the remaining offset in the gain stage and conversion circuitry can be measured directly as the result of the conversion. This figure can be subtracted from subsequent conversions with the same gain setting to reduce offset error to below 1 LSW.

The on-chip temperature sensor is selected by writing the code "1111" to the MUX3..0 bits in ADMUX register when the ADC4 channel is used as an ADC input.

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

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

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

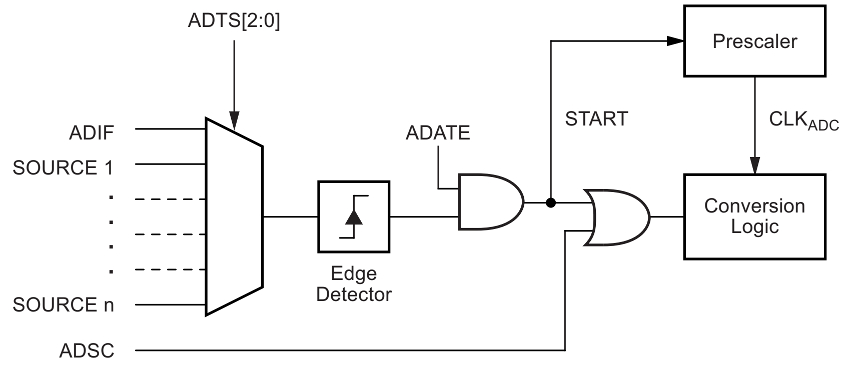
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

17.3 Starting a Conversion

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

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

Figure 17-2. ADC Auto Trigger Logic

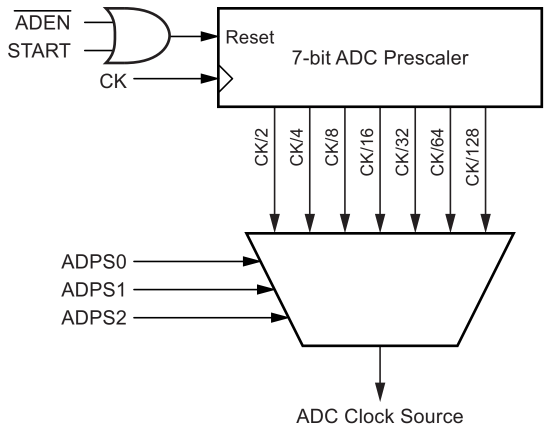


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the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

If auto triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to 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.

17.4 Prescaling and Conversion Timing

Figure 17-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate. It is not recommended to use a higher input clock frequency than 1MHz.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

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

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

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

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

In free running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 17-1 on page 106.

Figure 17-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

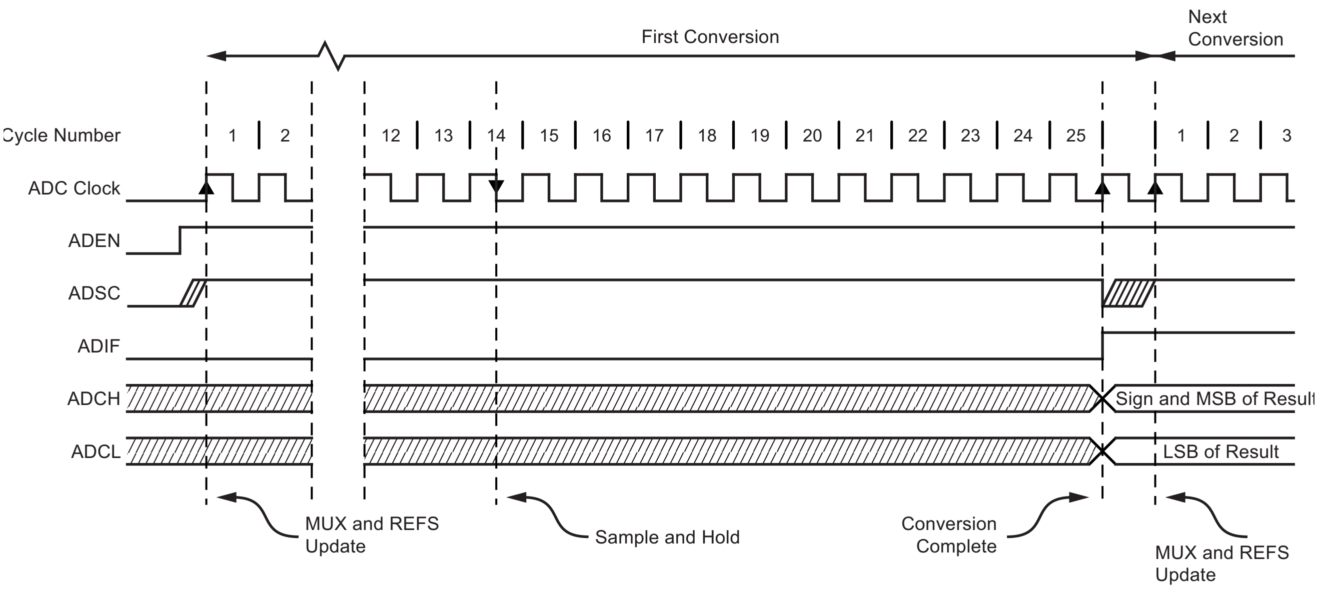


Figure 17-5. ADC Timing Diagram, Single Conversion

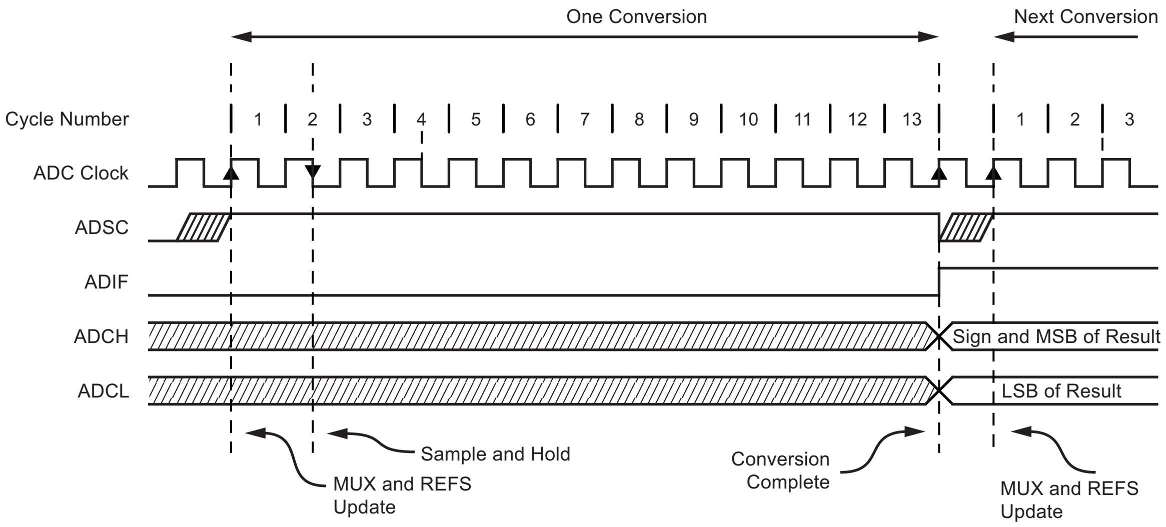


Figure 17-6. ADC Timing Diagram, Auto Triggered Conversion

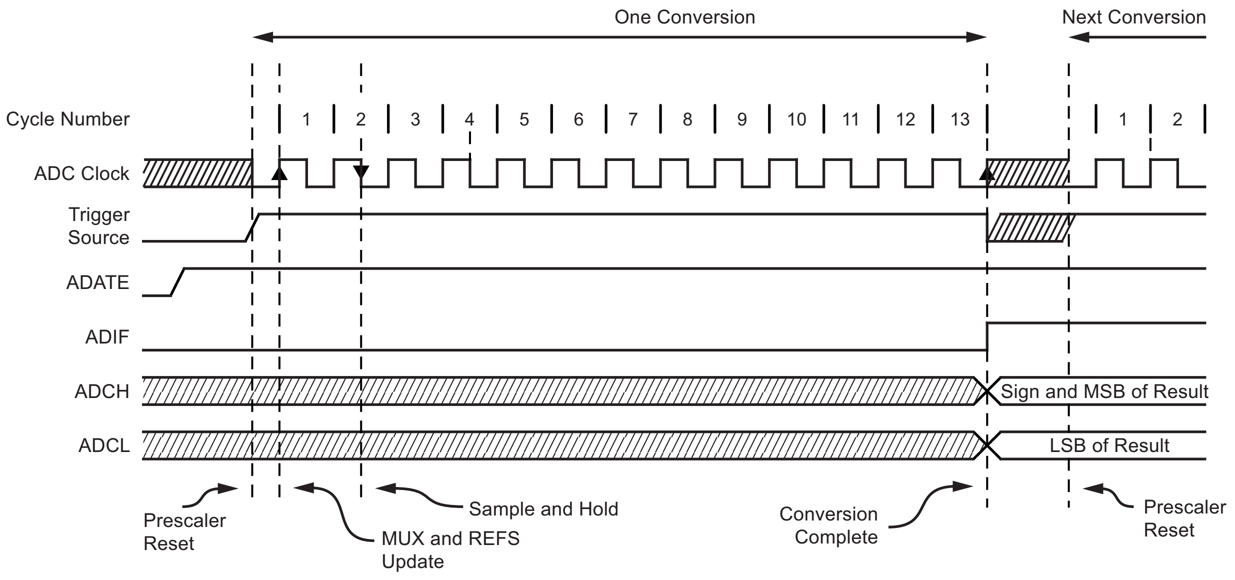


Figure 17-7. ADC Timing Diagram, Free Running Conversion

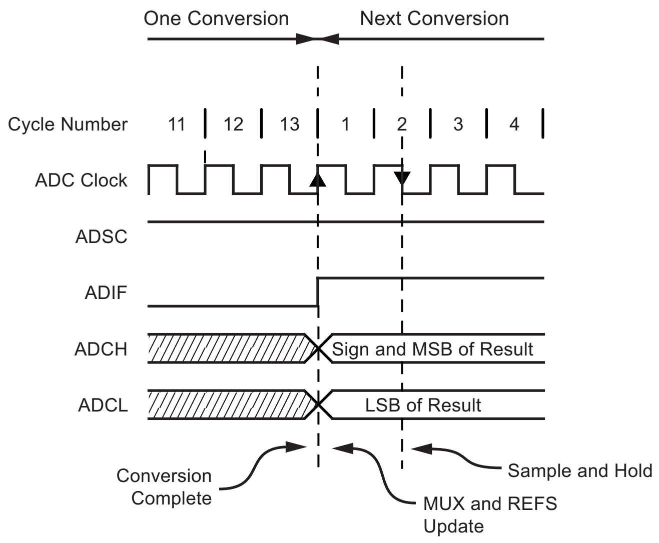


Table 17-1. ADC Conversion Time

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

17.5 Changing Channel or Reference Selection

The MUX3..0 and REFS2..0 bits in the ADMUX register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and voltage reference selection only takes place at a safe point during the conversion. The channel and voltage reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and voltage reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or voltage reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If auto triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX register, in order to control which conversion will be affected by the new 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:

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

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

17.5.1 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.

17.5.2 ADC Voltage Reference

The voltage reference for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either V_{CC}, or internal 1.1V/2.56V voltage reference, or external AREF pin. The first ADC conversion result after switching voltage reference source may be inaccurate, and the user is advised to discard this result.

17.6 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:

a. Make sure that the ADC is enabled and is not busy converting. Single conversion mode must be selected and the ADC conversion complete interrupt must be enabled.
b. Enter ADC noise reduction mode (or idle mode). The ADC will start a conversion once the CPU has been halted.
c. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC conversion complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC conversion complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering 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.

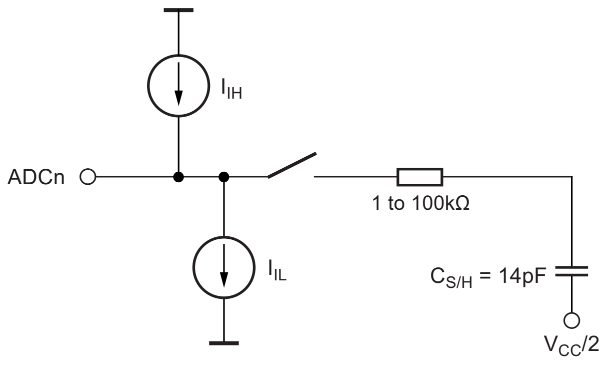
17.6.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 17-8. 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 capacitor through the series resistance (combined resistance in the input path).

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

Signal components higher than the nyquist frequency ($f_{ADC}/2$) should not be present to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 17-8. Analog Input Circuitry



17.6.2 Analog Noise Canceling Techniques

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

a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. Use the ADC noise canceler function to reduce induced noise from the CPU.

c. If any port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

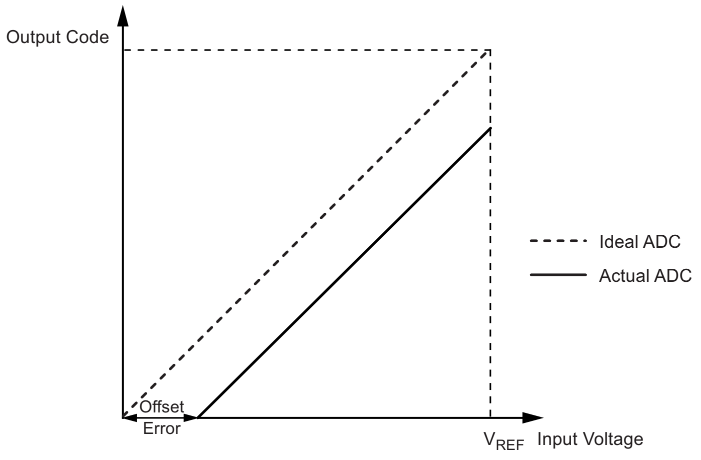
17.6.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n–1.

Several parameters describe the deviation from the ideal behavior:

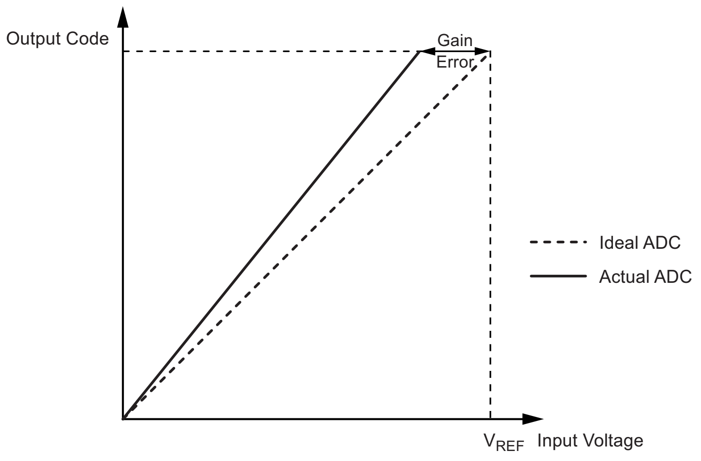
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 17-9. Offset Error



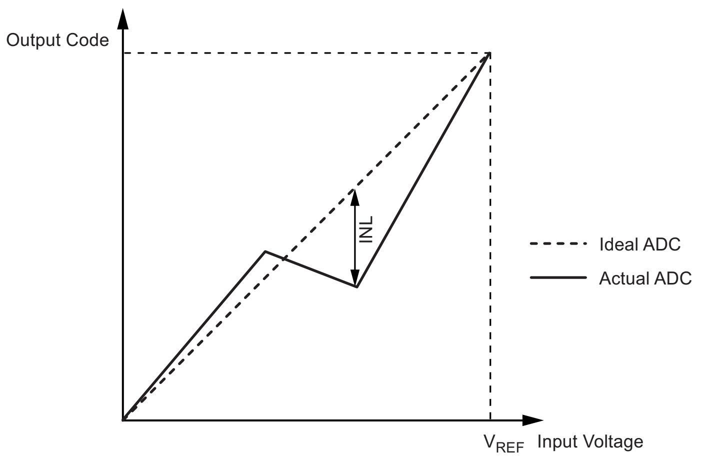
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 17-10. Gain Error



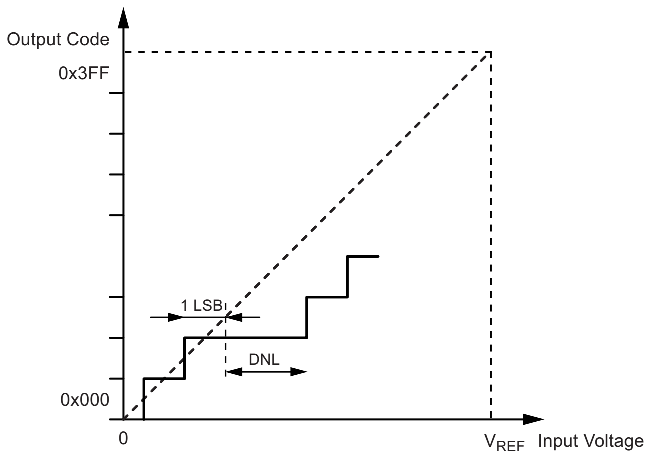
- Integral non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 17-11. 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 17-12. 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.

17.7 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversion as there are three types of conversions: single ended conversion, unipolar differential conversion and bipolar differential conversion.

17.7.1 Single Ended Conversion

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \times 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 17-3 on page 112 and Table 17-4 on page 113). 0x000 represents analog ground, and 0x3FF represents the selected voltage reference minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

17.7.2 Unipolar Differential Conversion

If differential channels and an unipolar input mode are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \times 1024}{V_{REF}} \times GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference (see Table 17-3 on page 112 and Table 17-4 on page 113). The voltage on the positive pin must always be larger than the voltage on the negative pin or otherwise the voltage difference is saturated to zero. The result is presented in one-sided form, from 0x000 (0d) to 0x3FF (+1023d). The GAIN is either 1x or 20x.

17.7.3 Bipolar Differential Conversion

As default the ADC converter operates in the unipolar input mode, but the bipolar input mode can be selected by writing the BIN bit in the ADCSRB to one. In the bipolar input mode two-sided voltage differences are allowed and thus the voltage on the negative input pin can also be larger than the voltage on the positive input pin. If differential channels and a bipolar input mode are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \times 512}{V_{REF}} \times GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (–512d) through 0x000 (+0d) to 0x1FF (+511d). The GAIN is either 1x or 20x.

However, if the signal is not bipolar by nature (9 bits + sign as the 10th bit), this scheme loses one bit of the converter dynamic range. Then, if the user wants to perform the conversion with the maximum dynamic range, the user can perform a quick polarity check of the result and use the unipolar differential conversion with selectable differential input pairs (see the input polarity reversal mode ie. the IPR bit in the Section 17.7.8 "ADC Control and Status Register B – ADCSRB" on page 115). When the polarity check is performed, it is sufficient to read the MSB of the result (ADC9 in ADCH). If the bit is one, the result is negative, and if this bit is zero, the result is positive.

17.7.4 Temperature Measurement (Preliminary description)

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC4 channel. Selecting the ADC4 channel by writing the MUX3..0 bits in ADMUX register to “1111” enables the temperature sensor. The internal 1.1V voltage reference must also be selected for the ADC voltage reference source in the temperature sensor measurement. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor.

The measured voltage has a linear relationship to the temperature as described in Table 17-2. The voltage sensitivity is approximately 1mV/°C and the accuracy of the temperature measurement is ±10°C after bandgap calibration.

Table 17-2. Temperature versus Sensor Output Voltage (Typical Case)

Temperature/°C	–45°C	+25°C	+105°C
Voltage/mV	242mV	314mv	403mV

The values described in Table 17-2 are typical values. However, due to the process variation the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results the temperature measurement can be calibrated in the application software. The software calibration requires that a calibration value is measured and stored in a register or EEPROM for each chip, as a part of the production test. The software calibration can be done utilizing the formula:

Temperature = $k \times V_{TEMP} + T_{OS}$

where V_{TEMP} is the ADC reading of the temperature sensor signal, k is a fixed coefficient and T_{OS} is the temperature sensor offset value determined and stored into EEPROM as a part of production test.

17.7.5 ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	REFS2	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..6,4 – REFS2..REFS0: Voltage Reference Selection Bits**

These bits select the voltage reference (V_{REF}) for the ADC, as shown in Table 17-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set). Whenever these bits are changed, the next conversion will take 25 ADC clock cycles. If active channels are used, using V_{CC} or an external AREF higher than (V_{CC} – 1V) as a voltage reference is not recommended, as this will affect the ADC accuracy.

Table 17-3. Voltage Reference Selections for ADC

REFS2	REFS1	REFS0	Voltage Reference (V_{REF}) Selection
0	0	0	V_{CC} used as voltage reference, disconnected from PB0 (AREF).
0	0	1	External voltage reference at PB0 (AREF) pin, internal voltage reference turned off.
0	1	0	Internal 1.1V voltage reference without external bypass capacitor, disconnected from PB0 (AREF).
0	1	1	Internal 1.1V voltage reference with external bypass capacitor at PB0 (AREF) pin.
1	1	0	Internal 2.56V voltage reference without external bypass capacitor, disconnected from PB0 (AREF).(1)
1	1	1	Internal 2.56V voltage reference with external bypass capacitor at PB0 (AREF) pin.(1)

Note: 1. The device requires a supply voltage of 3V in order to generate 2.56V reference voltage.

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC data register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC data register immediately, regardless of any ongoing conversions. For a complete description of this bit, see Section 17.7.7 "The ADC Data Register – ADCL and ADCH" on page 115.

- **Bits 3:0 – MUX3:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. In case of differential input (ADC0 - ADC1 or ADC2 - ADC3), gain selection is also made with these bits. Selecting ADC2 or ADC0 as both inputs to the differential gain stage enables offset measurements. Selecting the single-ended channel ADC4 enables the temperature sensor. Refer to Table 17-4 on page 113 for details. If these bits are changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSRA is set).

Table 17-4. Input Channel Selections

MUX3..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
0000	ADC0 (PB5)	N/A		
0001	ADC1 (PB2)			
0010	ADC2 (PB4)			
0011	ADC3 (PB3)			
0100	N/A	ADC2 (PB3)	ADC2 (PB3)	1x
0101[1]		ADC2 (PB3)	ADC2 (PB3)	20x
0110		ADC2 (PB3)	ADC3 (PB4)	1x
0111		ADC2 (PB3)	ADC3 (PB4)	20x
1000		ADC0 (PB5)	ADC0 (PB5)	1x
1001		ADC0 (PB5)	ADC0 (PB5)	20x
1010		ADC0 (PB5)	ADC1 (PB2)	1x
1011		ADC0 (PB5)	ADC1 (PB2)	20x
1100	1.1V/2.56V	N/A		
1101	0V			
1110	N/A			
1111	ADC4[2]			

Notes: 1. For offset calibration only. See Section 17.2 "Operation" on page 102

2. For temperature sensor

17.7.6 ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	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 executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a read-modify-write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

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

When this bit is written to one and the I-bit in SREG is set, the ADC conversion complete interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 17-5. 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

17.7.7 The ADC Data Register – ADCL and ADCH

17.7.7.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	**ADCH**
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

17.7.7.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	**ADCH**
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC data register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in Section 17.7 "ADC Conversion Result" on page 111.

17.7.8 ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	BIN	ACME	IPR	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– BIN: Bipolar Input Mode**

The gain stage is working in the unipolar mode as default, but the bipolar mode can be selected by writing the BIN bit in the ADCSRB register. In the unipolar mode only one-sided conversions are supported and the voltage on the positive input must always be larger than the voltage on the negative input. Otherwise the result is saturated to the voltage reference. In the bipolar mode two-sided conversions are supported and the result is represented in the two's complement form. In the unipolar mode the resolution is 10 bits and the bipolar mode the resolution is 9 bits + 1 sign bit.

- **Bit 5 – IPR: Input Polarity Mode**

The input polarity mode allows software selectable differential input pairs and full 10 bit ADC resolution, in the unipolar input mode, assuming a pre-determined input polarity. If the input polarity is not known it is actually possible to determine the polarity first by using the bipolar input mode (with 9 bit resolution + 1 sign bit ADC measurement). And once determined, set or clear the polarity reversal bit, as needed, for a succeeding 10 bit unipolar measurement.

- **Bits 4..3 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as 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 17-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free running mode
0	0	1	Analog comparator
0	1	0	External interrupt request 0
0	1	1	Timer/Counter compare match A
1	0	0	Timer/Counter overflow
1	0	1	Timer/Counter compare match B
1	1	0	Pin change interrupt request

17.7.9 Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	–	–	ADC0D	ADC2D	ADC3D	ADC1D	AIN1D	AIN0D	DIDR0
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	

- **Bits 5..2 – 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.

18. debugWIRE On-chip Debug System

18.1 Features

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

18.2 Overview

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

18.3 Physical Interface

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

Figure 18-1. The debugWIRE Setup

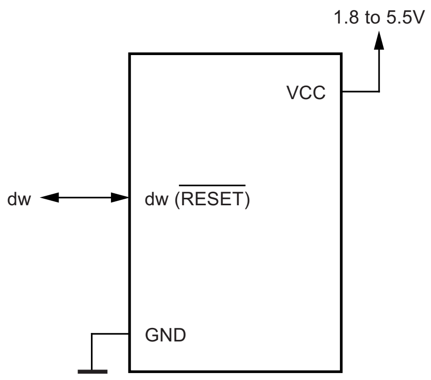


Figure 18-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

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

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

18.4 Software Break Points

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

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

18.5 Limitations of debugWIRE

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

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

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

18.6 debugWIRE Related Register in I/O Memory

The following section describes the registers used with the debugWire.

18.6.1 debugWire Data Register – DWDR

Bit	7	6	5	4	3	2	1	0	
	DWDR[7:0]								**DWDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The DWDR register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

19. Self-Programming the Flash

The device provides a self-programming mechanism for downloading and uploading program code by the MCU itself. The self-programming can use any available data interface and associated protocol to read code and write (program) that code into the program memory.

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

Alternative 1, fill the buffer before a page erase

- Fill temporary page buffer
- Perform a page erase
- Perform a page write

Alternative 2, fill the buffer after page erase

- Perform a page erase
- Fill temporary page buffer
- Perform a page write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be re-written. When using alternative 1, the boot loader provides an effective read-modify-write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page.

19.1 Performing Page Erase by SPM

To execute page erase, set up the address in the Z-pointer, write "00000011" to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- The CPU is halted during the page erase operation.

19.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write "00000001" to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a Page Write operation or by writing the CTPB bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM page load operation, all data loaded will be lost.

19.3 Performing a Page Write

To execute page write, set up the address in the Z-pointer, write "00000101" to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- The CPU is halted during the page write operation.

19.4 Addressing the Flash During Self-Programming

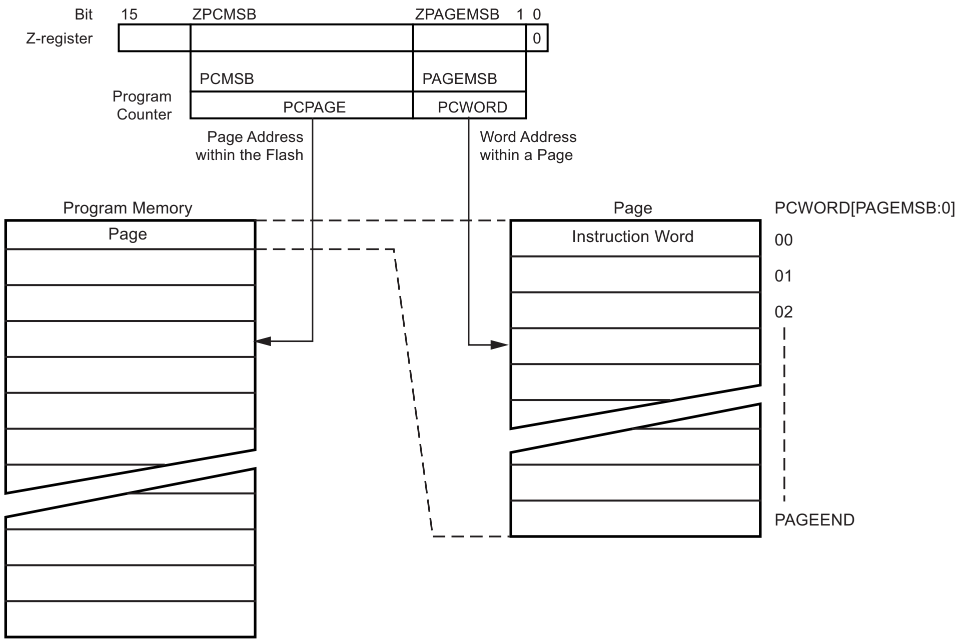
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the flash is organized in pages (see Table 20-6 on page 126), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 19-1. Note that the page erase and page write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the page erase and page write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 19-1. Addressing the Flash During SPM[1]



Note: 1. The different variables used in Figure 19-1 are listed in Table 20-6 on page 126.

19.4.1 Store Program Memory Control and Status Register – SPMCSR

The store program memory control and status register contains the control bits needed to control the program memory operations.

Bit	7	6	5	4	3	2	1	0	
	–	**–**	–	CTPB	**RFLB**	**PGWRT**	**PGERS**	**SPMEN**	**SPMCSR**
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and always read as zero.

- **Bit 4 – CTPB: Clear Temporary Page Buffer**

If the CTPB bit is written while filling the temporary page buffer, the temporary page buffer will be cleared and the data will be lost.

- **Bit 3 – RFLB: Read Fuse and Lock Bits**

An LPM instruction within three cycles after RFLB and SPMEN are set in the SPMCSR register, will read either the lock bits or the fuse bits (depending on Z0 in the Z-pointer) into the destination register. See Section 19.4.2 “EEPROM Write Prevents Writing to SPMCSR” on page 121 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page Write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either CTPB, RFLB, PGWRT, or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During page erase and page write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than “10001”, “01001”, “00101”, “00011” or “00001” in the lower five bits will have no effect.

19.4.2 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to flash. Reading the fuses and lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEWE) in the EECR register and verifies that the bit is cleared before writing to the SPMCSR register.

19.4.3 Reading the Fuse and Lock Bits from Software

It is possible to read both the fuse and lock bits from software. To read the lock bits, load the Z-pointer with 0x0001 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the RFLB and SPMEN bits are set in SPMCSR, the value of the lock bits will be loaded in the destination register. The RFLB and SPMEN bits will auto-clear upon completion of reading the lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	**LB2**	**LB1**

The algorithm for reading the fuse low byte is similar to the one described above for reading the lock bits. To read the fuse low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse low byte (FLB) will be loaded in the destination register as shown below. Refer to Table 20-5 on page 125 for a detailed description and mapping of the fuse low byte.

Bit	7	6	5	4	3	2	1	0
Rd	**FLB7**	**FLB6**	**FLB5**	**FLB4**	**FLB3**	**FLB2**	**FLB1**	**FLB0**

Similarly, when reading the fuse high byte, load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse high byte (FHB) will be loaded in the destination register as shown below. Refer to Table 20-4 for detailed description and mapping of the fuse high byte.

Bit	7	6	5	4	3	2	1	0
Rd	**FHB7**	**FHB6**	**FHB5**	**FHB4**	**FHB3**	**FHB2**	**FHB1**	**FHB0**

Fuse and lock bits that are programmed, will be read as zero. Fuse and lock bits that are unprogrammed, will be read as one.

19.4.4 Preventing Flash Corruption

During periods of low V_{CC}, the flash program can be corrupted because the supply voltage is too low for the CPU and the flash to operate properly. These issues are the same as for board level systems using the flash, and the same design solutions should be applied.

A flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR® RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal brown-out detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
2. Keep the AVR core in power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR register and thus the flash from unintentional writes.

19.4.5 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 19-1 shows the typical programming time for flash accesses from the CPU.

Table 19-1. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7ms	4.5ms

20. Memory Programming

This section describes the different methods for programming the Atmel® ATtiny25/45/85 memories.

20.1 Program And Data Memory Lock Bits

The ATtiny25/45/85 provides two lock bits which can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional security listed in Table 20-2. The lock bits can only be erased to "1" with the chip erase command.

Program memory can be read out via the debugWIRE interface when the DWEN fuse is programmed, even if the lock bits are set. Thus, when lock bit security is required, should always debugWIRE be disabled by clearing the DWEN fuse.

Table 20-1. Lock Bit Byte[1]

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
	5	–	1 (unprogrammed)
	4	–	1 (unprogrammed)
	3	–	1 (unprogrammed)
	2	–	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. "1" means unprogrammed, "0" means programmed

Table 20-2. Lock Bit Protection Modes[1][2]

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the flash and EEPROM is disabled in high-voltage and serial programming mode. The fuse bits are locked in both serial and high-voltage programming mode[1]. debugWire is disabled.
3	0	0	Further programming and verification of the flash and EEPROM is disabled in high-voltage and serial programming mode. The fuse bits are locked in both serial and high-voltage programming mode[1]. debugWire is disabled.

Notes: 1. Program the fuse bits before programming the LB1 and LB2.

2. "1" means unprogrammed, "0" means programmed

20.2 Fuse Bytes

The Atmel® ATtiny25/45/85 has three fuse bytes. Table 20-4, Table 20-5 and Table 20-6 on page 126 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 20-3. Fuse Extended Byte

Fuse High Byte	Bit No	Description	Default Value
	7	-	1 (unprogrammed)
	6	-	1 (unprogrammed)
	5	-	1 (unprogrammed)
	4	-	1 (unprogrammed)
	3	-	1 (unprogrammed)
	2	-	1 (unprogrammed)
	1	-	1 (unprogrammed)
SELFPRGEN	0	Self-programming enable	1 (unprogrammed)

Table 20-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External reset disable	1 (unprogrammed)
DWEN(2)	6	DebugWIRE enable	1 (unprogrammed)
SPIEN(3)	5	Enable serial program and data downloading	0 (programmed, SPI prog. enabled)
WDTON(4)	4	Watchdog timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the chip erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(5)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1(5)	1	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL0(5)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes:
1. See Section 9.3.2 “Alternate Functions of Port B” on page 50 for description of RSTDISBL and DWEN fuses.
2. DWEN must be unprogrammed when lock bit security is required.
   See Section 20.1 “Program And Data Memory Lock Bits” on page 123
3. The SPIEN fuse is not accessible in SPI programming mode.
4. See Section 7.9.1 “Watchdog Timer Control Register – WDTCR” on page 38 for details.
5. See Table 7-2 on page 35 for BODLEVEL fuse decoding.
6. When programming the RSTDISBL fuse, high-voltage serial programming has to be used to change fuses to perform further programming.

Table 20-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8[1]	7	Divide clock by 8	0 (unprogrammed)
CKOUT[2]	6	Clock output enable	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[3]
SUT0	4	Select start-up time	0 (programmed)[3]
CKSEL3	3	Select clock source	0 (programmed)[4]
CKSEL2	2	Select clock source	0 (programmed)[4]
CKSEL1	1	Select clock source	1 (unprogrammed)[4]
CKSEL0	0	Select clock source	0 (programmed)[4]

Notes:
1. See Section 5.10 "System Clock Prescaler" on page 26 for details.
2. The CKOUT fuse allows the system clock to be output on PORTB4. See "Section 5.9 "Clock Output Buffer" on page 26 for details.
3. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 5-7 on page 23 for details.
4. The default setting of CKSEL1..0 results in internal RC oscillator at 8.0MHz. See Table 5-6 on page 23 for details.

The status of the fuse bits is not affected by chip erase. Note that the fuse bits are locked if lock bit1 (LB1) is programmed. program the fuse bits before programming the lock bits.

20.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves programming mode. This does not apply to the EESAVE fuse which will take effect once it is programmed. The fuses are also latched on power-up in normal mode.

20.3 Signature Bytes

All Atmel® microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and high-voltage programming mode, also when the device is locked. The three bytes reside in a separate address space.

20.3.1 ATtiny25 Signature Bytes

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x91 (indicates 2KB flash memory).
3. 0x002: 0x08 (indicates ATtiny25 device when 0x001 is 0x91).

20.3.2 ATtiny45 Signature Bytes

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x92 (indicates 4KB flash memory).
3. 0x002: 0x06 (indicates ATtiny45 device when 0x001 is 0x92).

20.3.3 ATtiny85 Signature Bytes

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x93 (indicates 8KB flash memory).
3. 0x002: 0x0B (indicates ATtiny85 device when 0x001 is 0x93).

20.4 Calibration Byte

Signature area of the Atmel ATtiny25/45/85 has one byte of calibration data for the internal RC oscillator. This byte resides in the high byte of address 0x000. During reset, this byte is automatically written into the OSCCAL register to ensure correct frequency of the calibrated RC oscillator.

20.5 Page Size

Table 20-6. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATtiny25	1Kwords (2Kbytes)	16 words	PC[3:0]	64	PC[9:4]	9
ATtiny45	2Kwords (4Kbytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATtiny85	4Kwords (8Kbytes)	32 words	PC[4:0]	128	PC[11:5]	11

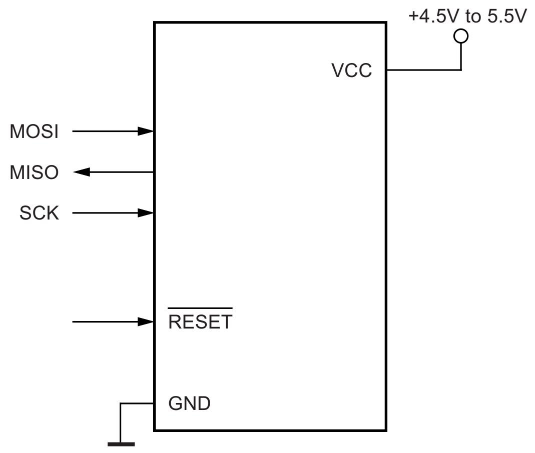
Table 20-7. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATtiny25	128 bytes	4 bytes	EEA[1:0]	32	EEA[6:2]	6
ATtiny45	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7
ATtiny85	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

20.6 Serial Downloading

Both the flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the programming enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 20-8 on page 126, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 20-1. Serial Programming and Verify[1]



Note: 1. If the device is clocked by the internal oscillator, it is no need to connect a clock source to the CLKI pin.

Table 20-8. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB0	I	Serial data in
MISO	PB1	O	Serial data out
SCK	PB2	I	Serial clock

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the chip erase instruction. The chip erase operation turns the content of every memory location in both the program and EEPROM arrays into 0xFF.

Depending on CKSEL fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2CPU clock cycles for f_{ck} < 12MHz, 3CPU clock cycles for $f_{ck} \geq$ 12MHz

High: > 2CPU clock cycles for f_{ck} < 12MHz, 3CPU clock cycles for $f_{ck} \geq$ 12MHz

20.6.1 Serial Programming Algorithm

When writing serial data to the Atmel® ATtiny25/45/85, data is clocked on the rising edge of SCK.

When reading data from the Atmel ATtiny25/45/85, data is clocked on the falling edge of SCK. See Figure 20-2 on page 128 and Figure 20-3 on page 129 for timing details.

To program and verify the ATtiny25/45/85 in the serial programming mode, the following sequence is recommended (see four byte instruction formats in Table 20-10 on page 128):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to "0". In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to "0".
2. Wait for at least 20ms and enable serial programming by sending the programming enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the programming enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new programming enable command.
4. The flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the load program memory page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The program memory page is stored by loading the write program memory page instruction with the 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page.
   (See Table 20-9 on page 128.) Accessing the serial programming interface before the flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 20-9 on page 128.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the Load EEPROM memory page instruction. The EEPROM memory page is stored by loading the write EEPROM memory page Instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM memory page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 20-7 on page 126). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to "1".
   Turn V_{CC} power off.

Table 20-9. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	4.0ms
t_{WD_ERASE}	4.0ms
t_{WD_FUSE}	4.5ms

Figure 20-2. Serial Programming Waveforms

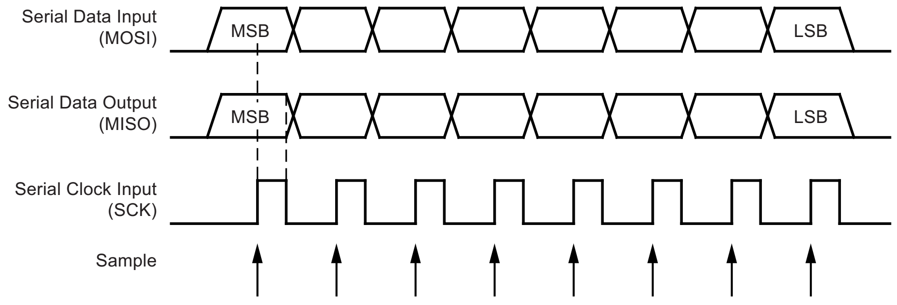


Table 20-10. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming after $\overline{\text{RESET}}$ goes low.
Chip erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase EEPROM and flash.
Read program memory	0010 **H**000	0000 000**a**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from program memory at word address **a**:**b**.
Load program memory page	0100 **H**000	000x xxxx	xxx**b bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to program memory page at word address **b**. Data low byte must be loaded before data high byte is applied within the same address.
Write program memory page	0100 1100	0000 000**a**	**bb**xx xxxx	xxxx xxxx	Write Program memory page at address **a**:**b**.
Read EEPROM memory	1010 0000	000x xxxx	xx**bb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**.
Write EEPROM memory	1100 0000	000x xxxx	xx**bb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**.
Load EEPROM memory page (Page access)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Write EEPROM memory Page (page access)	1100 0010	00xx xxxx	xx**bb bb00**	xxxx xxxx	Write EEPROM page at address **b**.
Read lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. "0" = programmed, "1" = unprogrammed. See Table 20-1 on page 123 for details.

Note: **a** = address high bits, **b** = address low bits, **H** = 0 – Low byte, 1 – high byte, **o** = data out, **i** = data in, x = don't care

Table 20-10. Serial Programming Instruction Set (Continued)

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Write lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write lock bits. Set bits = “0” to program lock bits. See Table 20-1 on page 123 for details.
Read signature byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read signature byte **o** at address **b**.
Write fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 20-5 on page 125 for details.
Write fuse high bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 20-4 on page 124 for details.
Write extended fuse bits	1010 1100	1010 0100	xxxx xxxx	**xxxx xxxi**	Set bits = “0” to program, “1” to unprogram. See Table 20-3 on page 124 for details.
Read fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read fuse bits. “0” = programmed, “1” = unprogrammed. See Table 20-5 on page 125 for details.
Read fuse high bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read fuse high bits. “0” = pro-grammed, “1” = unprogrammed. See Table 20-4 on page 124 for details.
Read extended fuse bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read extended fuse bits. “0” = pro-grammed, “1” = unprogrammed. See Table 20-3 on page 124 for details.
Read calibration byte	0011 1000	000x xxxx	0000 0000	**oooo oooo**	Read calibration byte
Poll RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	If **o** = “1”, a programming operation is still busy. Wait until this bit returns to “0” before applying another command.

Note: **a** = address high bits, **b** = address low bits, **H** = 0 – Low byte, 1 – high byte, **o** = data out, **i** = data in, x = don’t care

20.6.2 Serial Programming Characteristics

Figure 20-3. Serial Programming Timing

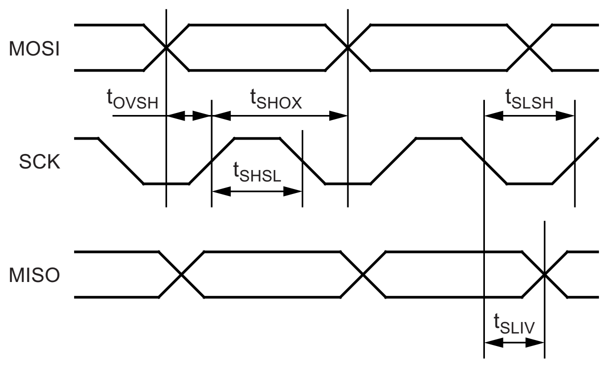


Table 20-11. Serial Programming Characteristics, T_A = –40°C to 125°C, V_{CC} = 2.7 to 5.5V (Unless Otherwise Noted)

Parameter	Symbol	Min	Typ	Max	Units
Oscillator frequency (ATtiny25/45/85V)	$1/t_{CLCL}$	0		4	MHz
Oscillator period (ATtiny25/45/85V)	t_{CLCL}	250			ns
Oscillator frequency (ATtiny25/45/85L, VCC = 2.7 to 5.5V)	$1/t_{CLCL}$	0		10	MHz
Oscillator period (ATtiny25/45/85L, VCC = 2.7 to 5.5V)	t_{CLCL}	100			ns
Oscillator frequency (ATtiny25/45/85, V_{CC} = 4.5V to 5.5V)	$1/t_{CLCL}$	0		20	MHz
Oscillator period (ATtiny25/45/85, V_{CC} = 4.5V to 5.5V)	t_{CLCL}	50			ns
SCK pulse width high	t_{SHSL}	2 t_{CLCL*}			ns
SCK pulse width low	t_{SLSH}	2 t_{CLCL*}			ns
MOSI setup to SCK high	t_{OVSH}	t_{CLCL}			ns
MOSI hold after SCK high	t_{SHOX}	2 t_{CLCL}			ns

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

20.7 High-voltage Serial Programming

This section describes how to program and verify flash program memory, EEPROM data memory, lock bits and fuse bits in the Atmel® ATtiny25/45/85.

Figure 20-4. High-voltage Serial Programming

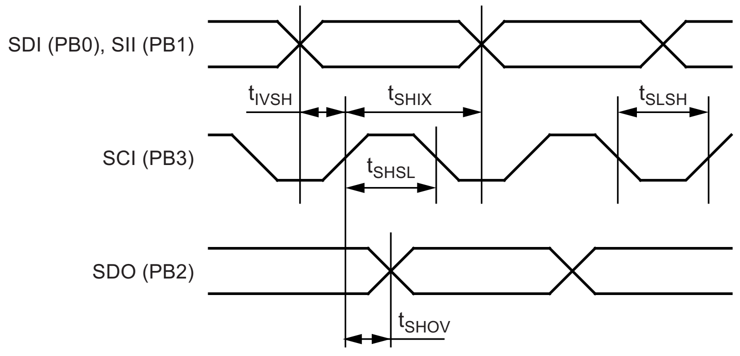


Table 20-12. Pin Name Mapping

Signal Name in High-voltage Serial Programming Mode	Pin Name	I/O	Function
SDI	PB0	I	Serial data input
SII	PB1	I	Serial instruction input
SDO	PB2	O	Serial data output
SCI	PB3	I	Serial clock input (min. 220ns period)

Table 20-13. High-voltage Serial Programming Characteristics T_A = 25°C ±10%, V_{CC} = 5.0V ±10% (Unless otherwise noted)

Parameter	Symbol	Min	Typ	Max	Units
SCI (PB3) pulse width high	t_{SHSL}	125			ns
SCI (PB3) pulse width low	t_{SLSH}	125			ns
SDI (PB0), SII (PB1) Valid to SCI (PB3) high	t_{IVSH}	50			ns
SDI (PB0), SII (PB1) hold after SCI (PB3) high	t_{SHIX}	50			ns
SCI (PB3) high to SDO (PB2) valid	t_{SHOV}		16		ns
Wait after Instr. 3 for write fuse bits	t_{WLWH_PFB}		2.5		ms

Table 20-14. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
SDI	Prog_enable[0]	0
SII	Prog_enable[1]	0
SDO	Prog_enable[2]	0

20.8 High-voltage Serial Programming Algorithm Sequence

To program and verify the Atmel® ATtiny25/45/85 in the high-voltage serial programming mode, the following sequence is recommended (See instruction formats in Table 20-16):

20.8.1 Enter High-voltage Serial Programming Mode

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

1. Apply 4.5 to 5.5V between V_{CC} and GND.
2. Set RESET pin to "0" and toggle SCI at least six times.
3. Set the prog_enable pins listed in Table 20-14 to "000" and wait at least 100ns.
4. Apply V_{HVRST} – 5.5V to RESET. Keep the prog_enable pins unchanged for at least t_{HVRST} after the high-voltage has been applied to ensure the prog_enable signature has been latched.
5. Shortly after latching the prog_enable signature, the device will actively output data on the prog_enable[2]/SDO pin, and the resulting drive contention may increase the power consumption. To minimize this drive contention, release the prog_enable[2] pin after t_{HVRST} has elapsed.
6. Wait at least 50µs before giving any serial instructions on SDI/SII.

Table 20-15. High-voltage Reset Characteristics

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

20.8.2 Considerations for Efficient Programming

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

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

20.8.3 Chip Erase

The chip erase will erase the flash and EEPROM[1] memories plus lock bits. The lock bits are not reset until the program memory has been completely erased. The fuse bits are not changed. A chip erase must be performed before the flash and/or EEPROM are re-programmed.

Note: 1. The EEPROM memory is preserved during chip erase if the EESAVE fuse is programmed.

1. Load command "chip erase" (see Table 20-16 on page 134).
2. Wait after instr. 3 until SDO goes high for the "chip erase" cycle to finish.
3. Load command "no operation".

20.8.4 Programming the Flash

The flash is organized in pages, see Table 20-10 on page 128. When programming the flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire flash memory:

1. Load command "write flash" (see Table 20-16 on page 134).
2. Load flash page buffer.
3. Load flash high address and program page. Wait after instr. 3 until SDO goes high for the "page programming" cycle to finish.
4. Repeat 2 through 3 until the entire flash is programmed or until all data has been programmed.
5. End page programming by loading command "no operation".

When writing or reading serial data to the Atmel® ATtiny25/45/85, data is clocked on the rising edge of the serial clock, see Figure 20-6 on page 133, Figure 20-7 on page 136 and Figure 20-17 on page 136 for details.

Figure 20-5. Addressing the Flash which is Organized in Pages

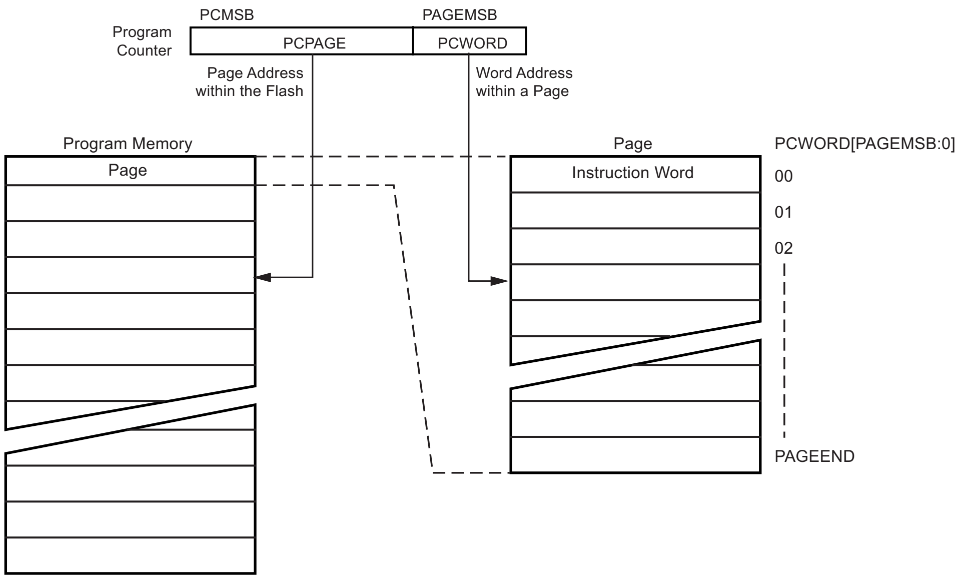
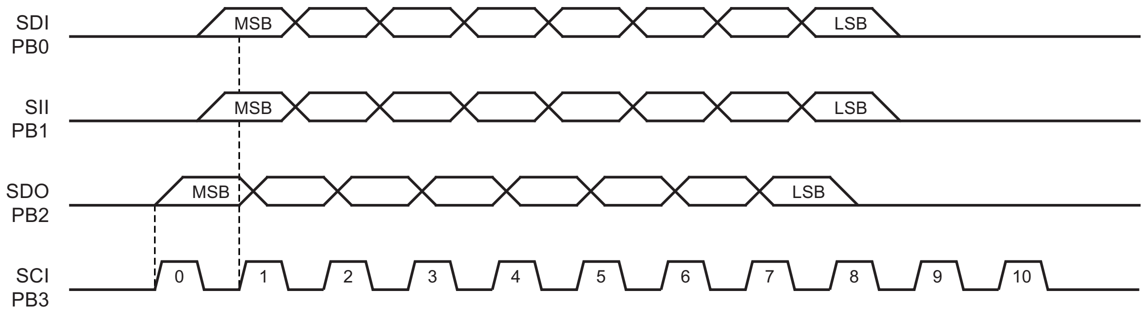


Figure 20-6. High-voltage Serial Programming Waveforms



20.8.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 20-11 on page 130. When programming the EEPROM, the data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to Table 20-16 on page 134):

1. Load command "write EEPROM".
2. Load EEPROM page buffer.
3. Program EEPROM page. Wait after instr. 2 until SDO goes high for the "page programming" cycle to finish.
4. Repeat 2 through 3 until the entire EEPROM is programmed or until all data has been programmed.
5. End page programming by loading command "no operation".

20.8.6 Reading the Flash

The algorithm for reading the flash memory is as follows (refer to Table 20-16 on page 134):

1. Load command "read flash".
2. Read flash low and high bytes. The contents at the selected address are available at serial output SDO.

20.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Table 20-16 on page 134):

1. Load command "read EEPROM".
2. Read EEPROM byte. The contents at the selected address are available at serial output SDO.

20.8.8 Programming and Reading the Fuse and Lock Bits

The algorithms for programming and reading the fuse low/high bits and lock bits are shown in Table 20-16 on page 134.

20.8.9 Reading the Signature Bytes and Calibration Byte

The algorithms for reading the signature bytes and calibration byte are shown in Table 20-16 on page 134.

20.8.10 Power-off sequence

Set SCI to "0". Set RESET to "1". Turn V_{CC} power off.

Table 20-16. High-voltage Serial Programming Instruction Set for ATtiny25/45/85

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Chip erase	SDI SII SDO	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		Wait after Instr.3 until SDO goes high for the chip erase cycle to finish.
Load “write flash” command	SDI SII SDO	0_0001_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter flash programming code.
Load flash page buffer	SDI SII SDO	0_ **bbbb_bbbb** _00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_**dddd_dddd**_00 0_0011_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1101_00 x_xxxx_xxxx_xx	Repeat after Instr. 1 - 5 until the entire page buffer is filled or until all data within the page is filled. See note 1.
	SDI SII SDO	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx				Instr 5.
Load flash high address and program page	SDI SII SDO	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		Wait after Instr 3 until SDO goes high. Repeat Instr. 2 - 3 for each loaded flash page until the entire flash or all data is programmed. Repeat Instr. 1 for a new 256 byte page. See note 1.
Load “read flash” command	SDI SII SDO	0_0000_0010_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter flash read mode.
Read flash low and high bytes	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	Repeat Instr. 1, 3 - 6 for each new address. Repeat Instr. 2 for a new 256 byte page.
	SDI SII SDO	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx			Instr 5 - 6.
Load “write EEPROM” command	SDI SII SDO	0_0001_0001_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter EEPROM programming mode.
Load EEPROM page buffer	SDI SII SDO	0_00**bb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Repeat Instr. 1 - 4 until the entire page buffer is filled or until all data within the page is filled. See note 2.
Program EEPROM page	SDI SII SDO	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx			Wait after Instr. 2 until SDO goes high. Repeat Instr. 1 - 2 for each loaded EEPROM page until the entire EEPROM or all data is programmed.

Note: **a** = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits,
x = don’t care, **1** = lock Bit1, **2** = lock bit2, **3** = CKSEL0 fuse, **4** = CKSEL1 fuse, **5** = SUT0 fuse, **6** = SUT1 fuse, **7** = CKDIV8, fuse, **8** = WDTON fuse, **9** = EESAVE fuse, **A** = SPIEN fuse, **B** = RSTDISBL fuse, **C** = BODLEVEL0 fuse, **D**= BODLEVEL1 fuse, **E** = MONEN fuse, **F** = SPMEN fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.
3. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in high-voltage serial programming, only in SPI programming.

Table 20-16. High-voltage Serial Programming Instruction Set for ATtiny25/45/85 (Continued)

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Write EEPROM byte	SDI SII SDO	0_00**bb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	Repeat Instr. 1 - 5 for each new address. Wait after Instr. 5 until SDO goes high. See Note 3.
	SDI SII SDO	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx				Instr. 5
Load "read EEPROM" command	SDI SII SDO	0_0000_0011_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter EEPROM read mode.
Read EEPROM byte	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**aaaa_aaaa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**0_00	Repeat Instr. 1, 3 - 4 for each new address. Repeat Instr. 2 for a new 256 byte page.
Write fuse low bits	SDI SII SDO	0_0100_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**A987_6543**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **A - 3** = "0" to program the fuse bit.
Write fuse high bits	SDI SII SDO	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_000**F_EDCB**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **F - B** = "0" to program the fuse bit.
Write lock bits	SDI SII SDO	0_0010_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**21**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait after Instr. 4 until SDO goes high. Write **2 - 1** = "0" to program the lock bit.
Read fuse low bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **A_9876_543**x_xx		Reading **A - 3** = "0" means the fuse bit is programmed.
Read fuse high bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1010_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1110_00 x_xx**FE_DCB**x_xx		Reading **F - B** = "0" means the fuse bit is programmed.
Read lock bits	SDI SII SDO	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 x_xxxx_x**21**x_xx		Reading **2, 1** = "0" means the lock bit is programmed.
Read signature bytes	SDI SII SDO	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**bb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	Repeats Instr 2 4 for each signature byte address.
Read calibration byte	SDI SII SDO	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx	
Load "no operation" command	SDI SII SDO	0_0000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				

Note: **a** = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits,
x = don't care, **1** = lock Bit1, **2** = lock bit2, **3** = CKSEL0 fuse, **4** = CKSEL1 fuse, **5** = SUT0 fuse, **6** = SUT1 fuse, **7** = CKDIV8, fuse, **8** = WDTON fuse, **9** = EESAVE fuse, **A** = SPIEN fuse, **B** = RSTDISBL fuse, **C** = BODLEVEL0 fuse, **D**= BODLEVEL1 fuse, **E** = MONEN fuse, **F** = SPMEN fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.
3. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in high-voltage serial programming, only in SPI programming.

20.9 High-voltage Serial Programming Characteristics

Figure 20-7. High-voltage Serial Programming Timing

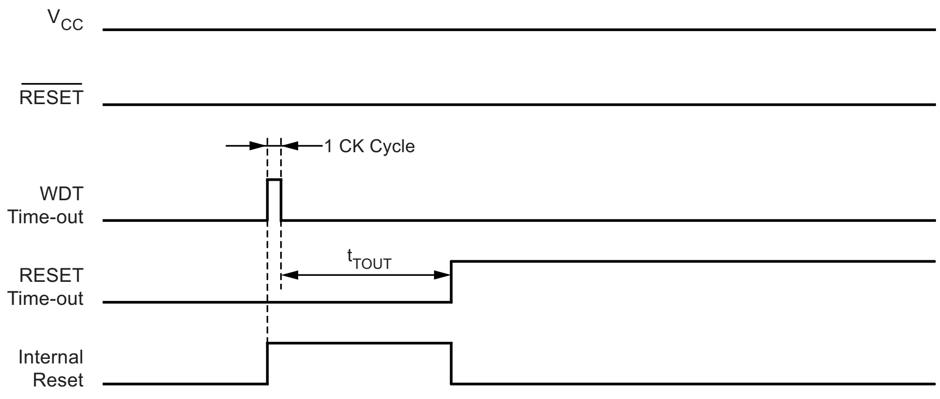


Table 20-17. High-voltage Serial Programming Characteristics T_A = 25°C ±10%, V_{CC} = 5.0V ±10% (Unless otherwise noted)

Parameter	Symbol	Min	Typ	Max	Units
SCI (PB3) pulse width high	t_{SHSL}	110			ns
SCI (PB3) pulse width low	t_{SLSH}	110			ns
SDI (PB0), SII (PB1) valid to SCI (PB3) high	t_{IVSH}	50			ns
SDI (PB0), SII (PB1) hold after SCI (PB3) high	t_{SHIX}	50			ns
SCI (PB3) high to SDO (PB2) valid	t_{SHOV}		16		ns
Wait after Instr. 3 for write fuse bits	t_{WLWH_PFB}		2.5		ms

21. Electrical Characteristics

21.1 Absolute Maximum Ratings

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Min.	Typ.	Max.	Unit
Operating temperature	– 40		+ 125	°C
Storage temperature	– 65		+ 150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	– 0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to ground	– 0.5		+ 13.0	V
Maximum operating voltage		6.0		V
DC current per I/O pin		40.0		mA
DC current V_{CC} and GND pins		200.0		mA

Table 21-1. DC Characteristics T_A = – 40°C to 125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)(1)

Parameter	Condition	Symbol	Min.(2)	Typ.	Max.(3)	Units
Input low voltage	Except $\overline{\text{RESET}}$ and XTAL pins	V_{IL}	– 0.5		$0.3V_{CC}$	V
Input low voltage	XTAL pin	V_{IL1}	– 0.5		$0.1V_{CC}$	V
Input low voltage	$\overline{\text{RESET}}$ pin	V_{IL2}	– 0.5		$0.1V_{CC}$	V
Input high-voltage	Except $\overline{\text{RESET}}$ and XTAL pins	V_{IH}	$0.6V_{CC}$(3)		V_{CC} + 0.5	V
Input high-voltage	XTAL pin	V_{IH1}	$0.9V_{CC}$(3)		V_{CC} + 0.5	V
Input high-voltage	$\overline{\text{RESET}}$ pin	V_{IH2}	$0.9V_{CC}$(3)		V_{CC} + 0.5	V
Output low voltage(4) (port B) except PB5	I_{OL} = 8mA, V_{CC} = 5V I_{OL} = 5mA, V_{CC} = 3V	V_{OL}			0.6 0.5	V V
Output high-voltage(5) (port B) except PB5	I_{OH} = –8mA, V_{CC} = 5V I_{OH} = –5mA, V_{CC} = 3V	V_{OH}	4.1 2.3			V V
Output low voltage(4) PB5	I_{OL} = 1mA	V_{OL1}			0.6	V
Output High-voltage(5) PB5	I_{OH} = –200µA, V_{CC} = 5V	V_{OH1}	3.2			V

Notes:
1. All DC characteristics contained in this data sheet result from actual silicon characterization.
2. "Max" means the highest value where the pin is guaranteed to be read as low.
3. "Min" means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (8mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (8mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
6. All I/O modules are turned off (PRR = 0xFF) for all I_{CC} values.
7. Brown-out detection (BOD) disabled.

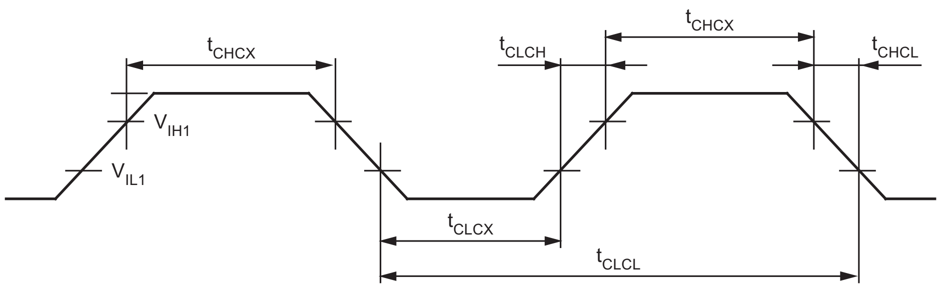
Table 21-1. DC Characteristics T_A = – 40°C to 125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)[1] (Continued)

Parameter	Condition	Symbol	Min.[2]	Typ.	Max.[3]	Units
Input leakage Current I/O pin except $\overline{\text{RESET}}$	Vcc = 5.5V, pin low (absolute value)	I_{IL}			50	nA
Input leakage current I/O pin except $\overline{\text{RESET}}$	Vcc = 5.5V, pin high (absolute value)	I_{IH}			50	nA
Reset pull-up resistor		R_{RST}	30		60	kΩ
I/O pin pull-up resistor		R_{pu}	20		50	kΩ
Power supply current	Active 4MHz, V_{CC} = 3V	I_{CC}[6]		1.25	3	mA
	Active 8MHz, V_{CC} = 5V			5	10	mA
	Active 16MHz, V_{CC} = 5V			10	15	mA
	Idle 4MHz, V_{CC} = 3V			0.4	0.5	mA
	Idle 8MHz, V_{CC} = 5V			1.2	2	mA
	Idle 16MHz, V_{CC} = 5V			2.5	5	mA
Power-down mode[7]	WDT enabled, V_{CC} = 3V			5	30	µA
	WDT disabled, V_{CC} = 3V			2	24	µA
	WDT enabled, V_{CC} = 5V			9	50	µA
	WDT disabled, V_{CC} = 5V			3	36	µA

Notes:
1. All DC characteristics contained in this data sheet result from actual silicon characterization.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. “Min” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (8mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (8mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
6. All I/O modules are turned off (PRR = 0xFF) for all I_{CC} values.
7. Brown-out detection (BOD) disabled.

21.2 External Clock Drive Waveforms

Figure 21-1. External Clock Drive Waveforms



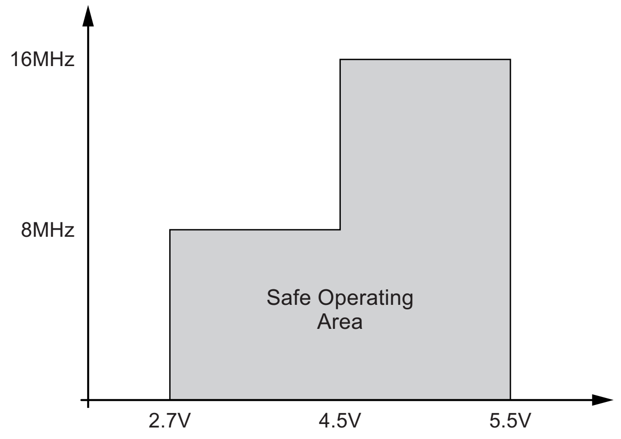
21.3 External Clock Drive

Table 21-2. External Clock Drive[1]

		V_{CC} = 2.7 to 5.5V		V_{CC} = 4.5 to 5.5V		
Parameter	Symbol	Min.	Max.	Min.	Max.	Units
Clock frequency	$1/t_{CLCL}$	0	8	0	16	MHz
Clock period	t_{CLCL}	100		50		ns
High time	t_{CHCX}	40		20		ns
Low time	t_{CLCX}	40		20		ns
Rise time	t_{CLCH}		1.6		0.5	μs
Fall time	t_{CHCL}		1.6		0.5	μs
Change in period from one clock cycle to the next	Δt_{CLCL}		2		2	%

Note: 1. All DC Characteristics contained in this data sheet result from actual silicon characterization.

Figure 21-2. Maximum Frequency versus V_{CC}



21.4 ADC Characteristics – Preliminary Data

Table 21-3. ADC Characteristics, Single Ended Channels. – 40°C to +125°C(1)

Parameter	Condition	Symbol	Min(1)	Typ(1)	Max(1)	Units
Resolution	Single ended conversion				10	Bits
Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			2		LSB
	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1MHz			3		LSB
	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz noise reduction mode			1.5		LSB
	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1MHz noise reduction mode			2.5		LSB
Integral non-linearity (INL)	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			1		LSB
Differential non-linearity (DNL)	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			0.5		LSB
Gain error	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			2.5		LSB
Offset error	Single ended conversion V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz			1.5		LSB
Conversion time	Free running conversion		13		260	µs
Clock frequency			50		1000	kHz
Analog supply voltage		AVCC	V_{CC} – 0.3(2)		V_{CC} + 0.3(3)	V
Input voltage		V_{IN}	GND		V_{REF} – 50mV	V
Input bandwidth				38.5		kHz
Internal voltage reference		V_{INT}	1.0	1.1	1.2	V
Analog input resistance		R_{AIN}		100		MW

Notes: 1. All DC characteristics contained in this data sheet result from actual silicon characterization.
2. Minimum for AVCC is 2.7V.
3. Maximum for AVCC is 5.5V.

21.5 Calibrated RC Oscillator Accuracy

Table 21-4. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{CC}	Temperature	Calibration Accuracy
Factory calibration	8.0MHz	3V	25°C	±1%
User calibration	7.3 to 8.1MHz	2.7V to 5.5V	–40°C ±125°C	±14%

22. Typical Characteristics

The data contained in this section is extracted from preliminary silicon characterization and will be updated upon final characterization.

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail to rail output is used as clock source.

The power consumption in power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as
CL $\times$ VCC $\times$ f where CL = load capacitance, VCC = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in power-down mode with watchdog timer enabled and power-down mode with watchdog timer disabled represents the differential current drawn by the watchdog timer.

22.1 Active Supply Current

Figure 22-1. Active Supply Current versus Frequency (0.1 to 1.0MHz)

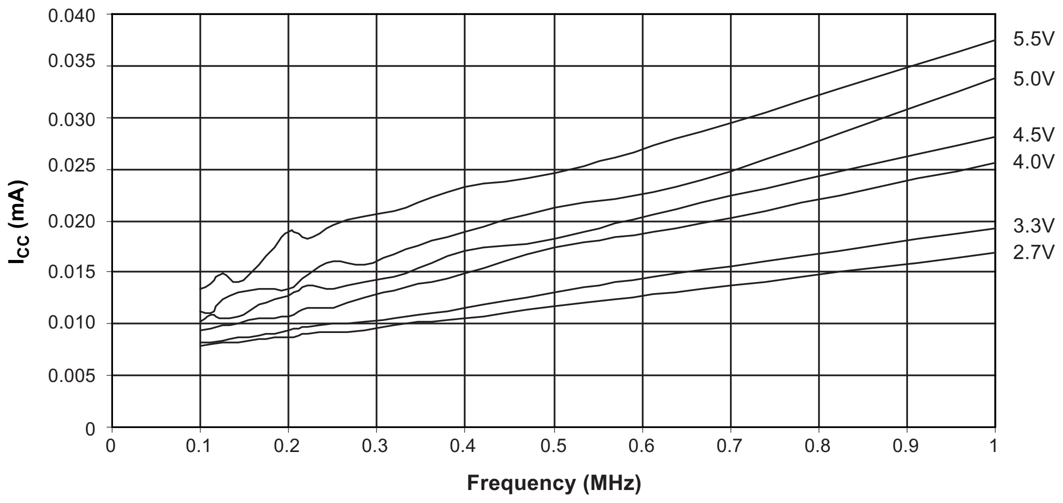


Figure 22-2. Active Supply Current versus Frequency (1 to 20MHz)

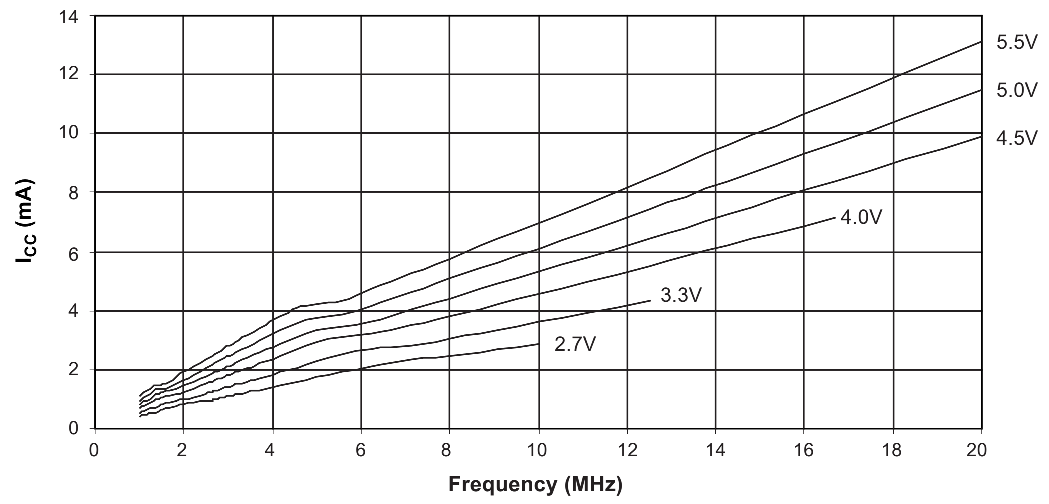


Figure 22-3. Active Supply Current versus V_{CC} (Internal RC Oscillator, 128kHz)

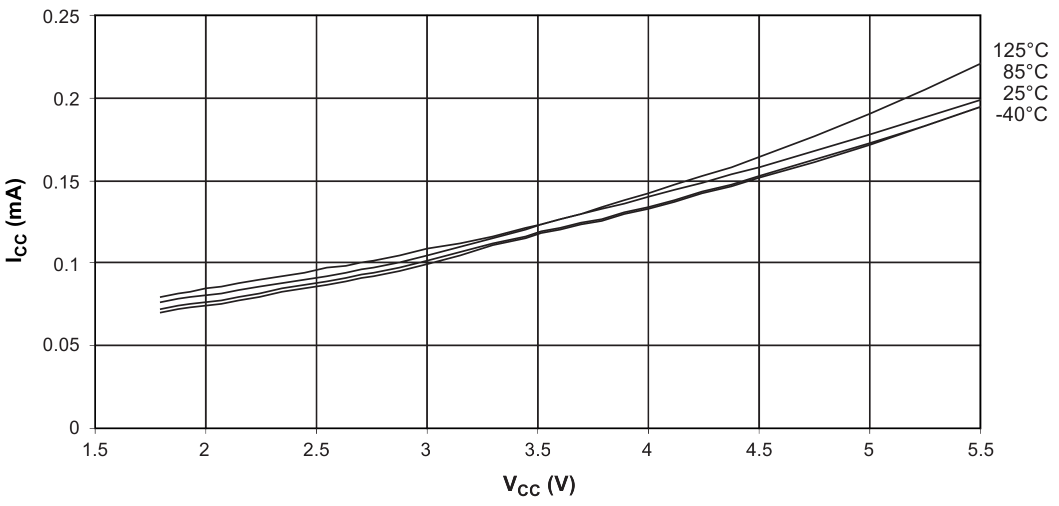


Figure 22-4. Active Supply Current versus V_{CC} (Internal RC Oscillator, 1MHz)

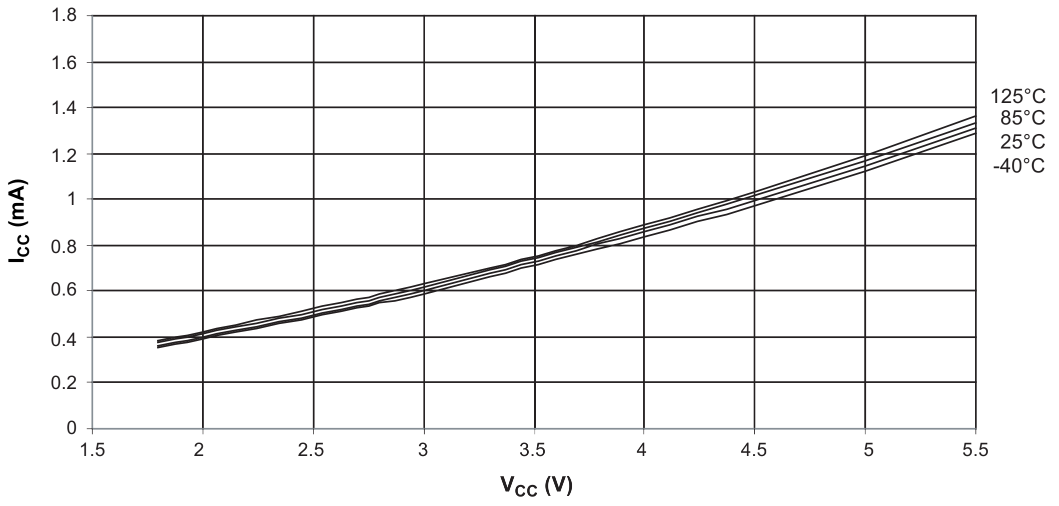
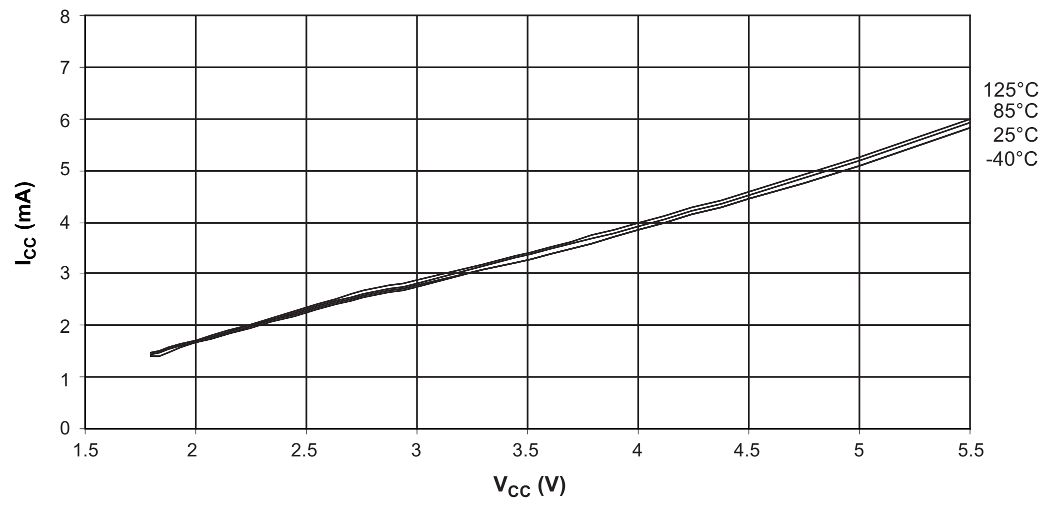


Figure 22-5. Active Supply Current versus V_{CC} (Internal RC Oscillator, 8MHz)



22.2 Idle Supply Current

Figure 22-6. Idle Supply Current versus Frequency (0.1 to 1.0MHz)

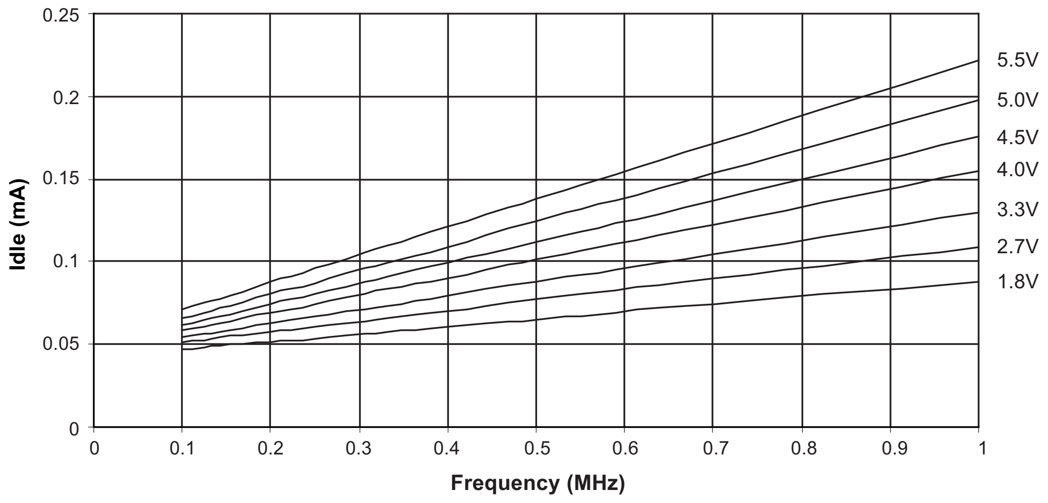


Figure 22-7. Idle Supply Current versus Frequency (1 to 20MHz)

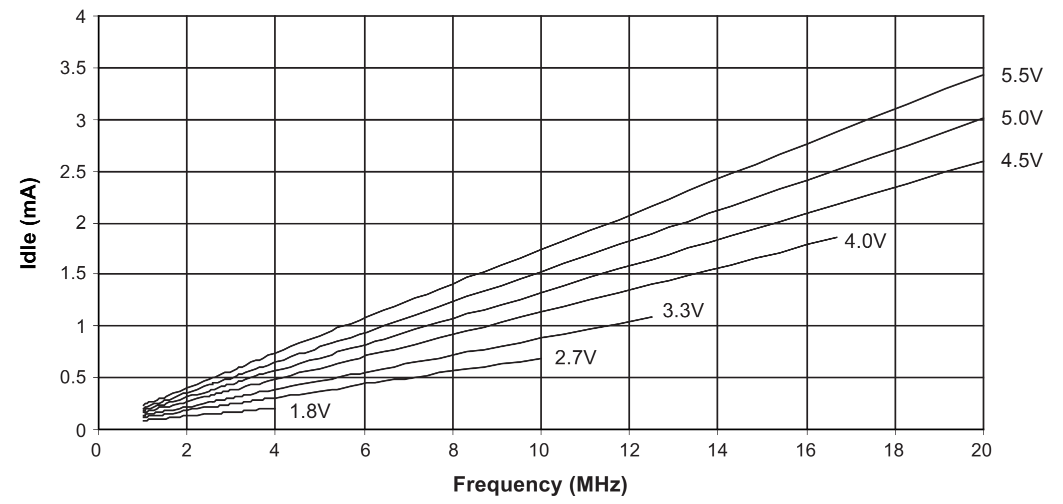


Figure 22-8. Idle Supply Current versus V_{CC} (Internal RC Oscillator, 128kHz)

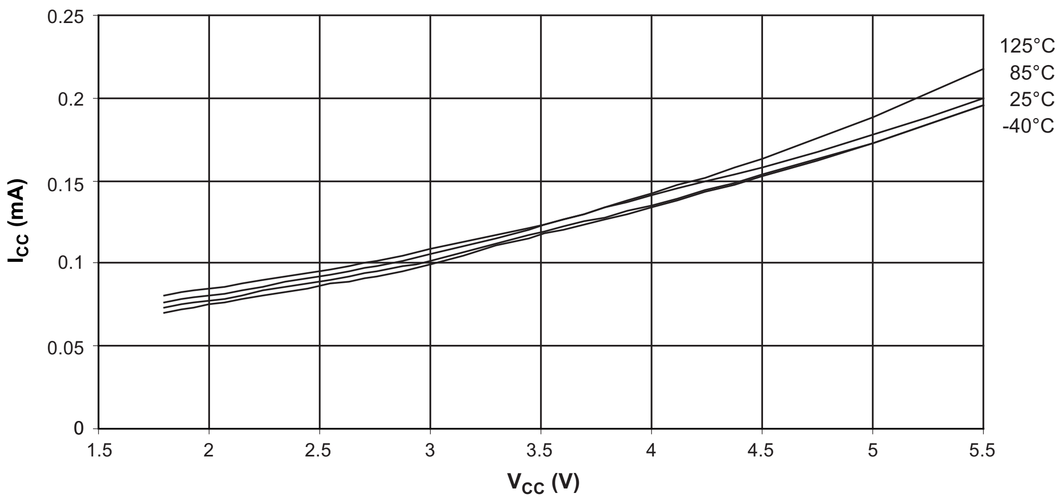


Figure 22-9. Idle Supply Current versus V_{CC} (Internal RC Oscillator, 1MHz)

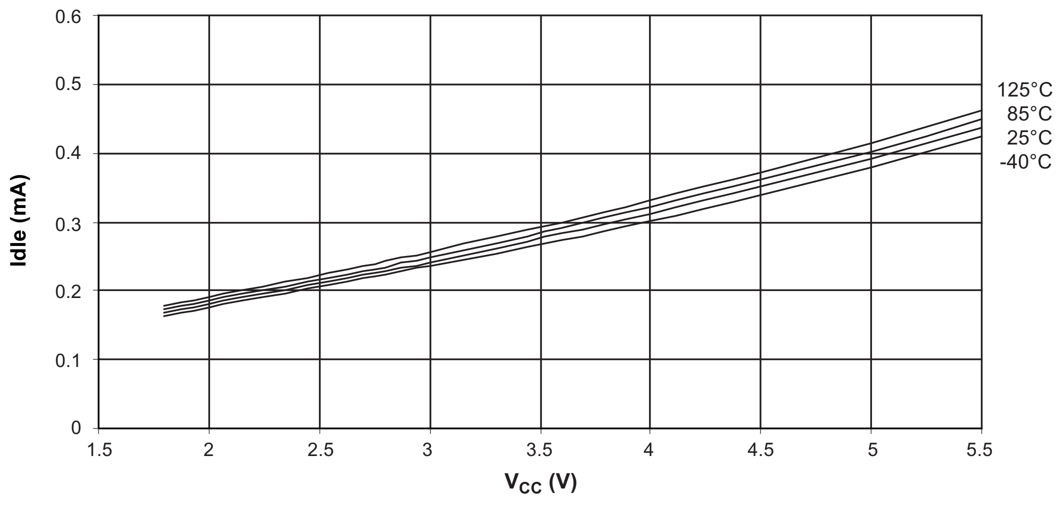
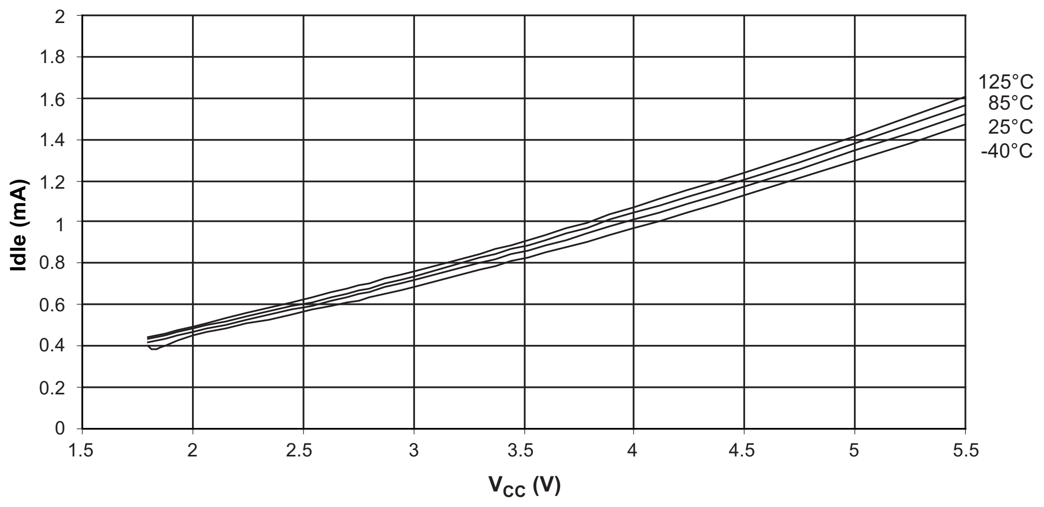


Figure 22-10. Idle Supply Current versus V_{CC} (Internal RC Oscillator, 8MHz)



22.2.1 Using the Power Reduction Register

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in active and idle mode. The enabling or disabling of the I/O modules are controlled by the power reduction register. See Section 6.6 "Power Reduction Register" on page 30 for details.

Table 22-1. Additional Current Consumption for the different I/O modules (absolute values)

PRR bit	Typical numbers		
	V_{CC} = 2V, F = 1MHz	V_{CC} = 3V, F = 4MHz	V_{CC} = 5V, F = 8MHz
PRTIM1	43µA	270µA	1090µA
PRTIM0	5.0µA	28µA	116µA
PRUSI	4.0µA	25µA	102µA
PRADC	13µA	84µA	351µA

Table 22-2. Additional Current Consumption (percentage) in Active and Idle mode

PRR bit	Additional Current consumption compared to Active with external clock (see Figure 22-1 and Figure 22-2)	Additional Current consumption compared to Idle with external clock (see Figure 22-6 and Figure 22-7)
PRTIM1	17.3%	68.4%
PRTIM0	1.8%	7.3%
PRUSI	1.6%	6.4%
PRADC	5.4%	21.4%

It is possible to calculate the typical current consumption based on the numbers from Table 22-2 for other V_{CC} and frequency settings than listed in Table 22-1.

22.2.1.1Example 1

Calculate the expected current consumption in idle mode with USI, TIMER0, and ADC enabled at V_{CC} = 2.0V and F = 1MHz. From Table 22-2 on page 145, third column, we see that we need to add 6.4% for the USI, 7.3% for the TIMER0 module, and 21.4% for the ADC module. Reading from Figure 22-9, we find that the idle current consumption is ~0.25mA at V_{CC} = 3.0V and F = 1MHz. The total current consumption in idle mode with USI, TIMER0, and ADC enabled, gives:

$I_{CCtotal} \approx (0.25)\ \text{mA} \times (1 + 0.064 + 0.073 + 0.214) \approx 0.337\text{mA}$

22.3 Power-Down Supply Current

Figure 22-11. Power-Down Supply Current versus V_{CC} (Watchdog Timer Disabled)

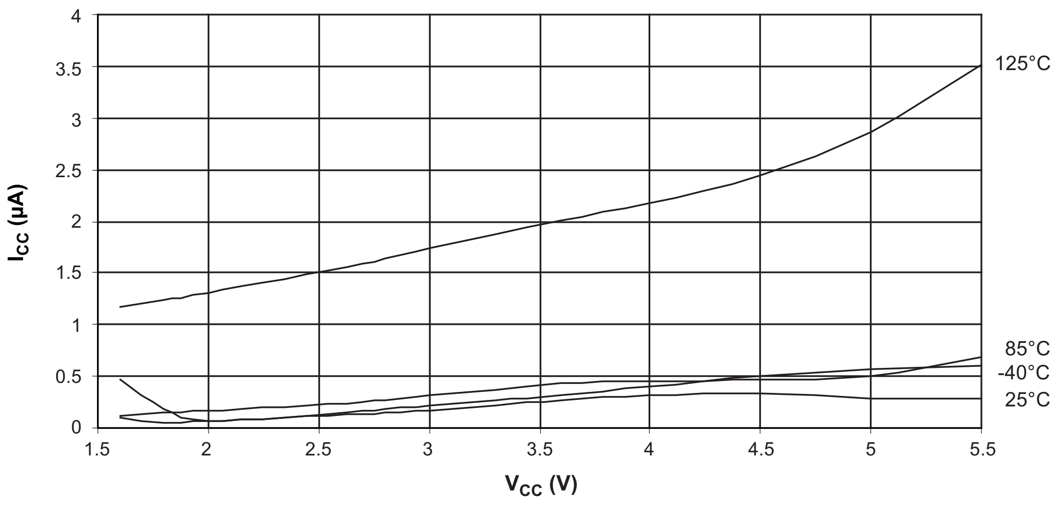
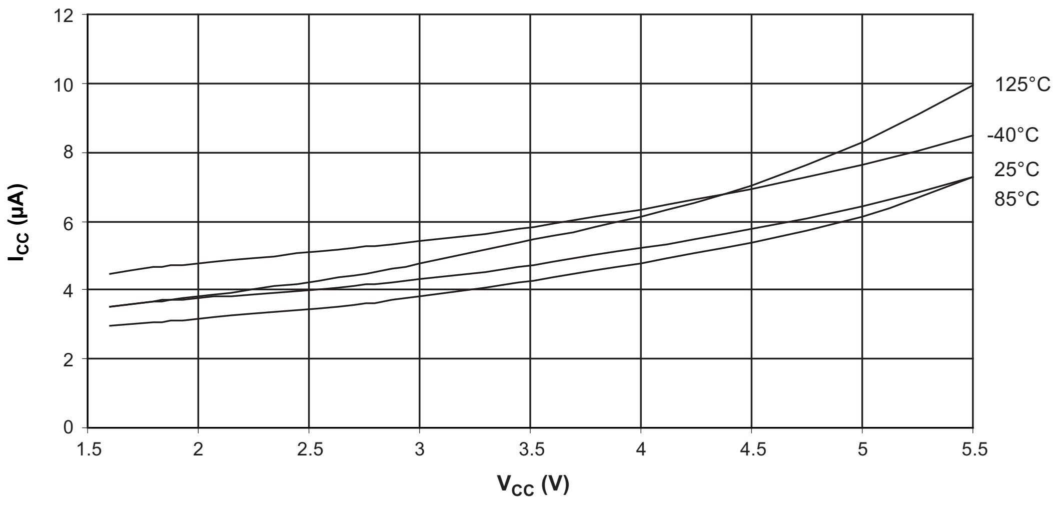


Figure 22-12. Power-Down Supply Current versus V_{CC} (Watchdog Timer Enabled)



22.4 Pin Pull-up

Figure 22-13. I/O Pin Pull-Up Resistor Current versus Input Voltage (V_{CC} = 1.8V)

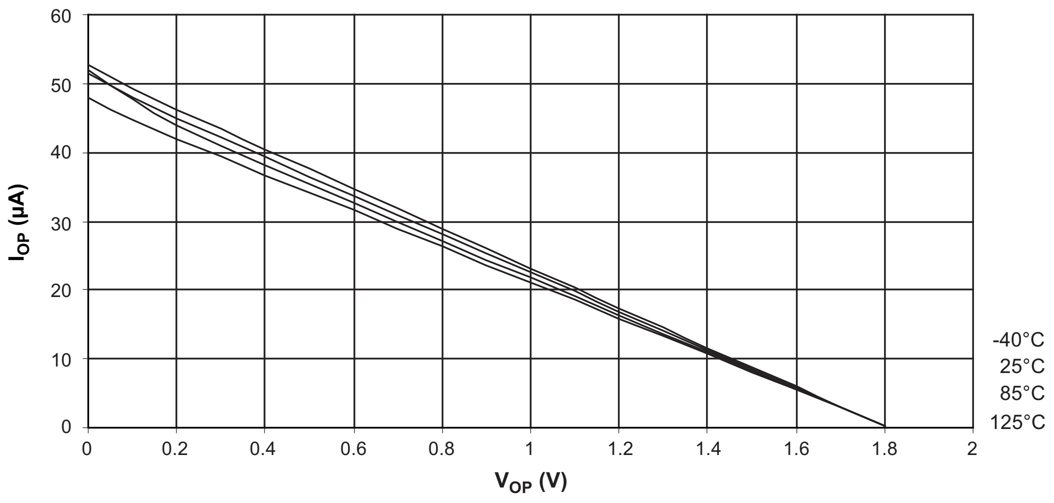


Figure 22-14. I/O Pin Pull-Up Resistor Current versus Input Voltage (V_{CC} = 2.7V)

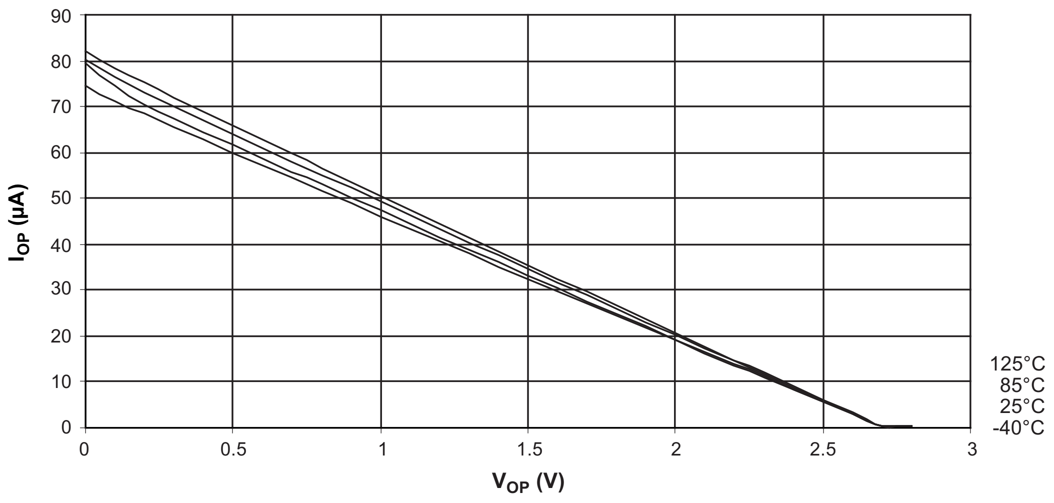


Figure 22-15. I/O Pin Pull-Up Resistor Current versus Input Voltage (V_{CC} = 5.0V)

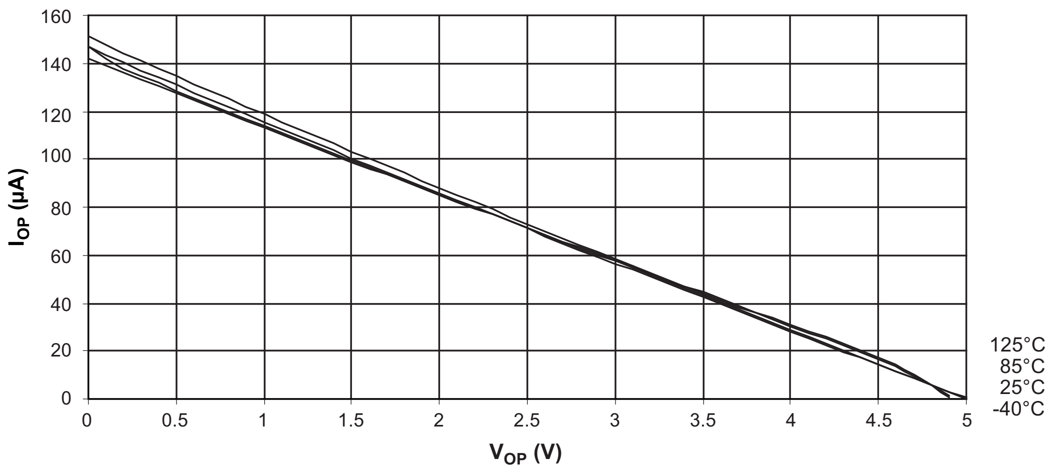


Figure 22-16. Reset Pull-Up Resistor Current versus Reset Pin Voltage (V_{CC} = 1.8V)

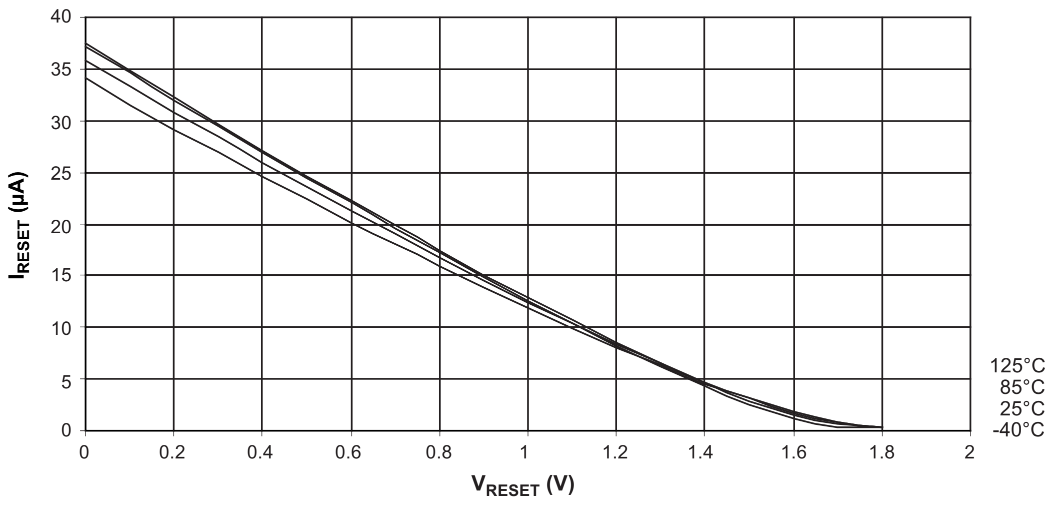


Figure 22-17. Reset Pull-Up Resistor Current versus Reset Pin Voltage (V_{CC} = 2.7V)

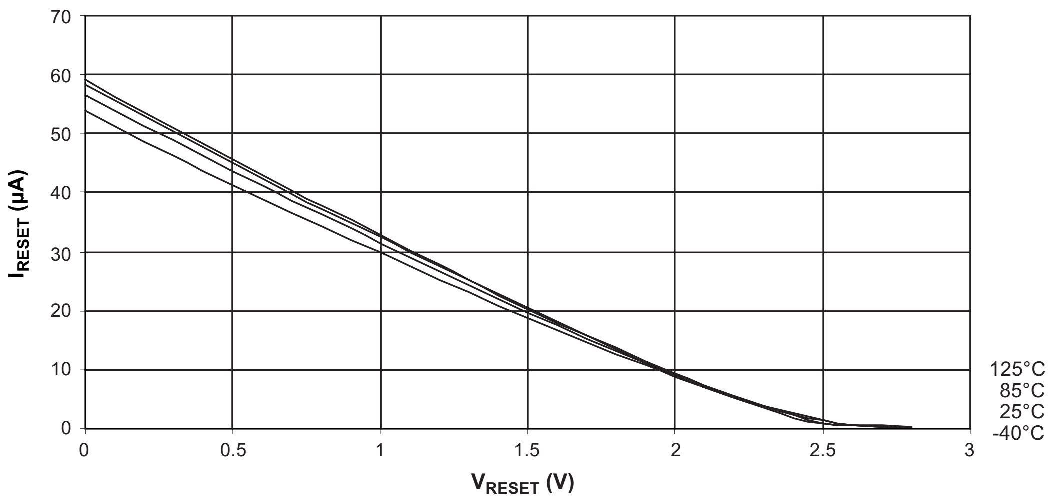
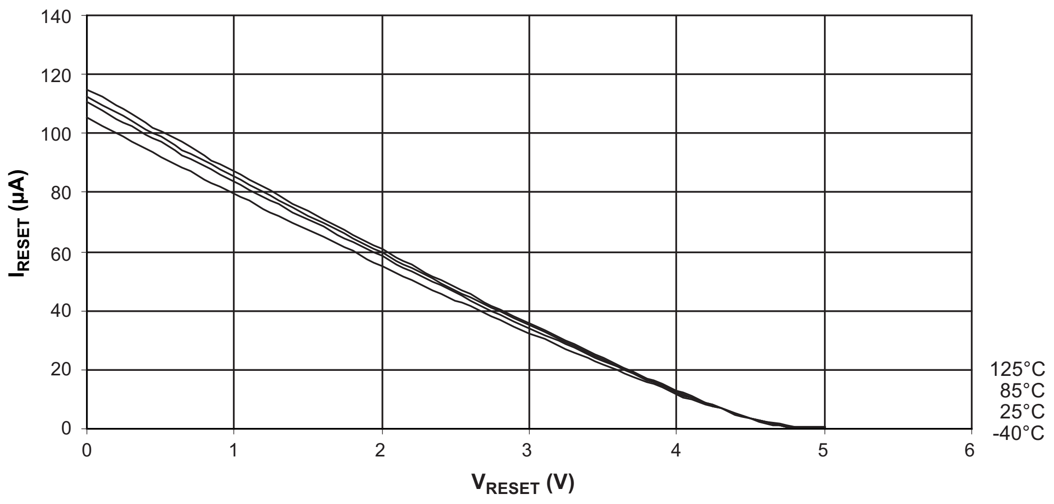


Figure 22-18. Reset Pull-Up Resistor Current versus Reset Pin Voltage (V_{CC} = 5.0V)



22.5 Pin Driver Strength

Figure 22-19. I/O Pin Source Current versus Output Voltage (V_{CC} = 1.8V)

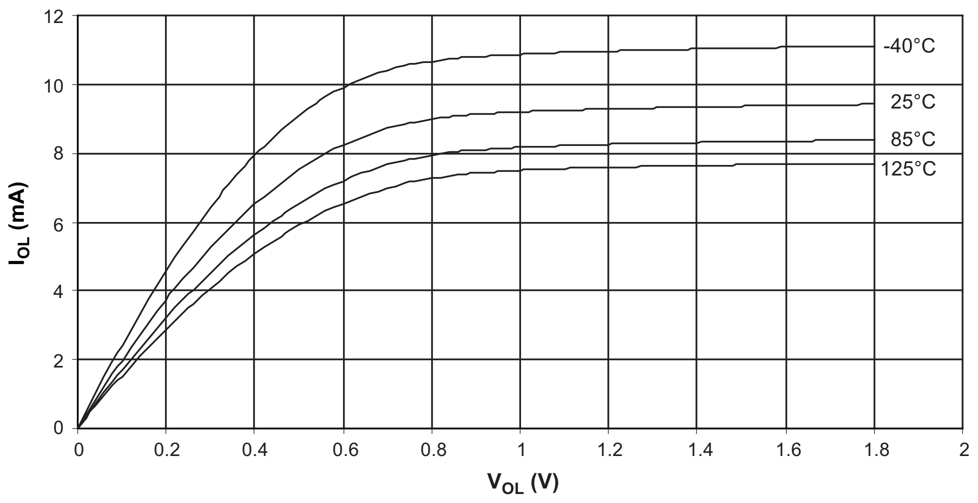


Figure 22-20. I/O Pin Source Current versus Output Voltage (V_{CC} = 3V)

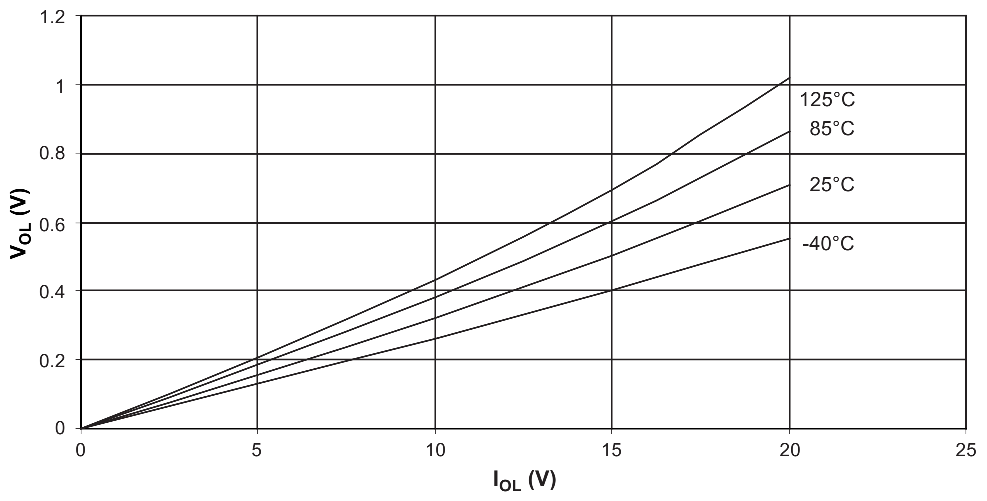


Figure 22-21. I/O Pin Source Current versus Output Voltage (V_{CC} = 5V)

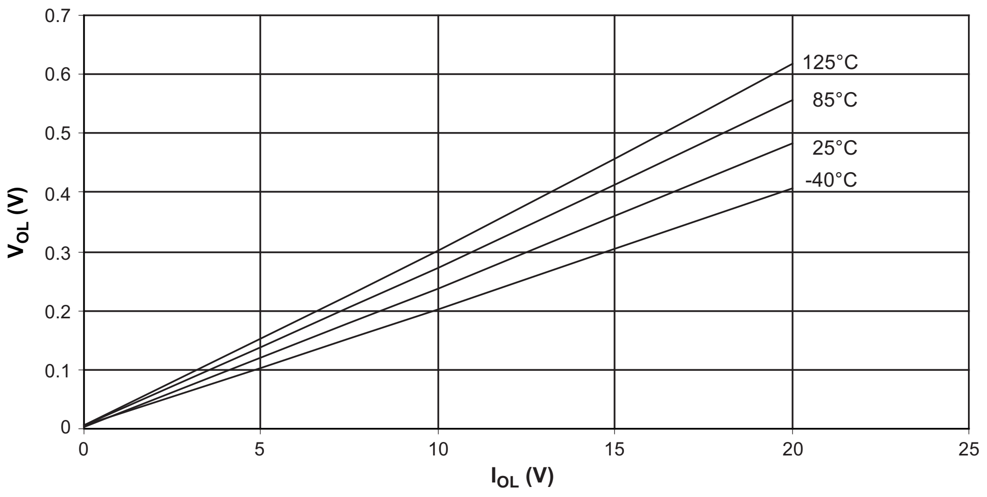


Figure 22-22. I/O Pin Sink Current versus Output Voltage (V_{CC} = 1.8V)

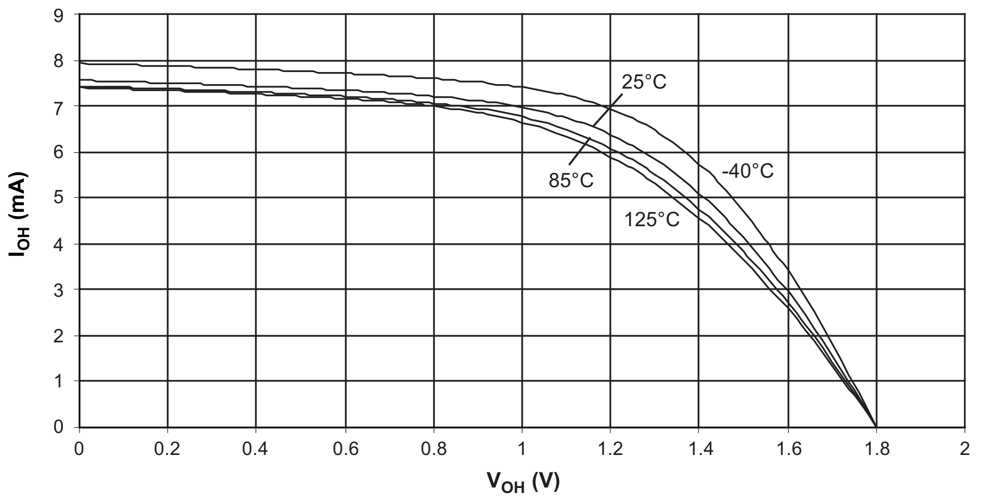


Figure 22-23. I/O Pin Sink Current versus Output Voltage (V_{CC} = 3V)

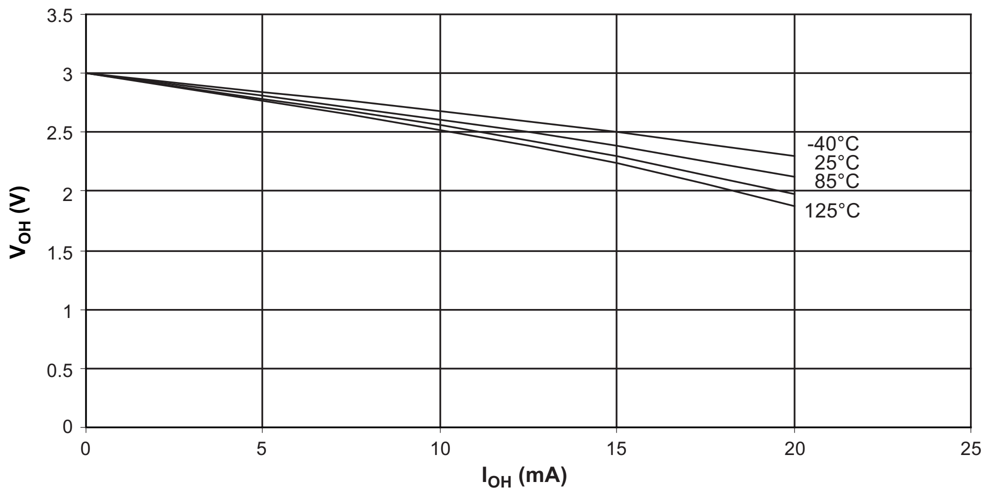
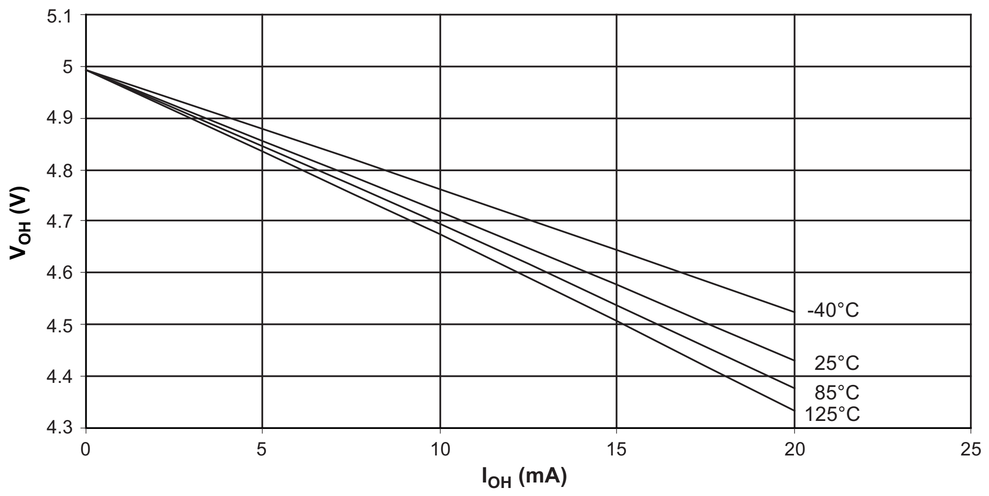


Figure 22-24. I/O Pin Sink Current versus Output Voltage (V_{CC} = 5.0V)



22.6 Pin Thresholds and Hysteresis

Figure 22-25. I/O Pin Input Threshold Voltage versus V_{CC} (VIH, I/O Pin Read As '1')

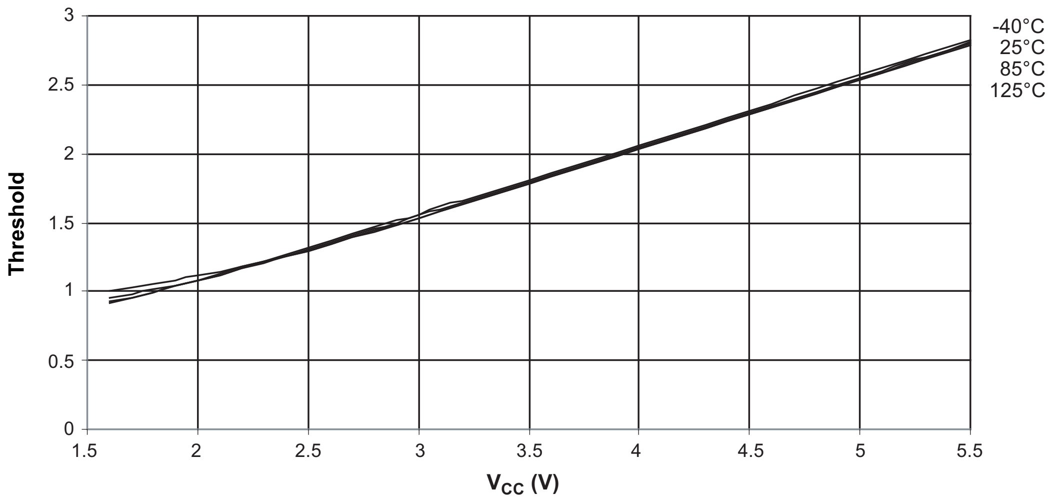


Figure 22-26. I/O Pin Input Threshold Voltage versus V_{CC} (VIL, I/O Pin Read As '0')

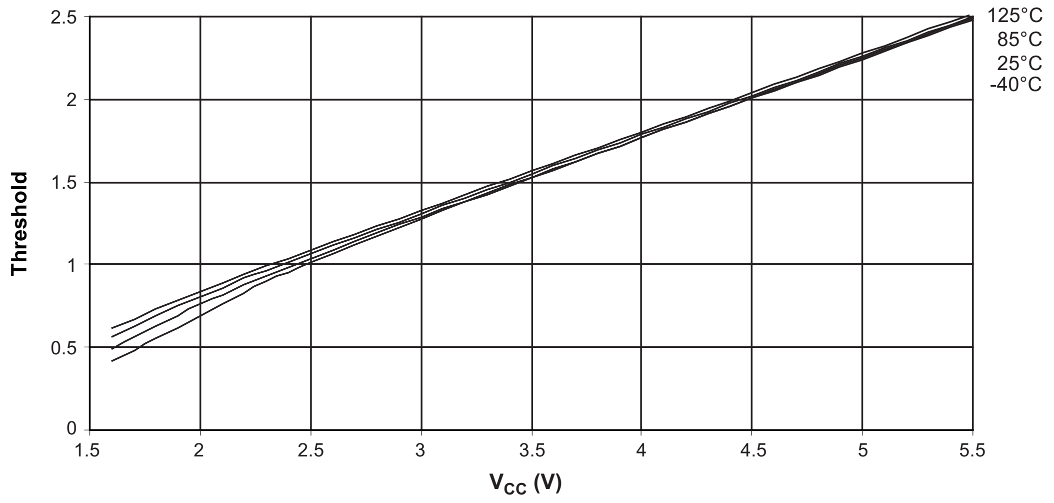


Figure 22-27. I/O Pin Input Hysteresis versus V_{CC}

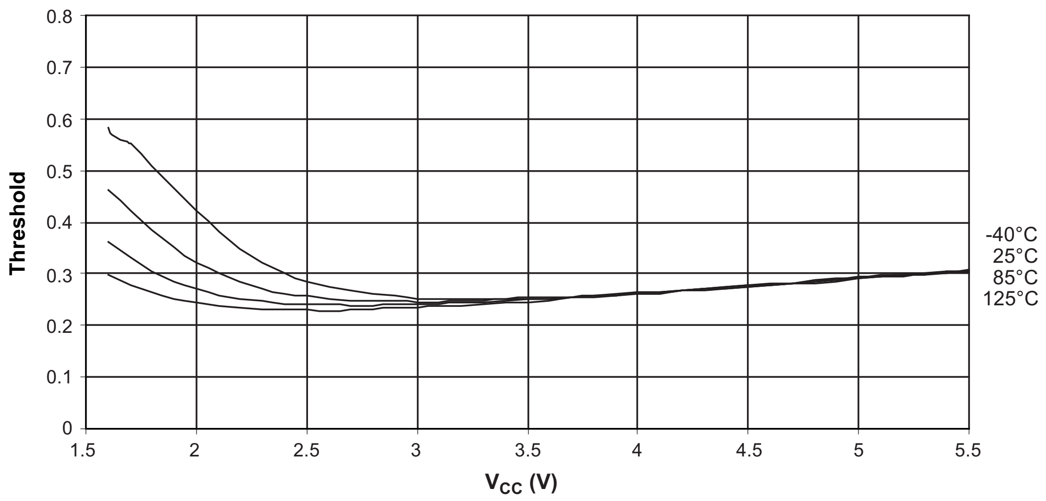


Figure 22-28. Reset Input Threshold Voltage versus V_{CC} (VIH, Reset Pin Read As '1')

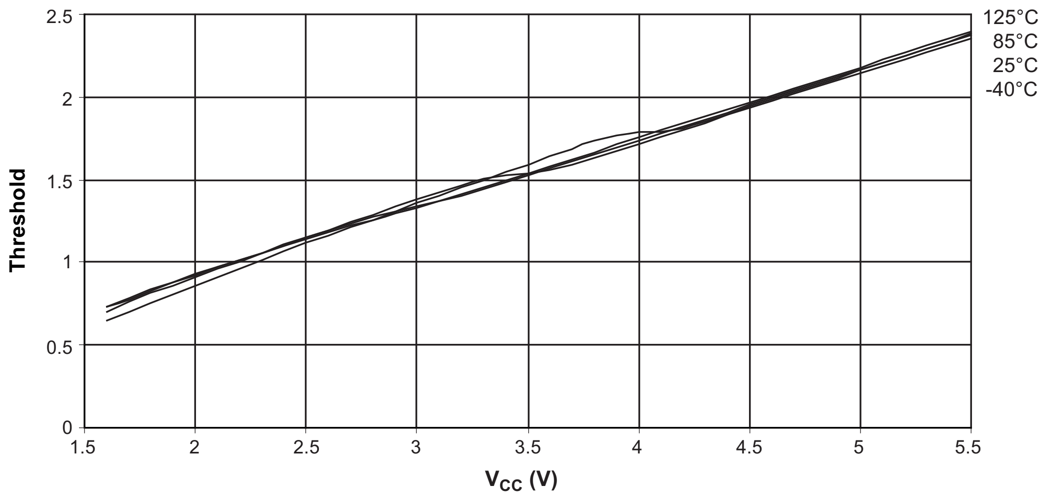


Figure 22-29. Reset Input Threshold Voltage versus V_{CC} (VIL, Reset Pin Read As '0')

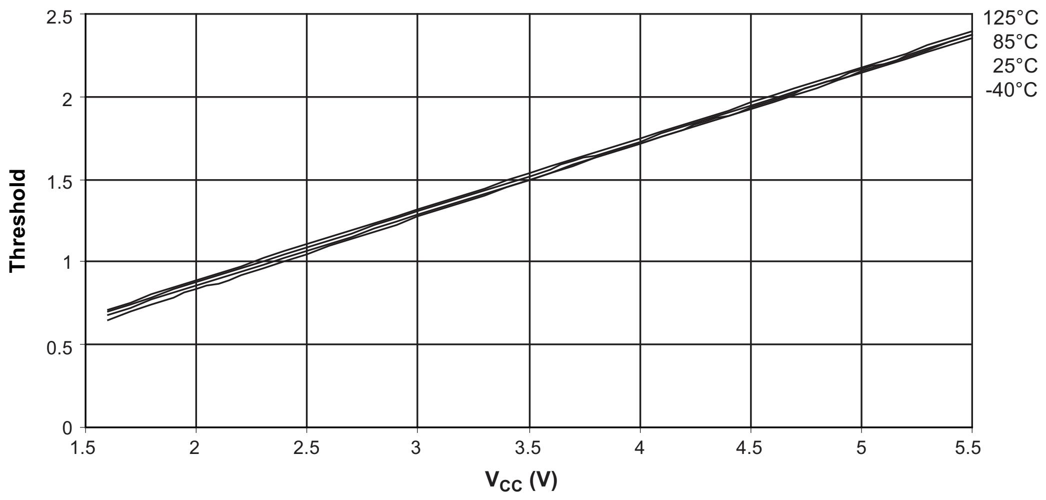
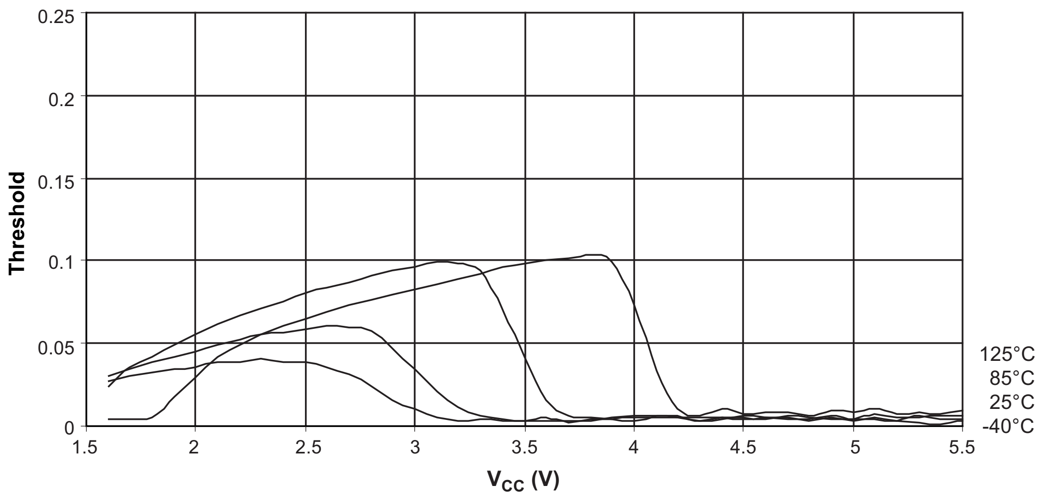


Figure 22-30. Reset Input Pin Hysteresis versus V_{CC}



22.7 BOD Thresholds and Analog Comparator Offset

Figure 22-31. BOD Thresholds versus Temperature (BODLEVEL Is 4.3V)

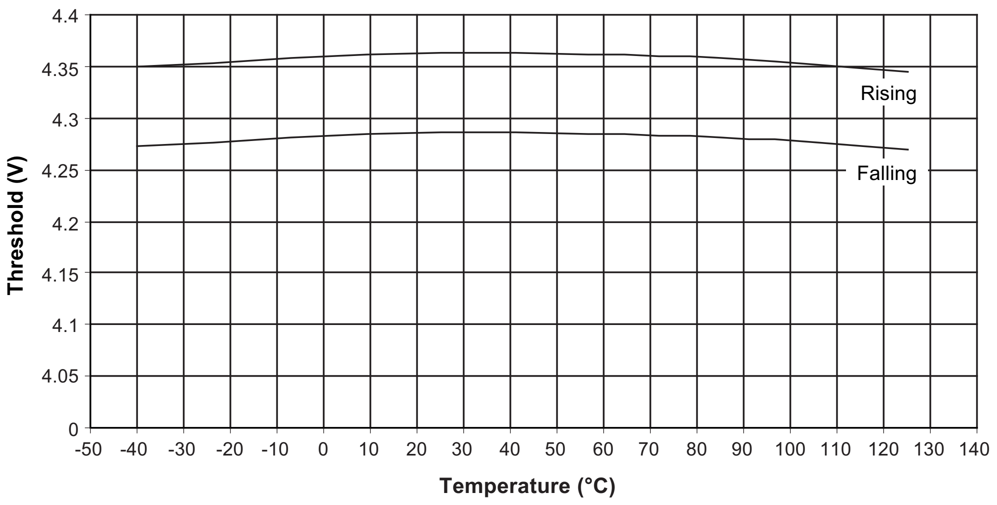


Figure 22-32. BOD Thresholds versus Temperature (BODLEVEL Is 2.7V)

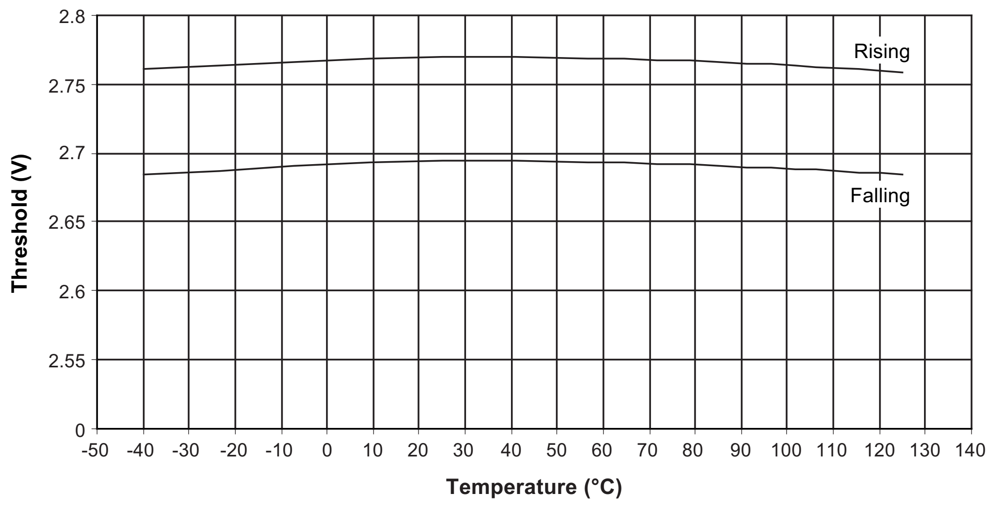
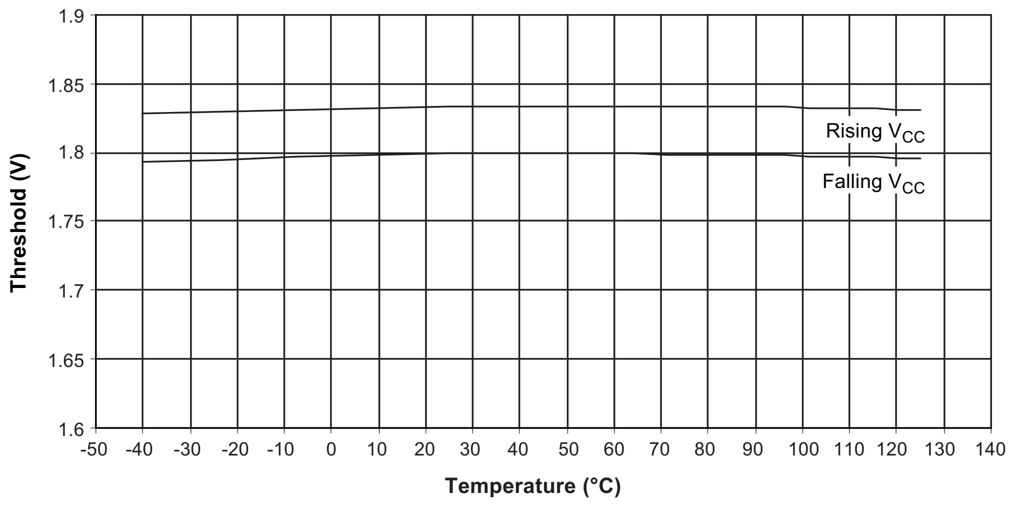


Figure 22-33. BOD Thresholds versus Temperature (BODLEVEL Is 1.8V)



22.8 Internal Oscillator Speed

Figure 22-34. Watchdog Oscillator Frequency versus V_{CC})

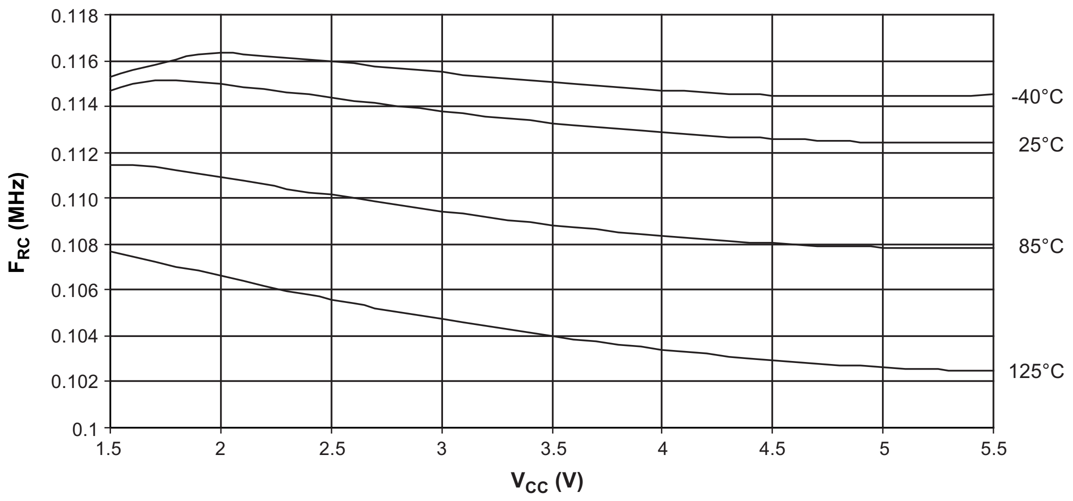


Figure 22-35. Watchdog Oscillator Frequency versus Temperature

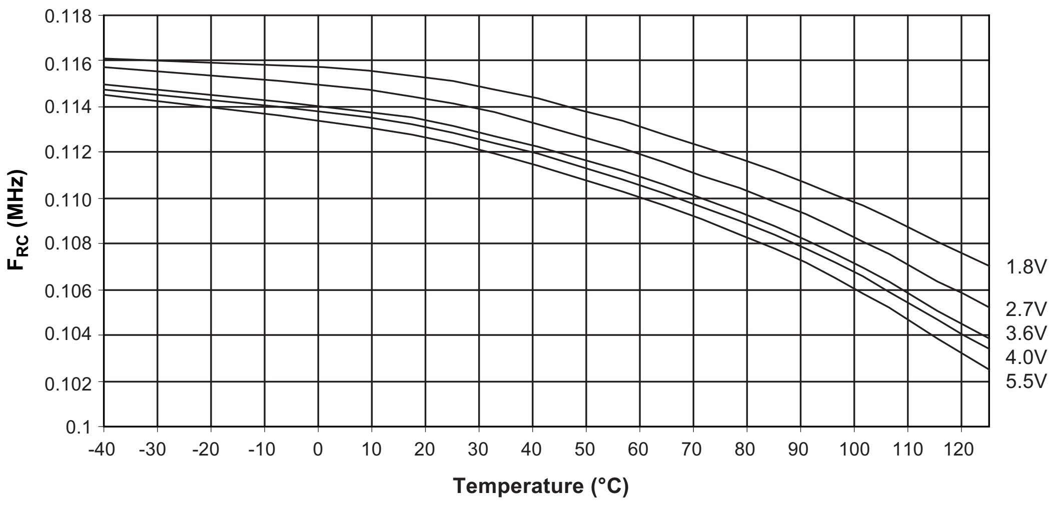


Figure 22-36. Calibrated 8MHz RC Oscillator Frequency versus Temperature

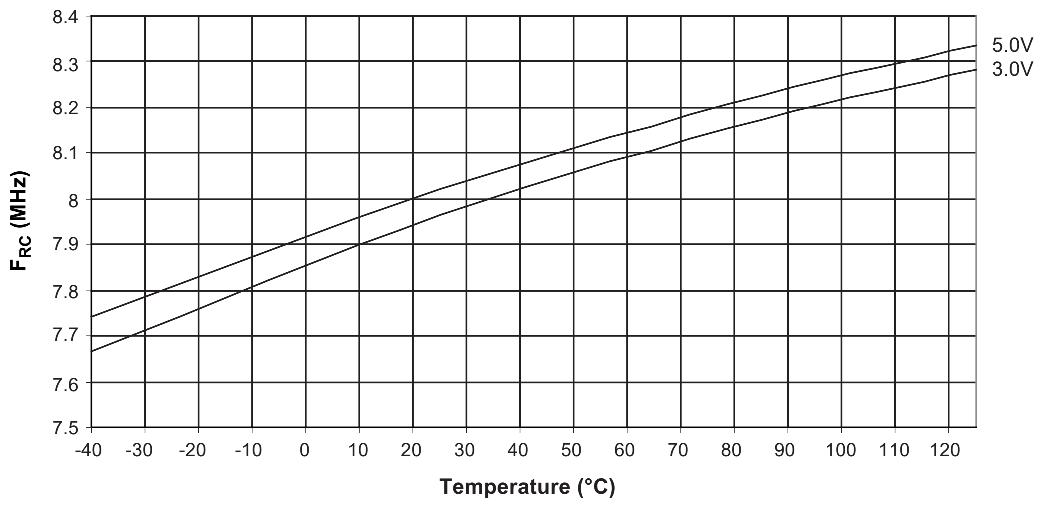


Figure 22-37. Calibrated 8MHz RC Oscillator Frequency versus V_{CC}

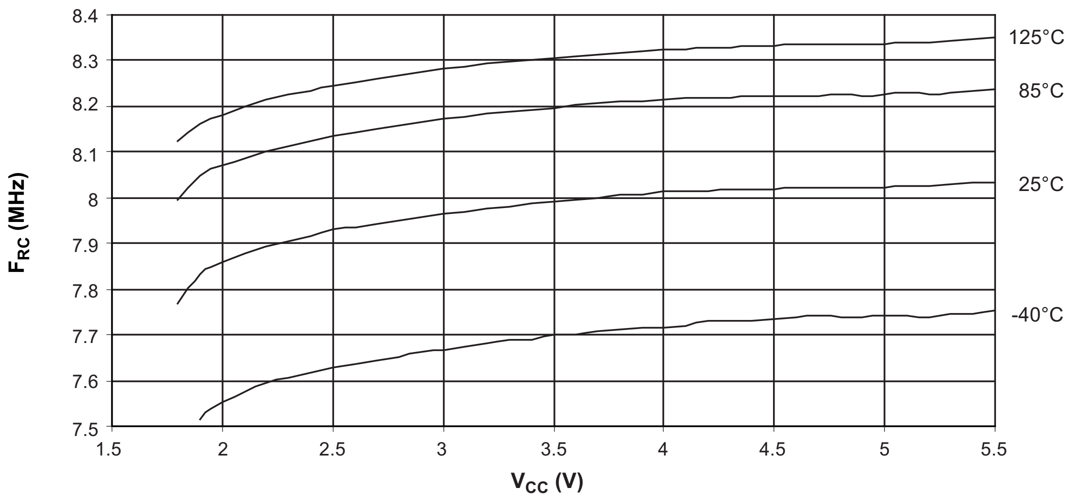
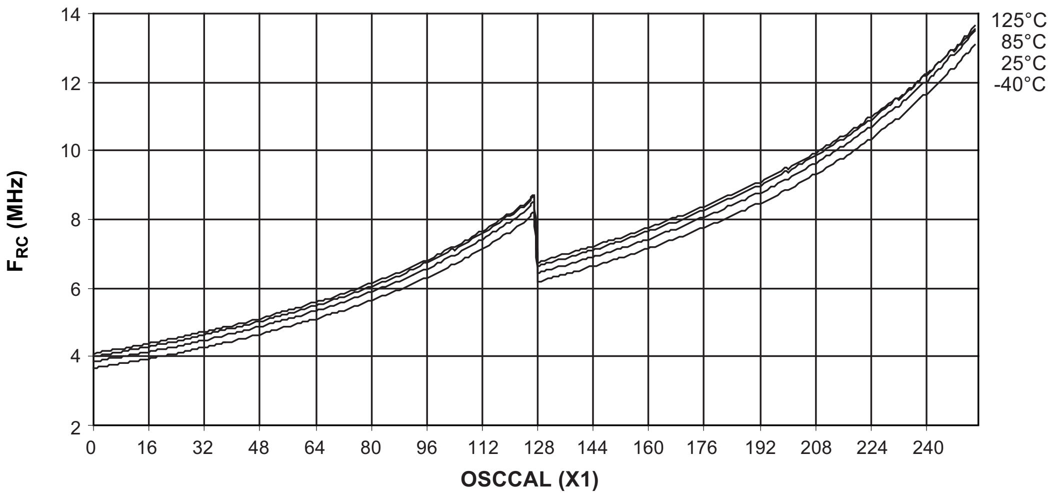


Figure 22-38. Calibrated 8MHz RC Oscillator Frequency versus OSCCAL Value



22.9 Current Consumption of Peripheral Units

Figure 22-39. Brownout Detector Current versus V_{CC}

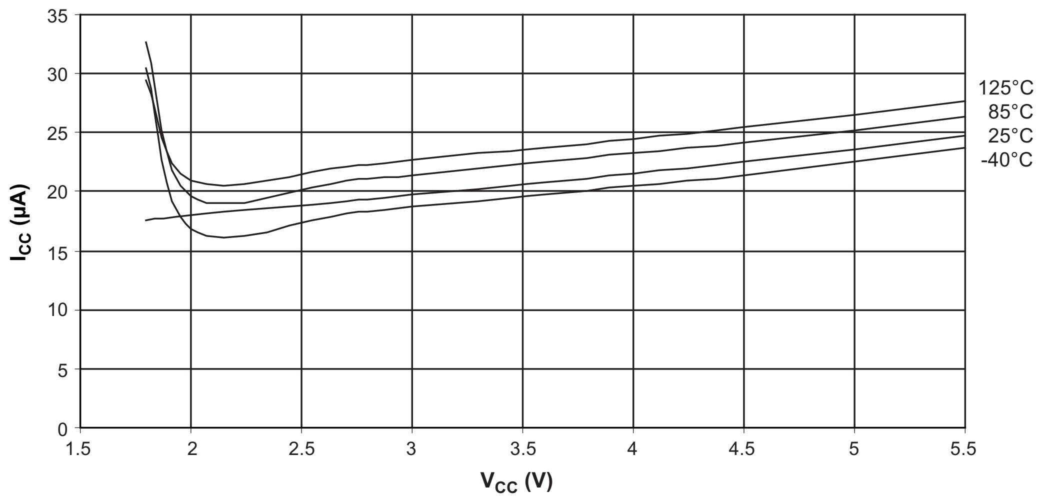
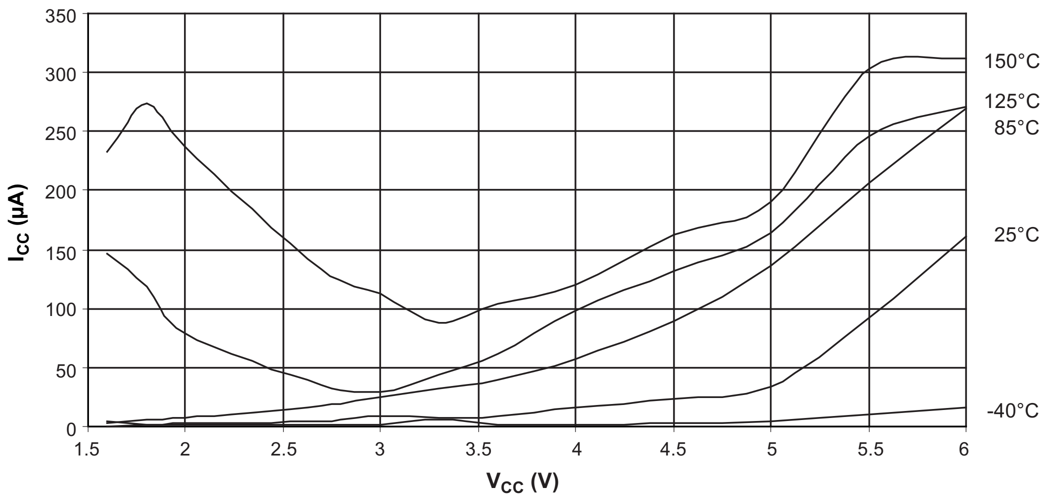


Figure 22-40. Analog Comparator Current versus V_{CC}



22.10 Current Consumption in Reset and Reset Pulse width

Figure 22-41. Reset Supply Current versus V_{CC} (0.1 to 1.0MHz, Excluding Current through the Reset Pull-up)

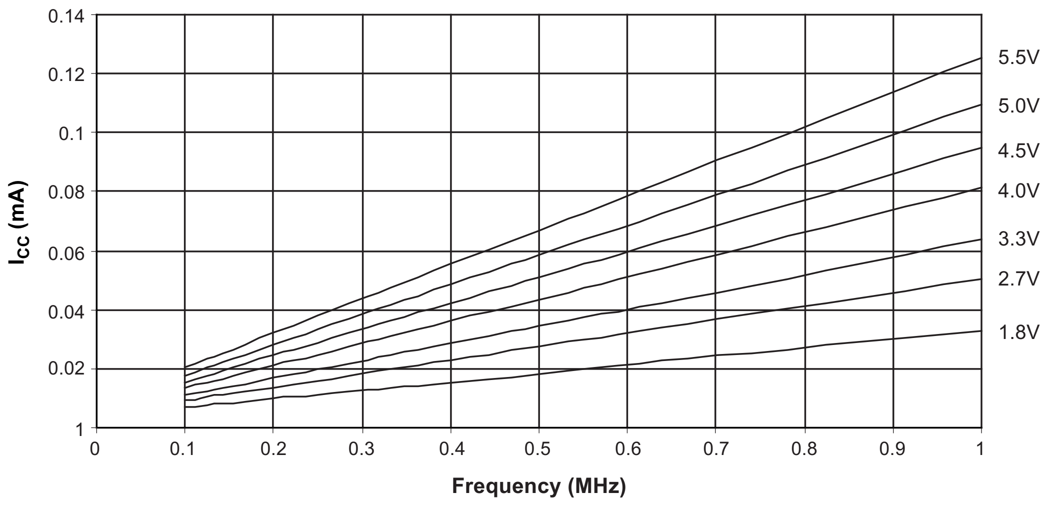


Figure 22-42. Reset Supply Current versus V_{CC} (1 to 24MHz, Excluding Current through the Reset Pull-up)

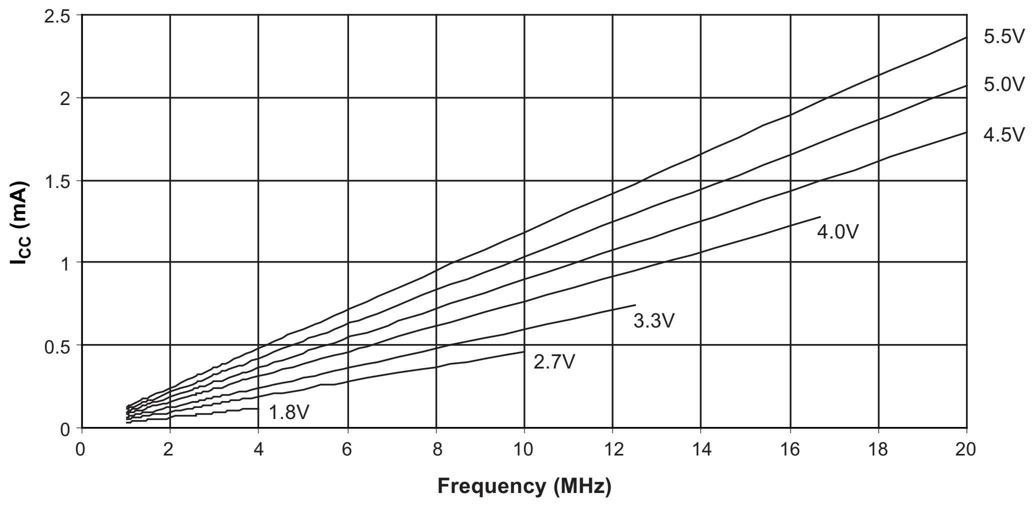
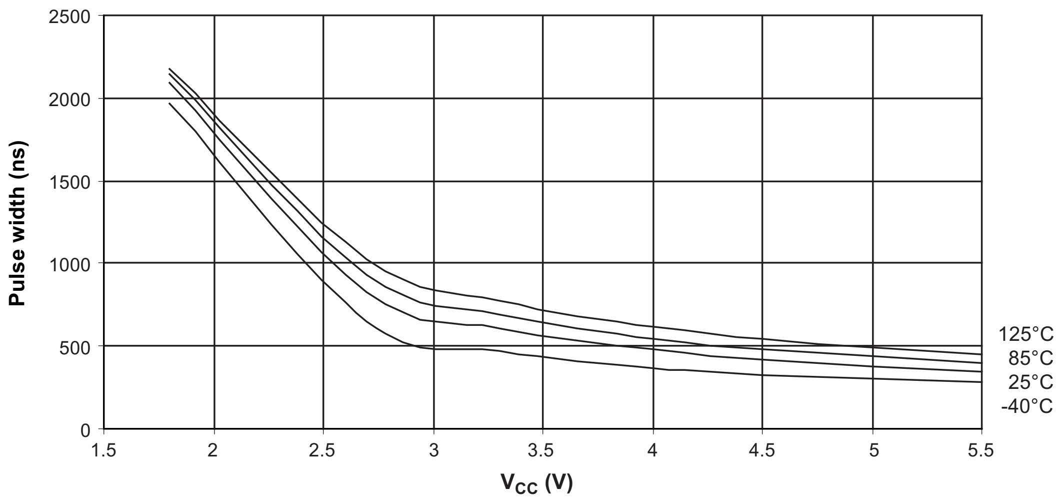


Figure 22-43. Reset Pulse Width versus V_{CC}



22.11 Analog to Digital Converter

Figure 22-44. Analog to Digital Converter Differential Mode OFFSET versus V_{CC}

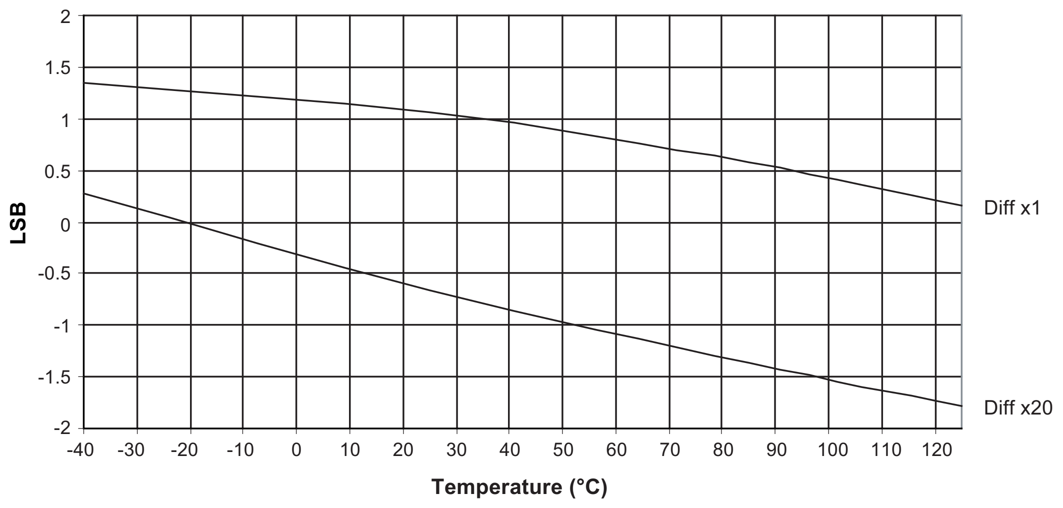


Figure 22-45. Analog to Digital Converter Single Ended Mode OFFSET versus V_{CC}

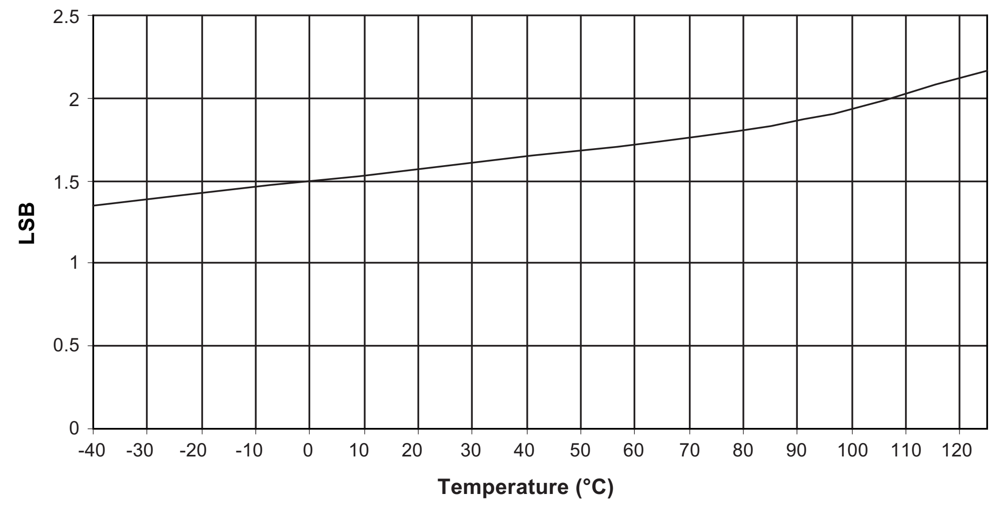


Figure 22-46. Analog to Digital Converter Differential Mode GAIN versus V_{CC}

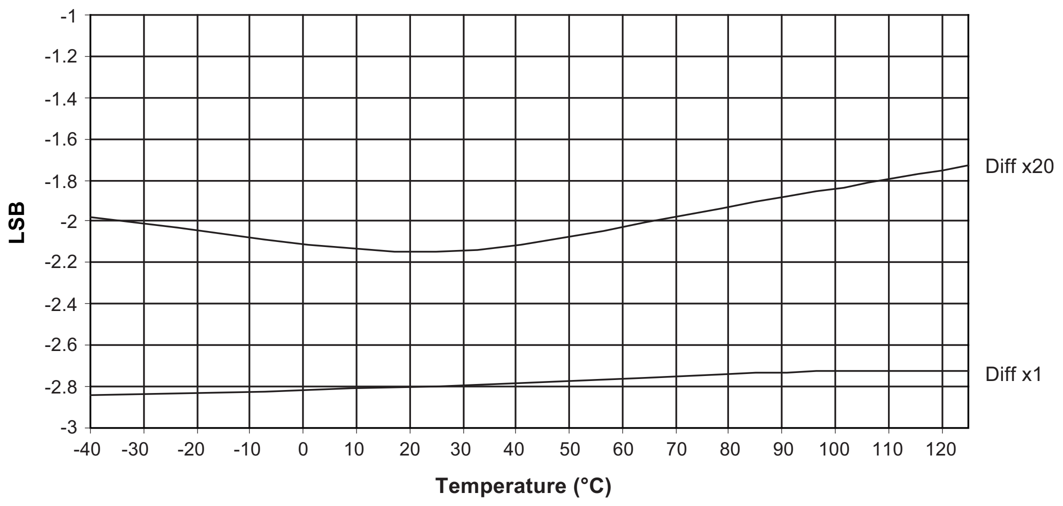


Figure 22-47. Analog to Digital Converter Single Ended Mode GAIN versus V_{CC}

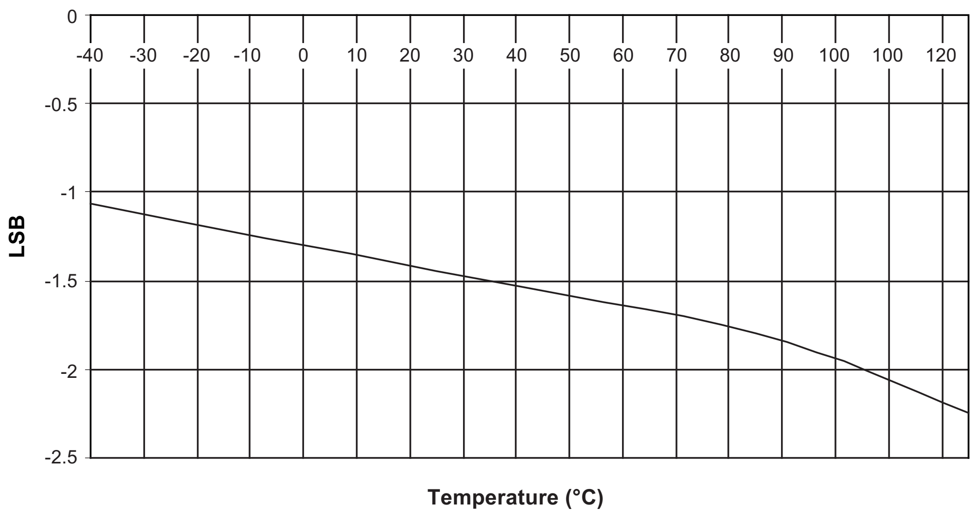


Figure 22-48. Analog to Digital Converter Differential Mode DNL versus V_{CC}

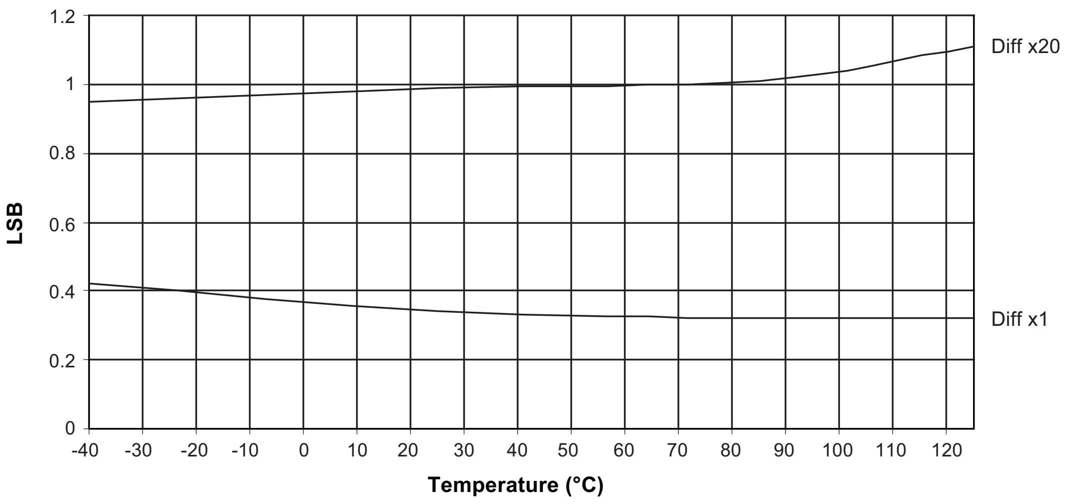


Figure 22-49. Analog to Digital Converter Single Ended Mode DNL versus V_{CC}

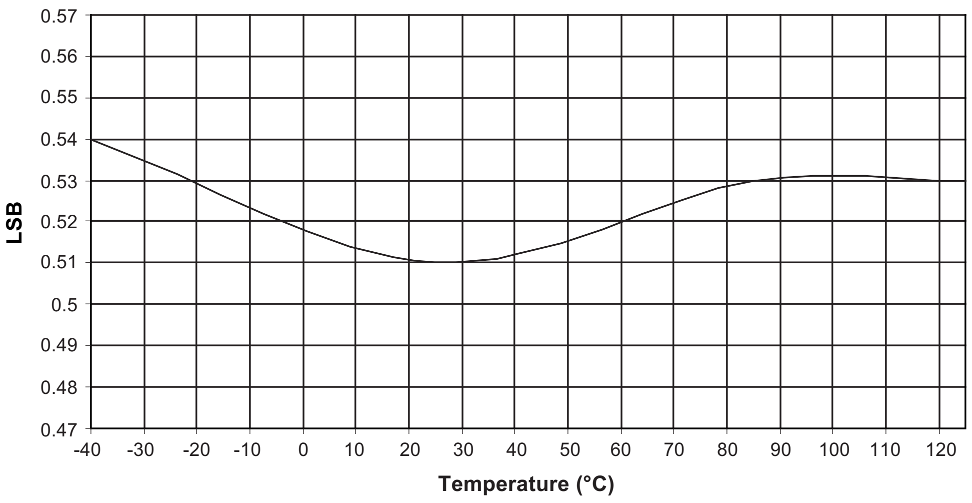


Figure 22-50. Analog to Digital Converter Differential Mode INL versus V_{CC}

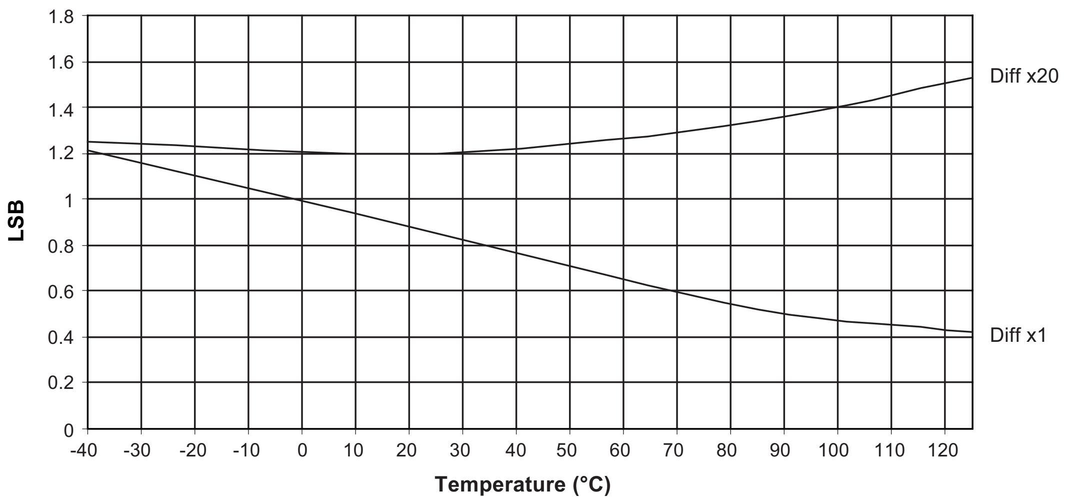
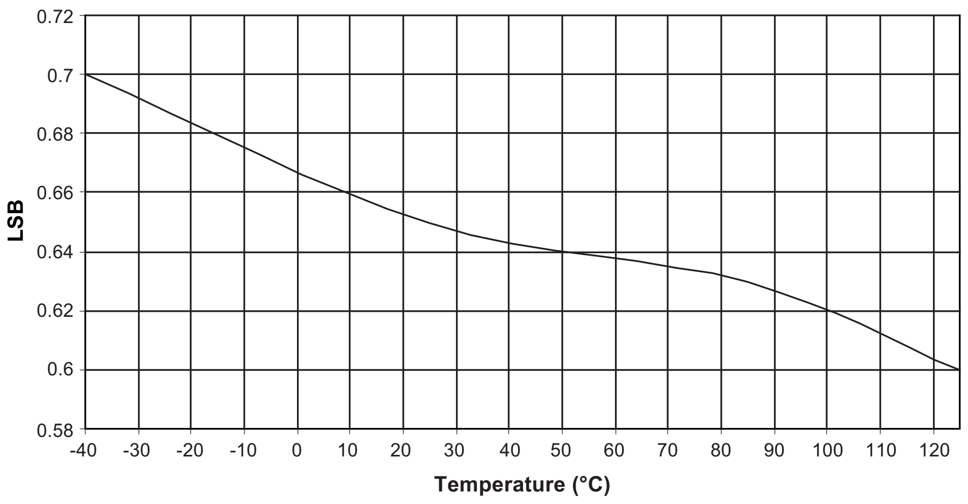


Figure 22-51. Analog to Digital Converter Single Ended Mode INL versus V_{CC}



23. 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	6
0x3E	SPH	–	–	–	–	–	–	–	SP8	9
0x3D	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	9
0x3C	Reserved	–								
0x3B	GIMSK	–	INT0	PCIE	–	–	–	–	–	54
0x3A	GIFR	–	INTF0	PCIF	–	–	–	–	–	55
0x39	TIMSK	–	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	–	70
0x38	TIFR	–	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	–	71
0x37	SPMCSR	–	–	–	CTPB	RFLB	PGWRT	PGERS	SPMEN	121
0x36	Reserved	–								
0x35	MCUCR	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	28, 49, 54
0x34	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	37
0x33	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	69
0x32	TCNT0	Timer/Counter0								70
0x31	OSCCAL	Oscillator Calibration Register								24
0x30	TCCR1	CTC1	PWM1A	COM1A1	COM1A0	CS13	CS12	CS11	CS10	77
0x2F	TCNT1	Timer/Counter1								79
0x2E	OCR1A	Timer/Counter1 Output Compare Register A								79
0x2D	OCR1C	Timer/Counter1 Output Compare Register C								79
0x2C	GTCCR	TSM	PWM1B	COM1B1	COM1B0	FOC1B	FOC1A	PSR1	PSR0	73, 78
0x2B	OCR1B	Timer/Counter1 Output Compare Register B								80
0x2A	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–		WGM01	WGM00	66
0x29	OCR0A	Timer/Counter0 – Output Compare Register A								70
0x28	OCR0B	Timer/Counter0 – Output Compare Register B								70
0x27	PLLCSR	SM	–	–	–	–	PCKE	PLLE	PLOCK	81
0x26	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	26
0x25	DT1A	DT1AH3	DT1AH2	DT1AH1	DT1AH0	DT1AL3	DT1AL2	DT1AL1	DT1AL0	87
0x24	DT1B	DT1BH3	DT1BH2	DT1BH1	DT1BH0	DT1BL3	DT1BL2	DT1BL1	DT1BL0	87
0x23	DTPS1	–	–	–	–	–	–	DTPS11	DTPS10	86
0x22	DWDR	DWDR[7:0]								118
0x21	WDTCR	WDTIF	WDTIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	38
0x20	PRR	–	–	–	–	PRTIM1	PRTIM0	PRUSI	PRADC	30
0x1F	EEARH	–	–	–	–	–	–	–	EEAR8	14
0x1E	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	14
0x1D	EEDR	EEPROM Data Register								14
0x1C	EECR	–	–	EEPM1	EEPM0	EERIE	EEMWE	EEWE	EERE	14

Notes:
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.

23. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B	Reserved	–								
0x1A	Reserved	–								
0x19	Reserved	–								
0x18	PORTB	–	–	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	53
0x17	DDRB	–	–	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	53
0x16	PINB	–	–	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	53
0x15	PCMSK	–	–	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	55
0x14	DIDR0	–	–	ADC0D	ADC2D	ADC3D	ADC1D	EIN1D	AIN0D	100, 116
0x13	GPIOR2	General Purpose I/O Register 2								
0x12	GPIOR1	General Purpose I/O Register 1								
0x11	GPIOR0	General Purpose I/O Register 0								
0x10	USIBR	USI Buffer Register								94
0x0F	USIDR	USI Data Register								94
0x0E	USISR	USICIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	95
0x0D	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	96
0x0C	Reserved	–								
0x0B	Reserved	–								
0x0A	Reserved	–								
0x09	Reserved	–								
0x08	ACSR	ACD	ACBG	ACO	ACI	ACIE	–	ACIS1	ACIS0	98
0x07	ADMUX	REFS1	REFS0	ADLAR	REFS2	MUX3	MUX2	MUX1	MUX0	112
0x06	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	114
0x05	ADCH	ADC Data Register High Byte								115
0x04	ADCL	ADC Data Register Low Byte								115
0x03	ADCSRB	BIN	ACME	IPR	–	–	ADTS2	ADTS1	ADTS0	98, 115
0x02	Reserved	–								
0x01	Reserved	–								
0x00	Reserved	–								

Notes: 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.

24. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add two registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with carry two registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl, K	Add immediate to word	Rdh: Rdl ← Rdh: Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two registers	Rd ← Rd – Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract constant from register	Rd ← Rd – K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with carry two registers	Rd ← Rd – Rr – C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with carry constant from Reg.	Rd ← Rd – K – C	Z,C,N,V,H	1
SBIW	Rdl, K	Subtract immediate from word	Rdh:Rdl ← Rdh:Rdl – K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND registers	Rd ← Rd × Rr	Z,N,V	1
ANDI	Rd, K	Logical AND register and constant	Rd ← Rd × K	Z,N,V	1
OR	Rd, Rr	Logical OR registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR register and constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd, K	Set Bit(s) in register	Rd ← Rd v K	Z,N,V	1
CBR	Rd, K	Clear Bit(s) in register	Rd ← Rd × (0xFF – K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for zero or minus	Rd ← Rd × Rd	Z,N,V	1
CLR	Rd	Clear register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set register	Rd ← 0xFF	None	1
Branch Instructions					
RJMP	k	Relative jump	PC ← PC + k + 1	None	2
IJMP		Indirect jump to (Z)	PC ← Z	None	2
RCALL	k	Relative subroutine call	PC ← PC + k + 1	None	3
ICALL		Indirect call to (Z)	PC ← Z	None	3
RET		Subroutine return	PC ← STACK	None	4
RETI		Interrupt return	PC ← STACK	I	4
CPSE	Rd, Rr	Compare, skip if equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd, Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd, Rr	Compare with carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd, K	Compare register with immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if bit in register cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if bit in register is set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if bit in I/O register cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if bit in I/O register is set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if status flag set	if (SREG (s) = 1) then PC ← PC + k + 1	None	1/2
BRBC	s, k	Branch if status flag cleared	if (SREG (s) = 0) then PC ← PC + k + 1	None	1/2
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

24. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRCC	k	Branch if carry cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if greater or equal, signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if less than zero, signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag is cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
Bit and bit-test instructions					
SBI	P,b	Set bit in I/O register	I/O (P, b) ← 1	None	2
CBI	P,b	Clear bit in I/O register	I/O (P, b) ← 0	None	2
LSL	Rd	Logical shift left	Rd(n+1) ← Rd (n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical shift right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate left through carry	Rd(0) ← C,Rd(n+1) ← Rd (n), C ← Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate right through carry	Rd(7) ← C, Rd (n) ← Rd(n+1),C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic shift right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap nibbles	Rd(3.0) ← Rd(7..4),Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to register	Rd(b) ← T	None	1
SEC		Set carry	C ← 1	C	1
CLC		Clear carry	C ← 0	C	1
SEN		Set negative flag	N ← 1	N	1
CLN		Clear negative flag	N ← 0	N	1
SEZ		Set zero flag	Z ← 1	Z	1
CLZ		Clear zero flag	Z ← 0	Z	1
SEI		Global interrupt enable	I ← 1	I	1
CLI		Global interrupt disable	I ← 0	I	1
SES		Set signed test flag	S ← 1	S	1
CLS		Clear signed test flag	S ← 0	S	1
SEV		Set twos complement overflow.	V ← 1	V	1
CLV		Clear twos complement overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set half carry flag in SREG	H ← 1	H	1

24. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear half carry flag in SREG	H ← 0	H	1
Data transfer instructions					
MOV	Rd, Rr	Move between registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy register word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load immediate	Rd ← K	None	1
LD	Rd, X	Load indirect	Rd ← (X)	None	2
LD	Rd, X+	Load indirect and post-inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load indirect and pre-dec.	X ← X – 1, Rd ← (X)	None	2
LD	Rd, Y	Load indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load indirect and post-inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load indirect and pre-dec.	Y ← Y – 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load indirect with displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load indirect and post-inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load indirect and pre-dec.	Z ← Z – 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load indirect with displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store indirect	(X) ← Rr	None	2
ST	X+, Rr	Store indirect and post-inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store indirect and pre-dec.	X ← X – 1, (X) ← Rr	None	2
ST	Y, Rr	Store indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store indirect and post-inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store indirect and pre-dec.	Y ← Y – 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store indirect with displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store indirect and post-inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store indirect and pre-dec.	Z ← Z – 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store indirect with displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store direct to SRAM	(k) ← Rr	None	2
LPM		Load program memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load program memory and post-inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store program memory	(z) ← R1:R0	None	
IN	Rd, P	In port	Rd ← P	None	1
OUT	P, Rr	Out port	P ← Rr	None	1
PUSH	Rr	Push register on stack	STACK ← Rr	None	2
POP	Rd	Pop register from stack	Rd ← STACK	None	2
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep	(see specific descr. for sleepfunction)	None	1
WDR		Watchdog reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For on-chip debug only	None	N/A

25. Ordering Information

Power Supply	Speed (MHz)	Ordering Code	Package	Operation Range
2.7 - 5.5V	8 - 16[3]	ATtiny25/45/85-15ST ATtiny25/45/85-15ST1 ATtiny25/45/85-15SZ	T5	Automotive (–40°C to +85°C) Automotive (–40°C to +105°C) Automotive (–40°C to +125°C)
2.7 - 5.5V	8 - 16[3]	ATtiny25/45/85-15MT ATtiny25/45/85-15MT1 ATtiny25/45/85-15MZ	PC	Automotive (–40°C to +85°C) Automotive (–40°C to +105°C) Automotive (–40°C to +125°C)

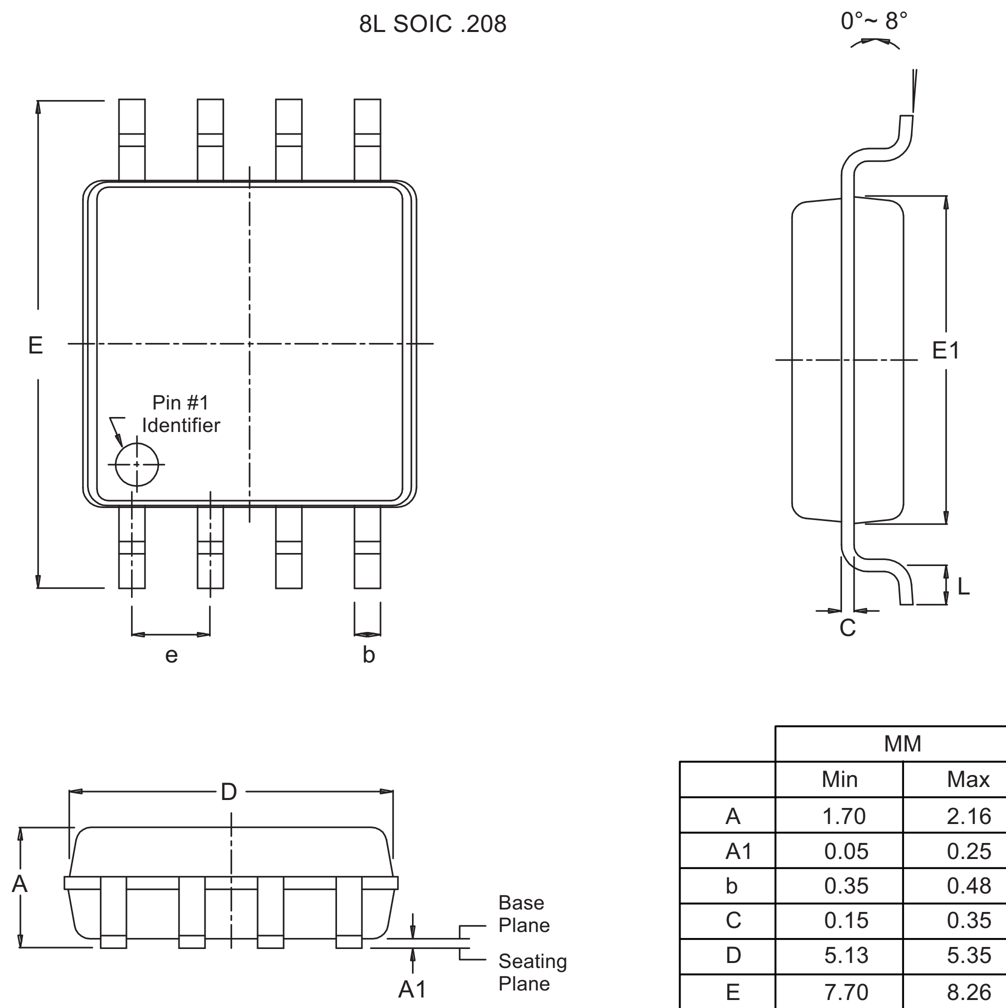
Notes: 1. Green and ROHS packaging
2. Tape and Reel with Dry-pack delivery.
3. For Speed versus V_{CC},see Figure 21-2 on page 139.

Table 25-1. Package Type

Package Type	
T5	T5 - 8-lead, 0.208" body width Plastic gull wing small outline package
PC	PC - 20-lead, 4.0 x 4.0mm body, 0.50mm pitch Quad flat no lead package (QFN)

26. Packaging Information

26.1 T5



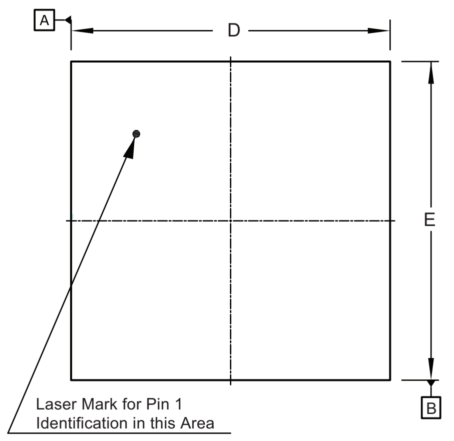
	MM		INCH	
	Min	Max	Min	MAx
A	1.70	2.16	.066	.085
A1	0.05	0.25	.002	.010
b	0.35	0.48	.014	.015
C	0.15	0.35	.006	.014
D	5.13	5.35	.202	.211
E	7.70	8.26	.303	.325
E1	5.18	5.40	.204	.212
L	0.51	0.85	.020	.033
e	1.27			
Q	0°		8°	

07/27/07

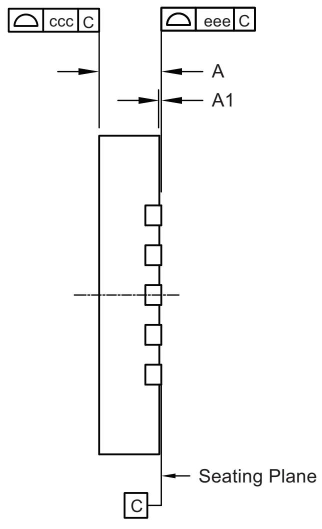
	TITLE	GPC	DRAWING NO.	REV.
Atmel **Package Drawing Contact:** packagedrawings@atmel.com	T5, 8-Lead, 0.208" Body Width Plastic: Quad Wing Small Outline Package (SOIC)		T5	B

26.2 PC

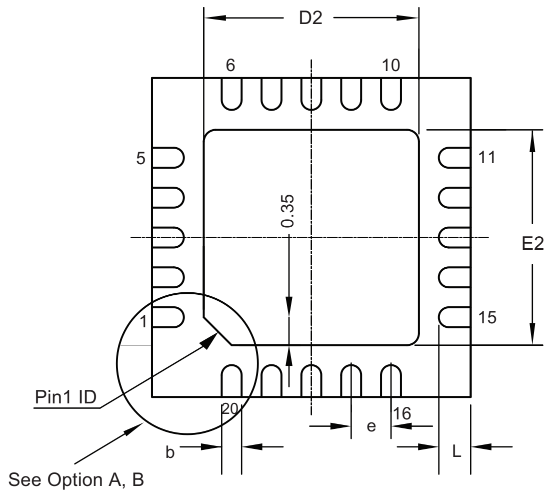
DRAWINGS NOT SCALED



Top View



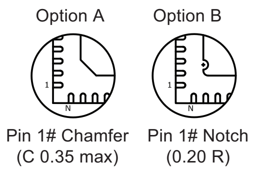
Side View



Bottom View

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.70	0.75	0.80	
A1	0.00	0.02	0.05	
D/E	3.90	4.00	4.10	
D2/E2	2.50	2.60	2.70	
L	0.35	0.45	0.55	
b	0.20	0.25	0.30	
e	0.50 BSC			
Tolerances of Form and Position				
ccc	0.10			
eee	0.08			
n	20			



Notes: 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation WGGD-5 for proper dimensions, tolerances, datums, etc. (excepted D2/E2 Min et Nom.)
2. Dimensions b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip.
If the terminal has the optical radius on the other end of the terminal, the dimensions should not be measured in that radius area.

08/01/14

	TITLE	GPC	DRAWING NO.	REV.
Atmel **Package Drawing Contact:** packagedrawings@atmel.com	PC, 20-Lds - 0.50mm Pitch, 4x4x0.8mm Body size Very very Thin Quad Flat No Lead Package (WQFN) Sawn	ZVF	PC	J

27. Errata

The revision letter in this section refers to the revision of the Atmel® ATtiny25/45/85 device.

27.1 ATtiny25, Revision E

1. No known errata. Flash security improvements.

27.2 ATtiny45, Revision G

1. No known errata. Flash security improvements.

27.3 ATtiny85, Revision C

1. No known errata. Flash security improvements.

28. Revision History

Please note that the following page numbers referred to in this section refer to the specific revision mentioned, not to this document.

Revision No.	History
7598J-AVR-12/14	• Put datasheet in the latest template
7598I-AVR-10/12	• Section 27 “Package Information” on page 188 updated
7598H-AVR-07/09	• Absolute Maximum Ratings updated
7598G-AVR-03/08	• Modified Section 6. “Power Management and Sleep Modes” on page 28 • Modified Section 6.1 “MCU Control Register – MCUCR” on page 28 • Modified Table 6-2 on page 29. • Added Section 6.5 “Limitations” on page 29. • Modified Section 6.6 “Power Reduction Register” on page 30.
7598F-AVR-11/07	• Correction to ICC Active, Table 21-1 on page 137.
7598E-AVR-03/07	• POR updated, see Section 7.3 “Power-on Reset” on page 34
7598D-AVR-02/07	• Clarification of power On reset specifications table. • Errata list updated. • Added QFN packages.
7598C-AVR-09/06	• Correction of package codification and drawings.
7598B-AVR-08/06	• Clarification of several TBD values • Addition of the power on reset specification • DC characteristics limits completed after corner run characterization • Typical characteristic curves produced
7598A-AVR-04/06	• Automotive grade created: Features: • Change voltage and temperature range (2.7V – 5.5V), (–40°C, +125°C) • Adapt Stand-by current to automotive temperature range Packages: • PDIP removed • Ordering info limited to automotive versions (green only, dry pack) DC and AC parameters Only PRELIMINARY values are produced

29. Table of Contents

Enabling Unlimited Possibilities®