ATtiny261/ATtiny461/ATtiny861 Automotive

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

DATASHEET

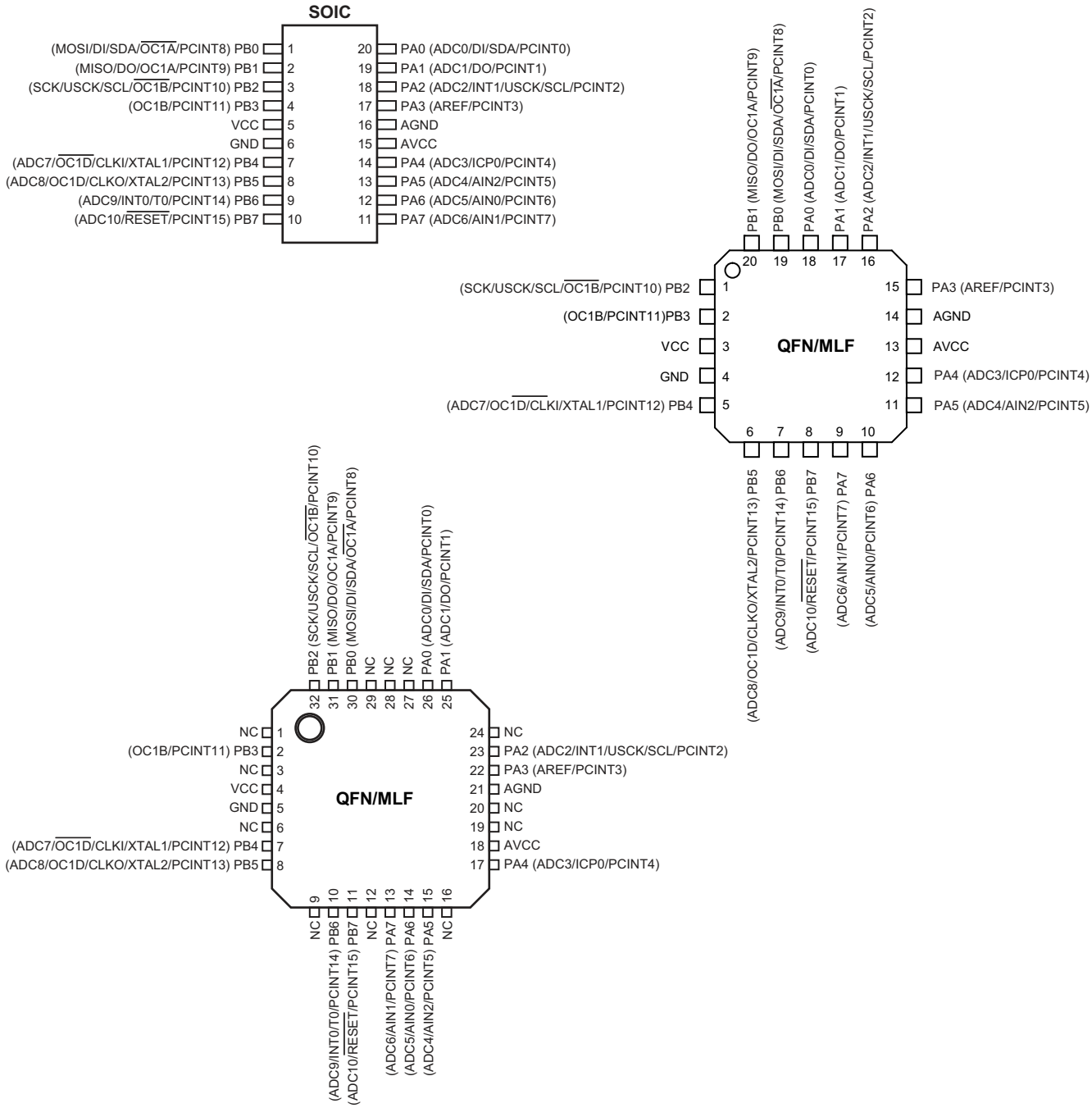
Features

- High performance, low power AVR® 8-Bit microcontroller
- Advanced RISC architecture
  - 123 powerful instructions – most single clock cycle execution
  - 32 x 8 general purpose working registers
  - Fully static operation
- Non-volatile program and data memories
  - 2/4/8K byte of in-system programmable program memory flash (Atmel® ATtiny261/461/861)
    - Endurance: 10,000 write/erase cycles
  - 128/256/512 bytes in-system programmable EEPROM (Atmel ATtiny261/461/861)
    - Endurance: 100,000 write/erase cycles
  - 128/256/512 bytes internal SRAM (ATtiny261/461/861)
  - Programming lock for self-programming flash program and EEPROM data security
- Peripheral features
  - 8/16-bit Timer/Counter with prescaler
  - 8/10-bit high speed Timer/Counter with separate prescaler
    - 3 high frequency PWM outputs with separate output compare registers
    - Programmable dead time generator
  - Universal serial interface with start condition detector
  - 10-bit ADC
    - 11 single ended channels
    - 16 differential ADC channel pairs
    - 15 differential ADC channel pairs with programmable gain (1x, 8x, 20x, 32x)
  - 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
  - 16 programmable I/O lines
  - 20-pin SOIC, 32-pad MLF and 20-lead TSSOP
- Operating voltage:
  - 2.7 - 5.5V for Atmel ATtiny261/461/861
- Speed grade:
  - Atmel® ATtiny261/461/861: 0 - 8MHz at 2.7 - 5.5V, 0 - 16MHz at 4.5 - 5.5V
  - Operating temperature: Automotive (–40°C to +125°C)
- Low power consumption
  - Active mode ATD On: 1MHz, 2.7V, 25°C: 300µA
  - Power-down mode no watchdog: 2.7V, 25°C: 0.12µA

1. Pin Configurations

Figure 1-1. Pinout ATtiny261/461/861



Note: The large center pad underneath the QFN/MLF package should be soldered to ground on the board to ensure good mechanical stability.

1.1 Disclaimer

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

1.2 Automotive Quality Grade

The Atmel® ATtiny261/461/861 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 Atmel ATtiny261/461/861 have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, the product is available in only one temperature grade, see Table 1-1.

Table 1-1. Temperature Grade Identification for Automotive Products

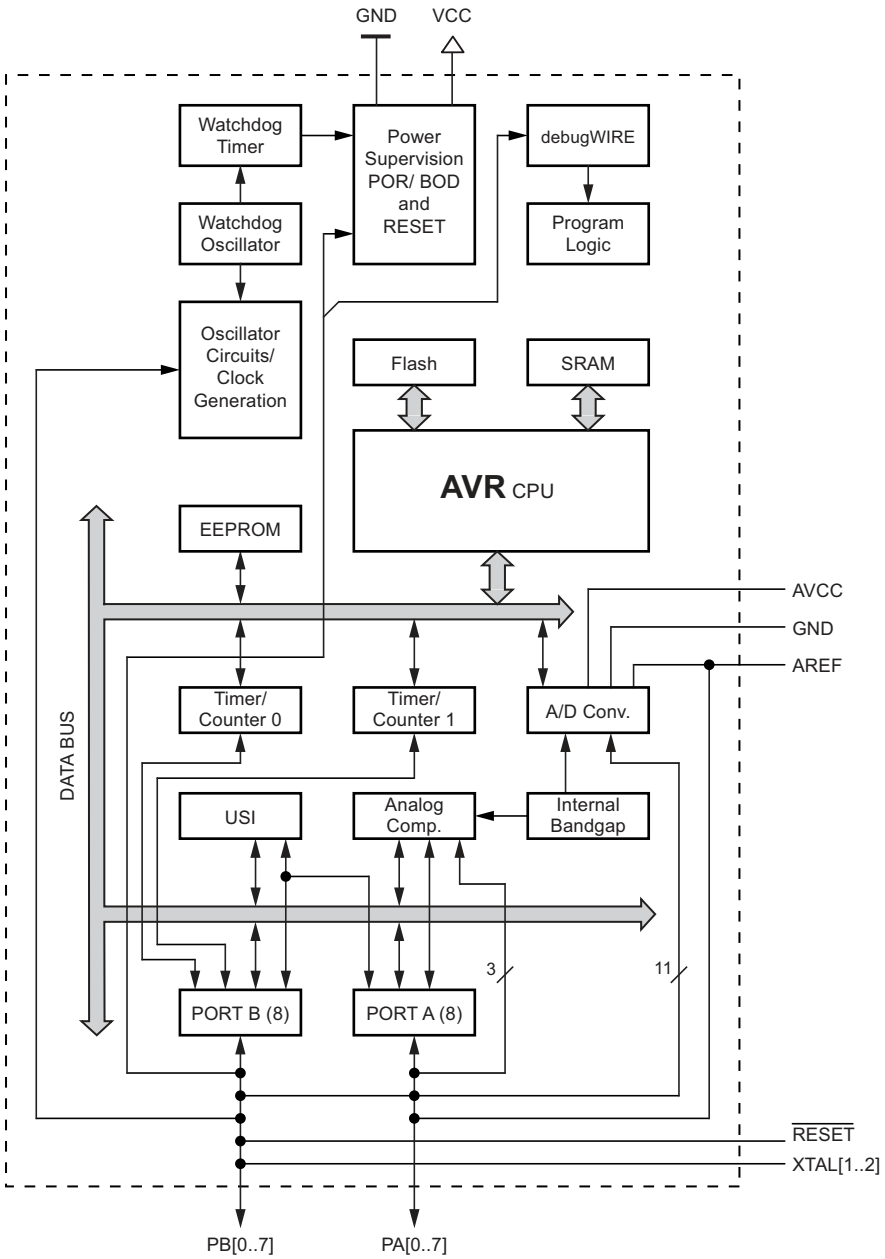
Temperature	Temperature Identifier	Comments
–40; +125	Z	Full automotive temperature range

2. Overview

The Atmel® ATtiny261/461/861 is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the Atmel ATtiny261/461/861 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The 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® ATtiny261/461/861 provides the following features: 2/4/8Kbyte of in-system programmable flash, 128/256/512 bytes EEPROM, 128/256/512 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 high density non-volatile memory technology. The on-chip ISP flash allows the program memory to be re-programmed in-system through an SPI serial interface, by a conventional non-volatile memory programmer or by an on-chip boot code running on the AVR core.

The Atmel ATtiny261/461/861 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

2.2 Pin Descriptions

2.2.1 VCC

Supply voltage.

2.2.2 GND

Ground.

2.2.3 AVCC

Analog supply voltage.

2.2.4 AGND

Analog ground.

2.2.5 Port A (PA7..PA0)

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

Port A also serves the functions of various special features of the Atmel ATtiny261/461/861 as listed on Section 12.3.2 “Alternate Functions of Port A” on page 62.

2.2.6 Port B (PB7..PB0)

Port B is an 8-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 ATtiny261/461/861 as listed on

Section 12.3.1 “Alternate Functions of Port B” on page 59.

2.2.7 $\overline{\text{RESET}}$

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 23-3 on page 174. Shorter pulses are not guaranteed to generate a reset.

3. Resources

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

4. About Code Examples

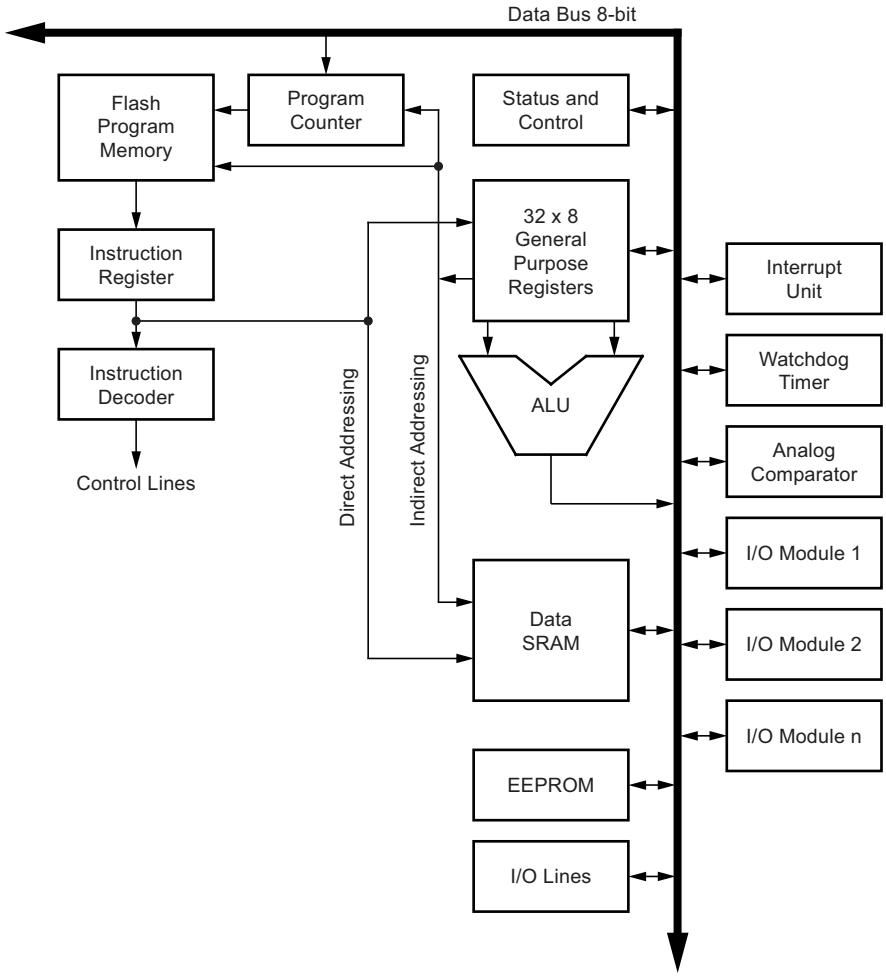
This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling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

5. AVR CPU Core

5.1 Overview

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.

Figure 5-1. Block Diagram of the AVR Architecture



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

The fast-access register file contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle arithmetic logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the register file, the operation is executed, and the result is stored back in the register file – 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 have a single 16-bit word format. Most AVR instructions are 16-bit 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.

5.2 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.

5.3 Status Register

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

5.3.1 SREG – AVR Status Register

The AVR® status register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**SREG**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The global interrupt enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the global interrupt enable register is cleared, none of the interrupts are enabled 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 ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the two 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 arithmetics. 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.

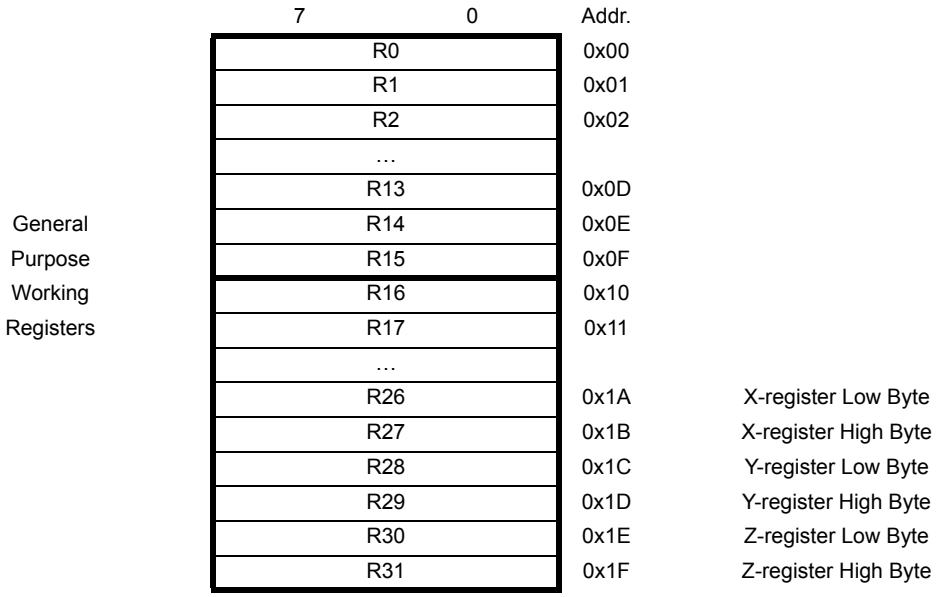
5.4 General Purpose Register File

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

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

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

Figure 5-2. AVR CPU General Purpose Working Registers



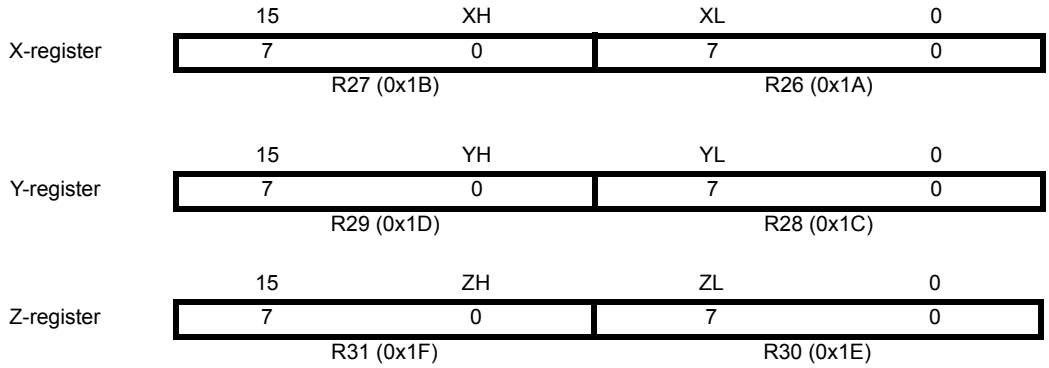
Most of the instructions operating on the register file have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 5-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not 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.

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

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

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



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

5.5 Stack Pointer

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

The stack pointer points to the data SRAM stack area where the subroutine and interrupt stacks are located. This stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The stack pointer must be set to point above 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.

5.5.1 SPH and SPL – Stack Pointer Register

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

5.6 Instruction Execution Timing

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

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

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

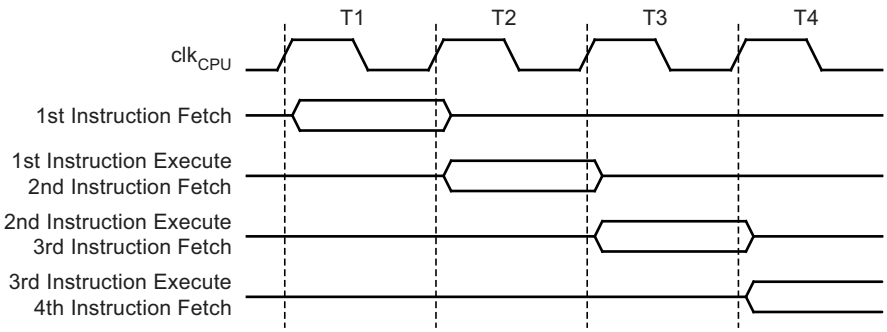
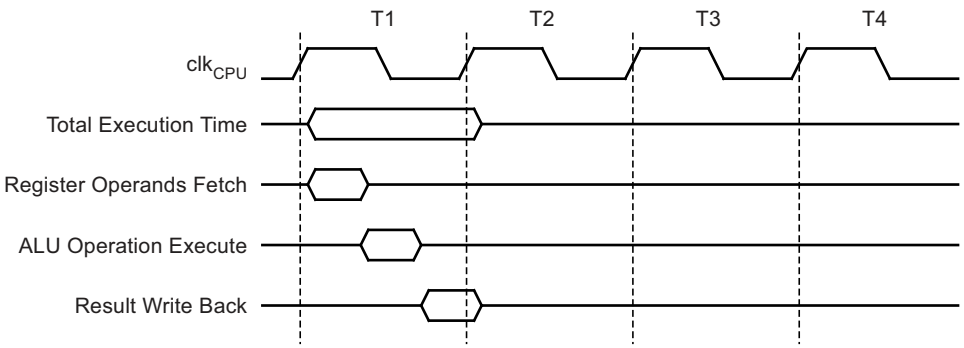


Figure 5-5 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 5-5. Single Cycle ALU Operation



5.7 Reset and Interrupt Handling

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

The lowest addresses in the program memory space are by default defined as the reset and interrupt vectors. The complete list of vectors is shown in Section 10. “Interrupts” on page 47. 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, EEMPE   ; start EEPROM write
sbi    EECR, EEPE
out    SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

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

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

Assembly Code Example

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

C Code Example

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

5.7.1 Interrupt Response Time

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

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

6. AVR Memories

This section describes the different memories in the Atmel® ATtiny261/461/861. The AVR architecture has two main memory spaces, the Data memory and the Program memory space. In addition, the Atmel ATtiny261/461/861 features an EEPROM memory for data storage. All three memory spaces are linear and regular.

6.1 In-System Re-programmable Flash Program Memory

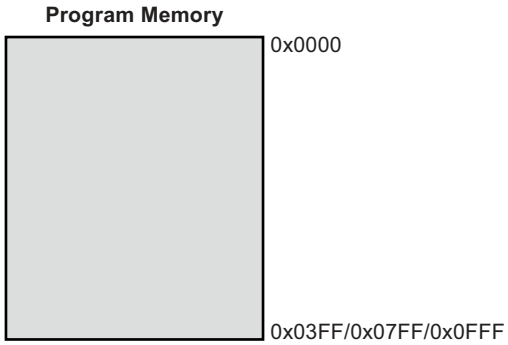
The Atmel ATtiny261/461/861 contains 2/4/8K byte on-chip in-system reprogrammable flash memory for program storage. Since all AVR® instructions are 16 or 32 bits wide, the flash is organized as 1024/2048/4096 × 16.

The flash memory has an endurance of at least 10,000 write/erase cycles. The Atmel ATtiny261/461/861 program counter (PC) is 10/11/12 bits wide, thus addressing the 1024/2048/4096 program memory locations. Section 22. “Memory Programming” on page 156 contains a detailed description on flash data serial downloading using the SPI pins.

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

Timing diagrams for instruction fetch and execution are presented in Section 5.6 “Instruction Execution Timing” on page 13.

Figure 6-1. Program Memory Map



6.2 SRAM Data Memory

Figure 6-2 shows how the Atmel ATtiny261/461/861 SRAM memory is organized.

The lower 224/352/608 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.

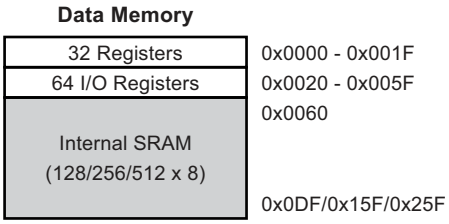
The direct addressing reaches the entire data space.

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 Atmel ATtiny261/461/861 are all accessible through all these addressing modes. The register file is described in Section 5.4 “General Purpose Register File” on page 11.

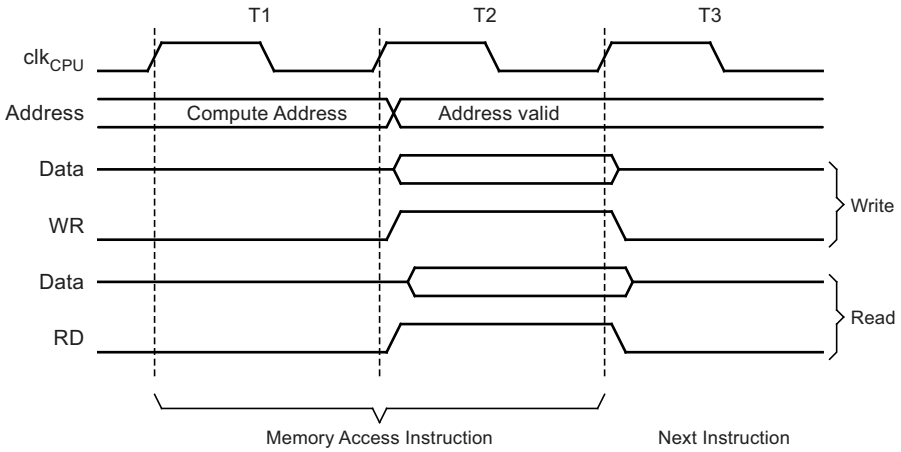
Figure 6-2. Data Memory Map



6.2.1 Data Memory Access Times

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

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



6.3 EEPROM Data Memory

The Atmel® ATtiny261/461/861 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 22.9 “Serial Downloading” on page 167.

6.3.1 EEPROM Read/Write Access

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

The write access times for the EEPROM are given in Table 6-1 on page 22. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write 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 6.3.6 “Preventing EEPROM Corruption” on page 20 for details on how to avoid problems in these situations.

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

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

6.3.2 Atomic Byte Programming

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

6.3.3 Split Byte Programming

It is possible to split the erase and write cycle 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).

6.3.4 Erase

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

6.3.5 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEPMn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in Table 1-1 on page 4). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated oscillator is used to time the EEPROM accesses. Make sure the oscillator frequency is within the requirements described in Section 7.12.1 “OSCCAL – Oscillator Calibration Register” on page 31.

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 (r18:r17) in address register
        out     EEARH, r18
        out     EEARL, r17
        ; Write data (r16) to data register
        out     EEDR, r16
        ; Write logical one to EEMPE
        sbi     EECR,EEMPE
        ; Start eeprom write by setting EEPE
        sbi     EECR,EEPE
        ret
```

C Code Example

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

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 (r18:r17) in address register
        out     EEARH, r18
        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 */
        EEAR = ucAddress;
        /* Start eeprom read by writing EERE */
        EECR |= (1<<EERE);
        /* Return data from data register */
        return EEDR;
}
```

6.3.6 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.

6.4 I/O Memory

The I/O space definition of the Atmel® ATtiny261/461/861 is shown in Section 25. “Register Summary” on page 189.

All Atmel ATtiny261/461/861 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.

6.4.1 General Purpose I/O Registers

The Atmel ATtiny261/461/861 contains three general purpose I/O registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags. General purpose I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

6.5 Register Description

6.5.1 EEARH and EEARL – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**EEAR8**	**EEARH**
0x1E (0x3E)	**EEAR7**	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	**EEARL**
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
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	X	
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 Atmel ATtiny261/461/861.

- **Bits 8:0 – EEAR8:0: EEPROM Address**

The EEPROM address registers – EEARH and EEARL – 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.

6.5.2 EEDR – EEPROM Data Register

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

- **Bits 7:0 – EEDR7:0: EEPROM Data**

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

6.5.3 EECR – EEPROM Control Register

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

- **Bit 7 – Res: Reserved Bit**

This bit is reserved for future use and will always read as 0 in Atmel® ATtiny261/461/861. 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 ATtiny261/461/861 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 6-1. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 6-1. EEPROM Mode Bits

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

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

Writing EERIE to 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.

6.5.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	**MSB**							**LSB**	**GPIOR2**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6.5.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	**MSB**							**LSB**	**GPIOR1**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6.5.6 GPIOR0 – General Purpose I/O Register 0

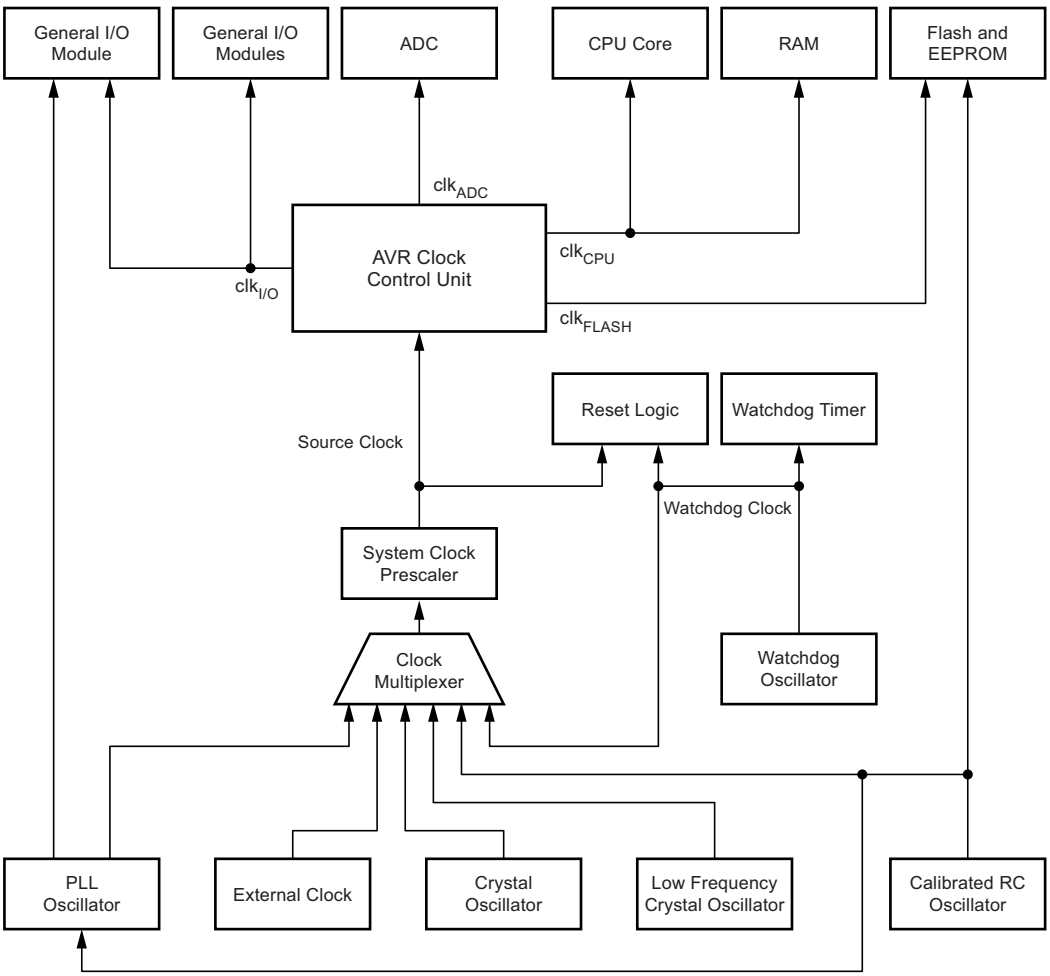
Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	**MSB**							**LSB**	**GPIOR0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

7. System Clock and Clock Options

7.1 Clock Systems and their Distribution

Figure 7-1 presents the principal clock systems in the AVR® and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in Section 8. “Power Management and Sleep Modes” on page 34. The clock systems are detailed below.

Figure 7-1. Clock Distribution



7.1.1 CPU Clock – clk_{CPU}

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

7.1.2 I/O Clock – $clk_{I/O}$

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

7.1.3 Flash Clock – clk_{FLASH}

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

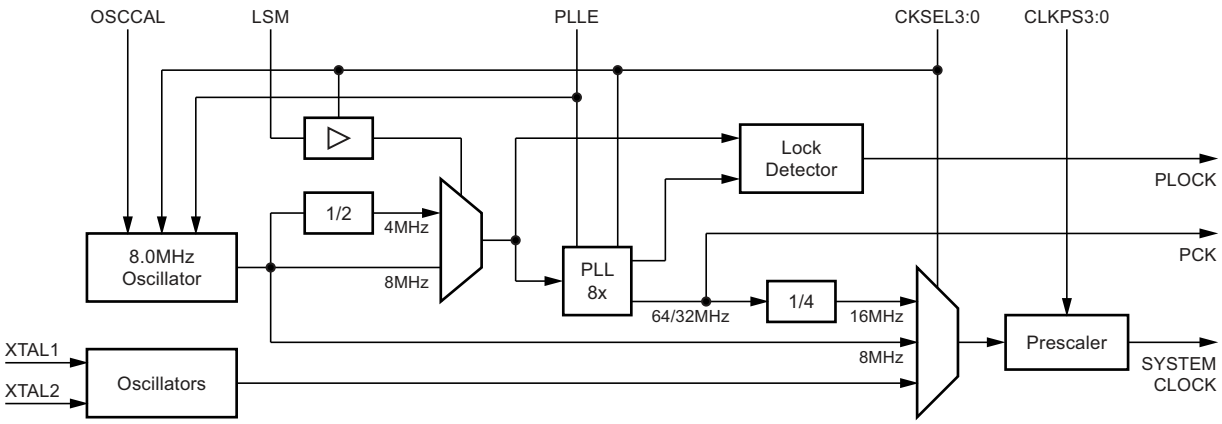
7.1.4 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

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

The internal PLL in Atmel® ATtiny261/461/861 generates a clock frequency that is 8x multiplied from a source input. By default, the PLL uses the output of the internal 8.0MHz RC oscillator as source. Alternatively, if the LSM bit of the PLLCSR is set the PLL will use the output of the RC oscillator divided by two. 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/Counter1or as a system clock. See Figure 7-2. The frequency of the fast peripheral clock is divided by two when LSM of PLLCSR is set, resulting in a clock frequency of 32MHz. Note, that LSM can not be set if PLL_{CLK} is used as a system clock.

Figure 7-2. PCK Clocking System



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 when:

- The PLLE bit of the PLLCSR register is set.
- The CKSEL fuse are programmed to ‘0001’.

The PLLCSR bit PLOCK is set when PLL is locked.

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

7.2 Clock Sources

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

Table 7-1. Device Clocking Options Select(1) versus PB4 and PB5 Functionality

Device Clocking Option	CKSEL3..0	PB4	PB5
External clock	0000	XTAL1	I/O
PLL clock	0001	I/O	I/O
Calibrated internal RC oscillator 8.0MHz	0010	I/O	I/O
Watchdog oscillator 128kHz	0011	I/O	I/O
External low-frequency crystal	01xx	XTAL1	XTAL2
External crystal/ceramic resonator (0.4 - 0.9MHz)	1000	XTAL1	XTAL2
External crystal/ceramic resonator (0.4 - 0.9MHz)	1001	XTAL1	XTAL2
External crystal/ceramic resonator (0.9 - 3.0MHz)	1010	XTAL1	XTAL2
External crystal/ceramic resonator (0.9 - 3.0MHz)	1011	XTAL1	XTAL2
External crystal/ceramic resonator (3.0 - 8.0MHz)	1100	XTAL1	XTAL2
External crystal/ceramic resonator (3.0 - 8.0MHz)	1101	XTAL1	XTAL2
External crystal/ceramic resonator (8.0 - 16.0MHz)	1110	XTAL1	XTAL2
External crystal/ceramic resonator (8.0 - 16.0MHz)	1111	XTAL1	XTAL2

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

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

Table 7-2. Number of Watchdog Oscillator Cycles

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

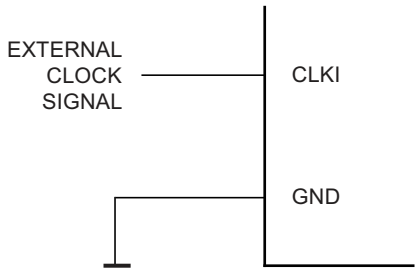
7.3 Default Clock Source

The device is shipped with CKSEL = “0010”, SUT = “10”, and CKDIV8 programmed. The default clock source setting is therefore the internal RC oscillator running at 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.

7.4 External Clock

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

Figure 7-3. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 7-3.

Table 7-3. 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 7.11 “System Clock Prescaler” on page 31 for details.

7.5 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 7-4. When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 7-5. See also Section 7-2 “PCK Clocking System” on page 25.

Table 7-4. PLLCK Operating Modes

CKSEL3..0	Nominal Frequency
0001	16MHz

Table 7-5. Start-up Times for the PLLCK

SUT1..0	Start-up Time from Power Down	Additional Delay from Power-On-Reset (V_{CC} = 5.0V)	Recommended Usage
00	14CK + 1K (1024) + 4ms	4ms	BOD enabled
01	14CK + 16K (16384) + 4ms	4ms	Fast rising power
10	14CK + 1K (1024) + 64ms	4ms	Slowly rising power
11	14CK + 16K (16384) + 64ms	4ms	Slowly rising power

7.6 Calibrated Internal RC Oscillator

By default, the internal RC oscillator provides an approximate 8.0MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. See Table 23-1 on page 173 and Section 24.9 “Internal Oscillator Speed” on page 188 for more details. The device is shipped with the CKDIV8 fuse programmed. See Section 7.11 “System Clock Prescaler” on page 31 for more details.

This clock may be selected as the system clock by programming the CKSEL fuses as shown in Table 7-6. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL register and thereby automatically calibrates the RC oscillator. The accuracy of this calibration is shown as factory calibration in Table 23-1 on page 173.

By changing the OSCCAL register from SW, see Section 7.12.1 “OSCCAL – Oscillator Calibration Register” on page 31, it is possible to get a higher calibration accuracy than by using the factory calibration. The accuracy of this calibration is shown as User calibration in Table 23-1 on page 173.

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 22.4 “Calibration Byte” on page 158.

Table 7-6. Internal Calibrated RC Oscillator Operating Modes(1)(2)

Frequency Range (MHz)	CKSEL3..0
7.84 - 8.16	0010

Notes: 1. The device is shipped with this option selected.
2. If 8MHz frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8.

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

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

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6CK	14CK	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.

7.7 128 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 “0011”.

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

Table 7-8. Start-up Times for the 128 kHz Internal Oscillator

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

7.8 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 ‘0100’. The crystal should be connected as shown in Figure 7-4. 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 7-9.

Table 7-9. Start-up Times for the Low Frequency Crystal Oscillator Clock Selection

SUT1..0	Start-up Time from Power Down and Power Save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended usage
00	1K (1024) CK(1)	4ms	Fast rising power or BOD enabled
01	1K (1024) CK(1)	64ms	Slowly rising power
10	32K (32768) CK	64ms	Stable frequency at start-up
11	Reserved		

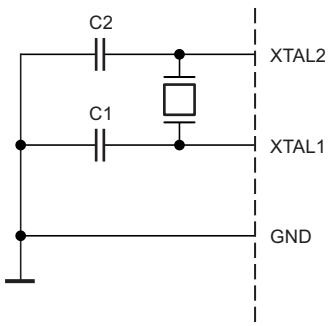
Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

7.9 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 7-4. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 7-10 on page 30. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 7-4. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 7-10.

Table 7-10. Crystal Oscillator Operating Modes

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

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

The CKSEL0 fuse together with the SUT1..0 fuses select the start-up times as shown in Table 7-11.

Table 7-11. 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 + 4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	14CK + 65 ms	Ceramic resonator, slowly rising power
0	10	1K (1024) CK[2]	14CK	Ceramic resonator, BOD enabled
0	11	1K (1024)CK[2]	14CK + 4.1 ms	Ceramic resonator, fast rising power
1	00	1K (1024)CK[2]	14CK + 65 ms	Ceramic resonator, slowly rising power
1	01	16K (16384) CK	14CK	Crystal oscillator, BOD enabled
1	10	16K (16384) CK	14CK + 4.1 ms	Crystal oscillator, fast rising power
1	11	16K (16384) CK	14CK + 65 ms	Crystal oscillator, slowly rising power

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

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

7.10 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.

7.11 System Clock Prescaler

The Atmel® ATtiny261/461/861 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 7-12 on page 33.

7.11.1 Switching Time

When switching between prescaler settings, the system clock prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than 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 × 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.

7.12 Register Description

7.12.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
0x31 (0x51)	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

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

The oscillator calibration register is used to trim the calibrated internal RC oscillator to remove process variations from the oscillator frequency. A pre-programmed calibration value is automatically written to this register during chip reset, giving the Factory calibrated frequency as specified in Table 23-1 on page 173.

The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 23-1 on page 173. Calibration outside that range is not guaranteed.

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

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 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.

7.12.2 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	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®ATtiny261/461/861 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 7-12.

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

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

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

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

Table 7-12. Clock Prescaler Select

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

8. Power Management and Sleep Modes

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.

8.1 Sleep Modes

Figure 7-1 on page 24 presents the different clock systems in the Atmel® ATtiny261/461/861, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 8-1 shows the different sleep modes and their wake up sources.

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

	Active Clock Domains					Oscillators	Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{PCK}	Main Clock Source Enabled	INT0, INT1 and Pin Change	SPM/EEPROM Ready	ADC	WDT Interrupt	USI Interrupt	Other I/O
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	
Standby							X(1)			X	X	

Note: 1. For INT0 and INT1, only level interrupt.

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, power-down, or standby) will be activated by the SLEEP instruction. See Table 8-2 on page 37 for a summary.

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

8.2 Idle Mode

When the SM1..0 bits are written to 00, the SLEEP instruction makes the MCU enter idle mode, stopping the CPU but allowing 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.

8.3 ADC Noise Reduction Mode

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

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart 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.

8.4 Power-down Mode

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

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 11. “External Interrupts” on page 49 for details.

8.5 Standby Mode

When the SM1..0 bits are written to 11 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter standby mode. This mode is identical to power-down with the exception that the oscillator is kept running. From standby mode, the device wakes up in six clock cycles.

8.6 Power Reduction Register

The power reduction register (PRR), see Section 8.8.2 “PRR – Power Reduction Register” on page 37, 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. See Section 24.3 “Supply Current of I/O Modules” on page 182 for examples. In all other sleep modes, the clock is already stopped.

8.7 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.

8.7.1 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 19. “ADC – Analog to Digital Converter” on page 132 for details on ADC operation.

8.7.2 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 18. “AC – Analog Comparator” on page 129 for details on how to configure the analog comparator.

8.7.3 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 9.5 “Brown-out Detection” on page 41 for details on how to configure the brown-out detector.

8.7.4 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 9.7 “Internal Voltage Reference” on page 42 for details on the start-up time.

8.7.5 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 9.8 “Watchdog Timer” on page 42 for details on how to configure the Watchdog Timer.

8.7.6 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 Section 12.2.5 “Digital Input Enable and Sleep Modes” on page 56 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 registers (DIDR0, DIDR1). Refer to Section 19.10.5 “DIDR0 – Digital Input Disable Register 0” on page 149 or Section 19.10.6 “DIDR1 – Digital Input Disable Register 1” on page 149 for details.

8.8 Register Description

8.8.1 MCUCR – MCU Control Register

The MCU control register contains control bits for power management.

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

- **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 8-2 on page 37.

Table 8-2. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	ADC noise reduction
1	0	Power-down
1	1	Standby

- **Bit 2 – Res: Reserved Bit**

This bit is a reserved ed bit in the Atmel® ATtiny261/461/861 and will always read as zero.

8.8.2 PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	–	-	-	-	**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 ATtiny261/461/861 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. Also analog comparator needs this clock.

9. System Control and Reset

9.1 Resetting the AVR

During reset, all I/O registers are set to their initial values, and the program starts execution from the reset vector. The instruction placed at the reset vector must be a RJMP – relative jump – instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 9-1 on page 39 shows the reset logic. See Section 23.5 “System and Reset Characteristics” on page 174 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.

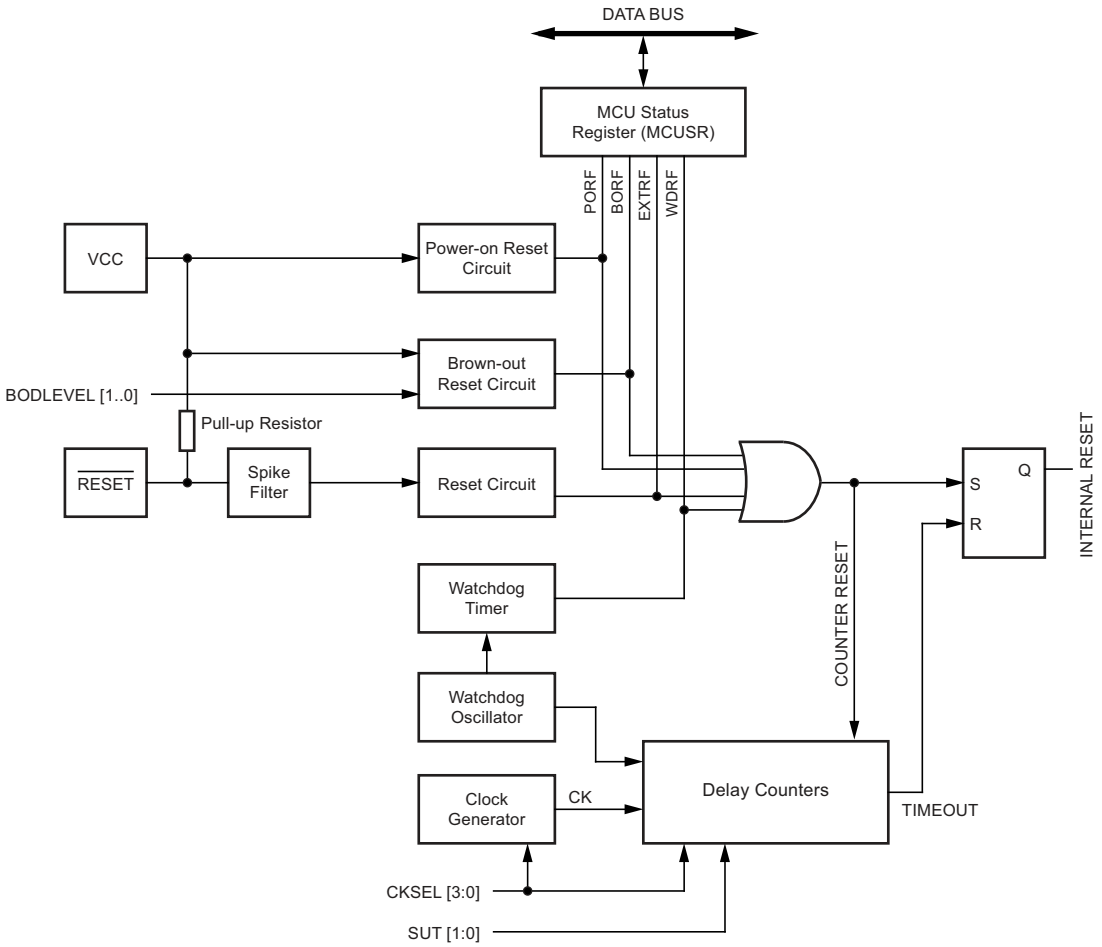
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 7.2 “Clock Sources” on page 26.

9.2 Reset Sources

The Atmel® ATtiny261/461/861 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 9-1. Reset Logic



9.3 Power-on Reset

A power-on reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in Section 23.5 “System and Reset Characteristics” on page 174. 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 9-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

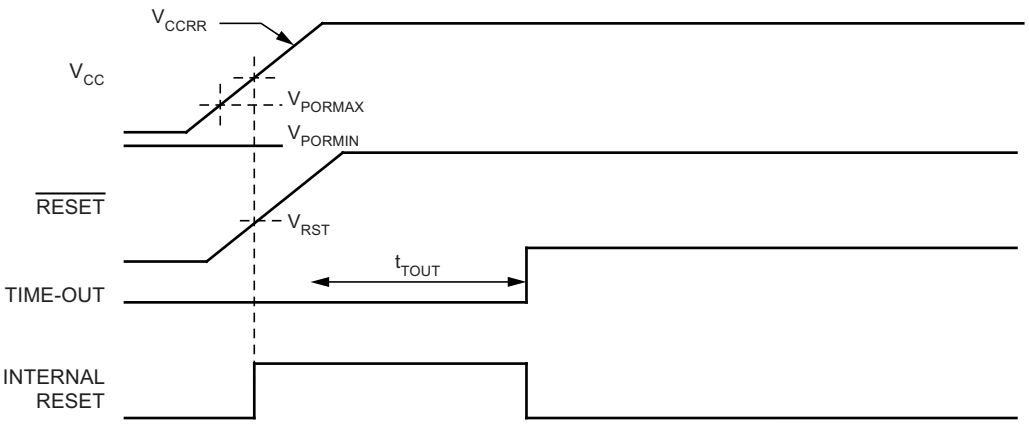


Figure 9-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally

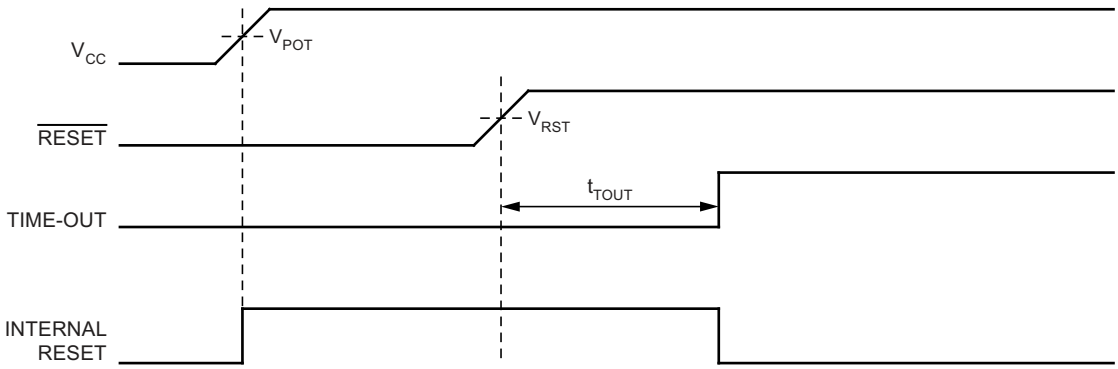


Table 9-1. Power On Reset Specifications

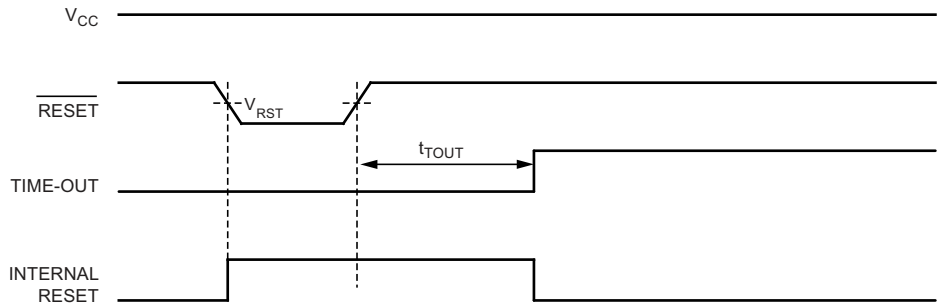
Parameter	Symbol	Min	Typ	Max	Unit
Power-on reset threshold voltage (rising)	V_{POT}	1.1	1.4	1.7	V
Power-on reset threshold voltage (falling)(1)		0.8	1.3	1.6	V
VCC max. start voltage to ensure internal power-on reset signal	V_{PORMAX}			0.4	V
VCC min. start voltage to ensure internal power-on reset signal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

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

9.4 External Reset

An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin if enabled. Reset pulses longer than the minimum pulse width (see Section 23.5 “System and Reset Characteristics” on page 174) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the reset threshold voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period – t_{TOUT} – has expired.

Figure 9-4. External Reset During Operation



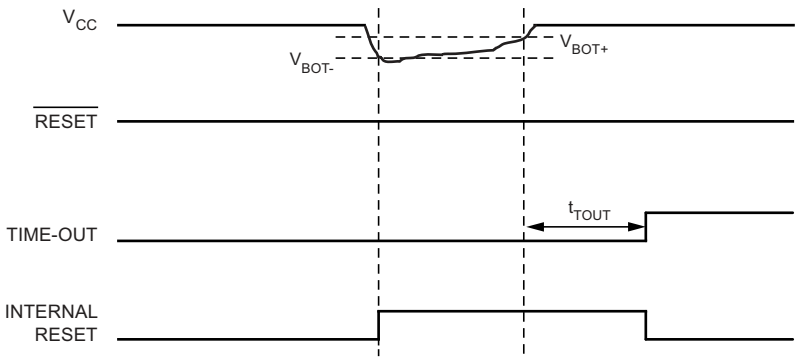
9.5 Brown-out Detection

Atmel® ATtiny261/461/861 has an on-chip brown-out detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL fuses. The trigger level has a hysteresis to ensure spike free brown-out detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} – V_{HYST}/2$.

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

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Section 23.5 “System and Reset Characteristics” on page 174.

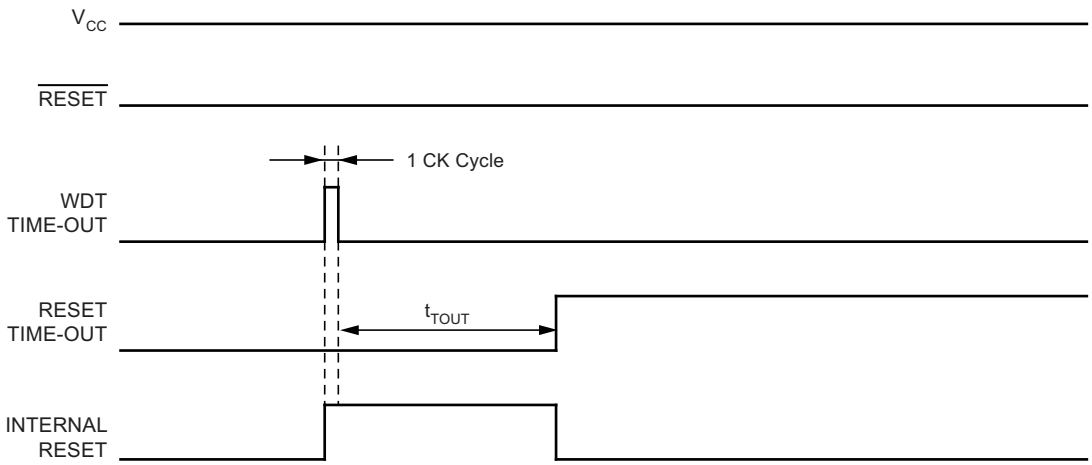
Figure 9-5. Brown-out Reset During Operation



9.6 Watchdog Reset

When the watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the time-out period t_{TOUT}. Refer to Section 9.8 “Watchdog Timer” on page 42 for details on operation of the watchdog timer.

Figure 9-6. Watchdog Reset During Operation



9.7 Internal Voltage Reference

Atmel® ATtiny261/461/861 features an internal bandgap reference. This reference is used for brown-out detection, and it can be used as an input to the analog comparator or the ADC.

9.7.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Section 23.5 “System and Reset Characteristics” on page 174. 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.

9.8 Watchdog Timer

The watchdog timer is clocked from an on-chip oscillator which runs at 128kHz. By controlling the watchdog timer prescaler, the watchdog reset interval can be adjusted as shown in Table 9-4 on page 45. 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 ATtiny261/461/861 resets and executes from the reset vector. For timing details on the watchdog reset, refer to Table 9-4 on page 45.

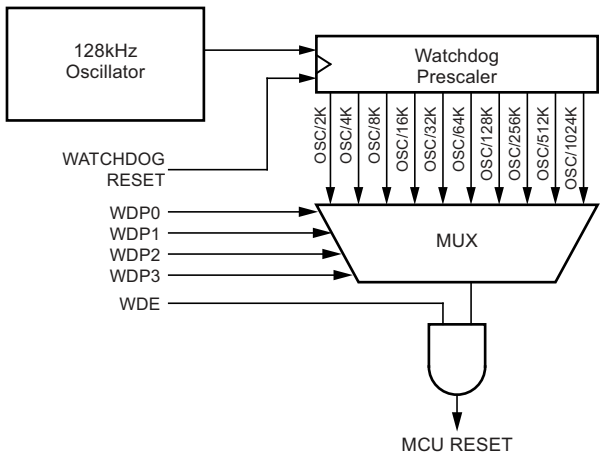
The watchdog timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the watchdog to wake-up from power-down.

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 9-2 on page 43 Refer to Section 9.9 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 43 for details.

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

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

Figure 9-7. Watchdog Timer



9.9 Timed Sequences for Changing the Configuration of the Watchdog Timer

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

9.9.1 Safety Level 1

In this mode, the watchdog timer is initially disabled, but can be enabled by writing the WDE bit to 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.

9.9.2 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.

9.10 Register Description

9.10.1 MCUSR – MCU Status Register

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

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

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny261/461/861 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.

9.10.2 WDTCR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
0x21 (0x41)	**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 9-3. Watchdog Timer Configuration

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

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

This bit must be set when the WDE bit is written to 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 9.9 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 43

- **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 9.9 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 43

In safety level 1, WDE is overridden by WDRF in MCUSR. See Section 9.10.1 “MCUSR – MCU Status Register” on page 44 for description of WDRF. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared before disabling the watchdog with the procedure described above. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

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

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

The WDP3..0 bits determine the watchdog timer prescaling when the watchdog timer is enabled. The different prescaling values and their corresponding time-out periods are shown in Table 9-4 on page 45.

Table 9-4. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16ms
0	0	0	1	4K (4096) cycles	32ms
0	0	1	0	8K (8192) cycles	64ms
0	0	1	1	16K (16384) cycles	0.125s
0	1	0	0	32K (32764) cycles	0.25s
0	1	0	1	64K (65536) cycles	0.5s
0	1	1	0	128K (131072) cycles	1.0s
0	1	1	1	256K (262144) cycles	2.0s
1	0	0	0	512K (524288) cycles	4.0s
1	0	0	1	1024K (1048576) cycles	8.0s

Note: 1. f selecting a reserved code WDT time-out is selected to be one of the legal selections.

Table 9-4. Watchdog Timer Prescale Select (Continued)

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
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. f selecting a reserved code WDT time-out is selected to be one of the legal selections.

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.

10. Interrupts

This section describes the specifics of the interrupt handling as performed in Atmel® ATtiny261/461/861. For a general explanation of the AVR® interrupt handling, refer to Section 5.7 “Reset and Interrupt Handling” on page 14.

10.1 Interrupt Vectors in ATtiny261/461/861

Table 10-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset, watchdog reset
2	0x0001	INT0	External interrupt request 0
3	0x0002	PCINT	Pin change interrupt request
4	0x0003	TIMER1_COMPA	Timer/Counter1 compare match A
5	0x0004	TIMER1_COMPB	Timer/Counter1 compare match B
6	0x0005	TIMER1_OVF	Timer/Counter1 overflow
7	0x0006	TIMER0_OVF	Timer/Counter0 overflow
8	0x0007	USI_START	USI start
9	0x0008	USI_OVF	USI overflow
10	0x0009	EE_RDY	EEPROM ready
11	0x000A	ANA_COMP	Analog comparator
12	0x000B	ADC	ADC conversion complete
13	0x000C	WDT	Watchdog time-out
14	0x000D	INT1	External interrupt request 1
15	0x000E	TIMER0_COMPA	Timer/Counter0 compare match A
16	0x000F	TIMER0_COMPB	Timer/Counter0 compare match B
17	0x0010	TIMER0_CAPT	Timer/Counter1 capture event
18	0x0011	TIMER1_COMPD	Timer/Counter1 compare match D
19	0x0012	FAULT_PROTECTION	Timer/Counter1 fault protection

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® ATtiny261/461/861 is:

```
Address      Labels Code                 Comments
0x0000              rjmp   RESET         ; Reset Handler
0x0001              rjmp   EXT_INT0      ; IRQ0 Handler
0x0002              rjmp   PCINT         ; PCINT Handler
0x0003              rjmp   TIM1_COMPA    ; Timer1 CompareA Handler
0x0004              rjmp   TIM1_COMPB    ; Timer1 CompareB Handler
0x0005              rjmp   TIM1_OVF      ; Timer1 Overflow Handler
0x0006              rjmp   TIM0_OVF      ; Timer0 Overflow Handler
0x0007              rjmp   USI_START     ; USI Start Handler
0x0008              rjmp   USI_OVF       ; USI Overflow Handler
0x0009              rjmp   EE_RDY        ; EEPROM Ready Handler
0x000A              rjmp   ANA_COMP      ; Analog Comparator Handler
0x000B              rjmp   ADC           ; ADC Conversion Handler
0x000C              rjmp   WDT           ; WDT Interrupt Handler
0x000D              rjmp   EXT_INT1      ; IRQ1 Handler
0x000E              rjmp   TIM0_COMPA    ; Timer0 CompareA Handler
0x000F              rjmp   TIM0_COMPB    ; Timer0 CompareB Handler
0x0010              rjmp   TIM0_CAPT     ; Timer0 Capture Event Handler
0x0011              rjmp   TIM1_COMPD    ; Timer1 CompareD Handler
0x0012              rjmp   FAULT_PROTECTION; Timer1 Fault Protection
0x0013       RESET: ldi    r16, low(RAMEND); Main program start
0x0014              ldi    r17, high(RAMEND); Tiny861 have also SPH
0x0015              out    SPL, r16      ; Set Stack Pointer to top of RAM
0x0016              out    SPH, r17      ; Tiny861 have also SPH
0x0017              sei                  ; Enable interrupts
0x0018       <instr> xxx
...    ...    ...    ...
```

11. External Interrupts

The external interrupts are triggered by the INT0 or INT1 pin or any of the PCINT15..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0, INT1 or PCINT15..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 PCINT15..0 pin toggles. The PCMSK register control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT15..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 and INT1 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 or INT1 requires the presence of an I/O clock, described in Section 7.1 “Clock Systems and their Distribution” on page 24. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt 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 7. “System Clock and Clock Options” on page 24.

11.1 Register Description

11.1.1 MCUCR – MCU Control Register

The MCU register contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	**–**	**PUD**	**SE**	**SM1**	**SM0**	**–**	**ISC01**	**ISC00**	**MCUCR**
Read/Write	R	R/W	R/W	R/W	R/W	R	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 or INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 or INT1 pin that activate the interrupt are defined in Table 11-1. The value on the INT0 or INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 11-1. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 or INT1 generates an interrupt request.
0	1	Any logical change on INT0 or INT1 generates an interrupt request.
1	0	The falling edge of INT0 or INT1 generates an interrupt request.
1	1	The rising edge of INT0 or INT1 generates an interrupt request.

11.1.2 GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	INT1	INT0	PCIE1	PCIE0	–	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R/w	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 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 INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of external interrupt request 1 is executed from the INT1 interrupt vector.

- **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 – PCIE1: Pin Change Interrupt Enable**

When the PCIE1 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 PCINT7..0 or PCINT15..12 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI interrupt vector. PCINT7..0 and PCINT15..12 pins are enabled individually by the PCMSK0 and PCMSK1 register.

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

When the PCIE0 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 PCINT11..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI interrupt vector. PCINT11..8 pins are enabled individually by the PCMSK1 register.

- **Bits 3..0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny261/461/861 and will always read as zero.

11.1.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	INT1	INTF0	PCIF	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 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 INT1 is configured as a level interrupt.

- **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 PCINT15 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.

- **Bits 4:0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny261/461/861 and will always read as zero.

11.1.4 PCMSK0 – Pin Change Mask Register A

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**PCINT7**	**PCINT6**	**PCINT5**	**PCINT4**	**PCINT3**	**PCINT2**	**PCINT1**	**PCINT0**	**PCMSK0**
Read/Write	R/W	R/W	R/W	R/w	R/W	R/W	R/W	R/W	
Initial Value	1	1	0	0	1	0	0	0	

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

Each PCINT7:0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.1.5 PCMSK1 – Pin Change Mask Register B

Bit	7	6	5	4	3	2	1	0	
0x22 (0x42)	**PCINT15**	**PCINT14**	**PCINT13**	**PCINT12**	**PCINT11**	**PCINT10**	**PCINT9**	**PCINT8**	**PCMSK1**
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	

- **Bits 7:0 – PCINT15:8: Pin Change Enable Mask 15..8**

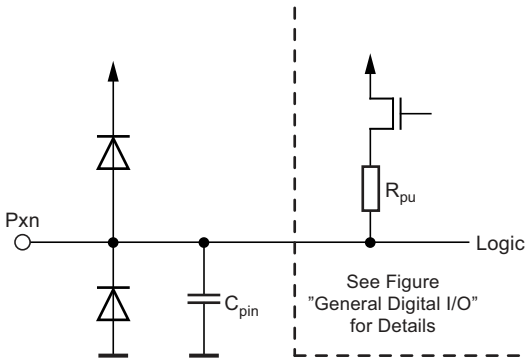
Each PCINT15:8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT11:8 is set and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin, and if PCINT15:12 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT15:8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

12. I/O Ports

12.1 Overview

All AVR® ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 12-1. Refer to Section 23. “Electrical Characteristics” on page 171 for a complete list of parameters.

Figure 12-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O registers and bit locations are listed in Section 12.4 “Register Description” on page 64.

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.

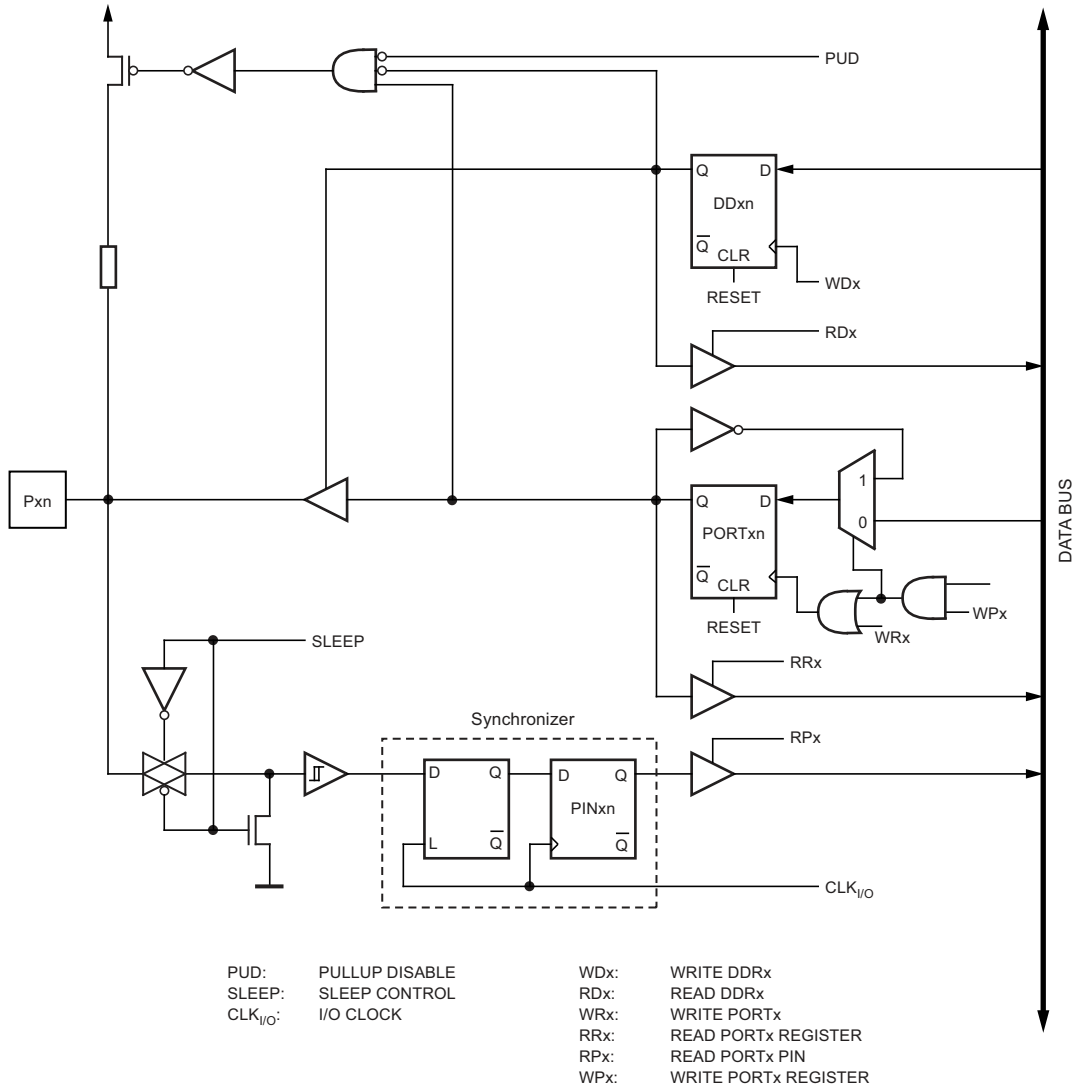
Using the I/O port as general digital I/O is described in Section 12.2 “Ports as General Digital I/O” on page 53. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 12.3 “Alternate Port Functions” on page 57. Refer to the individual module sections for a full description of the alternate functions.

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

12.2 Ports as General Digital I/O

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

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



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

12.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in Section 12.4 “Register Description” on page 64, 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).

12.2.2 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.

12.2.3 Switching Between Input and Output

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

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

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

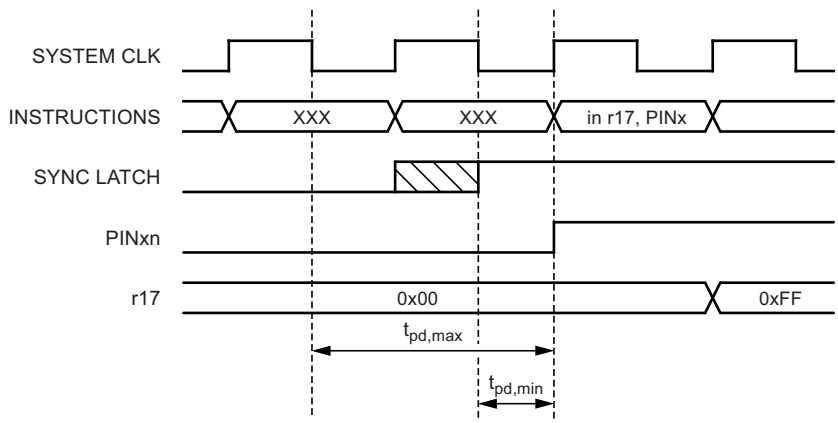
Table 12-1. Port Pin Configurations

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

12.2.4 Reading the Pin Value

Independent of the setting of data direction bit DDxn, the port pin can be read through the PINxn register bit. As shown in Figure 12-2 on page 53, the PINxn register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 12-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

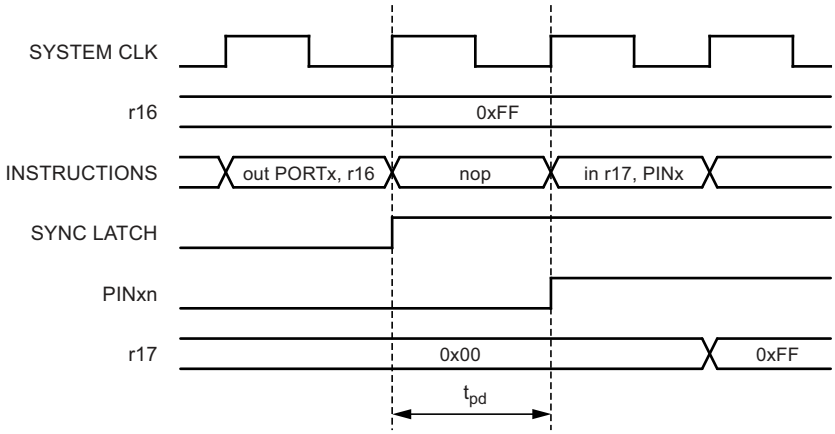
Figure 12-3. Synchronization when Reading an Externally Applied Pin Value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn register at the succeeding positive clock edge. As indicated by the two arrows 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 12-4. 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 12-4. 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.

12.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 12-2 on page 53, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 12.3 “Alternate Port Functions” on page 57.

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.

12.2.6 Unconnected Pins

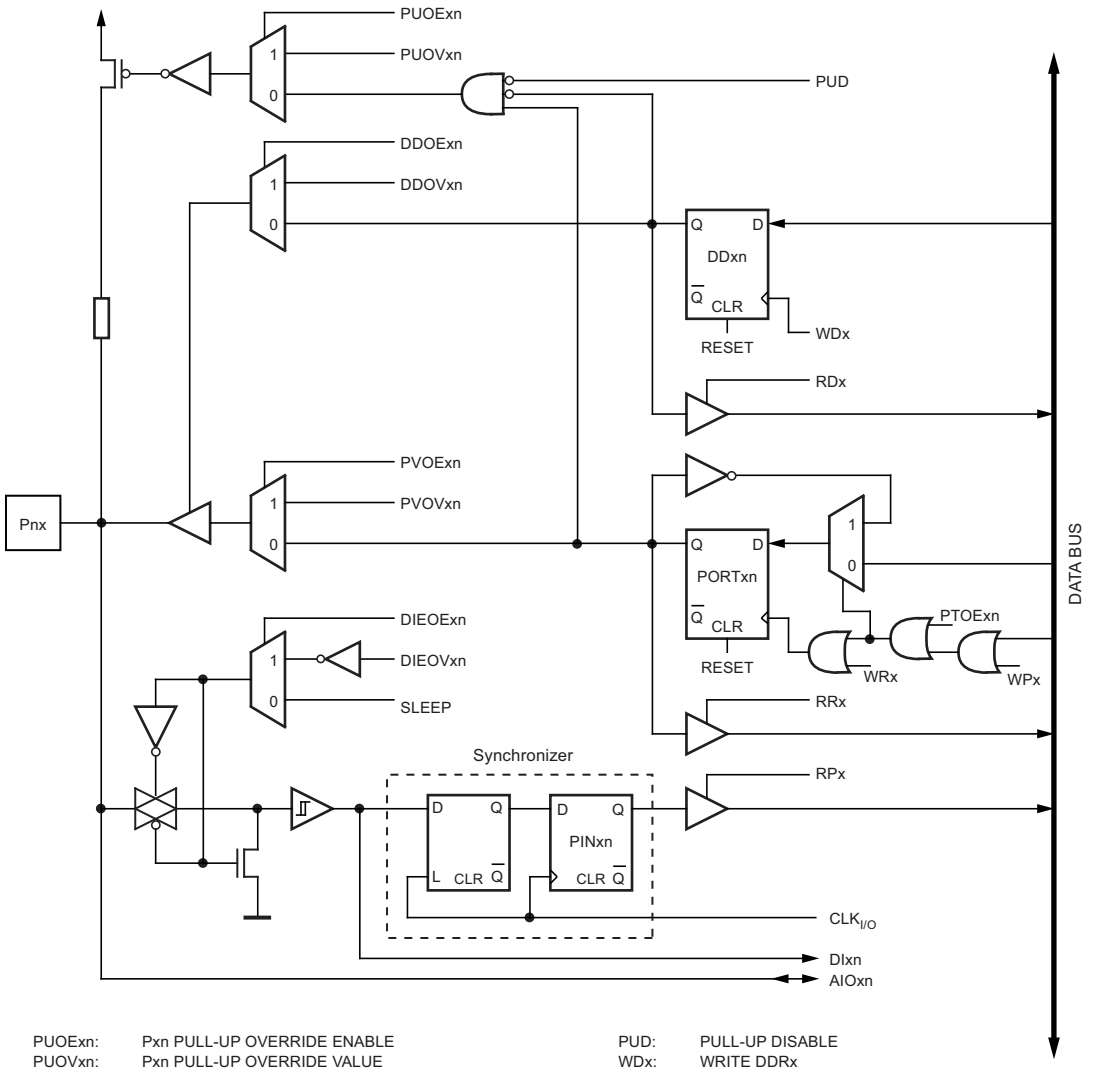
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

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

12.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 12-5 shows how the port pin control signals from the simplified Figure 12-2 on page 53 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 12-5. Alternate Port Functions(1)



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

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

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

Table 12-2. Generic Description of Overriding Signals for Alternate Functions

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

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

12.3.1 Alternate Functions of Port B

The port B pins with alternate function are shown in Table 12-3.

Table 12-3. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB7	$\overline{\text{RESET}}$ / dW / ADC10 / PCINT15
PB6	ADC9 / T0 / INT0 / PCINT14
PB5	XTAL2 / CLKO / OC1D / ADC8 / PCINT13
PB4	XTAL1 / CLKI / $\overline{\text{OC1D}}$ / ADC7 / PCINT12
PB3	OC1B / PCINT11
PB2	SCK / USCK / SCL / $\overline{\text{OC1B}}$ /PCINT10
PB1	MISO / DO / OC1A / PCINT9
PB0	MOSI / DI / SDA / $\overline{\text{OC1A}}$ / PCINT8

The alternate pin configuration is as follows:

- **Port B, Bit 7 - $\overline{\text{RESET}}$/ dW/ ADC10/ PCINT15**

RESET, reset pin: When the RSTDISBL fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on power-on reset and brown-out reset as its reset sources. When the RSTDISBL fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.

If PB7 is used as a reset pin, DDB7, PORTB7 and PINB7 will all read 0.

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

ADC10: ADC input channel 10. Note that ADC input channel 10 uses analog power.

PCINT15: Pin change interrupt source 15.

- **Port B, Bit 6 - ADC9/ T0/ INT0/ PCINT14**

ADC9: ADC input Channel 9. Note that ADC input channel 9 uses analog power.

T0: Timer/Counter0 counter source.

INT0: The PB6 pin can serve as an external interrupt source 0.

PCINT14: Pin change interrupt source 14.

- **Port B, Bit 5 - XTAL2/ CLKO/ ADC8/ PCINT13**

XTAL2: Chip clock oscillator pin 2. Used as clock pin for crystal oscillator or low-frequency crystal oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

CLKO: The divided system clock can be output on the PB5 pin, if the CKOUT fuse is programmed, regardless of the PORTB5 and DDB5 settings. It will also be output during reset.

OC1D output compare match output: The PB5 pin can serve as an external output for the Timer/Counter1 compare match D when configured as an output (DDA1 set). The OC1D pin is also the output pin for the PWM mode timer function.

ADC8: ADC input channel 8. Note that ADC input channel 8 uses analog power.

PCINT13: Pin Change Interrupt source 13.

- **Port B, Bit 4 - XTAL1/ CLKI/ OC1B/ ADC7/ PCINT12**

XTAL1/CLKI: 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.

$\overline{OC1D}$: Inverted output compare match output: The PB4 pin can serve as an external output for the Timer/Counter1 compare match D when configured as an output (DDA0 set). The $\overline{OC1D}$ pin is also the inverted output pin for the PWM mode timer function.

ADC7: ADC input channel 7. Note that ADC input channel 7 uses analog power.

PCINT12: Pin change interrupt source 12.

- **Port B, Bit 3 - OC1B/ PCINT11**

OC1B, output compare match output: The PB3 pin can serve as an external output for the Timer/Counter1 compare match B. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT11: Pin change interrupt source 11.

- **Port B, Bit 2 - SCK/ USCK/ SCL/ $\overline{OC1B}$/ PCINT10**

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 DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

USCK: Three-wire mode universal serial interface clock.

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

$\overline{OC1B}$: Inverted output compare match output: The PB2 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB2 set). The $\overline{OC1B}$ pin is also the inverted output pin for the PWM mode timer function.

PCINT10: Pin change interrupt source 10.

- **Port B, Bit 1 - MISO/ DO/ OC1A/ PCINT9**

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.

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).

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.

PCINT9: Pin change interrupt source 9.

- **Port B, Bit 0 - MOSI/ DI/ SDA/ $\overline{OC1A}$/ PCINT8**

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.

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.

SDA: Two-wire mode serial interface data.

$\overline{OC1A}$: 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 $\overline{OC1A}$ pin is also the inverted output pin for the PWM mode timer function.

PCINT8: Pin change interrupt source 8.

Table 12-4 and Table 12-5 relate the alternate functions of Port B to the overriding signals shown in Figure 12-5 on page 57.

Table 12-4. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/RESET/dW/ ADC10/PCINT15	PB6/ADC9/T0/INT0/ PCINT14	PB5/XTAL2/CLKO/ OC1D/ADC8/PCINT13(1)	PB4/XTAL1/OC1D/ADC7/PCINT12(1)
PUOE	$\overline{\text{RSTDISBL}}$(1) × DWEN(1)	0	$\overline{\text{INTRC}}$ × $\overline{\text{EXTCLK}}$	INTRC
PUOV	1	0	0	0
DDOE	RSTDISBL(1) × DWEN(1)	0	$\overline{\text{INTRC}}$ × $\overline{\text{EXTCLK}}$	INTRC
DDOV	debugWire transmit	0	0	0
PVOE	0	0	OC1D Enable	OC1D enable
PVOV	0	0	OC1D	OC1D
PTOE	0	0	0	0
DIEOE	0	$\overline{\text{RSTDISBL}}$ + (PCINT5 × PCIE + ADC9D)	INTRC × EXTCLK + PCINT4 × PCIE + ADC8D	$\overline{\text{INTRC}}$ + PCINT12 × PCIE + ADC7D
DIEOV	ADC10D	ADC9D	(INTRC × EXTCLK) + ADC8D	INTRC × ADC7D
DI	PCINT15	T0/INT0/PCINT14	PCINT13	PCINT12
AIO	RESET / ADC10	ADC9	XTAL2, ADC8	XTAL1, ADC7

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

Table 12-5. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/OC1B/ PCINT11	PB2/SCK/USCK/SCL/ OC1B/PCINT10	PB1/MISO/DO/OC1A/ PCINT9	PB0/MOSI/DI/SDA/ OC1A/PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	USI_TWO_WIRE × $\overline{\text{USIPOS}}$	0	USI_TWO_WIRE × $\overline{\text{USIPOS}}$
DDOV	0	(USI_SCL_HOLD + $\overline{\text{PORTB2}}$) × DDB2 × $\overline{\text{USIPOS}}$	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB0}}$) × DDB0 × $\overline{\text{USIPOS}}$
PVOE	OC1B enable	$\overline{\text{OC1B Enable}}$ + $\overline{\text{USIPOS}}$ × USI_TWO_WIRE × DDB2	OC1A Enable + $\overline{\text{USIPOS}}$ × USI_THREE_WIRE	$\overline{\text{OC1A Enable}}$ + (USI_TWO_WIRE × DDB0 × $\overline{\text{USIPOS}}$)
PVOV	OC1B	OC1B	OC1A + (DO × $\overline{\text{USIPOS}}$)	OC1A
PTOE	0	USITC × $\overline{\text{USIPOS}}$	0	0
DIEOE	PCINT11 × PCIE	PCINT10 × PCIE + USISIE × $\overline{\text{USIPOS}}$	PCINT9 × PCIE	PCINT8 × PCIE + (USISIE × $\overline{\text{USIPOS}}$)
DIEOV	0	0	0	0
DI	PCINT11	USCK/SCL/PCINT10	PCINT9	DI/SDA/PCINT8
AIO				

Note: INTRC means that one of the internal RC oscillators are selected (by the CKSEL fuses), EXTCK means that external clock is selected (by the CKSEL fuses).

12.3.2 Alternate Functions of Port A

The port A pins with alternate function are shown in Table 12-6.

Table 12-6. Port B Pins Alternate Functions

Port Pin	Alternate Function
PA7	ADC6 / AIN0 / PCINT7
PA6	ADC5 / AIN1 / PCINT6
PA5	ADC4 / AIN2 / PCINT5
PA4	ADC3 /ICP0/ PCINT4
PA3	AREF / PCINT3
PA2	ADC2 / INT1 / USCK / SCL / PCINT2
PA1	ADC1 / DO / PCINT1
PA0	ADC0 / DI / SDA / PCINT0

The alternate pin configuration is as follows:

- **Port A, Bit 7- ADC6/AIN0/PCINT7**

ADC6: Analog to digital converter, channel 6.

AIN0: Analog comparator 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.

PCINT7: Pin change interrupt source 8.

- **Port A, Bit 6 - ADC5/AIN1/PCINT6**

ADC5: Analog to digital converter, channel 5.

AIN1: Analog comparator 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.

PCINT6: Pin change interrupt source 6.

- **Port A, Bit 5 - ADC4/AIN2/PCINT5**

ADC4: Analog to digital converter, channel 4.

AIN2: Analog comparator 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.

PCINT5: Pin change interrupt source 5.

- **Port A, Bit 4 - ADC3/ICP0/PCINT4**

ADC3: Analog to digital converter, channel 3.

ICP0: Timer/Counter0 input capture pin.

PCINT4: Pin change interrupt source 4.

- **Port A, Bit 3 - AREF/PCINT3**

AREF: External analog reference for ADC. Pull-up and output driver are disabled on PA3 when the pin is used as an external reference or internal voltage reference with external capacitor at the AREF pin.

PCINT3: Pin change interrupt source 3.

- **Port A, Bit 2 - ADC2/INT1/USCK/SCL/PCINT2**

ADC2: Analog to digital converter, channel 2.

INT1: The PA2 pin can serve as an external interrupt source 1.

USCK: Three-wire mode universal serial interface clock.

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

PCINT2: Pin change interrupt source 2.

- **Port A, Bit 1 - ADC1/DO/PCINT1**

ADC1: Analog to digital converter, channel 1.

DO: Three-wire mode universal serial interface data output. Three-wire mode data output overrides PORTA1 value and it is driven to the port when data direction bit DDA1 is set. PORTA1 still enables the pull-up, if the direction is input and PORTA1 is set.

PCINT1: Pin change interrupt source 1.

- **Port A, Bit 0 - ADC0/DI/SDA/PCINT0**

ADC0: Analog to digital converter, channel 0.

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.

SDA: Two-wire mode serial interface data.

PCINT0: Pin change interrupt source 0.

Table 12-7 and Table 12-8 relate the alternate functions of port A to the overriding signals shown in Figure 12-5 on page 57.

Table 12-7. Overriding Signals for Alternate Functions in PA7..PA4

Signal Name	PA7/ADC6/AIN0/ PCINT7	PA6/ADC5/AIN1/ PCINT6	PA5/ADC4/AIN2/ PCINT5	PA4/ADC3/ICP0/ PCINT4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
PTOE	0	0	0	0
DIEOE	PCINT7 × PCIE + ADC6D	PCINT6 × PCIE + ADC5D	PCINT5 × PCIE + ADC4D	PCINT4 × PCIE + ADC3D
DIEOV	ADC6D	ADC5D	ADC4D	ADC3D
DI	PCINT7	PCINT6	PCINT5	ICP0/PCINT4
AIO	ADC6, AIN0	ADC5, AIN1	ADC4, AIN2	ADC3

Table 12-8. Overriding Signals for Alternate Functions in PA3..PA0

Signal Name	PA3/AREF/ PCINT3	PA2/ADC2/INT1/ USCK/SCL/PCINT2	PA1/ADC1/DO/ PCINT1	PA0/ADC0/DI/SDA/ PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	USI_TWO_WIRE × USIPOS	0	USI_TWO_WIRE × USIPOS
DDOV	0	(USI_SCL_HOLD + $\overline{\text{PORTB2}}$) × DDB2 × USIPOS	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB0}}$) × DDRB0 × USIPOS
PVOE	0	USI_TWO_WIRE × DDRB2	USI_THREE_WIRE × USIPOS	USI_TWO_WIRE × DDRB0 × USIPOS
PVOV	0	0	DO × USIPOS	0
PTOE	0	USI_PTOE × USIPOS	0	0
DIEOE	PCINT3 × PCIE	PCINT2 × PCIE + INT1 + ADC2D + USISIE × USIPOS	PCINT1 × PCIE + ADC1D	PCINT0 × PCIE + ADC0D + USISIE × USIPOS
DIEOV	0	ADC2D	ADC1D	ADC0D
DI	PCINT3	USCK/SCL/INT1/ PCINT2	PCINT1	DI/SDA/PCINT0
AIO	AREF	ADC2	ADC1	ADC0

12.4 Register Description

12.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	**-**	**PUD**	**SE**	**SM1**	**SM0**	**-**	**ISC01**	**ISC00**	**MCUCR**
Read/Write	R	R/W	R/W	R/W	R/W	R	R	R	
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 12.2.1 “Configuring the Pin” on page 53 for more details about this feature.

12.4.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	**PORTA7**	**PORTA6**	**PORTA5**	**PORTA4**	**PORTA3**	**PORTA2**	**PORTA1**	**PORTA0**	**PORTA**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.4.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	**DDA7**	**DDA6**	**DDA5**	**DDA4**	**DDA3**	**DDA2**	**DDA1**	**DDA0**	**DDRA**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.4.4 PINA – Port A Input Pins Address

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

12.4.5 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	**PORTB7**	**PORTB6**	**PORTB5**	**PORTB4**	**PORTB3**	**PORTB2**	**PORTB1**	**PORTB0**	**PORTB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.4.6 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	**DDB7**	**DDB6**	**DDB5**	**DDB4**	**DDB3**	**DDB2**	**DDB1**	**DDB0**	**DDRB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.4.7 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	**PINB7**	**PINB6**	**PINB5**	**PINB4**	**PINB3**	**PINB2**	**PINB1**	**PINB0**	**PINB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

13. Timer/Counter0 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 $f_{CLK\_I/O}/1024$. See Table 13-1 on page 68 for details.

13.1 Prescaler Reset

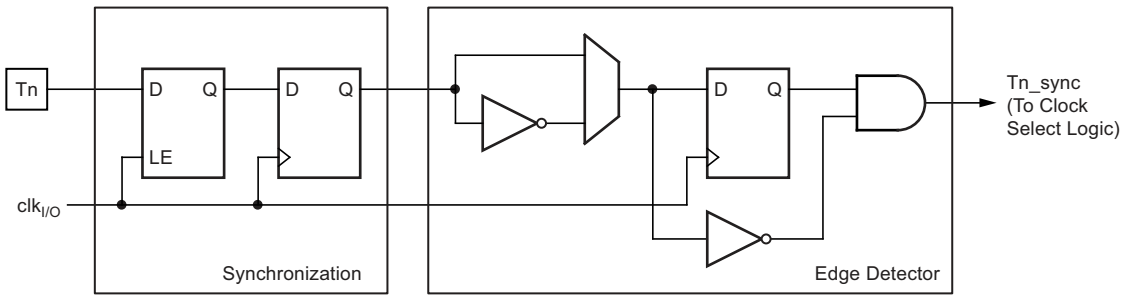
The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling 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.

13.2 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 13-1 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. See Table 13-1 on page 68 for details.

Figure 13-1. 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.

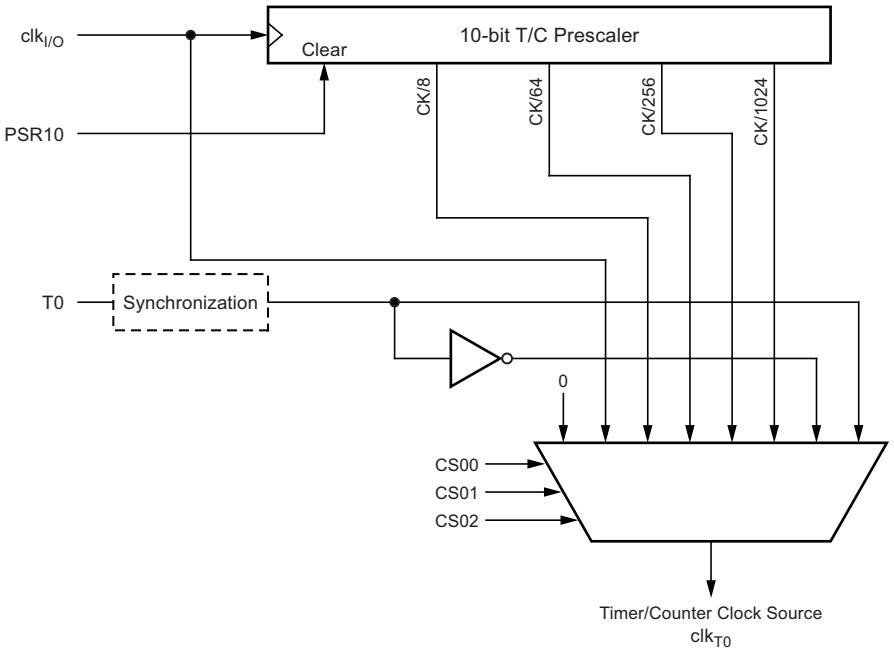
Enabling and disabling of the clock input must be done when T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

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 13-2. Prescaler for Timer/Counter0



Note: 1. The synchronization logic on the input pins (T0) is shown in Figure 13-1.

13.3 Register Description

13.3.1 TCCR0B – Timer/Counter0 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	-	-	-	TSM	PSR0	CS02	CS01	CS01	TCCR0B
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	

- **Bit 4 – 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 3 – 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.

- **Bits 2, 1, 0 – CS02, CS01, CS00: Clock Select0, Bit 2, 1, and 0**

The clock select0 bits 2, 1, and 0 define the prescaling source of timer0.

Table 13-1. 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.

14. Timer/Counter0

14.1 Features

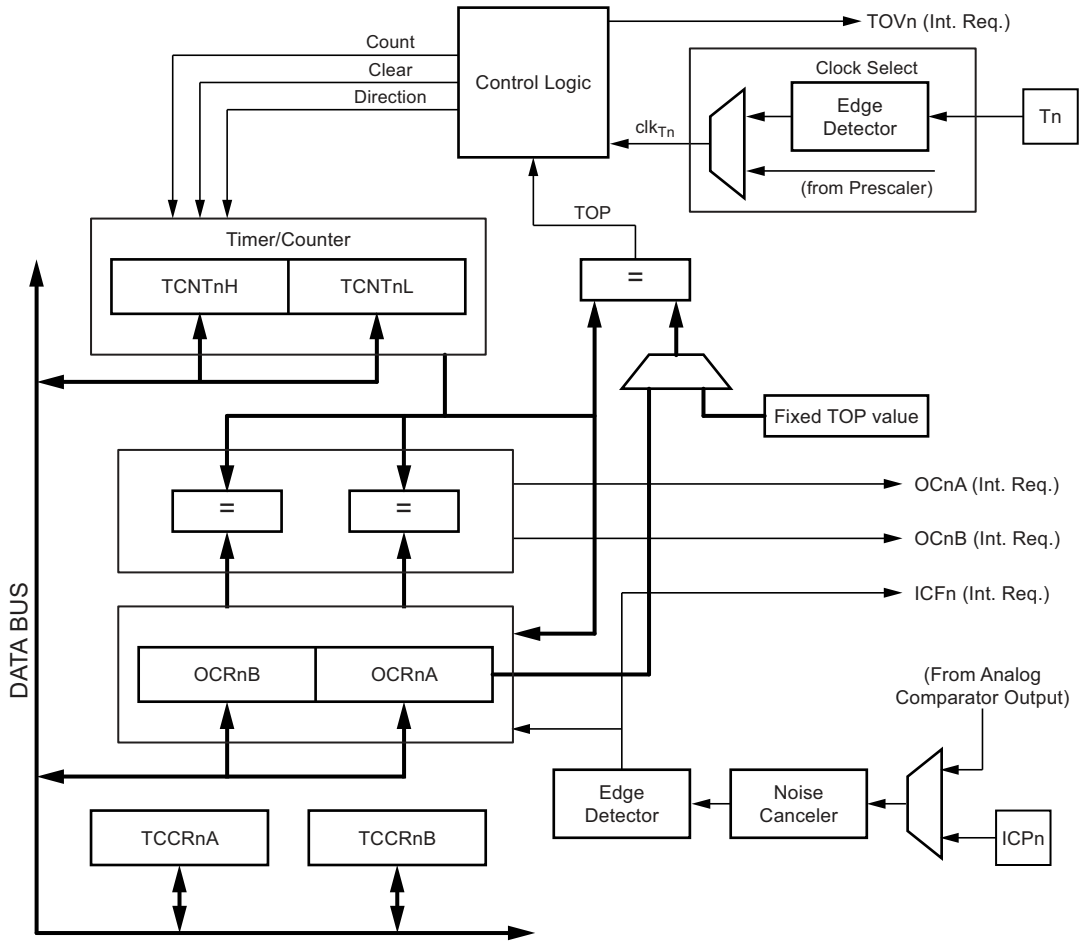
- Clear timer on compare match (auto reload)
- Input capture unit
- Four independent interrupt sources (TOV0, OCF0A, OCF0B, ICF0)
- 8-bit mode with two independent output compare units
- 16-bit mode with one independent output compare unit

14.2 Overview

Timer/Counter0 is a general purpose 8-/16-bit Timer/Counter module, with two/one output compare units and input capture feature.

The Timer/Counter0 general operation is described in 8-/16-bit mode. A simplified block diagram of the 8-/16-bit Timer/Counter is shown in Figure 14-1. For the actual placement of I/O pins, refer to Section 1-1 “Pinout ATtiny261/461/861” on page 3. CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in the Section 14.10 “Register Description” on page 80.

Figure 14-1. 8-/16-bit Timer/Counter Block Diagram



14.2.1 Registers

The Timer/Counter0 low byte register (TCNT0L) and output compare registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in Figure 14-1) 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.

In 16-bit mode the Timer/Counter consists one more 8-bit register, the Timer/Counter0 high byte register (TCNT0H). Furthermore, there is only one output compare unit in 16-bit mode as the two output compare registers, OCR0A and OCR0B, are combined to one 16-bit output compare register.

OCR0A contains the low byte of the word and OCR0B contains the high byte of the word. When accessing 16-bit registers, special procedures described in Section 14.9 “Accessing Registers in 16-bit Mode” on page 76 must be followed.

14.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the output compare unit, in this case compare unit A or compare unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0L for accessing Timer/Counter0 counter value and so on.

The definitions in Table 14-1 are also used extensively throughout the document.

Table 14-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255) in 8-bit mode or 0xFFFF (decimal 65535) in 16-bit mode.
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/0xFFFF (MAX) or the value stored in the OCR0A register.

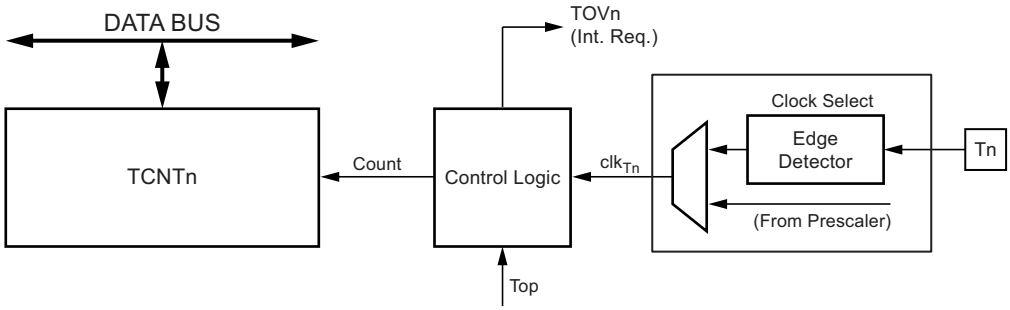
14.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The clock select logic is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter control register 0 B (TCCR0B), and controls which clock source and edge the Timer/Counter uses to increment 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}). For details on clock sources and prescaler, see Section 13. “Timer/Counter0 Prescaler” on page 66.

14.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 14-3 shows a block diagram of the counter and its surroundings.

Table 14-2. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT0 by 1.

clk_{Tn} Timer/Counter clock, referred to as clk_{T0} in the following.

top Signalize that TCNT0 has reached maximum value.

The counter is incremented at each timer clock (clk_{T0}) until it passes its TOP value and then restarts from BOTTOM. The counting sequence is determined by the setting of the CTC0 bit located in the Timer/Counter control register (TCCR0A). For more details about counting sequences, see Section 14.5 “Modes of Operation” on page 71. 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 Timer/Counter overflow flag (TOV0) is set when the counter reaches the maximum value and it can be used for generating a CPU interrupt.

14.5 Modes of Operation

The mode of operation is defined by the Timer/Counter width (TCW0), input capture enable (ICEN0) and wave generation mode (CTC0) bits in Section 14.10.1 “TCCR0A – Timer/Counter0 Control Register A” on page 80. Table 14-3 shows the different modes of operation.

Table 14-3. Modes of Operation

Mode	ICEN0	TCW0	CTC0	Mode of Operation	TOP Value	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal 8-bit mode	0xFF	Immediate	MAX (0xFF)
1	0	0	1	8-bit CTC	OCR0A	Immediate	MAX (0xFF)
2	0	1	X	16-bit mode	0xFFFF	Immediate	MAX (0xFFFF)
3	1	0	X	8-bit input capture mode	0xFF	Immediate	MAX (0xFF)
4	1	1	X	16-bit input capture mode	0xFFFF	Immediate	MAX (0xFFFF)

14.5.1 Normal 8-bit Mode

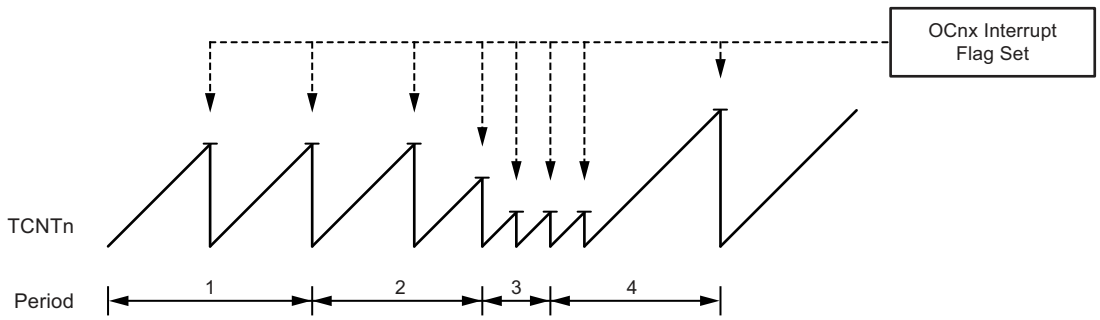
In the normal 8-bit mode, see Table 14-3 on page 71, the counter (TCNT0L) is incrementing until it overruns when it passes its maximum 8-bit value (MAX = 0xFF) and then restarts from the bottom (0x00). The overflow flag (TOV0) will be set in the same timer clock cycle as the TCNT0L becomes zero. The 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 8-bit mode, a new counter value can be written anytime. The output compare unit can be used to generate interrupts at some given time.

14.5.2 Clear Timer on Compare Match (CTC) 8-bit Mode

In clear timer on compare or CTC mode, see Table 14-3 on page 71, 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 14-2. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 14-2. 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. 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.

14.5.3 16-bit Mode

In 16-bit mode, see Table 14-3 on page 71, the counter (TCNT0H/L) is a incrementing until it overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the bottom (0x0000). The overflow flag (TOV0) will be set in the same timer clock cycle as the TCNT0H/L becomes zero. The TOV0 flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 flag, the timer resolution can be increased by software.

There are no special cases to consider in the normal mode, a new counter value can be written anytime. The output compare unit can be used to generate interrupts at some given time.

14.5.4 8-bit Input Capture Mode

The Timer/Counter0 can also be used in an 8-bit input capture mode, see Table 14-3 on page 71 for bit settings. For full description, see Section 14.6 “Input Capture Unit” on page 73.

14.5.5 16-bit Input Capture Mode

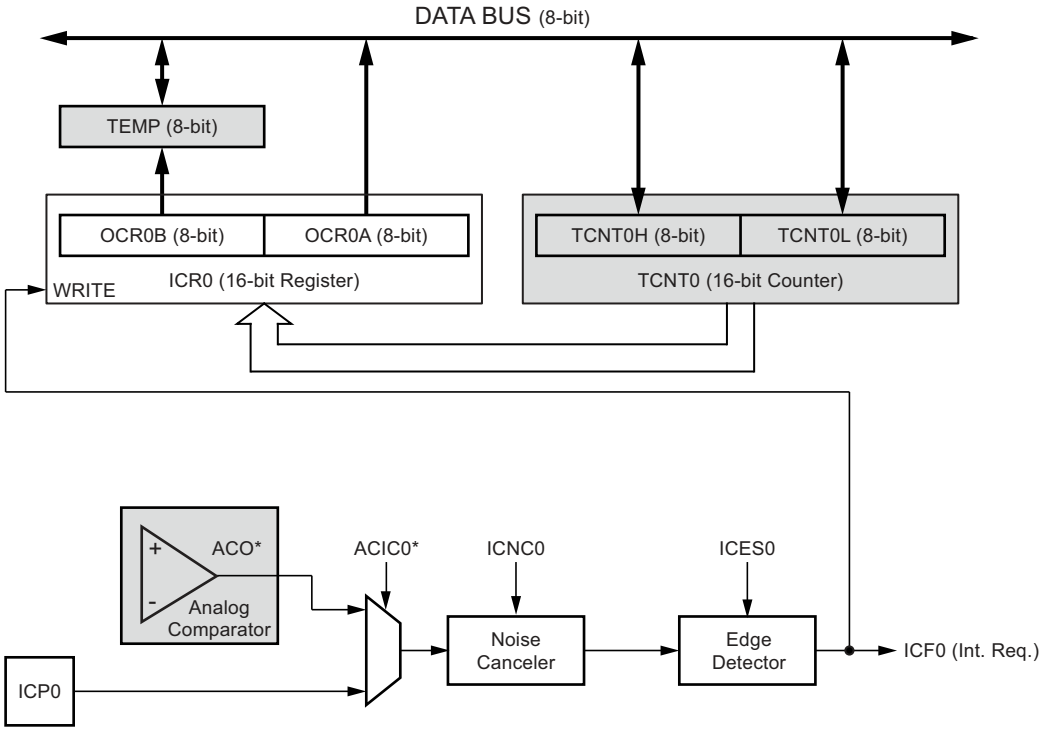
The Timer/Counter0 can also be used in a 16-bit input capture mode, see Table 14-3 on page 71 for bit settings. For full description, see the Section 14.6 “Input Capture Unit” on page 73.

14.6 Input Capture Unit

The Timer/Counter incorporates an input capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP0 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The input capture unit is illustrated by the block diagram shown in Figure 14-3. The elements of the block diagram that are not directly a part of the input capture unit are gray shaded.

Figure 14-3. Input Capture Unit Block Diagram



The output compare register OCR0A is a dual-purpose register that is also used as an 8-bit input capture register ICR0. In 16-bit input capture mode the output compare register OCR0B serves as the high byte of the input capture register ICR0. In 8-bit input capture mode the output compare register OCR0B is free to be used as a normal output compare register, but in 16-bit input capture mode the output compare unit cannot be used as there are no free output compare register(s). Even though the input capture register is called ICR0 in this section, it is referring to the output compare register(s).

When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP0), alternatively on the *Analog Comparator Output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the value of the counter (TCNT0) is written to the *Input Capture Register* (ICR0). The *Input Capture Flag* (ICF0) is set at the same system clock as the TCNT0 value is copied into input capture register. If enabled (TICIE0=1), the input capture flag generates an input capture interrupt. The ICF0 flag is automatically cleared when the interrupt is executed. Alternatively the ICF0 flag can be cleared by software by writing a logical one to its I/O bit location.

14.6.1 Input Capture Trigger Source

The default trigger source for the input capture unit is the *Input Capture pin* (ICP0). Timer/Counter0 can alternatively use the analog comparator output as trigger source for the input capture unit. The analog comparator is selected as trigger source by setting the *Analog Comparator Input Capture Enable* (ACIC0) bit in the *Timer/Counter Control Register A* (TCCR0A). Be aware that changing trigger source can trigger a capture. The input capture flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP0) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T0 pin (Figure 13-1 on page 68). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. An input capture can also be triggered by software by controlling the port of the ICP0 pin.

14.6.2 Noise Canceler

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

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

14.6.3 Using the Input Capture Unit

The main challenge when using the input capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR0 register before the next event occurs, the ICR0 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the input capture interrupt, the ICR0 register should be read as early in the interrupt handler routine as possible. The maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

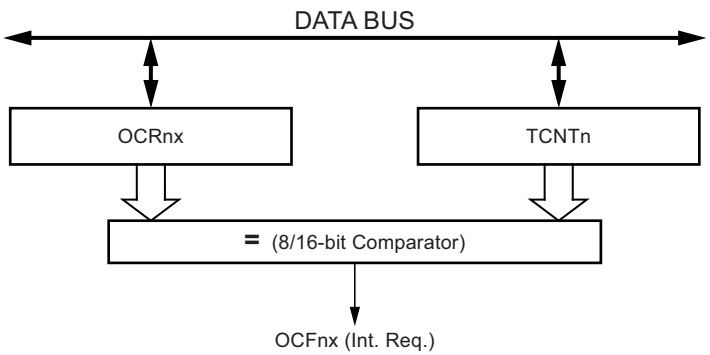
Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR0 register has been read. After a change of the edge, the input capture flag (ICF0) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the trigger edge change is not required (if an interrupt handler is used).

14.7 Output Compare Unit

The comparator continuously compares Timer/Counter (TCNT0) with the output compare registers (OCR0A and OCR0B), and whenever the Timer/Counter equals to the output compare registers, the comparator signals a match. A match will set the output compare flag at the next timer clock cycle. In 8-bit mode the match can set either the output compare flag OCF0A or OCF0B, but in 16-bit mode the match can set only the output compare flag OCF0A as there is only one output compare unit. 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. Figure 14-4 shows a block diagram of the output compare unit.

Figure 14-4. Output Compare Unit, Block Diagram



14.7.1 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0H/L register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0A/B to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

14.7.2 Using the Output Compare Unit

Since writing TCNT0H/L will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0H/L when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0H/L equals the OCR0A/B value, the compare match will be missed.

14.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set. Figure 14-5 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value.

Figure 14-5. Timer/Counter Timing Diagram, no Prescaling

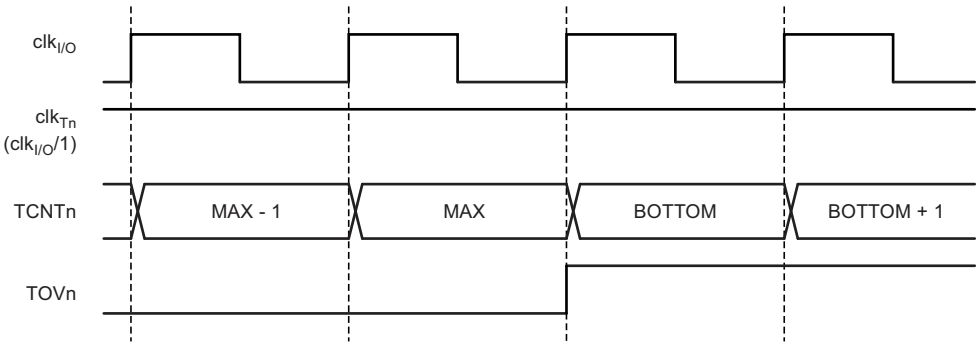


Figure 14-6 shows the same timing data, but with the prescaler enabled.

Figure 14-6. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

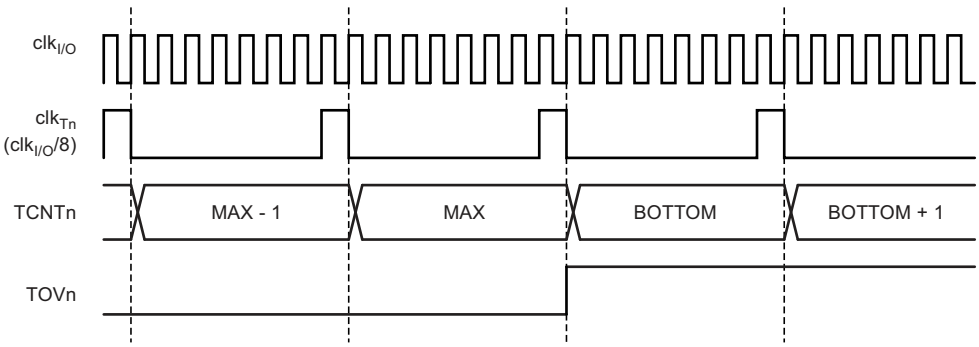


Figure 14-7 shows the setting of OCF0A and OCF0B in normal mode.

Figure 14-7. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

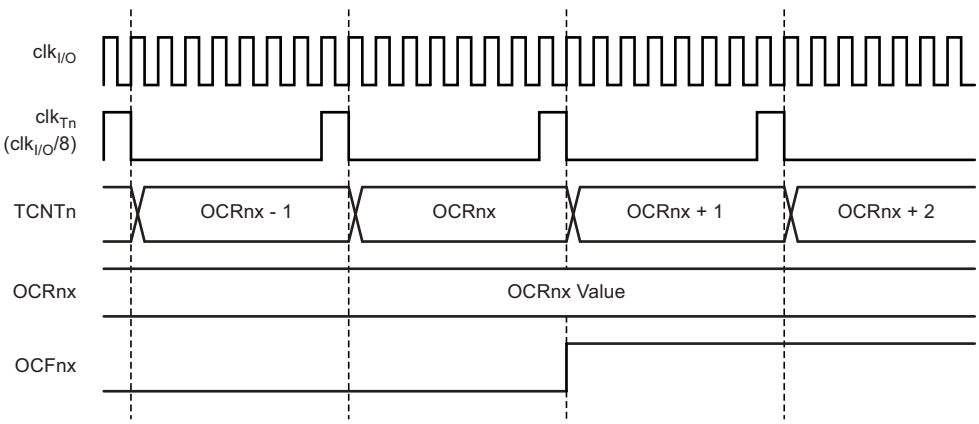
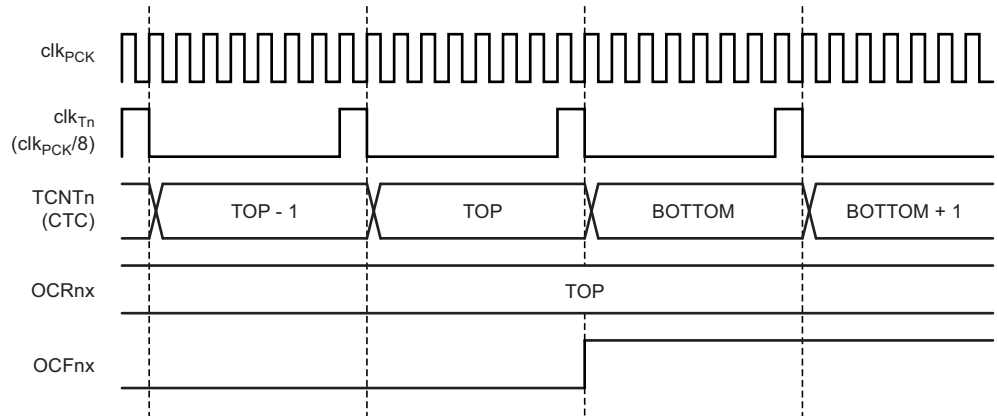


Figure 14-8 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode.

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



14.9 Accessing Registers in 16-bit Mode

In 16-bit mode (the TCW0 bit is set to one) the TCNT0H/L and OCR0A/B or TCNT0L/H and OCR0B/A are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. The 16-bit Timer/Counter has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

There is one exception in the temporary register usage. In the output compare mode the 16-bit output compare register OCR0A/B is read without the temporary register, because the output compare register contains a fixed value that is only changed by CPU access. However, in 16-bit Input Capture mode the ICR0 register formed by the OCR0A and OCR0B registers must be accessed with the temporary register.

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

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR0A/B registers.

Assembly Code Example

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

C Code Example

```
unsigned int i;
...
/* Set TCNT0 to 0x01FF */
TCNT0H = 0x01;
TCNT0L = 0xff;

/* Read TCNT0 into i */
i = TCNT0L;
i |= ((unsigned int)TCNT0H << 8);
...
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNT0H/L value in the r17:r16 register pair.

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

The following code examples show how to do an atomic read of the TCNT0 register contents. Reading any of the OCR0 register can be done by using the same principle.

.

Assembly Code Example

```
TIM0_ReadTCNT0:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Read TCNT0 into r17:r16
        in      r16,TCNT0L
        in      r17,TCNT0H
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

C Code Example

```
unsigned int TIM0_ReadTCNT0( void )
{
unsigned char sreg;
unsigned int i;
/* Save global interrupt flag */
sreg = SREG;
/* Disable interrupts */
_CLI();
/* Read TCNT0 into i */
i = TCNT0L;
i |= ((unsigned int)TCNT0H << 8);
/* Restore global interrupt flag */
SREG = sreg;
return i;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNT0H/L value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNT0H/L register contents. Writing any of the OCR0A/B registers can be done by using the same principle.

```
Assembly Code Example
TIM0_WriteTCNT0:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Set TCNT0 to r17:r16
        out     TCNT0H,r17
        out     TCNT0L,r16
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

```
C Code Example
void TIM0_WriteTCNT0(unsigned int i)
{
unsigned char sreg;
/* Save global interrupt flag */
sreg = SREG;
/* Disable interrupts */
_CLI();
/* Set TCNT0 to i */
TCNT0H = (i >> 8);
TCNT0L = (unsigned char)i;
/* Restore global interrupt flag */
SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT0H/L.

14.9.1 Reusing the temporary high byte register

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

14.10 Register Description

14.10.1 TCCR0A – Timer/Counter0 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	TCW0	ICEN0	ICNC0	ICES0	ACIC0	–	–	CTC0	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– TCW0: Timer/Counter0 Width**

When this bit is written to one 16-bit mode is selected as described Figure 14-5 on page 75. Timer/Counter0 width is set to 16-bits and the output compare registers OCR0A and OCR0B are combined to form one 16-bit output compare register. Because the 16-bit registers TCNT0H/L and OCR0B/A are accessed by the AVR CPU via the 8-bit data bus, special procedures must be followed. These procedures are described in Section 14.9 “Accessing Registers in 16-bit Mode” on page 76.

- **Bit 6– ICEN0: Input Capture Mode Enable**

When this bit is written to one, the input capture mode is enabled.

- **Bit 5 – ICNC0: Input Capture Noise Canceler**

Setting this bit activates the input capture noise canceler. When the noise canceler is activated, the input from the input capture pin (ICP0) is filtered. The filter function requires four successive equal valued samples of the ICP0 pin for changing its output. The input capture is therefore delayed by four system clock cycles when the noise canceler is enabled.

- **Bit 4 – ICES0: Input Capture Edge Select**

This bit selects which edge on the input capture pin (ICP0) that is used to trigger a capture event. When the ICES0 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES0 bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICES0 setting, the counter value is copied into the input capture register. The event will also set the input capture flag (ICF0), and this can be used to cause an input capture interrupt, if this interrupt is enabled.

- **Bit 3 - ACIC0: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter0 to be triggered by the analog comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter0 input capture interrupt. When written logic zero, no connection between the analog comparator and the input capture function exists. To make the comparator trigger the Timer/Counter0 input capture interrupt, the TICIE0 bit in the timer interrupt mask register (TIMSK) must be set.

- **Bits 2:1 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny261/461/861 and will always read as zero.

- **Bit 0 – CTC0: Waveform Generation Mode**

This bit controls the counting sequence of the counter, the source for maximum (TOP) counter value, see Figure 14-5 on page 75.

Modes of operation supported by the Timer/Counter unit are: normal mode (counter) and clear timer on compare match (CTC) mode (see Section 14.5 “Modes of Operation” on page 71).

14.10.2 TCNT0L – Timer/Counter0 Register Low Byte

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0L[7:0]								TCNT0L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter0 register low byte, TCNT0L, gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0L register blocks (disables) the compare match on the following timer clock. Modifying the counter (TCNT0L) while the counter is running, introduces a risk of missing a compare match between TCNT0L and the OCR0x registers. In 16-bit mode the TCNT0L register contains the lower part of the 16-bit Timer/Counter0 register.

14.10.3 TCNT0H – Timer/Counter0 Register High Byte

Bit	7	6	5	4	3	2	1	0	
0x14 (0x34)	TCNT0H[7:0]								TCNT0H
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 16-bit mode is selected (the TCW0 bit is set to one) the Timer/Counter register TCNT0H combined to the Timer/Counter register TCNT0L gives direct access, both for read and write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 14.9 “Accessing Registers in 16-bit Mode” on page 76

14.10.4 OCR0A – Timer/Counter0 Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x13 (0x33)	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 (TCNT0L). A match can be used to generate an output compare interrupt.

In 16-bit mode the OCR0A register contains the low byte of the 16-bit output compare register. To ensure that both the high and the low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 14.9 “Accessing Registers in 16-bit Mode” on page 76.

14.10.5 OCR0B – Timer/Counter0 Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	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 (TCNT0L in 8-bit mode and TCNTH in 16-bit mode). A match can be used to generate an output compare interrupt.

In 16-bit mode the OCR0B register contains the high byte of the 16-bit output compare register. To ensure that both the high and the low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 14.9 “Accessing Registers in 16-bit Mode” on page 76.

14.10.6 TIMSK – Timer/Counter0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	**OCIE1D**	**OCIE1A**	**OCIE1B**	**OCIE0A**	**OCIE0B**	**TOIE1**	**TOIE0**	**TICIE0**	**TIMSK**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – 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 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 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.

- **Bit 0 – TICIE0: Timer/Counter0, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the status register is set (interrupts globally enabled), the Timer/Counter1 input capture interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 47) is executed when the ICF0 flag, located in TIFR, is set.

14.10.7 TIFR – Timer/Counter0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	**OCF1D**	**OCF1A**	**OCF1B**	**OCF0A**	**OCF0B**	**TOV1**	**TOV0**	**ICF0**	**TIFR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **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.

The OCF0A is also set in 16-bit mode when a compare match occurs between the Timer/Counter and 16-bit data in OCR0B/A. The OCF0A is not set in input capture mode when the output compare register OCR0A is used as an Input capture register.

- **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.

The OCF0B is not set in 16-bit output compare mode when the output compare register OCR0B is used as the high byte of the 16-bit output compare register or in 16-bit input capture mode when the output compare register OCR0B is used as the high byte of the input capture register.

- **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.

- **Bits 0 – ICF0: Timer/Counter0, Input Capture Flag**

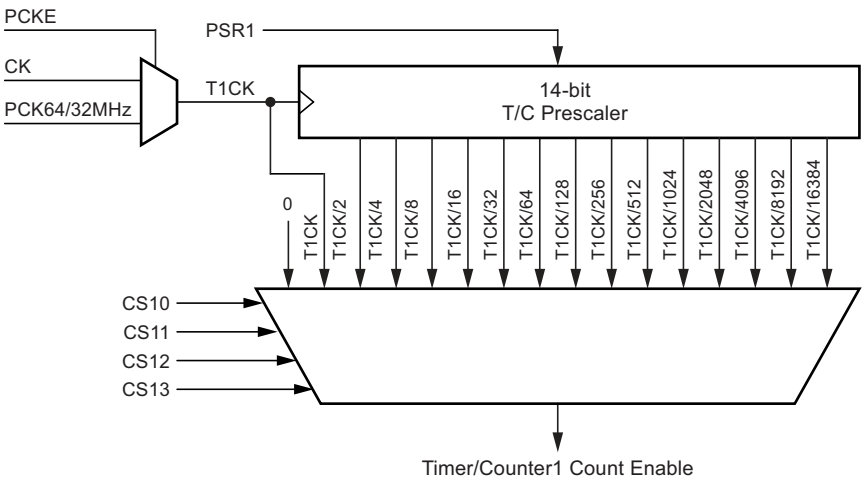
This flag is set when a capture event occurs on the ICP0 pin. When the input capture register (ICR0) is set to be used as the TOP value, the ICF0 flag is set when the counter reaches the TOP value.

ICF0 is automatically cleared when the input capture interrupt vector is executed. Alternatively, ICF0 can be cleared by writing a logic one to its bit location.

15. Timer/Counter1 Prescaler

Figure 15-1 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 a clock time base and asynchronous mode uses the fast peripheral clock (PCK) as a clock time base. The PCKE bit from the PLLCSR register enables the asynchronous mode when it is set (‘1’).

Figure 15-1. 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 15-1 on page 86 and the Timer/Counter1 control register, TCCR1B.

The frequency of the fast peripheral clock is 64MHz or 32MHz in low speed mode (the LSM bit in PLLCSR register is set to one). The low speed mode is recommended to use when the supply voltage below 2.7V are used.

15.1 Prescaler Reset

Setting the PSR1 bit in TCCR1B register resets the prescaler. It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution.

15.2 Prescaler Initialization for Asynchronous Mode

To change Timer/Counter1 to the asynchronous mode follow the procedure below:

1. Enable PLL.
2. Wait 100µs for PLL to stabilize.
3. Poll the PLOCK bit until it is set.
4. Set the PCKE bit in the PLLCSR register which enables the asynchronous mode.

15.3 Register Description

15.3.1 PLLCSR – PLL Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x29 (0x49)	**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 low speed mode is set, if the LSM bit is written 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 2.7V, because the Timer/Counter1 is not running fast enough on low voltage levels. It is recommended that the Timer/Counter1 is stopped whenever the LSM bit is changed.

Note, that LSM can not be set if PLL_{CLK} is used as a system clock.

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

These bits are reserved bits in the ATtiny261/461/861 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 a 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. It is safe to set this bit only when the PLL is locked i.e the PLOCK bit is 1. Note that the PCKE bit can be set only, if the PLL has been enabled earlier. The PLL is enabled when the CKSEL fuse has been programmed to 0x0001 (the PLL clock mode is selected) or the PLLE bit has been set to one.

- **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. The PLOCK bit should be ignored during initial PLL lock-in sequence when PLL frequency overshoots and undershoots, before reaching steady state. The steady state is obtained within 100µs. After PLL lock-in it is recommended to check the PLOCK bit before enabling PCK for Timer/Counter1.

15.3.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	-	PSR1	DTPS11	DTPS10	CS13	CS12	CS11	CS10	TCCR1B
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 - Res: Reserved Bit**

- **Bit 6 - 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.

- **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 15-1. 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.

16. Timer/Counter1

16.1 Features

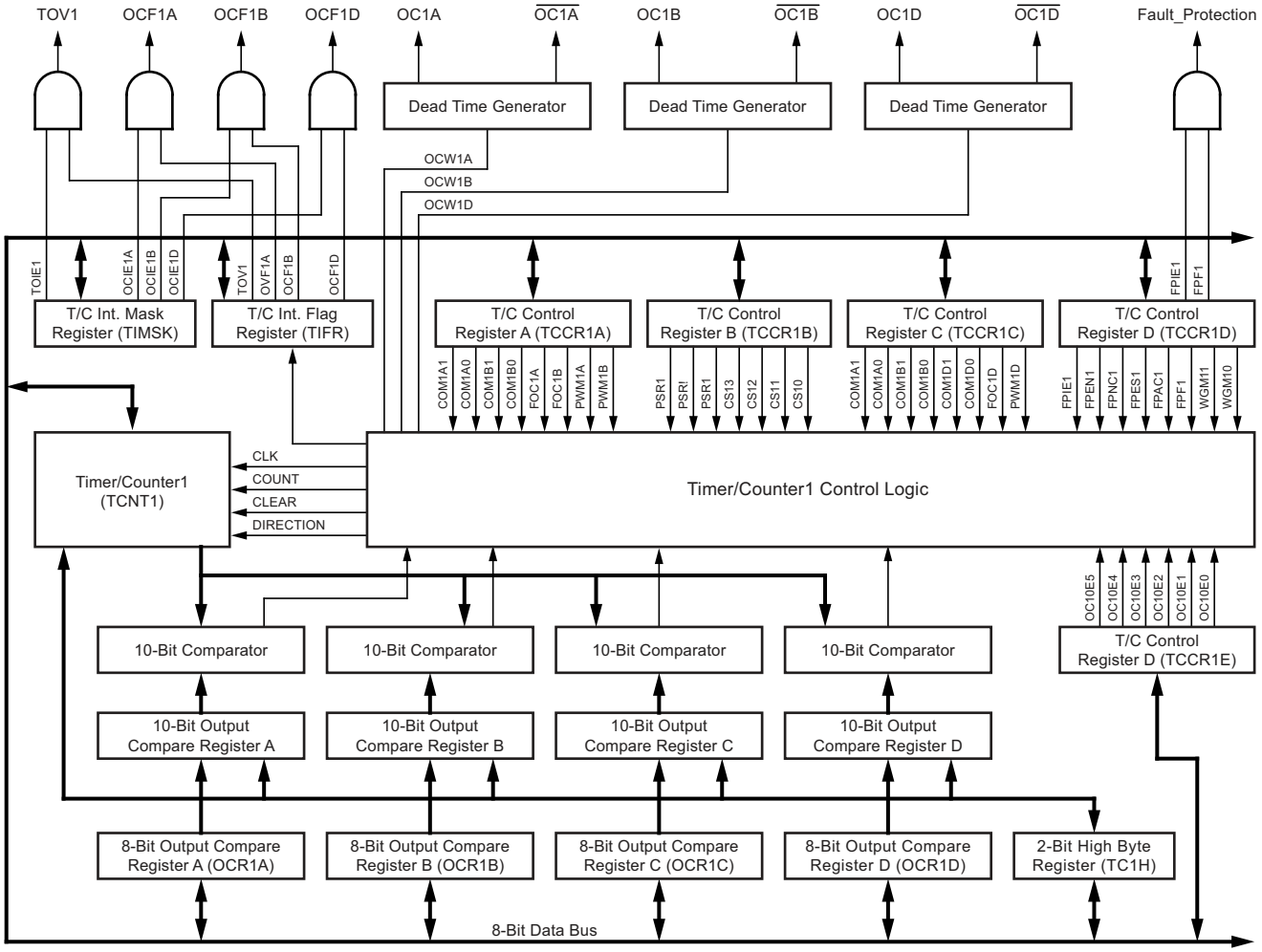
- 10/8-bit accuracy
- Three independent output compare units
- Clear timer on compare match (auto reload)
- Glitch free, phase and frequency correct pulse width modulator (PWM)
- Variable PWM period
- Independent dead time generators for each PWM channels
- Five independent interrupt sources (TOV1, OCF1A, OCD1B, OCF1D, FPF1)
- High speed asynchronous and synchronous clocking modes
- Separate prescaler unit

16.2 Overview

Timer/Counter1 is a general purpose high speed Timer/Counter module, with three independent output compare units, and with PWM support.

The Timer/Counter1 features a high resolution and a high accuracy usage with the lower prescaling opportunities. It can also support three accurate and high speed pulse width modulators using clock speeds up to 64MHz. In PWM mode Timer/Counter1 and the output compare registers serve as triple stand-alone PWMs with non-overlapping non-inverted and inverted outputs. Similarly, the high prescaling opportunities make this unit useful for lower speed functions or exact timing functions with infrequent actions. A simplified block diagram of the Timer/Counter1 is shown in Figure 16-1. For actual placement of the I/O pins, refer to Section 1-1 “Pinout ATtiny261/461/861” on page 3. The device-specific I/O register and bit locations are listed in the Section 16.11 “Register Description” on page 108.

Figure 16-1. Timer/Counter1 Block Diagram



16.2.1 Speed

The maximum speed of the Timer/Counter1 is 64MHz. However, if a supply voltage below 2.7V is used, it is recommended to use the low speed mode (LSM), because the Timer/Counter1 is not running fast enough on low voltage levels. In the low speed mode the fast peripheral clock is scaled down to 32MHz. For more details about the low speed mode, see Section 15.3.1 “PLLCSR – PLL Control and Status Register” on page 85.

16.2.2 Accuracy

The Timer/Counter1 is a 10-bit Timer/Counter module that can alternatively be used as an 8-bit Timer/Counter. The Timer/Counter1 registers are basically 8-bit registers, but on top of that there is a 2-bit high byte register (TC1H) that can be used as a common temporary buffer to access the two MSBs of the 10-bit Timer/Counter1 registers by the AVR CPU via the 8-bit data bus, if the 10-bit accuracy is used. Whereas, if the two MSBs of the 10-bit registers are written to zero the Timer/Counter1 is working as an 8-bit Timer/Counter. When reading the low byte of any 8-bit register the two MSBs are written to the TC1H register, and when writing the low byte of any 8-bit register the two MSBs are written from the TC1H register. Special procedures must be followed when accessing the 10-bit Timer/Counter1 values via the 8-bit data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.2.3 Registers

The Timer/Counter (TCNT1) and output compare registers (OCR1A, OCR1B, OCR1C and OCR1D) are 8-bit registers that are used as a data source to be compared with the TCNT1 contents. The OCR1A, OCR1B and OCR1D registers determine the action on the OC1A, OC1B and OC1D pins and they can also generate the compare match interrupts. The OCR1C holds the Timer/Counter TOP value, i.e. the clear on compare match value. The Timer/Counter1 high byte register (TC1H) is a 2-bit register that is used as a common temporary buffer to access the MSB bits of the Timer/Counter1 registers, if the 10-bit accuracy is used.

Interrupt request (overflow TOV1, and compare matches OCF1A, OCF1B, OCF1D and fault protection FPF1) signals are visible in the timer interrupt flag register (TIFR) and Timer/Counter1 control register D (TCCR1D). The interrupts are individually masked with the timer interrupt mask register (TIMSK) and the FPIE1 bit in the Timer/Counter1 control register D (TCCR1D).

Control signals are found in the Timer/Counter control registers TCCR1A, TCCR1B, TCCR1C, TCCR1D and TCCR1E.

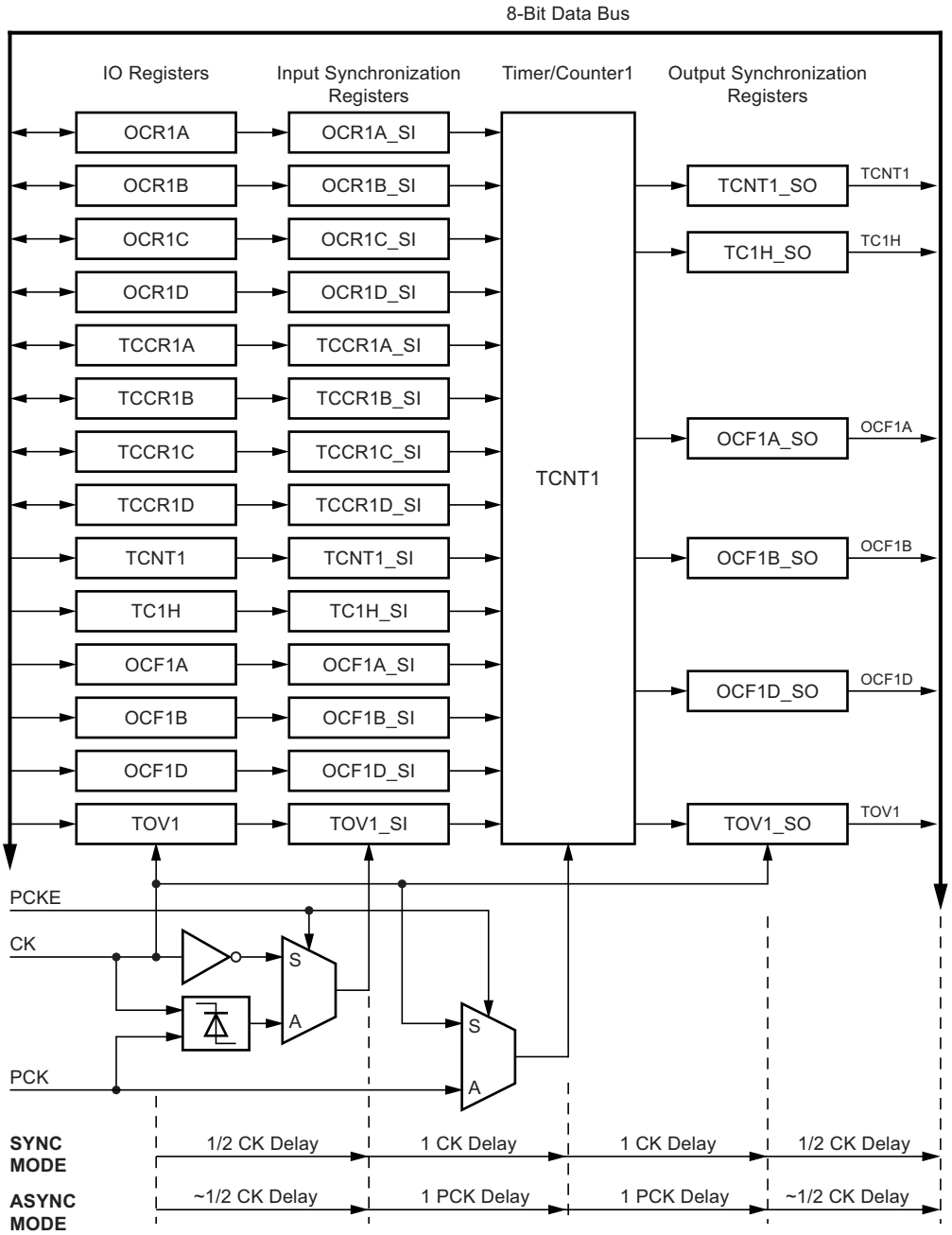
16.2.4 Synchronization

In asynchronous clocking mode the Timer/Counter1 and the prescaler allow running the CPU from any clock source while the prescaler is operating on the fast peripheral clock (PCK) having frequency of 64MHz (or 32MHz in low speed mode). This is possible because there is a synchronization boundary between the CPU clock domain and the fast peripheral clock domain. Figure 16-2 shows Timer/Counter 1 synchronization register block diagram and describes 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 TCCR1A, TCCR1B, TCCR1C, TCCR1D, OCR1A, OCR1B, OCR1C and OCR1D can be read back right after writing the register. The read back values are delayed for the Timer/Counter1 (TCNT1) register, Timer/Counter1 High Byte Register (TC1H) and flags (OCF1A, OCF1B, OCF1D and TOV1), because of the input and output synchronization.

The system clock frequency must be lower than half of the PCK frequency, because 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.

Figure 16-2. Timer/Counter1 Synchronization Register Block Diagram



16.2.5 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, B, C or D. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on. The definitions in Table 16-1 are used extensively throughout the document.

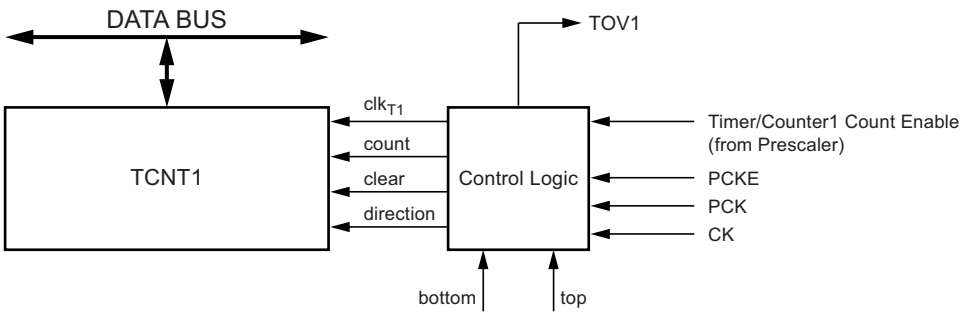
Table 16-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0.
MAX	The counter reaches its MAXimum value when it becomes 0x3FF (decimal 1023).
TOP	The counter reaches the TOP value (stored in the OCR1C) when it becomes equal to the highest value in the count sequence. The TOP has a value 0x0FF as default after reset.

16.3 Counter Unit

The main part of the Timer/Counter1 is the programmable bi-directional counter unit. Figure 16-3 shows a block diagram of the counter and its surroundings.

Figure 16-3. Counter Unit Block Diagram



Signal description (internal signals):

count TCNT1 increment or decrement enable.

direction Select between increment and decrement.

clear Clear TCNT1 (set all bits to zero).

clk_{Tn} Timer/Counter clock, referred to as clk_{T1} in the following.

top Signalize that TCNT1 has reached maximum value.

bottom Signalize that TCNT1 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T1}). The timer clock is generated from an synchronous system clock or an asynchronous PLL clock using the clock select bits (CS13:0) and the PCK enable bit (PCKE). When no clock source is selected (CS13:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, regardless of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence of the Timer/Counter1 is determined by the setting of the WGM10 and PWM1x bits located in the Timer/Counter1 control registers (TCCR1A, TCCR1C and TCCR1D). For more details about advanced counting sequences and waveform generation, see Section 16.7 “Modes of Operation” on page 97. The Timer/Counter overflow flag (TOV1) is set according to the mode of operation selected by the PWM1x and WGM10 bits. The overflow flag can be used for generating a CPU interrupt.

16.3.1 Counter Initialization for Asynchronous Mode

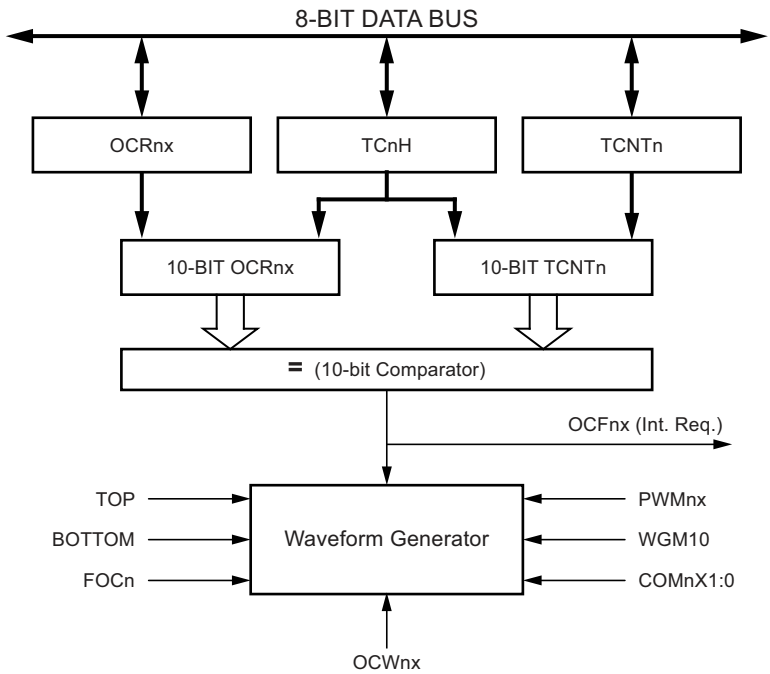
To change Timer/Counter1 to the asynchronous mode follow the procedure below:

1. Enable PLL.
2. Wait 100µs for PLL to stabilize.
3. Poll the PLOCK bit until it is set.
4. Set the PCKE bit in the PLLCSR register which enables the asynchronous mode.

16.4 Output Compare Unit

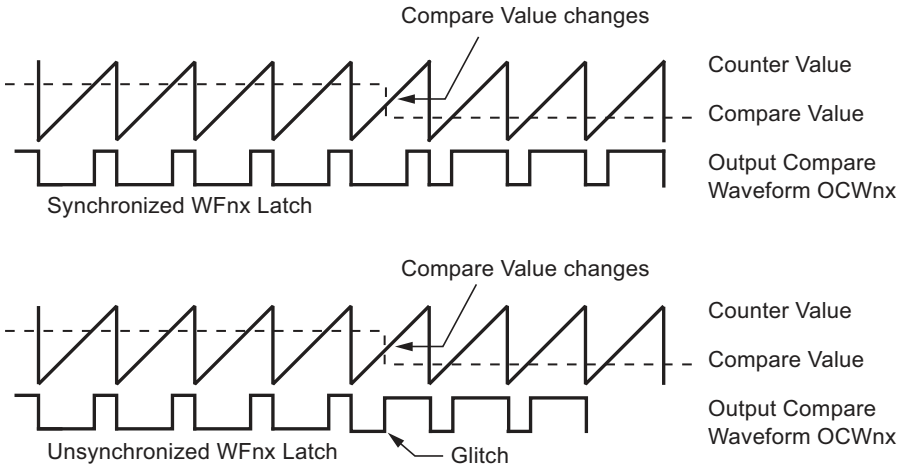
The comparator continuously compares TCNT1 with the output compare registers (OCR1A, OCR1B, OCR1C and OCR1D). Whenever TCNT1 equals to the output compare register, the comparator signals a match. A match will set the output compare flag (OCF1A, OCF1B or OCF1D) at the next timer clock cycle. If the corresponding 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 PWM1x, WGM10 and compare output mode (COM1x1:0) bits. The top and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (see Section 16.7 “Modes of Operation” on page 97). Figure 16-4 shows a block diagram of the output compare unit.

Figure 16-4. Output Compare Unit, Block Diagram



The OCR1x registers are double buffered when using any of the pulse width modulation (PWM) modes. For the normal mode of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x 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. See Figure 16-5 for an example. During the time between the write and the update operation, a read from OCR1A, OCR1B, OCR1C or OCR1D will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A, OCR1B, OCR1C or OCR1D.

Figure 16-5. Effects of Unsynchronized OCR Latching



16.4.1 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 (FOC1x) bit. Forcing compare match will not set the OCF1x flag or reload/clear the timer, but the waveform output (OCW1x) will be updated as if a real compare match had occurred (the COM1x1:0 bits settings define whether the waveform output (OCW1x) is set, cleared or toggled).

16.4.2 Compare Match Blocking by TCNT1 Write

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

16.4.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is down-counting.

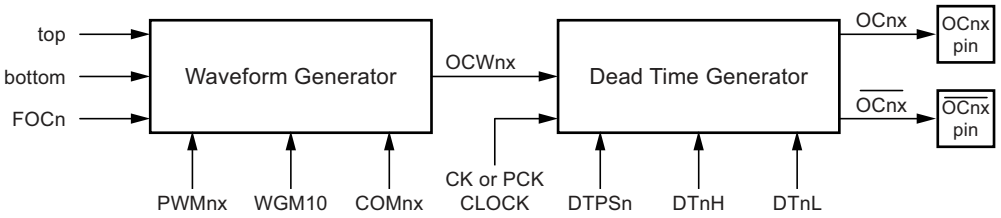
The setup of the waveform Output (OCW1x) should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OCW1x value is to use the force output compare (FOC1x) strobe bits in normal mode. The OC1x keeps its value even when changing between waveform generation modes.

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

16.5 Dead Time Generator

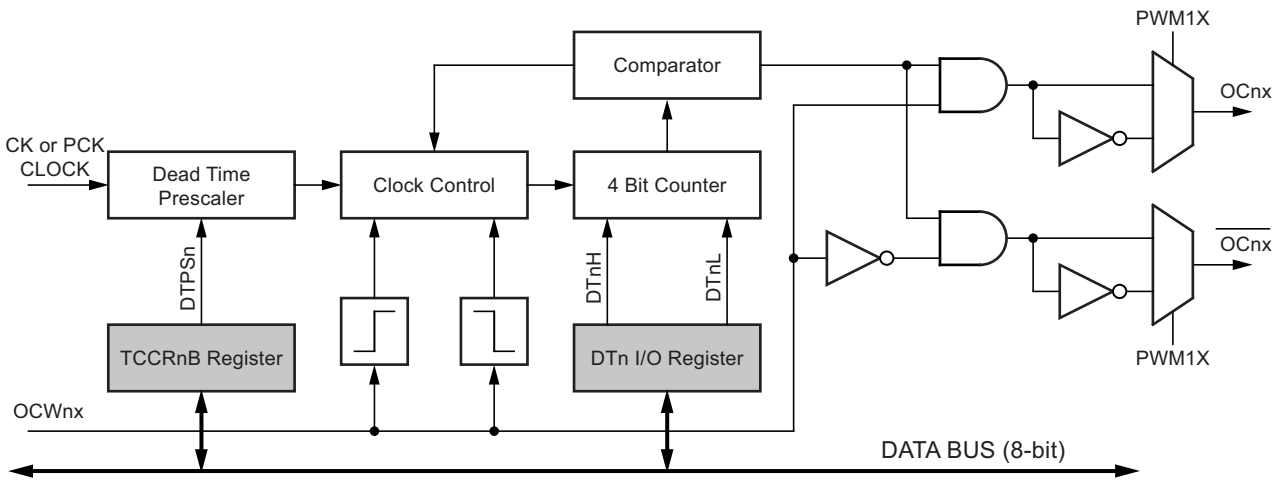
The dead time generator is provided for the Timer/Counter1 PWM output pairs to allow driving external power control switches safely. The dead time generator is a separate block that can be used to insert dead times (non-overlapping times) for the Timer/Counter1 complementary output pairs OC1x and $\overline{OC1x}$ when the PWM mode is enabled and the COM1x1:0 bits are set to “01”. The sharing of tasks is as follows: the waveform generator generates the waveform output (OCW1x) and the dead time generator generates the non-overlapping PWM output pair from the waveform output. Three 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 16-6. Output Compare Unit, Block Diagram



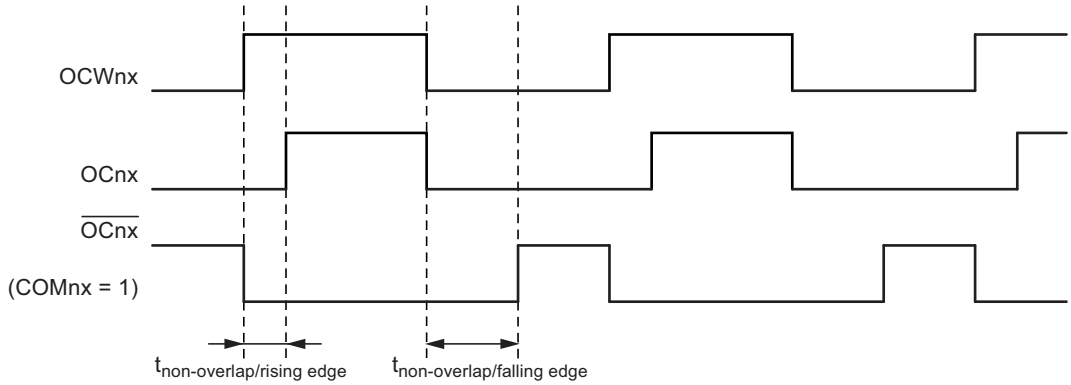
The dead time generation is based on the 4-bit down counters that count the dead time, as shown in Figure 16-7. There is a dedicated prescaler in front of the dead time generator that can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8. This provides for large range of dead times that can be generated. The prescaler is controlled by two control bits DTPS11..10. 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 $\overline{OC1x}$ 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 DT1H or DT1L value from DT1 I/O register, depending on the edge of the waveform output (OCW1x) when the dead time insertion is started. The output compare output are delayed by one timer clock cycle at minimum from the waveform output when the dead time is adjusted to zero. The outputs OC1x and $\overline{OC1x}$ are inverted, if the PWM inversion mode bit PWM1X is set. This will also cause both outputs to be high during the dead time.

Figure 16-7. Dead Time Generator



The length of the counting period is user adjustable by selecting the dead time prescaler setting by using the DTPS11:10 control bits, and selecting then the dead time value in I/O register DT1. The DT1 register consists of two 4-bit fields, DT1H and DT1L that control the dead time periods of the PWM output and its' complementary output separately in terms of the number of prescaled dead time generator clock cycles. Thus the rising edge of OC1x and $\overline{OC1x}$ can have different dead time periods as the $t_{non\text{-}overlap\ /\ rising\ edge}$ is adjusted by the 4-bit DT1H value and the $t_{non\text{-}overlap\ /\ falling\ edge}$ is adjusted by the 4-bit DT1L value.

Figure 16-8. The Complementary Output Pair, COM1x1:0 = 1



16.6 Compare Match Output Unit

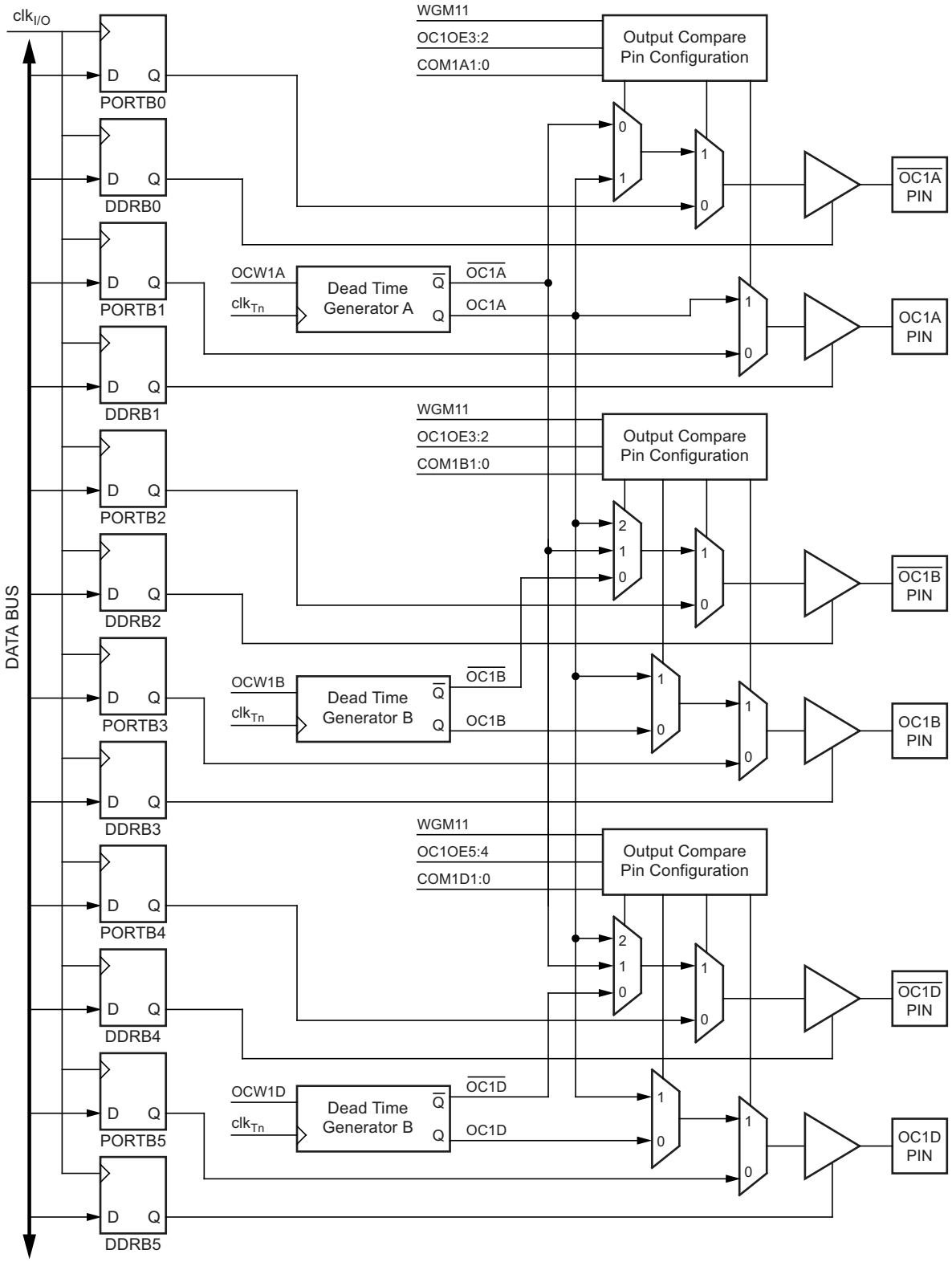
The compare output mode (COM1x1:0) bits have two functions. The waveform generator uses the COM1x1:0 bits for defining the inverted or non-inverted waveform output (OCW1x) at the next compare match. Also, the COM1x1:0 bits control the OC1x and $\overline{OC1x}$ pin output source. Figure 16-9 shows a simplified schematic of the logic affected by the COM1x1:0 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 COM1x1:0 bits are shown.

In normal mode (non-PWM) the dead time generator is disabled and it is working like a synchronizer: the output compare (OC1x) is delayed from the waveform output (OCW1x) by one timer clock cycle. Whereas in fast PWM mode and in phase and frequency correct PWM mode when the COM1x1:0 bits are set to “01” both the non-inverted and the inverted output compare output are generated, and an user programmable dead time delay is inserted for these complementary output pairs (OC1x and $\overline{OC1x}$). The functionality in PWM modes is similar to normal mode when any other COM1x1:0 bit setup is used. When referring to the OC1x state, the reference is for the output compare output (OC1x) from the dead time generator, not the OC1x pin. If a system reset occur, the OC1x is reset to “0”.

The general I/O port function is overridden by the output compare (OC1x / $\overline{OC1x}$) from the dead time generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC1x and $\overline{OC1x}$ pins (DDR_OC1x and DDR_$\overline{OC1x}$) must be set as output before the OC1x and $\overline{OC1x}$ values are visible on the pin. The port override function is independent of the output compare mode.

The design of the output compare pin configuration logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. For output compare pin configurations refer to Table 16-2 on page 98, Table 16-3 on page 99, Table 16-4 on page 100, and Table 16-5 on page 102.

Figure 16-9. Compare Match Output Unit, Schematic



16.6.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM1x1:0 bits differently in normal mode and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the waveform generator that no action on the OCW1x output is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 16-6 on page 108. For fast PWM mode, refer to Table 16-7 on page 108, and for the phase and frequency correct PWM refer to Table 16-8 on page 109. A change of the COM1x1:0 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 FOC1x strobe bits.

16.7 Modes of Operation

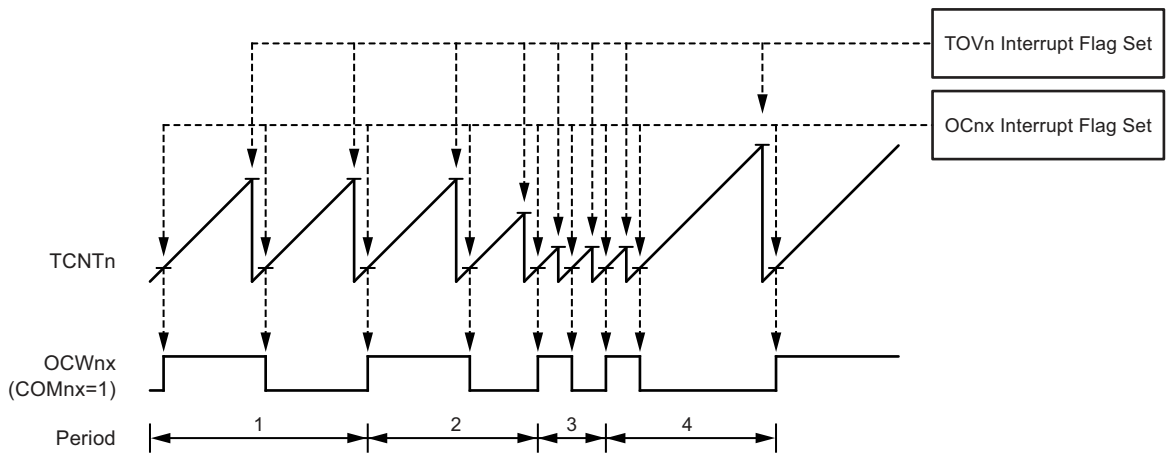
The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (bits PWM1x and WGM10) and compare output mode (COM1x1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted, non-inverted or complementary. For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared, or toggled at a compare match.

16.7.1 Normal Mode

The simplest mode of operation is the Normal mode (PWM1x = 0), the counter counts from BOTTOM to TOP (defined as OCR1C) then restarts from BOTTOM. The OCR1C defines the TOP value for the counter, hence also its resolution, and allows control of the compare match output frequency. In toggle compare output mode the waveform output (OCW1x) is toggled at compare match between TCNT1 and OCR1x. In non-inverting compare output mode the waveform output is cleared on the compare match. In inverting compare output mode the waveform output is set on compare match.

The timing diagram for the normal mode is shown in Figure 16-10. The counter value (TCNT1) that is shown as a histogram in the timing diagram is incremented until the counter value matches the TOP value. The counter is then cleared at the following clock cycle The diagram includes the waveform output (OCW1x) in toggle compare mode. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1.

Figure 16-10. Normal Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set in the same clock cycle as the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 11th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt, that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

The 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. For generating a waveform, the OCW1x output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1x1:0 = 1). The OC1x 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 $f_{OC1x} = f_{clkT1}/4$ when OCR1C is set to zero. The waveform frequency is defined by the following equation:

$$f_{OC1x} = \frac{f_{clkT1}}{2 \cdot (1 + OCR1C)}$$

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

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

The output compare pin configurations in normal mode are described in Table 16-2.

Table 16-2. Output Compare Pin Configurations in Normal Mode

COM1x1	COM1x0	$\overline{OC1x}$ Pin	OC1x Pin
0	0	Disconnected	Disconnected
0	1	Disconnected	OC1x
1	0	Disconnected	OC1x
1	1	Disconnected	OC1x

16.7.2 Fast PWM Mode

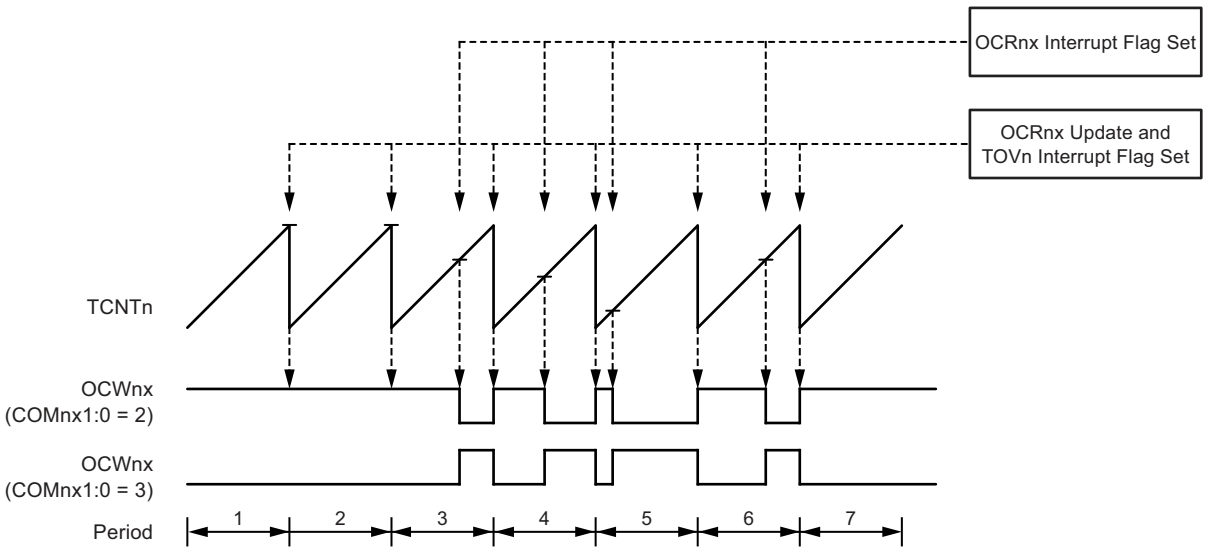
The fast pulse width modulation or fast PWM mode (PWM1x = 1 and WGM10 = 0) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP (defined as OCR1C) then restarts from BOTTOM. In non-inverting compare output mode the waveform output (OCW1x) is cleared on the compare match between TCNT1 and OCR1x and set at BOTTOM. In inverting compare output mode, the waveform output is set on compare match and cleared at BOTTOM. In complementary compare output mode the waveform output is cleared on the compare match and set at BOTTOM.

Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase and frequency 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.

The timing diagram for the fast PWM mode is shown in Figure 16-11. The counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation.

The diagram includes the Waveform Output in non-inverted and inverted compare output modes. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1.

Figure 16-11. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) 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 OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and setting the COM1x1:0 to three will produce an inverted PWM output. Setting the COM1x1:0 bits to one will enable complementary compare output mode and produce both the non-inverted (OC1x) and inverted output ($\overline{OC1x}$). The actual 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 waveform output (OCW1x) at the compare match between OCR1x and TCNT1, and clearing (or setting) the waveform output 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:

$$f_{OCnxPWM} = \frac{f_{clkT1}}{N}$$

The *N* variable represents the number of steps in single-slope operation. The value of *N* equals either to the TOP value.

The extreme values for the OCR1C register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1C is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR1C equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting the waveform output (OCW1x) to toggle its logical level on each Compare Match (COM1x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC1} = f_{clkT1}/4$ when OCR1C is set to three.

The general I/O port function is overridden by the output compare value (OC1x / $\overline{OC1x}$) from the dead time generator, if either of the COM1x1:0 bits are set and the data direction register bits for the OC1X and $\overline{OC1X}$ pins are set as an output. If the COM1x1:0 bits are cleared, the actual value from the port register will be visible on the port pin. The output compare pin configurations are described in Table 16-3.

Table 16-3. Output Compare Pin Configurations in Fast PWM Mode

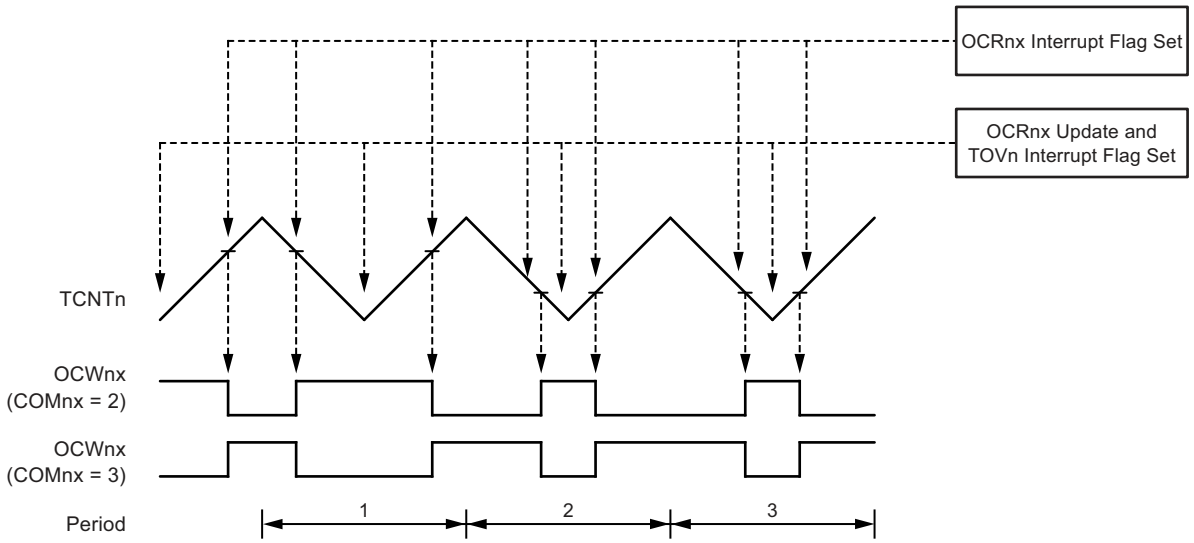
COM1x1	COM1x0	$\overline{OC1x}$ Pin	OC1x Pin
0	0	Disconnected	Disconnected
0	1	OC1x	OC1x
1	0	Disconnected	OC1x
1	1	Disconnected	OC1x

16.7.3 Phase and Frequency Correct PWM Mode

The phase and frequency correct PWM mode (PWMx = 1 and WGM10 = 1) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP (defined as OCR1C) and then from TOP to BOTTOM. In non-inverting compare output mode the waveform output (OCW1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while down-counting. In inverting output compare mode, the operation is inverted. In complementary compare output mode, the waveform output is cleared on the compare match and set at BOTTOM. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The timing diagram for the phase and frequency correct PWM mode is shown on Figure 16-12 in which the TCNT1 value is shown as a histogram for illustrating the dual-slope operation. The counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The diagram includes the waveform output (OCW1x) in non-inverted and inverted compare output mode. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1.

Figure 16-12. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) 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 the phase and frequency correct PWM mode, the compare unit allows generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and setting the COM1x1:0 to three will produce an inverted PWM output. Setting the COM1A1:0 bits to one will enable complementary compare output mode and produce both the non-inverted (OC1x) and inverted output ($\overline{OC1x}$). The actual values 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 waveform output (OCW1x) at the compare match between OCR1x and TCNT1 when the counter increments, and setting (or clearing) the waveform output at compare match when the counter decrements. The PWM frequency for the output when using the phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clkT1}}{N}$$

The *N* variable represents the number of steps in dual-slope operation. The value of *N* equals to the TOP value.

The extreme values for the OCR1C register represent special cases when generating a PWM waveform output in the phase and frequency correct PWM mode. If the OCR1C 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.

The general I/O port function is overridden by the Output Compare value (OC1x / $\overline{OC1x}$) from the dead time generator, if either of the COM1x1:0 bits are set and the data direction register bits for the OC1X and $\overline{OC1X}$ pins are set as an output. If the COM1x1:0 bits are cleared, the actual value from the port register will be visible on the port pin. The configurations of the output compare pins are described in Table 16-4.

Table 16-4. Output Compare pin configurations in Phase and Frequency Correct PWM Mode

COM1x1	COM1x0	$\overline{OC1x}$ Pin	OC1x Pin
0	0	Disconnected	Disconnected
0	1	OC1x	OC1x
1	0	Disconnected	OC1x
1	1	Disconnected	OC1x

16.7.4 PWM6 Mode

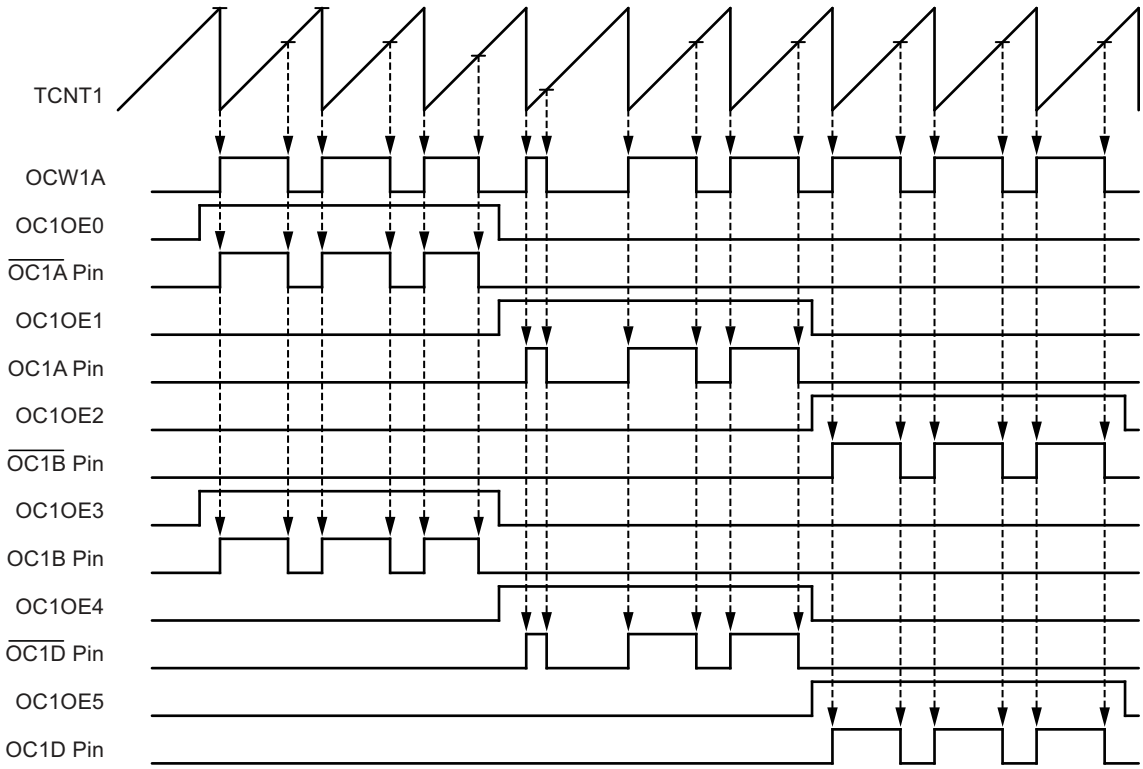
The PWM6 Mode (PWM1A = 1, WGM11 = 1 and WGM10 = x) provide PWM waveform generation option e.g. for controlling brushless DC (BLDC) motors. In the PWM6 mode the OCR1A register controls all six output compare waveforms as the same waveform output (OCW1A) from the waveform generator is used for generating all waveforms. The PWM6 mode also provides an output compare override enable register (OC1OE) that can be used with an instant response for disabling or enabling the output compare pins. If the output compare override enable bit is cleared, the actual value from the port register will be visible on the port pin.

The PWM6 mode provides two counter operation modes, a single-slope operation and a dual-slope operation. If the single-slope operation is selected (the WGM10 bit is set to 0), the counter counts from BOTTOM to TOP (defined as OCR1C) then restart from BOTTOM like in fast PWM Mode. The PWM waveform is generated by setting (or clearing) the waveform output (OCW1A) at the compare match between OCR1A and TCNT1, and clearing (or setting) the waveform output at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM). The Timer/Counter overflow flag (TOV1) is set each time the counter reaches the TOP and, if the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

Whereas, if the dual-slope operation is selected (the WGM10 bit is set to 1), the counter counts repeatedly from BOTTOM to TOP (defined as OCR1C) and then from TOP to BOTTOM like in phase and frequency correct PWM mode. The PWM waveform is generated by setting (or clearing) the waveform output (OCW1A) at the compare match between OCR1A and TCNT1 when the counter increments, and clearing (or setting) the waveform output at the he compare match between OCR1A and TCNT1 when the counter decrements. The Timer/Counter overflow flag (TOV1) is set each time the counter reaches the BOTTOM and, if the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

The timing diagram for the PWM6 Mode in single-slope operation (WGM11 = 0) when the COM1A1:0 bits are set to “10” is shown in Figure 16-13. The counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The timing diagram includes output compare pins $\overline{OC1A}$ and OC1A, and the corresponding output compare override enable bits (OC1OE1..OC1OE0).

Figure 16-13. PWM6 Mode, Single-slope Operation, Timing Diagram



The general I/O port function is overridden by the output compare value (OC1x / $\overline{OC1x}$) from the dead time generator if either of the COM1x1:0 bits are set. The output compare pins can also be overridden by the output compare override enable bits OC1OE5..OC1OE0. If an override enable bit is cleared, the actual value from the port register will be visible on the port pin and, if the override enable bit is set, the output compare pin is allowed to be connected on the port pin. The output compare pin configurations are described in Table 16-5.

Table 16-5. Output Compare Pin configurations in PWM6 Mode

COM1A1	COM1A0	$\overline{OC1A}$ Pin (PB0)	OC1A Pin (PB1)
0	0	Disconnected	Disconnected
0	1	OC1A × OC1OE0	OC1A × OC1OE1
1	0	OC1A × OC1OE0	OC1A × OC1OE1
1	1	OC1A × OC1OE0	OC1A × OC1OE1
COM1B1	**COM1B0**	**$\overline{OC1B}$ Pin (PB2)**	**OC1B Pin (PB3)**
0	0	Disconnected	Disconnected
0	1	OC1A × OC1OE2	OC1A × OC1OE3
1	0	OC1A × OC1OE2	OC1A × OC1OE3
1	1	OC1A × OC1OE2	OC1A × OC1OE3
COM1D1	**COM1D0**	**$\overline{OC1D}$ Pin (PB4)**	**OC1D Pin (PB5)**
0	0	Disconnected	Disconnected
0	1	OC1A × OC1OE4	OC1A × OC1OE5
1	0	OC1A × OC1OE4	OC1A × OC1OE5
1	1	OC1A × OC1OE4	OC1A × OC1OE5

16.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set.

Figure 16-14 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 and frequency correct PWM mode. Figure 16-15 shows the same timing data, but with the prescaler enabled, in all modes other than phase and frequency correct PWM mode. Figure 16-16 shows the setting of OCF1A, OCF1B and OCF1D in all modes, and Figure 16-17 shows the setting of TOV1 in phase and frequency correct PWM mode.

Figure 16-14. Timer/Counter Timing Diagram, no Prescaling

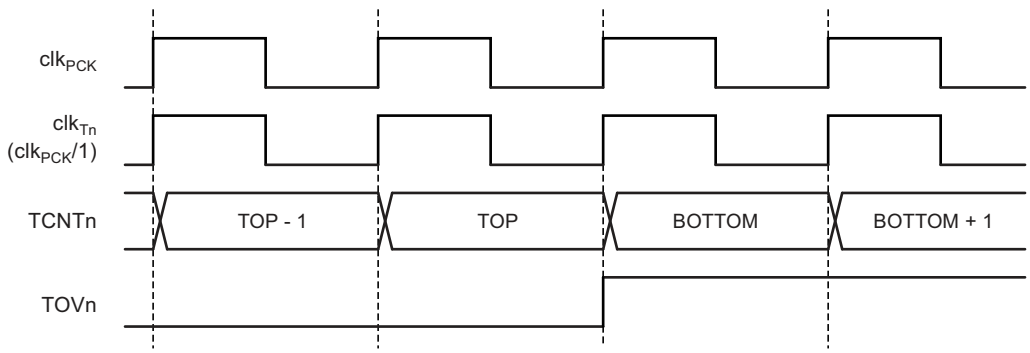


Figure 16-15. Timer/Counter Timing Diagram, with Prescaler (f_{clkT1}/8)

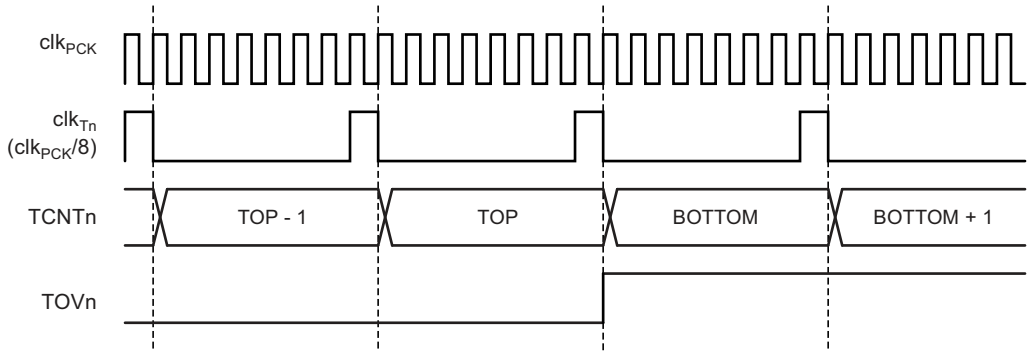


Figure 16-16. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler (f_{clkT1}/8)

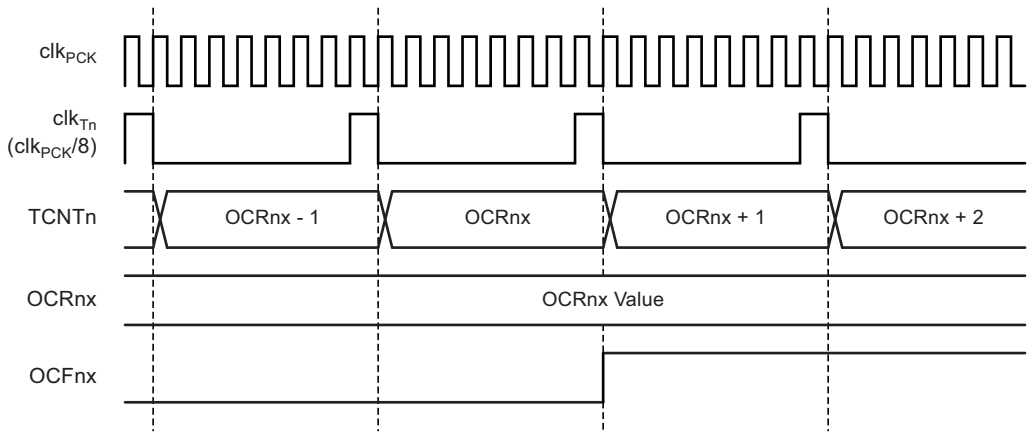
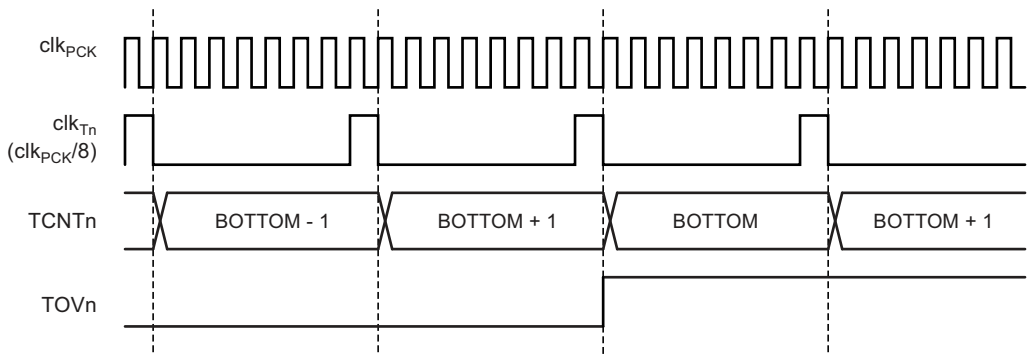


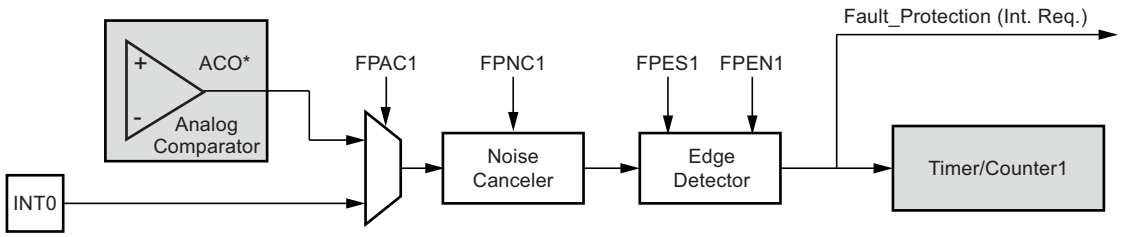
Figure 16-17. Timer/Counter Timing Diagram, with Prescaler (f_{clkT1}/8)



16.9 Fault Protection Unit

The Timer/Counter1 incorporates a fault protection unit that can disable the PWM output pins, if an external event is triggered. The external signal indicating an event can be applied via the external interrupt INT0 pin or alternatively, via the analog-comparator unit. The fault protection unit is illustrated by the block diagram shown in Figure 16-18. The elements of the block diagram that are not directly a part of the fault protection unit are gray shaded.

Figure 16-18. Fault Protection Unit Block Diagram



When the fault protection mode is enabled by the fault protection enable (FPEN1) bit and a change of the logic level (an event) occurs on the *external interrupt pin* (INT0), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a fault protection mode will be triggered. When a fault protection is triggered, the COM1x bits are cleared, output comparators are disconnected from the PWM output pins and the PORTB register bits are connected on the PWM output pins. The *Fault Protection Enable* (FPEN1) is automatically cleared at the same system clock as the COM1nx bits are cleared. If the *Fault Protection Interrupt Enable* bit (FPIE1) is set, a fault protection interrupt is generated and the FPEN1 bit is cleared. Alternatively the FPEN1 bit can be polled by software to figure out when the Timer/Counter has entered to fault protection mode.

16.9.1 Fault Protection Trigger Source

The main trigger source for the fault protection unit is the *external interrupt pin* (INT0). Alternatively the analog comparator output can be used as trigger source for the fault protection unit. The analog comparator is selected as trigger source by setting the *Fault Protection Analog Comparator* (FPAC1) bit in the *Timer/Counter1 Control Register* (TCCR1D). Be aware that changing trigger source can trigger a fault protection mode. Therefore it is recommended to clear the FPF1 flag after changing trigger source, setting edge detector or enabling the fault protection.

Both the external interrupt pin (INT0) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T0 pin (Figure 13-1 on page 66). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. An input capture can also be triggered by software by controlling the port of the INT0 pin.

16.9.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector. The noise canceler is enabled by setting the *Fault Protection Noise Canceler* (FPNC1) bit in Timer/Counter1 control register D (TCCR1D). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input. The noise canceler uses the system clock and is therefore not affected by the prescaler.

16.10 Accessing 10-Bit Registers

If 10-bit values are written to the TCNT1 and OCR1A/B/C/D registers, the 10-bit registers can be byte accessed by the AVR® CPU via the 8-bit data bus using two read or write operations. The 10-bit registers have a common 2-bit Timer/Counter1 high byte register (TC1H) that is used for temporary storing of the two MSBs of the 10-bit access. The same TC1H register is shared between all 10-bit registers. Accessing the low byte triggers the 10-bit read or write operation. When the low byte of a 10-bit register is written by the CPU, the high byte stored in the TC1H register, and the low byte written are both copied into the 10-bit register in the same clock cycle. When the low byte of a 10-bit register is read by the CPU, the high byte of the 10-bit register is copied into the TC1H register in the same clock cycle as the low byte is read.

To do a 10-bit write, the high byte must be written to the TC1H register before the low byte is written. For a 10-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 10-bit timer registers assuming that no interrupts updates the TC1H register. The same principle can be used directly for accessing the OCR1A/B/C/D registers.

Assembly Code Example

```
...
; Set TCNT1 to 0x01FF
ldi     r17,0x01
ldi     r16,0xFF
out     TC1H,r17
out     TCNT1,r16
; Read TCNT1 into r17:r16
in      r16,TCNT1
in      r17,TC1H
...
```

C Code Example

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TC1H = 0x01;
TCNT1 = 0xFF;
/* Read TCNT1 into i */
i = TCNT1;
i |= ((unsigned int)TC1H << 8);
...
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

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

It is important to notice that accessing 10-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 10-bit register, and the interrupt code updates the TC1H register by accessing the same or any other of the 10-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the TC1H register, the main code must disable the interrupts during the 16-bit access.

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

.

Assembly Code Example

```
TIM1_ReadTCNT1:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Read TCNT1 into r17:r16
        in      r16,TCNT1
        in      r17,TC1H
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

C Code Example

```
unsigned int TIM1_ReadTCNT1(void)
{
unsigned char sreg;
unsigned int i;
/* Save global interrupt flag */
sreg = SREG;
/* Disable interrupts */
_CLI();
/* Read TCNT1 into i */
i = TCNT1;
i |= ((unsigned int)TC1H << 8);
/* Restore global interrupt flag
SREG = sreg;
return i;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

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

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

```
Assembly Code Example
TIM1_WriteTCNT1:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Set TCNT1 to r17:r16
        out     TC1H,r17
        out     TCNT1,r16
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

```
C Code Example
void TIM1_WriteTCNT1(unsigned int i)
{
unsigned char sreg;
unsigned int i;
/* Save global interrupt flag */
sreg = SREG;
/* Disable interrupts */
_CLI();
/* Set TCNT1 to i */
TC1H = (i >> 8);
TCNT1 = (unsigned char)i;
/* Restore global interrupt flag */
SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

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

16.10.1 Reusing the temporary high byte register

If writing to more than one 10-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

16.11 Register Description

16.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM1A	PWM1B	TCCR1A
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	

- **Bits 7,6 - COM1A1, COM1A0: Comparator A Output Mode, Bits 1 and 0**

These bits control the behavior of the waveform output (OCW1A) and the connection of the output compare pin (OC1A). If one or both of the COM1A1:0 bits are set, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. The complementary $\overline{OC1B}$ output is connected only in PWM modes when the COM1A1:0 bits are set to “01”. Note that the data direction register (DDR) bit corresponding to the OC1A and $\overline{OC1A}$ pins must be set in order to enable the output driver.

The function of the COM1A1:0 bits depends on the PWM1A, WGM10 and WGM11 bit settings. Table 16-6 shows the COM1A1:0 bit functionality when the PWM1A bit is set to normal mode (non-PWM).

Table 16-6. Compare Output Mode, Normal Mode (non-PWM)

COM1A1..0	OCW1A Behavior	OC1A Pin	$\overline{OC1A}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Toggle on compare match	Connected	Disconnected
10	Clear on compare match	Connected	Disconnected
11	Set on compare match	Connected	Disconnected

Table 16-7 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to fast PWM mode.

Table 16-7. Compare Output Mode, Fast PWM Mode

COM1A1..0	OCW1A Behavior	OC1A	OC1A
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match Set when TCNT1 = 0x000	Connected	Connected
10	Cleared on compare match Set when TCNT1 = 0x000	Connected	Disconnected
11	Set on compare match Cleared when TCNT1 = 0x000	Connected	Disconnected

Table 16-8 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to Phase and Frequency Correct PWM Mode.

Table 16-8. Compare Output Mode, Phase and Frequency Correct PWM Mode

COM1A1..0	OCW1A Behavior	OC1A Pin	$\overline{OC1A}$ Pin
00	Normal port operation.	Disconnected	Disconnected
01	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Connected
10	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Disconnected
11	Set on compare match when up-counting Cleared on compare match when down-counting	Connected	Disconnected

Table 16-9 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to single-slope PWM6 Mode. In the PWM6 Mode the same waveform output (OCW1A) is used for generating all waveforms and the output compare values OC1A and $\overline{OC1A}$ are connected on the all OC1x and $\overline{OC1x}$ pins as described below.

Table 16-9. Compare Output Mode, Single-Slope PWM6 Mode

COM1A1..0	OCW1A Behavior	OC1x Pin	$\overline{OC1x}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match Set when TCNT1 = 0x000	OC1A	OC1A
10	Cleared on compare match Set when TCNT1 = 0x000	OC1A	OC1A
11	Set on compare match Cleared when TCNT1 = 0x000	OC1A	OC1A

Table 16-10 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to dual-slope PWM6 Mode.

Table 16-10. Compare Output Mode, Dual-Slope PWM6 Mode

COM1A1..0	OCW1A Behavior	OC1x Pin	$\overline{OC1x}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match when up-counting Set on compare match when down-counting	OC1A	OC1A
10	Cleared on compare match when up-counting Set on compare match when down-counting	OC1A	OC1A
11	Set on compare match when up-counting Cleared on compare match when down-counting	OC1A	OC1A

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

These bits control the behavior of the waveform output (OCW1B) and the connection of the output compare pin (OC1B). If one or both of the COM1B1:0 bits are set, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. The complementary $\overline{OC1B}$ output is connected only in PWM modes when the COM1B1:0 bits are set to “01”.

Note that the data direction register (DDR) bit corresponding to the OC1B pin must be set in order to enable the output driver.

The function of the COM1B1:0 bits depends on the PWM1B and WGM10 bit settings. Table 16-11 shows the COM1B1:0 bit functionality when the PWM1B bit is set to normal mode (non-PWM).

Table 16-11. Compare Output Mode, Normal Mode (non-PWM)

COM1B1..0	OCW1B Behavior	OC1B Pin	$\overline{\text{OC1B}}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Toggle on compare match	Connected	Disconnected
10	Clear on compare match	Connected	Disconnected
11	Set on compare match	Connected	Disconnected

Table 16-12 shows the COM1B1:0 bit functionality when the PWM1B and WGM10 bits are set to Fast PWM Mode.

Table 16-12. Compare Output Mode, Fast PWM Mode

COM1B1..0	OCW1B Behavior	OC1B Pin	$\overline{\text{OC1B}}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match Set when TCNT1 = 0x000	Connected	Connected
10	Cleared on compare match Set when TCNT1 = 0x000	Connected	Disconnected
11	Set on compare match Cleared when TCNT1 = 0x000	Connected	Disconnected

Table 16-13 shows the COM1B1:0 bit functionality when the PWM1B and WGM10 bits are set to phase and frequency correct PWM mode.

Table 16-13. Compare Output Mode, Phase and Frequency Correct PWM Mode

COM1B1..0	OCW1B Behavior	OC1B Pin	$\overline{\text{OC1B}}$ Pin
00	Normal port operation.	Disconnected	Disconnected
01	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Connected
10	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Disconnected
11	Set on compare match when up-counting Cleared on compare match when down-counting	Connected	Disconnected

- **Bit 3 - FOC1A: Force Output Compare Match 1A**

The FOC1A bit is only active when the PWM1A bit specify a non-PWM mode.

Writing a logical one to this bit forces a change in the waveform output (OCW1A) and the output compare pin (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 is always read as zero.

- **Bit 2 - FOC1B: Force Output Compare Match 1B**

The FOC1B bit is only active when the PWM1B bit specify a non-PWM mode.

Writing a logical one to this bit forces a change in the waveform output (OCW1B) and the output compare pin (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 is always read as zero.

- **Bit 1 - PWM1A: Pulse Width Modulator A Enable**

When set (one) this bit enables PWM mode based on comparator OCR1A

- **Bit 0 - PWM1B: Pulse Width Modulator B Enable**

When set (one) this bit enables PWM mode based on comparator OCR1B.

16.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	**PWM1X**	**PSR1**	**DTPS11**	**DTPS10**	**CS13**	**CS12**	**CS11**	**CS10**	**TCCR1B**
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 - PWM1X: PWM Inversion Mode**

When this bit is set (one), the PWM Inversion Mode is selected and the dead time generator outputs, OC1x and $\overline{OC1x}$ are inverted.

- **Bit 6 - PSR1: Prescaler Reset Timer/Counter1**

When this bit is set (one), the Timer/Counter1 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.

- **Bits 5,4 - DTPS11, DTPS10: Dead Time Prescaler Bits**

The Timer/Counter1 control register B is a 8-bit read/write register.

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 and DTPS10 from the dead time Prescaler register. These bits define the division factor of the dead time prescaler. The division factors are given in Table 16-14.

Table 16-14. 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

- **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 16-15. Timer/Counter1 Prescaler 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.

16.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	COM1A1S	COM1A0S	COM1B1S	COM1B0S	COM1D1	COM1D0	FOC1D	PWM1D	TCCR1C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7,6 - COM1A1S, COM1A0S: Comparator A Output Mode, Bits 1 and 0**

These bits are the shadow bits of the COM1A1 and COM1A0 bits that are described in Section 16.11.1 “TCCR1A – Timer/Counter1 Control Register A” on page 108.

- **Bits 5,4 - COM1B1S, COM1B0S: Comparator B Output Mode, Bits 1 and 0**

These bits are the shadow bits of the COM1A1 and COM1A0 bits that are described in Section 16.11.1 “TCCR1A – Timer/Counter1 Control Register A” on page 108.

- **Bits 3,2 - COM1D1, COM1D0: Comparator D Output Mode, Bits 1 and 0**

These bits control the behavior of the waveform output (OCW1D) and the connection of the output compare pin (OC1D). If one or both of the COM1D1:0 bits are set, the OC1D output overrides the normal port functionality of the I/O pin it is connected to. The complementary $\overline{OC1D}$ output is connected only in PWM modes when the COM1D1:0 bits are set to “01”. Note that the data direction register (DDR) bit corresponding to the OC1D pin must be set in order to enable the output driver.

The function of the COM1D1:0 bits depends on the PWM1D and WGM10 bit settings. Table 16-16 shows the COM1D1:0 bit functionality when the PWM1D bit is set to a normal mode (non-PWM).

Table 16-16. Compare Output Mode, Normal Mode (non-PWM)

COM1D1..0	OCW1D Behavior	OC1D Pin	$\overline{\text{OC1D}}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Toggle on compare match	Connected	Disconnected
10	Clear on compare match	Connected	Disconnected
11	Set on compare match	Connected	Disconnected

Table 16-17 shows the COM1D1:0 bit functionality when the PWM1D and WGM10 bits are set to fast PWM mode.

Table 16-17. Compare Output Mode, Fast PWM Mode

COM1D1..0	OCW1D Behavior	OC1D Pin	$\overline{\text{OC1D}}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match Set when TCNT1 = 0x000	Connected	Connected
10	Cleared on compare match Set when TCNT1 = 0x000	Connected	Disconnected
11	Set on compare match Clear when TCNT1 = 0x000	Connected	Disconnected

Table 16-18 on page 113 shows the COM1D1:0 bit functionality when the PWM1D and WGM10 bits are set to phase and frequency correct PWM mode.

Table 16-18. Compare Output Mode, Phase and Frequency Correct PWM Mode

COM1D1..0	OCW1D Behavior	OC1D Pin	$\overline{\text{OC1D}}$ Pin
00	Normal port operation.	Disconnected	Disconnected
01	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Connected
10	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Disconnected
11	Set on compare match when up-counting Cleared on compare match when down-counting	Connected	Disconnected

- **Bit 1 - FOC1D: Force Output Compare Match 1D**

The FOC1D bit is only active when the PWM1D bit specify a non-PWM mode.

Writing a logical one to this bit forces a change in the waveform output (OCW1D) and the output compare pin (OC1D) according to the values already set in COM1D1 and COM1D0. If COM1D1 and COM1D0 written in the same cycle as FOC1D, 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 COM1D1 and COM1D0 takes place as if a compare match had occurred, but no interrupt is generated. The FOC1D bit is always read as zero.

- **Bit 0 - PWM1D: Pulse Width Modulator D Enable**

When set (one) this bit enables PWM mode based on comparator OCR1D.

16.11.4 TCCR1D – Timer/Counter1 Control Register D

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	FPIE1	FPEN1	FPNC1	FPES1	FPAC1	FPF1	WGM11	WGM10	TCCR1D
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 - FPIE1: Fault Protection Interrupt Enable**

Setting this bit (to one) enables the fault protection Interrupt.

- **Bit 6– FPEN1: Fault Protection Mode Enable**

Setting this bit (to one) activates the fault protection mode.

- **Bit 5 – FPNC1: Fault Protection Noise Canceler**

Setting this bit activates the fault protection noise canceler. When the noise canceler is activated, the input from the fault protection pin (INT0) is filtered. The filter function requires four successive equal valued samples of the INT0 pin for changing its output. The fault protection is therefore delayed by four oscillator cycles when the noise canceler is enabled.

- **Bit 4 – FPES1: Fault Protection Edge Select**

This bit selects which edge on the fault protection pin (INT0) is used to trigger a fault event. When the FPES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the FPES1 bit is written to one, a rising (positive) edge will trigger the fault.

- **Bit 3 - FPAC1: Fault Protection Analog Comparator Enable**

When written logic one, this bit enables the fault protection function in Timer/Counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the fault protection front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 fault protection interrupt. When written logic zero, no connection between the analog comparator and the fault protection function exists. To make the comparator trigger the Timer/Counter1 fault frotection interrupt, the FPIE1 bit in the Timer/Counter1 control register D (TCCR1D) must be set.

- **Bit 2- FPF1: Fault Protection Interrupt Flag**

When the FPIE1 bit is set (one), the fault protection interrupt is enabled. Activity on the pin will cause an interrupt request even, if the fault protection pin is configured as an output. The corresponding interrupt of fault protection interrupt request is executed from the fault protection interrupt vector. The bit FPF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, FPF1 is cleared after a synchronization clock cycle by writing a logical one to the flag. When the SREG I-bit, FPIE1 and FPF1 are set, the Fault Interrupt is executed.

- **Bits 1:0 - WGM11, WGM10: Waveform Generation Mode Bits**

This bit associated with the PWMx bits control the counting sequence of the counter, the source for type of waveform generation to be used, see Table 16-19. Modes of operation supported by the Timer/Counter1 are: Normal mode (counter), fast PWM mode, phase and frequency correct PWM and PWM6 modes.

Table 16-19. Waveform Generation Mode Bit Description

PWM1x	WGM11..10	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	xx	Normal	OCR1C	Immediate	TOP
1	00	Fast PWM	OCR1C	TOP	TOP
1	01	Phase and frequency correct PWM	OCR1C	BOTTOM	BOTTOM
1	10	PWM6 / single-slope	OCR1C	TOP	TOP
1	11	PWM6 / dual-slope	OCR1C	BOTTOM	BOTTOM

16.11.5 TCCR1E – Timer/Counter1 Control Register E

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	-	-	OC1OE5	OC1OE4	OC1OE3	OC1OE2	OC1OE1	OC1OE0	TCCR1E
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 7:6 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny261/461/861 and always reads as zero.

- **Bits 5:0 – OC1OE5:OC1OE0: Output Compare Override Enable Bits**

These bits are the output compare override enable bits that are used to connect or disconnect the output compare pins in PWM6 modes with an instant response on the corresponding output compare pins.

The actual value from the port register will be visible on the port pin, when the output compare override enable bit is cleared. Table 16-20 shows the output compare override enable bits and their corresponding output compare pins.

Table 16-20. Output Compare Override Enable Bits versus Output Compare Pins

OC1OE0	OC1OE1	OC1OE2	OC1OE3	OC1OE4	OC1OE5
OC1A (PB0)	OC1A (PB1)	OC1B (PB2)	OC1B (PB3)	OC1D (PB4)	OC1D (PB5)

16.11.6 TCNT1 – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	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.

The Timer/Counter1 is realized as a 10-bit up/down 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. When a 10-bit accuracy is preferred, special procedures must be followed for accessing the 10-bit TCNT1 register via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105. Alternatively the Timer/Counter1 can be used as an 8-bit Timer/Counter. Note that the Timer/Counter1 always starts counting up after writing the TCNT1 register.

16.11.7 TC1H – Timer/Counter1 High Byte

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	-	-	-	-	-	-	TC19	TC18	TC1H
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The temporary Timer/Counter1 register is an 2-bit read/write register.

- **Bits 7:2 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny261/461/861 and always reads as zero.

- **Bits 1:0 - TC19, TC18: Two MSB bits of the 10-bit accesses**

If 10-bit accuracy is used, the Timer/Counter1 high byte register (TC1H) is used for temporary storing the MSB bits (TC19, TC18) of the 10-bit accesses. The same TC1H register is shared between all 10-bit registers within the Timer/Counter1. Note that special procedures must be followed when accessing the 10-bit TCNT1 register via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.8 OCR1A – Timer/Counter1 Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	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 TCCR1A. 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.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.9 OCR1B – Timer/Counter1 Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	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.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.10 OCR1C – Timer/Counter1 Output Compare Register C

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	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, and a compare match will clear TCNT1. This register has the same function in normal mode and PWM modes.

Note that, if a smaller value than three is written to the output compare register C, the value is automatically replaced by three as it is a minimum value allowed to be written to this register.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.11 OCR1D – Timer/Counter1 Output Compare Register D

Bit	7	6	5	4	3	2	1	0	
0x2A (0x4A)	MSB							LSB	OCR1D
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 D is an 8-bit read/write register.

The Timer/Counter output compare register D contains data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in TCCR1A. A compare match does only occur if Timer/Counter1 counts to the OCR1D value. A software write that sets TCNT1 and OCR1D to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF1D after a synchronization delay following the compare event.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit output compare registers via the 8-bit AVR data bus. These procedures are described in Section 16.10 “Accessing 10-Bit Registers” on page 105.

16.11.12 TIMSK – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	OCIE1D	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	TICIE0	TIMSK
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- OCIE1D: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1D bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 compare matchD, interrupt is enabled. The corresponding interrupt at vector $010 is executed if a compare matchD occurs. The compare flag in Timer/Counter1 is set (one) in the Timer/Counter interrupt flag register.

- **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 matchA, interrupt is enabled. The corresponding interrupt at vector $003 is executed if a compare matchA 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 matchB, interrupt is enabled. The corresponding interrupt at vector $009 is executed if a compare matchB occurs. The compare flag in Timer/Counter1 is set (one) in the Timer/Counter interrupt flag register.

- **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.

16.11.13 TIFR – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	OCF1D	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	ICF0	TIFR
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- OCF1D: Output Compare Flag 1D**

The OCF1D bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1D - output compare register 1D. OCF1D is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1D is cleared, after synchronization clock cycle, by writing a logic one to the flag. When the I-bit in SREG, OCIE1D, and OCF1D are set (one), the Timer/Counter1 D compare match interrupt is executed.

- **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 and fast PWM mode the TOV1 bit is set (one) each time the counter reaches TOP at the same clock cycle when the counter is reset to BOTTOM. In phase and frequency correct PWM mode the TOV1 bit is set (one) each time the counter reaches BOTTOM at the same clock cycle when zero is clocked to the counter.

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

16.11.14 DT1 – Timer/Counter1 Dead Time Value

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	DT1H3	DT1H2	DT1H1	DT1H0	DT1L3	DT1L2	DT1L1	DT1L0	DT1
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 is an 8-bit read/write register.

The dead time delay of all Timer/Counter1 channels are adjusted by the dead time value register, DT1. The register consists of two fields, DT1H3..0 and DT1L3..0, one for each complementary output. Therefore a different dead time delay can be adjusted for the rising edge of OC1x and the rising edge of $\overline{\text{OC1x}}$.

- **Bits 7:4- DT1H3:DT1H0: Dead Time Value for OC1x Output**

The dead time value for the OC1x 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- DT1L3:DT1L0: Dead Time Value for $\overline{\text{OC1x}}$ Output**

The dead time value for the $\overline{\text{OC1x}}$ 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.

17. USI – Universal Serial Interface

17.1 Features

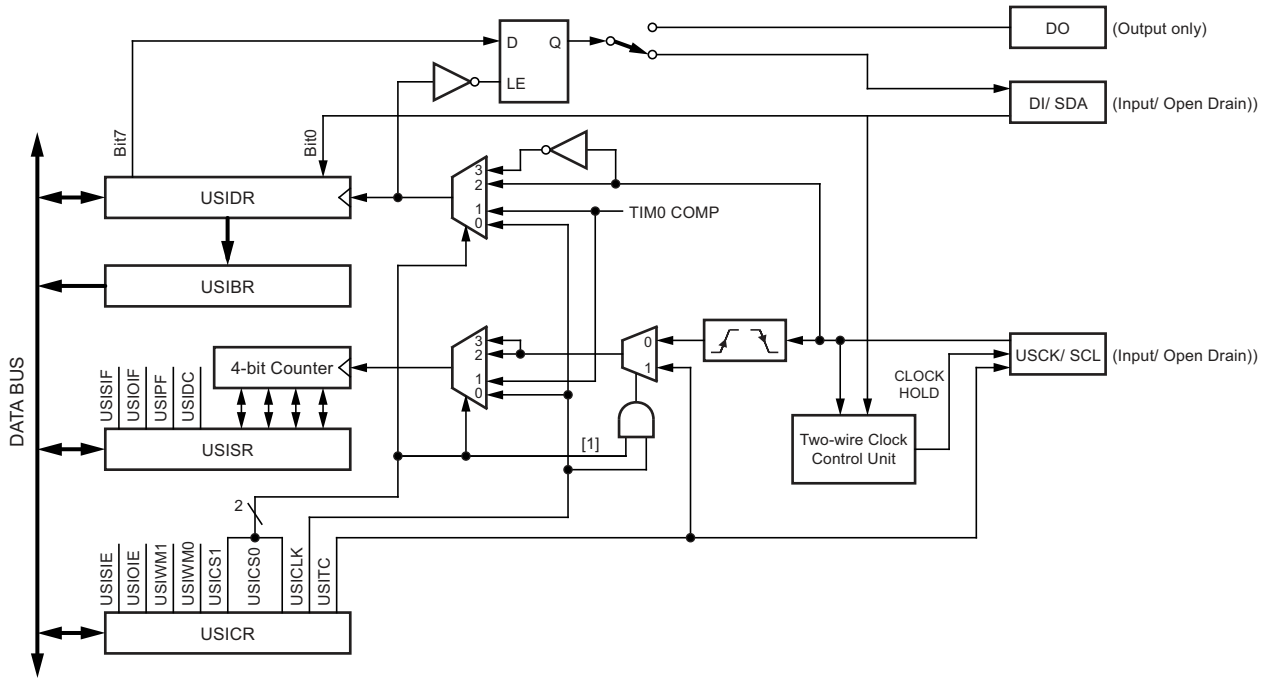
- Two-wire synchronous data transfer (master or slave)
- Three-wire synchronous data transfer (master or slave)
- Data received interrupt
- Wakeup from idle mode
- In two-wire mode: Wake-up from all sleep modes, including power-down mode
- Two-wire start condition detector with interrupt capability

17.2 Overview

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.

A simplified block diagram of the USI is shown on Figure 17-1 For the actual placement of I/O pins, refer to Section 1-1 “Pinout ATtiny261/461/861” on page 3. CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in Section 17.5 “Register Descriptions” on page 125.

Figure 17-1. Universal Serial Interface, Block Diagram



The 8-bit USI data 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 USI data register is a serial shift register and the most significant bit that is the output of the serial shift register is connected to one of two output pins depending of the wire mode configuration. A transparent latch is inserted between the USI data 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 USI data 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.

17.3 Functional Descriptions

17.3.1 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 17-2. Three-wire Mode Operation, Simplified Diagram

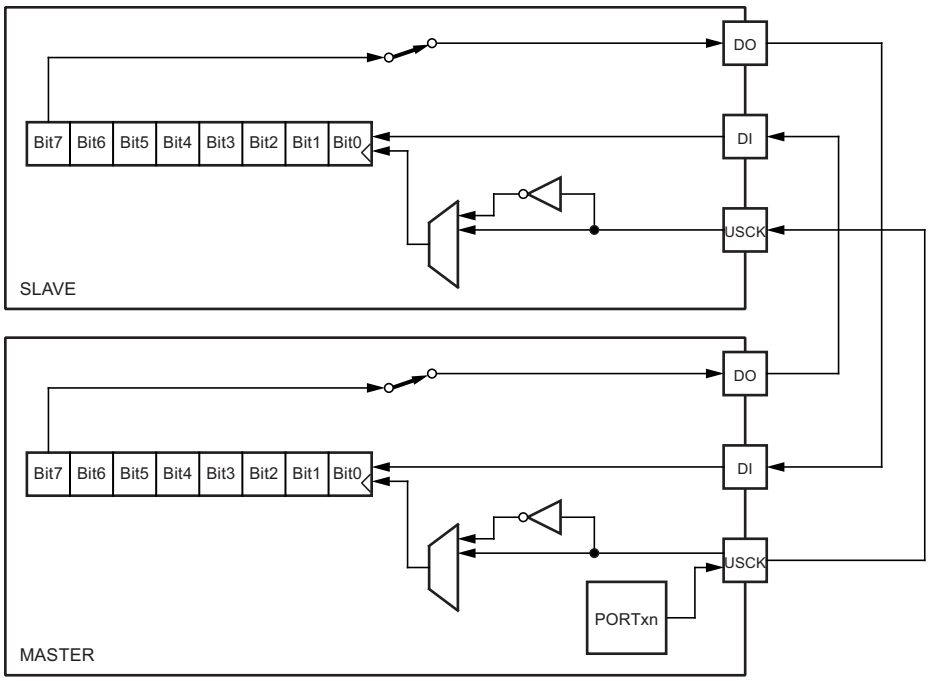
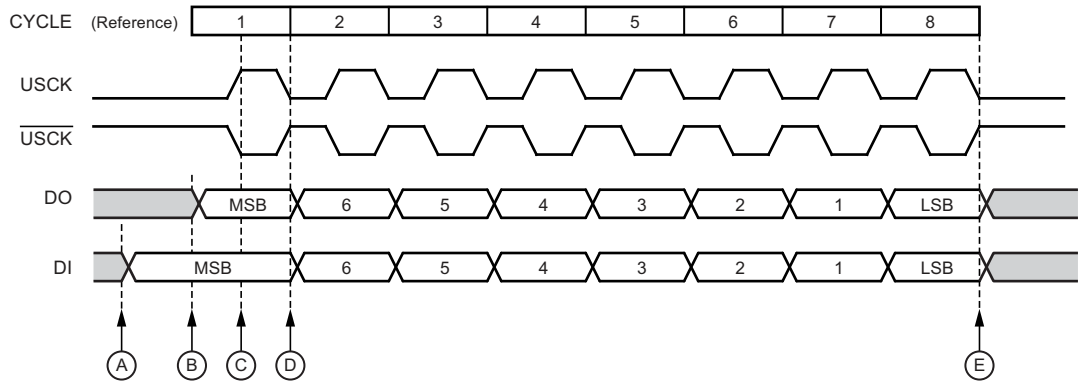


Figure 17-2 shows two USI units operating in three-wire mode, one as master and one as slave. The two USI data register are interconnected in such way that after eight USCK clocks, the data in each register are interchanged. The same clock also increments the USI's 4-bit counter. The counter overflow (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 17-3. Three-wire Mode, Timing Diagram



The three-wire mode timing is shown in Figure 17-3 At the top of the figure is a USCK cycle reference. One bit is shifted into the USI data 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 17-3), 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 USI 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.

17.3.2 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 DDRA or DDRB 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
```

17.3.3 SPI Slave Operation Example

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

```
init:
       ldi           r16,(1<<USIWM0)|(1<<USICS1)
       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 USI data register clock. The loop is repeated until the USI counter overflow flag is set.

17.3.4 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 17-4. Two-wire Mode Operation, Simplified Diagram

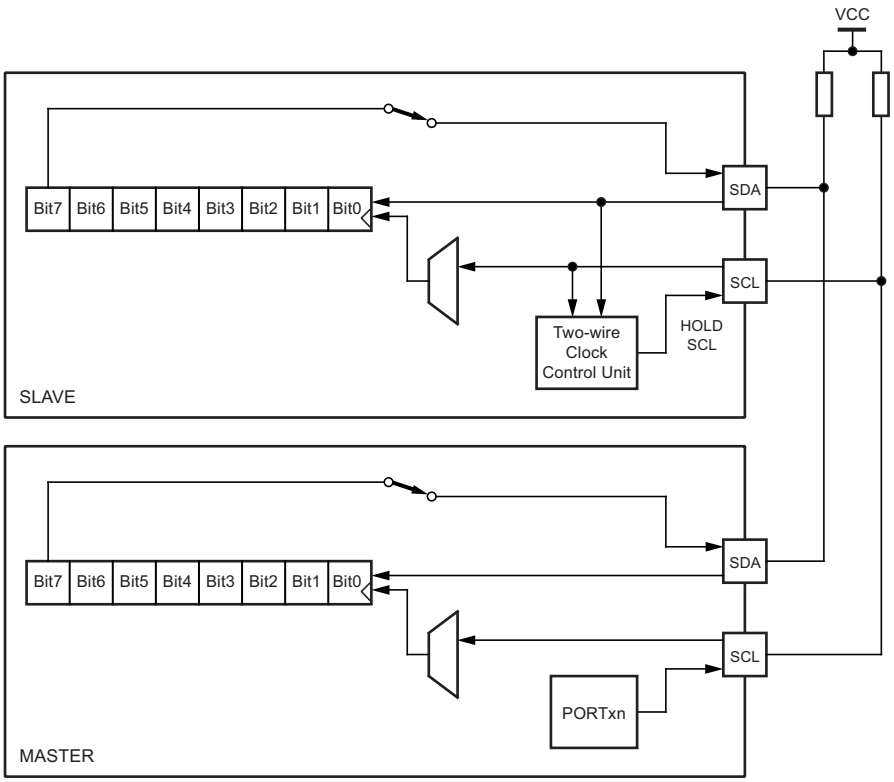
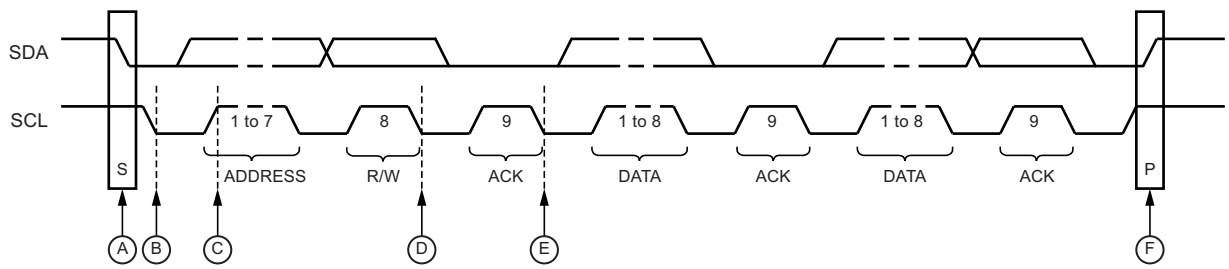


Figure 17-4 shows two USI units operating in two-wire mode, one as master and one as slave. It is only the physical layer that is shown since the system operation is highly dependent of the communication scheme used. The main differences between the master and slave operation at this level, is the serial 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 17-5. Two-wire Mode, Typical Timing Diagram

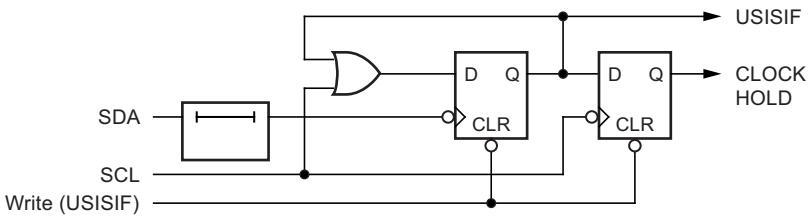


Referring to the timing diagram (Figure 17-5 on page 123), a bus transfer involves the following steps:

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 USI data register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 17-6) detects the start condition and sets the USISIF flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the master has forced 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 USI data 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 USI data 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 17-6. Start Condition Detector, Logic Diagram



17.3.5 Start Condition Detector

The start condition detector is shown in Figure 17-6 The SDA line is delayed (in the range of 50 to 300ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in two-wire mode.

The start condition detector is working asynchronously and can therefore wake up the processor from the power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case the oscillator start-up time set by the CKSEL fuses (see Section 7.1 "Clock Systems and their Distribution" on page 24) must also be taken into the consideration. Refer to the USISIF bit description on page 126 for further details.

17.4 Alternative USI Usage

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

17.4.1 Half-duplex Asynchronous Data Transfer

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

17.4.2 4-bit Counter

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

17.4.3 12-bit Timer/Counter

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

17.4.4 Edge Triggered External Interrupt

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

17.4.5 Software Interrupt

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

17.5 Register Descriptions

17.5.1 USIDR – USI Data Register

Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	**MSB**							**LSB**	**USIDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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 USI data 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 USI data register.

17.5.2 USIBR – USI Buffer Register

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

The content of the serial register is loaded to the USI buffer register when the transfer 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.

17.5.3 USISR – USI Status Register

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

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

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

When two-wire mode is selected, the USISIF flag is set (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 wake up the processor from Idle sleep mode.

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

When Two-wire mode is selected, the USIPF Flag is set (one) when a stop condition is detected. The flag is cleared by writing a 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 USI data 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.

17.5.4 USICR – USI Control Register

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

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

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

Setting this bit to one enables the start condition detector interrupt. If there is a pending interrupt when the USISIE and the global interrupt enable flag is set to one, this will immediately be executed. Refer to the USISIF bit description on page 126 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 126 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 USI data 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 17-1.

Table 17-1. 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 USI data 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 USI data 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 USI data 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 17-2 on page 128 shows the relationship between the USICS1..0 and USICLK setting and clock source used for the USI data register and the 4-bit counter.

Table 17-2. Relations between the USICS1..0 and USICLK Setting

USICS1	USICS0	USICLK	USI Data 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 USI data 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 USI data 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 17-2).

- **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 DDB2 must be set as output (to one). This feature allows easy clock generation when implementing master devices. The bit will be read as zero.

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

17.5.5 USIPP – USI Pin Position

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	-	-	-	-	-	-	-	**USIPOS**	**USIPP**
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny261/461/861 and always reads as zero.

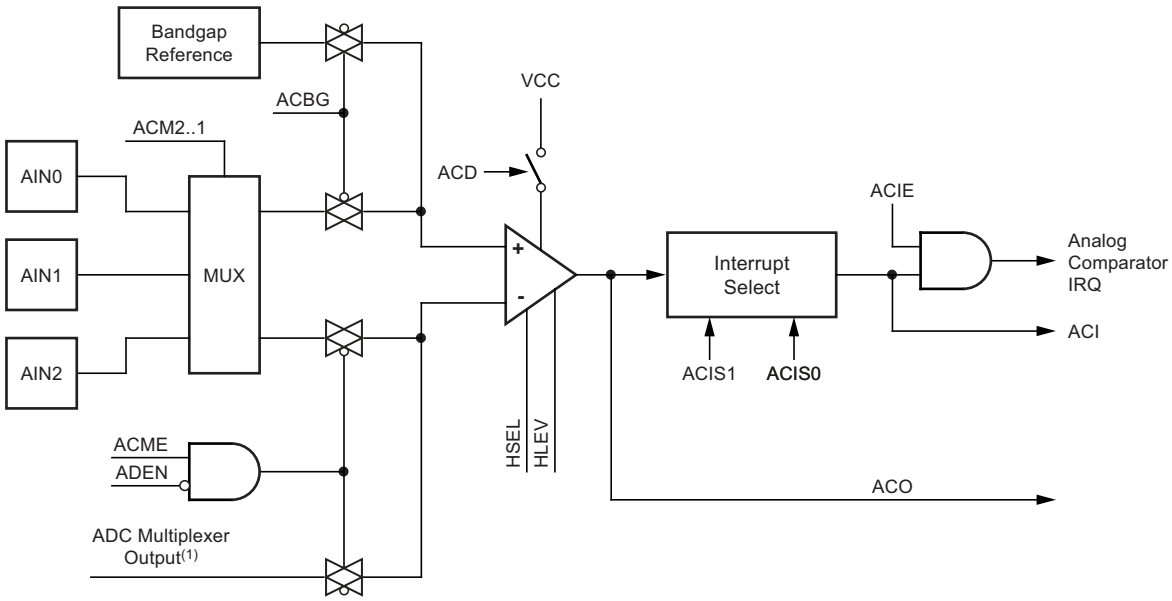
- **Bit 0 – USIPOS: USI Pin Position**

Setting this bit to one changes the USI pin position. As default pins PB2..PB0 are used for the USI pin functions, but when writing this bit to one the USIPOS bit is set the USI pin functions are on pins PA2..PA0.

18. AC – Analog Comparator

The analog comparator compares the input values on the selectable positive pin (AIN0, AIN1 or AIN2) and selectable negative pin (AIN0, AIN1 or AIN2). When the voltage on the positive pin is higher than the voltage on the negative pin, 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 18-1.

Figure 18-1. Analog Comparator Block Diagram(2)



Notes: 1. See Table 18-2 on page 130.

2. Refer to Figure 1-1 on page 3 and Table 12.3.2 on page 62 for analog comparator pin placement.

18.1 Register Description

18.1.1 ACSRA – Analog Comparator Control and Status Register A

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

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

When this bit is written 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 ACSRA. 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 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, AIN1 or AIN2 depending on the ACM2..0 bits 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 – 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 18-2 “Analog Comparator Multiplexed Input” on page 130.

- **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 18-1.

Table 18-1. 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.

18.2 Analog Comparator Multiplexed Input

When the analog to digital converter (ADC) is configured as single ended input channel, it is possible to select any of the ADC10..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), MUX5..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 18-2. If ACME is cleared or ADEN is set, either AIN0, AIN1 or AIN2 is applied to the negative input to the analog comparator.

Table 18-2. Analog Comparator Multiplexed Input

ACME	ADEN	MUX5..0	ACM2..0	Positive Input	Negative Input
0	x	xxxxxx	000	AIN0	AIN1
0	x	xxxxxx	001	AIN0	AIN2
0	x	xxxxxx	010	AIN1	AIN0
0	x	xxxxxx	011	AIN1	AIN2
0	x	xxxxxx	100	AIN2	AIN0
0	x	xxxxxx	101,110,111	AIN2	AIN1
1	1	xxxxxx	000	AIN0	AIN1
1	0	000000	000	AIN0	ADC0
1	0	000000	01x	AIN1	ADC0
1	0	000000	1xx	AIN2	ADC0
1	0	000001	000	AIN0	ADC1
1	0	000001	01x	AIN1	ADC1

Table 18-2. Analog Comparator Multiplexed Input (Continued)

ACME	ADEN	MUX5..0	ACM2..0	Positive Input	Negative Input
1	0	000001	1xx	AIN2	ADC1
1	0	000010	000	AIN0	ADC2
1	0	000010	01x	AIN1	ADC2
1	0	000010	1xx	AIN2	ADC2
1	0	000011	000	AIN0	ADC3
1	0	000011	01x	AIN1	ADC3
1	0	000011	1xx	AIN2	ADC3
1	0	000100	000	AIN0	ADC4
1	0	000100	01x	AIN1	ADC4
1	0	000100	1xx	AIN2	ADC4
1	0	000101	000	AIN0	ADC5
1	0	000101	01x	AIN1	ADC5
1	0	000101	1xx	AIN2	ADC5
1	0	000110	000	AIN0	ADC6
1	0	000110	01x	AIN1	ADC6
1	0	000110	1xx	AIN2	ADC6
1	0	000111	000	AIN0	ADC7
1	0	000111	01x	AIN1	ADC7
1	0	000111	1xx	AIN2	ADC7
1	0	001000	000	AIN0	ADC8
1	0	001000	01x	AIN1	ADC8
1	0	001000	1xx	AIN2	ADC8
1	0	001001	000	AIN0	ADC9
1	0	001001	01x	AIN1	ADC9
1	0	001001	1xx	AIN2	ADC9
1	0	001010	000	AIN0	ADC10
1	0	001010	01x	AIN1	ADC10
1	0	001010	1xx	AIN2	ADC10

18.2.1 ACSRB – Analog Comparator Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	**HSEL**	**HLEV**	**-**	**-**	**-**	**ACM2**	**ACM1**	**ACM0**	**ACSRB**
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – HSEL: Hysteresis Select**

When this bit is written logic one, the hysteresis of the analog comparator is switched on. The hysteresis level is selected by the HLEV bit.

- **Bit 6 – HLEV: Hysteresis Level**

When the hysteresis is enabled by the HSEL bit, the hysteresis level, HLEV, bit selects the hysteresis level that is either 20mV (HLEV=0) or 50mV (HLEV=1).

- **Bit 2:0 – ACM2:ACM0: Analog Comparator Multiplexer**

The analog comparator multiplexer bits select the positive and negative input pins of the analog comparator. The different settings are shown in Table 18-2.

19. ADC – Analog to Digital Converter

19.1 Features

- 10-bit resolution
- 1.0 LSB integral non-linearity
- ±2 LSB absolute accuracy
- 65 - 260µs conversion time
- Up to 15kSPS at maximum resolution
- 11 multiplexed single ended input channels
- 16 differential input pairs
- 15 differential input pairs 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 cancel
- Unipolar/bipolar input mode
- Input polarity reversal mode

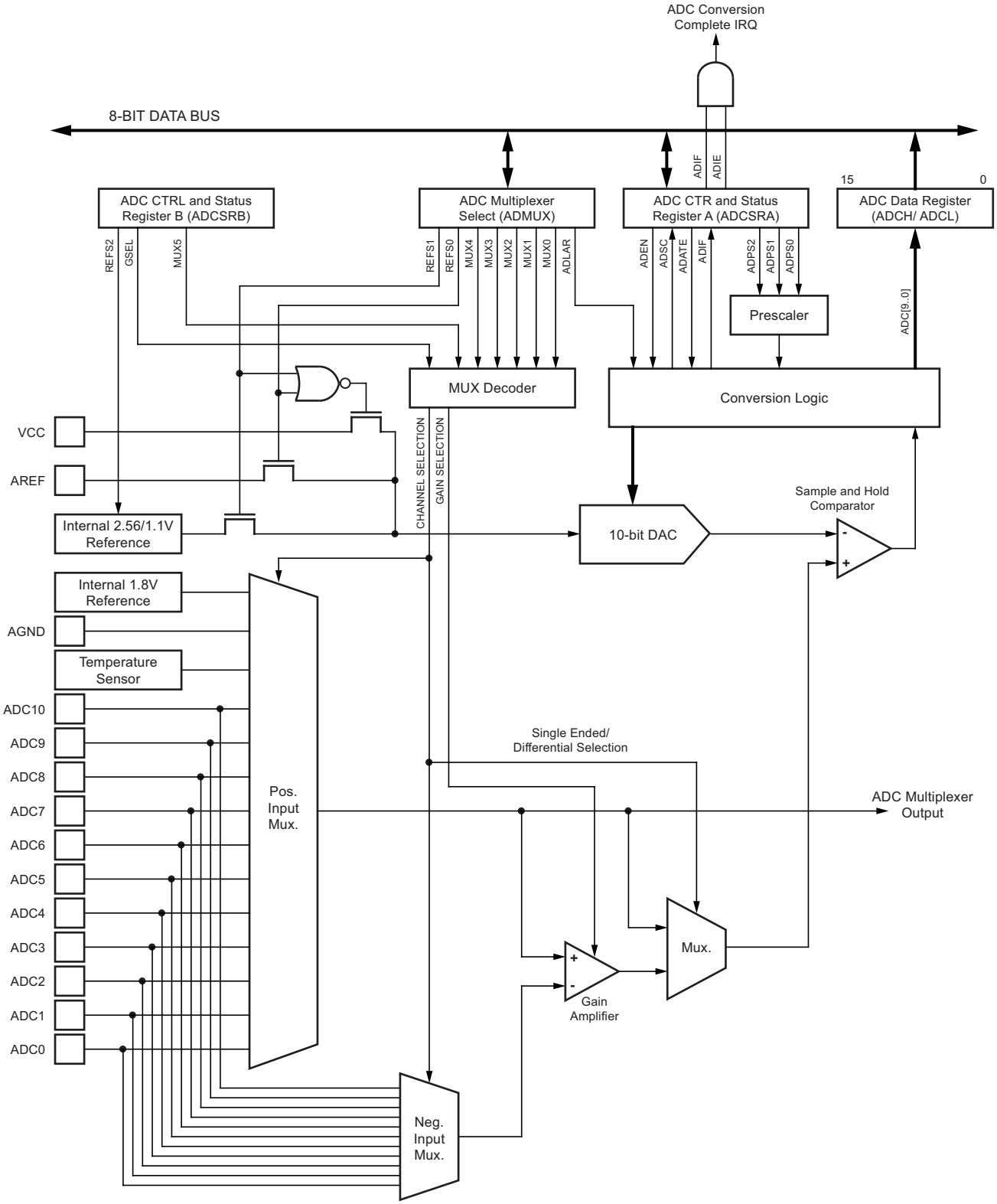
19.2 Overview

The ATtiny261/461/861 features a 10-bit successive approximation ADC. The ADC is connected to a 11-channel analog multiplexer which allows 16 differential voltage input combinations and 11 single-ended voltage inputs constructed from the pins PA7..PA0 or PB7..PB4. The differential input is equipped with a programmable gain stage, providing amplification steps of 1x, 8x, 20x or 32x 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 19-1 on page 133.

Internal reference voltages of nominally 1.1V or 2.56V are provided On-chip. The Internal reference voltage of 2.56V, can optionally be externally decoupled at the AREF (PA3) pin by a capacitor, for better noise performance. Alternatively, V_{CC} can be used as reference voltage for single ended channels. There is also an option to use an external voltage reference and turn-off the internal voltage reference. These options are selected using the REFS2:0 bits of the ADMUX control register.

Figure 19-1. Analog to Digital Converter Block Schematic



19.3 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 MUX5..0 bits in ADMUX. Any of the 11 ADC input pins ADC10..0 can be selected as single ended inputs to the ADC. The positive and negative inputs to the differential gain amplifier are described in Table 19-5 on page 145.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input pair by the selected gain factor, 1x, 8x, 20x or 32x, according to the setting of the MUX5..0 bits in ADMUX and the GSEL bit in ADCSRB. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

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

The on-chip temperature sensor is selected by writing the code “111111” to the MUX5..0 bits in ADMUX register when the ADC11 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.

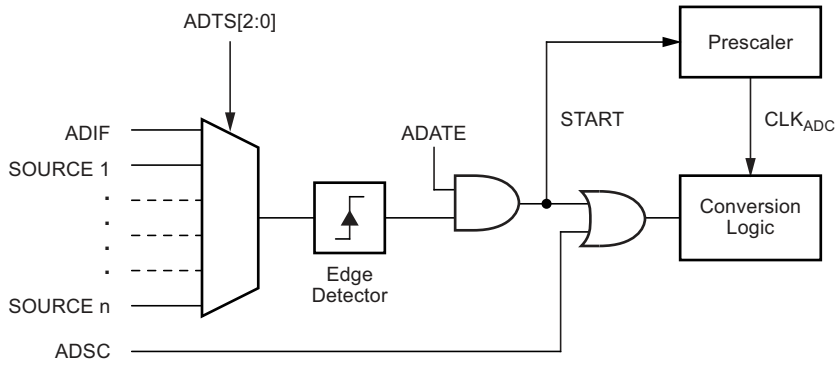
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

19.4 Starting a Conversion

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

Alternatively, a conversion can be triggered automatically by various sources. Auto triggering is enabled by setting the ADC auto trigger enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC trigger select bits, ADTS in ADCSRB (see 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 19-2. ADC Auto Trigger Logic

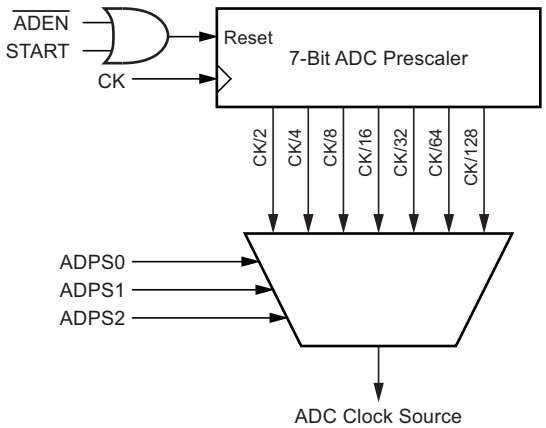


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the 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.

19.5 Prescaling and Conversion Timing

Figure 19-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

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

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

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

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 14.5 ADC clock cycles after the start of 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 19-1 on page 137.

Figure 19-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

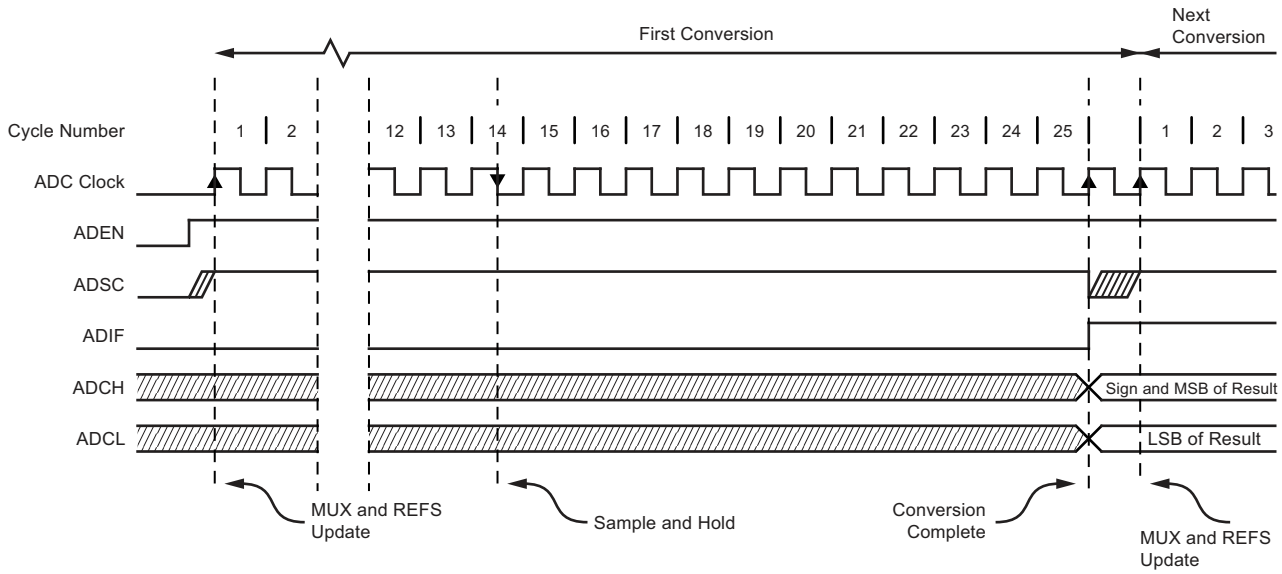


Figure 19-5. ADC Timing Diagram, Single Conversion

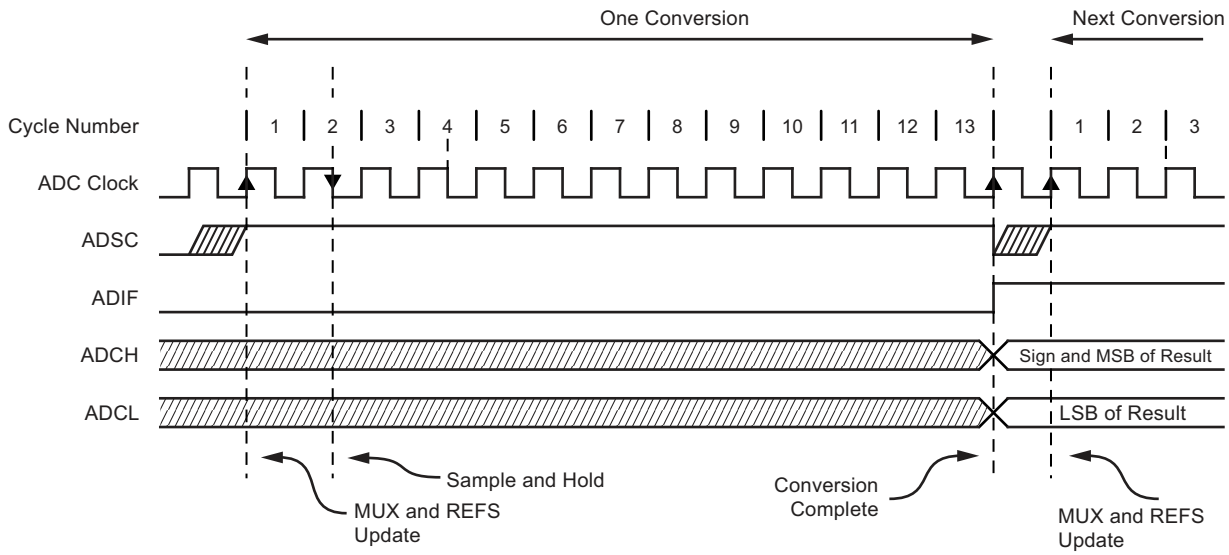


Figure 19-6. ADC Timing Diagram, Auto Triggered Conversion

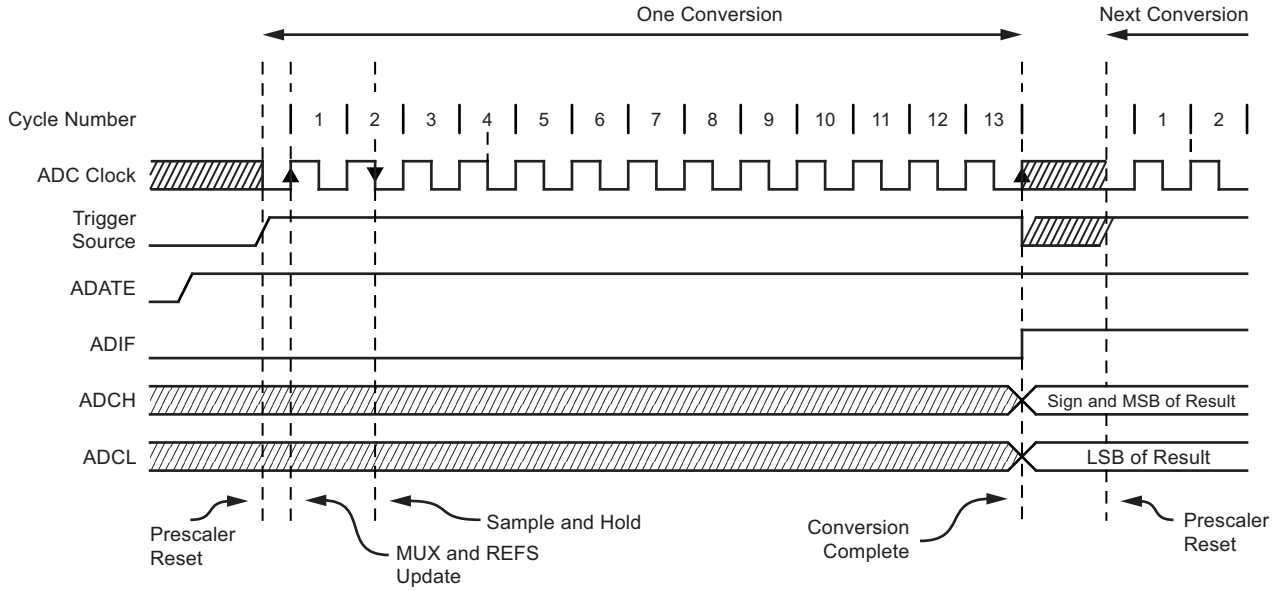


Figure 19-7. ADC Timing Diagram, Free Running Conversion

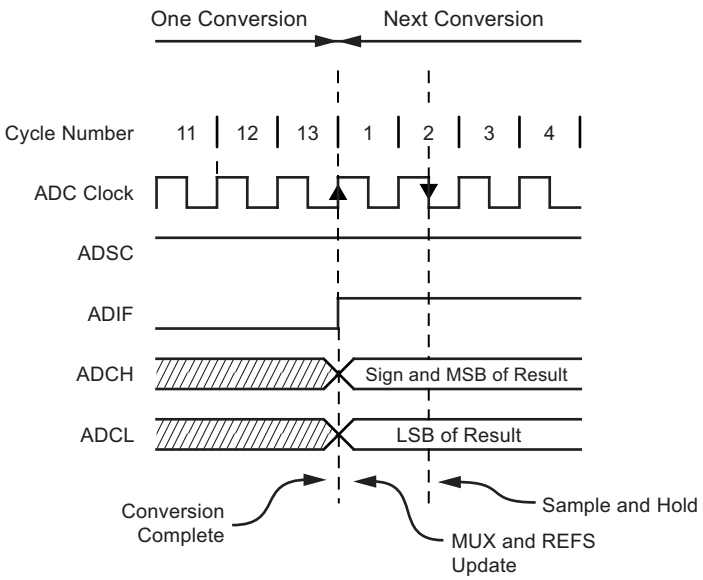


Table 19-1. ADC Conversion Time

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

19.6 Changing Channel or Reference Selection

The MUX5:0 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 reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set).

Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

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

19.6.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In single conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In free running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

19.6.2 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.

19.7 ADC Noise Canceler

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

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.

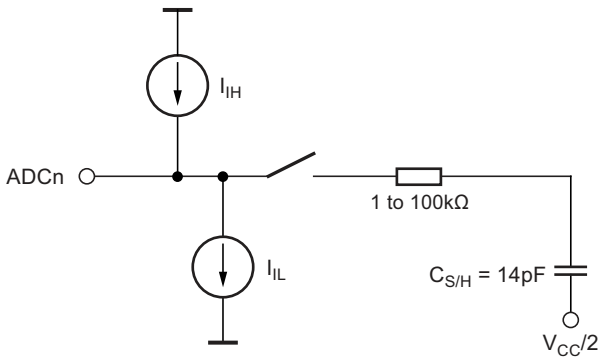
19.7.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 19-8 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 10kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedent 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 19-8. Analog Input Circuitry



19.7.2 Analog Noise Canceling Techniques

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

a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. Use the ADC noise 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.

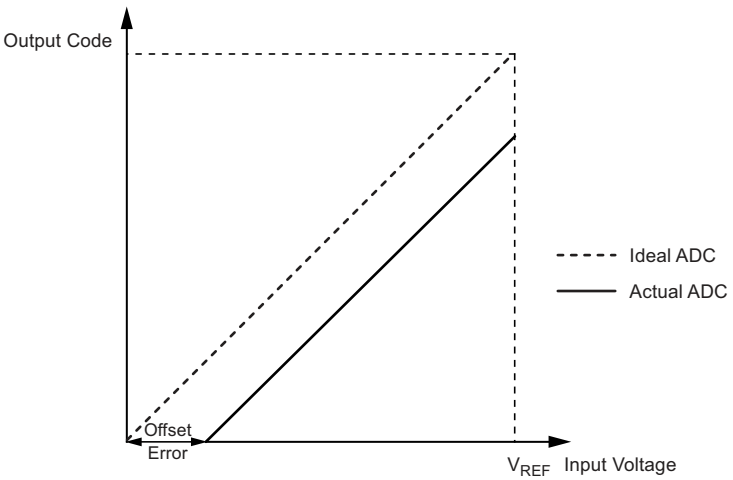
19.7.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n-1.

Several parameters describe the deviation from the ideal behavior:

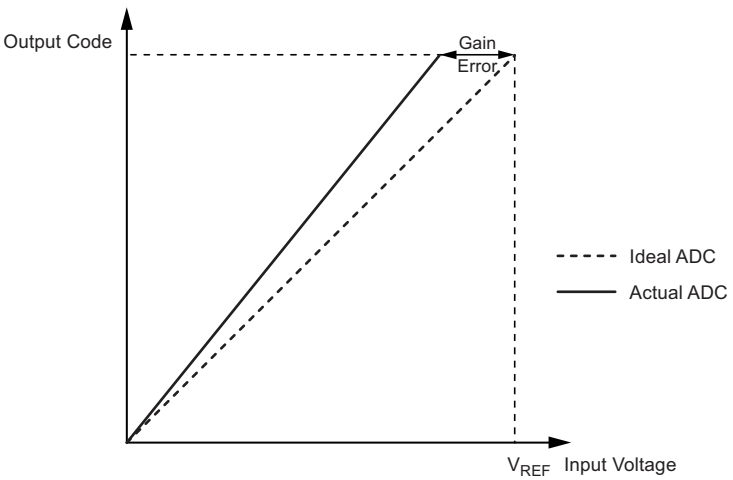
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 19-9. Offset Error



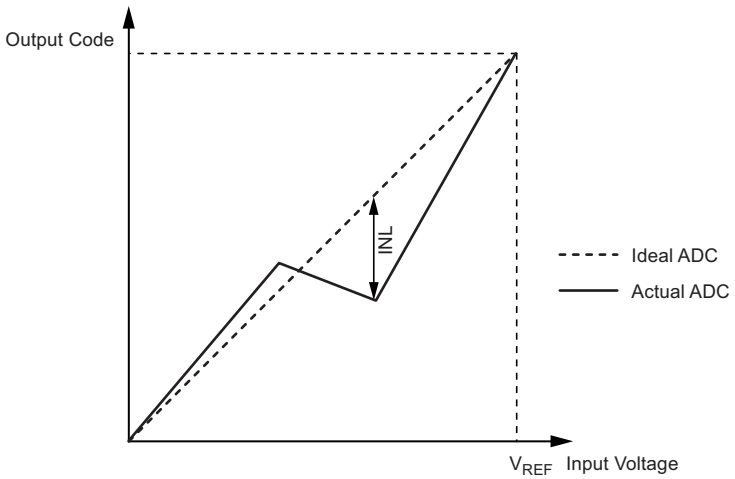
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

Figure 19-10. Gain Error



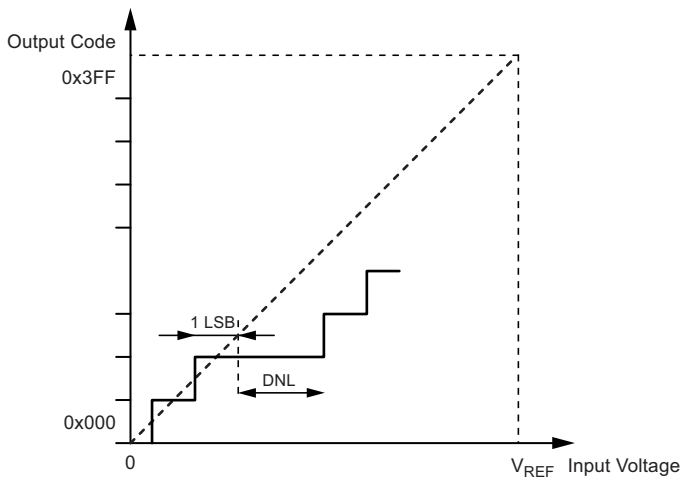
- Integral non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 19-11. 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 19-12. 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.

19.8 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversion as there are three types of conversions: single ended conversion, unipolar differential conversion and bipolar differential conversion.

19.8.1 Single Ended Conversion

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 19-4 on page 144 and Table 19-5 on page 145). 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.

19.8.2 Unipolar Differential Conversion

If differential channels and an unipolar input mode are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot 1024}{V_{REF}} \cdot GAIN$$

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 19-4 on page 144 and Table 19-5 on page 145). 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, 8x, 20x or 32x.

19.8.3 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

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot 512}{V_{REF}} \cdot GAIN$$

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, 8x, 20x or 32x.

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 pair. 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.

19.9 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC input. MUX[4..0] bits in ADMUX register 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 19-2. The voltage sensitivity is approximately 1LSB/°C and the accuracy of the temperature measurement is ±10°C using manufacturing calibration values (TS_GAIN, TS_OFFSET).

The values described in Table 19-2 are typical values. However, due to the process variation the temperature sensor output varies from one chip to another.

Table 19-2. Temperature versus Sensor Output Voltage (Typical Case): Example ADC Values

Temperature / °C	–40°C	+25 °C	+125 °C
	0x00F6	0x0144	0c01B8

19.9.1 Manufacturing Calibration

Calibration values determined during test are available in the signature row.

The temperature in degrees Celsius can be calculated using the formula:

$$T = \frac{([\langle ADCH << 8\rangle \ \ I\ ADCL] - \langle 273 + 25 - TS\_OFFSET\rangle) \times 128}{TS\_GAIN} + 25$$

Where:

a. ADCH & ADCL are the ADC data registers,

b. is the temperature sensor gain

c. TS_OFFSET is the temperature sensor offset correction term

TS_GAIN is the unsigned fixed point 8-bit temperature sensor gain factor in 1/128th units stored in the signature row.

TS_OFFSET is the signed twos complement temperature sensor offset reading stored in the signature row.

The table below summarizes the parameter signature row address vs product.

Table 19-3. Parameter Signature Row Address versus Product

	ATtiny261	ATtiny461	ATtiny861
TS_OFFSET	0x1F	0x05	0x05
TS_GAIN	0x1E	0x07	0x07

The following code example allows to read Signature Row data

```
.equ TS_GAIN = 0x0007
.equ TS_OFFSET = 0x0005
       LDI R30,LOW(TS_GAIN)
       LDI R31,HIGH (TS_GAIN)
       RCALL Read_signature_row
       MOV R17,R16                      ; Save R16 result
       LDI R30,LOW(TS_OFFSET)
       LDI R31,HIGH (TS_OFFSET)
       RCALL Read_signature_row
       ; R16 holds TS_OFFSET and R17 holds TS_GAIN
       Read_signature_row:
       IN R16,SPMCSR                    ; Wait for SPMEN ready
       SBRC R16,SPMEN                   ; Exit loop here when SPMCSR is free
       RJMP Read_signature_row
       LDI R16,((1<<SIGRD)|(1<<SPMEN))  ; We need to set SIGRD and SPMEN together
       OUT SPMCSR,R16                   ; and execute the LPM within 3 cycles
       LPM R16,Z
       RET
```

19.10 Register Description

19.10.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:REFS0: Voltage Reference Selection Bits**

These bits and the REFS2 bit from the ADC control and status register B (ADCSRB) select the voltage reference for the ADC, as shown in Table 19-4. 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 AVCC or an external AREF higher than (AVCC - 1V) is not recommended, as this will affect ADC accuracy. The internal voltage reference options may not be used if an external voltage is being applied to the AREF pin.

Table 19-4. Voltage Reference Selections for ADC

REFS2	REFS1	REFS0	Voltage Reference (V_{REF}) Selection
X	0	0	V_{CC} used as voltage reference, disconnected from AREF
X	0	1	External voltage reference at AREF pin, internal voltage reference turned off
0	1	0	Internal 1.1V voltage reference
0	1	1	Reserved
1	1	0	Internal 2.56V voltage reference without external bypass capacitor, disconnected from AREF
1	1	1	Internal 2.56V voltage reference with external bypass capacitor at AREF pin

- **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 19.10.3 “ADCL and ADCH – The ADC Data Register” on page 147.

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

These bits and the MUX5 bit from the ADC control and status register B (ADCSRB) select which combination of analog inputs are connected to the ADC. In case of differential input, gain selection is also made with these bits. Selecting the same pin as both inputs to the differential gain stage enables offset measurements. Selecting the single-ended channel ADC11 enables the temperature sensor. Refer to Table 19-5 on page 145 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 19-5. Input Channel Selections

MUX5..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0 (PA0)	NA	NA	NA
000001	ADC1 (PA1)			
000010	ADC2 (PA2)			
000011	ADC3 (PA4)			
000100	ADC4 (PA5)			
000101	ADC5 (PA6)			
000110	ADC6 (PA7)			
000111	ADC7 (PB4)			
001000	ADC8 (PB5)			
001001	ADC9 (PB6)			
001010	ADC10 (PB7)			
001011	NA	ADC0 (PA0)	ADC1 (PA1)	20x
001100		ADC0 (PA0)	ADC1 (PA1)	1x
001101		ADC1 (PA1)	ADC1 (PA1)	20x
001110		ADC2 (PA2)	ADC1 (PA1)	20x
001111		ADC2 (PA2)	ADC1 (PA1)	1x
010000	N/A	ADC2 (PA2)	ADC3 (PA4)	1x
010001		ADC3 (PA4)	ADC3 (PA4)	20x
010010		ADC4 (PA5)	ADC3 (PA4)	20x
010011		ADC4 (PA5)	ADC3 (PA4)	1x
010100	NA	ADC4 (PA5)	ADC5 (PA6)	20x
010101		ADC4 (PA5)	ADC5 (PA6)	1x
010110		ADC5 (PA6)	ADC5 (PA6)	20x
010111		ADC6 (PA7)	ADC5 (PA6)	20x
011000		ADC6 (PA7)	ADC5 (PA6)	1x
011001	NA	ADC8 (PB5)	ADC9 (PB6)	20x
011010		ADC8 (PB5)	ADC9 (PB6)	1x
011011		ADC9 (PB6)	ADC9 (PB6)	20x
011100		ADC10 (PB7)	ADC9 (PB6)	20x
011101		ADC10 (PB7)	ADC9 (PB6)	1x
011110	1.1V	N/A	N/A	N/A
011111	0V			
100000	N/A	ADC0(PA0)	ADC1(PA1)	20x/32x
100001		ADC0(PA0)	ADC1(PA1)	1x/8x
100010		ADC1(PA1	ADC0(PA0)	20x/32x
100011		ADC1(PA1)	ADC0(PA0)	1x/8x
100100	N/A	ADC1(PA1)	ADC2(PA2)	20x/32x
100101		ADC1(PA1)	ADC2(PA2)	1x/8x
100110		ADC2(PA2	ADC1(PA1)	20x/32x
100111		ADC2(PA2)	ADC1(PA1)	1x/8x

Note: 1. For temperature sensor

Table 19-5. Input Channel Selections (Continued)

MUX5..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
101000	N/A	ADC2(PA2)	ADC0(PA0)	20x/32x
101001		ADC2(PA2)	ADC0(PA0)	1x/8x
101010		ADC0(PA0)	ADC2(PA2)	20x/32x
101011		ADC0(PA0)	ADC2(PA2)	1x/8x
101100	N/A	ADC4(PA5)	ADC5(PA6)	20x/32x
101101		ADC4(PA5)	ADC5(PA6)	1x/8x
101110		ADC5(PA6)	ADC4(PA5)	20x/32x
101111		ADC5(PA6)	ADC4(PA5)	1x/8x
110000	N/A	ADC5(PA6)	ADC6(PA7)	20x/32x
110001		ADC5(PA6)	ADC6(PA7)	1x/8x
110010		ADC6(PA7)	ADC5(PA6)	20x/32x
110011		ADC6(PA7)	ADC5(PA6)	1x/8x
110100	N/A	ADC6(PA7)	ADC4(PA5)	20x/32x
110101		ADC6(PA7)	ADC4(PA5)	1x/8x
110110		ADC4(PA5)	ADC6(PA7)	20x/32x
110111		ADC4(PA5)	ADC6(PA7)	1x/8x
111000	N/A	ADC0(PA0)	ADC0(PA0)	20x/32x
111001		ADC0(PA0)	ADC0(PA0)	1x/8x
111010		ADC1(PA1)	ADC1(PA1)	20x/32x
111011		ADC2(PA2)	ADC2(PA2)	20x/32x
111100	N/A	ADC4(PA5)	ADC4(PA5)	20x/32x
111101		ADC5(PA6)	ADC5(PA6)	20x/32x
111110		ADC6(PA7)	ADC6(PA7)	20x/32x
111111	ADC11	N/A	N/A	N/A

Note: 1. For temperature sensor

19.10.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	**ADEN**	**ADSC**	**ADATE**	**ADIF**	**ADIE**	**ADPS2**	**ADPS1**	**ADPS0**	**ADCSRA**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to 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 19-6. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

19.10.3 ADCL and ADCH – The ADC Data Register

19.10.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	–	–	–	–	–	–	**ADC9**	**ADC8**	**ADCH**
0x04 (0x24)	**ADC7**	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**	**ADC1**	**ADC0**	**ADCL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

19.10.3.2ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x04 (0x24)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC data register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in 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 19.8 “ADC Conversion Result” on page 142.

19.10.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	BIN	GSEL	-	REFS2	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– 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.

- **Bits 6 – GSEL: Gain Select**

The gain select bit selects the 32x gain instead of the 20x gain and the 8x gain instead of the 1x gain when the gain select bit is written to one.

- **Bits 5 – Res: Reserved Bit**

This bit is a reserved bit in the ATtiny261/461/861 and will always read as zero.

- **Bits 4 – REFS2: Reference Selection Bit**

These bit selects either the voltage reference of 1.1 V or 2.56 V for the ADC, as shown in Table 19-4. If active channels are used, using AVCC or an external AREF higher than (AVCC - 1V) is not recommended, as this will affect ADC accuracy. The internal voltage reference options may not be used if an external voltage is being applied to the AREF pin.

- **Bits 3 – MUX5: Analog Channel and Gain Selection Bit 5**

The MUX5 bit is the MSB of the analog channel and gain selection bits. Refer to Table 19-5 for details. If this bit is changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSRA is set).

- **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 19-7. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free running mode
0	0	1	Analog comparator
0	1	0	External interrupt request 0
0	1	1	Timer/Counter0 compare match A
1	0	0	Timer/Counter0 overflow
1	0	1	Timer/Counter0 compare match B
1	1	0	Timer/Counter1 overflow
1	1	1	Watchdog interrupt request

19.10.5 DIDR0 – Digital Input Disable Register 0

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

- **Bits 7:4,2:0 – ADC6D:ADC0D: ADC6:0 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 ADC7:0 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.

- **Bit 3 – AREFD: AREF Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AREF 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 AREF 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.

19.10.6 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	ADC10D	ADC9D	ADC8D	ADC7D	-				DIDR1
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..4 – ADC10D..ADC7D: ADC10..7 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 ADC10:7 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.

20. debugWIRE On-chip Debug System

20.1 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

20.2 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.

20.3 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 20-1. The debugWIRE Setup

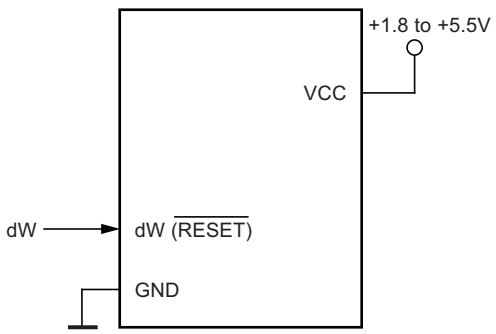


Figure 20-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

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 20 kΩ. 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.

20.4 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.

20.5 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.

20.6 Register Description

The following section describes the registers used with the debugWire.

20.6.1 DWDR – debugWire Data Register

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	**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.

21. 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.

21.1 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.

21.2 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.

21.3 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.

21.4 Addressing the Flash During Self-Programming

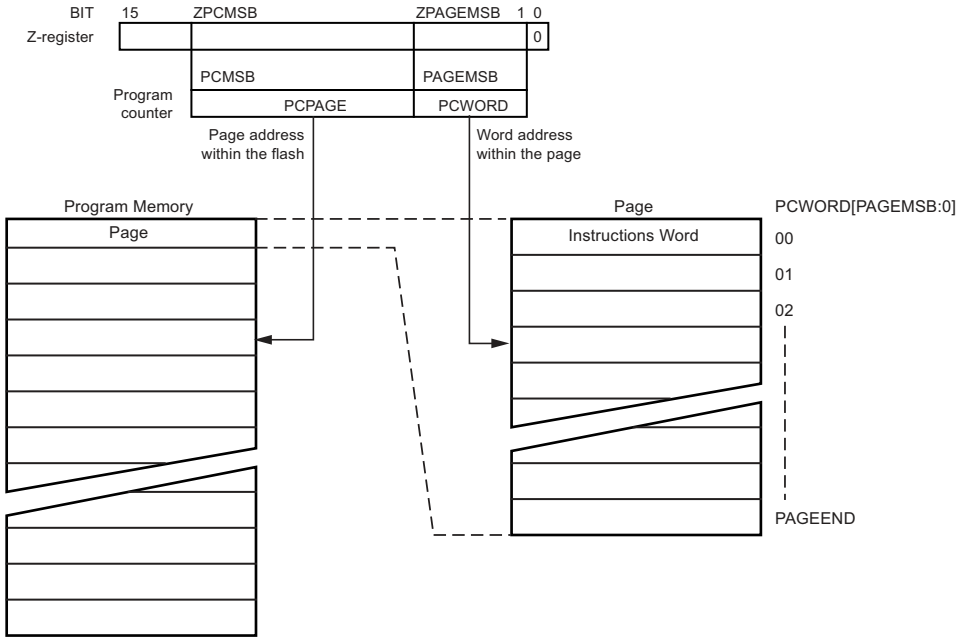
The Z-pointer is used to address the SPM commands.

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 22-7 on page 159), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 21-1. 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 21-1. Addressing the Flash During SPM(1)



Note: 1. The different variables used in Figure 21-1 are listed in Table 22-7 on page 159.

21.4.1 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to flash. Reading the fuses and lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR register and verifies that the bit is cleared before writing to the SPMCSR register.

21.4.2 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 22-5 on page 158 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 22-4 on page 157 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.

21.4.3 Preventing Flash Corruption

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

21.4.4 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 21-1 shows the typical programming time for flash accesses from the CPU.

Table 21-1. SPM Programming Time(1)

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7ms	4.5ms

Note: 1. Minimum and maximum programming time is per individual operation.

21.5 Register Description

21.5.1 SPMCSR – Store Program Memory Control and Status Register

The store program memory control and status register contains the control bits needed to control the program memory operations.

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	**–**	**–**	**SIGRD**	**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 7:6 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny261/461/861 and always read as zero.

- **Bit 5 – SIGRD: Signature Row Read**

If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register for details. An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect.

- **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 21.4.1 “EEPROM Write Prevents Writing to SPMCSR” on page 153 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.

22. Memory Programming

This section describes the different methods for Programming the ATtiny261/461/861 memories.

22.1 Program And Data Memory Lock Bits

The ATtiny261/461/861 provides two lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional security listed in Table 22-2. The lock bits can only be erased to “1” with the chip erase command. The ATtiny261/461/861 has no separate boot loader section. The SPM instruction is enabled for the whole flash, if the SELFPROGEN fuse is programmed (“0”), otherwise it is disabled.

Table 22-1. 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 22-2. 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).
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).

Notes: 1. Program the fuse bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

22.2 Fuse Bytes

The ATtiny261/461/861 has three fuse bytes. Table 22-3, Table 22-4 and Table 22-5 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 22-3. 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 22-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External reset disable	1 (unprogrammed)
DWEN(2)	6	DebugWIRE enable	1 (unprogrammed)
SPIEN(3)	6	Enable serial program and data downloading	0 (programmed, SPI prog. enabled)
WDTON(4)	4	Watchdog timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the chip erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(5)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1(5)	1	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL0(5)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes:
1. See Section 12.3.1 “Alternate Functions of Port B” on page 59 for description of RSTDISBL and DWEN fuses.
2. DWEN must be unprogrammed when lock bit security is required. See Section 22.1 “Program And Data Memory Lock Bits” on page 156.
3. The SPIEN fuse is not accessible in SPI programming mode.
4. See Section 9.10.2 “WDTCR – Watchdog Timer Control Register” on page 44 for details.
5. See Table 23-4 on page 174 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 22-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8(1)	7	Divide clock by 8	0 (programmed)
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 7.11 “System Clock Prescaler” on page 31 for details.
2. The CKOUT Fuse allows the system clock to be output on PORTB5. See Section 7.10 “Clock Output Buffer” on page 31 for details.
3. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 7-7 on page 28 for details.
4. The default setting of CKSEL3..0 results in internal RC oscillator at 8.0MHz. See Table 7-6 on page 28 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.

22.2.1 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.

22.3 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. The ATtiny261/461/861signature bytes are given in Table 22-6.

Table 22-6. Device ID

Parts	Signature Bytes Address		
	0x000	0x001	0x002
ATtiny261	0x1E	0x91	0x0C
ATtiny461	0x1E	0x92	0x08
ATtiny861	0x1E	0x93	0x0D

22.4 Calibration Byte

Signature area of the ATtiny261/461/861 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.

22.5 Reading the Signature Row from Software

To read the signature row from software, load the Z-pointer with the signature byte address given in Table 22-6 on page 158 and set the SIGRD and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the signature row lock bits or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction set manual.

Note: Before attempting to set SPMEN it is important to test this bit is cleared showing that the hardware is ready for a new operation.

22.6 Page Size

Table 22-7. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATtiny261	1K words (2K bytes)	16 words	PC[3:0]	64	PC[9:4]	9
ATtiny461	2K words (4K bytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATtiny861	4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11

Table 22-8. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATtiny261	128 bytes	2 bytes	EEA[1:0]	64	EEA[6:2]	6
ATtiny461	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7
ATtiny861	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

22.7 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify flash program memory, EEPROM Data memory, memory lock bits, and fuse bits in the ATtiny261/461/861. Pulses are assumed to be at least 250 ns unless otherwise noted.

22.7.1 Signal Names

In this section, some pins of the ATtiny261/461/861 are referenced by signal names describing their functionality during parallel programming, see Figure 22-1 on page 160 and Table 22-9 on page 160. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 22-11 on page 160.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 22-12 on page 161.

Figure 22-1. Parallel Programming

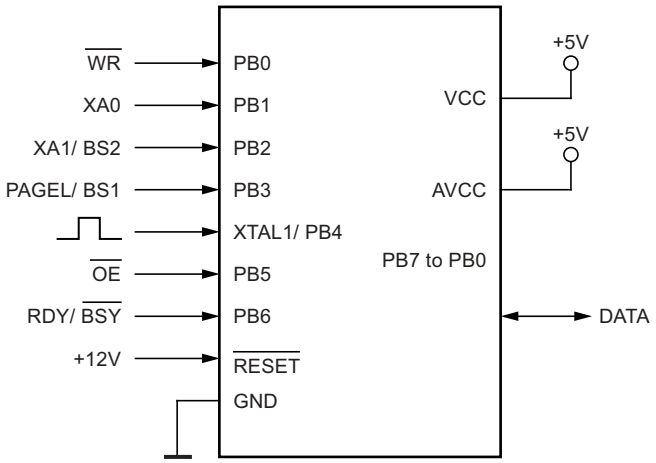


Table 22-9. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
$\overline{WR}$	PB0	I	Write pulse (active low).
XA0	PB1	I	XTAL action bit 0
XA1/BS2	PB2	I	XTAL action bit 1. Byte select 2 (“0” selects low byte, “1” selects 2’nd high byte).
PAGEL/BS1	PB3	I	Byte select 1 (“0” selects low byte, “1” selects high byte). Program memory and EEPROM data page load.
$\overline{OE}$	PB5	I	Output enable (active low).
RDY/$\overline{BSY}$	PB6	O	0: Device is busy programming, 1: Device is ready for new command.
DATA I/O	PA7-PA0	I/O	Bi-directional data bus (output when $\overline{OE}$ is low).

Table 22-10. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PAGEL/BS1	Prog_enable[3]	0
XA1/BS2	Prog_enable[2]	0
XA0	Prog_enable[1]	0
WR	Prog_enable[0]	0

Table 22-11. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load flash or EEPROM address (high or low address byte determined by BS1).
0	1	Load data (high or low data byte for flash determined by BS1).
1	0	Load command
1	1	No action, idle

Table 22-12. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip erase
0100 0000	Write fuse bits
0010 0000	Write lock bits
0001 0000	Write flash
0001 0001	Write EEPROM
0000 1000	Read signature bytes and calibration byte
0000 0100	Read fuse and lock bits
0000 0010	Read flash
0000 0011	Read EEPROM

22.8 Parallel Programming

22.8.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND.
2. Set $\overline{\text{RESET}}$ to “0” and toggle XTAL1 at least six times.
3. Set the Prog_enable pins listed in Table 22-10 on page 160 to “0000” and wait at least 100ns.
4. Apply 11.5 - 12.5V to $\overline{\text{RESET}}$. Any activity on Prog_enable pins within 100ns after +12V has been applied to $\overline{\text{RESET}}$, will cause the device to fail entering programming mode.
5. Wait at least 50µs before sending a new command.

22.8.2 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.

22.8.3 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 reprogrammed.

Note: 1. The EEPRPOM memory is preserved during chip erase if the EESAVE fuse is programmed.

Load command “Chip Erase”

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

22.8.4 Programming the Flash

The flash is organized in pages, see Table 22-7 on page 159. 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:

A. Load command “Write Flash”

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

B. Load address low byte

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

C. Load data low byte

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

D. Load data high byte

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

1. E. Latch Data
2. Set BS1 to “1”. This selects high data byte.
3. Give PAGEL a positive pulse. This latches the data bytes. (See Figure 22-3 for signal waveforms)

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 22-2 on page 163. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a page write.

G. Load address high byte

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

H. Program page

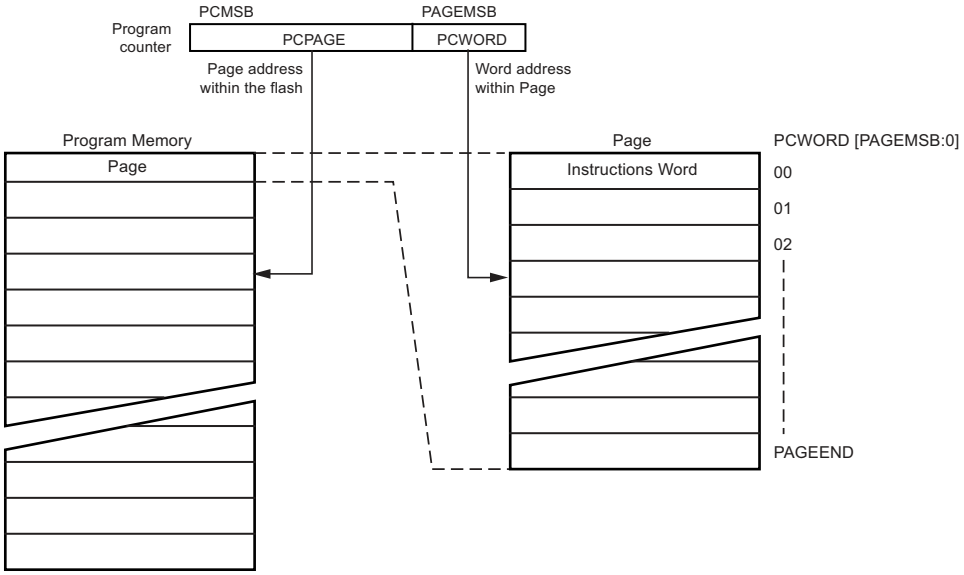
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (see Figure 22-3 for signal waveforms).

I. Repeat B through H until the entire flash is programmed or until all data has been programmed.

J. End page programming

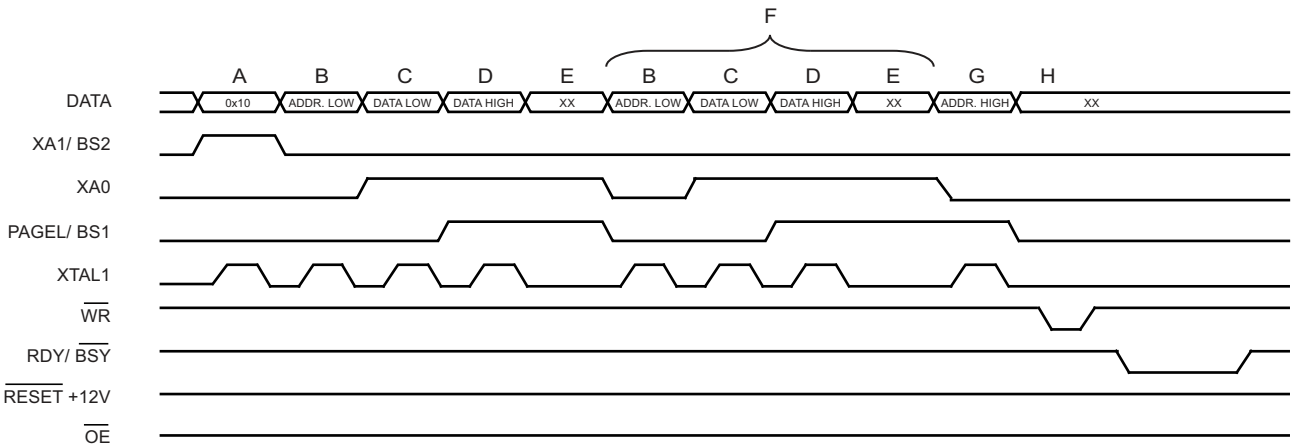
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for no operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 22-2. Addressing the Flash Which is Organized in Pages(1)



Note: 1. PCPAGE and PCWORD are listed in Table 22-7 on page 159.

Figure 22-3. Programming the Flash Waveforms(1)



Note: 1. "XX" is don't care. The letters refer to the programming description above.

22.8.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 22-8 on page 159. When programming the EEPROM, the program 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 Section 22.8.4 "Programming the Flash" on page 162 for details on command, address and data loading):

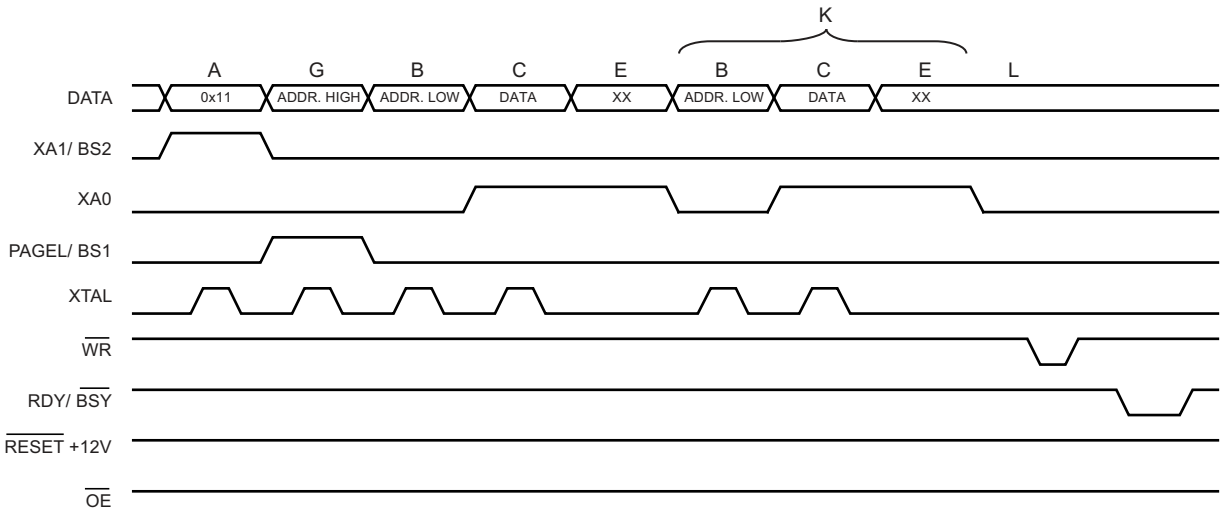
1. A: Load command "0001 0001".
2. G: Load address high byte (0x00 - 0xFF).
3. B: Load address low byte (0x00 - 0xFF).
4. C: Load data (0x00 - 0xFF).
5. E: Latch data (give PAGEL a positive pulse).

K: Repeat 3 through 5 until the entire buffer is filled.

L: Program EEPROM page

1. Set BS to “0”.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 22-4 for signal waveforms).

Figure 22-4. Programming the EEPROM Waveforms



22.8.6 Reading the Flash

The algorithm for reading the flash memory is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and address loading):

1. A: Load command “0000 0010”.
2. G: Load address high byte (0x00 - 0xFF).
3. B: Load address low byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The flash word low byte can now be read at DATA.
5. Set BS to “1”. The flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

22.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and address loading):

1. A: Load command “0000 0011”.
2. G: Load address high byte (0x00 - 0xFF).
3. B: Load address low byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

22.8.8 Programming the Fuse Low Bits

The algorithm for programming the fuse low bits is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and data loading):

1. A: Load command “0100 0000”.
2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

22.8.9 Programming the Fuse High Bits

The algorithm for programming the fuse high bits is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and data loading):

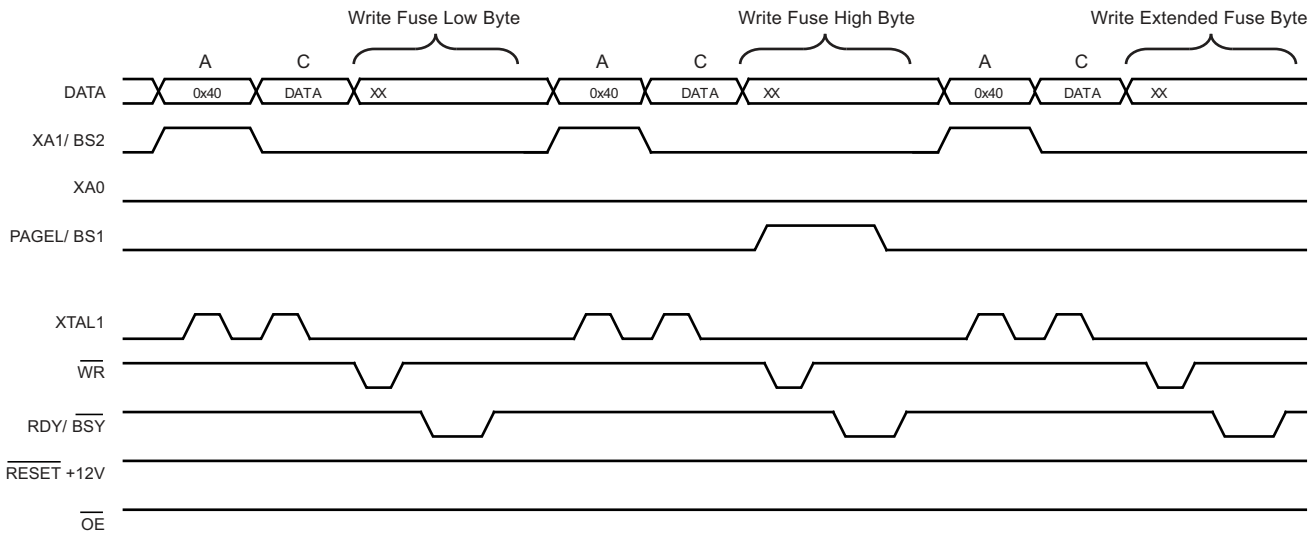
1. A: Load command “0100 0000”.
2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

22.8.10 Programming the Extended Fuse Bits

The algorithm for programming the extended fuse bits is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and data loading):

1. 1. A: Load command “0100 0000”.
2. 2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. 3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. 4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. 5. Set BS2 to “0”. This selects low data byte.

Figure 22-5. Programming the FUSES Waveforms



22.8.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and data loading):

1. A: Load command “0010 0000”.
2. C: Load data low byte. Bit n = “0” programs the lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the boot lock bits by any external programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

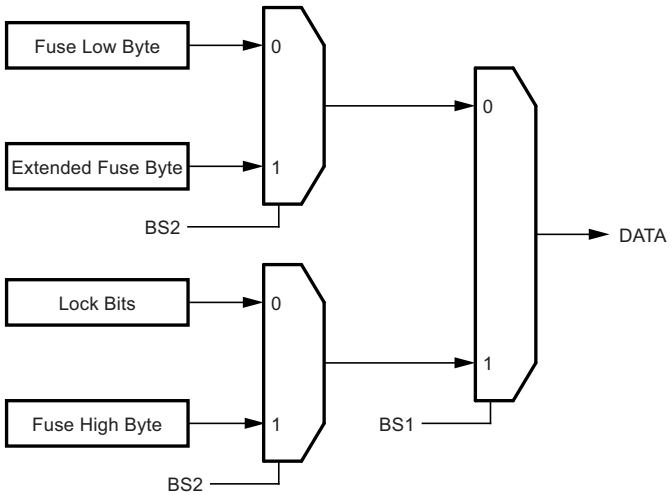
The lock bits can only be cleared by executing chip erase.

22.8.12 Reading the Fuse and Lock Bits

The algorithm for reading the fuse and lock bits is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command loading):

1. A: Load command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the fuse low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the fuse high bits can now be read at DATA (“0” means programmed).
4. Set OE to “0”, BS2 to “1”, and BS1 to “0”. The status of the extended fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 22-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



22.8.13 Reading the Signature Bytes

The algorithm for reading the signature bytes is as follows (refer to Section 22.8.4 “Programming the Flash” on page 162 for details on command and address loading):

1. A: Load command “0000 1000”.
2. B: Load address low byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0”, and BS to “0”. The selected signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

22.8.14 Reading the Calibration Byte

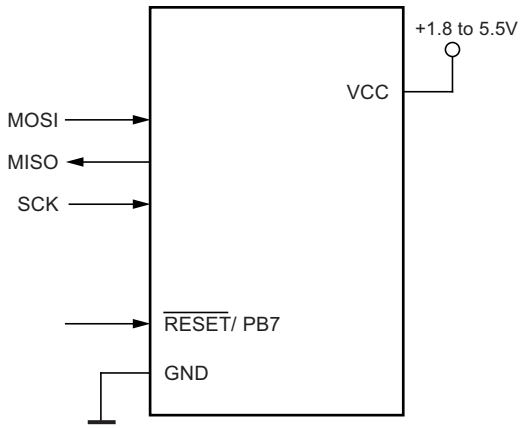
The algorithm for reading the Calibration byte is as follows (refer to “Programming the Flash” on page 162 for details on Command and Address loading):

1. A: Load command “0000 1000”.
2. B: Load address low byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

22.9 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 22-13 on page 167, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 22-7. 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 22-13. 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: > 2 CPU clock cycles for f_{ck} < 12MHz, 3 CPU clock cycles for f_{ck} ≥ 12MHz

High: > 2 CPU clock cycles for f_{ck} < 12MHz, 3 CPU clock cycles for f_{ck} ≥ 12MHz

22.9.1 Serial Programming Algorithm

When writing serial data to the ATtiny261/461/861, data is clocked on the rising edge of SCK.

When reading data from the ATtiny261/461/861, data is clocked on the falling edge of SCK. See Figure 23-6 on page 179 and Figure 23-7 on page 179 for timing details.

To program and verify the ATtiny261/461/861 in the serial programming mode, the following sequence is recommended (see four byte instruction formats in Table 22-15 on page 169):

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 22-14). 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 22-14). 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 22-8). 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 22-14. 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

22.9.2 Serial Programming Instruction set

Table 22-15 and Figure 22-8 on page 170 describes the Instruction set.

Table 22-15. Serial Programming Instruction Set

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming enable	$AC	$53	$00	$00
Chip erase (program memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{\text{BSY}}$	$F0	$00	$00	data byte out
Load Instructions				
Load extended address byte(1)	$4D	$00	Extended adr	$00
Load program memory page, high byte	$48	adr MSB	adr LSB	high data byte in
Load program memory page, low byte	$40	adr MSB	adr LSB	low data byte in
Load EEPROM memory page (page access)	$C1	$00	0000 000aa	data byte in
Read Instructions				
Read program memory, high byte	$28	adr MSB	adr LSB	high data byte out
Read program memory, low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM memory	$A0	$00	00aa aaaa	data byte out
Read lock bits	$58	$00	$00	data byte out
Read signature byte	$30	$00	0000 000aa	data byte out
Read fuse bits	$50	$00	$00	data byte out
Read fuse high bits	$58	$08	$00	data byte out
Read extended fuse bits	$50	$08	$00	data byte out
Read calibration byte	$38	$00	$00	data byte out
Write Instructions(6)				
Write program memory page	$4C	adr MSB	adr LSB	$00
Write EEPROM memory	$C0	$00	00aa aaaa	data byte in
Write EEPROM memory page (page access)	$C2	$00	00aa aa00	$00
Write lock bits	$AC	$E0	$00	data byte in
Write fuse bits	$AC	$A0	$00	data byte in
Write fuse high bits	$AC	$A8	$00	data byte in
Write extended fuse bits	$AC	$A4	$00	data byte in

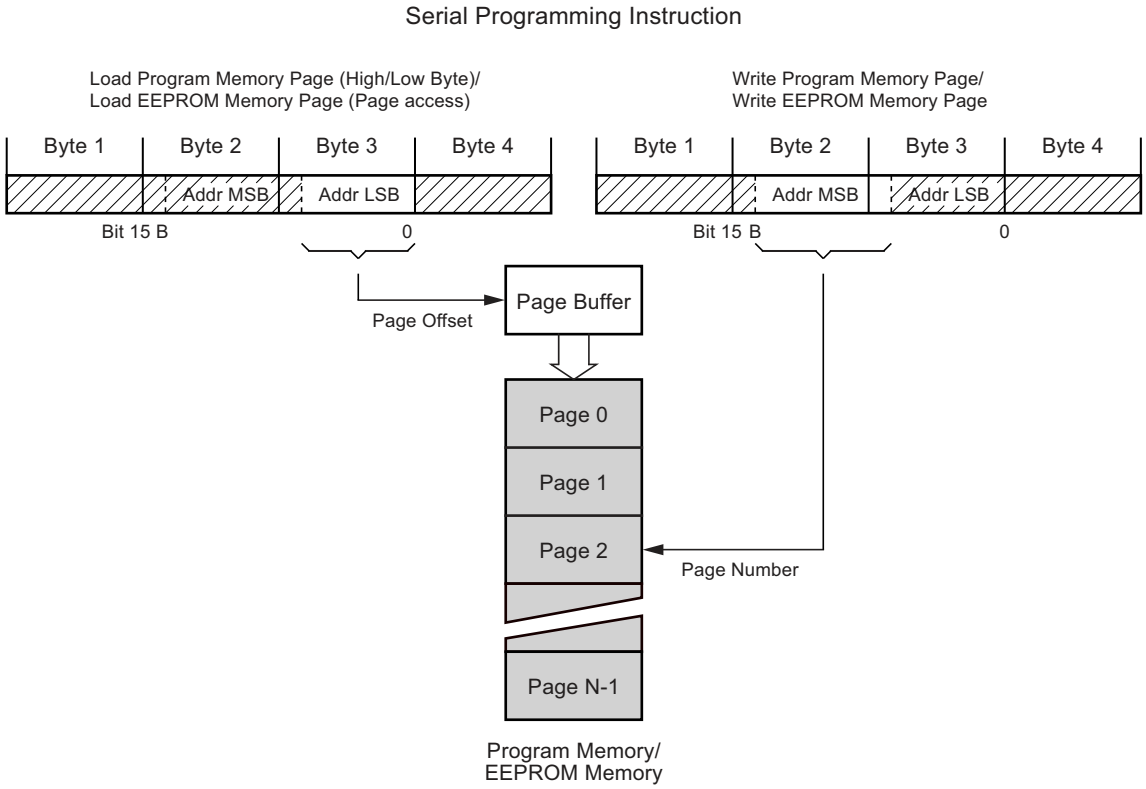
Notes:
1. Not all instructions are applicable for all parts.
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’).
5. Refer to the corresponding section for fuse and lock bits, calibration and signature bytes and page size.
6. Instructions accessing program memory use a word address. This address may be random within the page range.
7. See http://www.atmel.com/avr for application notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page, see Figure 22-8 on page 170.

Figure 22-8. Serial Programming Instruction Example



23. Electrical Characteristics

23.1 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	–55		+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
Injection current at V_{CC} = 0V		±5.0(1)		mA
Injection current at V_{CC} = 5V		±1.0		mA

Notes: 1. Maximum current per port = ±30mA

23.2 DC Characteristics

T_A = –40°C to +125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)(1)

Parameter	Condition	Symbol	Min.	Typ.	Max.	Unit
Input low-voltage	Except XTAL1 and $\overline{\text{RESET}}$ pin	V_{IL}	–0.5		0.2V_{CC}	V
Input high-voltage	Except XTAL1 and $\overline{\text{RESET}}$ pin	V_{IH}	0.7V_{CC}(3)		V_{CC} + 0.5	V
Input low-voltage	XTAL1 pin, external clock selected	V_{IL1}	–0.5		0.1V_{CC}	V
Input high-voltage	XTAL1 pin, external clock selected	V_{IH1}	0.8V_{CC}(3)		V_{CC} + 0.5	V
Input low-voltage	$\overline{\text{RESET}}$ pin	V_{IL2}	–0.5		0.2V_{CC}	V
Input high-voltage	$\overline{\text{RESET}}$ pin	V_{IH2}	0.9V_{CC}(3)		V_{CC} + 0.5	V
Input low-voltage	$\overline{\text{RESET}}$ pin as I/O	V_{IL3}	–0.5		0.2V_{CC}	V
Input high-voltage	$\overline{\text{RESET}}$ pin as I/O	V_{IH3}	0.7V_{CC}(3)		V_{CC} + 0.5	V

Notes:
1. All DC characteristics contained in this data sheet are based on simulation and characterization of ATtiny261/461/861 AVR microcontrollers manufactured in a typical process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual automotive silicon.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. “Min” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
6. Values using methods described in Section 8.7 “Minimizing Power Consumption” on page 35. Power reduction is enabled (PRR = 0xFF) and there is no I/O drive.
7. BOD disabled.

23.2 DC Characteristics (Continued)

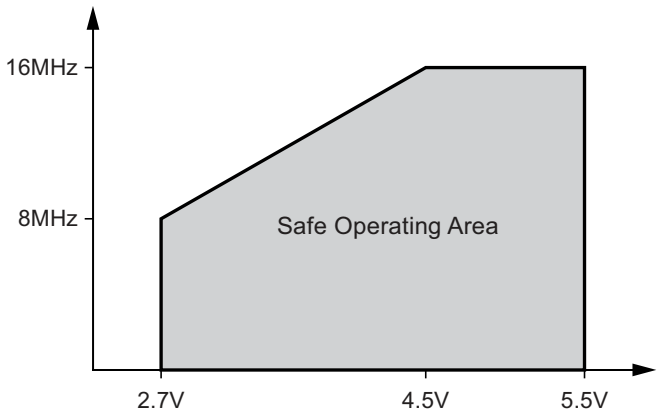
T_A = –40°C to +125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)[1]

Parameter	Condition	Symbol	Min.	Typ.	Max.	Unit
Output low voltage[4] (except reset pin)	I_{OL} = 10mA, V_{CC} = 5V I_{OL} = 5mA, V_{CC} = 3V	V_{OL}			0.6 0.5	V V
Output high-voltage[5] (except reset pin)	I_{OH} = –10mA, V_{CC} = 5V I_{OH} = –5mA, V_{CC} = 3V	V_{OH}	4.3 2.5			V V
Input leakage current I/O pin	Vcc = 5.5V, pin low (absolute value)	I_{IL}			250	nA
Input leakage current I/O pin	Vcc = 5.5V, pin high (absolute value)	I_{IH}			250	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[6] ATD On	I_{CC}		1.45	2	mA
	Active 8MHz, V_{CC} = 5V[6] ATD On			3.96	6	mA
	Active 16MHz, V_{CC} = 5V[6] ATD On			7.16	10	mA
	Idle 4MHz, V_{CC} = 3V[6]			0.25	0.4	mA
	Idle 8MHz, V_{CC} = 5V[6]			0.96	1.2	mA
	Idle 16MHz, V_{CC} = 5V[6]			1.91	2.5	mA
Power-down mode	WDT enabled, V_{CC} = 3V[7]	I_{CC}		4	50	µA
	WDT disabled, V_{CC} = 3V[7]			0.12	30	µA
	WDT enabled, V_{CC} = 5V[7]			6	75	µA
	WDT disabled, V_{CC} = 5V[7]			0.13	45	µA
Analog comparator Input offset voltage	V_{CC} = 5V, 0.1V_{CC} < Vin < V_{CC} – 100mV	V_{ACIO}		10	40	mV
Analog comparator Input leakage current	V_{CC} = 5V, Vin = V_{CC}/2	I_{ACLK}	–50		50	nA

Notes:
1. All DC characteristics contained in this data sheet are based on simulation and characterization of ATtiny261/461/861 AVR microcontrollers manufactured in a typical process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual automotive silicon.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. “Min” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10mA at V_{CC} = 5V, 5mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
6. Values using methods described in Section 8.7 “Minimizing Power Consumption” on page 35. Power reduction is enabled (PRR = 0xFF) and there is no I/O drive.
7. BOD disabled.

23.3 Speed Grades

Figure 23-1. Maximum Frequency versus V_{CC}



23.4 Clock Characteristics

23.4.1 Calibrated Internal RC Oscillator Accuracy

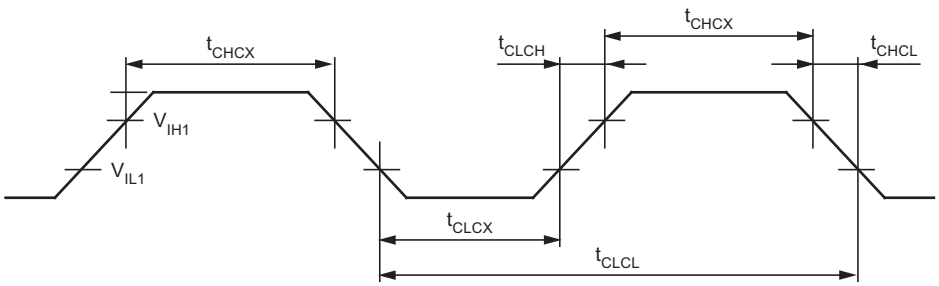
Table 23-1. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{CC}	Temperature	Calibration Accuracy
Factory Calibration	8.0MHz	3V	25°C	±2%
		2.7V to 5.5V(1)	–40°C to +125°C	±17%

Notes: 1. Voltage range for ATtiny261/461/861.

23.4.2 External Clock Drive Waveforms

Figure 23-2. External Clock Drive Waveforms



23.4.3 External Clock Drive

Table 23-2. External Clock Drive

Parameter	Symbol	V_{CC} = 2.7 - 5.5V		V_{CC} = 4.5 - 5.5V		Unit
		Min.	Max.	Min.	Max.	
Clock frequency	$1/t_{CLCL}$	0	8	0	16	MHz
Clock period	t_{CLCL}	125		62.5		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	%

23.5 System and Reset Characteristics

Table 23-3. Reset, Brown-out and Internal Voltage Characteristics(1)

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Minimum pulse width on $\overline{\text{RESET}}$ pin	V_{CC} = 3V	t_{RST}			2.5	µs
Brown-out detector hysteresis		V_{HYST}		50		mV
Min pulse width on brown-out reset		t_{BOD}		2		µs
Bandgap reference voltage	V_{CC} = 3.0V, 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

Notes: 1. Values are guidelines only.

Table 23-4. BODLEVEL Fuse Coding(1)

BODLEVEL [2..0] Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Unit
111	BOD disabled			
110	1.68	1.8	1.92	V
101	2.5	2.7	2.9	
100	4.0	4.3	4.6	
011	Reserved			
010				
001				
000				

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

23.6 ADC Characteristics

Table 23-5. ADC Characteristics, Single Ended Channels, –40°C +125°C

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Resolution	Single ended conversion			10		Bits
Absolute accuracy	V_{CC} = 4V, V_{REF} = 4V, ADC clock = 200kHz	TUE		2.0	4.0	LSB
Integral non-linearity	V_{CC} = 4V,V_{REF} = 4V, ADC clock = 200kHz	INL		0.6	1.8	LSB
Differential non-linearity (DNL)	V_{CC} = 4V, V_{REF} = 4V, ADC clock = 200kHz	DNL		0.2	0.6	LSB
Gain error	V_{CC} = 4V, V_{REF} = 4V, ADC clock = 200kHz		–5.0	–2.0	+3.0	LSB
Offset error	V_{CC} = 4V, V_{REF} = 4V, ADC clock = 200kHz		–3.5	+1.2	+3.5	LSB
External reference voltage		V_{REF}	2.56		AVCC	V
Clock frequency			50		200	kHz
Analog supply frequency		AVCC	V_{CC} – 0.3		V_{CC} + 0.3	V
Internal voltage reference		V_{INT}	1.0	1.1	1.2	V
Analog input resistance		R_{AIN}		100		MΩ
Reference input resistance		R_{Ref}	21	30	39	kΩ
Temperature sensor accuracy	After firmware calibration Internal V_{REF} V_{CC} = 3V			±10		°C

Table 23-6. ADC Characteristics, Differential Channels, –40°C +125°C

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Resolution	Differential conversion, gain = 1x or 8x			8		Bits
	Differential conversion, gain = 20x or 32x			8		
Absolute accuracy	Gain = 1x / 8x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz	TUE		1.7	4.0	LSB
	Gain = 20x / 32x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			2.0	5.0	
	Gain = 1x / 8x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			2.3	5.0	
	Gain = 20x / 32x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			3.0	6.0	

Table 23-6. ADC Characteristics, Differential Channels, –40°C +125°C (Continued)

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Integral non-linearity	Gain = 1x / 8x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz	INL		0.3	1.5	LSB
	Gain = 20x / 32x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			0.7	3.0	
	Gain = 1x / 8x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			1.0	3.0	
	Gain = 20x / 32x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			2.0	6.0	
Differential Non-linearity	Gain = 1x / 8x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz	DNL		0.3	1.0	LSB
	Gain = 20x / 32x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			0.4	1.2	
	Gain = 1x / 8x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			0.4	1.0	
	Gain = 20x / 32x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz			0.8	2.5	
Gain error	Gain = 1x / 8x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz		–4.0	+2.0	+4.0	LSB
	Gain = 20x / 32x, BIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz		–4.0	+1.4	+4.0	
	Gain = 1x / 8x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz		–5.0	–2.6	+5.0	
	Gain = 20x / 32x, UNIPOLAR, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz		–5.0	–0.8	+5.0	
Offset error	Gain = 1x, V_{CC} = 5V, V_{REF} = 4V, ADC clock = 200kHz		–3.0	+0.6	+3.0	LSB
External reference voltage		V_{REF}	2.56		AVCC – 0.5	V

23.7 Parallel Programming Characteristics

Figure 23-3. Parallel Programming Timing, Including some General Timing Requirements

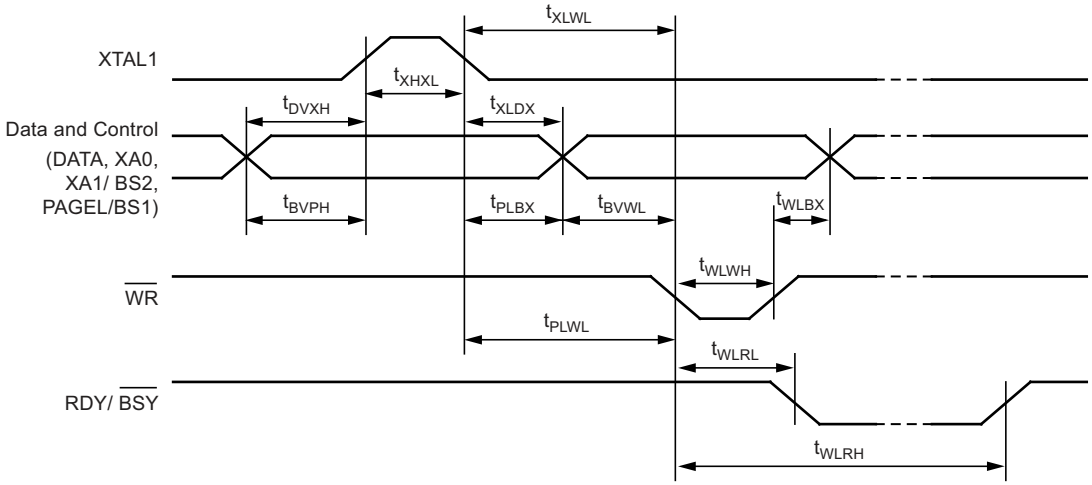
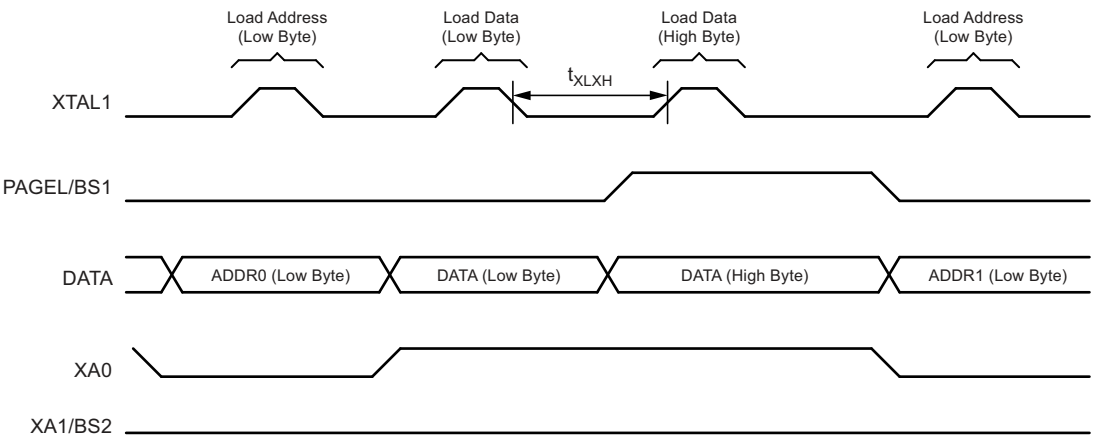
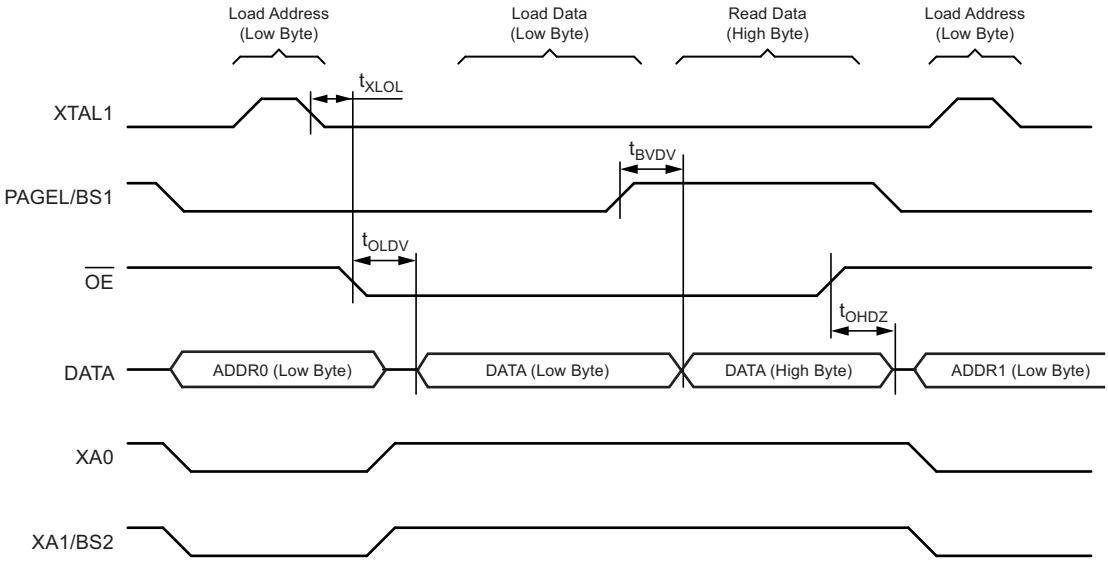


Figure 23-4. Parallel Programming Timing, Loading Sequence with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 23-3 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 23-5. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 23-3 on page 177 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 23-7. Parallel Programming Characteristics, V_{CC} = 5V ±10%

Parameter	Symbol	Min	Typ	Max	Unit
Programming enable voltage	V_{PP}	11.5		12.5	V
Programming enable current	I_{PP}			250	µA
Data and control valid before XTAL1 high	t_{DVXH}	67			ns
XTAL1 low to XTAL1 high	t_{XLXH}	200			ns
XTAL1 pulse width high	t_{XHXL}	150			ns
Data and control hold after XTAL1 low	t_{XLDX}	67			ns
XTAL1 low to $\overline{WR}$ low	t_{XLWL}	0			ns
BS1 valid before PAGEL high	t_{BVPH}	67			ns
PAGEL pulse width high	t_{PHPL}	150			ns
BS1 hold after PAGEL low	t_{PLBX}	67			ns
BS2/1 hold after $\overline{WR}$ low	t_{WLBX}	67			ns
PAGEL low to $\overline{WR}$ low	t_{PLWL}	67			ns
BS1 valid to $\overline{WR}$ low	t_{BVWL}	67			ns
$\overline{WR}$ pulse width low	t_{WLWH}	150			ns
$\overline{WR}$ low to RDY/$\overline{BSY}$ low	t_{WLRL}	0		1	µs
$\overline{WR}$ low to RDY/$\overline{BSY}$ high(1)	t_{WLRH}	3.7		4.5	ms
$\overline{WR}$ low to RDY/$\overline{BSY}$ high for chip erase(2)	t_{WLRH_CE}	7.5		9	ms
XTAL1 low to $\overline{OE}$ low	t_{XLOL}	0			ns
BS1 valid to DATA valid	t_{BVDV}	0		250	ns
$\overline{OE}$ low to DATA valid	t_{OLDV}			250	ns
$\overline{OE}$ high to DATA tri-stated	t_{OHDZ}			250	ns

Notes: 1. t_{WLRH} is valid for the write flash, write EEPROM, write fuse bits and write lock bits commands.
2. t_{WLRH_CE} is valid for the chip erase command.

23.8 Serial Programming Characteristics

Figure 23-6. Serial Programming Waveforms

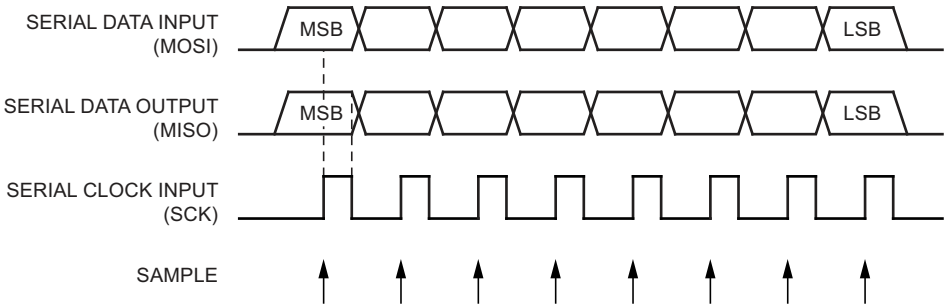


Figure 23-7. Serial Programming Timing

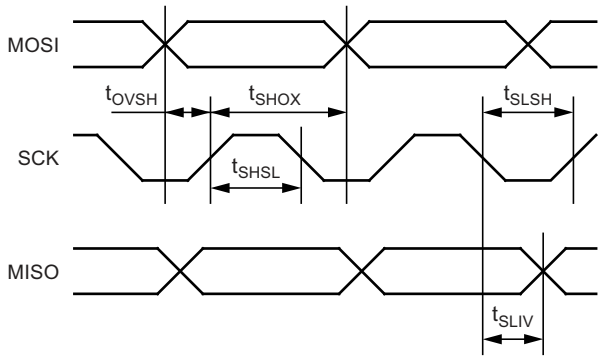


Table 23-8. Serial Programming Characteristics, T_A = –40°C to +125°C, V_{CC} = 2.7 - 5.5V (Unless Otherwise Noted)

Parameter	Symbol	Min	Typ	Max	Unit
Oscillator frequency (ATtiny261/461/861V)	$1/t_{CLCL}$	0		4	MHz
Oscillator period (ATtiny261/461/861V)	t_{CLCL}	250			ns
Oscillator frequency (ATtiny261/461/861L, VCC = 2.7 - 5.5V)	$1/t_{CLCL}$	0		10	MHz
Oscillator period (ATtiny261/461/861L, VCC = 2.7 - 5.5V)	t_{CLCL}	100			ns
Oscillator frequency (ATtiny261/461/861, V_{CC} = 4.5V - 5.5V)	$1/t_{CLCL}$	0		16	MHz
Oscillator period (ATtiny261/461/861, V_{CC} = 4.5V - 5.5V)	t_{CLCL}	50			ns
SCK pulse width high	t_{SHSL}	2 t_{CLCL}*			ns
SCK pulse width low	t_{SLSH}	2 t_{CLCL}*			ns
MOSI setup to SCK high	t_{OVSH}	t_{CLCL}			ns
MOSI hold after SCK high	t_{SHOX}	2 t_{CLCL}			ns
SCK low to MISO valid	t_{SLIV}	TBD	TBD	TBD	ns

Note: 1. 2 t_{CLCL} for f_{ck} < 12MHz, 3 t_{CLCL} for f_{ck} ≥ 12MHz

24. Typical Characteristics

The data contained in this section is largely based on simulations and characterization of similar devices in the same process and design methods. These values are preliminary values representing design targets, and will be updated after characterization of actual automotive silicon.

Thus, the data should be treated as indications of how the part will behave.

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as $C_L \times V_{CC} \times f$ where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in power-down mode with watchdog timer enabled and power-down mode with watchdog timer disabled represents the differential current drawn by the watchdog timer.

Unless otherwise specified the data contained in this chapter are for –40°C to +85°C.

24.1 Active Supply Current

Figure 24-1. Active Supply Current versus Low Frequency (0.1 - 1.0MHz)

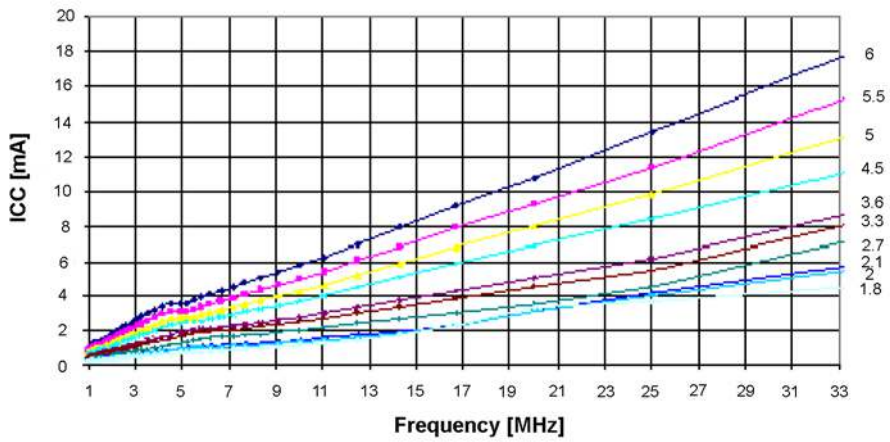
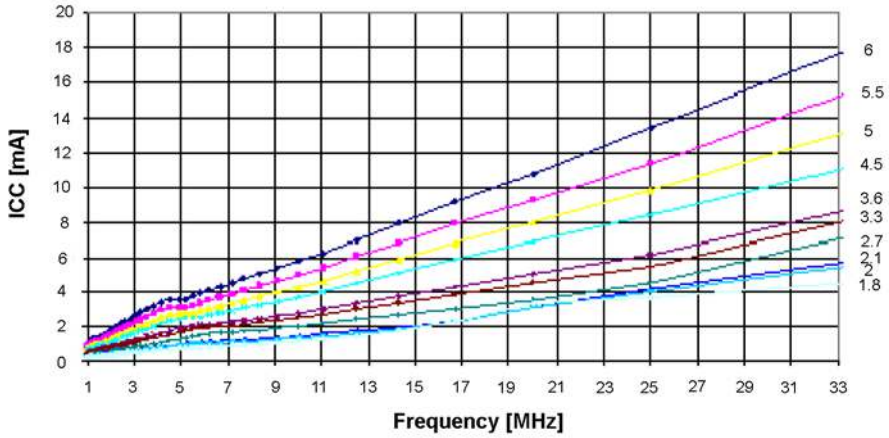
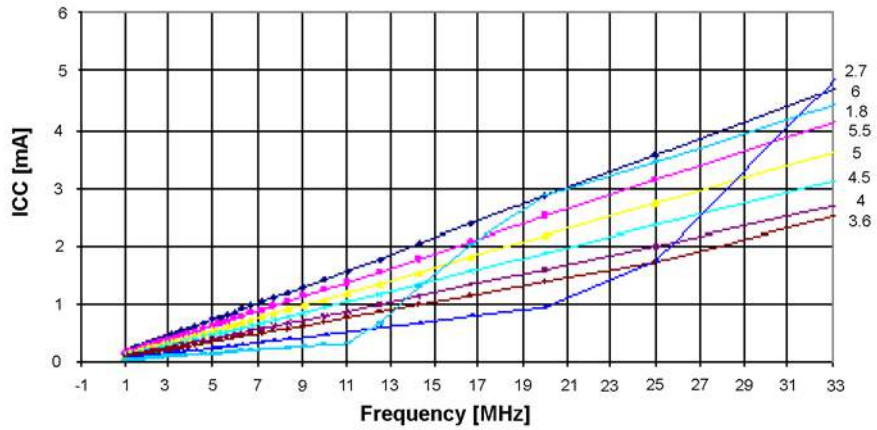


Figure 24-2. Active Supply Current versus Frequency (1 - 16MHz)



24.2 Idle Supply Current

Figure 24-3. Idle Supply Current versus Frequency (1 - 16MHz)



24.3 Supply Current of I/O Modules

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules are controlled by the power reduction register. See Section 8.8.2 “PRR – Power Reduction Register” on page 37 for details.

Table 24-1. 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	65µA	423µA	1787µA
PRTIM0	7µA	39µA	165µA
PRUSI	5µA	25µA	457µA
PRADC	18µA	111µA	102µA

Table 24-2. Additional Current Consumption (Percentage) in Active and Idle Mode

PRR Bit	Additional Current Consumption Compared to Active with External Clock (see Figure 24-1 and Figure 24-2)	Additional Current Consumption Compared to Idle with External Clock
PRTIM1	26.9%	103.7%
PRTIM0	2.6%	10.0%
PRUSI	1.7%	6.5%
PRADC	7.1%	27.3%

It is possible to calculate the typical current consumption based on the numbers from Table 24-1 for other V_{CC} and frequency settings than listed in Table 24-2.

24.3.1 Example

Calculate the expected current consumption in idle mode with TIMER0, ADC, and USI enabled at V_{CC} = 2.0V and F = 1MHz. From Table 24-2, third column, we see that we need to add 10% for the TIMER0, 27.3% for the ADC, and 6.5% for the USI module. Reading from Figure 24-3 on page 181, we find that the idle current consumption is ~0,085mA at V_{CC} = 2.0V and F = 1MHz. The total current consumption in idle mode with TIMER0, ADC, and USI enabled, gives:

$ICCtotal \approx 0.085mA \times (1 + 0.10 + 0.273 + 0.065) \approx 0.122mA$

24.4 Power-down Supply Current

Figure 24-4. Power-down Supply Current versus V_{CC} (Watchdog Timer Disabled)

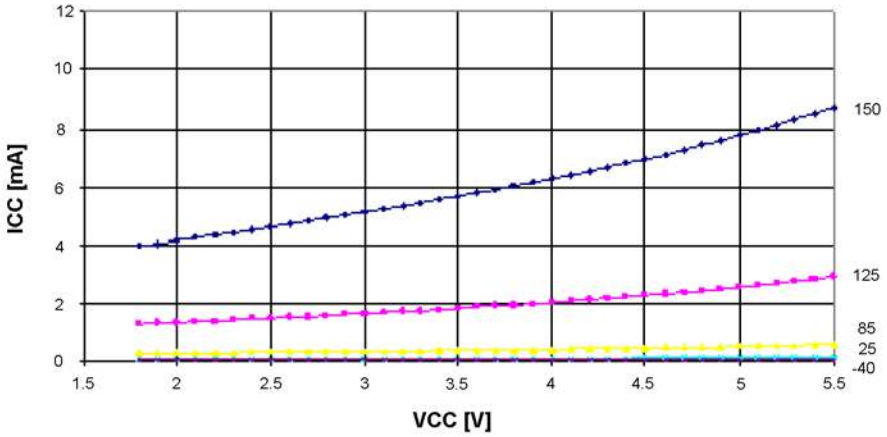
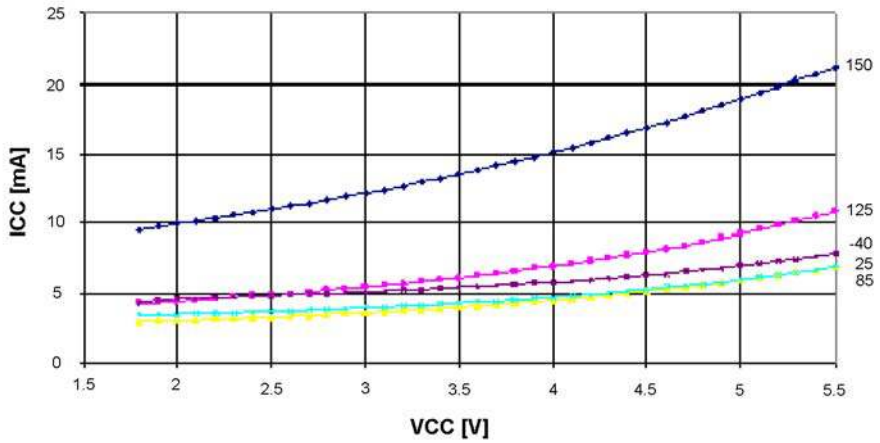


Figure 24-5. Power-down Supply Current versus V_{CC} (Watchdog Timer Enabled)



24.5 Pin Pull-up

Figure 24-6. I/O Pin pull-up Resistor Current versus Input Voltage (V_{CC} = 5V)

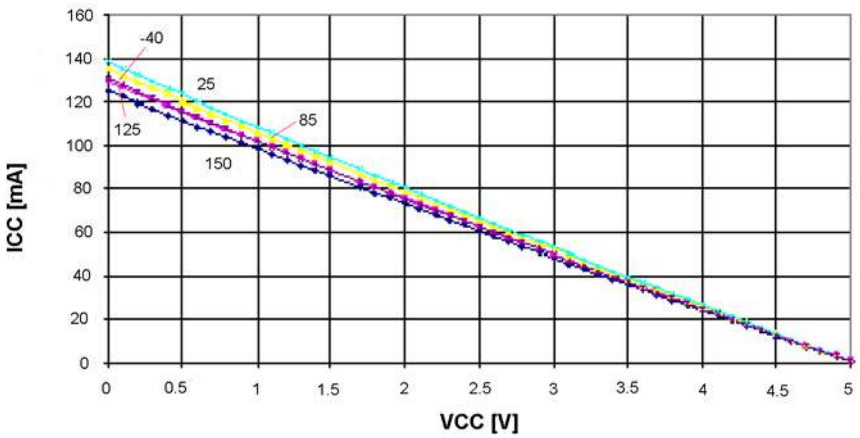
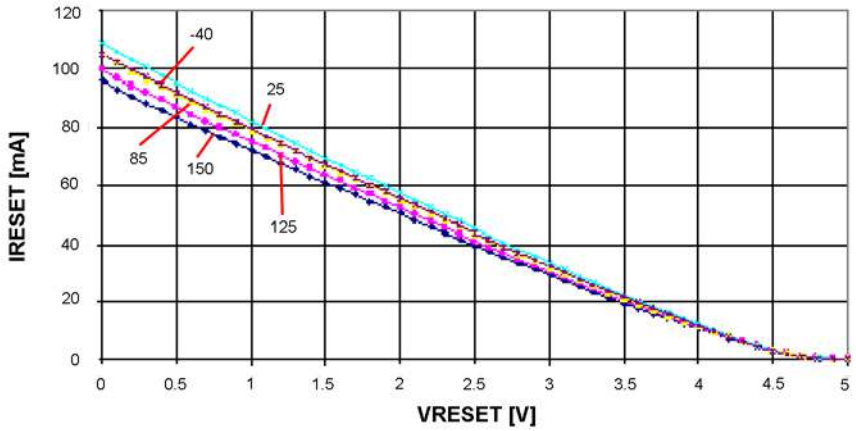


Figure 24-7. Reset Pull-up Resistor Current versus Reset Pin Voltage (V_{CC} = 5V)



24.6 Pin Driver Strength

Figure 24-8. I/O Pin Output Voltage versus Sink Current (V_{CC} = 3V)

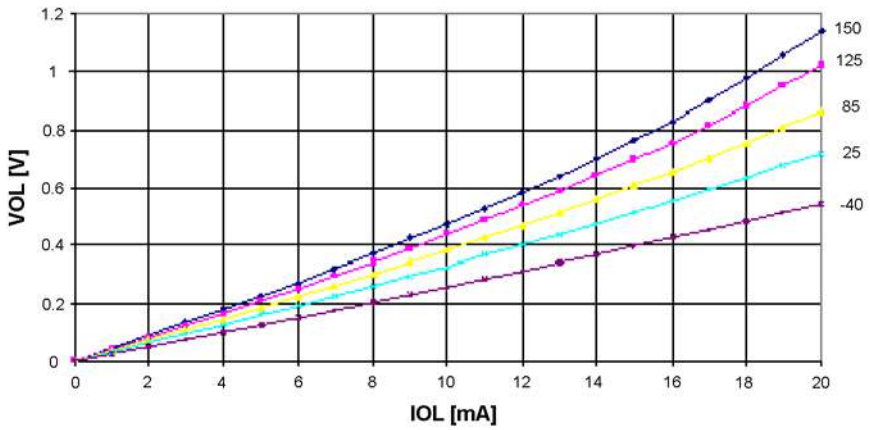


Figure 24-9. I/O Pin Output Voltage versus Sink Current (V_{CC} = 5V)

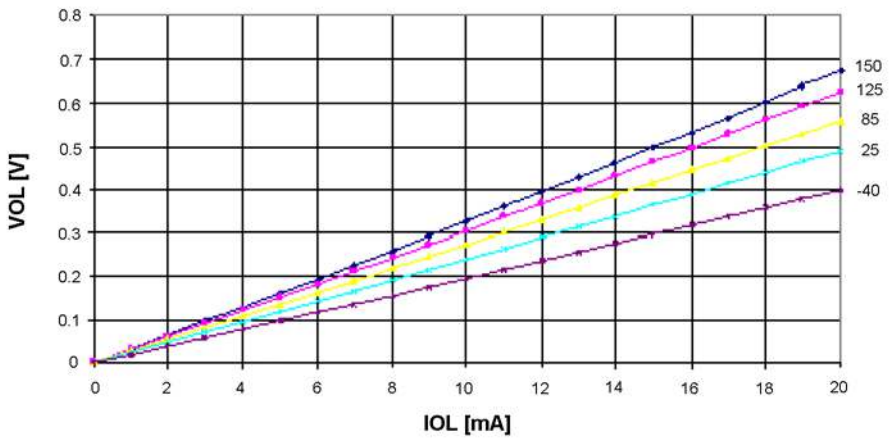


Figure 24-10. I/O Pin Output Voltage versus Source Current (V_{CC} = 3V)

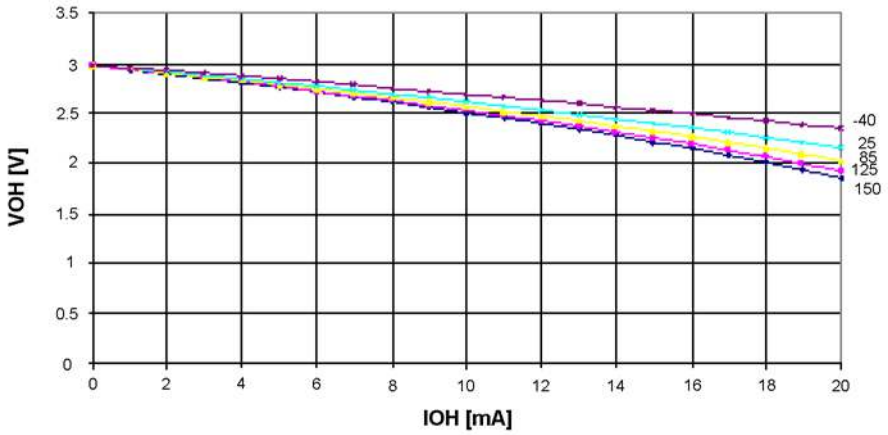
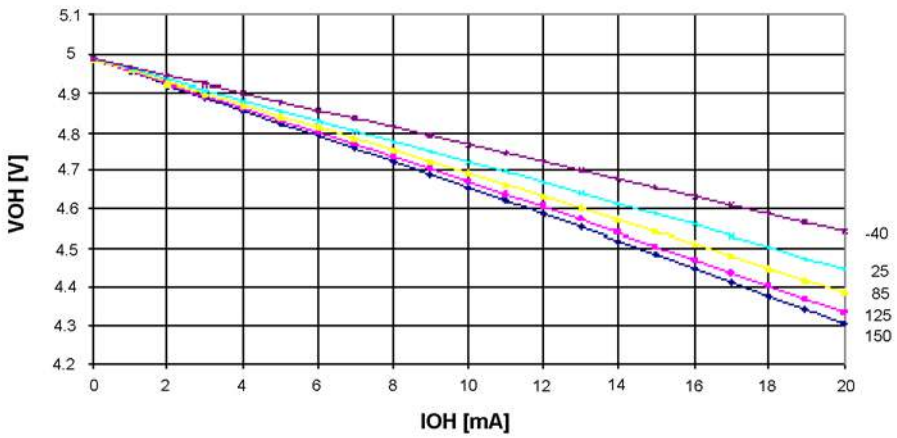


Figure 24-11. I/O Pin Output Voltage versus Source Current (V_{CC} = 5V)



24.7 Pin Threshold and Hysteresis

Figure 24-12. I/O Pin Input Threshold Voltage versus V_{CC} (V_{IH}, I/O Pin Read as '1')

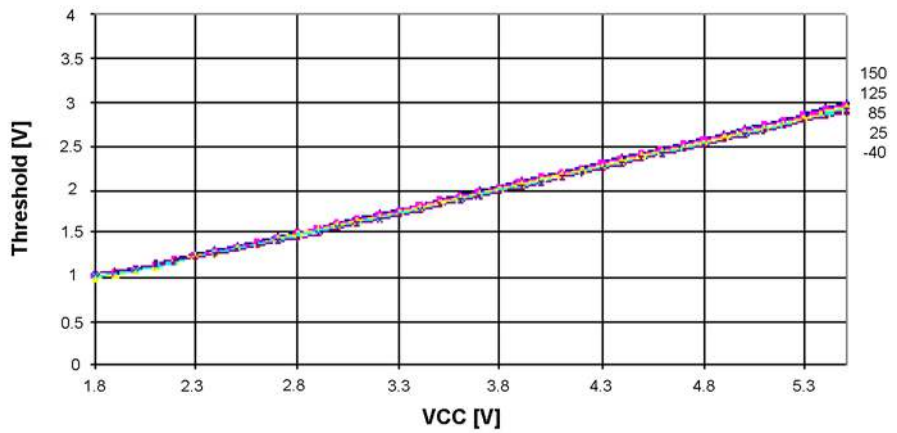


Figure 24-13. I/O Pin Input Threshold Voltage versus V_{CC} (V_{IL}, I/O Pin Read as '0')

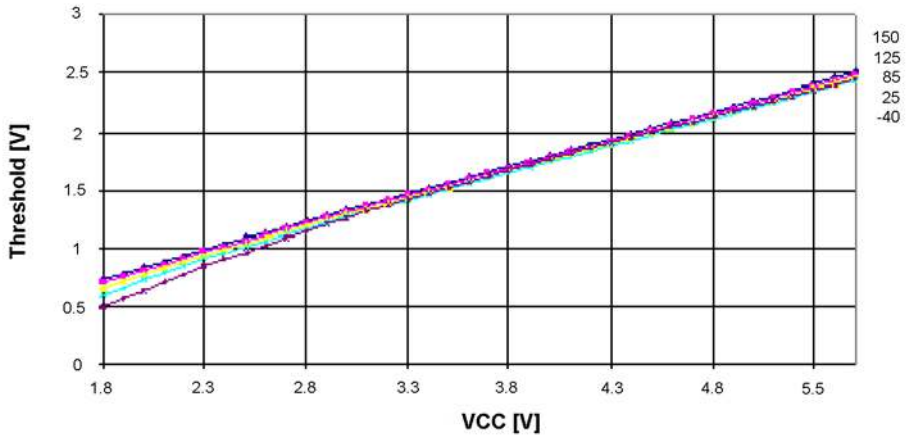


Figure 24-14. Reset Input Threshold Voltage versus V_{CC} (V_{IH}, Reset Read as '1')

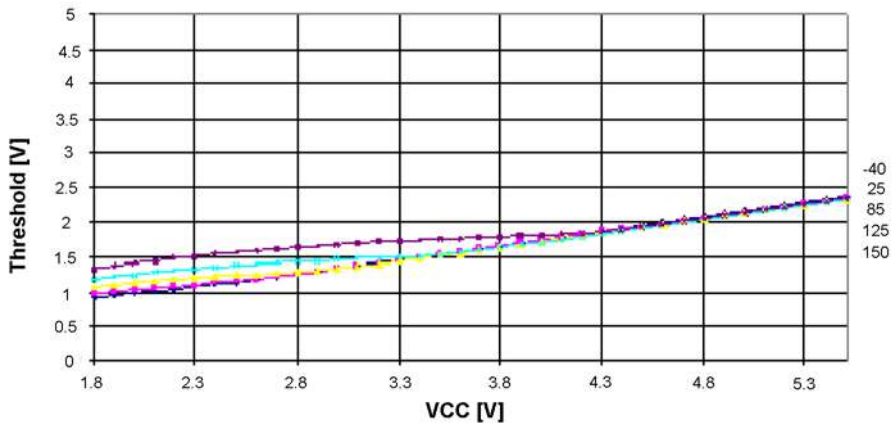
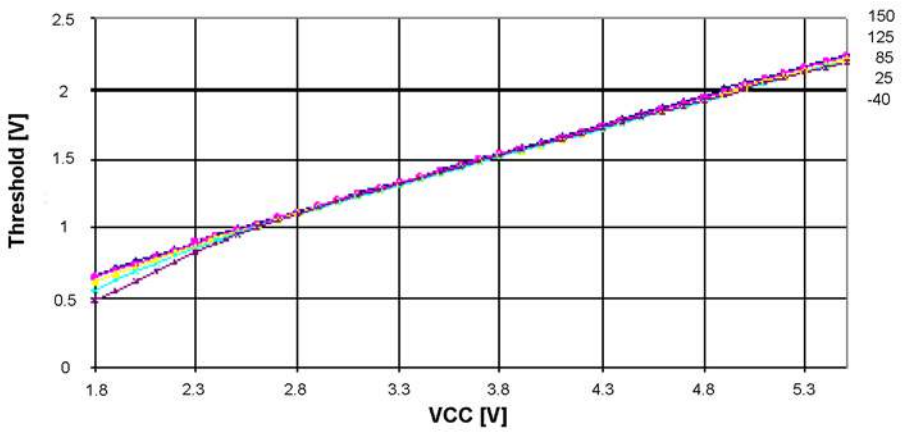


Figure 24-15. Reset Input Threshold Voltage versus V_{CC} (V_{IL}, Reset Read as '0')



24.8 BOD Threshold and Analog Comparator Offset

Figure 24-16. BOD Threshold versus Temperature (BOD Level is 4.3V)

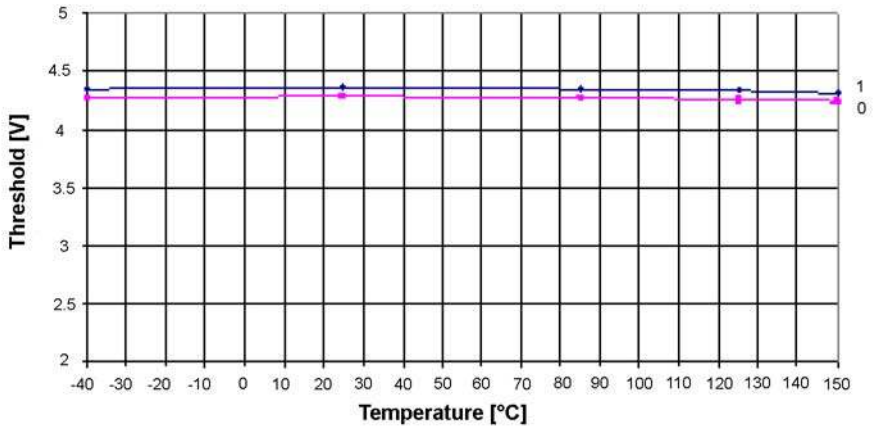


Figure 24-17. BOD Threshold versus Temperature (BOD Level is 2.7V)

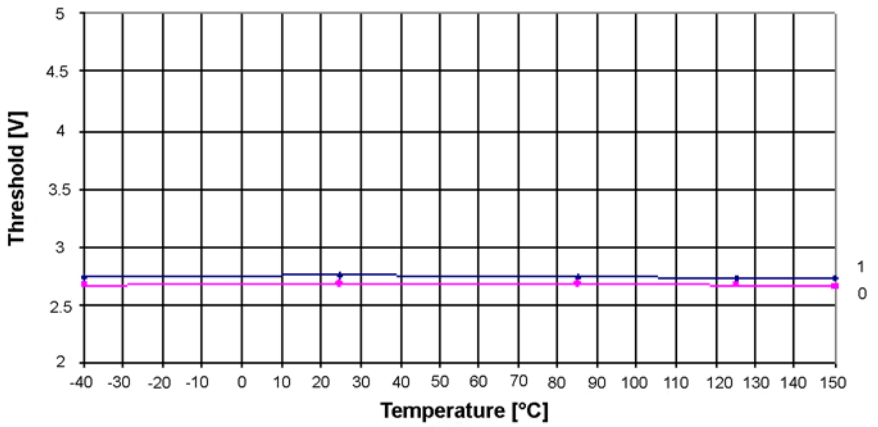
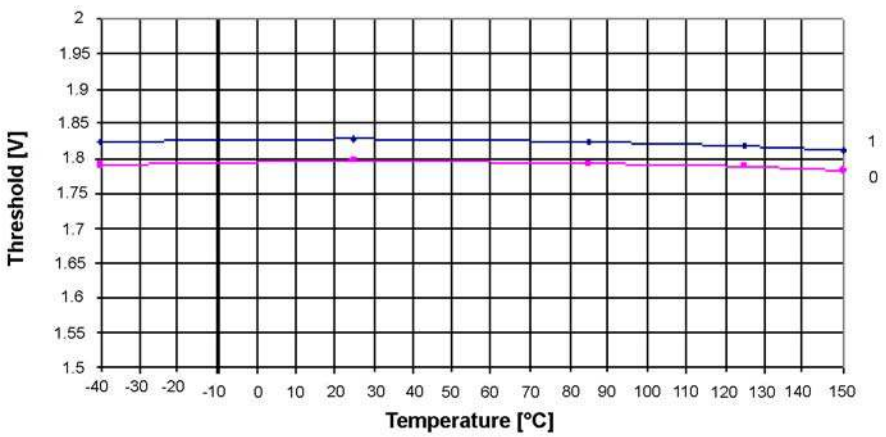


Figure 24-18. BOD Threshold versus Temperature (BOD Level is 1.8V)



24.9 Internal Oscillator Speed

Figure 24-19. Watchdog Oscillator Frequency versus Temperature

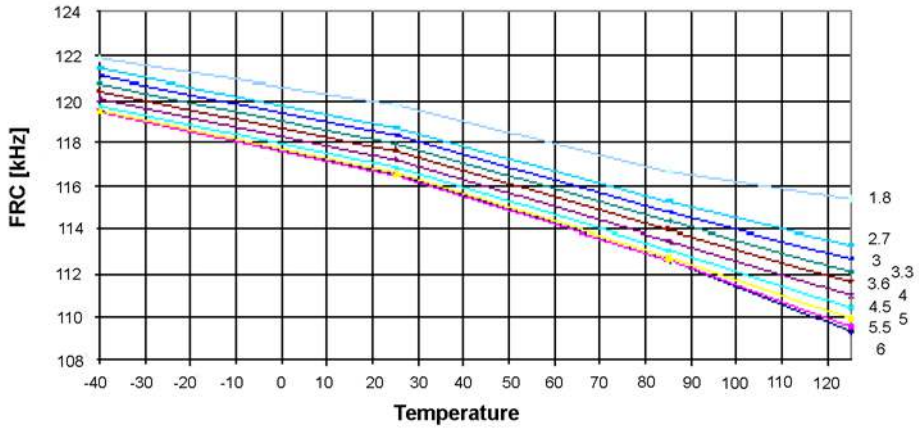
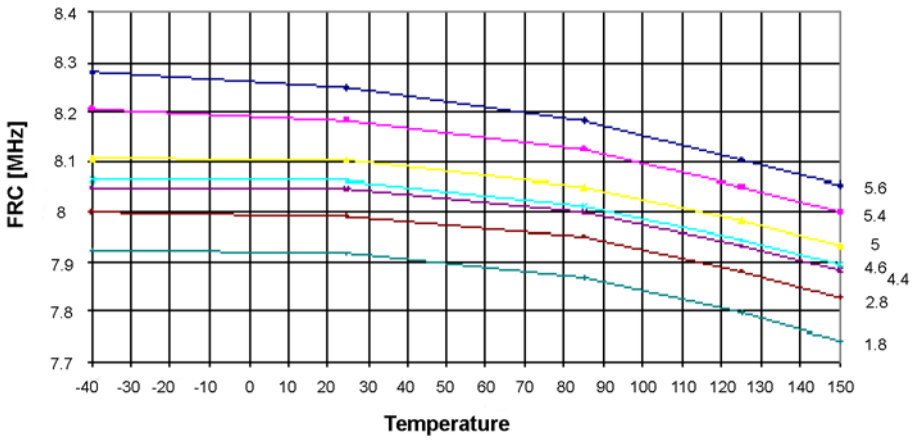


Figure 24-20. Calibrated 8.0MHz RC Oscillator Frequency versus Temperature



25. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	10
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	12
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x3C (0x5C)	Reserved	–								
0x3B (0x5B)	GIMSK	INT1	INT0	PCIE1	PCIE0	–	–	–	–	50
0x3A (0x5A)	GIFR	INTF1	INTF0	PCIF	–	–	–	–	–	50
0x39 (0x59)	TIMSK	OCIE1D	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	TICIE0	82, 117
0x38 (0x58)	TIFR	OCF1D	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	ICF0	83, 118
0x37 (0x57)	SPMCSR	–	–	SIGRD	CTPB	RFLB	PGWRT	PGERS	SPMEN	155
0x36 (0x56)	PRR					PRTIM1	PRTIM0	PRUSI	PRADC	35
0x35 (0x55)	MCUCR	–	PUD	SE	SM1	SM0	–	ISC01	ISC00	36, 64, 49
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	44,
0x33 (0x53)	TCCR0B	–	–	–	TSM	PSR0	CS02	CS01	CS00	68
0x32 (0x52)	TCNT0L	Timer/Counter0 counter register low byte								81
0x31 (0x51)	OSCCAL	Oscillator calibration register								31
0x30 (0x50)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM1A	PWM1B	108
0x2F (0x4F)	TCCR1B	PWM1X	PSR1	DTPS11	DTPS10	CS13	CS12	CS11	CS10	155
0x2E (0x4E)	TCNT1	Timer/Counter1 counter register								115
0x2D (0x4D)	OCR1A	Timer/Counter1 output compare register A								116
0x2C (0x4C)	OCR1B	Timer/Counter1 output compare register B								116
0x2B (0x4B)	OCR1C	Timer/Counter1 output compare register C								116
0x2A (0x4A)	OCR1D	Timer/Counter1 output compare register D								117
0x29 (0x49)	PLLCSR	LSM					PCKE	PLLE	PLOCK	85
0x28 (0x48)	CLKPR	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0	32
0x27 (0x47)	TCCR1C	COM1A1S	COM1A0S	COM1B1S	COM1B0S	COM1D1	COM1D0	FOC1D	PWM1D	112
0x26 (0x46)	TCCR1D	FPIE1	FPEN1	FPNC1	FPES1	FPAC1	FPF1	WGM11	WGM10	114
0x25 (0x45)	TC1H							TC19	TC18	115
0x24 (0x44)	DT1	DT1H3	DT1H2	DT1H1	DT1H0	DT1L3	DT1L2	DT1L1	DT1L0	118
0x23 (0x43)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	51
0x22 (0x42)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	51
0x21 (0x41)	WDTCR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	44
0x20 (0x40)	DWDR	DWDR[7:0]								35
0x1F (0x3F)	EEARH								EEAR8	21
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	21
0x1D (0x3D)	EEDR	EEPROM data register								22
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	22
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	65
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	65
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	65
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65

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.

25. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x15 (0x35)	TCCR0A	TCW0	ICEN0	ICNC0	ICES0	ACIC0			CTC0	80
0x14 (0x34)	TCNT0H	Timer/Counter0 counter register high byte								81
0x13 (0x33)	OCR0A	Timer/Counter0 output compare register A								81
0x12 (0x32)	OCR0B	Timer/Counter0 output compare register B								81
0x11 (0x31)	USIPP								USIPOS	128
0x10 (0x30)	USIBR	USI buffer register								126
0x0F (0x2F)	USIDR	USI data register								125
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	126
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	127
0x0C (0x2C)	GPIOR2	General purpose I/O register 2								23
0x0B (0x2B)	GPIOR1	General purpose I/O register 1								23
0x0A (0x2A)	GPIOR0	General purpose I/O register 0								23
0x09 (0x29)	ACSRB	HSEL	HLEV				ACM2	ACM1	ACM0	131
0x08 (0x28)	ACSRA	ACD	ACBG	ACO	ACI	ACIE	ACME	ACIS1	ACIS0	129
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	144
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	146
0x05 (0x25)	ADCH	ADC data register high byte								147
0x04 (0x24)	ADCL	ADC data register low byte								147
0x03 (0x23)	ADCSRB	BIN	GSEL		REFS2	MUX5	ADTS2	ADTS1	ADTS0	148
0x02 (0x22)	DIDR1	ADC10D	ADC9D	ADC8D	ADC7D					149
0x01 (0x21)	DIDR0	ADC6D	ADC5D	ADC4D	ADC3D	AREFD	ADC2D	ADC1D	ADC0D	149
0x00 (0x20)	TCCR1E	–	-	OC1OE5	OC1OE4	OC1OE3	OC1OE2	OC1OE1	OC1OE0	115

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.

26. 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

26. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BREQ	k	Branch if equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if greater or equal, signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if less than zero, signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag is cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
Bit and Bit-test Instructions					
SBI	P,b	Set bit in I/O register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear bit in I/O register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical shift left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical shift right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate left through carry	Rd(0) ← C,Rd(n+1) ← Rd(n),C ← Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate right through carry	Rd(7) ← C,Rd(n) ← Rd(n+1),C ←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic shift right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap nibbles	Rd(3..0) ← Rd(7..4),Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit store from register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to register	Rd(b) ← T	None	1
SEC		Set carry	C ← 1	C	1
CLC		Clear carry	C ← 0	C	1
SEN		Set negative flag	N ← 1	N	1
CLN		Clear negative flag	N ← 0	N	1
SEZ		Set zero flag	Z ← 1	Z	1
CLZ		Clear zero flag	Z ← 0	Z	1
SEI		Global interrupt enable	I ← 1	I	1

26. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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
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	

26. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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 description for sleep function)	None	1
WDR		Watchdog reset	(see specific description for WDR/Timer)	None	1
BREAK		Break	For on-chip debug Only	None	N/A

27. Ordering Information

Table 27-1. Available Product Offering

Ordering Code[2]	Speed (MHz)[3]	Power Supply (V)	Package[1]	Operation Range
ATtiny261-15SZ	16	2.7 - 5.5	TG	Automotive (–40° to +125°C)
ATtiny261-15MZ	16	2.7 - 5.5	PN	Automotive (–40° to +125°C)
ATtiny261-15XZ	16	2.7 - 5.5	6G	Automotive (–40° to +125°C)
ATtiny261-15MAZ	16	2.7 - 5.5	PC	Automotive (–40° to +125°C)
ATtiny461-15SZ	16	2.7 - 5.5	TG	Automotive (–40° to +125°C)
ATtiny461-15MZ	16	2.7 - 5.5	PN	Automotive (–40° to +125°C)
ATtiny461-15XZ	16	2.7 - 5.5	6G	Automotive (–40° to +125°C)
ATtiny461-15MAZ	16	2.7 - 5.5	PC	Automotive (–40° to +125°C)
ATtiny861-15SZ	16	2.7 - 5.5	TG	Automotive (–40° to +125°C)
ATtiny861-15MZ	16	2.7 - 5.5	PN	Automotive (–40° to +125°C)
ATtiny861-15XZ	16	2.7 - 5.5	6G	Automotive (–40° to +125°C)
ATtiny861-15MAZ	16	2.7 - 5.5	PC	Automotive (–40° to +125°C)

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also halide free and fully green.
3. For speed versus V_{CC}, see Figure 23.3 on page 173.

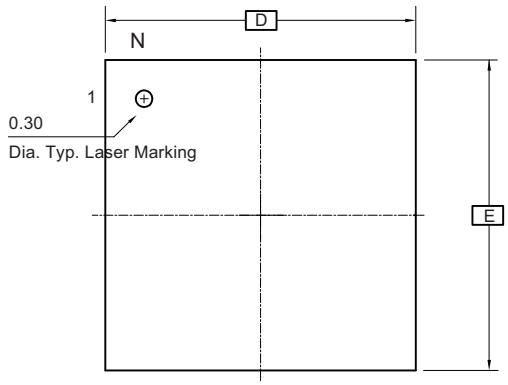
Table 27-2. Package Types

Package Type	
PN	32-pad, 5.0x5.0mm body, lead pitch 0.50mm, quad flat no lead package (QFN)
TG	20-lead, 0.300" wide body lead, plastic gull wing small outline package (SOIC)
6G	20-lead, 4.4x6.5mm body, 0.65mm pitch, lead length: 0.6mm Thin shrink small outline package (TSSOP)
PC	20-lead, 4.0x4.0mm body, 0.50mm pitch, quad flat no lead package (QFN)

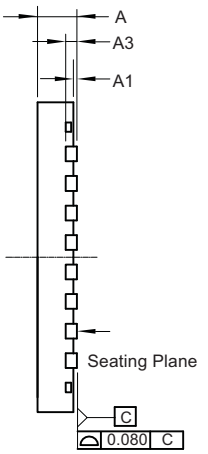
28. Packaging Information

Figure 28-1. PN

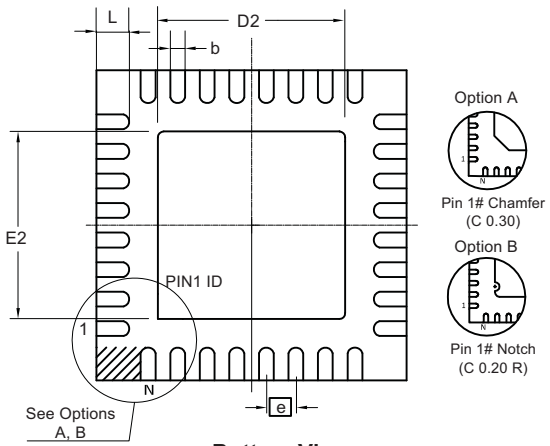
Drawings not scaled



Top View



Side View



Bottom View

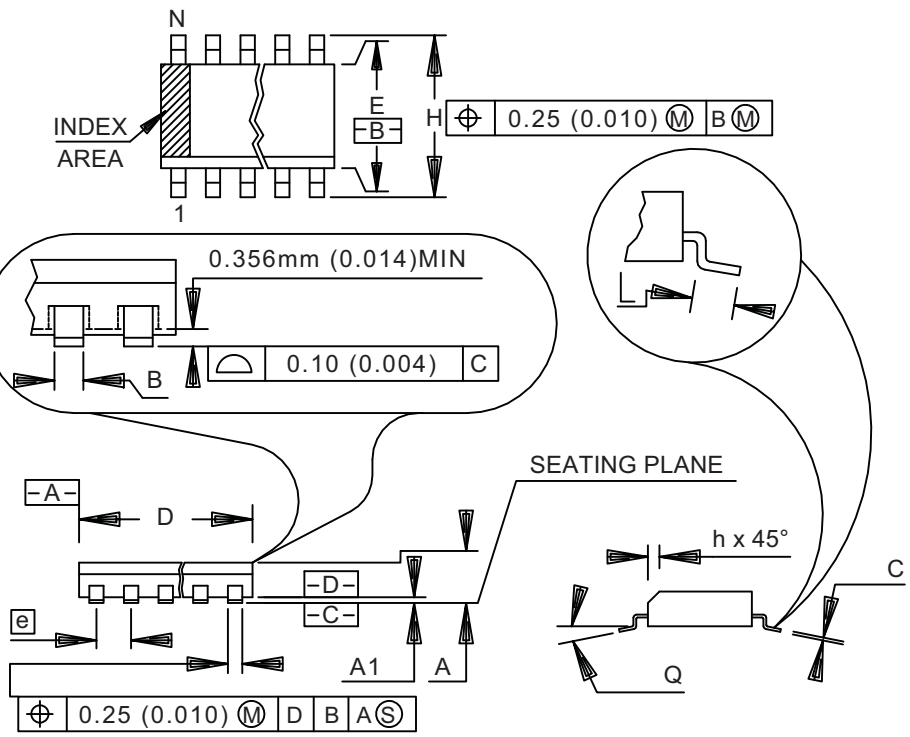
COMMON DIMENSIONS (Unit of Measure = mm)				
Symbol	**MIN**	**NOM**	**MAX**	**NOTE**
A	0.80	0.85	0.90	
A1	0.00	----	0.05	
A3		0.20 REF		
D/E		5.00 BSC		
D2/E2	3.00	3.10	3.20	
L	0.30	0.40	0.50	
b	0.18	0.25	0.30	2
e		0.50 BSC		
n		32		

Notes: 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation VHHD-2, for proper dimensions, tolerances, datums, etc.
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.

01/31/12

Atmel **Package Drawing Contact:** packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	PN, 32 Leads - 0.50mm Pitch, 5x5mm Very Thin Quad Flat no Lead Package (VQFN) Sawn	ZMF	PN	I

Figure 28-2. TG

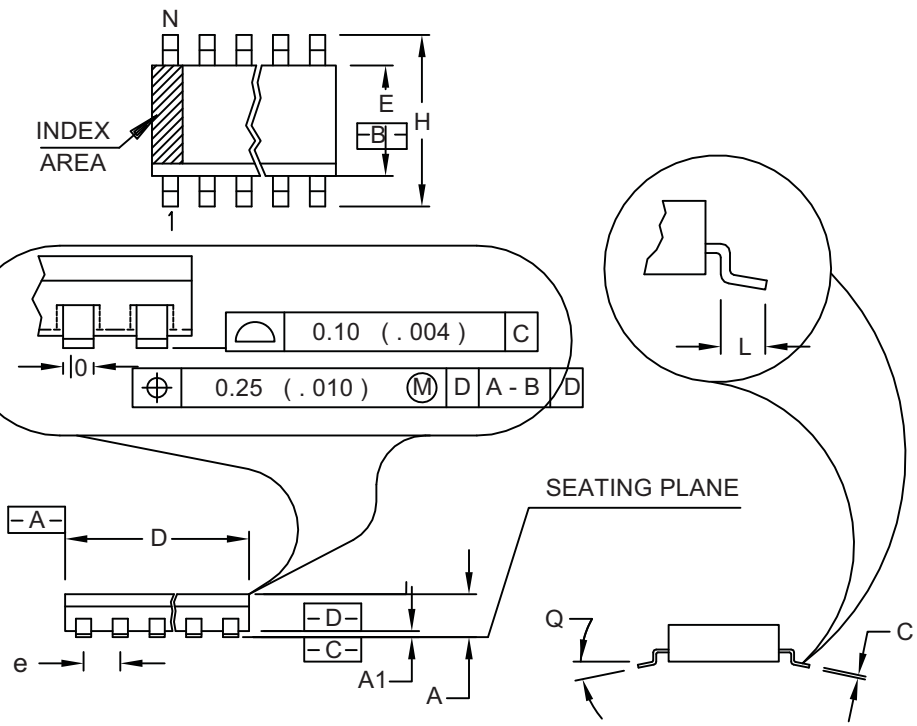


	MM		INCH	
A	2.35	2.65	.093	.104
A1	0.10	0.30	.004	.012
B	0.35	0.49	.014	.019
C	0.23	0.32	.009	.013
D	12.60	13.00	.496	.512
E	7.40	7.60	.291	.299
e	1.27	BSC	.050	BSC
H	10.00	10.65	.394	.419
h	0.25	0.75	.010	.029
L	0.40	1.27	.016	.050
N	20		20	
Q	0°		8°	

09/10/07

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	TG, 20 Lead, 0.300" Body Width Plastic Gull Wing Small outline Package (SOIC)		TG	N

Figure 28-3. 6G

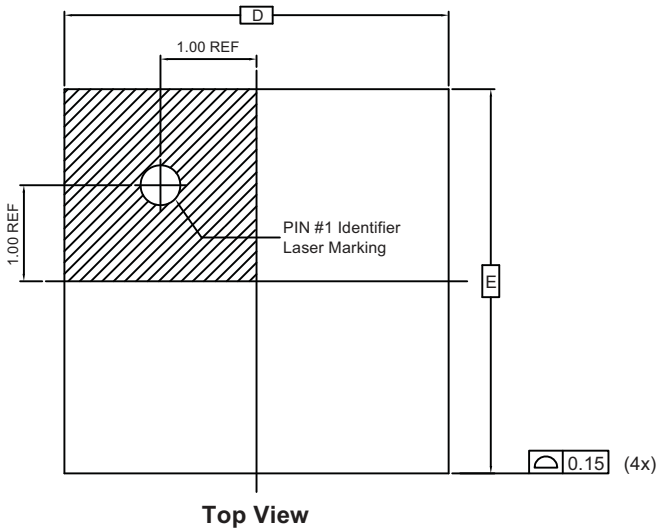


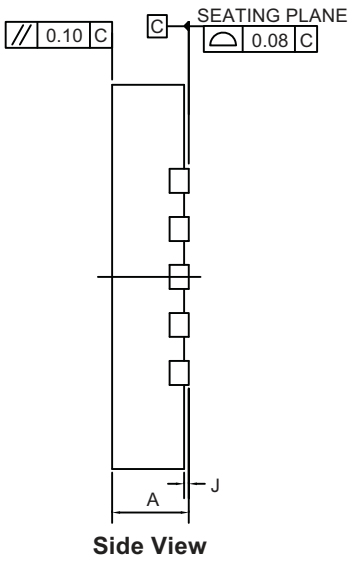
	MM		INCH	
A	----	1.10	----	.043
A1	0.05	0.15	.002	.006
b	0.19	0.30	.007	.012
C	0.09	0.20	.003	.008
D	6.40	6.60	.252	.260
E	4.30	4.50	.169	.177
e	0.65 BSC		.026 BSC	
H	6.40 BSC		.252 BSC	
L	0.50	0.70	.020	.028
N	20		20	
Q	0° ~8°		0° ~8°	

20/12/07

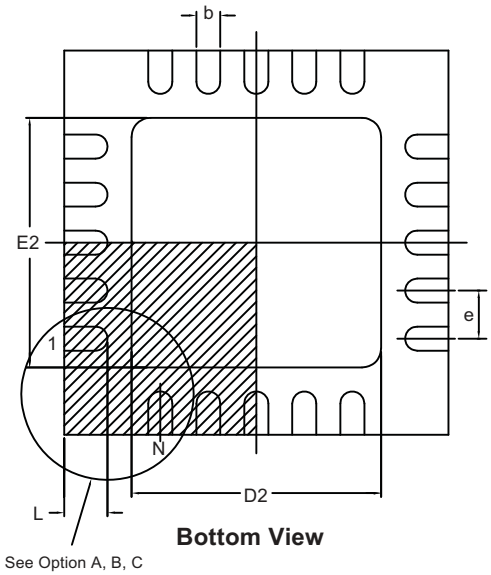
Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	6G, 20 Leads - 4.4x6.5mm Body - 0.65mm Pitch - Lead length: 0.6mm THIN SHRINK SMALL OUTLINE		6G	A

Figure 28-4. PC



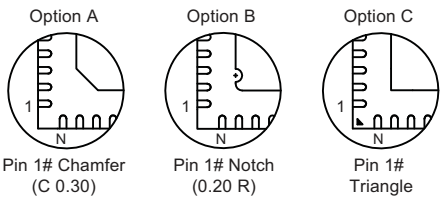


DRAWINGS NOT SCALED



COMMON DIMENSIONS IN MM

SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.70	0.75	0.80	
J	0.00	----	0.05	
D/E	3.90	4.00	4.10	
D2/E2	2.50	2.60	2.70	
N	20			
e	0.50 BSC			
L	0.35	0.45	0.55	
b	0,20	0.25	0.30	



Compliant JEDEC Standard MO-220 Variation WGGD-5

06/25/09

Atmel **Package Drawing Contact:** packagedrawings@atmel.com	TITLE PC, 20-Lead - 4.0x4.0mm Body, 0.50mm Pitch Quad Flat No Lead Package (QFN)	GPC	DRAWING NO. PC	REV. I

29. Errata

29.1 Errata ATtiny261

The revision letter in this section refers to the revision of the ATtiny261 device.

29.1.1 Rev A

Trigger reference levels of the analog comparator.

29.2 Errata ATtiny461

The revision letter in this section refers to the revision of the ATtiny461 device.

29.2.1 Rev B

Trigger reference levels of the analog comparator.

29.3 Errata ATtiny861

The revision letter in this section refers to the revision of the ATtiny861 device.

29.3.1 Rev B

Trigger reference levels of the analog comparator.

29.4 Errata Description

29.4.1 Trigger Reference Levels of the Analog Comparator

The analog comparator should not used with trigger reference levels of less than 300mV with respect to GND.

Problem Fix/Workaround

No workaround.

30. 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
7753G-AVR-06/14	• Put datasheet in the latest template
7753F-AVR-01/11	• Trigger reference levels of the analog comparator errata added
7753E-AVR-06/10	• Ordering Information updated • DC Characteristics table updated
7753D-AVR-11/09	• Ordering Information updated • QFN pinout added • Internal RC Oscillator Accuracy updated
7753C-AVR-07/09	• QFN package added • ADC characteristics updated • Temps sensor updated • Typical characteristics updated
7753B-AVR-08/08	• Added 6G product offering to Ordering Information.
7753A-AVR-11/07	• First datasheet draft - initial automotive version. Started from industrial datasheet doc2588 rev.B - 01/07

31. Table of Contents

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**