ATtiny1634

8-bit AVR Microcontroller with 16KB In-system Programmable Flash

PRELIMINARY DATASHEET

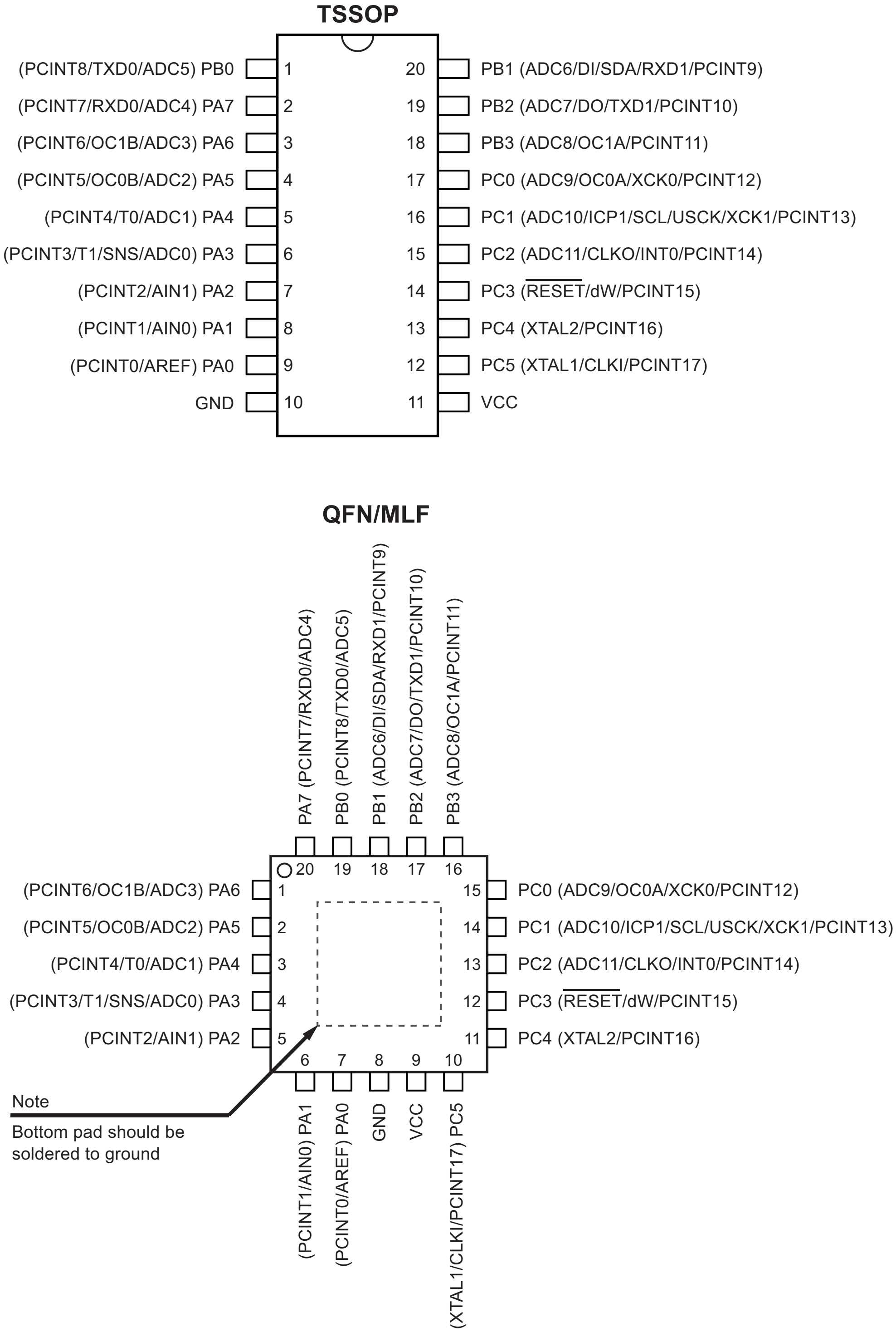
Features

- High-performance, low-power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 125 powerful instructions – most single-clock-cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
- High-endurance, nonvolatile memory segments
  - 16KB of in-system, self programmable Flash program memory
    - Endurance: 10,000 write/erase cycles
  - 256 bytes of in-system programmable EEPROM
    - Endurance: 100,000 write/erase cycles
  - 1KB of internal SRAM
  - Data retention: 20 years at 85°C/100 years at 25°C
  - Programming lock for self programming Flash and EEPROM data security
- Peripheral features
  - Dedicated hardware and QTouch® Library support for capacitive touch sensing
  - One 8-bit and one 16-bit Timer/Counter with two PWM channels, each
  - 12-channel, 10-bit ADC
  - Programmable ultra-low-power watchdog timer
  - On-chip analog comparator
  - Two full duplex USARTs with start frame detection
  - Universal serial interface
  - Slave I^2C serial interface
- Special microcontroller features
  - debugWIRE on-chip debug system
  - In-system programmable via SPI port
  - Internal and external interrupt sources
    - Pin change interrupt on 18 pins
- Low-power idle, ADC noise reduction, standby and power-down modes
- Enhanced power-on reset circuit
- Programmable brown-out detection circuit with supply voltage sampling
- Calibrated 8MHz oscillator with temperature calibration option
- Calibrated 32kHz ultra-low-power oscillator
- On-chip temperature sensor

- I/O and packages
  - 18 programmable I/O lines
  - 20-pad QFN/MLF and 20-pin TSSOP
- Operating voltage:
  - 2.7V to 5.5V
- Speed grade:
  - 0 to 8MHz at 2.7V to 5.5V
  - 0 to 12MHz at 4.5V to 5.5V
- Temperature range: –40°C to +125°C
- Low power consumption
  - Active mode: 1.2mA at 3V and 4MHz
  - Idle mode: 0.23mA at 3V and 4MHz
  - Power-down mode (WDT-enabled): 2µA at 3V
  - Power-down mode (WDT-disabled): 150nA at 3V

1. Pin Configurations

Figure 1-1. Pinout of Atmel ATtiny1634


TSSOP
(PCINT8/TXD0/ADC5) PB0 1
(PCINT7/RXD0/ADC4) PA7 2
(PCINT6/OC1B/ADC3) PA6 3
(PCINT5/OC0B/ADC2) PA5 4
(PCINT4/T0/ADC1) PA4 5
(PCINT3/T1/SNS/ADC0) PA3 6
(PCINT2/AIN1) PA2 7
(PCINT1/AIN0) PA1 8
(PCINT0/AREF) PA0 9
GND 10
20 PB1 (ADC6/DI/SDA/RXD1/PCINT9)
19 PB2 (ADC7/DO/TXD1/PCINT10)
18 PB3 (ADC8/OC1A/PCINT11)
17 PC0 (ADC9/OC0A/XCK0/PCINT12)
16 PC1 (ADC10/ICP1/SCL/USCK/XCK1/PCINT13)
15 PC2 (ADC11/CLKO/INT0/PCINT14)
14 PC3 ($\overline{\text{RESET}}$/dW/PCINT15)
13 PC4 (XTAL2/PCINT16)
12 PC5 (XTAL1/CLKI/PCINT17)
11 VCC
QFN/MLF
PA7 (PCINT7/RXD0/ADC4) 20
PB0 (PCINT8/TXD0/ADC5) 19
PB1 (ADC6/DI/SDA/RXD1/PCINT9) 18
PB2 (ADC7/DO/TXD1/PCINT10) 17
PB3 (ADC8/OC1A/PCINT11) 16
(PCINT6/OC1B/ADC3) PA6 1
(PCINT5/OC0B/ADC2) PA5 2
(PCINT4/T0/ADC1) PA4 3
(PCINT3/T1/SNS/ADC0) PA3 4
(PCINT2/AIN1) PA2 5
15 PC0 (ADC9/OC0A/XCK0/PCINT12)
14 PC1 (ADC10/ICP1/SCL/USCK/XCK1/PCINT13)
13 PC2 (ADC11/CLKO/INT0/PCINT14)
12 PC3 ($\overline{\text{RESET}}$/dW/PCINT15)
11 PC4 (XTAL2/PCINT16)
6 (PCINT1/AIN0) PA1
7 (PCINT0/AREF) PA0
8 GND
9 VCC
10 (XTAL1/CLKI/PCINT17) PC5

Note

Bottom pad should be soldered to ground

1.1 Pin Descriptions

1.1.1 VCC

Supply voltage.

1.1.2 GND

Ground.

1.1.3 XTAL1

Input to the inverting amplifier of the oscillator and the internal clock circuit. This is an alternative pin configuration of PC5.

1.1.4 XTAL2

Output from the inverting amplifier of the oscillator. Alternative pin configuration of PC4.

1.1.5 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length generates a reset, even if the clock is not running, provided the reset pin has not been disabled. The minimum pulse length is given in Table 25-5 on page 218. Shorter pulses are not guaranteed to generate a reset. The reset pin can also be used as a (weak) I/O pin.

1.1.6 Port A (PA7:PA0)

This is an 8-bit, bidirectional I/O port with internal pull-up resistors (selected for each bit). Output buffers have the following drive characteristics:

- PA7, PA4:PA0: Symmetrical, with standard sink and source capability
- PA6, PA5: Asymmetrical, with high sink and standard source capability

As inputs, port pins that are externally pulled low draw current, provided that pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port has alternate pin functions to serve special features of the device (see Section 11.3.1 "Alternate Functions of Port A" on page 61).

1.1.7 Port B (PB3:PB0)

This is a 4-bit, bidirectional I/O port with internal pull-up resistors (selected for each bit).Output buffers have the following drive characteristics:

- PB3: Asymmetrical, with high sink and standard source capability
- PB2:PB0: Symmetrical, with standard sink and source capability

As inputs, port pins that are externally pulled low draw current, provided that pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port has alternate pin functions to serve special features of the device (see Section 11.3.2 "Alternate Functions of Port B" on page 64).

1.1.8 Port C (PC5:PC0)

This is a 6-bit, bidirectional I/O port with internal pull-up resistors (selected for each bit). Output buffers have the following drive characteristics:

- PC5:PC1: Symmetrical, with standard sink and source capability
- PC3: Weak sink/source
- PC0: Asymmetrical, with high sink and standard source capability

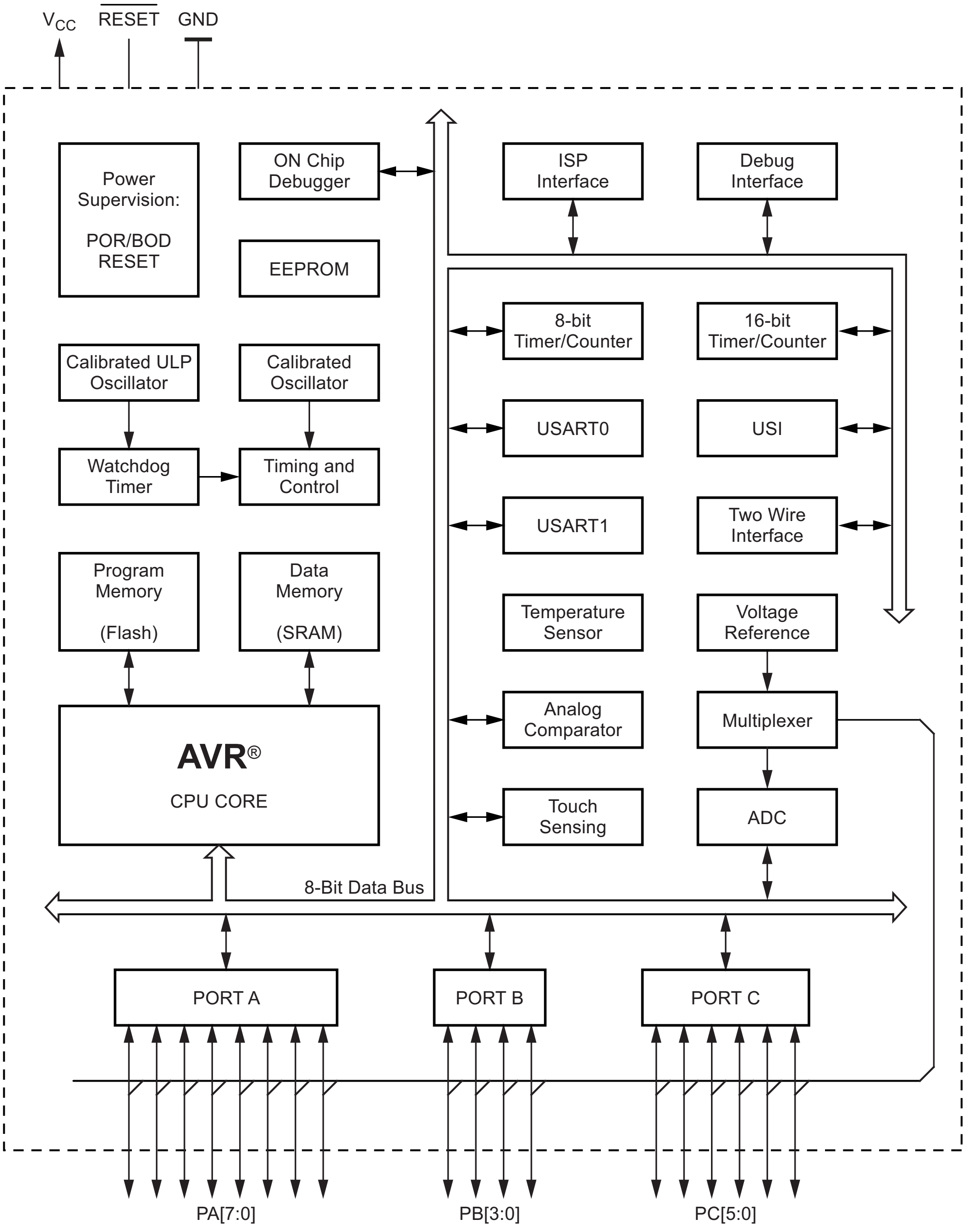
As inputs, port pins that are externally pulled low draw current, provided that pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port has alternate pin functions to serve special features of the device (see Section 11.3.3 "Alternate Functions of Port C" on page 66).

2. Overview

The Atmel® ATtiny1634 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 ATtiny1634 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Figure 2-1. Block Diagram



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

The Atmel® ATtiny1634 provides the following features:

- 16KB of in-system programmable Flash
- 1KB of SRAM data memory
- 256 bytes of EEPROM data memory
- 18 general purpose I/O lines
- 32 general purpose working registers
- An 8-bit Timer/Counter with two PWM channels
- A16-bit Timer/Counter with two PWM channels
- Internal and external interrupts
- A 10-bit ADC with 5 internal and 12 external channels
- An ultra-low-power, programmable watchdog timer with internal oscillator
- Two programmable USARTs with start frame detection
- A slave two-wire interface (TWI)
- A universal serial interface (USI) with start condition detector
- A calibrated 8MHz oscillator
- A calibrated 32kHz, ultra-low-power oscillator
- Four software-selectable power saving modes

The device includes the following modes for saving power:

- Idle mode: stops the CPU while allowing the Timer/Counter, ADC, analog comparator, SPI, TWI, and interrupt system to continue functioning.
- ADC noise reduction mode: minimizes switching noise during ADC conversions by stopping the CPU and all I/O modules except the ADC
- Power-down mode: registers keep their contents and all chip functions are disabled until the next interrupt or hardware reset.
- Standby mode: the oscillator is running while the rest of the device is sleeping, allowing very fast start-up combined with low power consumption.

The device is manufactured using high density nonvolatile memory technology from Atmel. The Flash program memory can be re-programmed in-system through a serial interface, by a conventional nonvolatile memory programmer or by an on-chip boot code running on the AVR core.

The Atmel ATtiny1634 AVR is supported by a full suite of program and system development tools including C compilers, macro assemblers, program debugger/simulators, and evaluation kits.

3. Automotive Quality Grade

The Atmel® ATtiny1634 has been developed and manufactured according to the most stringent requirements of the international ISO-TS-16949 grade 1 standard. This datasheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the Atmel ATtiny1634 have been verified during regular product qualification in compliance with AEC-Q100.

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

Table 3-1. Temperature Grade Identification for Automotive Products

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

4. General Information

4.1 Resources

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

4.2 Code Examples

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

For I/O registers located in the extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI" and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically, this means "LDS" and "STS" combined with "SBRS", "SBRC", "SBR" and "CBR". Note that not all AVR® devices include an extended I/O map.

4.3 Capacitive Touch Sensing

Atmel QTouch® Library provides an easy-to-use solution for touch-sensitive interfaces on Atmel AVR microcontrollers. QTouch Library includes support for QTouch and QMatrix acquisition methods.

Touch sensing is easily added to any application by linking the QTouch Library and using the application programming interface (API) of the library to define the touch channels and sensors. The application then calls the API to retrieve channel information and determine the state of the touch sensor.

QTouch Library is free and can be downloaded from the Atmel website. For more information and implementation details, see the QTouch Library User Guide—also available from the Atmel website.

4.4 Data Retention

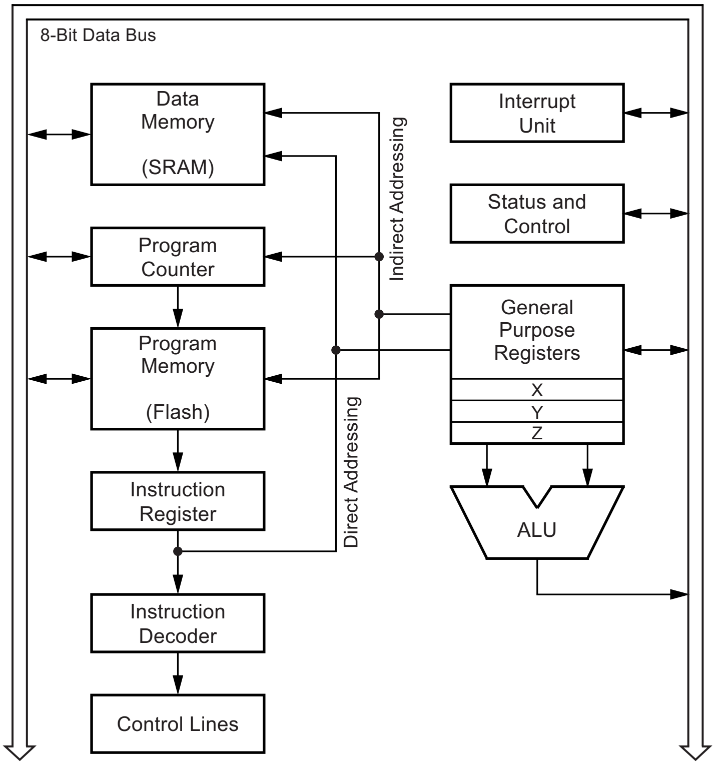
Reliability qualification results show that the projected data retention failure rate is much less than 1PPM over 20 years at 85°C or 100 years at 25°C.

5. CPU Core

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

5.1 Architectural Overview

Figure 5-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture—with separate memories and buses for program and data. Instructions in the program memory are executed with 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 an 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 these address pointers can also be used as an address pointer for lookup tables in Flash program memory. These added function registers are the 16-bit X, Y, and Z register described later in this section.

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

Program flow is provided by conditional and unconditional jump and call instructions capable of directly addressing the whole address space. Most AVR instructions have a single 16-bit word format, but 32-bit-wide instructions also exist. The actual instruction set varies because some devices only implement part of the instruction set.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM and as a result 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. In addition, the Atmel® ATtiny1634 has extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

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. For more information, see the external "AVR Instruction Set" document and Section 28. "Instruction Set Summary" on page 248.

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. In many cases this makes using the dedicated compare instructions unnecessary, resulting in faster and more compact code. For more information, see the external "AVR Instruction Set" document and Section 28. "Instruction Set Summary" on page 248.

The status register is neither automatically stored when entering an interrupt routine nor restored when returning from an interrupt. This must be handled by software.

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 below shows the structure of the 32 general purpose working registers in the CPU.

Figure 5-2. General Purpose Working Registers

7 0	Addr.	Special Function
R0	0x00	
R1	0x01	
R2	0x02	
R3	0x03	
…	...	
R12	0x0C	
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
…	...	
R26	0x1A	X register low byte
R27	0x1B	X register high byte
R28	0x1C	Y register low byte
R29	0x1D	Y register high byte
R30	0x1E	Z register low byte
R31	0x1F	Z register high byte

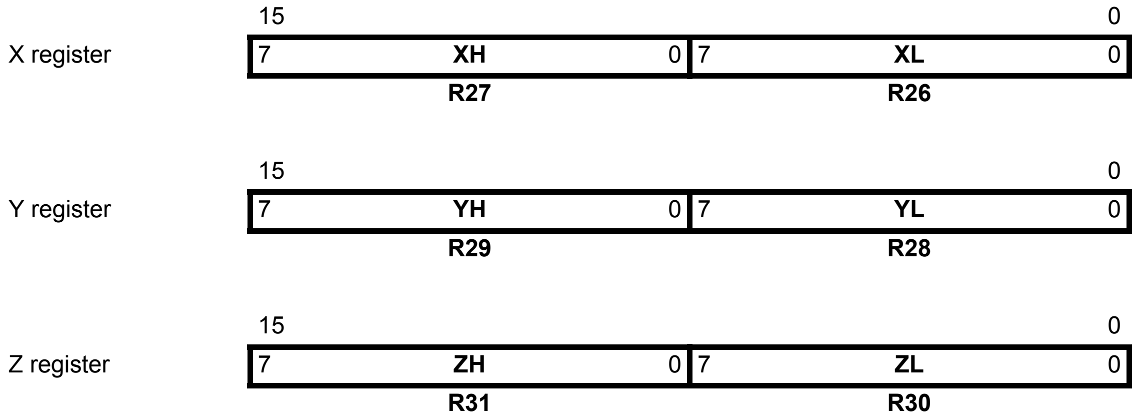
Most of the instructions operating on the register file have direct access to all registers, and most of them are single-cycle instructions.

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

5.4.1 The X, Y, and Z registers

The registers R26..R31 have functions going beyond 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 below.

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



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

5.5 Stack Pointer

The stack is mainly used for storing temporary data, local variables, and return addresses after interrupts and subroutine calls. The stack pointer registers (SPH and SPL) always point to the top of the stack. Note that the stack grows from higher memory locations to lower memory locations. This means that the PUSH instructions decreases and the POP instruction increases the stack pointer value.

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

The pointer is decremented by one when data is put on the stack with the PUSH instruction, and incremented by one when data is fetched with the POP instruction. It is decremented by two when the return address is put on the stack by a subroutine call or a jump to an interrupt service routine, and incremented by two when data is fetched by a return from a subroutine (the RET instruction) or a return from an interrupt service routine (the RETI instruction).

The AVR® stack pointer is typically implemented as two 8-bit registers in the I/O register file. The width of the stack pointer and the number of bits implemented is device-dependent. In some AVR devices all data memory can be addressed using SPL only. In this case the SPH register is not implemented.

The stack pointer must be set to point above the I/O register areas, the minimum value being the lowest SRAM address. For more information, see Table 6-2 on page 17.

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} which is generated directly from the selected chip clock source. 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 for obtaining 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

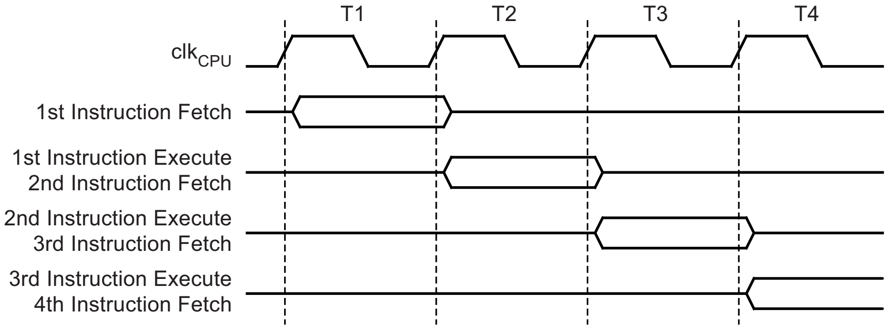
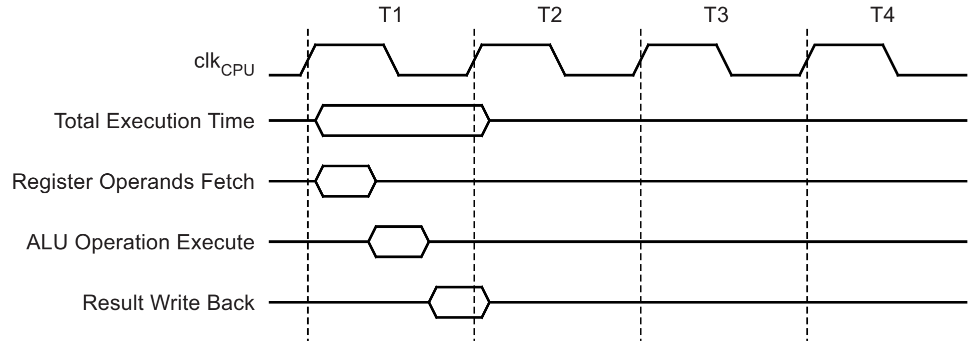


Figure 5-5 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed with the result stored back to the destination register.

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 defined as the reset and interrupt vectors by default. 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 the priority level. RESET has the highest priority, followed by 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 is 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) is (are) set and remembered until the global interrupt enable bit is set, at which point it is executed based on its priority.

The second type of interrupts triggers 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 is not triggered.

When the AVR exits from an interrupt, it always returns to the main program and executes 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 are immediately disabled. No interrupt is 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) */
```

Note: See Section 4.2 “Code Examples” on page 7.

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

Assembly Code Example

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

C Code Example

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

Note: See Section 4.2 “Code Examples” on page 7.

5.7.1 Interrupt Response Time

The minimum interrupt execution response for all the enabled AVR® interrupts is four clock cycles. 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 sleep mode selected.

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.

5.8 Register Description

5.8.1 CCP – Configuration Change Protection Register

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	**CCP[7:0]**								**CCP**
Read/Write	W	W	W	W	W	W	W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

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

When the protected I/O register signature is written, CCP0 reads as “1” as long as the protected feature is enabled, while CCP[7:1] always reads as “0”.

Table 5-1 shows the signatures recognized.

Table 5-1. Signatures Recognized by the Configuration Change Protection Register

Signature	Registers	Description
0xD8	CLKSR, CLKPR, WDTCSR[1]	Protected I/O register

Note: 1. Only WDE and WDP[3:0] bits are protected in WDTCSR.

5.8.2 SPH and SPL — Stack Pointer Registers

Initial Value	0	0	0	0	0	RAMEND	RAMEND	RAMEND	
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**–**	**–**	**–**	**–**	**–**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
Bit	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	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

- **Bits 10:0 – SP[10:0]: Stack Pointer**

The stack pointer register points to the top of the stack, which is implemented growing from higher memory locations to lower memory locations. Thus, a “stack PUSH” command decreases the stack pointer.

The stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled.

5.8.3 SREG – Status Register

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

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

The global interrupt enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the global interrupt enable register is cleared, none of the interrupts are enabled regardless 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 the 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. For more information, see the “Instruction Set Description” section.

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

The S bit is always an exclusive or between the negative flag N and the two’s complement overflow flag V. For more information, see the “Instruction Set Description” section.

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

The two’s complement overflow flag V supports two’s complement arithmetics. For more information, see the “Instruction Set Description” section.

- **Bit 2 – N: Negative Flag**

The negative flag N indicates a negative result in an arithmetic or logic operation. For more information, see the “Instruction Set Description” section.

- **Bit 1 – Z: Zero Flag**

The zero flag Z indicates a zero result in an arithmetic or logic operation. For more information, see the “Instruction Set Description” section.

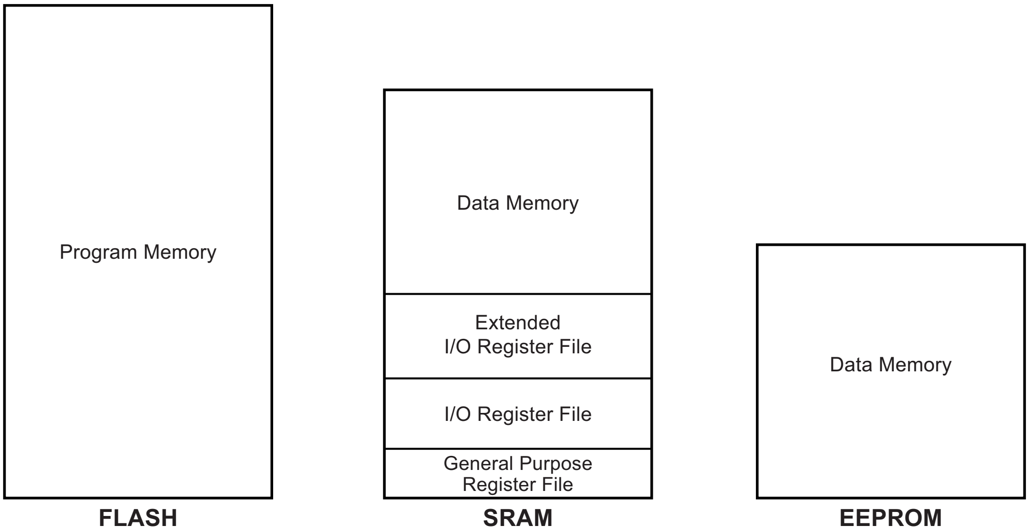
- **Bit 0 – C: Carry Flag**

The carry flag C indicates a carry in an arithmetic or logic operation. For more information, see the “Instruction Set Description” section.

6. Memories

The AVR® architecture makes a distinction between program memory and data memory, locating each memory type in a separate address space. Executable code is located in nonvolatile program memory (Flash), whereas data can be placed in either volatile (SRAM) or nonvolatile memory (EEPROM) (see Figure 6-1).

Figure 6-1. Memory Overview



All memory spaces are linear and regular.

6.1 Program Memory (Flash)

The Atmel® ATtiny1634 contains 16KB of on-chip, in-system reprogrammable Flash memory for program storage. Flash memories are nonvolatile, i.e., they retain stored information even when not powered.

Because all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8192 x 16 bits. The program counter (PC) is 13 bits wide and thus capable of addressing all 8192 locations of program memory as illustrated in Table 6-1.

Table 6-1. Size of Program Memory (Flash)

Device	Flash Size		Address Range
Atmel ATtiny1634	16KB	8192 words	0x0000 – 0x1FFF

Constant tables can be allocated within the entire address space of program memory. See the LPM (load program memory) and SPM (store program memory) instructions in Section 28. "Instruction Set Summary" on page 248. Flash program memory can also be programmed from an external device as described in Section 24. "External Programming" on page 202.

Timing diagrams for instruction fetch and execution are presented in Section 5.6 "Instruction Execution Timing" on page 11.

The Flash memory has a minimum endurance of 10,000 write/erase cycles.

6.2 Data Memory (SRAM) and Register Files

Table 6-2 shows how the data memory and register files of the Atmel® ATtiny1634 are organized. These memory areas are volatile, i.e., they do not retain information when power is off or interrupted.

Table 6-2. Layout of Data Memory and Register Area

Device	Memory Area	Size	Long Address [1]	Short Address [2]
Atmel ATtiny1634	General purpose register file	32B	0x0000 – 0x001F	n/a
	I/O register file	64B	0x0020 – 0x005F	0x00 – 0x3F
	Extended I/O register file	160B	0x0060 – 0x00FF	n/a
	Data SRAM	1024B	0x0100 – 0x04FF	n/a

Note: 1. Also known as data address. This mode of addressing covers the entire data memory and register area. The address is contained in a 16-bit area of two-word instructions.

2. Also known as direct I/O address. This mode of addressing covers part of the register area only. It is used by instructions where the address is embedded in the instruction word.

The 1280 memory locations include the general purpose register file, I/O register file, extended I/O register file, and the internal data memory.

For compatibility with future devices, reserved bits should be written to "0" if accessed. Reserved I/O memory addresses should never be written.

6.2.1 General Purpose Register File

The first 32 locations are reserved for the general purpose register file. These registers are described in detail in Section 5.4 "General Purpose Register File" on page 9.

6.2.2 I/O Register File

Following the general purpose register file, the next 64 locations are reserved for I/O registers. Registers in this area are used mainly for communicating with I/O and peripheral units of the device. Data can be transferred between I/O space and the general purpose register file using instructions such as IN, OUT, LD, ST, and derivatives.

All I/O registers in this area can be accessed with the instructions IN and OUT. These I/O specific instructions address the first location in the I/O register area as 0x00 and the last as 0x3F.

The low 32 registers (address range 0x00...0x1F) are accessible by some bit-specific instructions. In these registers, bits are easily set and cleared using SBI and CBI, while bit-conditional branches are readily constructed using SBIC, SBIS, SBRC, and SBRS instructions.

Registers in this area may also be accessed with LD/LDD/LDS and ST/STD/STS instructions. These instructions treat the entire volatile memory as one data space and as a result address I/O registers starting at 0x20.

For more information, see Section 28. "Instruction Set Summary" on page 248.

The Atmel ATtiny1634 also contains three general purpose I/O registers that can be used for storing any information (see GPIOR0, GPIOR1, and GPIOR2 in Section 27. "Register Summary" on page 245). These general purpose I/O registers are particularly useful for storing global variables and status flags because they are accessible to bit-specific instructions such as SBI, CBI, SBIC, SBIS, SBRC, and SBRS.

6.2.3 Extended I/O Register File

Following the standard I/O register file, the next 160 locations are reserved for extended I/O registers. The ATtiny1634 is a complex microcontroller with more peripheral units than can be addressed with the IN and OUT instructions. Registers in the extended I/O area must be accessed using LD/LDD/LDS and ST/STD/STS instructions (see Section 28. "Instruction Set Summary" on page 248).

See Section 27. "Register Summary" on page 245 for a list of I/O registers.

6.2.4 Data Memory (SRAM)

Following the general purpose register file and the I/O register files, the remaining 1024 locations are reserved for the internal data SRAM.

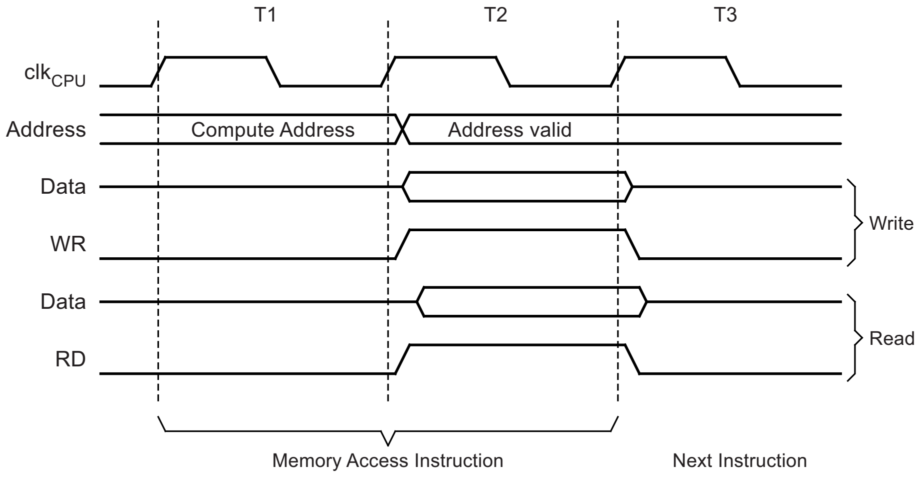
There are five addressing modes available:

- Direct – this mode of addressing reaches the entire data space.
- Indirect
- Indirect with displacement – this mode of addressing reaches 63 address locations from the base address given by the Y or Z register.
- Indirect with pre-decrement – in this mode the address register is automatically decremented before access. Address pointer registers (X, Y, and Z) are located in the general purpose register file, in registers R26 to R31 (see Section 5.4 "General Purpose Register File" on page 9).
- Indirect with post-increment – in this mode the address register is automatically incremented after access. Address pointer registers (X, Y, and Z) are located in the general purpose register file, in registers R26 to R31 (see Section 5.4 "General Purpose Register File" on page 9).

All addressing modes can be used on the entire volatile memory, including the general purpose register file, the I/O register files, and the data memory.

Internal SRAM is accessed in two clk_{CPU} cycles as illustrated below in Figure 6-2.

Figure 6-2. On-chip Data SRAM Access Cycles



6.3 Data Memory (EEPROM)

The Atmel® ATtiny1634 contains 256 bytes of nonvolatile data memory. This EEPROM is organized as a separate data space in which single bytes can be read and written. All access registers are located in the I/O space.

The EEPROM memory layout is summarized in Table 6-3.

Table 6-3. Size of Nonvolatile Data Memory (EEPROM)

Device	EEPROM Size	Address Range
Atmel ATtiny1634	256B	0x00 – 0xFF

The internal 8MHz oscillator is used to time EEPROM operations. The frequency of the oscillator must be within the requirements as described in Section 7.5.3 "OSCCAL0 – Oscillator Calibration Register" on page 32.

When powered by heavily filtered supplies, the supply voltage, V_{CC}, is likely to rise or fall slowly on power-up and power-down. Slow rise and fall times may put the device in a state where it is running at supply voltages lower than specified. To avoid problems in such situations, see Section 6.3.5 “Preventing EEPROM Corruption” on page 20.

The EEPROM has a minimum endurance of 100,000 write/erase cycles.

6.3.1 Programming Methods

There are two methods for EEPROM programming:

- Atomic byte programming – this is the simple mode of programming where target locations are erased and written in a single operation. In this operating mode the target is guaranteed to always be erased before writing but programming times are longer.
- Split byte programming – it is possible to split the erase and write cycle into two different operations. This is useful when short access times are required, for example, when supply voltage is falling. In order to take advantage of this method, target locations must be erased before writing to them. This can be done at times when the system allows time-critical operations, typically at start-up and initialization.

The programming method is selected using the EEPROM programming mode bits (EEPM1 and EEPM0) in the EEPROM control register (EECR) (see Table 6-4 on page 23). The write and erase times are given in the same table.

Because EEPROM programming takes some time, the application must wait for one operation to complete before starting the next. This can be done by either polling the EEPROM program enable bit (EEPE) in the EEPROM control register (EECR), or via the EEPROM ready interrupt. The EEPROM interrupt is controlled by the EEPROM ready interrupt enable (EERIE) bit in EECR.

6.3.2 Read

To read an EEPROM memory location, follow the procedure below:

1. Poll the EEPROM program enable bit (EEPE) in the EEPROM control register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
2. Write the target address to the EEPROM address registers (EEARH/EEARL).
3. Start the read operation by setting the EEPROM read enable bit (EERE) in the EEPROM control register (EECR). During the read operation, the CPU is stopped for four clock cycles before executing the next instruction.
4. Read data from the EEPROM data register (EEDR).

6.3.3 Erase

In order to prevent unintentional EEPROM writes, a specific procedure must be followed to erase memory locations. To erase an EEPROM memory location, follow the procedure below:

1. Poll the EEPROM program enable bit (EEPE) in the EEPROM control register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
2. Set mode of programming to erase by writing the EEPROM programming mode bits (EEPM0 and EEPM1) in the EEPROM control register (EECR).
3. Write the target address to the EEPROM address registers (EEARH/EEARL).
4. Enable erase by setting the EEPROM master program enable (EEMPE) in the EEPROM control register (EECR). Within four clock cycles, start the erase operation by setting the EEPROM program enable bit (EEPE) in the EEPROM control register (EECR). During the erase operation, the CPU is stopped for two clock cycles before executing the next instruction.

The EEPE bit remains set until the erase operation has completed. While the device is busy programming, it is not possible to perform any other EEPROM operations.

6.3.4 Write

In order to prevent unintentional EEPROM writes, a specific procedure must be followed to write to memory locations.

Before writing data to EEPROM, the target location must be erased. This can be done either in the same operation or as part of a split operation. Writing to an unerased EEPROM location results in corrupted data.

To write an EEPROM memory location, follow the procedure below:

1. Poll the EEPROM program enable bit (EEPE) in the EEPROM control register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
2. Set the mode of programming by writing the EEPROM programming mode bits (EEPM0 and EEPM1) in the EEPROM control register (EECR). Alternatively, data can be written in one operation or the write procedure can be split up in erase only and write only.
3. Write the target address to the EEPROM address registers (EEARH/EEARL).
4. Write the target data to the EEPROM data register (EEDR).
5. Enable write by setting the EEPROM master program enable (EEMPE) in the EEPROM control register (EECR). Within four clock cycles, start the write operation by setting the EEPROM program enable bit (EEPE) in the EEPROM control register (EECR). During the write operation, the CPU is stopped for two clock cycles before executing the next instruction.

The EEPE bit remains set until the write operation has completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

6.3.5 Preventing EEPROM Corruption

During periods of low V_{CC}, the EEPROM data may become 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; thus the same design solutions should be applied.

At low supply voltages data in EEPROM may become corrupted in two ways:

- The supply voltage is too low to maintain proper operation of an otherwise legitimate EEPROM program sequence.
- The supply voltage is too low for the CPU and instructions may be executed incorrectly.

EEPROM data corruption is avoided by keeping the device in reset during periods of insufficient power supply voltage. This is easily done by enabling the internal brown-out detector (BOD). If BOD detection levels are not sufficient for the design, an external reset circuit for low V_{CC} can be used.

Provided that supply voltage is sufficient, an EEPROM write operation is completed even when a reset occurs.

6.3.6 Program Examples

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 (such as by disabling interrupts globally) so that no interrupts 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 registers
        out    EEARH, r18
        out    EEARL, r17
        ; Write data (r19) to data register
        out    EEDR, r19
        ; Write logical one to EEMPE
        sbi    EECR, EEMPE
        ; Start eeprom write by setting EEPE
        sbi    EECR, EEPE
        ret
```

Note: See Section 4.2 “Code Examples” on page 7.

C Code Example

```
void EEPROM_write(unsigned int 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);
}
```

Note: See Section 4.2 “Code Examples” on page 7.

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts 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 registers
        out    EEARH, r18
        out    EEARL, r17
        ; Start eeprom read by writing EERE
        sbi    EECR, EERE
        ; Read data from data register
        in     r16, EEDR
        ret
```

Note: See Section 4.2 "Code Examples" on page 7.

C Code Example

```
unsigned char EEPROM_read(unsigned int 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;
}
```

Note: See Section 4.2 "Code Examples" on page 7.

6.4 Register Description

6.4.1 EEARL – EEPROM Address Register Low

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

- **Bits 7:0 – EEAR[7:0]: EEPROM Address**

The EEPROM address register is required by the read and write operations to indicate the memory location being accessed.

EEPROM data bytes are addressed linearly over the entire memory range (0...[256-1]). The initial value of these bits is undefined and a legitimate value must therefore be written to the register before EEPROM is accessed.

Devices with 256 bytes of EEPROM, or less, do not require a high address register (EEARH). In such devices the high address register is therefore left out but, for compatibility issues, the remaining register is still referred to as the low byte of the EEPROM address register (EEARL).

Devices that do not fill an entire address byte, i.e., devices with an EEPROM size less than 256, implement read-only bits in the unused locations. Unused bits are located in the most significant end of the address register and always read as "0".

6.4.2 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 – EEDR[7:0]: EEPROM Data**

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

6.4.3 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	

- **Bits 7, 6 – Res: Reserved Bits**

These bits are reserved and always read as “0”.

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

EEPROM programming mode bits define the action triggered when EEPE is written. Data can be programmed in a single atomic operation, where the previous value is automatically erased before the new value is programmed, or erase and write can be split into two different operations. The programming times for the different modes are shown in Table 6-4.

Table 6-4. EEPROM Programming Mode Bits and Programming Times

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4ms	Atomic (erase and write in one operation)
0	1	1.8ms	Erase only
1	0	1.8ms	Write only
1	1	–	Reserved

When EEPE is set, any write to EEPMn is ignored.

During reset, the EEPMn bits are reset to 0b00 unless the EEPROM is busy programming.

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

Writing this bit to “1” enables the EEPROM ready interrupt. Provided the I bit in SREG is set, the EEPROM ready interrupt is triggered when nonvolatile memory is ready for programming.

Writing this bit to “0” disables the EEPROM ready interrupt.

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

The EEMPE bit determines whether writing EEPE to “1” takes effect or not.

When EEMPE is set and EEPE is written within four clock cycles, the EEPROM at the selected address is programmed. The hardware clears the EEMPE bit to “0” after four clock cycles.

If EEMPE is “0”, the EEPE bit has no effect.

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

This is the programming enable signal of the EEPROM. The EEMPE bit must be set before EEPE is written, otherwise EEPROM is not programmed.

When EEPE is written, the EEPROM is programmed according to the EEPMn bit settings. When EEPE has been set, the CPU is stopped for two cycles before the next instruction is executed. After the write access time has elapsed, the EEPE bit is cleared by the hardware.

Note that an EEPROM write operation blocks all software programming of Flash, fuse bits, and lock bits.

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

This is the read strobe of the EEPROM. When the target address has been set up in the EEAR, the EERE bit must be written to "1" to trigger the EEPROM read operation.

EEPROM read access takes one instruction and the requested data is available immediately. When the EEPROM is read, the CPU is stopped 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 not possible to read the EEPROM or to change the address register (EEAR).

6.4.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	**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	

This register may be used freely for storing any kind of data.

6.4.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	**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	

This register may be used freely for storing any kind of data.

6.4.6 GPIOR0 – General Purpose I/O Register 0

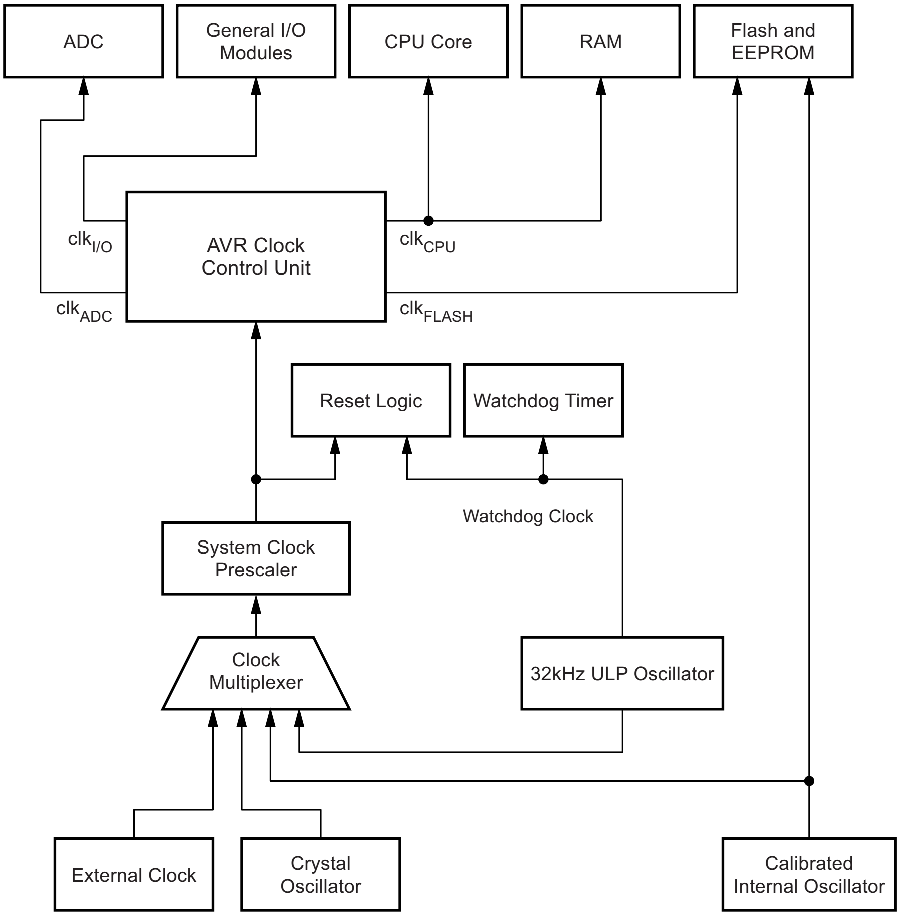
Bit	7	6	5	4	3	2	1	0	
0x14 (0x34)	**MSB**							**LSB**	**GPIOR0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

This register may be used freely for storing any kind of data.

7. Clock System

Figure 7-1 presents the principal clock systems and their distribution in Atmel® ATtiny1634. Not all of the clocks need to be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be stopped by using different sleep modes and power reduction register bits as described in Section 8. "Power Management and Sleep Modes" on page 34. The clock systems can be seen in detail below.

Figure 7-1. Clock Distribution



7.1 Clock Subsystems

The clock subsystems are described in the following sections.

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. Stopping 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, such as the Timer/Counter module. 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 stopped.

7.1.3 Flash Clock – clk_{FLASH}

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

7.1.4 ADC Clock – clk_{ADC}

The ADC has a dedicated clock domain. This allows the CPU and I/O clocks to be stopped in order to reduce noise generated by digital circuitry, resulting in more accurate ADC conversion results.

7.2 Clock Sources

The device can use any of the following sources for the system clock:

- See Section 7.2.1 “External Clock” on page 26
- See Section 7.2.2 “Calibrated Internal 8MHz Oscillator” on page 27
- See Section 7.2.3 “Internal 32kHz Ultra-Low-power (ULP) Oscillator” on page 27
- See Section 7.2.4 “Crystal Oscillator/Ceramic Resonator” on page 27

The clock source is selected using either CKSEL bits in the CLKSR register or CKSEL fuses. The difference between CKSEL fuses and bits is that CKSEL fuses are automatically loaded to CKSEL bits at device power-on or at reset. The initial value of CKSEL bits is therefore determined by CKSEL fuses.

CKSEL fuse bits can be read by firmware (see Section 23.4 “Reading Lock, Fuse, and Signature Data from Software” on page 200), but firmware cannot write to fuse bits. The CKSEL bits must thus be used if the system clock source needs to be changed at run time. The clock system has been designed to guarantee glitch-free performance when switching between clock sources (see Section 7.5.1 “CLKSR – Clock Setting Register” on page 29).

When the device wakes up from power-down, 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, the internal 32kHz oscillator is used for generating an additional delay, allowing supply voltage to reach a stable level before normal device operation is started.

System clock alternatives are discussed in the following sections.

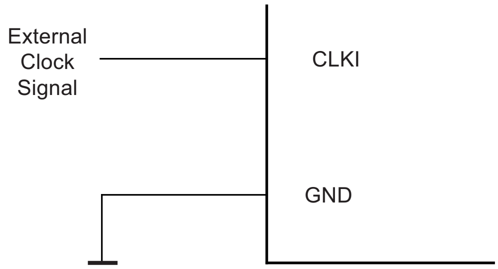
7.2.1 External Clock

To drive the device from an external clock source, CLKI should be connected as shown in Figure 7-2.

Sudden changes in the external clock frequency should always be avoided to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. Ensure that the MCU is kept in reset during changes of clock frequency of this magnitude.

Stable operation for large step changes in system clock frequency is guaranteed when using the system clock prescaler (see Section 7.3 “System Clock Prescaler” on page 28).

Figure 7-2. External Clock Drive Configuration



Start-up time for this clock source is determined by the SUT bit as shown in Table 7-2 on page 29.

7.2.2 Calibrated Internal 8MHz Oscillator

The internal 8MHz oscillator operates with no external components and, by default, provides a clock source with an approximate frequency of 8MHz. Though voltage- and temperature-dependent, this clock can be very accurately calibrated by the user. For more information, see Table 25-2 on page 217, Section 26-39 “Calibrated Oscillator Frequency (Nominal = 1MHz) versus V_{CC}” on page 243 and Section 26-40 “Calibrated Oscillator Frequency (Nominal = 1MHz) versus Temperature” on page 244.

During reset, the hardware loads the preprogrammed calibration value into the OSCCAL0 register and automatically calibrates the oscillator at the same time. For more information on automatic loading of preprogrammed calibration values, see Section 23.3.2 “Calibration Bytes” on page 199.

It is possible to reach higher accuracies than factory defaults, especially when the application allows temperature and voltage ranges to be narrowed. The firmware can reprogram the calibration data in OSCCAL0 either at start-up or during run time. The continuous run-time calibration method allows firmware to monitor voltage and temperature and compensate for any detected variations (see Section 7.5.3 “OSCCAL0 – Oscillator Calibration Register” on page 32, Section 20.12 “Temperature Measurement” on page 182 and Table 20-3 on page 183). The accuracy of this calibration is referred to as “user calibration” in Table 25-2 on page 217.

The OSCTCAL0A and OSCTCAL0B oscillator temperature calibration registers can be used for one-time temperature calibration of oscillator frequency. For more information, see Section 7.5.4 “OSCTCAL0A – Oscillator Temperature Calibration Register A” on page 32 and Section 7.5.5 “OSCTCAL0B – Oscillator Temperature Calibration Register B” on page 33. During reset, hardware loads the preprogrammed calibration values into the OSCTCAL0A and OSCTCAL0B registers.

Start-up time for this clock source is determined by the SUT bit as shown in Table 7-2 on page 29.

7.2.3 Internal 32kHz Ultra-Low-power (ULP) Oscillator

The internal 32kHz oscillator is a low-power oscillator that operates with no external components. It provides a clock source with an approximate frequency of 32kHz. The frequency depends on supply voltage, temperature, and batch variations. For more details on accuracy, see Table 25-3 on page 217.

During reset, the hardware loads the preprogrammed calibration value into the OSCCAL1 register and automatically calibrates the oscillator at the same time. The accuracy of this calibration is referred to as “factory calibration” in Table 25-3 on page 217. For more information on automatic loading of preprogrammed calibration value, see Section 23.3.2 “Calibration Bytes” on page 199.

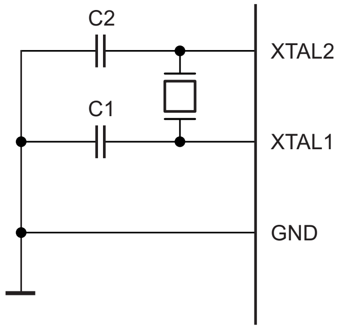
When this oscillator is used as the chip clock, it is still used for the watchdog timer and for the reset time-out.

Start-up time for this clock source is determined by the SUT bit as shown in Table 7-2 on page 29.

7.2.4 Crystal Oscillator/Ceramic Resonator

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 7-3. A quartz crystal or a ceramic resonator may be used.

Figure 7-3. Crystal Oscillator Connections



Capacitors C1 and C2 should always be equal, both for 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 indicated in Table 7-1. The capacitor values given by the manufacturer should be used for ceramic resonators.

Table 7-1. Crystal Oscillator Operating Modes

Frequency Range	Recommended C1 and C2	Note
< 1MHz	–	Crystals only. Not ceramic resonators.
> 1MHz	12 – 22pF	

The oscillator can operate in different modes, each optimized for a specific frequency range (see Table 7-4 on page 31).

Start-up time for this clock source is determined by the SUT bit as shown in Table 7-2 on page 29.

7.2.5 Default Clock Settings

The device is shipped with the following fuse settings:

- Calibrated internal 8MHz oscillator (see CKSEL bits on 30)
- Longest possible start-up time (see SUT bit on 29)
- System clock prescaler set to 8 (see CKDIV8 fuse bit on 198)

The default setting results in a 1MHz system clock and ensures all users can make their desired clock source setting using an
in-system or high-voltage programmer.

7.3 System Clock Prescaler

The Atmel® ATtiny1634 system clock can be divided by selecting the setting shown in Section 7.5.2 “CLKPR – Clock Prescale Register” on page 31. This feature allows power consumption to be reduced when little processing power is required. The feature can also be used with all clock source options and affects 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 shown in Table 7-4 on page 31.

7.3.1 Switching Prescaler Setting

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

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU’s clock frequency. This means 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 could not be predicted exactly.

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

7.4 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT_IO bit has to be programmed. The CKOUT fuse determines the initial value of the CKOUT_IO bit that is loaded to the CLKSR register when the device is powered up or has been reset. The clock output can be switched at run time by setting the CKOUT_IO bit in CLKSR as described in Section 7.5.1 “CLKSR – Clock Setting Register” on page 29.

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

7.5 Register Description

7.5.1 CLKSR – Clock Setting Register

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	OSCRDY	CSTR	CKOUT_IO	SUT	CKSEL3	CKSEL2	CKSEL1	CKSEL0	CLKSR
Read/Write	R	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See bit description					

- **Bit 7 – OSCRDY: Oscillator Ready**

This bit is set when oscillator time-out is complete. When OSCRDY is set, the oscillator is stable and the clock source can be changed safely.

- **Bit 6 – CSTR: Clock Select Trigger**

This bit triggers the clock selection. It can be used to enable the oscillator in advance and select the clock source before the oscillator is stable.

If CSTR is set at the same time as the CKSEL bits are written, the contents are directly copied to the CKSEL register and the system clock is immediately switched.

If CKSEL bits are written without setting CSTR, the oscillator selected by the CKSEL bits is enabled, but the system clock is not yet switched.

- **Bit 5 – CKOUT_IO: Clock Output**

This bit enables the clock output buffer. The CKOUT fuse determines the initial value of the CKOUT_IO bit that is loaded to the CLKSR register when the device is powered up or has been reset.

- **Bit 4 – SUT: Start-Up Time**

The SUT and CKSEL bits define the start-up time of the device as shown in Table 7-2 below. The initial value of the SUT bit is determined by the SUT fuse. The SUT fuse is loaded to the SUT bit when the device is powered up or has been reset.

Table 7-2. Device Start-up Times

SUT	CKSEL	Clock	From Power-Down[1][2]	From Reset [3]
0[4]	0000	External	6 CK	22 CK + 16ms
	0010[4]	Internal 8MHz	6 CK	20 CK + 16ms
	0100	Internal 32kHz	6 CK	22 CK + 16ms
	0001 0011 0101 ... 0111	Reserved		
	1XX0	Ceramic resonator[5]	258 CK[6]	274 CK + 16ms
	1XX1	Crystal oscillator	16K CK	16K CK + 16ms

Note:
1. Device start-up time from power-down sleep mode.
2. When BOD has been disabled by software, the wake-up time from sleep mode is approximately 60µs to ensure the BOD is working correctly before MCU continues executing code.
3. Device start-up time after reset.
4. The device is shipped with this option selected.
5. This option is not suitable for use with crystals.
6. This option should not be used when operating close to the maximum frequency of the device and only if frequency stability at start-up is not important for the application.
7. This option is intended for use with ceramic resonators and will ensure frequency stability at start-up. It 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.

Table 7-2. Device Start-up Times (Continued)

SUT	CKSEL	Clock	From Power-Down[1][2]	From Reset [3]
1	0000 ... 0111 1XX1	Reserved		
	1XX0	Ceramic resonator	1K CK[7]	1K CK +16ms

Note: 1. Device start-up time from power-down sleep mode.

2. When BOD has been disabled by software, the wake-up time from sleep mode is approximately 60µs to ensure the BOD is working correctly before MCU continues executing code.
3. Device start-up time after reset.
4. The device is shipped with this option selected.
5. This option is not suitable for use with crystals.
6. This option should not be used when operating close to the maximum frequency of the device and only if frequency stability at start-up is not important for the application.
7. This option is intended for use with ceramic resonators and will ensure frequency stability at start-up. It 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.

- **Bits 3:0 – CKSEL[3:0]: Clock Select Bits**

These bits select the clock source of the system clock and can be written at run time. The clock system ensures glitch-free switching of the clock source. CKSEL fuses determine the initial value of the CKSEL bits when the device is powered up or reset.

The clock alternatives are shown in Table 7-3 below.

Table 7-3. Device Clocking Options

CKSEL[3:0][1]	Frequency	Device Clocking Option
0000	Any	External Clock (see 26)
0010	8MHz	Calibrated Internal 8MHz Oscillator (see 27)[2]
0100	32kHz	Internal 32kHz Ultra-Low-power (ULP) Oscillator (see 27)
00X1 0101 ... 0111	—	Reserved
100X	0.4...0.9MHz	Crystal Oscillator/Ceramic Resonator (see 27)
101X	0.9...3MHz	
110X	3...8MHz	
111X	> 8MHz	

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

2. This is the default setting. The device is shipped with this fuse combination.

To avoid unintentional switching of clock source, when changing the CKSEL bits, a protected change sequence must be observed as follows:

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

7.5.2 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	–	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR
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 and always read as “0”.

- **Bits 3:0 – CLKPS[3:0]: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written at run time to vary the clock frequency to meet the application requirements. Because 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 found in Table 7-4 on page 31.

To avoid unintentional changes of clock frequency, when changing the CLKPS bits, a protected change sequence must be observed as follows:

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

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

Table 7-4. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1(1)
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8(2)
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	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Note: 1. This is the initial value when the CKDIV8 fuse has been unprogrammed.

2. This is the initial value when the CKDIV8 fuse has been programmed. The device is shipped with the CKDIV8 fuse programmed.

The initial value of clock prescaler bits is determined by the CKDIV8 fuse (see Table 23-5 on page 198). When CKDIV8 is unprogrammed, the system clock prescaler is set to “1” and, when programmed, to “8”. Any value can be written to the CLKPS bits regardless of the CKDIV8 fuse bit setting.

When CKDIV8 is programmed, the initial value of CLKPS bits result in a clock division factor of eight at start-up. This is useful when the selected clock source has a higher frequency than allowed under the existing operating conditions (see Section 25.3 “Speed” on page 217).

7.5.3 OSCCAL0 – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
(0x63)	**CAL07**	**CAL06**	**CAL05**	**CAL04**	**CAL03**	**CAL02**	**CAL01**	**CAL00**	**OSCCAL0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device-specific calibration value								

Although temperature slope and frequency are in part controlled by the OSCTCAL0A and OSCTCAL0B registers, it is possible to replace factory calibration by simply writing to this register only. Optimal accuracy is achieved when OSCCAL0, OSCTAL0A, and OSCTCAL0B are calibrated together.

- **Bits 7:0 – CAL0[7:0]: Oscillator Calibration Value**

The oscillator calibration register is used to trim the internal 8MHz oscillator and to remove process variations from the oscillator frequency. A preprogrammed calibration value is automatically written to this register during chip reset, resulting in the factory-calibrated frequency specified in Table 25-2 on page 217.

The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies specified in Table 25-2 on page 217. Calibration outside this range is not guaranteed.

The lowest oscillator frequency is reached by programming these bits to “0”. Increasing the register value increases the oscillator frequency. A typical frequency response curve is shown in Section 26-38 “Calibrated Oscillator Frequency (Nominal = 8MHz) versus OSCCAL Value” on page 243.

Note that this oscillator is used to time EEPROM and Flash write accesses and that this has a corresponding effect on write times. Do not calibrate to more than 8.8MHz if EEPROM or Flash is to be written. Otherwise, the EEPROM or Flash write may fail.

To ensure stable operation of the MCU, the calibration value should be changed in small steps. A step change in frequency of more than 2% from one cycle to the next can lead to unpredictable behavior. In addition, the difference between two consecutive register values should not exceed 0x20.

7.5.4 OSCTCAL0A – Oscillator Temperature Calibration Register A

Bit	7	6	5	4	3	2	1	0	
(0x64)	**Oscillator Temperature Calibration Data**								**OSCTCAL0A**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device-specific calibration value								

This register is used for changing the temperature slope and frequency of the internal 8MHz oscillator. A preprogrammed calibration value is automatically written to this register during chip reset, resulting in the factory-calibrated frequency specified in Table 25-2 on page 217.

This register does not need updating if factory defaults in OSCCAL0 are overwritten, although optimal accuracy is achieved when OSCCAL0, OSCTAL0A, and OSCTCAL0B are calibrated together.

- **Bit 7 – Sign of Oscillator Temperature Calibration Value**

This is the sign bit of the calibration data.

- **Bits 6:0 – Oscillator Temperature Calibration Value**

These bits contain the numerical value of the calibration data.

7.5.5 OSCTCAL0B – Oscillator Temperature Calibration Register B

Bit	7	6	5	4	3	2	1	0	
(0x65)	Oscillator Temperature Calibration Data								OSCTCAL0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device-specific calibration value								

A preprogrammed calibration value is automatically written to this register during chip reset, resulting in the factory-calibrated frequency specified in Table 25-2 on page 217.

This register does not need updating if factory defaults in OSCCAL0 are overwritten, although optimal accuracy is achieved when OSCCAL0, OSCTAL0A, and OSCTCAL0B are calibrated together.

- **Bit 7 – Temperature Compensation Enable**

When this bit is set, the contents of registers OSCTCAL0A and OSCTCAL0B are used for calibration. When this bit is cleared, the temperature compensation hardware is disabled and the OSCTCAL0A and OSCTCAL0B registers have no effect on the frequency of the internal 8MHz oscillator.

Note that temperature compensation has a strong effect on oscillator frequency and when enabled or disabled, the OSCCAL0 register therefore also must be adjusted to compensate for this effect.

- **Bits 6:0 – Temperature Compensation Step Adjust**

These bits control the step size of the calibration data in OSCTCAL0A. The largest step size is achieved for 0x00 and the smallest step size for 0x7F.

7.5.6 OSCCAL1 – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
(0x66)	–	–	–	–	–	–	CAL11	CAL10	OSCCAL1
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	Device-specific calibration value								

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved and always read as “0”.

- **Bits 1:0 – CAL1[1:0]: Oscillator Calibration Value**

The oscillator calibration register is used to trim the internal 32kHz oscillator and to remove process variations from the oscillator frequency. A preprogrammed calibration value is automatically written to this register during chip reset, resulting in the factory-calibrated frequency as specified in Table 25-3 on page 217.

The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 25-3 on page 217. Calibration outside this range is not guaranteed.

The lowest oscillator frequency is reached by programming these bits to “0”. Increasing the register value increases the oscillator frequency.

8. Power Management and Sleep Modes

Their industry-leading, high-performance code efficiency makes AVR® microcontrollers an ideal choice for low-power applications. In addition, sleep modes enable the application to shut down unused modules in the MCU and thus save power. The AVR provides various sleep modes allowing the user to tailor the power consumption to meet application requirements.

8.1 Sleep Modes

Figure 7-1 on page 25 presents the different clock systems and their distribution in the Atmel® ATtiny1634. The figure is helpful for selecting an appropriate sleep mode. Table 8-1 shows the different sleep modes and the sources that may be used for wake-up.

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

	Oscillators	Active Clock Domains				Wake-Up Sources							
Sleep Mode	Main Clock Source Enabled	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	Watchdog Interrupt	INT0 and Pin Change	SPM/EEPROM Ready Interrupt	ADC Interrupt	USART	USI	TWI Slave	Other I/O
Idle	X			X	X	X	X	X	X	X	X	X	X
ADC noise reduction	X				X	X	X$^{(4)}$	X	X	X$^{(1)}$	X$^{(2)}$	X$^{(3)}$	
Standby	X					X	X$^{(4)}$			X$^{(1)}$	X$^{(2)}$	X$^{(3)}$	
Power-down						X	X$^{(4)}$			X$^{(1)}$	X$^{(2)}$	X$^{(3)}$	

Notes: 1. Start frame detection only.

2. Start condition only.

3. Address match interrupt only.

4. For INT0 level interrupt only.

To enter a sleep mode, the SE bit in MCUCR must be set and a SLEEP instruction must be executed. The SMn bits in MCUCR select what sleep mode is 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 sleep mode, the MCU wakes up. The MCU is then stopped for four cycles in addition to the start-up time, executes the interrupt routine and resumes execution from the instruction following SLEEP. The contents of the register file and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the reset vector.

Note that if a level triggered interrupt is used for wake-up, the changed level must be held for some time to wake up the MCU (and for the MCU to enter the interrupt service routine). For more information, see Section 10.2 “External Interrupts” on page 48.

8.1.1 Idle Mode

This sleep mode basically stops clk_{CPU} and clk_{FLASH} while allowing other clocks to run. In idle mode the CPU is stopped but the following peripherals continue to operate:

- Watchdog and interrupt system
- Analog comparator and ADC
- USART, TWI, and Timer/Counters

Idle mode allows the MCU to wake up from external triggered interrupts as well as internal ones, such as 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 ACSRA (see Section 19.2.1 “ACSRA – Analog Comparator Control and Status Register” on page 170). This reduces power consumption in idle mode.

If the ADC is enabled, a conversion automatically starts when this mode is entered.

8.1.2 ADC Noise Reduction Mode

This sleep mode stops $\text{clk}_{\text{I/O}}$, clk_{CPU} and $\text{clk}_{\text{FLASH}}$ while allowing other clocks to run. In ADC noise reduction mode the CPU is stopped but the following peripherals continue to operate:

- Watchdog (if enabled) and external interrupts
- ADC
- USART start frame detector and TWI

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

The following events can wake up the MCU:

- Watchdog reset, external reset, and brown-out reset
- External level interrupt on INT0 and pin change interrupt
- ADC conversion complete interrupt and SPM/EEPROM ready interrupt
- USI start condition, USART start frame detection, and TWI address match

8.1.3 Power-Down Mode

This sleep mode stops all generated clocks, allowing operation of asynchronous modules only. In power-down mode the oscillator is stopped while the following peripherals continue to operate:

- Watchdog (if enabled), external interrupts

The following events can wake up the MCU:

- Watchdog reset, external reset, and brown-out reset
- External level interrupt on INT0 and pin change interrupt
- USI start condition, USART start frame detection, and TWI address match

8.1.4 Standby Mode

Standby 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.

8.2 Power Reduction Register

The power reduction register (PRR) (see Section 8.4.2 “PRR – Power Reduction Register” on page 37) provides a method for reducing power consumption by stopping the clock to individual peripherals. When the clock for a peripheral is stopped:

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

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

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

8.3 Minimizing Power Consumption

There are several issues to consider when trying to minimize 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 running. 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.

8.3.1 Analog to Digital Converter

If enabled, the ADC is 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 is an extended conversion. For more information on ADC operation, see Section 20. “Analog to Digital Converter” on page 172.

8.3.2 Analog Comparator

When entering idle mode, the analog comparator should be disabled if it is not being used. When entering ADC noise reduction mode, the analog comparator should be disabled. The analog comparator is automatically disabled in the other sleep modes. 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 is enabled, regardless of the sleep mode. For details on how to configure the analog comparator, see Section 19. “Analog Comparator” on page 169.

8.3.3 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 BODPD fuses, it is enabled in all sleep modes and thus always consumes power. In the deeper sleep modes this significantly impacts overall power consumption. If the brown-out detector is needed in the application, this module can also be set to sampled BOD mode to save power. For more information on how to configure the brown-out detector, see Section 9.2.4 “Brown-out Detection” on page 41.

8.3.4 Internal Voltage Reference

The internal voltage reference is enabled when it is needed by brown-out detection, the analog comparator, or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference is disabled and does not consume 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. For more information on the start-up time, see the internal band-gap reference in Table 25-5 on page 218.

8.3.5 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 is enabled in all sleep modes and thus always consumes power. In the deeper sleep modes this impacts overall power consumption. For more information on how to configure the watchdog timer, see “Watchdog Timer” on page 43.

8.3.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is to then 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 are 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 will then be enabled. For more information on what pins are enabled, see Section 11.2.5 “Digital Input Enable and Sleep Modes” on page 57. 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 uses excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the digital input disable register (DIDR0). For more information, see Section 20.13.5 “DIDR0 – Digital Input Disable Register 0” on page 186.

8.3.7 On-chip Debug System

If the on-chip debug system is enabled by the DWEN fuse and the chip enters sleep mode, the main clock source is enabled and thus always consumes power. In the deeper sleep modes this significantly impacts overall power consumption.

8.4 Register Description

8.4.1 MCUCR – MCU Control Register

The MCU control register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	–	SM1	SM0	SE	–	–	ISC01	ISC00	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 3:2 – Res: Reserved Bits**

These bits are reserved and always read as “0”.

- **Bits 6:5 – SM[1:0]: Sleep Mode Select Bits 1 and 0**

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

Table 8-2. Sleep Mode Select

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

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

- **Bit 4 – 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 keep the MCU from entering sleep mode unless it is the programmer’s purpose, Atmel recommends writing the sleep enable (SE) bit to “1” just before the execution of the SLEEP instruction and to clear it immediately after wake-up.

8.4.2 PRR – Power Reduction Register

The power reduction register provides a method to reduce power consumption by allowing peripheral clock signals to be disabled.

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	PRTWI	PRTIM1	PRTIM0	PRUSI	PRUSART1	PRUSART0	PRADC	PRR
Read/Write	R	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**

This bit is a reserved bit and always reads as “0”.

- **Bit 6 – PRTWI: Power Reduction Two-wire Interface**

Writing a logic one to this bit shuts down the two-wire interface module.

- **Bit 5 – PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When Timer/Counter1 is enabled, operation continues the same way as before the shutdown.

- **Bit 4 – PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When Timer/Counter0 is enabled, operation continues the same way as before the shutdown.

- **Bit 3 – 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 reinitialized to ensure proper operation.

- **Bit 2 – PRUSART1: Power Reduction USART1**

Writing a logic one to this bit shuts down the USART1 module. When the USART1 is enabled, operation continues the same way as before the shutdown.

- **Bit 1 – PRUSART0: Power Reduction USART0**

Writing a logic one to this bit shuts down the USART0 module. When the USART0 is enabled, operation continues the same way as before the shutdown.

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

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

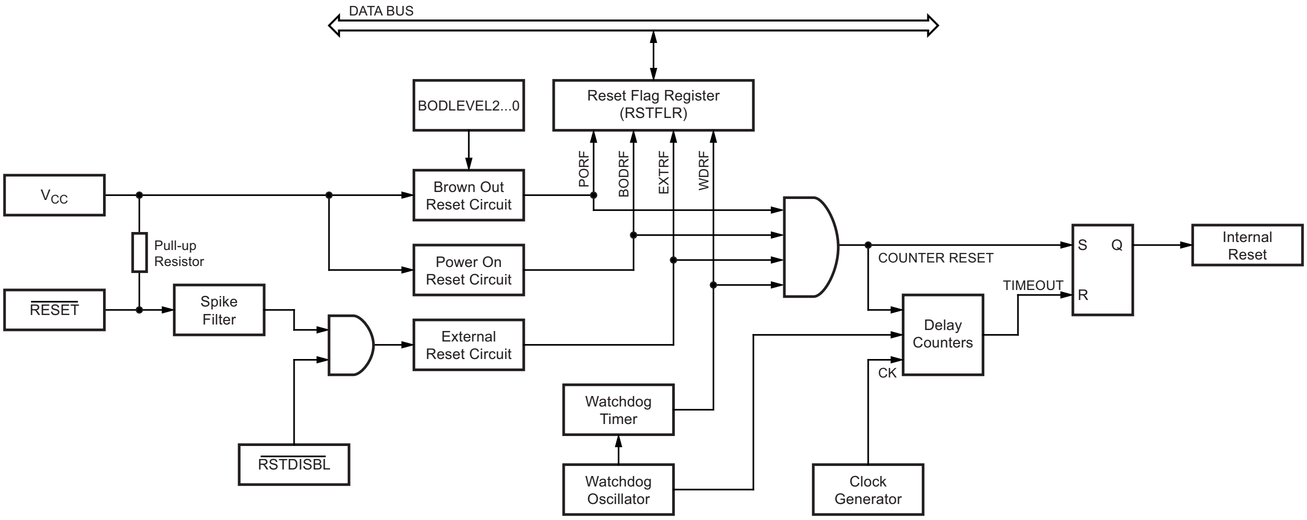
9. System Control and Reset

9.1 Resetting the AVR

During reset, all I/O registers are set to their initial values and the program starts execution from the reset vector. The instruction placed at the reset vector should be a JMP (two-word, direct jump) instruction to the reset handling routine, although other one- or two-word jump instructions can be used. If the program never enables an interrupt source, the interrupt vectors are not used and regular program code can be placed at these locations.

The circuit diagram in Figure 9-1 shows the reset logic. Electrical parameters of the reset circuitry are defined in Section 25.5 "System and Reset" on page 218.

Figure 9-1. Reset Logic



The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, extending the internal reset period. This allows the power to reach a stable level before normal operation starts.

9.2 Reset Sources

The Atmel® ATtiny1634 has four sources of reset:

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

9.2.1 Power-on Reset

A power-on reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in Section 25.5 "System and Reset" on page 218. 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 supply voltage failure.

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} falls below the detection level.

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

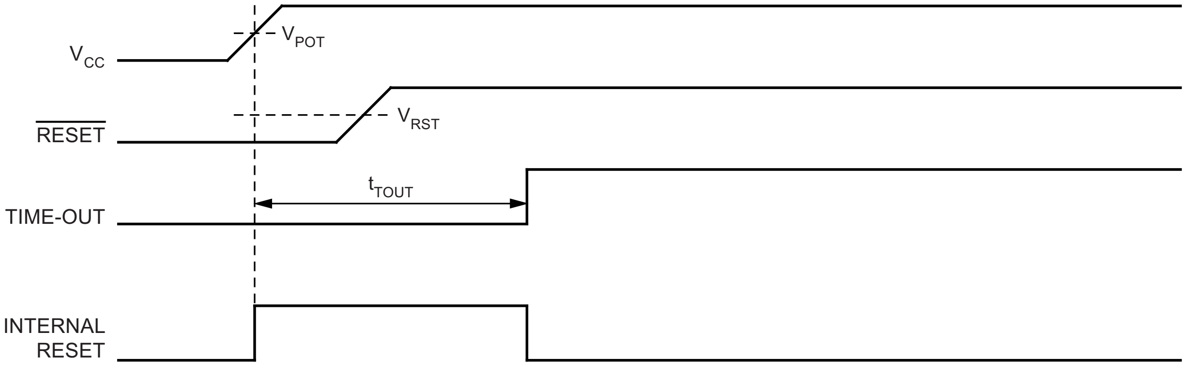
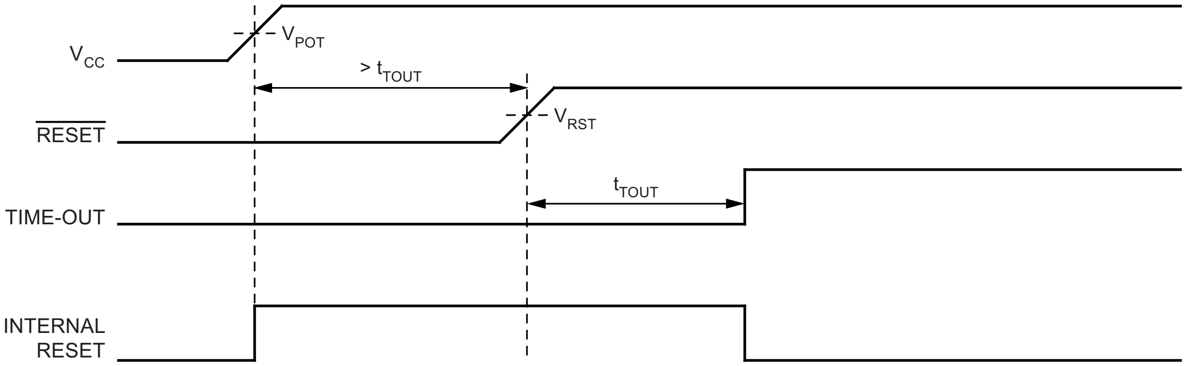


Figure 9-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally

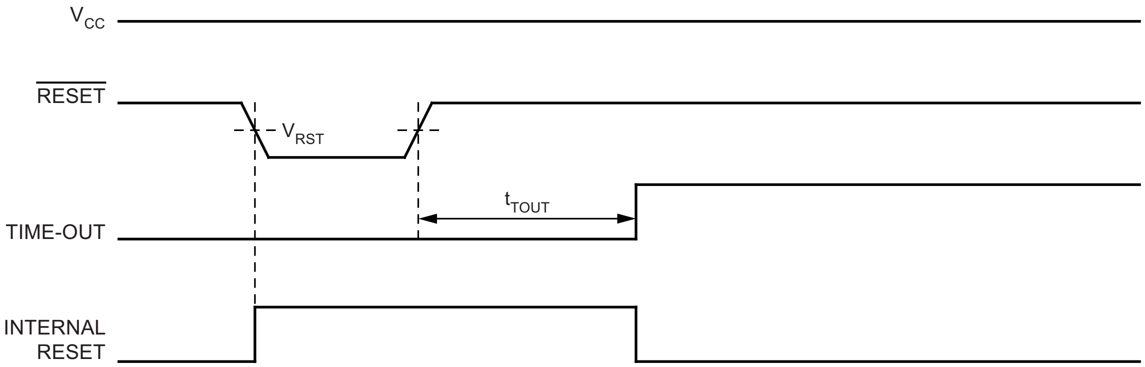


9.2.2 External Reset

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 25.5 "System and Reset" on page 218) 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.

External reset is ignored during the power-on start-up count. After the reset at power-on, the internal reset is delayed only if the RESET pin is low when the initial power-on delay count is completed. (see Figure 9-2 and Figure 9-3).

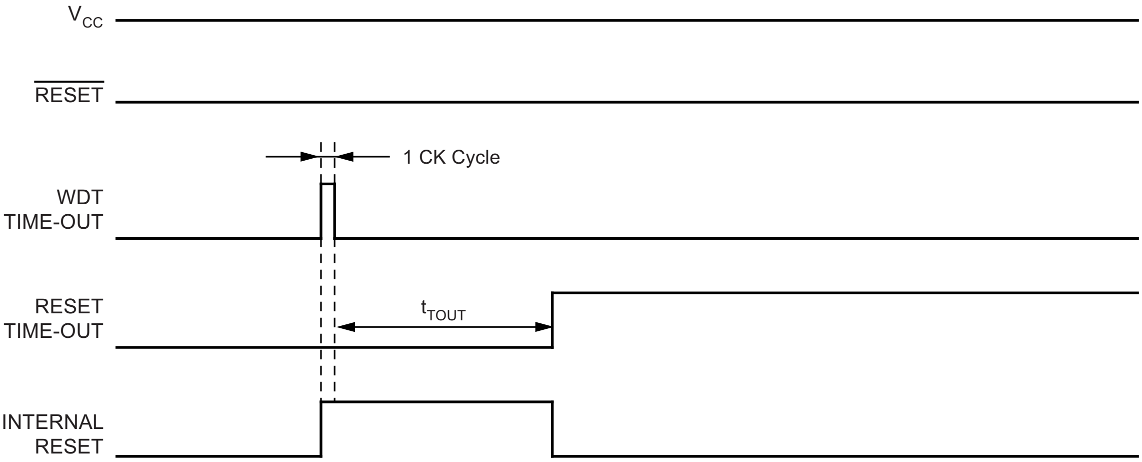
Figure 9-4. External Reset During Operation



9.2.3 Watchdog Reset

When the watchdog times out, it generates a short reset pulse. The delay timer starts counting the time-out period t_{TOUT} on the falling edge of this pulse. See Section 9.4 "Watchdog Timer" on page 43 for more information about operation of the watchdog timer and Table 25-5 on page 218 for more information about the reset time-out.

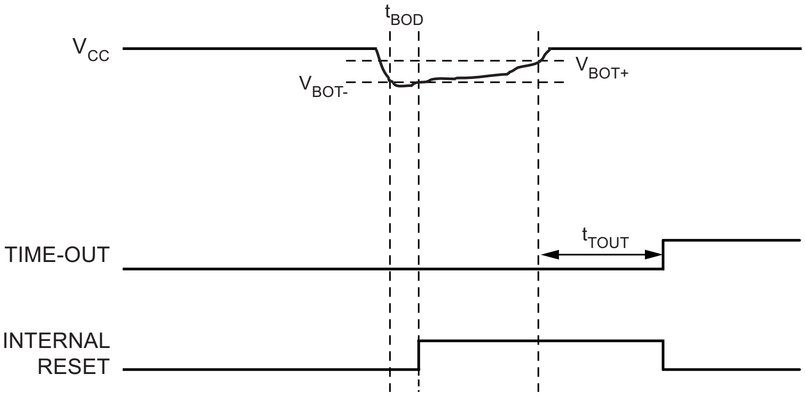
Figure 9-5. Watchdog Reset During Operation



9.2.4 Brown-out Detection

The brown-out detection (BOD) circuit monitors that the V_{CC} level is kept above a configurable trigger level V_{BOT}. When the BOD is enabled, a BOD reset occurs when V_{CC} falls and remains below the trigger level for the length of the detection time t_{BOD}. The reset is kept activated until V_{CC} rises above the trigger level again.

Figure 9-6. Brown-out Detection Reset



The BOD circuit does not detect a drop in V_{CC} unless the voltage stays below the trigger level for the detection time t_{BOD} (see Section 25.5 “System and Reset” on page 218).

The BOD circuit has three operating modes:

- **Disabled:** In this mode V_{CC} is not monitored. It is thus recommended only for applications where the power supply remains stable.
- **Enabled:** In this mode the V_{CC} level is continuously monitored. If V_{CC} drops below V_{BOT} for at least t_{BOD}, a brown-out reset is generated.
- **Sampled:** In this mode the V_{CC} level is sampled on each negative edge of a 1kHz clock that has been derived from the 32kHz ULP oscillator. The BOD is turned off between each sample. Compared to the mode where BOD is constantly enabled, this operating mode reduces power consumption but fails to detect drops in V_{CC} between two positive edges of the 1kHz clock. When a brown-out is detected in this mode, the BOD circuit is set to enabled mode to ensure that the device is kept in reset until V_{CC} has risen above V_{BOT}. The BOD returns to sampled mode after reset has been released and the fuses have been read in.

The BOD operating mode is selected using BODACT and BODPD fuse bits. The BODACT fuse bits determine how the BOD operates in active and idle mode as shown in Table 9-1.

Table 9-1. Setting BOD Operating Mode in Active and Idle Modes

BODACT1	BODACT0	Operating Mode
0	0	Reserved
0	1	Sampled
1	0	Enabled
1	1	Disabled

The BODPD fuse bits determine the operating mode in all sleep modes except idle mode as shown in Table 9-2.

Table 9-2. Setting BOD Operating Mode in Sleep Modes Other Than Idle

BODPD1	BODPD0	Operating Mode
0	0	Reserved
0	1	Sampled
1	0	Enabled
1	1	Disabled

See Section 23.2 “Fuse Bits” on page 197.

9.3 Internal Voltage Reference

The Atmel® ATtiny1634 features an internal band-gap reference. This reference is used for brown-out detection and can be used as an input to the analog comparator or the ADC. The band-gap voltage varies with supply voltage and temperature.

9.3.1 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 25.5 "System and Reset" on page 218. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (see Section 9.2.4 "Brown-out Detection" on page 41)
2. When the internal reference is connected to the analog comparator (by setting the ACBG bit in ACSRA)
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.

9.4 Watchdog Timer

The watchdog timer is clocked from the internal 32kHz ultra-low-power oscillator (see Section 7.2.3 "Internal 32kHz Ultra-Low-power (ULP) Oscillator" on page 27). By controlling the watchdog timer prescaler, the watchdog reset interval can be adjusted as shown in Table 9-5 on page 46. 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 ATtiny1634 resets and executes from the reset vector. For more information on watchdog reset timing, see Table 9-5 on page 46.

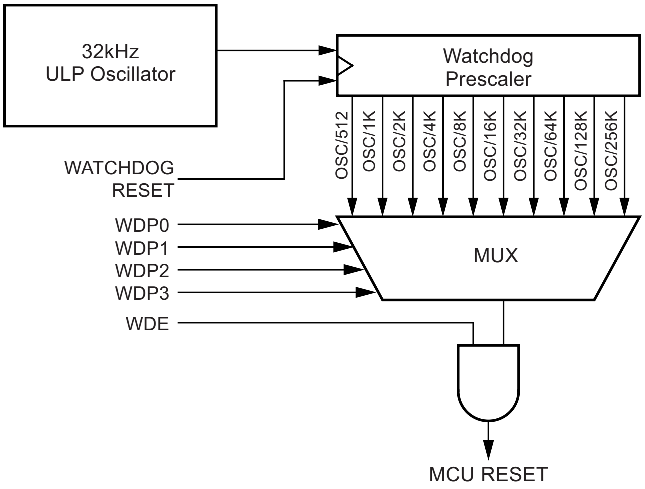
The watchdog timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the watchdog to wake up from power-down.

To prevent unintentional disabling of the watchdog or unintentionally changing the time-out period, two different safety levels are selected by the WDTON fuse as shown in Table 9-3. For more information, see Section 9.4.1 "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 44.

Table 9-3. 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



9.4.1 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.

- Safety level 1

  In this mode, the watchdog timer is initially disabled, but can be enabled by writing the WDE bit to "1" without any restriction. A timed sequence is needed when disabling an enabled watchdog timer. To disable an enabled watchdog timer, the following steps must be completed:

  a. Write the signature for change enable of protected I/O registers to register CCP.

  b. Within four instruction cycles, in the same operation, write WDE and WDP bits.

- Safety level 2

  In this mode, the watchdog timer is always enabled and the WDE bit always reads as "1". A timed sequence is needed when changing the watchdog time-out period. To change the watchdog time-out, the following steps must be completed:

  a. Write the signature for change enable of protected I/O registers to register CCP.

  b. Within four instruction cycles, write the WDP bit. The value written to WDE is irrelevant.

9.4.2 Code Examples

The following code example shows how to turn off the WDT. The example assumes that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts occur during execution of these functions.

Assembly Code Example

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

Note: See Section 4.2 "Code Examples" on page 7.

9.5 Register Description

9.5.1 MCUSR – MCU Status Register

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

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	–	–	–	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® ATtiny1634 and always read as “0”.

- **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 checking the reset flags.

9.5.2 WDTCSR – Watchdog Timer Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	WDIF	WDIE	WDP3	–	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

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

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

- **Bit 6 – WDIE: Watchdog Time-out Interrupt Enable**

When this bit is written to “1”, WDE is cleared, the I bit in the status register is set and 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 9-4. 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 – Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny1634 and always reads as "0".

- **Bit 3 – WDE: Watchdog Enable**

This bit enables and disables the watchdog timer (see Section 9.4.1 "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 44).

- **Bits 5, 2:0 – WDP[3:0]: Watchdog Timer Prescaler 3 - 0**

The WDP[3:0] 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 9-5.

Table 9-5. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	WDT Oscillator Cycles	Typical Time-out @V_{CC} = 5V
0	0	0	0	512 cycles	16ms
0	0	0	1	1K cycles	32ms
0	0	1	0	2K cycles	64ms
0	0	1	1	4K cycles	0.125s
0	1	0	0	8K cycles	0.25s
0	1	0	1	16K cycles	0.5s
0	1	1	0	32K cycles	1.0s
0	1	1	1	64K cycles	2.0s
1	0	0	0	128K cycles	4.0s
1	0	0	1	256K cycles	8.0s
1	0	1	0	Reserved[1]	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

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

10. Interrupts

This section provides specific information about how the Atmel® ATtiny1634 handles interrupts. For a general explanation of the AVR® interrupt handling, see Section 5.7 "Reset and Interrupt Handling" on page 12.

10.1 Interrupt Vectors

The interrupt vectors of the Atmel ATtiny1634 are described in Table 10-1.

Table 10-1. Reset and Interrupt Vectors

Vector No.	Program Address	Label	Interrupt Source
1	0x0000	RESET	External pin, power-on reset, brown-out reset, watchdog reset
2	0x0002	INT0	External interrupt request 0
3	0x0004	PCINT0	Pin change interrupt request 0
4	0x0006	PCINT1	Pin change interrupt request 1
5	0x0008	PCINT2	Pin change interrupt request 2
6	0x000A	WDT	Watchdog time-out
7	0x000C	TIM1_CAPT	Timer/Counter1 input capture
8	0x000E	TIM1_COMPA	Timer/Counter1 compare match A
9	0x0010	TIM1_COMPB	Timer/Counter1 compare match B
10	0x0012	TIM1_OVF	Timer/Counter1 overflow
11	0x0014	TIM0_COMPA	Timer/Counter0 compare match A
12	0x0016	TIM0_COMPB	Timer/Counter0 compare match B
13	0x0018	TIM0_OVF	Timer/Counter0 overflow
14	0x001A	ANA_COMP	Analog comparator
15	0x001C	ADC_READY	ADC conversion complete
16	0x001E	USART0_RXS	USART0 Rx start
17	0x0020	USART0_RXC	USART0 Rx complete
18	0x0022	USART0_DRE	USART0 data register empty
19	0x0024	USART0_TXC	USART0 Tx complete
20	0x0026	USART1_RXS	USART1 Rx start
21	0x0028	USART1_RXC	USART1 Rx complete
22	0x002A	USART1_DRE	USART1 data register empty
23	0x002C	USART1_TXC	USART1 Tx complete
24	0x002E	USI_STR	USI START
25	0x0030	USI_OVF	USI overflow
26	0x0032	TWI	Two-wire interface
27	0x0034	EE_RDY	EEPROM ready
28	0x0036	QTRIP	QTRIP QTouch

In case the program never enables an interrupt source, the interrupt vectors are not used. As a result, regular program code can be placed at these locations.

A typical and general setup for interrupt vector addresses in the Atmel® ATtiny1634 is shown in the program example below.

Assembly Code Example

```
        .org 0x0000                             ;Set address of next
                                                statement

               jmp                 RESET         ; Address 0x0000
               jmp                 INT0_ISR      ; Address 0x0002
               jmp                 PCINT0_ISR    ; Address 0x0004
               jmp                 PCINT1_ISR    ; Address 0x0006
               jmp                 PCINT2_ISR    ; Address 0x0008
               jmp                 WDT_ISR       ; Address 0x000A
               jmp                 TIM1_CAPT_ISR ; Address 0x000C
               jmp                 TIM1_COMPA_ISR; Address 0x000E
               jmp                 TIM1_COMPB_ISR; Address 0x0010
               jmp                 TIM1_OVF_ISR  ; Address 0x0012
               jmp                 TIM0_COMPA_ISR; Address 0x0014
               jmp                 TIM0_COMPB_ISR; Address 0x0016
               jmp                 TIM0_OVF_ISR  ; Address 0x0018
               jmp                 ANA_COMP_ISR  ; Address 0x001A
               jmp                 ADC_ISR       ; Address 0x001C
               jmp                 USART0_RXS_ISR; Address 0x001E
               jmp                 USART0_RXC_ISR; Address 0x0020
               jmp                 USART0_DRE_ISR; Address 0x0022
               jmp                 USART0_TXC_ISR; Address 0x0024
               jmp                 USART1_RXS_ISR; Address 0x0026
               jmp                 USART1_RXC_ISR; Address 0x0028
               jmp                 USART1_DRE_ISR; Address 0x002A
               jmp                 USART1_TXC_ISR; Address 0x002C
               jmp                 USI_START_ISR ; Address 0x002E
               jmp                 USI_OVF_ISR   ; Address 0x0030
               jmp                 TWI_ISR       ; Address 0x0032
               jmp                 EE_RDY_ISR    ; Address 0x0034
               jmp                 QTRIP_ISR     ; Address 0x0036

        RESET:                                   ; Main program start
               <instr>                           ; Address 0x0038
               ...
```

Note: See Section 4.2 “Code Examples” on page 7.

10.2 External Interrupts

External interrupts are triggered by the INT0 pin or by any of the PCINTn pins. Note that, if enabled, the interrupts trigger even if the INTn or PCINTn pins are configured as outputs. This feature makes it possible to generate software interrupts.

The pin change interrupts trigger as follows:

- Pin change interrupt 0 (PCI0): triggers if any enabled PCINT[7:0] pin toggles.
- Pin change interrupt 1 (PCI1): triggers if any enabled PCINT[11:8] pin toggles.
- Pin change interrupt 2 (PCI2): triggers if any enabled PCINT[17:12] pin toggles.

Registers PCMSK0, PCMSK1, and PCMSK2 control what pins contribute to the pin change interrupts.

Pin change interrupts on PCINT[17:0] are detected asynchronously, meaning these interrupts can also be used for waking the part from sleep modes other than idle mode.

External interrupt INT0 can be triggered by a falling or rising edge, or a low level. For more information, see Section 8.4.1 "MCUCR – MCU Control Register" on page 37. When INT0 is enabled and configured as level-triggered, the interrupt triggers as long as the pin is held low.

Note that detection of falling or rising edge interrupts on INT0 requires the presence of an I/O clock as described in Section 7. "Clock System" on page 25.

10.2.1 Low Level Interrupt

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

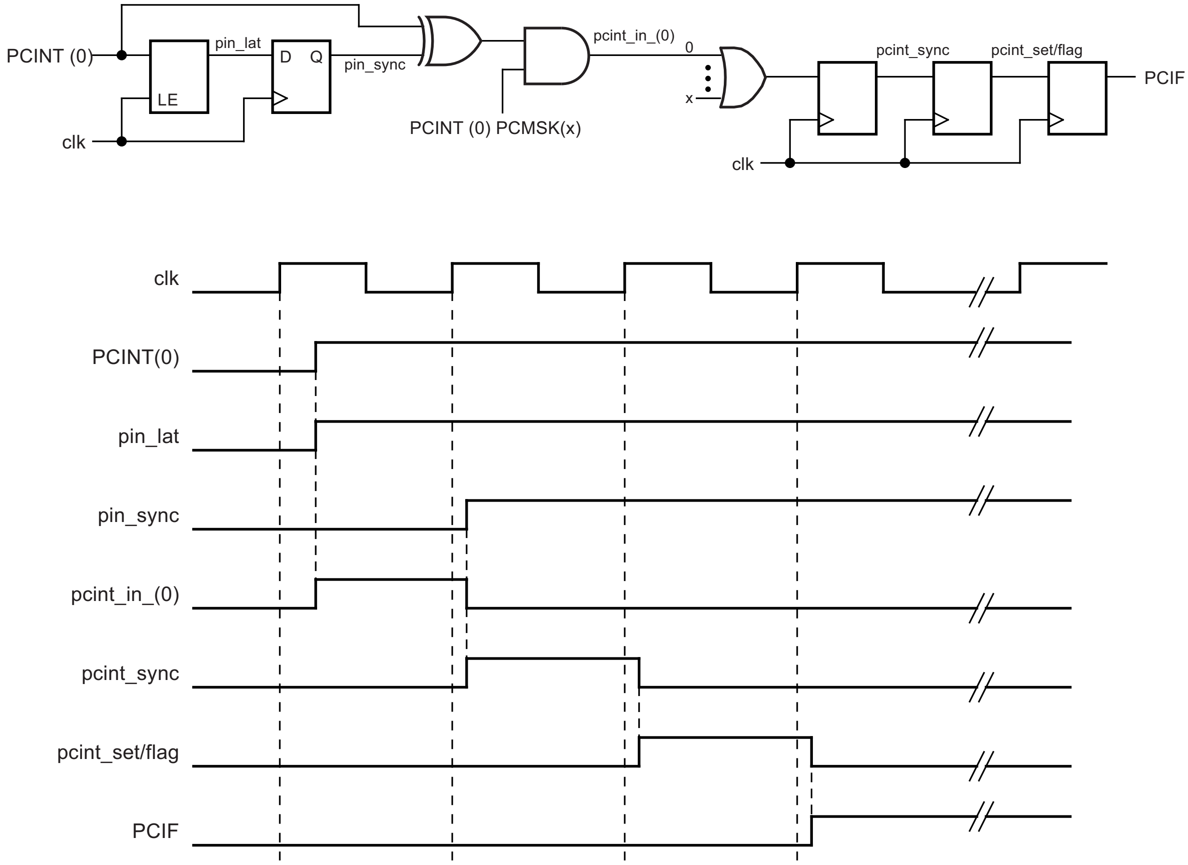
Note that if a level-triggered interrupt is used for wake-up from power-down, the required level must be held long enough for the MCU to complete the wake-up and trigger the level interrupt. If the level disappears before the end of the start-up time, the MCU still wakes up but no interrupt is generated. The start-up time is defined by the SUT and CKSEL fuses as described in Section 7. "Clock System" on page 25.

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

10.2.2 Pin Change Interrupt Timing

A timing example of a pin change interrupt is shown in Figure 10-1.

Figure 10-1. Timing of Pin Change Interrupts



10.3 Register Description

10.3.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	–	SM1	SM0	SE	–	–	ISC01	ISC00	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1:0 – ISC0[1:0]: Interrupt Sense Control 0 Bit 1 and Bit 0**

External interrupt 0 is triggered by activity on pin INT0, provided that the SREG I flag and the corresponding interrupt mask are set. The conditions required to trigger the interrupt are defined in Table 10-2.

Table 10-2. External Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request[(1)].
0	1	Any logical change on INT0 generates an interrupt request[(2)].
1	0	The falling edge of INT0 generates an interrupt request[(2)].
1	1	The rising edge of INT0 generates an interrupt request[(2)].

Note: 1. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

2. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt.

10.3.2 GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3C (0x5C)	–	INT0	PCIE2	PCIE1	PCIE0	–	–	–	GIMSK
Read/Write	R	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

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

These bits are reserved and always read as “0”.

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

The external interrupt for the INT0 pin is enabled when this bit and the I bit in the status register (SREG) are set. The trigger conditions are set with the ISC0n bits.

Activity on the pin causes an interrupt request even if INT0 has been configured as an output.

- **Bit 5 – PCIE2: Pin Change Interrupt Enable 2**

When this bit and the I bit of SREG are set, the pin change interrupt 2 is enabled. Any change on an enabled PCINT[17:12] pin causes a PCINT2 interrupt (see Table 10-1 on page 47).

Each pin can be individually enabled (see Section 10.3.4 “PCMSK2 – Pin Change Mask Register 2” on page 51).

- **Bit 4 – PCIE1: Pin Change Interrupt Enable 1**

When this bit and the I bit of SREG are set, the pin change interrupt 1 is enabled. Any change on an enabled PCINT[11:8] pin causes a PCINT1 interrupt (see Table 10-1 on page 47).

Each pin can be individually enabled (see Section 10.3.5 “PCMSK1 – Pin Change Mask Register 1” on page 52).

- **Bit 3 – PCIE0: Pin Change Interrupt Enable 0**

When this bit and the I bit of SREG are set, the pin change interrupt 0 is enabled. Any change on an enabled PCINT[7:0] pin causes a PCINT0 interrupt (see Table 10-1 on page 47).

Each pin can be individually enabled (see Section 10.3.6 "PCMSK0 – Pin Change Mask Register 0" on page 52).

10.3.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	–	INTF0	PCIF2	PCIF1	PCIF0	–	–	–	GIFR
Read/Write	R	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

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

These bits are reserved and always read as "0".

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

This bit is set when activity on INT0 has triggered an interrupt request. Provided that the I bit in SREG and the INT0 bit in GIMSK are set, the MCU jumps to the corresponding interrupt vector.

The flag is cleared when the interrupt service routine is executed. Alternatively, the flag can be cleared by writing a logic one to it.

This flag is always cleared when INT0 is configured as a level interrupt.

- **Bit 5 – PCIF2: Pin Change Interrupt Flag 2**

This bit is set when a logic change on any PCINT[17:12] pin has triggered an interrupt request. Provided that the I bit in SREG and the PCIE2 bit in GIMSK are set, the MCU jumps to the corresponding interrupt vector.

The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logic one to it.

- **Bit 4 – PCIF1: Pin Change Interrupt Flag 1**

This bit is set when a logic change on any PCINT[11:8] pin has triggered an interrupt request. Provided that the I bit in SREG and the PCIE1 bit in GIMSK are set, the MCU jumps to the corresponding interrupt vector.

The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logic one to it.

- **Bit 3 – PCIF0: Pin Change Interrupt Flag 0**

This bit is set when a logic change on any PCINT[7:0] pin has triggered an interrupt request. Provided that the I bit in SREG and the PCIE0 bit in GIMSK are set, the MCU jumps to the corresponding interrupt vector.

The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logic one to it.

10.3.4 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
0x29 (0x49)	–	–	PCINT17	PCINT16	PCINT15	PCINT14	PCINT13	PCINT12	PCMSK2
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 and always read as "0".

- **Bits 5:0 – PCINT[17:12]: Pin Change Enable Mask 17:12**

Each PCINTn bit selects if the pin change interrupt of the corresponding I/O pin is enabled. Pin change interrupt on a pin is enabled by setting the mask bit for the pin (PCINTn) and the corresponding group bit (PCIEn) in GIMSK.

When this bit is cleared, the pin change interrupt on the corresponding pin is disabled.

10.3.5 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	–	–	–	–	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved and always read as “0”.

- **Bits 3:0 – PCINT[11:8]: Pin Change Enable Mask 11:8**

Each PCINTn bit selects if the pin change interrupt of the corresponding I/O pin is enabled. Pin change interrupt on a pin is enabled by setting the mask bit for the pin (PCINTn) and the corresponding group bit (PCIEn) in GIMSK.

When this bit is cleared, the pin change interrupt on the corresponding pin is disabled.

10.3.6 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – PCINT[7:0]: Pin Change Enable Mask 7:0**

Each PCINTn bit selects if the pin change interrupt of the corresponding I/O pin is enabled. Pin change interrupt on a pin is enabled by setting the mask bit for the pin (PCINTn) and the corresponding group bit (PCIEn) in GIMSK.

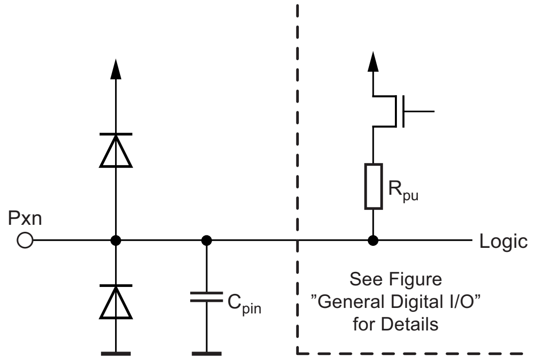
When this bit is cleared, the pin change interrupt on the corresponding pin is disabled.

11. I/O Ports

11.1 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 pull-up resistors (if configured as input). Most output buffers have symmetrical drive characteristics with both high-sink and source capability, while some are asymmetrical and have high-sink and standard 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 11-1 on page 53. For a complete list of parameters, see Section 25. "Electrical Characteristics" on page 215.

Figure 11-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lowercase "x" represents the numbering letter for the port and a lowercase "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 Table 11-1 on page 55.

Four I/O memory address locations are allocated for each port, one each for the data register – PORTx, data direction register – DDRx, pull-up enable register – PUEx, and the port input pins – PINx. The port input pins I/O location is read-only, while the data register, the data direction register, and the pull-up enable register are read/write. However, writing a logic one to a bit in the PINx register results in a toggle in the corresponding bit in the data register.

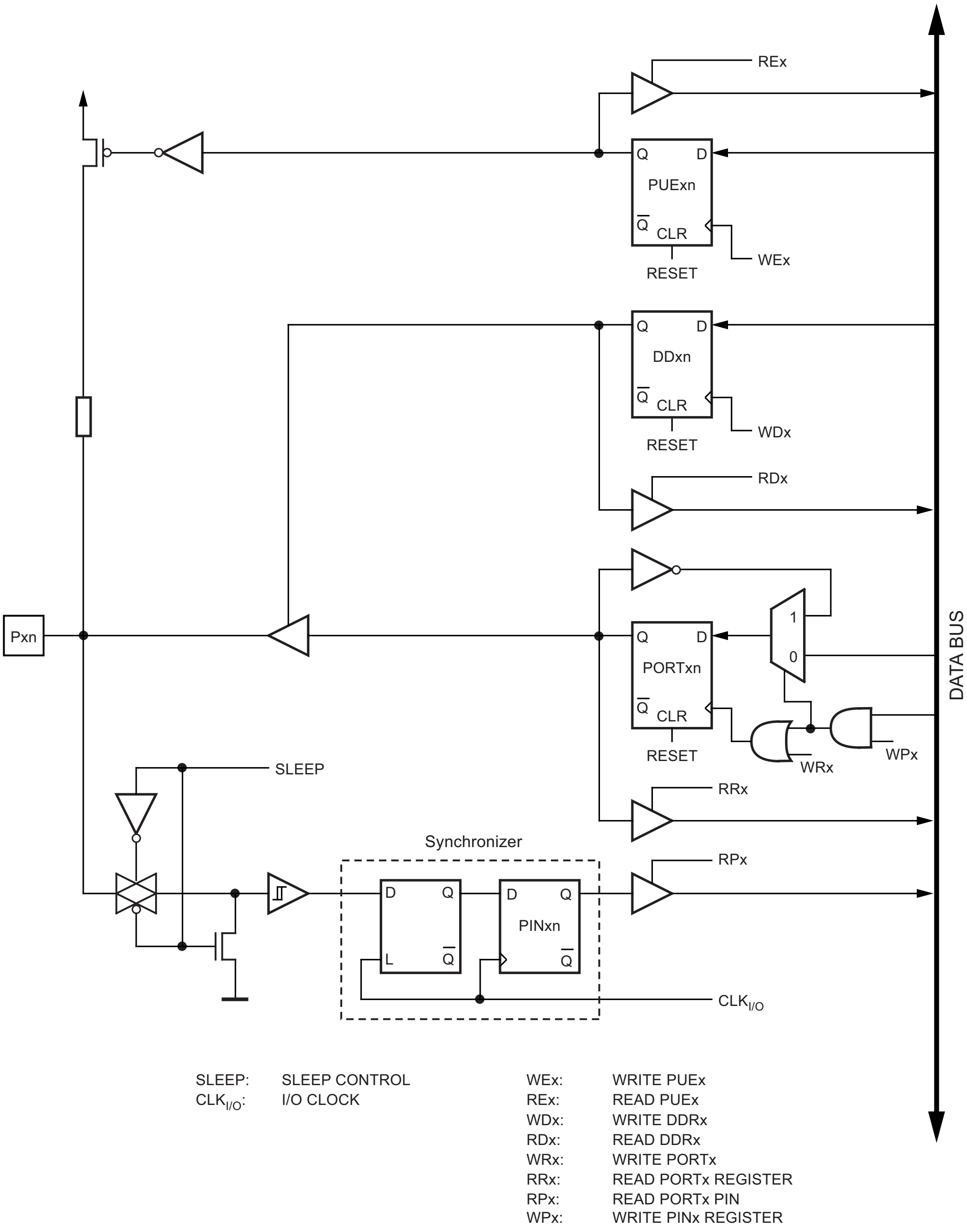
Using the I/O port as a general digital I/O is described in Section 11.2 "Ports as a General Digital I/O" on page 54. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 11.3 "Alternate Port Functions" on page 59. See 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 a general digital I/O.

11.2 Ports as a General Digital I/O

The ports are bidirectional I/O ports with optional internal pull-ups. Figure 11-2 shows a functional description of one I/O port pin, here generically called Pxn.

Figure 11-2. General Digital I/O[1]



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

11.2.1 Configuring the Pin

Each port pin consists of four register bits: DDxn, PORTxn, PUExn, and PINxn. As shown in Section 11.4 "Register Description" on page 69, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, the PUExn bits at the PUEx I/O address, and the PINxn bits at the PINx I/O address.

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

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

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

Table 11-1 summarizes the control signals for the pin value.

Table 11-1. Port Pin Configurations

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

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

11.2.2 Toggling the Pin

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

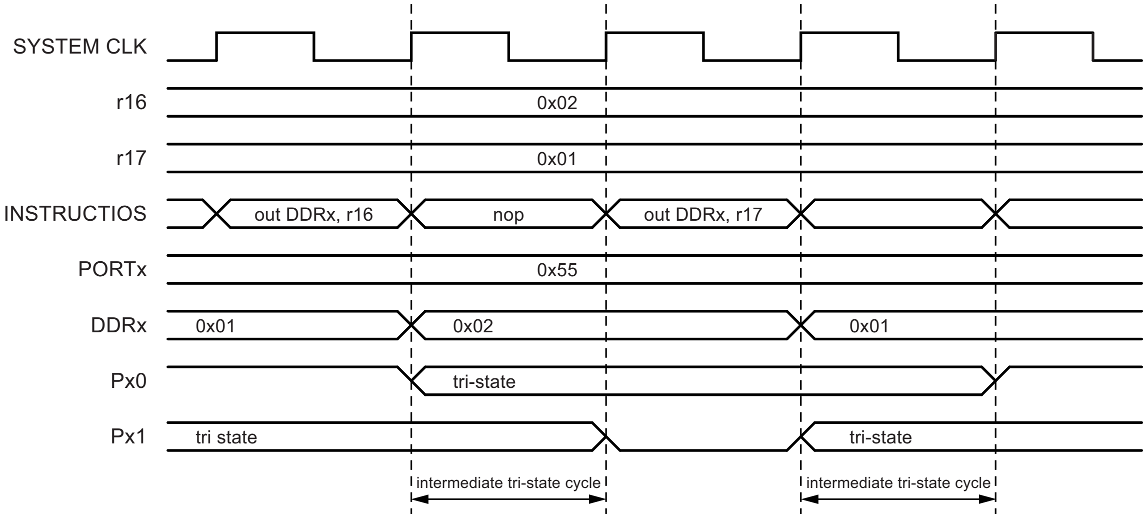
11.2.3 Break-Before-Make Switching

In break-before-make mode, switching the DDRxn bit from input to output introduces an immediate tri-state period lasting one system clock cycle, as indicated in Figure 11-3 on page 56. For example, if the system clock is 4MHz and the DDRxn is written to make an output, an immediate tri-state period of 250ns is introduced before the value of PORTxn is seen on the port pin.

To avoid glitches it is recommended that the maximum DDRxn toggle frequency is two system clock cycles. The break-before-make mode applies to the entire port and it is activated by the BBMx bit. For more details, see Section 11.4.1 "PORTCR – Port Control Register" on page 69.

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

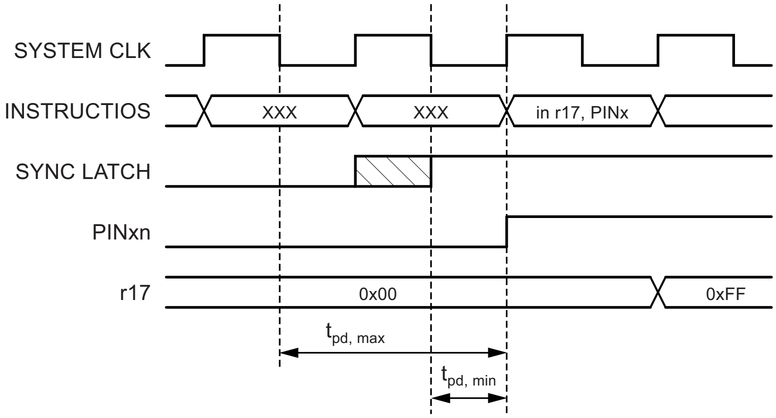
Figure 11-3. Switching Between Input and Output in Break-Before-Make Mode



11.2.4 Reading the Pin Value

Regardless of the setting of the data direction bit DDxn, the port pin can be read through the PINxn register bit. As shown in Figure 11-2 on page 54, 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; however, this also creates a delay. Figure 11-4 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$, respectively.

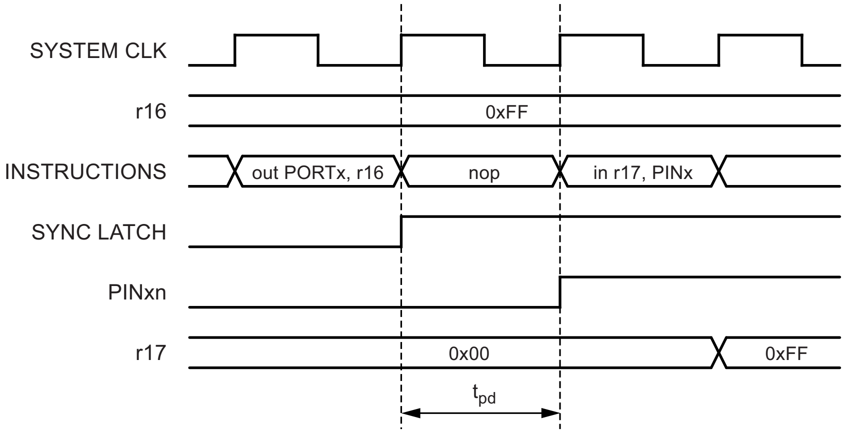
Figure 11-4. Synchronization when Reading an Externally Applied Pin Value



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

When reading back a software-assigned pin value, a NOP (no operation) instruction must be inserted as indicated in Figure 11-5 on page 57. The OUT instruction sets the SYNC LATCH signal at the positive edge of the clock. In this case, the propagation delay (tPD) through the synchronizer is one system clock period.

Figure 11-5. Synchronization when Reading a Software-assigned Pin Value



11.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 11-2 on page 54, the digital input signal can be clamped to ground at the input of the Schmitt trigger. The signal denoted SLEEP in the figure is set by the MCU sleep controller in power-down and standby modes to avoid high power consumption if some input signals are left floating or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 11.3 “Alternate Port Functions” on page 59.

If a logic high level (“1”) 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 is set when resuming from the above-mentioned sleep mode because the clamping in this sleep mode produces the requested logic change.

11.2.6 Unconnected Pins

If some pins are unused, Atmel recommends ensuring 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 for an unused pin is to enable the internal pull-up. In this case, the pull-up is disabled during reset. If low power consumption during reset is important, it is advisable to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended because this may cause excessive currents if the pin is accidentally configured as an output.

11.2.7 Program Examples

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

Assembly Code Example

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

Note: Two temporary registers are used to minimize the time from when 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.

C Code Example

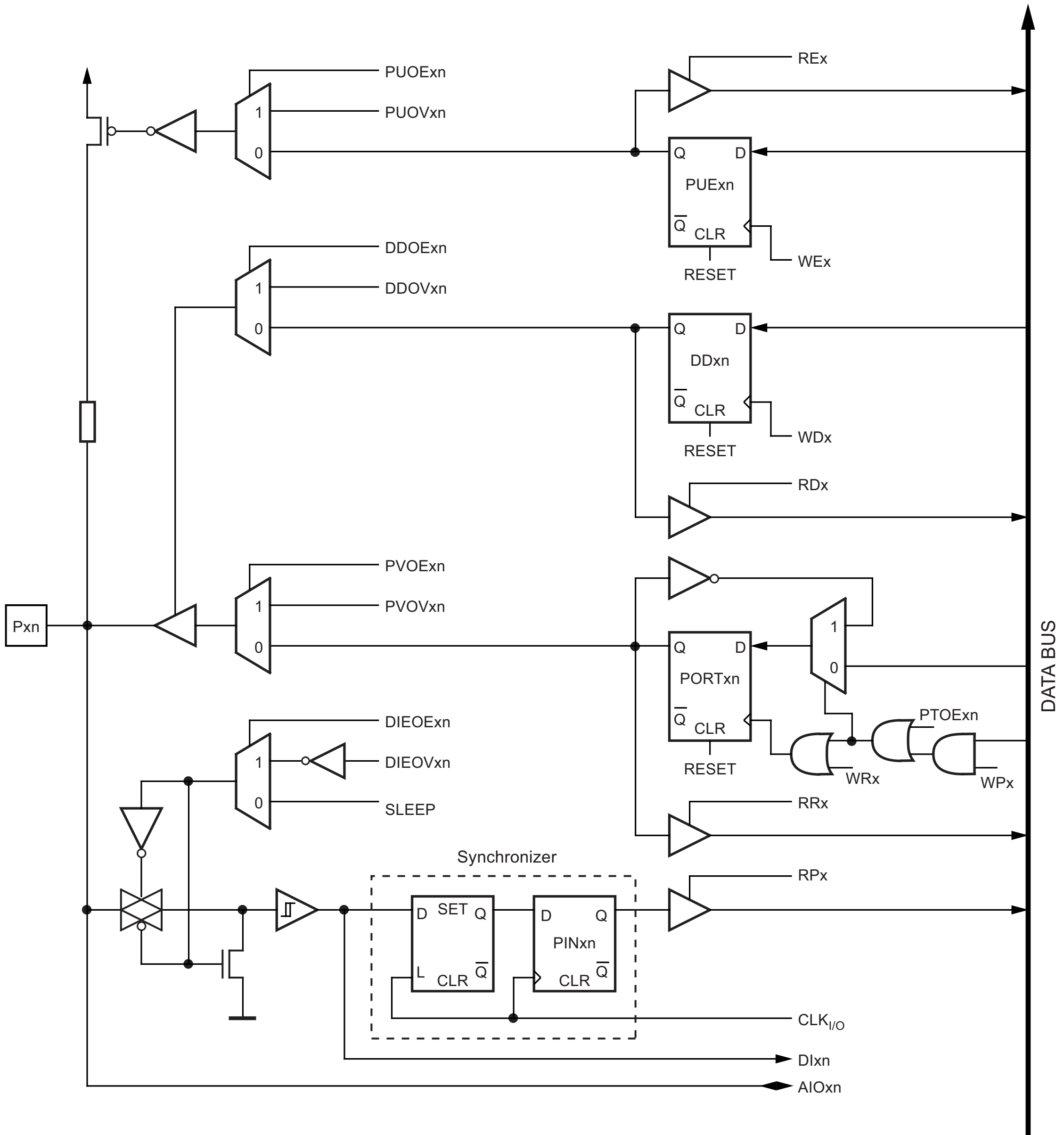
```
    unsigned char i;
        ...
        /* Define pull-ups and set outputs high */
        /* Define directions for port pins */
        PORTA = (1<<PA4)|(1<<PA1)|(1<<PA0);
        DDRA = (1<<DDA3)|(1<<DDA2)|(1<<DDA1)|(1<<DDA0);
        /* Insert nop for synchronization*/
        _NOP();
      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

Note: See Section 4.2 “Code Examples” on page 7.

11.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 11-6 below shows how the port pin control signals from the simplified Figure 11-2 on page 54 can be overridden by alternate functions.

Figure 11-6. Alternate Port Functions[1]



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

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

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

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

Table 11-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 11-6 are not shown in the subsequent tables. The overriding signals are generated internally in the modules having the alternate function.

Table 11-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up override enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when PUExn = 0b1.
PUOV	Pull-up override value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the PUExn register 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 uses 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 bidirectionally.

The following subsections briefly describe the alternate functions for each port and indicate the overriding signals to the alternate function. For more information, see the alternate function description.

11.3.1 Alternate Functions of Port A

The port A pins with alternate function are shown in Table 11-3.

Table 11-3. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA0	AREF: External analog reference PCINT0: Pin change interrupt 0, source 0
PA1	AIN0: Analog comparator, positive input PCINT1: Pin change interrupt 0, source 1
PA2	AIN1: Analog comparator, negative input PCINT2: Pin change interrupt 0, source 2
PA3	ADC0: ADC input channel 0 SNS: Sense line for capacitive measurement T1: Timer/Counter1 clock source PCINT3: Pin change interrupt 0, source 3
PA4	ADC1: ADC input channel 1 T0: Timer/Counter0 clock source PCINT4: Pin change interrupt 0, source 4
PA5	ADC2: ADC input channel 2 OC0B: Timer/Counter0 compare match B output PCINT5: Pin change interrupt 0, source 5
PA6	ADC3: ADC input channel 3 OC1B: Timer/Counter1 compare match B output PCINT6: Pin change interrupt 0, source 6
PA7	ADC4: ADC input channel 4 RXD0: UART0 data receiver PCINT7: Pin change interrupt 0, source 7

- **Port A, Bit 0 – AREF/PCINT0**
  - AREF: External analog reference for ADC. The pull-up and output driver are disabled on PA0 when the pin is used as an external reference or internal voltage reference with an external capacitor at the AREF pin.
  - PCINT0: Pin change interrupt source 0. The PA0 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 1 – AIN0/PCINT1**
  - AIN0: Analog comparator positive input. Configure the port pin as input with the internal pull-up switched off to keep the digital port function from interfering with the function of the analog comparator.
  - PCINT1: Pin change interrupt source 1. The PA1 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 2 – AIN1/PCINT2**
  - AIN1: Analog comparator negative input. Configure the port pin as input with the internal pull-up switched off to keep the digital port function from interfering with the function of the analog comparator.
  - PCINT2: Pin change interrupt source 2. The PA2 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 3 – ADC0/T1/PCINT3**
  - ADC0: Analog to digital converter, channel 0.
  - SNS: Sense line for capacitive measurement using QTouch technology. Connected to C_S.
  - T1: Timer/Counter1 counter source.
  - PCINT3: Pin change interrupt source 3. The PA3 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 4 – ADC1/T0/PCINT4**
  - ADC1: Analog to digital converter, channel 1.
  - T0: Timer/Counter0 counter source.
  - PCINT4: Pin change Interrupt source 4. The PA4 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 5 – ADC2/OC0B/PCINT5**
  - ADC2: Analog to digital converter, channel 2.
  - OC0B: Output compare match output. The PA5 pin can serve as an external output for the Timer/Counter0 compare match B. The PA5 pin has to be configured as an output (DDA5 set (“1”)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
  - PCINT5: Pin change interrupt source 5. The PA5 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 6 – ADC3/OC1B/PCINT6**
  - ADC3: Analog to digital converter, channel 3.
  - OC1B: Output compare match output. The PA6 pin can serve as an external output for the Timer/Counter1 compare match B. The pin has to be configured as an output (DDA6 set (“1”)) to serve this function. This is also the output pin for the PWM mode timer function.
  - PCINT6: Pin change interrupt source 6. The PA6 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 7 – ADC4/RXD0/PCINT7**
  - ADC4: Analog to digital converter, channel 4.
  - RXD0: UART0 data receiver.
  - PCINT7: Pin change interrupt source 7. The PA7 pin can serve as an external interrupt source for pin change interrupt 0.

Table 11-4 and Table 11-6 compares the alternate functions of port A to the overriding signals shown in Figure 11-6 on page 59.

Table 11-4. Overriding Signals for Alternate Functions in PA[7:5]

Signal Name	PA7/ADC4/RXD0/PCINT7	PA6/ADC3/OC1B/PCINT6	PA5/ADC2/OC0B/PCINT5
PUOE	RXD0_OE	0	0
PUOV	PUEA7	0	0
DDOE	RXD0_EN	0	0
DDOV	0	0	0
PVOE	0	OC1B enable	OC0B enable
PVOV	0	OC1B	OC0B
PTOE	0	0	0
DIEOE	(PCINT7 • PCIE0) + ADC4D	(PCINT6 • PCIE0) + ADC3D	(PCINT5 • PCIE) + ADC2D
DIEOV	PCINT7 • PCIE0	PCINT6 • PCIE0	PCINT5 • PCIE0
DI	RXD0/PCINT7 input	PCINT6 input	PCINT5 input
AIO	ADC4 input	ADC3 input	ADC2 input

Table 11-5. Overriding Signals for Alternate Functions in PA[4:2]

Signal Name	PA4/ADC1/T0/PCINT4	PA3/ADC0/SNS/T1/PCINT3	PA2/AIN1/PCINT2
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	0	0
DDOV	0	0	0
PVOE	0	0	0
PVOV	0	0	0
PTOE	0	0	0
DIEOE	(PCINT4 • PCIE0) + ADC1D	(PCINT3 • PCIE0) + ADC0D	(PCINT2 • PCI0) + AIN1D
DIEOV	PCINT4 • PCIE0	PCINT3 • PCIE0	PCINT2 • PCIE0
DI	T0/PCINT4 input	T1/PCINT3 input	PCINT2 input
AIO	ADC1 input	ADC0 or SNS input	Analog comparator negative input

Table 11-6. Overriding Signals for Alternate Functions in PA[1:0]

Signal Name	PA1/AIN0/PCINT1	PA0/AREF/PCINT0
PUOE	0	$\overline{\text{RESET}}$ • ($\overline{\text{REFS1}}$ • REFS0 + REFS1 • REFS0)
PUOV	0	0
DDOE	0	$\overline{\text{RESET}}$ • ($\overline{\text{REFS1}}$ • REFS0 + REFS1 • REFS0)
DDOV	0	0
PVOE	0	$\overline{\text{RESET}}$ • ($\overline{\text{REFS1}}$ • REFS0 + REFS1 • REFS0)
PVOV	0	0
PTOE	0	0
DIEOE	(PCINT1 • PCIE0) + AIN0D	(PCINT0 • PCIE0) + AREFD
DIEOV	PCINT1 • PCIE0	PCINT0 • PCIE0
DI	PCINT1 input	PCINT0 input
AIO	Analog comparator positive input	Analog reference

11.3.2 Alternate Functions of Port B

The port B pins with alternate function are shown in Table 11-7.

Table 11-7. Port B Pins Alternate Functions

Port Pin	Alternate Function	
PB0	ADC5: TXD0: PCINT8:	ADC input channel 5 UART0 data transmitter Pin change interrupt 1, source 8
PB1	ADC6: RXD1: DI: SDA: PCINT9:	ADC input channel 6 UART1 data receiver USI data input (three-wire mode) USI data input (two-wire mode) Pin change interrupt 1, source 9
PB2	ADC7: TXD1: DO: PCINT10:	ADC input channel 7 UART1 data transmitter USI data output (three-wire mode) Pin change interrupt 1, source 10
PB3	ADC8: OC1A: PCINT11:	ADC input channel 8 Timer/Counter1 compare match A output Pin change interrupt 1, source 11

- **Port B, Bit 0 – ADC5/TXD0/PCINT8**
  - ADC5: Analog to digital converter, channel 5.
  - TXD0: UART0 data transmitter.
  - PCINT8: Pin change interrupt source 8. The PB0 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 1 – ADC6/RXD1/DI/SDA/PCINT9**
  - ADC6: Analog to digital converter, channel 6.
  - RXD1: UART1 data receiver.
  - DI: Data input in USI three-wire mode. USI three-wire mode does not override normal port functions. The pin must therefore be configured as an input for the DI function.
  - SDA: Two-wire mode serial interface data.
  - PCINT9: Pin change interrupt source 9. The PB1 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 2 – ADC7/TXD1/DO/PCINT10**
  - ADC7: Analog to digital converter, channel 7.
  - TXD1: UART1 data transmitter.
  - DO: Data output in USI three-wire mode. Data output (DO) overrides the PORTB2 value and is driven to the port when the data direction bit DDB2 is set (“1”). However the PORTB2 bit still controls the pull-up, enabling pull-up if the direction is input and PORTB2 is set (“1”).
  - PCINT10: Pin change interrupt source 10. The PB2 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 3 – ADC8/OC1A/PCINT11**
  - ADC8: Analog to digital converter, channel 8.
  - OC1A: Output compare match output. The PB3 pin can serve as an external output for the Timer/Counter1 compare match A. The pin has to be configured as an output (DDB3 set (“1”)) to serve this function. This is also the output pin for the PWM mode timer function.
  - PCINT11: Pin change interrupt source 11. The PB3 pin can serve as an external interrupt source for pin change interrupt 1.

Table 11-8 on page 65 and Table 11-9 on page 65 compare the alternate functions of port B to the overriding signals shown in Figure 11-6 on page 59.

Table 11-8. Overriding Signals for Alternate Functions in PB[3:2]

Signal Name	PB3/ADC8/OC1A/PCINT11	PB2/ADC7/TXD1/DO/PCINT10
PUOE	0	TXD1_OE
PUOV	0	0
DDOE	0	TXD1_OE
DDOV	0	0
PVOE	OC1A enable	TXD1_OE + USI_THREE_WIRE
PVOV	OC1A	(TXD1_OE • TXD_PVOV) + ($\overline{\mathrm{TXD1\_OE}}$ • DO)
PTOE	0	0
DIEOE	PCINT11 • PCIE1 + ADC8D	PCINT10 • PCIE1 + ADC7D
DIEOV	PCINT11 • PCIE1	PCINT10 • PCIE1 + INT0
DI	PCINT11 input	PCINT10 input
AIO	ADC8 input	ADC7 input

Table 11-9. Overriding Signals for Alternate Functions in PB[1:0]

Signal Name	PB1/ADC5/RXD1/DI/SDA/PCINT9	PB0/ADC4/TXD0/PCINT8
PUOE	RXD1_OE	TXD0_OE
PUOV	PUEB1	0
DDOE	RXD1_EN + USI_TWO_WIRE	TXD0_OE
DDOV	$\overline{\mathrm{RXD1\_EN}}$) • ($\overline{\mathrm{SDA}}$ + $\overline{\mathrm{PORTB1}}$) • DDB1	
PVOE	$\overline{\mathrm{RXD1\_EN}}$) • USI_TWO_WIRE • DDB1	TXD0_OE
PVOV	0	TXD0_PVOV
PTOE	0	0
DIEOE	USISIE + (PCINT9 • PCIE1) + ADC6D	(PCINT8 • PCIE1) + ADC5D
DIEOV	USISIE + (PCINT9 • PCIE1)	PCINT8 • PCIE1
DI	RXD1/DI/SDA/PCINT9 input	PCINT8 input
AIO	ADC6 input	ADC5 input

11.3.3 Alternate Functions of Port C

The port C pins with alternate function are shown in Table 11-7.

Table 11-10. Port C Pins Alternate Functions

Port Pin	Alternate Function	
PC0	ADC9: XCK0: OC0A: PCINT12:	ADC input channel 9 USART 0 transfer clock (synchronous mode) Timer/Counter0 compare match A output Pin change interrupt 2, source 12
PC1	ADC10: XCK1: USCK: SCL: ICP1: PCINT13:	ADC input channel 10 USART 1 transfer clock (synchronous mode) USI clock (three-wire mode) USI clock (two-wire mode) Timer/Counter1 input capture pin Pin change interrupt 2, source 13
PC2	ADC11: INT0: CLKO: PCINT14:	ADC input channel 11 External interrupt 0 input System clock output Pin change interrupt 2, source 14
PC3	$\overline{\text{RESET}}$: dW: PCINT15:	Reset pin debugWIRE I/O Pin change interrupt 2, source 15
PC4	XTAL2: PCINT16:	Crystal oscillator output Pin change interrupt 2, source 16
PC5	XTAL1: CLKI: PCINT17:	Crystal oscillator input External clock input Pin change interrupt 2, source 17

- **Port C, Bit 0 – ADC9/XCK0/OC0A/PCINT12**
  - ADC9: Analog to digital converter, channel 9.
  - XCK0: USART0 transfer clock used by synchronous transfer mode only.
  - OC0A: Output compare match output. The PC0 pin can serve as an external output for the Timer/Counter0 compare match A. The PC0 pin has to be configured as an output (DDC0 set (“1”)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.
  - PCINT12: Pin change interrupt source 12. The PC0 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port C, Bit 1 – ADC10/XCK1/USCK/SCL/ICP1/PCINT13**
  - ADC10: Analog to digital converter, channel 10.
  - XCK1: USART1 transfer clock used only by synchronous transfer mode.
  - USCK: Three-wire mode universal serial interface clock.
  - SCL: Two-wire mode serial clock for USI two-wire mode.
  - ICP1: Input capture pin. The PC1 pin can act as an input capture pin for Timer/Counter1.
  - PCINT13: Pin change interrupt source 13. The PC1 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port C, Bit 2 – ADC11/INT0/CLKO/PCINT14**
  - ADC11: Analog to digital converter, channel 11.
  - INT0: External interrupt request 0.
  - CLKO: System clock output. The system clock can be output on the PC2 pin. The system clock is output if the CKOUT fuse is programmed, regardless of the PORTC2 and DDC2 settings. It is also output during reset.
  - PCINT14: Pin change interrupt source 14. The PC2 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port C, Bit 3 – $\overline{\text{RESET}}$/dW/PCINT15**
  - $\overline{\text{RESET}}$: External $\overline{\text{reset}}$ input is active low and enabled by unprogramming ("1") the RSTDISBL fuse. Pull-up is activated and output driver and digital input are deactivated when the pin is used as the $\overline{\text{RESET}}$ pin.
  - dW: When the debugWIRE enable (DWEN) fuse is programmed and lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bidirectional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.
  - PCINT15: Pin change interrupt source 15. The PC3 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port C, Bit 4 – XTAL2/PCINT16**
  - XTAL2: Chip clock oscillator pin 2. Used as clock pin for all chip clock sources except internal calibrated oscillator and external clock. When used as a clock pin, the pin cannot be used as an I/O pin. When using internal calibrated oscillator as a chip clock source, PC4 serves as an ordinary I/O pin.
  - PCINT16: Pin change interrupt source 16. The PC4 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port C, Bit 5 – XTAL1/CLKI/PCINT17**
  - XTAL1: Chip clock oscillator pin 1. Used for all chip clock sources except internal calibrated oscillator. When used as a clock pin, the pin cannot be used as an I/O pin. When using internal calibrated oscillator as a chip clock source, PC5 serves as an ordinary I/O pin.
  - CLKI: Clock input from an external clock source (see Section 7.2.1 "External Clock" on page 26).
  - PCINT17: Pin change interrupt source 17. The PC5 pin can serve as an external interrupt source for pin change interrupt 1.

Table 11-4 and Table 11-6 compares the alternate functions of port A to the overriding signals shown in Figure 11-6 on page 59.

Table 11-11. Overriding Signals for Alternate Functions in PC[5:3]

Signal Name	PC5/XTAL1/CLKI/PCINT17	PC4/XTAL2/ PCINT16	PC3/$\overline{\text{RESET}}$/dW/ PCINT15
PUOE	EXT_CLOCK(1) + EXT_OSC(2)	EXT_OSC(2)	$\overline{\text{RSTDISBL}}$(3) + DEBUGWIRE_ENABLE(4)
PUOV	0	0	1
DDOE	EXT_CLOCK(1) + EXT_OSC(2)	EXT_OSC(2)	$\overline{\text{RSTDISBL}}$(3) + DEBUGWIRE_ENABLE(4)
DDOV	0	0	DEBUGWIRE_ENABLE(4) • $\overline{\text{debugWIRE transmit}}$
PVOE	EXT_CLOCK(1) + EXT_OSC(2)	EXT_OSC(2)	$\overline{\text{RSTDISBL}}$(3) + DEBUGWIRE_ENABLE(4)
PVOV	0	0	0
PTOE	0	0	0
DIEOE	EXT_CLOCK(1) + EXT_OSC(2) + (PCINT17 • PCIE2)	EXT_OSC(2) + PCINT16 • PCIE2	RSTDISBL(3) + DEBUGWIRE_ENABLE(4) + PCINT15 • PCIE2
DIEOV	(EXT_CLOCK(1) • $\overline{\text{PWR\_DOWN}}$) + ($\overline{\text{EXT\_CLOCK}}$(1) • $\overline{\text{EXT\_CLOCK}}$(1) • PCINT17 • PCIE2)	$\overline{\text{EXT\_OSC}}$(2) • PCINT16 • PCIE2	DEBUGWIRE_ENABLE(4) + (RSTDISBL(3) • PCINT15 • PCIE2)
DI	CLOCK/PCINT17 input	PCINT16 input	dW/PCINT15 input
AIO	XTAL1	XTAL2	

Notes: 1. EXT_CLOCK = external clock is selected as system clock.

2. EXT_OSC = crystal oscillator or low frequency crystal oscillator is selected as system clock.

3. RSTDISBL is "1" when the fuse is "0" (programmed).

4. DebugWIRE is enabled when the DWEN fuse is programmed and lock bits are unprogrammed.

Table 11-12. Overriding Signals for Alternate Functions in PC[2:0]

Signal Name	PC2/ADC11/INT0/CLKO/ PCINT14	PC1/ADC10/XCK1/USCK/ SCL/ICP1/PCINT13	PC0/ADC9/XCK0/ OC0A/PCINT12
PUOE	CKOUT_IO	USI_TWO_WIRE	0
PUOV	0	0	0
DDOE	CKOUT_IO	USI_TWO_WIRE	0
DDOV	1	(USI_SCL_HOLD + $\overline{\text{PORTC1}}$) • DDC1	0
PVOE	CKOUT_IO	XCKO1_PVOE + USI_TWO_WIRE • DDC1	XCKO0_PVOE + OC0A Enable
PVOV	CKOUT_IO • System Clock	XCKO1_PVOV	XCKO0_PVOV + OC0A
PTOE	0	USI_PTOE	0
DIEOE	INT0 + (PCINT14 • PCIE2) + ADC11D	XCK1 input enable + USISIE + (PCINT13 • PCIE2) + ADC10D	XCK0 input enable + (PCINT12 • PCIE2) + ADC9D
DIEOV	INT0 + (PCINT14 • PCIE2)	USISIE + (PCINT13 • PCIE2)	PCINT12 • PCIE2
DI	INT0/PCINT14 input	XCK1/USCK/SCL/ICP1/ PCINT13 input	XCK0/PCINT12 input
AIO	ADC11 input	ADC10 input	ADC9 input

11.4 Register Description

11.4.1 PORTCR – Port Control Register

Bit	7	6	5	4	3	2	1	0	
0x13 (0x33)	–	–	–	–	–	BBMC	BBMB	BBMA	PORTCR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved and always read as “0”.

- **Bit 2 – BBMC: Break-Before-Make Mode Enable**

When this bit is set, the break-before-make mode is activated for the entire port C. The intermediate tri-state cycle is then inserted when writing DDRCn to make an output. For further information, see Section 11.2.3 “Break-Before-Make Switching” on page 55.

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

When this bit is set, the break-before-make mode is activated for the entire port B. The intermediate tri-state cycle is then inserted when writing DDRBn to make an output. For further information, see Section 11.2.3 “Break-Before-Make Switching” on page 55.

- **Bit 0 – BBMA: Break-Before-Make Mode Enable**

When this bit is set, the break-before-make mode is activated for the entire port A. The intermediate tri-state cycle is then inserted when writing DDRAn to make an output. For further information, see Section 11.2.3 “Break-Before-Make Switching” on page 55.

11.4.2 PUEA – Port A Pull-Up Enable Control Register

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	PUEA7	PUEA6	PUEA5	PUEA4	PUEA3	PUEA2	PUEA1	PUEA0	PUEA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

11.4.3 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	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	

11.4.4 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x10 (0x30)	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	

11.4.5 PINA – Port A Input Pins

Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	**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	

11.4.6 PUEB – Port B Pull-Up Enable Control Register

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	–	–	–	–	**PUEB3**	**PUEB2**	**PUEB1**	**PUEB0**	**PUEB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

11.4.7 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	–	–	–	–	**PORTB3**	**PORTB2**	**PORTB1**	**PORTB0**	**PORTB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

11.4.8 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	–	–	–	–	**DDB3**	**DDB2**	**DDB1**	**DDB0**	**DDRB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

11.4.9 PINB – Port B Input Pins

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	–	–	–	–	**PINB3**	**PINB2**	**PINB1**	**PINB0**	**PINB**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	N/A	N/A	N/A	N/A	

11.4.10 PUEC – Port C Pull-Up Enable Control Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	–	–	**PUEC5**	**PUEC4**	**PUEC3**	**PUEC2**	**PUEC1**	**PUEC0**	**PUEC**
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	

11.4.11 PORTC – Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	–	–	**PORTC5**	**PORTC4**	**PORTC3**	**PORTC2**	**PORTC1**	**PORTC0**	**PORTC**
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	

11.4.12 DDRC – Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	**–**	**–**	**DDC5**	**DDC4**	**DDC3**	**DDC2**	**DDC1**	**DDC0**	**DDRC**
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	

11.4.13 PINC – Port C Input Pins

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	**–**	**–**	**PINC5**	**PINC4**	**PINC3**	**PINC2**	**PINC1**	**PINC0**	**PINC**
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	N/A	N/A	N/A	N/A	N/A	

12. 8-bit Timer/Counter0 with PWM

12.1 Features

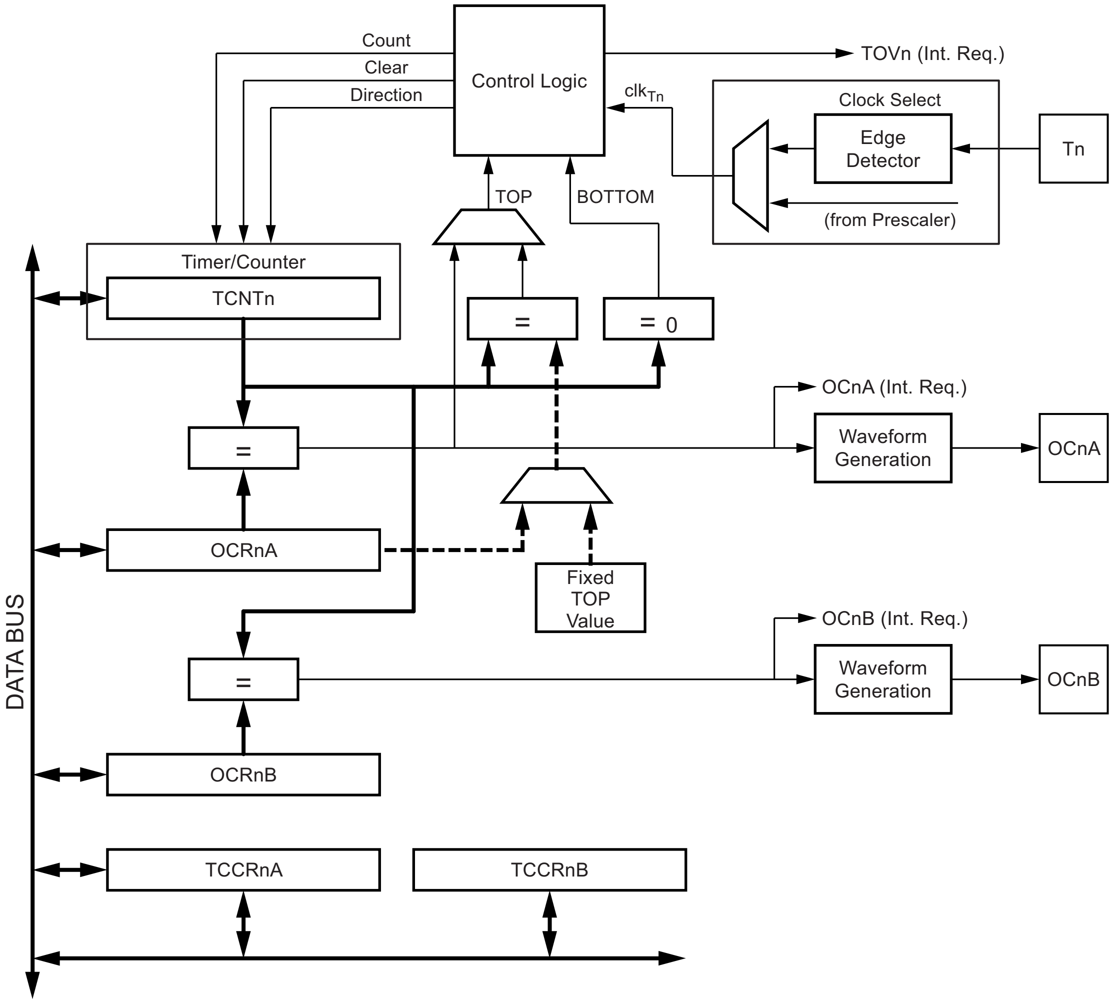
- Two independent output compare units
- Double-buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch-free phase-correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

12.2 Overview

Timer/Counter0 is a general purpose 8-bit timer/counter module with two independent output compare units and PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 12-1. For the specific placement of I/O pins, see Figure 1-1 on page 3. CPU-accessible I/O registers, including I/O bits and I/O pins are shown in bold. The device-specific I/O register and bit locations are listed in Section 12.9 "Register Description" on page 82.

Figure 12-1. 8-bit Timer/Counter Block Diagram



12.2.1 Registers

The Timer/Counter (TCNT0) and output compare registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated as "int.req." in Figure 12-1) 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 Figure 2-1.

The Timer/Counter can be clocked internally via the prescaler or by an external clock source on the T0 pin. The clock select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{T0}).

The double-buffered output compare registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins (OC0A and OC0B) (for more information, see Section 12.5 "Output Compare Unit" on page 74). The compare match event also sets the compare flag (OCF0A or OCF0B) which can be used to generate an output compare interrupt request.

12.2.2 Definitions

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

The definitions in Table 12-1 are also used extensively throughout the document.

Table 12-1. Definitions

Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A register. The assignment depends on the operating mode.

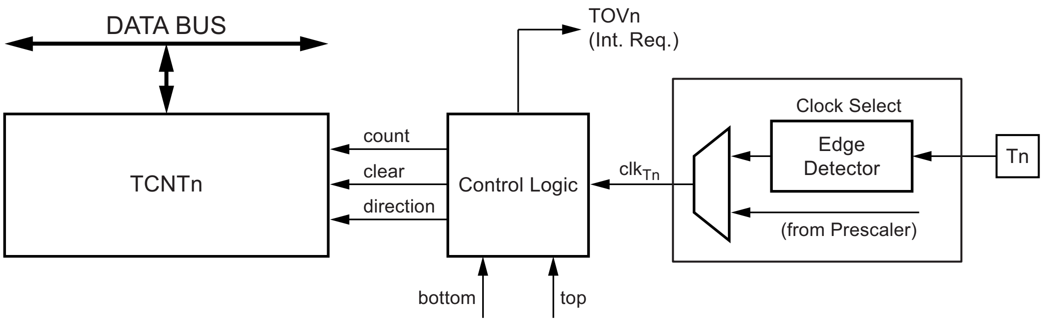
12.3 Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS0[2:0]) bits located in the Timer/Counter control register (TCCR0B). For details on clock sources and prescaler, see Section 14. "Timer/Counter Prescaler" on page 112.

12.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bidirectional counter unit. Figure 12-2 shows a block diagram of the counter and its surroundings.

Figure 12-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT0 by 1
Direction	Select between increment and decrement
Clear	Clear TCNT0 (set all bits to “0”)
clk$_{Tn}$	Timer/Counter clock, referred to as clk$_{T0}$ in the following
TOP	Indicates that TCNT0 has reached maximum value
BOTTOM	Indicates that TCNT0 has reached minimum value (“0”)

Depending on the operating mode used, the counter is cleared, incremented, or decremented at each timer clock (clk$_{T0}$). clk$_{T0}$ can be generated from an external or internal clock source, selected by the clock select bits (CS0[2:0]). When no clock source is selected (CS0[2:0] = 0), the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk$_{T0}$ is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter control register (TCCR0A) and the WGM02 bit located in the Timer/Counter control register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare output OC0A. For more details about advanced counting sequences and waveform generation, see Section 12.7 “Operating Modes” on page 76.

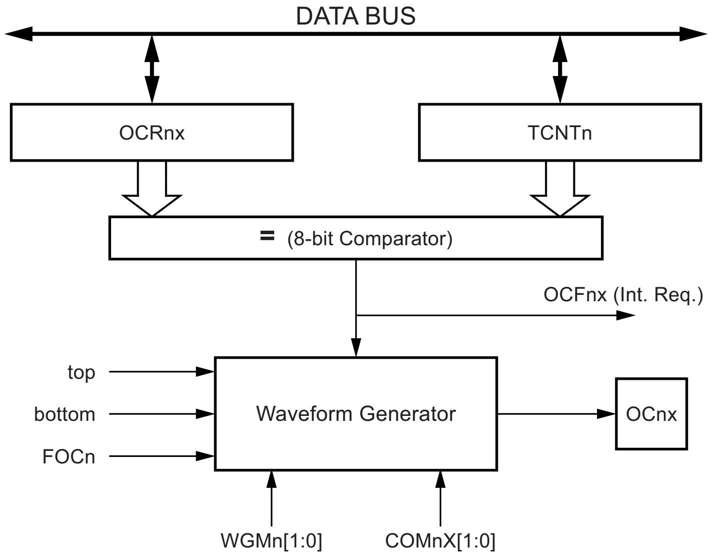
The Timer/Counter overflow flag (TOV0) is set according to the operating mode selected by the WGM0[1:0] bits. TOV0 can be used for generating a CPU interrupt.

12.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match sets the output compare flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared via software by writing a logic one to its I/O bit location. The waveform generator uses the match signal to generate an output according to the operating mode set by the WGM0[2:0] bits and compare output mode (COM0x1:0) bits. The MAX and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some operating modes (see Section 12.7 “Operating Modes” on page 76).

Figure 12-3 shows a block diagram of the output compare unit.

Figure 12-3. Output Compare Unit, Block Diagram



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

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

12.5.1 Force Output Compare

In non-PWM waveform generation modes the match output of the comparator can be forced by writing a “1” to the force output compare (0x) bit. Forcing compare match does not set the OCF0x flag or reload/clear the timer, but the OC0x pin is updated as if a real compare match had occurred (the COM0x[1:0] bits settings define whether the OC0x pin is set, cleared or toggled).

12.5.2 Compare Match Blocking by TCNT0 Write

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

12.5.3 Using the Output Compare Unit

Because writing TCNT0 in any operating mode blocks all compare matches for one timer clock cycle, risks arise from changing TCNT0 while using the output compare unit, regardless of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match is missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down-counting.

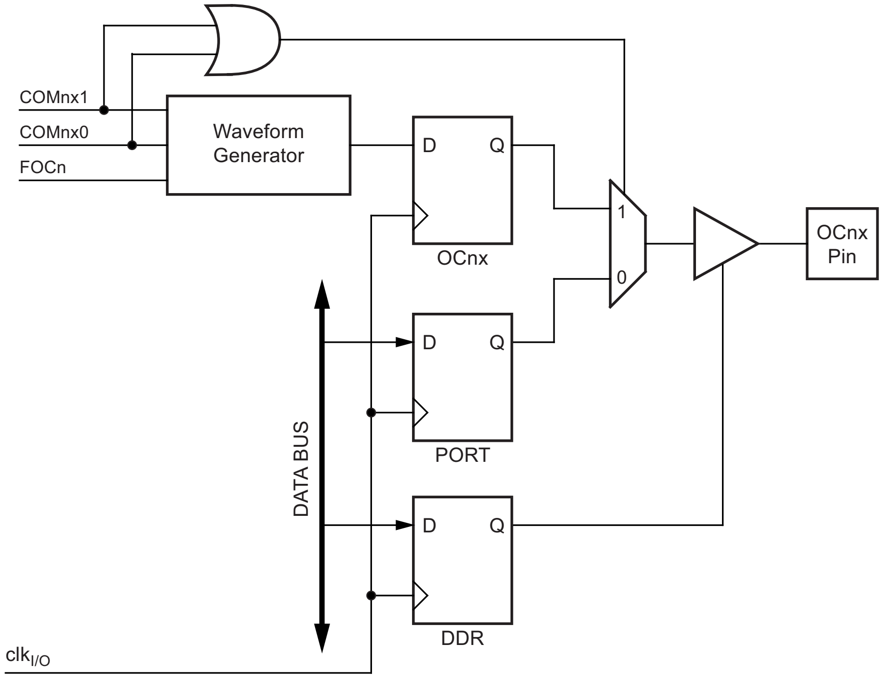
The setup of the OC0x should be performed before setting the data direction register for the port pin to output. The easiest way to set the OC0x value is to use the force output compare (0x) strobe bits in normal mode. The OC0x registers retain their values even when changing between waveform generation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

Be aware that the COM0x[1:0] bits are not double-buffered together with the compare value. Changing the COM0x[1:0] bits takes effect immediately.

12.6 Compare Match Output Unit

The compare output mode (COM0x[1:0]) bits have two functions. The waveform generator uses the COM0x[1:0] bits for defining the output compare (OC0x) state at the next compare match. Also, the COM0x[1:0] bits control the OC0x pin output source. Figure 12-4 on page 76 shows a simplified schematic of the logic affected by the COM0x[1:0] 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) affected by the COM0x[1:0] bits are displayed. When referring to the OC0x state, the reference is for the internal OC0x register, not the OC0x pin. If a system reset occurs, the OC0x register is reset to “0”.

Figure 12-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC0x) from the waveform generator if either of the COM0x[1:0] bits are set. However, the OC0x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC0x pin (DDR_OC0x) must be set as the output before the OC0x value is visible on the pin. The port override function is independent of the waveform generation mode.

The design of the output compare pin logic allows initialization of the OC0x state before the output is enabled. Note that some COM0x[1:0] bit settings are reserved for certain operating modes (see Section 12.9 “Register Description” on page 82).

12.6.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM0x[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0x[1:0] = 0 tells the waveform generator that no action on the OC0x register is to be performed on the next compare match. For compare output actions in the non-PWM modes, see Table 12-2 on page 82. For fast PWM mode, see Table 12-3 on page 82 and for phase correct PWM, see Table 12-4 on page 82.

A change in the COM0x[1:0] bits state has an effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the 0x strobe bits.

12.7 Operating Modes

The operating modes, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM0[2:0]) and compare output mode (COM0x[1:0]) bits. The compare output mode bits do not affect the counting sequence while the waveform generation mode bits do. The COM0x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes, the COM0x[1:0] bits control whether the output should be set, cleared, or toggled at a compare match (see Section 12.7 “Operating Modes” on page 76).

For detailed timing information, see Figure 12-8 on page 80, Figure 12-9 on page 80, Figure 12-10 on page 81, and Figure 12-11 on page 81 in Section 12.8 “Timer/Counter Timing Diagrams” on page 80.

12.7.1 Normal Mode

The simplest operating mode is the normal mode (WGM0[2:0] = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the BOTTOM (0x00). In normal operation the Timer/Counter overflow flag (TOV0) is set in the same timer clock cycle when the TCNT0 becomes “0”. The TOV0 flag in this case behaves as 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 normal mode; a new counter value can be written at any time.

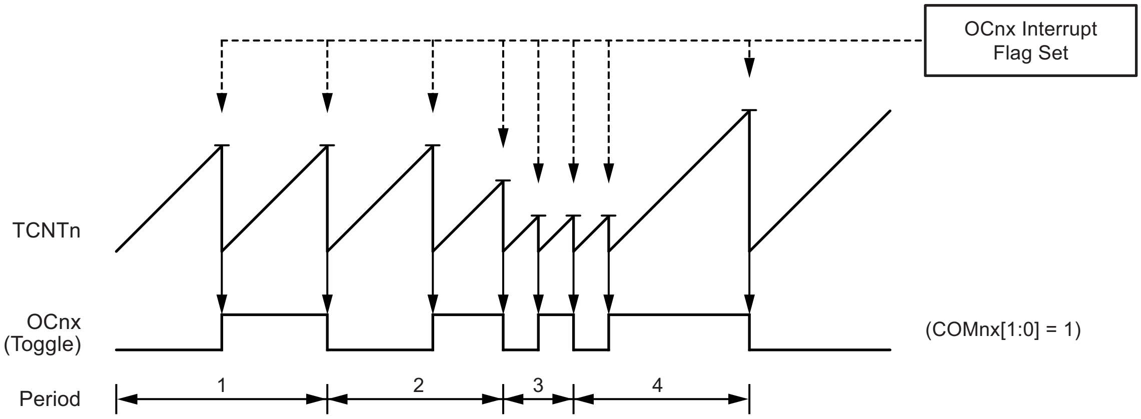
The output compare unit can be used to generate interrupts at a given time. Using the output compare to generate waveforms in normal mode is not recommended because this occupies too much CPU time.

12.7.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM0[2:0] = 2) the OCR0A register is used to manipulate the counter resolution. In CTC mode the counter is cleared to “0” when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter and therefore its resolution as well. 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 12-5. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 12-5. CTC Mode, Timing Diagram



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

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logic level on each compare match by setting the compare output mode bits to toggle mode (COM0A[1:0] = 1). The OC0A value is not visible on the port pin unless the data direction for the pin is set to output. The waveform generated has a maximum frequency of $_0$ = $f_{clk\_I/O}/2$ when OCR0A is set to “0” (0x00). The waveform frequency is defined by the following equation:

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

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

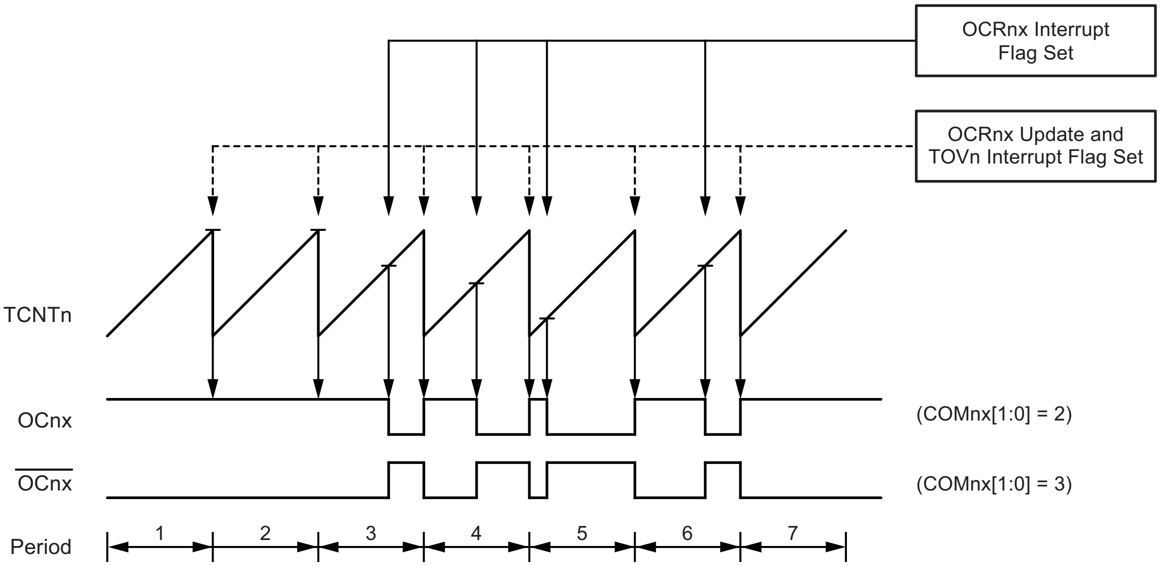
As for normal operating mode, the TOV0 flag is set in the same timer clock cycle in which the counter counts from MAX to 0x00.

12.7.3 Fast PWM Mode

The fast pulse width modulation or fast PWM mode (WGM0[2:0] = 3 or 7) provides a high-frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP, then restarts from BOTTOM. TOP is defined as 0xFF when WGM0[2:0] = 3, and OCR0A when WGM0[2:0] = 7. In non-inverting compare output mode the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting compare output mode the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be two times higher than the phase correct PWM mode that uses dual-slope operation. A high frequency makes the fast PWM mode highly suitable for power regulation, rectification, and DAC applications and also allows external components (coils, capacitors) of small physical size, thus reducing overall system cost.

In fast PWM mode the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode can be seen in Figure 12-6 on page 78. The TCNT0 value is shown in the timing diagram as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 12-6. Fast PWM Mode, Timing Diagram



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

In fast PWM mode the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x[1:0] bits to “2” produces a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x[1:0] to “3”. Setting the COM0A[1:0] bits to “1” allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 12-3 on page 82). The actual OC0x value is only visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x register at the compare match between OCR0x and TCNT0. The counter is cleared (changes from TOP to BOTTOM) by clearing (or setting) the OC0x register at the timer clock cycle.

The PWM frequency for the output can be calculated with this equation:

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

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

The extreme values for the OCR0A register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output is a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX results in a constantly high or low output (depending on the polarity of the output set by the COM0A[1:0] bits.)

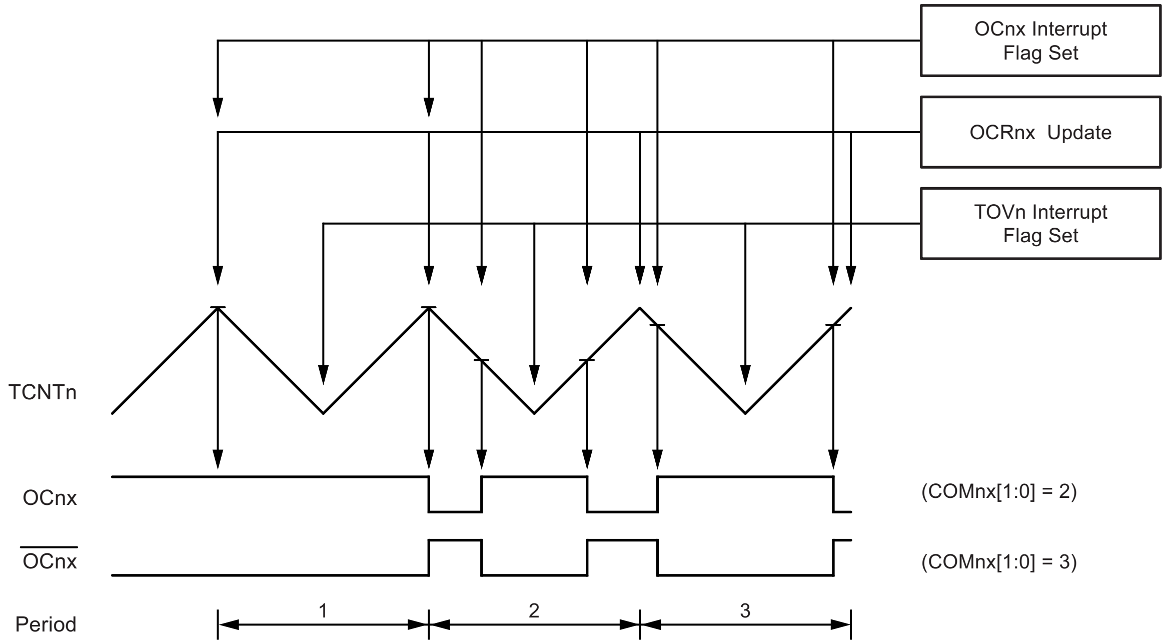
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logic level on each compare match (COM0x[1:0] = 1). The waveform generated has a maximum frequency of $_0$ = $f_{clk\_I/O}/2$ when OCR0A is set to "0". This feature is similar to the OC0A toggle in CTC mode except the double-buffer feature of the output compare unit is enabled in the fast PWM mode.

12.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM0[2:0] = 1 or 5) provides a high-resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM0[2:0] = 1, and OCR0A when WGM0[2:0] = 5. In non-inverting compare output mode the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x while up-counting and set on the compare match while down-counting. In inverting output compare mode the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value is equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode can be seen in Figure 12-7 on page 79. The TCNT0 value is shown in the timing diagram as a histogram to illustrate the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 12-7. Phase Correct PWM Mode, Timing Diagram



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

In phase correct PWM mode the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x[1:0] bits to "2" produces a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x[1:0] to "3": Setting the COM0A0 bits to "1" allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 12-4 on page 82). The actual OC0x value is only visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x register at compare match between OCR0x and TCNT0 when the counter decrements. When using phase correct PWM, the PWM frequency for the output can be calculated with this equation:

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

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

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

At the very start of period 2 in Figure 12-7 on page 79 OCn has a transition from high to low even though there is no compare match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that result in a transition without a compare match.

- OCR0A changes its value from MAX as shown in Figure 12-7 on page 79. When the OCR0A value is MAX, the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM, the OCn value at MAX must correspond to the result of an up-counting compare match.
- The timer starts counting from a value higher than the one in OCR0A, thus missing the compare match and hence the OCn change that would have happened on the way up.

12.8 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 12-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 12-8. Timer/Counter Timing Diagram, No Prescaling

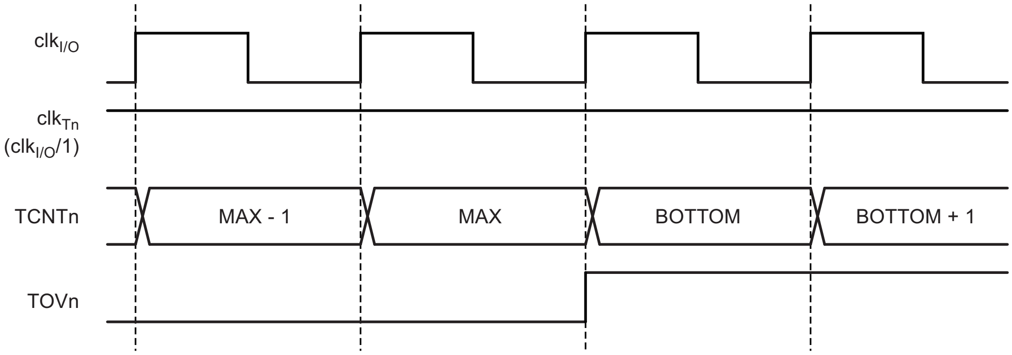


Figure 12-9 shows the same timing data, but with the prescaler enabled.

Figure 12-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

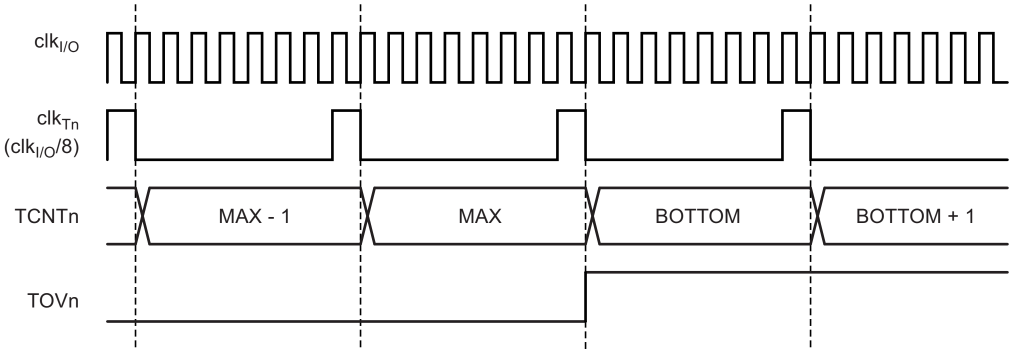


Figure 12-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode where OCR0A is TOP.

Figure 12-10.Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

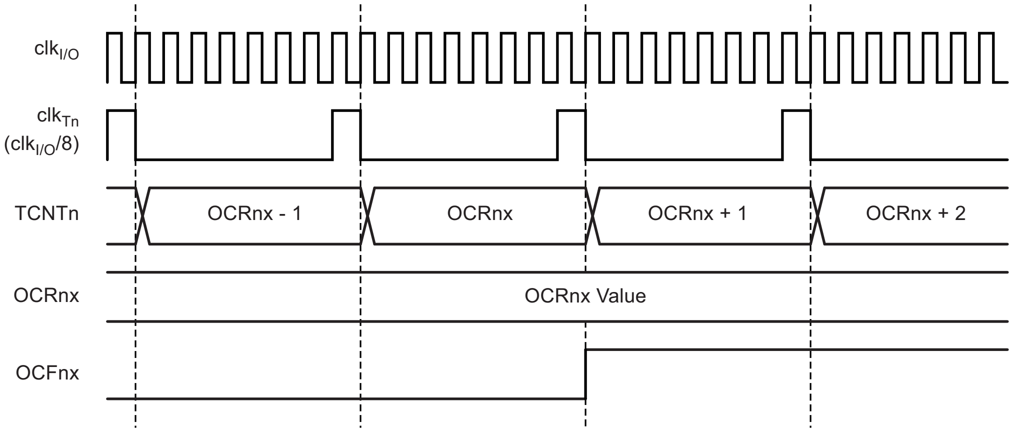
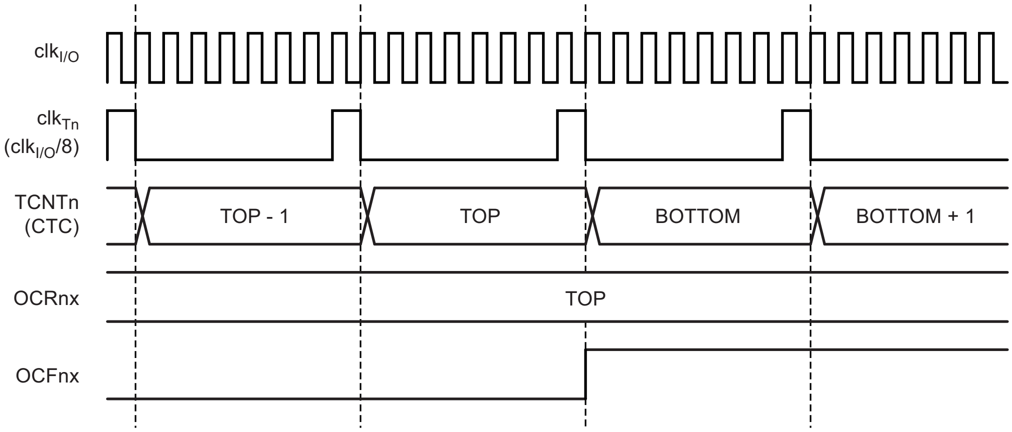


Figure 12-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 12-11.Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler ($f_{clk\_I/O}/8$)



12.9 Register Description

12.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM0A[1:0]: Compare Match Output A Mode**

These bits control the output compare pin (OC0A) behavior. If one or both of the COM0A[1:0] bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A[1:0] bits depends on the WGM0[2:0] bit setting. Table 12-2 shows the COM0A[1:0] bit functionality when the WGM0[2:0] bits are set to normal or CTC mode (non-PWM).

Table 12-2. Compare Output Mode, Non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 12-3 shows COM0A[1:0] bit functionality when WGM0[1:0] bits are set to fast PWM mode.

Table 12-3. Compare Output Mode, Fast PWM Mode[(1)]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected

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

1	0	Clear OC0A on compare match Set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on compare match Clear OC0A at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM (see Section 12.7.3 “Fast PWM Mode” on page 78).

Table 12-4 shows COM0A[1:0] bit functionality when WGM0[2:0] bits are set to phase correct PWM mode.

Table 12-4. Compare Output Mode, Phase Correct PWM Mode[(1)]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected

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

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

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP (see Section 12.7.4 “Phase Correct PWM Mode” on page 79).

- **Bits 5:4 – COM0B[1:0]: Compare Match Output B Mode**

These bits control the output compare pin (OC0B) behavior. If one or both of the COM0B[1:0] bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B[1:0] bits depends on the WGM0[2:0] bit setting. Table 12-5 shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to normal or CTC mode (non-PWM).

Table 12-5. Compare Output Mode, Non-PWM Mode

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

Table 12-6 shows COM0B[1:0] bit functionality when WGM0[2:0] bits are set to fast PWM mode.

Table 12-6. Compare Output Mode, Fast PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Reserved
1	0	Clear OC0B on compare match, set OC0B at BOTTOM (non-inverting mode)
1	1	Set OC0B on compare match, clear OC0B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM (see Section 12.7.3 "Fast PWM Mode" on page 78).

Table 12-7 shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

Table 12-7. Compare Output Mode, Phase Correct PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Reserved
1	0	Clear OC0B on compare match when up-counting. Set OC0B on compare match when down-counting.
1	1	Set OC0B on compare match when up-counting. Clear OC0B on compare match when down-counting.

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP (see Section 12.7.4 "Phase Correct PWM Mode" on page 79).

- **Bits 3:2 – Res: Reserved Bits**

These bits are reserved and always read as "0".

- **Bits 1:0 – WGM0[1:0]: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used (see Table 12-8). Operating modes supported by the Timer/Counter unit are normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 12.7 "Operating Modes" on page 76).

Table 12-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Operating Mode	TOP	Update of OCRx at	TOV Flag Set on(1)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

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

12.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

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

However, to ensure compatibility with future devices, this bit must be set to "0" when TCCR0B is written while operating in PWM mode. When writing a logic one to the FOC0A bit, an immediate compare match is forced on the waveform generation unit. The OC0A output is changed according to its COM0A[1:0] bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A[1:0] bits that determines the effect of the forced compare.

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

The FOC0A bit is always read as "0".

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

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

However, to ensure compatibility with future devices, this bit must be set to "0" when TCCR0B is written when operating in PWM mode. When writing a logic one to the FOC0B bit, an immediate compare match is forced on the waveform generation unit. The OC0B output is changed according to its COM0B[1:0] bits setting. Note that the FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B[1:0] bits that determines the effect of the forced compare.

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

The FOC0B bit is always read as "0".

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny1634 and always read as "0".

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

See description in Section 12.9.1 “TCCR0A – Timer/Counter Control Register A” on page 82.

- **Bits 2:0 – CS0[2:0]: Clock Select**

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

Table 12-9. Clock Select Bit Description

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

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

12.9.3 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x19 (0x39)	**TCNT0[7:0]**								**TCNT0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

12.9.4 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	**OCR0A[7:0]**								**OCR0A**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

12.9.5 OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	**OCR0B[7:0]**								**OCR0B**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

12.9.6 TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	**TOIE1**	**OCIE1A**	**OCIE1B**	**–**	**ICIE1**	**OCIE0B**	**TOIE0**	**OCIE0A**	**TIMSK**
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	

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

When the OCIE0B bit is written to “1” 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 (TIFR).

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

When the TOIE0 bit is written to “1” 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/Counter0 interrupt flag register (TIFR).

- **Bit 0 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to “1” 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/Counter0 interrupt flag register (TIFR).

12.9.7 TIFR – Timer/Counter0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	**TOV1**	OCF1B	OCF1A	**–**	**ICF1**	**OCF0B**	**TOV0**	**OCF0A**	**TIFR**
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	

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

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

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

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

The setting of this flag depends on the WGM0[2:0] bit setting (see Table 12-8 on page 84 and Section 12-8 “Waveform Generation Mode Bit Description” on page 84).

- **Bit 0 – 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.

13. 16-bit Timer/Counter1

13.1 Features

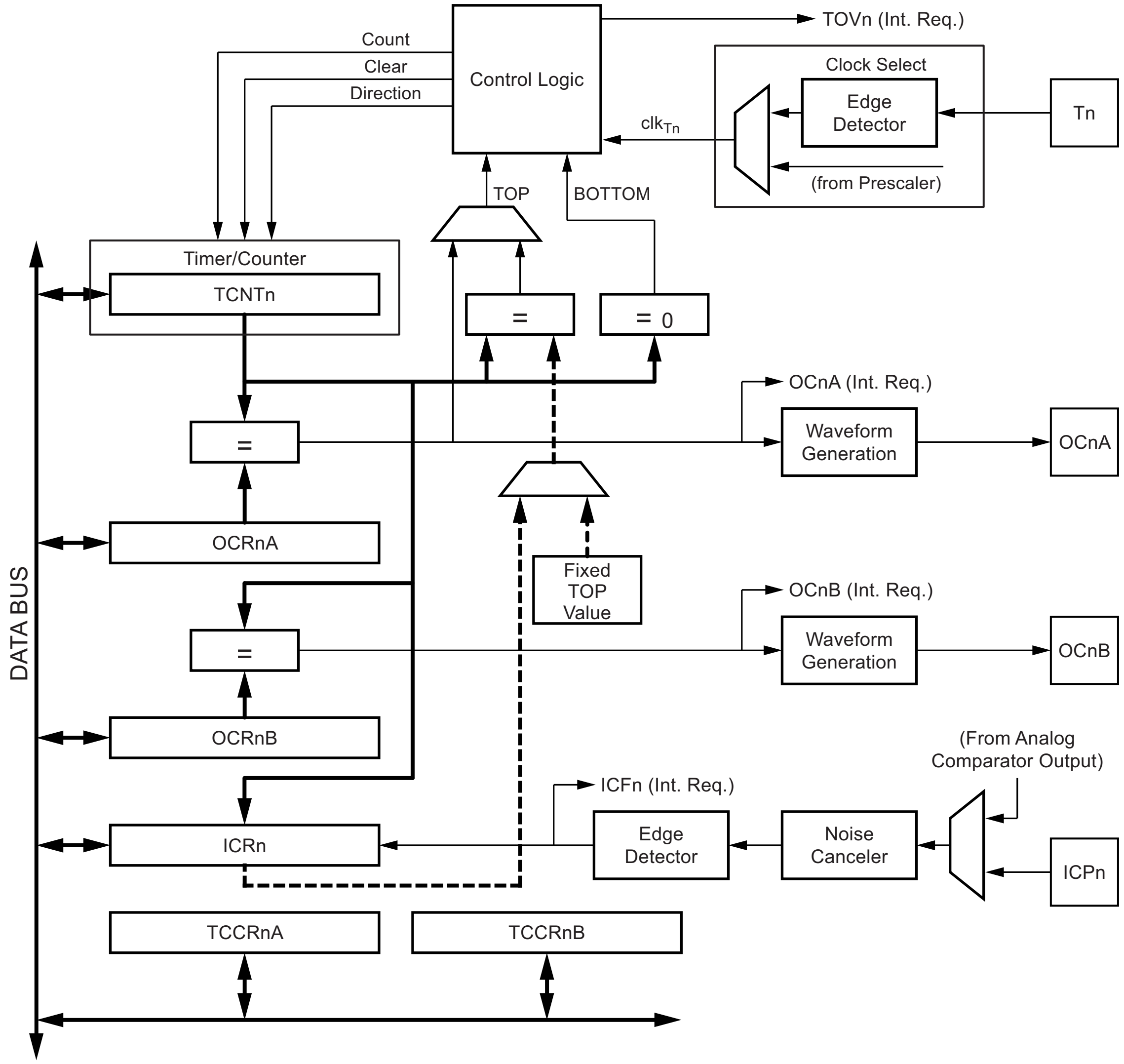
- True 16-bit design (i.e., allows 16-bit PWM)
- Two independent output compare units
- Double-buffered output compare registers
- One input capture unit
- Input capture noise canceler
- Clear timer on compare match (auto reload)
- Glitch-free, phase correct pulse width modulator (PWM)
- Variable PWM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

13.2 Overview

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

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 13-1 on page 88. For the specific placement of I/O pins, see Section 1-1 “Pinout of Atmel ATtiny1634” 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 13.11 “Register Description” on page 106.

Figure 13-1. 16-bit Timer/Counter Block Diagram



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

13.2.1 Registers

The Timer/Counter (TCNT1), output compare registers (OCR1A/B), and input capture register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in Section 13.10 “Accessing 16-bit Registers” on page 103. The Timer/Counter control registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated as “int.req.” in the figure) signals are all visible in the timer interrupt flag register (TIFR). All interrupts are individually masked with the timer interrupt mask register (TIMSK). TIFR and TIMSK are not shown in the figure.

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

The double-buffered output compare registers (OCR1A/B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pin (OC1A/B) (see Section 13.6 “Output Compare Units” on page 92). The compare match event also sets the compare match flag (OCF1A/B) which can be used to generate an output compare interrupt request.

The input capture register can capture the Timer/Counter value at a given external (edge-triggered) event on either the input capture pin (ICP1) or on the analog comparator pins (see Section 19. “Analog Comparator” on page 169). The input capture unit includes a digital filtering unit (noise canceler) for reducing the chance of capturing noise spikes.

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

13.2.2 Definitions

The following definitions are used extensively throughout the section.

Table 13-1. Definitions

Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned a fixed value or the value stored in a register. The assignment depends on the operating mode (see Table 13-5 on page 107).

13.2.3 Compatibility

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

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

The following control bits have been renamed but retained the same functionality and register locations:

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

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

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

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

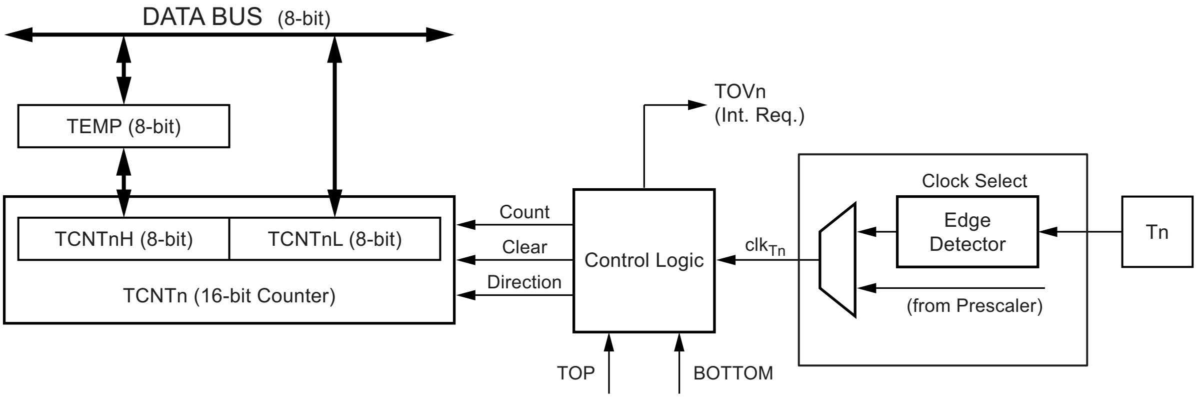
13.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS1[2:0]) bits located in the Timer/Counter control register B (TCCR1B). For details on clock sources and prescaler, see Section 14. “Timer/Counter Prescaler” on page 112.

13.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bidirectional counter unit. Figure 13-2 shows a block diagram of the counter and its surroundings.

Figure 13-2. Counter Unit Block Diagram



Description of internal signals used in Figure 13-2:

Count	Increment or decrement TCNT1 by 1
Direction	Select between increment and decrement
Clear	Clear TCNT1 (set all bits to “0”)
clk_{T1}	Timer/Counter clock
TOP	Indicates that TCNT1 has reached maximum value
BOTTOM	Indicates that TCNT1 has reached minimum value (“0”)

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

Depending on the operating mode used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the clock select bits (CS1[2:0]). When no clock source is selected (CS1[2:0] = 0), the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the waveform generation mode bits (WGM1[3:0]) located in the Timer/Counter control registers A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see Section 13.8 “Operating Modes” on page 95.

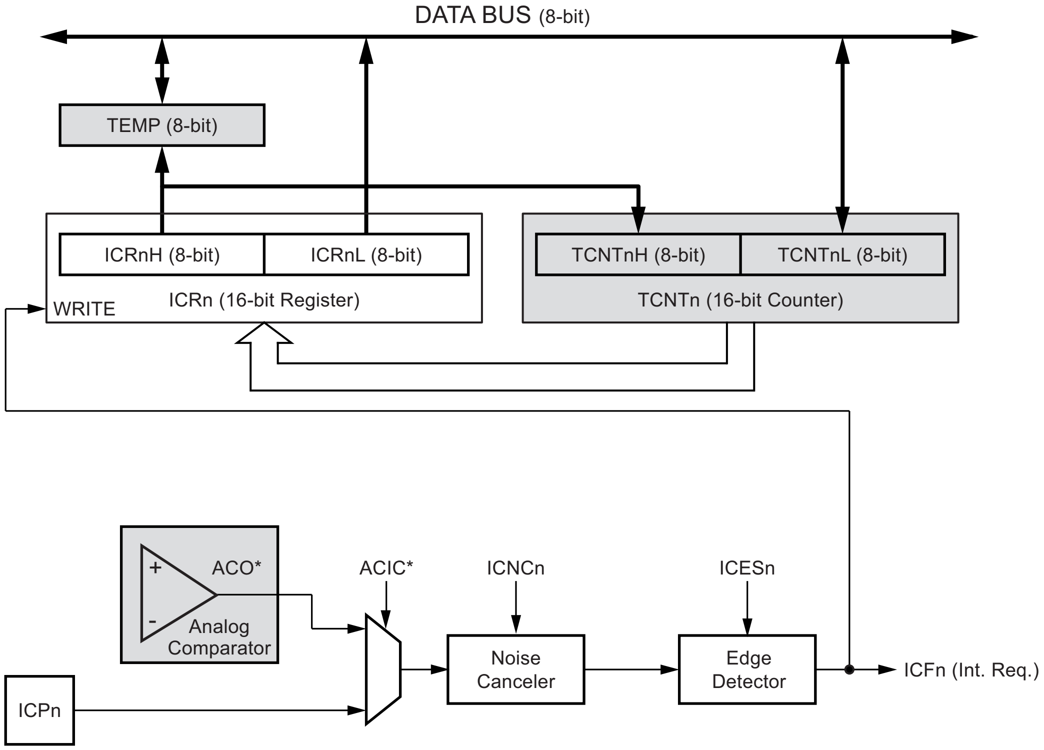
The Timer/Counter overflow flag (TOV1) is set according to the operating mode selected by the WGM1[3:0] bits. TOV1 can be used for generating a CPU interrupt.

13.5 Input Capture Unit

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

The input capture unit is illustrated by the block diagram shown in Figure 13-3 on page 91. The elements of the block diagram that are not directly a part of the input capture unit are gray shaded. The lowercase "n" in register and bit names indicates the Timer/Counter number.

Figure 13-3. Input Capture Unit Block Diagram



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

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

The ICR1 register can only be written when using a waveform generation mode that utilizes the ICR1 register for defining the counter's TOP value. In these cases, the waveform generation mode (WGM1[3:0]) bits must be set before the TOP value can be written to the ICR1 register. When writing the ICR1 register, the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers, see Section 13.10 "Accessing 16-bit Registers" on page 103.

13.5.1 Input Capture Trigger Source

The main trigger source for the input capture unit is the input capture pin (ICP1). Timer/Counter1 can alternatively use the analog comparator output as trigger source for the input capture unit. The analog comparator is selected as trigger source by setting the analog comparator input capture (ACIC) bit in the analog comparator control and status register (ACSRA). Be aware that changing the trigger source can trigger a capture. The input capture flag must therefore be cleared after the change.

Both the input capture pin (ICP1) and the analog comparator output (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 14-2 on page 113). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in waveform generation mode that uses ICR1 to define TOP.

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

13.5.2 Noise Canceler

The noise canceler uses a simple digital filtering technique to improve noise immunity. Consecutive samples are monitored in a pipeline four units deep. The signal going to the edge detector is allowed to change only when all four samples are equal.

The noise canceler is enabled by setting the input capture noise canceler (ICNC1) bit in Timer/Counter control register B (TCCR1B). When enabled, the noise canceler introduces an additional delay of four system clock cycles to a change applied to the input and before ICR1 is updated.

The noise canceler uses the system clock directly and is therefore not affected by the prescaler.

13.5.3 Using the Input Capture Unit

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

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

It is not advisable to use the input capture unit in any operating mode when the TOP value (resolution) is actively changed during operation.

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 ICR1 register has been read. After the edge is changed, the input capture flag (ICF1) must be cleared by software (writing a logic one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 flag is not required (if an interrupt handler is used).

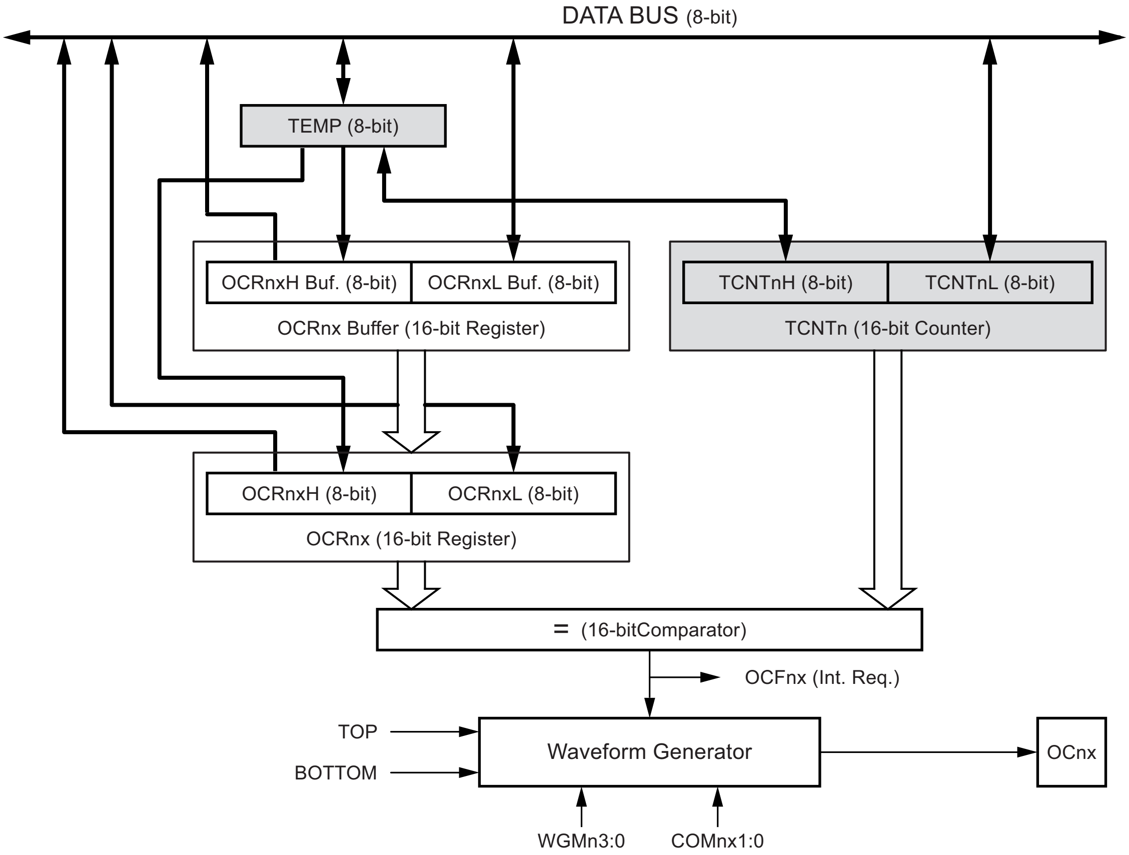
13.6 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the output compare register (OCR1x). If TCNT equals OCR1x, the comparator signals a match. A match sets the output compare flag (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the output compare flag generates an output compare interrupt. The OCF1x flag is automatically cleared when the interrupt is executed. Alternatively, the OCF1x flag can be cleared via software by writing a logic one to its I/O bit location. The waveform generator uses the match signal to generate an output according to the operating mode set by the waveform generation mode (WGM1[3:0]) bits and compare output mode (COM1x[1:0]) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some operating modes (Section 13.8 "Operating Modes" on page 95).

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

Figure 13-4 on page 93 shows a block diagram of the output compare unit. The lowercase "n" in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the "x" indicates output compare unit (A/B). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 13-4. Output Compare Unit, Block Diagram



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

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

For more information on how to access the 16-bit registers, see Section 13.10 “Accessing 16-bit Registers” on page 103.

13.6.1 Force Output Compare

In non-PWM waveform generation modes the match output of the comparator can be forced by writing a “1” to the force output compare (1x) bit. Forcing compare match does not set the OCF1x flag or reload/clear the timer, but the OC1x pin is updated as if a real compare match had occurred (the COM1x[1:0] bits settings define whether the OC1x pin is set, cleared, or toggled).

13.6.2 Compare Match Blocking by TCNT1 Write

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

13.6.3 Using the Output Compare Unit

Because writing TCNT1 in any operating mode blocks all compare matches for one timer clock cycle, risks arise from changing TCNT1 while using any of the output compare channels, regardless of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match is missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP is ignored and the counter continues to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is down-counting.

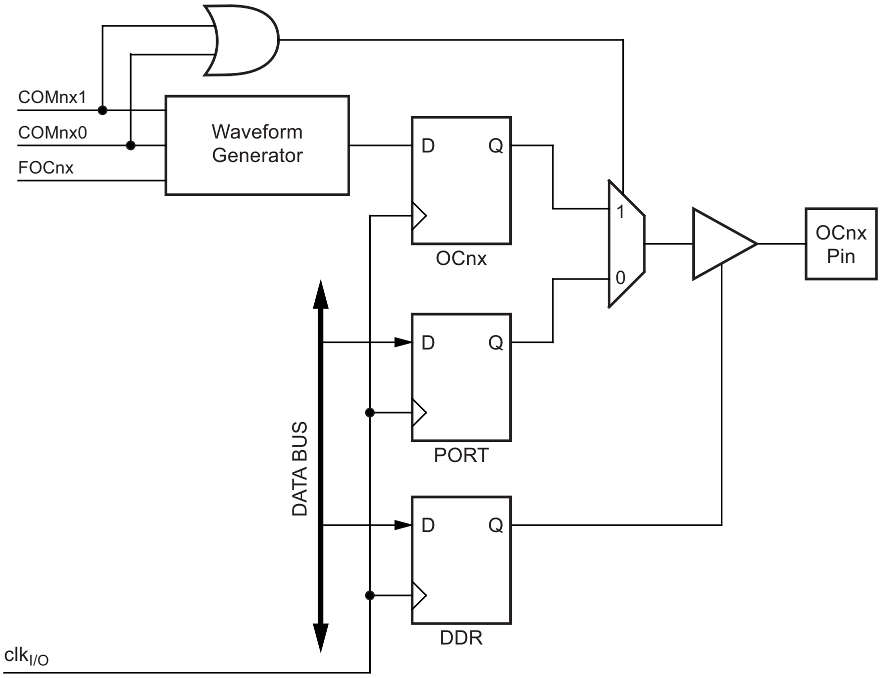
The setup of the OC1x should be performed before setting the data direction register for the port pin to output. The easiest way to set the OC1x value is to use the force output compare (1x) strobe bits in normal mode. The OC1x register retains its value even when changing between waveform generation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

Be aware that the COM1x[1:0] bits are not double-buffered together with the compare value. Changing the COM1x[1:0] bits takes effect immediately.

13.7 Compare Match Output Unit

The compare output mode (COM1x[1:0]) bits have two functions. The waveform generator uses the COM1x[1:0] bits for defining the output compare (OC1x) state at the next compare match. Secondly, the COM1x[1:0] bits control the OC1x pin output source. Figure 13-5 on page 94 shows a simplified schematic of the logic affected by the COM1x[1:0] 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) affected by the COM1x[1:0] bits are displayed. When referring to the OC1x state, the reference is for the internal OC1x register, not the OC1x pin. If a system reset occurs, the OC1x register is reset to “0”.

Figure 13-5. Compare Match Output Unit, Schematic (Non-PWM Mode)



The general I/O port function is overridden by the output compare (OC1x) from the waveform generator if either of the COM1x[1:0] bits are set. However, the OC1x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC1x pin (DDR_OC1x) must be set as the output before the OC1x value is visible on the pin. The port override function is generally independent of the waveform generation mode, but there are some exceptions. See Table 13-2 on page 106, Table 13-3 on page 106 and Table 13-4 on page 107 for more information.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x[1:0] bit settings are reserved for certain operating modes (see Section 13.11 “Register Description” on page 106).

The COM1x[1:0] bits have no effect on the input capture unit.

13.7.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM1x[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x[1:0] = 0 tells the waveform generator that no action on the OC1x register is to be performed on the next compare match. For compare output actions in the non-PWM modes, see Table 13-2 on page 106. For fast PWM mode, see Table 13-3 on page 106 and for phase correct and phase and frequency correct PWM, see Table 13-4 on page 107.

A change in the COM1x[1:0] bits state has an effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the 1x strobe bits.

13.8 Operating Modes

The operating mode, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM1[3:0]) and compare output mode (COM1x[1:0]) bits. The compare output mode bits do not affect the counting sequence while the waveform generation mode bits do. The COM1x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes, the COM1x[1:0] bits control whether the output should be set, cleared, or toggle at a compare match (see Section 13.7 “Compare Match Output Unit” on page 94). For detailed timing information, see Section 13.9 “Timer/Counter Timing Diagrams” on page 101.

13.8.1 Normal Mode

The simplest operating mode is the normal mode (WGM1[3:0] = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the Timer/Counter overflow flag (TOV1) is set in the same timer clock cycle when the TCNT1 becomes “0”. The TOV1 flag in this case behaves as a 17th 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 normal mode; a new counter value can be written at any time.

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

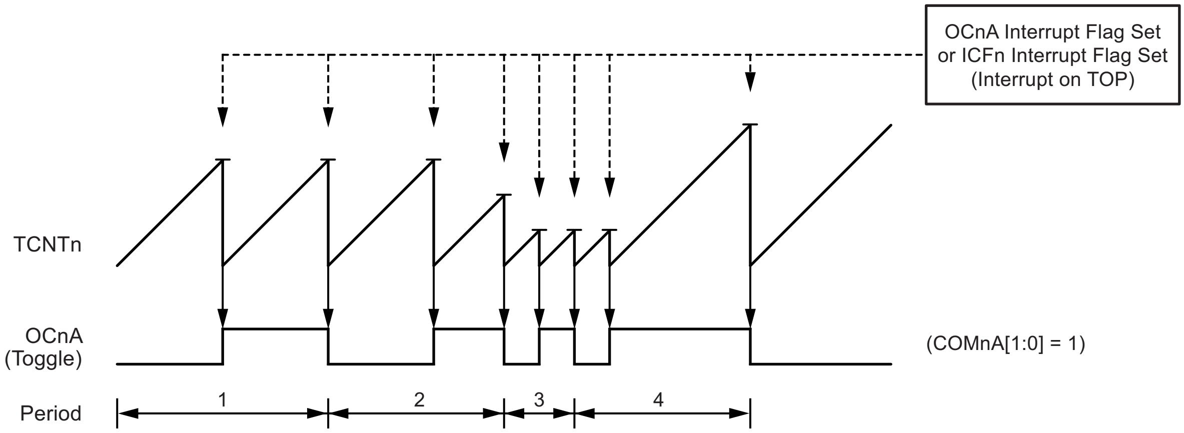
The output compare units can be used to generate interrupts at a given time. Using the output compare to generate waveforms in normal mode is not recommended because this occupies too much CPU time.

13.8.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM1[3:0] = 4 or 12) the OCR1A or ICR1 register is used to manipulate the counter resolution. In CTC mode the counter is cleared to “0” when the counter value (TCNT1) matches either the OCR1A (WGM1[3:0] = 4) or the ICR1 (WGM1[3:0] = 12). The OCR1A or ICR1 define the top value for the counter and therefore its resolution as well. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 13-6 on page 95. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 13-6. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by either using the OCF1A or ICF1 flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with no prescaler value or a low prescaler value must be done with care because the CTC mode does not have the double-buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter misses the compare match. The counter then has to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases, this feature is not desirable. An alternative is to use the fast PWM mode using OCR1A for defining TOP (WGM1[3:0] = 15) because the OCR1A is then double-buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logic level on each compare match by setting the compare output mode bits to toggle mode (COM1A[1:0] = 1). The OC1A value is not visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated has a maximum frequency of $_{1A}$ = $f_{clk\_I/O}/2$ when OCR1A is set to “0” (0x0000). The waveform frequency is defined by the following equation:

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

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

As for normal operating mode, the TOV1 flag is set in the same timer clock cycle in which the counter counts from MAX to 0x0000.

13.8.3 Fast PWM Mode

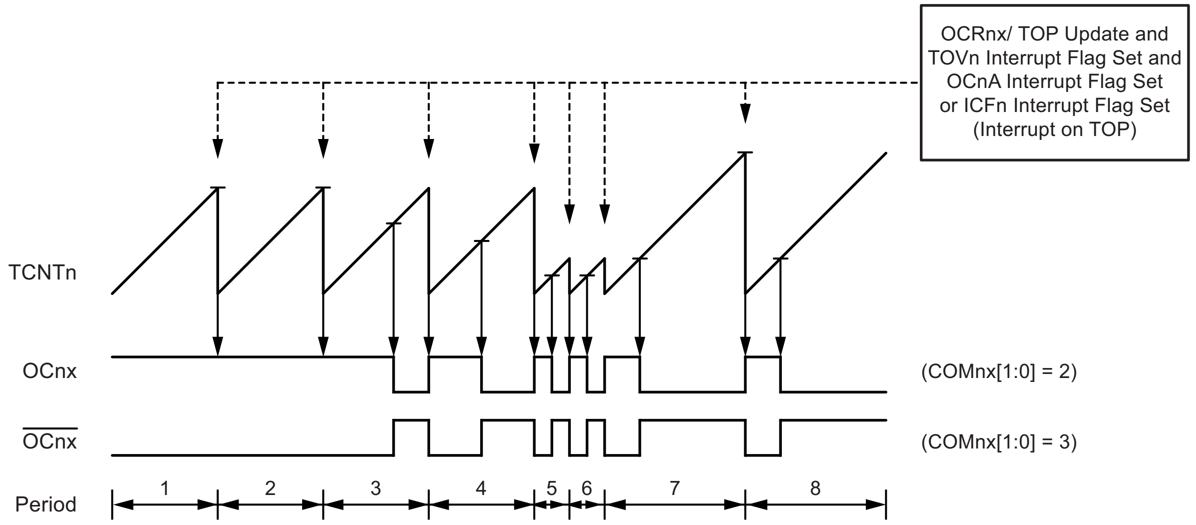
The fast pulse width modulation or fast PWM mode (WGM1[3:0] = 5, 6, 7, 14, or 15) provides a high-frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP, then restarts from BOTTOM. In non-inverting compare output mode the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x, and set at BOTTOM. In inverting compare output mode the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be two times higher than the phase correct and phase and frequency correct PWM modes that use dual-slope operation. A high frequency makes the fast PWM mode highly suitable for power regulation, rectification, and DAC applications and also allows external components (coils, capacitors) of small physical size, thus reducing overall system cost.

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

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

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM1[3:0] = 5, 6, or 7), the value in ICR1 (WGM1[3:0] = 14), or the value in OCR1A (WGM1[3:0] = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode can be seen in Figure 13-7. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is displayed in the timing diagram as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag is set when a compare match occurs.

Figure 13-7. Fast PWM Mode, Timing Diagram



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

When changing the TOP value, the program must ensure that the new TOP value is higher than or equal to the value of all the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to “0” when any of the OCR1x registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 register is not double-buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result is then that the counter misses the compare match at the TOP value. And so the counter has to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A register however, is double-buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written, the value written is placed in the OCR1A buffer register. The OCR1A compare register is then updated with the value in the buffer register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 flag is set.

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

In fast PWM mode the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x[1:0] bits to “2” produces a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x[1:0] to “3” (see Table 13-3 on page 106). The actual OC1x value is only visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1. The counter is cleared (changes from TOP to BOTTOM) by clearing (or setting) the OC1x register at the timer clock cycle.

The PWM frequency for the output can be calculated with this equation:

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

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

The extreme values for the OCR1x register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000), the output is a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP results in a constant high or low output (depending on the polarity of the output set by the COM1x[1:0] bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logic level on each compare match (COM1A[1:0] = 1). The waveform generated has a maximum frequency of $_{1A}$ = $f_{clk\_I/O}/2$ when OCR1A is set to “0” (0x0000). This feature is similar to the OC1A toggle in CTC mode except the double-buffer feature of the output compare unit is enabled in the fast PWM mode.

13.8.4 Phase Correct PWM Mode

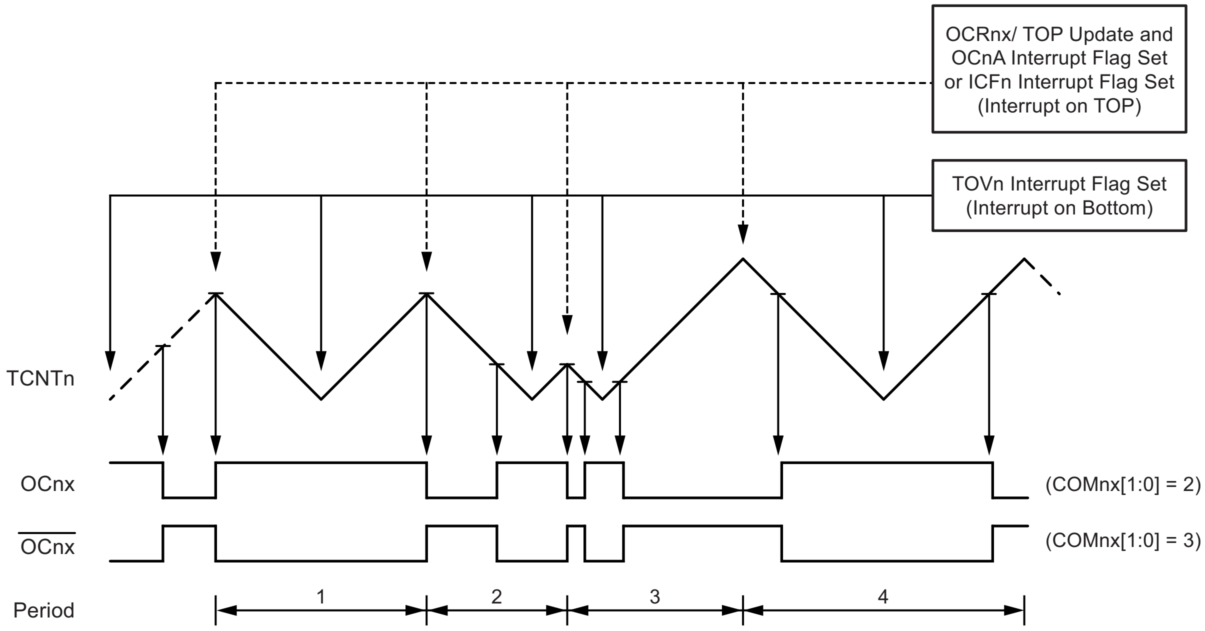
The phase correct pulse width modulation or phase correct PWM mode (WGM1[3:0] = 1, 2, 3, 10, or 11) provides a high-resolution phase correct PWM waveform generation option. The phase correct PWM mode is, the same way as the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting compare output mode the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting and set on the compare match while down-counting. In inverting output compare mode the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

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

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

In phase correct PWM mode the counter is incremented until the counter value matches any one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM1[3:0] = 1, 2, or 3), the value in ICR1 (WGM1[3:0] = 10), or the value in OCR1A (WGM1[3:0] = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value is equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode can be seen in Figure 13-8 on page 98. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is displayed in the timing diagram as a histogram to illustrate the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag is set when a compare match occurs.

Figure 13-8. Phase Correct PWM Mode, Timing Diagram


OCRnx/ TOP Update and OCnA Interrupt Flag Set or ICFn Interrupt Flag Set (Interrupt on TOP)
TOVn Interrupt Flag Set (Interrupt on Bottom)
TCNTn
OCnx
(COMnx[1:0] = 2)
$\overline{\text{OCnx}}$
(COMnx[1:0] = 3)
Period
1
2
3
4

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

When changing the TOP value, the program must ensure that the new TOP value is greater than or equal to the value of all the compare registers. If the TOP value is lower than any of the compare registers, a compare match never occurs between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to “0” when any of the OCR1x registers are written. As the third period shown in Figure 13-8 on page 98 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. This is due to the time of update of the OCR1x register. Because the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value while the length of the rising slope is determined by the new TOP value. When these two values differ, the two slopes of the period differ in length. The difference in length causes the unsymmetrical result on the output.

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

In phase correct PWM mode the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x[1:0] bits to “2” produces a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x[1:0] to “3” (see Table 13-4 on page 107). The actual OC1x value is only visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated with this equation:

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

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

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

13.8.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct pulse width modulation or phase and frequency correct PWM mode (WGM1[3:0] = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, the same way as the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting compare output mode, the output compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting and set on the compare match while down-counting. In inverting compare output mode the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

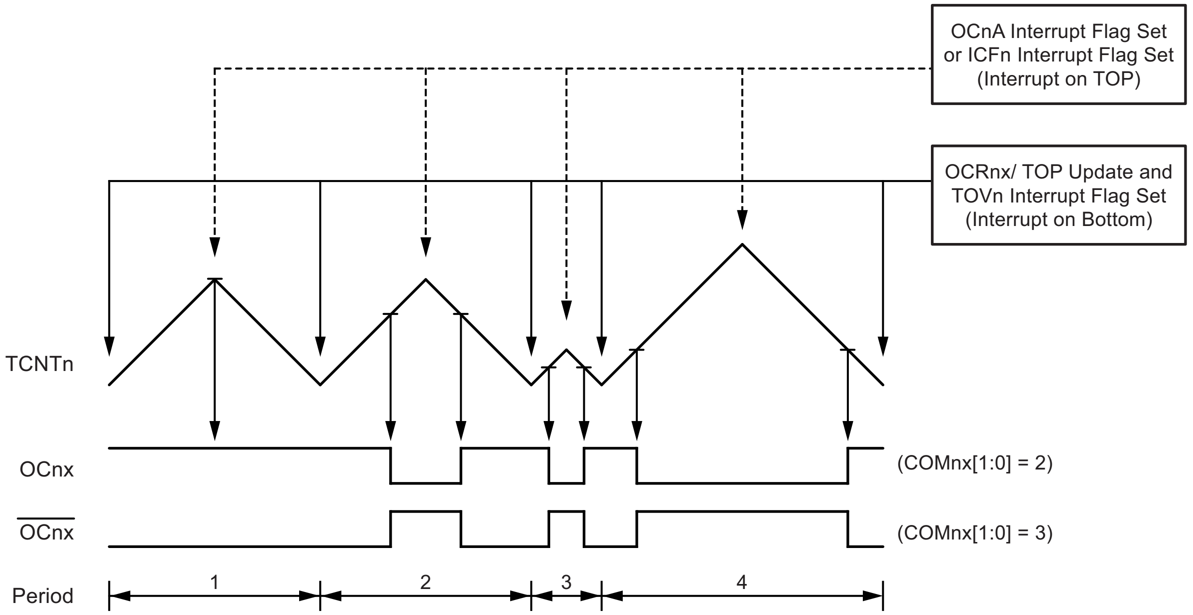
The main difference between the phase correct and the phase and frequency correct PWM mode is the time the OCR1x register is updated by the OCR1x buffer register (see Figure 13-8 on page 98 and Figure 13-9 on page 100).

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

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

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM1[3:0] = 8) or the value in OCR1A (WGM1[3:0] = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode can be seen in Figure 13-9 on page 100. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is displayed in the timing diagram as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag is set when a compare match occurs.

Figure 13-9. Phase and Frequency Correct PWM Mode, Timing Diagram



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

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

As shown in Figure 13-9 on page 100, the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Because the OCR1x registers are updated at BOTTOM, the length of the rising and the falling slopes is always equal. This gives symmetrical output pulses and is therefore frequency correct.

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

In phase and frequency correct PWM mode the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x[1:0] bits to “2” produces a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x[1:0] to “3” (see Table 13-4 on page 107). The actual OC1x value is only visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated with this equation:

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

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

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

13.9 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 and when the OCR1x register is updated with the OCR1x buffer value (only for modes utilizing double-buffering). Figure 13-10 shows a timing diagram for the setting of OCF1x.

Figure 13-10.Timer/Counter Timing Diagram, Setting of OCF1x, No Prescaling

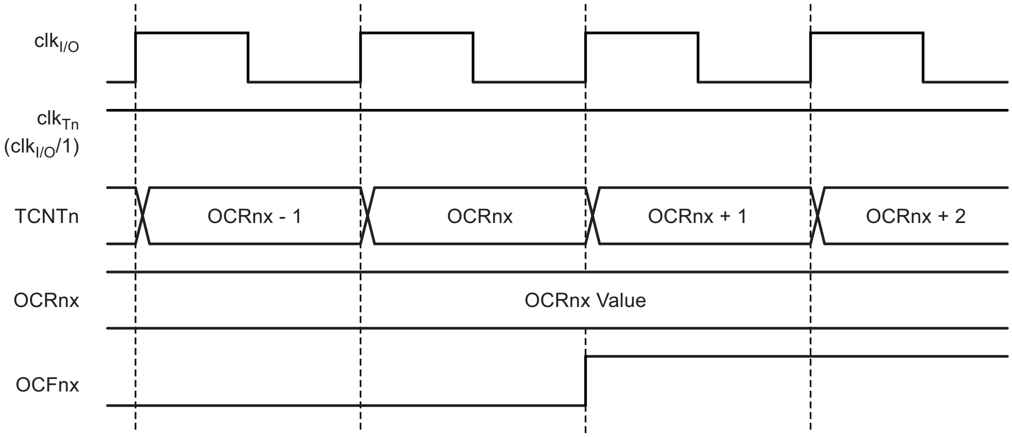


Figure 13-11 shows the same timing data but with the prescaler enabled.

Figure 13-11.Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}$/8)

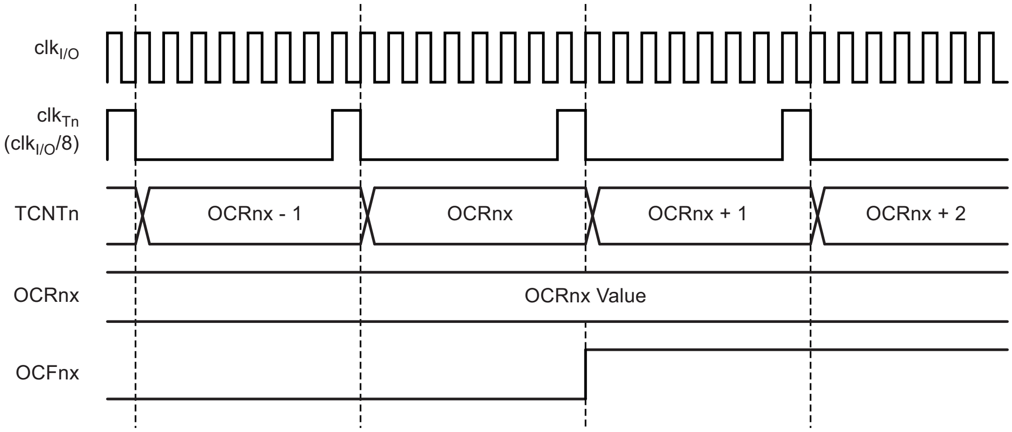


Figure 13-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode, the OCR1x register is updated at BOTTOM. The timing diagrams are the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1, etc. The same renaming applies for modes that set the TOV1 flag at BOTTOM.

Figure 13-12.Timer/Counter Timing Diagram, No Prescaling

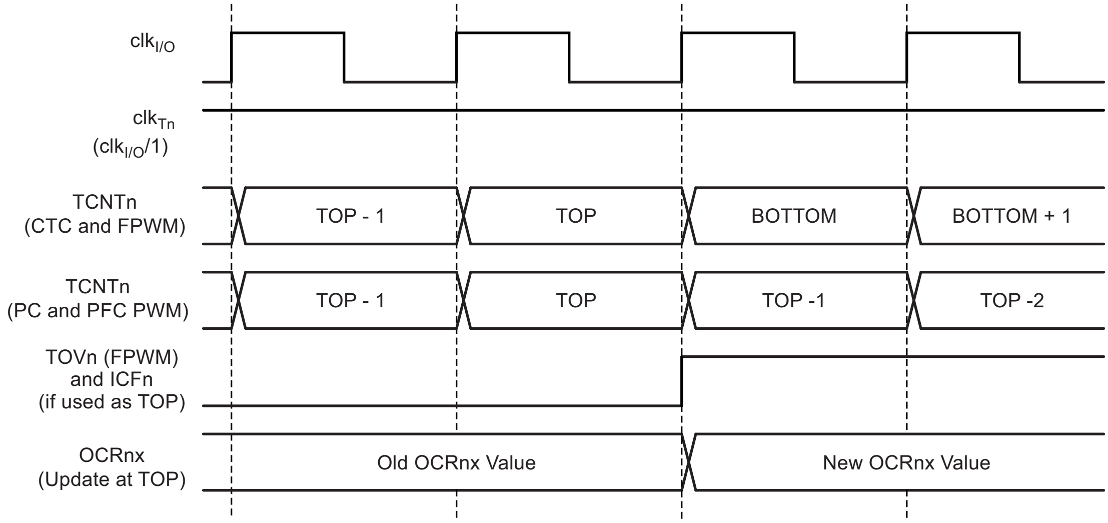
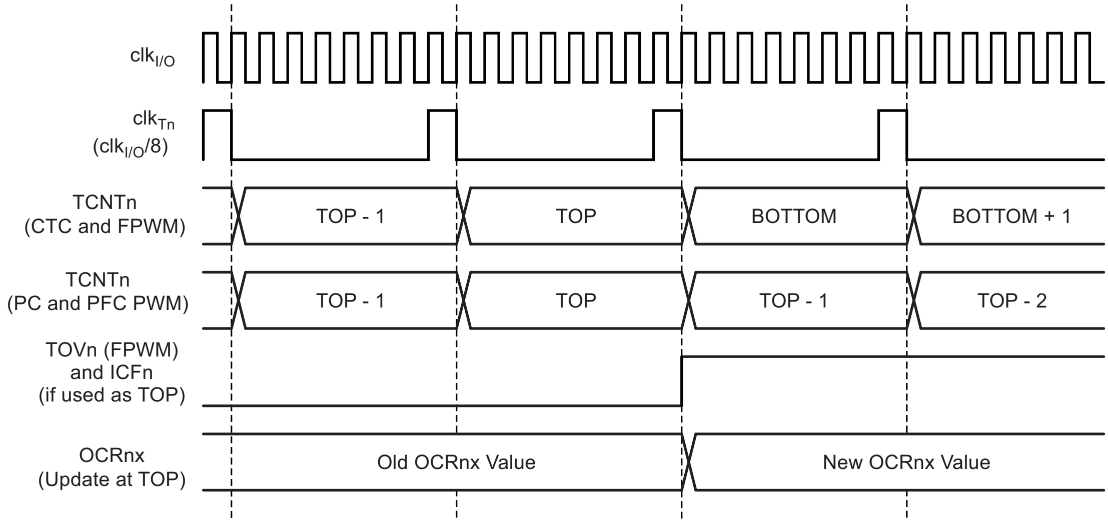


Figure 13-13 on page 102 shows the same timing data but with the prescaler enabled.

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



13.10 Accessing 16-bit Registers

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

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

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

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

Assembly Code Examples

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

C Code Examples

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

Note: See Section 4.2 “Code Examples” on page 7.

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

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

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

Assembly Code Example

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

C Code Example

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

Note: See Section 4.2 “Code Examples” on page 7.

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

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

Assembly Code Example

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

C Code Example

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

Note: See Section 4.2 “Code Examples” on page 7.

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

13.10.1 Reusing the Temporary High-byte Register

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

13.11 Register Description

13.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x72)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM1A[1:0]: Compare Output Mode for Channel A**

- **Bits 5:4 – COM1B[1:0]: Compare Output Mode for Channel B**

The COM1A[1:0] and COM1B[1:0] control the behavior of the output compare pins (OC1A and OC1B respectively). If one or both of the COM1A[1:0] bits are written to "1", the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B[1:0] bit are written to "1", the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that, to enable the output driver, the data direction register (DDR) bit corresponding to the OC1A or OC1B pin must be set.

When the OC1A or OC1B is connected to the pin, the function of the COM1x[1:0] bits depends on the WGM1[3:0] bits setting.

Table 13-2 shows COM1x[1:0] bit functionality when WGM1[3:0] bits are set to normal or CTC mode (non-PWM).

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

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

Table 13-3 shows COM1x[1:0] bit functionality when WGM1[3:0] bits are set to fast PWM mode.

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

COM1A1 COM1B1	COM1A0 COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected

0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected
		WGM13=1: Toggle OC1A on compare match, OC1B reserved

1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case, the compare match is ignored but the set or clear is done at BOTTOM (see Section 13.8.3 "Fast PWM Mode" on page 96).

Table 13-4 shows COM1x[1:0] bit functionality when WGM1[3:0] bits are set to phase correct or phase and frequency correct PWM mode.

Table 13-4. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM[1]

COM1A1 COM1B1	COM1A0 COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected

0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected
		WGM13=1: Toggle OC1A on compare match, OC1B reserved

1	0	Clear OC1A/OC1B on compare match when up-counting Set OC1A/OC1B on compare match when down-counting
1	1	Set OC1A/OC1B on compare match when up-counting Clear OC1A/OC1B on compare match when down-counting

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set (see Section 13.8.4 "Phase Correct PWM Mode" on page 98).

- **Bits 1:0 – WGM1[1:0]: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation is to be used (see Table 13-5 on page 107). Operating modes supported by the Timer/Counter unit are normal mode (counter), clear timer on compare match (CTC) mode, and three types of pulse width modulation (PWM) modes. (Section 13.8 "Operating Modes" on page 95).

Table 13-5. Waveform Generation Modes

Mode	WGM1[3:0]	Operating Mode	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0000	Normal	0xFFFF	Immediate	MAX
1	0001	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0010	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0011	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0100	CTC (clear timer on compare)	OCR1A	Immediate	MAX
5	0101	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0110	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0111	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1000	PWM, phase and freq. correct	ICR1	BOTTOM	BOTTOM
9	1001	PWM, phase and freq. correct	OCR1A	BOTTOM	BOTTOM
10	1010	PWM, phase correct	ICR1	TOP	BOTTOM
11	1011	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1100	CTC (clear timer on compare)	ICR1	Immediate	MAX
13	1101	Reserved	–	–	–
14	1110	Fast PWM	ICR1	TOP	TOP
15	1111	Fast PWM	OCR1A	TOP	TOP

13.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x71)	**ICNC1**	**ICES1**	**–**	**WGM13**	**WGM12**	**CS12**	**CS11**	**CS10**	**TCCR1B**
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

Setting this bit (to “1”) activates the input capture noise canceler. When the noise canceler is activated, the input from the input capture pin (ICP1) is filtered. The filter function requires four successive equal-valued samples of the ICP1 pin for changing its output. The input capture is therefore delayed by four oscillator cycles when the noise canceler is enabled.

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

This bit selects which edge on the input capture pin (ICP1) is used to trigger a capture event. When the ICES1 bit is written to “0”, a falling (negative) edge is used as trigger and when the ICES1 bit is written to “1”, a rising (positive) edge triggers the capture.

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

When the ICR1 is used as TOP value (see the description of the WGM1[3:0] bits located in the TCCR1A and the TCCR1B register), the ICP1 is disconnected and the input capture function thus disabled.

- **Bit 5 – Res: Reserved Bit**

This bit is a reserved bit in the Atmel$^{®}$ ATtiny1634 and always reads as “0”.

- **Bits 4:3 – WGM1[3:2]: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation is to be used (see Table 13-5 on page 107). Operating modes supported by the Timer/Counter unit are normal mode (counter), clear timer on compare match (CTC) mode, and three types of pulse width modulation (PWM) modes (see Section 13.8 “Operating Modes” on page 95).

- **Bits 2:0 – CS1[2:0]: Clock Select Bits**

The three clock select bits select the clock source to be used by the Timer/Counter (see Figure 13-10 and Figure 13-11).

Table 13-6. Clock Select Bit Description

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

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

13.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x70)	**FOC1A**	**FOC1B**	**–**	**–**	**–**	**–**	**–**	**–**	**TCCR1C**
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

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

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

The FOC1A/FOC1B bits are only active when the WGM1[3:0] bits specify a non-PWM mode. However, to ensure compatibility with future devices, while operating in PWM mode, these bits must be set to “0” when TCCR1B is written. When writing a logic one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1A/OC1B output is changed according to its COM1x[1:0] bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x[1:0] bits that determines the effect of the forced compare.

A FOC1A/FOC1B strobe does not generate any interrupt nor does it clear the timer in clear timer on compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as “0”.

13.11.4 TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x6F)	**TCNT1[15:8]**								**TCNT1H**
(0x6E)	**TCNT1[7:0]**								**TCNT1L**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access to the Timer/Counter unit 16-bit counter for read as well as write operations. To ensure that both high as well as 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 13.10 “Accessing 16-bit Registers” on page 103).

Modifying the counter (TCNT1) while the counter is running, poses the risk of missing a compare match between TCNT1 and one of the OCR1x registers.

Writing to the TCNT1 register blocks (removes) the compare match on the following timer clock for all compare units.

13.11.5 OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
(0x6D)	**OCR1A[15:8]**								**OCR1AH**
(0x6C)	**OCR1A[7:0]**								**OCR1AL**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.11.6 OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
(0x6B)	OCR1B[15:8]								OCR1BH
(0x6A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The output compare registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an output compare interrupt or to generate a waveform output on the OC1x pin.

The output compare registers are 16 bits in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high-byte register (TEMP). This temporary register is shared by all the other 16-bit registers (see Section 13.10 “Accessing 16-bit Registers” on page 103).

13.11.7 ICR1H and ICR1L – Input Capture Register 1

Bit	7	6	5	4	3	2	1	0	
(0x69)	ICR1[15:8]								ICR1H
(0x68)	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The input capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the analog comparator output for Timer/Counter1). The input capture can be used for defining the counter TOP value.

The input capture register is 16 bits in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high-byte register (TEMP). This temporary register is shared by all the other 16-bit registers (see Section 13.10 “Accessing 16-bit Registers” on page 103).

13.11.8 TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
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	

- **Bit 7 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to “1” and the I flag in the status register is set (interrupts globally enabled), the Timer/Counter1 overflow interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 47) is executed when the TOV1 flag located in TIFR is set.

- **Bit 6 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to “1” and the I flag in the status register is set (interrupts globally enabled), the Timer/Counter1 output compare A match interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 47) is executed when the OCF1A flag located in TIFR is set.

- **Bit 5 – OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable**

When this bit is written to “1” and the I flag in the status register is set (interrupts globally enabled), the Timer/Counter1 output compare B match interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 47) is executed when the OCF1B flag located in TIFR is set.

- **Bit 4 – Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny1634 and always reads as “0”.

- **Bit 3 – ICIE1: Timer/Counter1, Input Capture Interrupt Enable**

When this bit is written to “1” and the I flag in the status register is set (interrupts globally enabled), the Timer/Counter input capture interrupt is enabled. The corresponding interrupt vector (see Section 10. “Interrupts” on page 47) is executed when the ICF1 flag located in TIFR is set.

13.11.9 TIFR – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	**TOV1**	**OCF1B**	**OCF1A**	**–**	**ICF1**	**OCF0B**	**TOV0**	**OCF0A**	**TIFR**
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	

- **Bit 7 – TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag depends on the WGM1[3:0] bits setting. In normal and CTC modes the TOV1 flag is set when the timer overflows. See Table 13-5 on page 107 for the TOV1 flag behavior when using another WGM1[3:0] bit setting.

TOV1 is automatically cleared when the Timer/Counter1 overflow interrupt vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

- **Bit 6 – OCF1B: Timer/Counter1, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register B (OCR1B).

Note that a forced output compare (1B) strobe does not set the OCF1B flag.

OCF1B is automatically cleared when the output compare match B interrupt vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 5 – OCF1A: Timer/Counter1, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the output compare register A (OCR1A).

Note that a forced output compare (1A) strobe does not set the OCF1A flag.

OCF1A is automatically cleared when the output compare match A interrupt vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 4 – Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny1634 and always reads as “0”.

- **Bit 3 – ICF1: Timer/Counter1, Input Capture Flag**

This flag is set when a capture event occurs on the ICP1 pin. When the input capture register (ICR1) is set by the WGM1[3:0] to be used as the TOP value, the ICF1 flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the input capture interrupt vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

14. Timer/Counter Prescaler

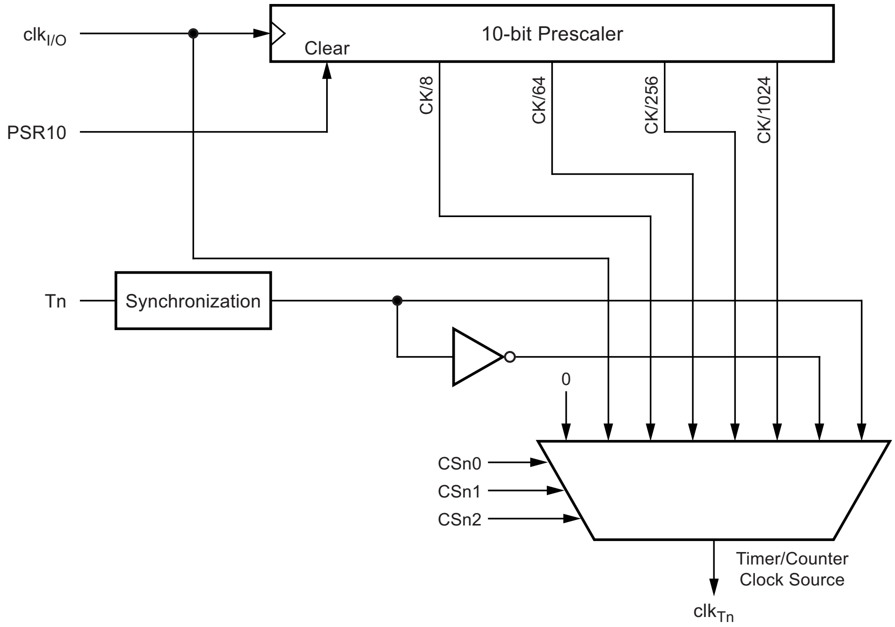
Timer/Counter0 and Timer/Counter1 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counters. Tn is used as a general name, where n = 0, 1.

The fastest Timer/Counter operation is achieved when the Timer/Counter is clocked directly by the system clock. Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock taps are:

- $f_{CLK\_I/O}/8$
- $f_{CLK\_I/O}/64$
- $f_{CLK\_I/O}/256$
- $f_{CLK\_I/O}/1024$

Figure 14-1 shows a block diagram of the Timer/Counter prescaler.

Figure 14-1. Prescaler for Timer/Counter0



Note: 1. The synchronization logic on the input pin (Tn) is shown in Figure 14-2 on page 113.

14.1 Prescaler Reset

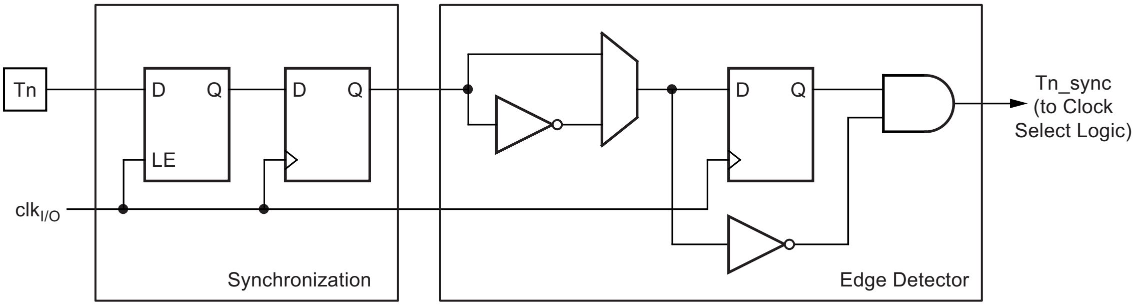
The prescaler is free running, i.e., it operates independently of the clock select logic of the Timer/Counter. Because the prescaler is not affected by the clock selection of Timer/Counters, the state of the prescaler has an effect where a prescaled clock is used. One example of prescaling artifacts occurs when the Timer/Counter is enabled while clocked by the prescaler. The time between Timer/Counter enable and the first count can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

To avoid prescaling artifacts, the prescaler reset can be used for synchronizing the Timer/Counter with program execution.

14.2 External Clock Source

An external clock source applied to the Tn pin can be used as the Timer/Counter clock (clk_{Tn}). The pin synchronization logic samples the Tn pin one time per system clock cycle. The synchronized (sampled) signal is then passed through the edge detector. Figure 14-2 shows a block diagram of the Tn synchronization and edge detector logic.

Figure 14-2. Tn Pin Sampling



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.

Depending on the clock select bits of the Timer/Counter, the edge detector generates one clk_{Tn} pulse for each positive or negative edge it detects.

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

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

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

An external clock source cannot be prescaled.

14.3 Register Description

14.3.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
(0x67)	**TSM**	–	–	–	–	–	–	**PSR10**	**GTCCR**
Read/Write	R/W	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

Writing the TSM bit to “1” activates the Timer/Counter synchronization mode. In this mode, the value that is written to the PSR10 bit is retained and therefore keeps the prescaler reset signal asserted.

This ensures that the Timer/Counter is stopped and can be configured without the risk of advancing during configuration. When the TSM bit is written to “0”, the PSR10 bit is cleared by the hardware and the Timer/Counter starts counting.

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

When this bit is “1”, the Timer/Counter prescaler is reset. Normally, this bit is cleared immediately by hardware, unless the TSM bit is set.

15. I^{2}C-Compatible, Two-wire Slave Interface

15.1 Features

- I^{2}C-compatible
- SMBus compatible (with reservations)
- 100kHz and 400kHz support at low system clock frequencies
- Slew-rate-limited output drivers
- Input filter provides noise suppression
- 7-bit and general call address recognition in hardware
- Address mask register for address masking or dual address match
- 10-bit addressing supported
- Optional software address recognition provides unlimited number of slave addresses
- Operates in all sleep modes, including power-down
- Slave arbitration allows support for SMBus address resolve protocol (ARP)

15.2 Overview

The two-wire interface (TWI) is a bidirectional, bus communication interface using two wires only. The TWI is I^{2}C-compatible and, with reservations, SMBus-compatible (see Section 15.3.10 “Compatibility with SMBus” on page 119).

A device connected to the bus must act as a master or slave. The master initiates a data transaction by addressing a slave on the bus and specifying whether it wants to transmit or receive data. One bus can have several masters, and an arbitration process handles priority if two or more masters try to transmit at the same time.

The TWI module in Atmel® ATtiny1634 implements slave functionality only. Lost arbitration, errors, collisions, and clock holds on the bus are detected in the hardware and indicated in separate status flags.

Both 7-bit and general address call recognition is implemented in hardware. 10-bit addressing is also supported. A dedicated address mask register can act as a second address match register or as a mask register for the slave address to match on a range of addresses. The slave logic continues to operate in all sleep modes, including power-down. This enables the slave to wake up from sleep on TWI address match. It is possible to disable the address matching and let this be handled in the software instead. This allows the slave to detect and respond to several addresses. Smart mode can be enabled to auto trigger operations and reduce software complexity.

The TWI module includes bus state logic that collects information to detect START and STOP conditions, bus collision, and bus errors. The bus state logic continues to operate in all sleep modes including power-down.

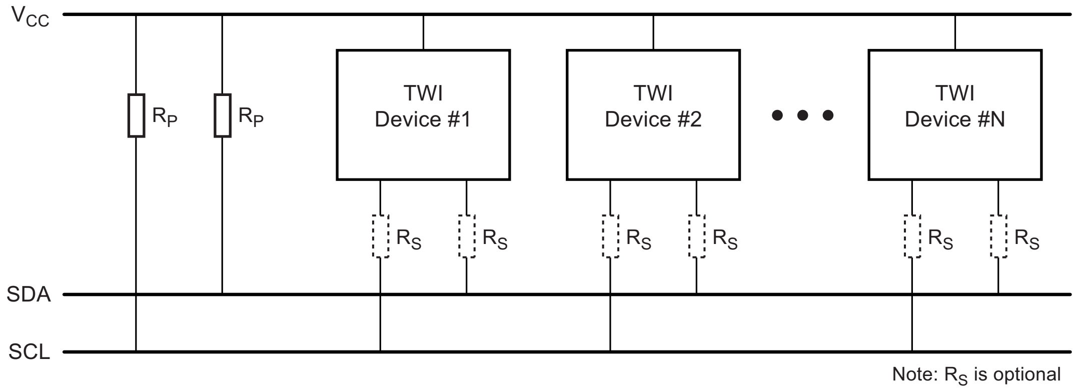
15.3 General TWI Bus Concepts

The two-wire interface (TWI) provides a simple two-wire bidirectional bus consisting of a serial clock line (SCL) and a serial data line (SDA). The two lines are open collector lines (wired-AND), and pull-up resistors (Rp) are the only external components needed to drive the bus. The pull-up resistors provide a high level on the lines when none of the connected devices are driving the bus. A constant current source can be used as an alternative to the pull-up resistors.

The TWI bus is a simple and efficient method for interconnecting multiple devices on a serial bus. A device connected to the bus can be a master or slave, with the master controlling the bus and all communication.

Figure 15-1 on page 115 illustrates the TWI bus topology.

Figure 15-1. TWI Bus Topology



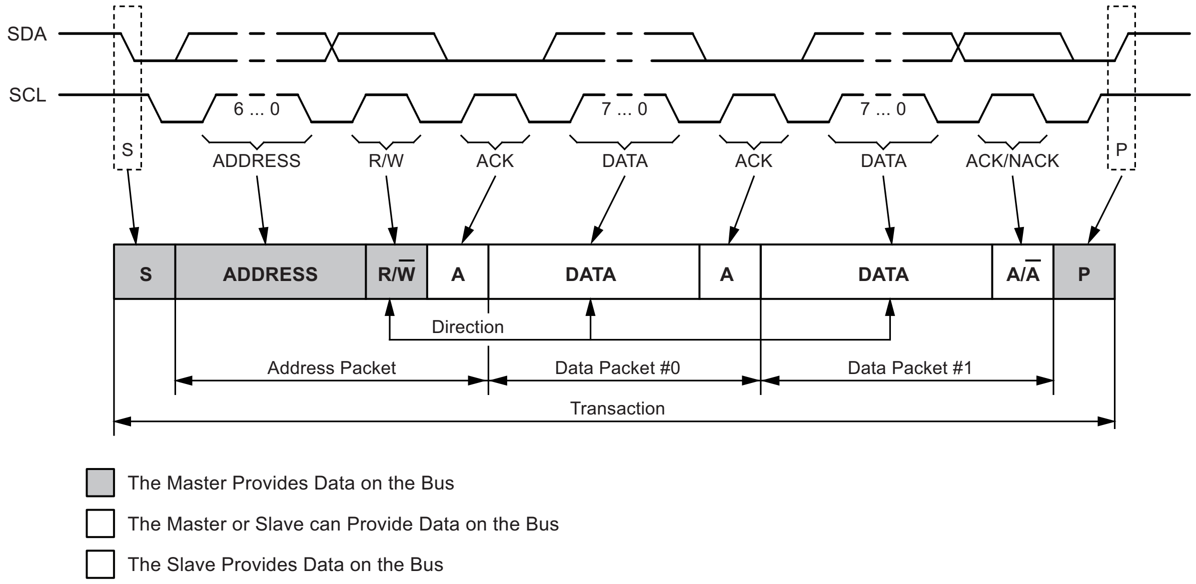
A unique address is assigned to all slave devices connected to the bus and the master uses this to address a slave and initiate a data transaction. 7-bit or 10-bit addressing can be used.

Several masters can be connected to the same bus. This is called a multimaster environment. An arbitration mechanism is provided for resolving bus ownership between masters because only one master device may own the bus at any given time.

A device can contain both master and slave logic and can emulate multiple slave devices by responding to more than one address.

Figure 15-2 shows a TWI transaction.

Figure 15-2. Basic TWI Transaction Diagram Topology



A master indicates the start of transaction by issuing a START condition (S) on the bus. An address packet with a slave address (ADDRESS) and an indication whether the master wishes to read or write data ($R/\overline{W}$) is then sent. After all data packets (DATA) are transferred, the master issues a STOP condition (P) on the bus to end the transaction. The receiver must acknowledge (A) or not-acknowledge ($\overline{A}$) each byte received.

The master provides the clock signal for the transaction, but a device connected to the bus is allowed to stretch the low level period of the clock to decrease the clock speed.

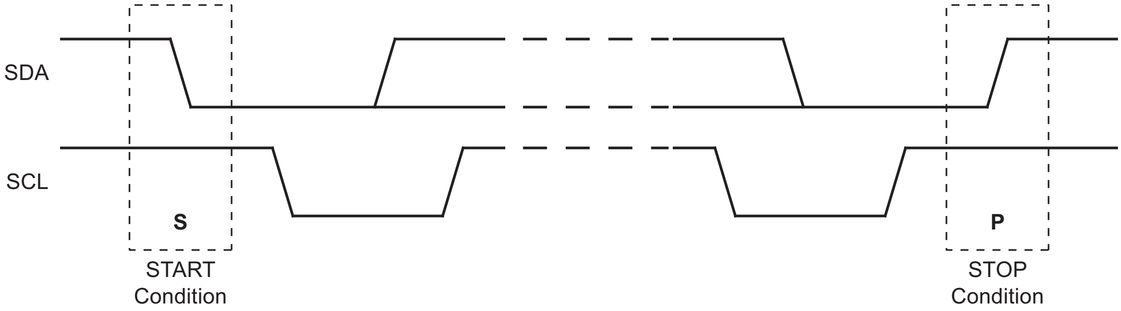
15.3.1 Electrical Characteristics

The TWI follows the electrical specifications and timing of I^2C and SMBus. For more information, see Section 25.6 “Two-wire Serial Interface” on page 219 and Section 15.3.10 “Compatibility with SMBus” on page 119.

15.3.2 START and STOP Conditions

Two unique bus conditions are used for marking the beginning (START) and end (STOP) of a transaction. The master issues a START condition (S) by indicating a high-to-low transition on the SDA line while the SCL line is kept high. The master completes the transaction by issuing a STOP condition (P), indicated by a low to high transition on the SDA line while the SCL line is kept high.

Figure 15-3. START and STOP Conditions

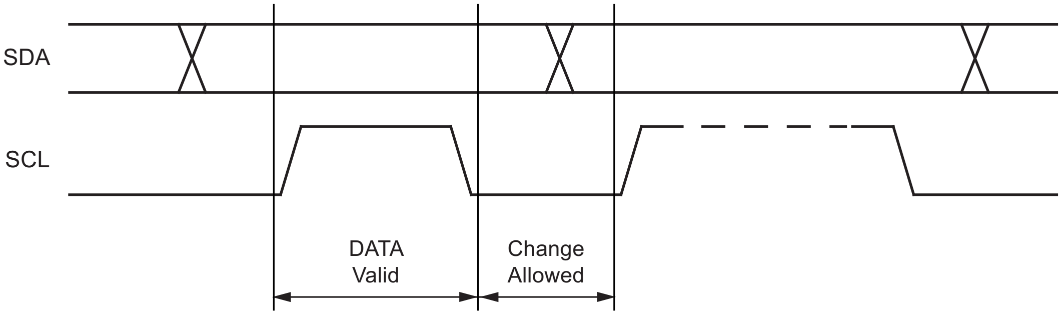


Multiple START conditions can be issued during a single transaction. A START condition not directly following a STOP condition are named a repeated START condition (Sr).

15.3.3 Bit Transfer

As illustrated by Figure 15-4, a bit transferred on the SDA line must be stable for the entire high period of the SCL line. The SDA value can thus only be changed during the low period of the clock. This is ensured in the hardware by the TWI module.

Figure 15-4. Data Validity



Combining bit transfers results in the formation of address and data packets. These packets consist of eight data bits (one byte) with the most significant bit transferred first, plus a single bit not-acknowledge (NACK) or acknowledge (ACK) response. The addressed device signals ACK by pulling the SCL line low and NACK by leaving the line SCL high during the ninth clock cycle.

15.3.4 Address Packet

After the START condition, a 7-bit address followed by a read/write ($R/\overline{W}$) bit is sent. This is always transmitted by the master. A slave recognizing its address acknowledges the address by pulling the data line low the next SCL cycle, while all other slaves should keep the TWI lines released and wait for the next START and address. The address packet consists of the combined 7-bit address, the $R/\overline{W}$ bit, and the acknowledge bit. Only one address packet for each START condition is given, even if 10-bit addressing is used.

The $R/\overline{W}$ specifies the direction of the transaction. If the $R/\overline{W}$ bit is low, it indicates a master write transaction and the master transmits its data after the slave has acknowledged its address. Conversely, for a master read operation the slave starts to transmit data after acknowledging its address.

15.3.5 Data Packet

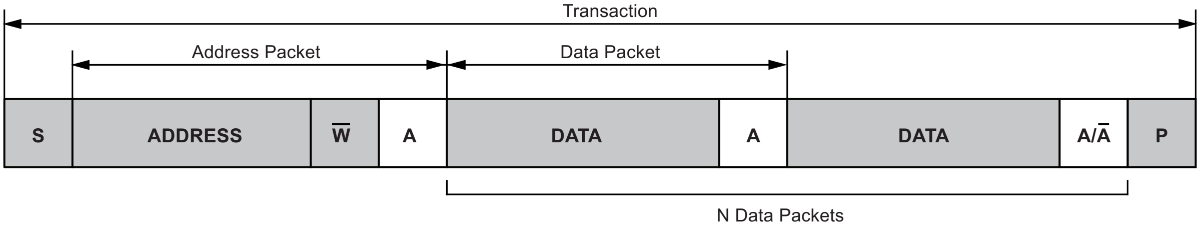
Data packets succeed an address packet or another data packet. All data packets are nine bits long, consisting of one data byte and an acknowledge bit. The direction bit in the previous address packet determines the direction in which the data is transferred.

15.3.6 Transaction

A transaction is the complete transfer from a START to a STOP condition including any repeated START conditions in between. The TWI standard defines three fundamental transaction modes: master write, master read, and combined transaction.

Figure 15-5 illustrates the master write transaction. The master initiates the transaction by issuing a START condition (S) followed by an address packet with direction bit set to “0” (ADDRESS+$\overline{\text{W}}$).

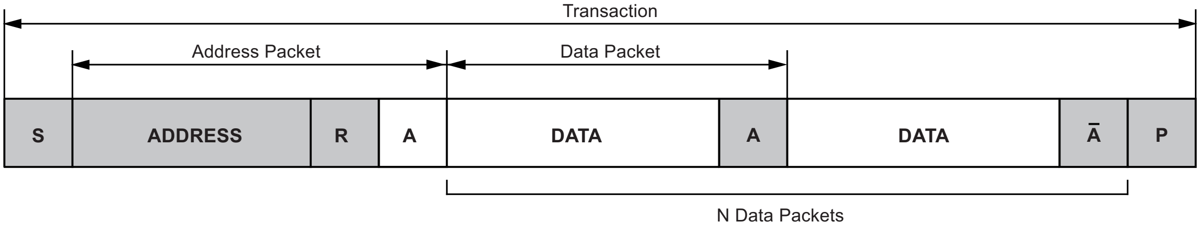
Figure 15-5. Master Write Transaction



Given that the slave acknowledges the address, the master can start transmitting data (DATA) and the slave will ACK or NACK (A/$\overline{\text{A}}$) each byte. If no data packets are to be transmitted, the master terminates the transaction by issuing a STOP condition (P) directly after the address packet. There are no limitations to the number of data packets that can be transferred. If the slave signals a NACK to the data, the master must assume that the slave cannot receive any more data and thus terminates the transaction.

Figure 15-6 illustrates the master read transaction. The master initiates the transaction by issuing a START condition followed by an address packet with the direction bit set to “1” (ADRESS+R). The addressed slave must acknowledge the address for the master to be allowed to continue the transaction.

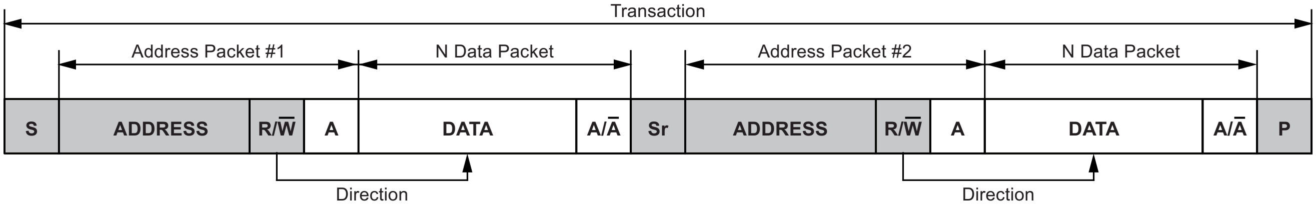
Figure 15-6. Master Read Transaction



Provided the slave acknowledges the address, the master can start receiving data from the slave. There are no limitations to the number of data packets that can be transferred. The slave transmits the data while the master signals ACK or NACK after each data byte. The master terminates the transfer with a NACK before issuing a STOP condition.

Figure 15-7 illustrates a combined transaction. A combined transaction consists of several read and write transactions separated by a repeated START conditions (Sr).

Figure 15-7. Combined Transaction

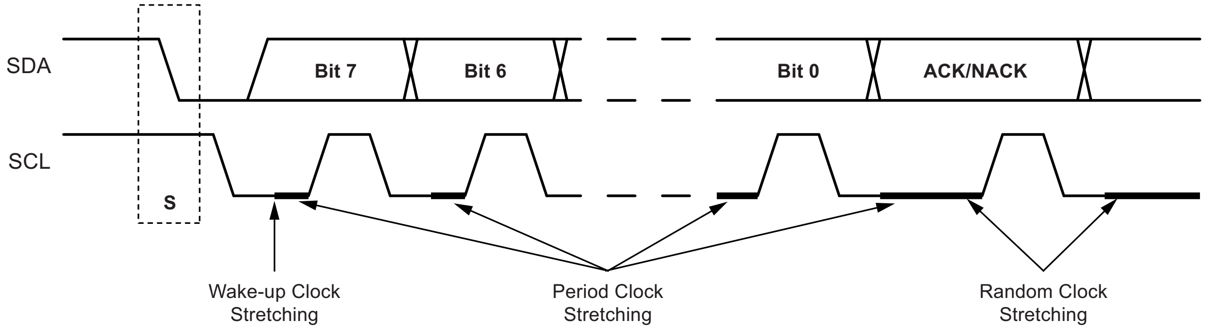


15.3.7 Clock and Clock Stretching

All devices connected to the bus are allowed to stretch the low period of the clock to slow down the overall clock frequency or to insert wait states while processing data. A device that needs to stretch the clock can do this by holding/forcing the SCL line low after it detects a low level on the line.

Three types of clock stretching can be defined as shown in Figure 15-8.

Figure 15-8. Clock Stretching



If the device is in sleep mode and a START condition is detected, the clock is stretched during the wake-up period for the device.

A slave device can slow down the bus frequency by stretching the clock periodically at the bit level. This allows the slave to run at a lower system clock frequency. However, the overall performance of the bus is correspondingly reduced. Both the master and slave device can randomly stretch the clock at the byte level before and after the ACK/NACK bit. This provides time to process incoming data, prepare outgoing data, or perform other time-critical tasks.

If the slave is stretching the clock, the master is forced into a wait state until the slave is ready and vice versa.

15.3.8 Arbitration

A master can only start a bus transaction if it has detected that the bus is idle. Because the TWI bus is a multimaster bus, two devices may initiate a transaction at the same time. This results in multiple masters owning the bus simultaneously. This is solved using an arbitration scheme where the master loses control of the bus if it is not able to transmit a high level on the SDA line. The masters which lose arbitration must then wait until the bus becomes idle (i.e., wait for a STOP condition) before attempting to reacquire bus ownership. Slave devices are not involved in the arbitration procedure.

Figure 15-9. TWI Arbitration

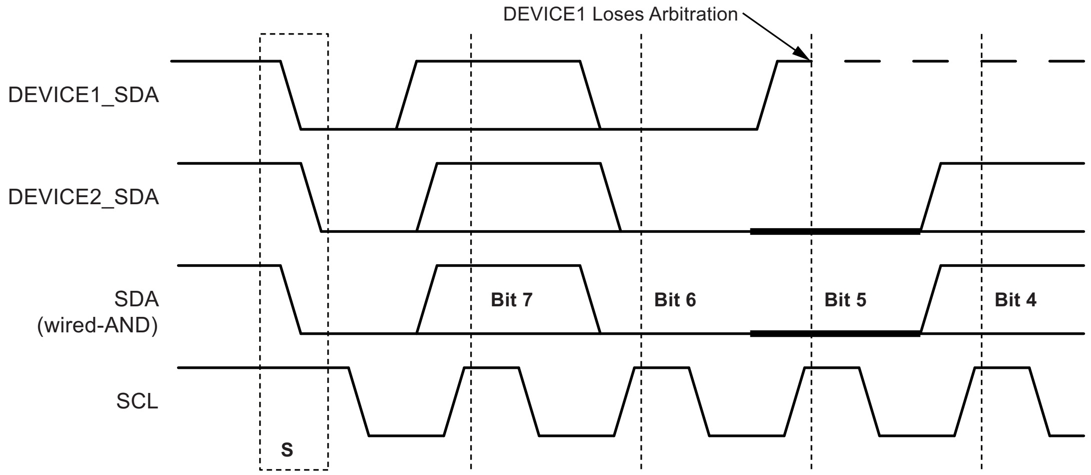


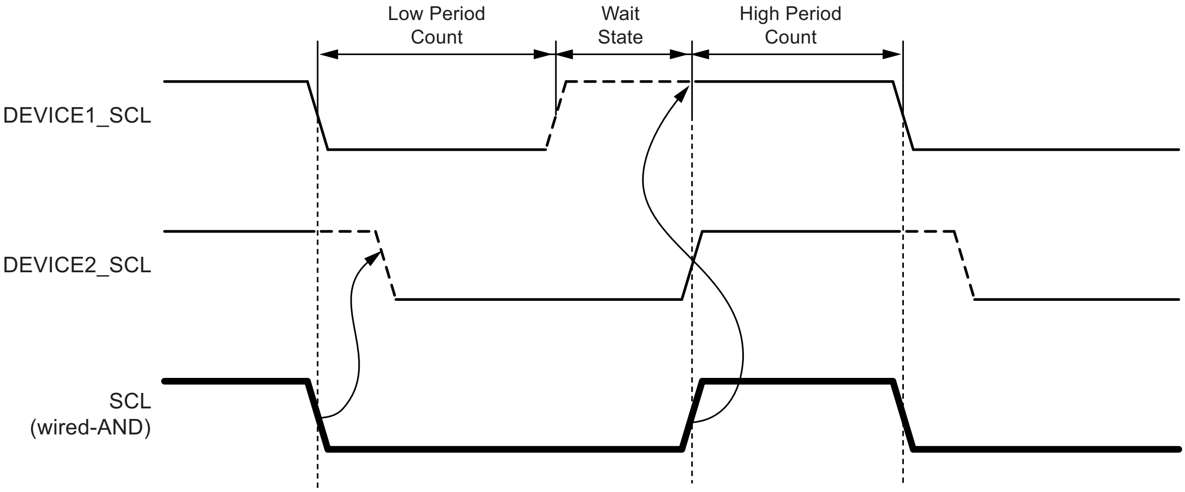
Figure 15-9 shows an example where two TWI masters are contending for bus ownership. Both devices are able to issue a START condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

Arbitration between a repeated START condition and a data bit, between a STOP condition and a data bit, or between a repeated START condition and STOP condition are not allowed and require special handling by the software.

15.3.9 Synchronization

A clock synchronization algorithm is necessary for solving situations where more than one master is trying to control the SCL line at the same time. The algorithm is based on the same principles used for clock stretching described in the previous section. Figure 15-10 shows an example where two masters are competing for control over the bus clock. The SCL line is the wired-AND result of the two masters' clock outputs.

Figure 15-10.Clock Synchronization



A high-to-low transition on the SCL line forces the line low for all masters on the bus and all masters start timing their low clock period. The timing length of the low clock period can vary among the masters. When a master (DEVICE1 in this case) has completed its low period, it releases the SCL line. However, the SCL line does not go high before all masters have released it. As a result, the SCL line is held low by the device with the longest low period (DEVICE2). Devices with shorter low periods must insert a wait state until the clock is released. All masters start their high period when the SCL line is released by all devices and has become high. The device which first completes its high period (DEVICE1) forces the clock line low and the procedure is then repeated. The result is that the device with the shortest clock period determines the high period while the low period of the clock is determined by the longest clock period.

15.3.10 Compatibility with SMBus

As with any other I^{2}C-compliant interface, there are known compatibility issues designers should be aware of before connecting a TWI device to SMBus devices. The following should be noted for use in SMBus environments:

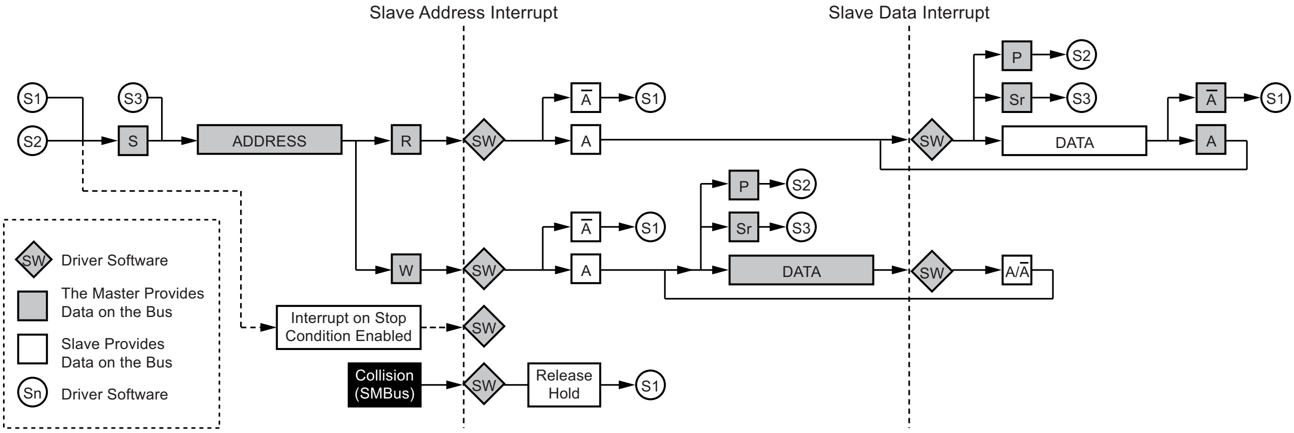
- All I/O pins of an AVR®, including those of the two-wire interface, have protection diodes to both supply voltage and ground (see Figure 11-1 on page 53). This is in contradiction to the requirements of the SMBus specifications. As a result, supply voltage must never be removed from the AVR or the protection diodes will pull the bus lines down. Power-down and sleep modes are not a problem, provided supply voltages remain.
- The data hold time of the TWI is lower than specified for SMBus.
- SMBus has a low speed limit, while I^{2}C does not. As a master in an SMBus environment, the AVR must ensure that the bus speed does not drop below specifications because lower bus speeds trigger time-outs in SMBus slaves. If the AVR is configured, a slave there may cause a bus lockup because the TWI module does not identify time-outs.

15.4 TWI Slave Operation

The TWI slave is byte-oriented with optional interrupts after each byte. There are separate interrupt flags for data interrupt and address/stop interrupt. Interrupt flags can be set to trigger the TWI interrupt, or be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, clock hold, collision, bus error, and read/write direction.

When an interrupt flag is set, the SCL line is forced low. This gives the slave time to respond or handle any data and most cases requires software interaction. Figure 15-11. shows the TWI-slave operation. The diamond-shaped symbols (SW) indicate where software interaction is required.

Figure 15-11.TWI Slave Operation



The number of interrupts generated is kept to a minimum by automatic handling of most conditions. Quick commands can be enabled to auto trigger operations and reduce software complexity.

Promiscuous mode can be enabled to allow the slave to respond to all received addresses.

15.4.1 Receiving Address Packets

When the TWI slave is properly configured, it waits for a START condition to be detected. When this happens, the successive address byte is received and checked by the address match logic and the slave will ACK the correct address. If the received address is not a match, the slave does not acknowledge the address and waits for a new START condition.

The slave address/stop interrupt flag is set when a START condition followed by a valid address packet is detected. A general call address also sets the interrupt flag.

A START condition immediately followed by a STOP condition is an illegal operation and the bus error flag is set when this happens.

The R/W direction flag reflects the direction bit received with the address. This can be read by software to determine the type of operation currently in progress.

Depending on the R/W direction bit and bus condition, one of four distinct cases (1 to 4) arises following the address packet. The different cases must be handled in software.

15.4.1.1 Case 1: Address Packet Accepted – Direction Bit Set

If the R/$\overline{W}$ direction flag is set, this indicates a master read operation. The SCL line is forced low, stretching the bus clock. If ACK is sent by the slave, the slave hardware sets the data interrupt flag indicating data is needed for transmit. If NACK is sent by the slave, the slave waits for a new START condition and address match.

15.4.1.2 Case 2: Address Packet Accepted – Direction Bit Cleared

If the R/$\overline{W}$ direction flag is cleared, this indicates a master write operation. The SCL line is forced low, stretching the bus clock. If ACK is sent by the slave, the slave waits for data to be received. Data, repeated START or STOP can be received after this. If NACK is indicated, the slave waits for a new START condition and address match.

15.4.1.3 Case 3: Collision

If the slave is not able to send a high level or NACK, the collision flag is set and it disables the data and acknowledge output from the slave logic. The clock hold is released. A START or repeated START condition is accepted.

15.4.1.4 Case 4: STOP Condition Received

Operation is the same as case 1 or 2 above with one exception: When the STOP condition is received, the slave address/stop flag is set, indicating that a STOP condition occurred, not an address match.

15.4.2 Receiving Data Packets

The slave knows when an address packet with the R/$\overline{W}$ direction bit cleared has been successfully received. After acknowledging this, the slave must be ready to receive data. When a data packet is received, the data interrupt flag is set and the slave must indicate ACK or NACK. After indicating a NACK, the slave must expect a STOP or repeated START condition.

15.4.3 Transmitting Data Packets

The slave knows when an address packet with the R/$\overline{W}$ direction bit set has been successfully received. It can then start sending data by writing to the slave data register. When a data packet transmission has been completed, the data interrupt flag is set. If the master indicates NACK, the slave must stop transmitting data and expect a STOP or repeated START condition.

15.5 Register Description

15.5.1 TWSCRA – TWI Slave Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x7F)	**TWSHE**	**–**	**TWDIE**	**TWASIE**	**TWEN**	**TWSIE**	**TWPME**	**TWSME**	**TWSCRA**
Read/Write	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TWSHE: TWI SDA Hold Time Enable**

When this bit is set, each negative transition of SCL triggers an additional internal delay before the device is allowed to change the SDA line. The added delay is approximately 50ns in length. This may be useful in SMBus systems.

- **Bit 6 – Res: Reserved Bit**

This bit is reserved and always reads as "0".

- **Bit 5 – TWDIE: TWI Data Interrupt Enable**

When this bit is set and interrupts are enabled, a TWI interrupt is generated when the data interrupt flag (TWDIF) in TWSSRA is set.

- **Bit 4 – TWASIE: TWI Address/Stop Interrupt Enable**

When this bit is set and interrupts are enabled, a TWI interrupt is generated when the address/stop interrupt flag (TWASIF) in TWSSRA is set.

- **Bit 3 – TWEN: Two-wire Interface Enable**

When this bit is set, the slave two-wire interface is enabled.

- **Bit 2 – TWSIE: TWI Stop Interrupt Enable**

Setting the stop interrupt enable (TWSIE) bit sets the TWASIF in the TWSSRA register when a STOP condition is detected.

- **Bit 1 – TWPME: TWI Promiscuous Mode Enable**

When this bit is set, the address match logic of the slave TWI responds to all received addresses. When this bit is cleared, the address match logic uses the TWSA register to determine which address to recognize as its own.

- **Bit 0 – TWSME: TWI Smart Mode Enable**

When this bit is set, the TWI slave enters smart mode where the acknowledge action is sent immediately after the TWI data register (TWSD) has been read. acknowledge action is defined by the TWAA bit in TWSCRB.

When this bit is cleared the acknowledge action is sent after TWCMDn bits in TWSCRB are written to 1X.

15.5.2 TWSCRB – TWI Slave Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x7E)	–	–	–	–	–	TWAA	TWCMD1	TWCMD0	TWSCRB
Read/Write	R	R	R	R	R	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved and always read as “0”.

- **Bit 2 – TWAA: TWI Acknowledge Action**

This bit defines the slave's acknowledge behavior after an address or data byte has been received from the master. Depending on the TWSME bit in TWSCRA, the acknowledge action is executed either when a valid command has been written to TWCMDn bits or when the data register has been read. Acknowledge action is also executed if the TWAIF flag is cleared after an address match or the TWDIF flag is cleared during master transmit (see Table 15-1).

Table 15-1. Acknowledge Action of TWI Slave

TWAA	Action	TWSME	When
0	Send ACK	0	When TWCMDn bits are written to “10” or “11”
		1	When TWSD is read
1	Send NACK	0	When TWCMDn bits are written to “10” or “11”
		1	When TWSD is read

- **Bits 1:0 – TWCMD[1:0]: TWI Command**

Writing these bits triggers the slave operation as defined in Table 15-2. The type of operation depends on the TWI slave interrupt flags, TWDIF and TWASIF. The acknowledge action is only executed when the slave receives data bytes or address bytes from the master.

Table 15-2. TWI Slave Command

TWCMD[1:0]	TWDIR	Operation
00	X	No action
01	X	Reserved
10	Used to complete transaction	
	0	Execute acknowledge action, then wait for any START (S/Sr) condition
	1	Wait for any START (S/Sr) condition
11	Used in response to an address byte (TWASIF is set)	
	0	Execute acknowledge action, then receive next byte
	1	Execute acknowledge action, then set TWDIF
	Used in response to a data byte (TWDIF is set)	
	0	Execute acknowledge action, then wait for next byte
	1	No action

Writing the TWCMD bits automatically releases the SCL line and clears the TWCH and slave interrupt flags.

TWAA and TWCMDn bits can be written at the same time. Acknowledge action is then executed before the command is triggered.

The TWCMDn bits are strobed and always read as “0”.

15.5.3 TWSSRA – TWI Slave Status register A

Bit	7	6	5	4	3	2	1	0	
(0x7D)	TWDIF	TWASIF	TWCH	TWRA	TWC	TWBE	TWDIR	TWAS	TWSSRA
Read/Write	R/W	R/W	R	R	R/W	R/W	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TWDIF: TWI Data Interrupt Flag**

This flag is set when a data byte has been successfully received, i.e., no bus errors or collisions have occurred during the operation. When this flag is set, the slave forces the SCL line low, stretching the TWI clock period. The SCL line is released by clearing the interrupt flags.

Writing a “1” to this bit clears the flag. This flag is also automatically cleared when writing a valid command to the TWCMDn bits in TWSCRB.

- **Bit 6 – TWASIF: TWI Address/Stop Interrupt Flag**

This flag is set when the slave detects that a valid address has been received or when a transmit collision has been detected. When this flag is set, the slave forces the SCL line low, stretching the TWI clock period. The SCL line is released by clearing the interrupt flags.

If TWASIE in TWSCRA is set, a STOP condition on the bus also sets TWASIF. A STOP condition sets the flag only if the system clock is faster than the minimum bus free time between STOP and START.

Writing a “1” to this bit clears the flag. This flag is also automatically cleared when writing a valid command to the TWCMDn bits in TWSCRB.

- **Bit 5 – TWCH: TWI Clock Hold**

This bit is set when the slave is holding the SCL line low.

This bit is read-only and set when TWDIF or TWASIF is set. The bit can be cleared indirectly by clearing the interrupt flags and releasing the SCL line.

- **Bit 4 – TWRA: TWI Receive Acknowledge**

This bit contains the most recently received acknowledge bit from the master.

This bit is read-only. When “0”, the most recent acknowledge bit from the master was ACK and, when “1”, the most recent acknowledge bit was NACK.

- **Bit 3 – TWC: TWI Collision**

This bit is set when the slave is unable to transfer a high data bit or a NACK bit. When a collision is detected, the slave commences normal operation and disables data and acknowledges output. No low values are shifted out onto the SDA line.

This bit is cleared by writing a “1” to it. The bit is also cleared automatically when a START or repeated START condition is detected.

- **Bit 2 – TWBE: TWI Bus Error**

This bit is set when an illegal bus condition has occurred during a transfer. An illegal bus condition occurs if a repeated START or STOP condition is detected and the number of bits from the previous START condition is not a multiple of nine.

This bit is cleared by writing a “1” to it.

- **Bit 1 – TWDIR: TWI Read/Write Direction**

This bit indicates the direction bit from the last address packet received from a master. When this bit is “1”, a master read operation is in progress. When the bit is “0”, a master write operation is in progress.

- **Bit 0 – TWAS: TWI Address or Stop**

This bit indicates why the TWASIF bit was last set. If “0”, a stop condition caused TWASIF to be set. If “1”, address detection caused TWASIF to be set.

15.5.4 TWSA – TWI Slave Address Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	**TWSA[7:0]**								**TWSA**
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 slave address register contains the TWI slave address used by the slave address match logic to determine if a master has addressed the slave. When using 7-bit or 10-bit address recognition mode, the high seven bits of the address register (TWSA[7:1]) represent the slave address. The least significant bit (TWSA0) is used for general call address recognition. Setting TWSA0 enables general call address recognition logic.

When using 10-bit addressing, the address match logic only supports hardware address recognition of the first byte of a 10-bit address. If TWSA[7:1] is set to “0b11110nn”, “nn” represents bits 9 and 8 of the slave address. The next byte received is then bits 7 to 0 in the 10-bit address, but this must be handled by software.

When the address match logic detects that a valid address byte has been received, the TWASIF is set and the TWDIR flag is updated.

If TWPME in TWSCRA is set, the address match logic responds to all addresses transmitted on the TWI bus. TWSA is not used in this mode.

15.5.5 TWSD – TWI Slave Data Register

Bit	7	6	5	4	3	2	1	0	
(0x7A)	**TWSD[7:0]**								**TWSD**
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 data register is used when transmitting and receiving data. During transfer, data is shifted from/to the TWSD register and to/from the bus. The data register thus cannot be accessed during byte transfers. This is protected in the hardware. The data register can only be accessed when the SCL line is held low by the slave, i.e., when TWCH is set.

When a master reads data from a slave, the data to be sent must be written to the TWSD register. The byte transfer is started when the master starts to clock the data byte from the slave. It is followed by the slave receiving the acknowledge bit from the master. The TWDIF and the TWCH bits are then set.

When a master writes data to a slave, the TWDIF and the TWCH flags are set when one byte has been received in the data register. If smart mode is enabled, reading the data register triggers the bus operation as set by the TWAA bit in TWSCRB.

Accessing TWSD clears the slave interrupt flags and the TWCH bit.

15.5.6 TWSAM – TWI Slave Address Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x7B)	**TWSAM[7:1]**							**TWAE**	**TWSAM**
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 7:1 – TWSAM[7:1]: TWI Address Mask**

Depending on the TWAE setting, these bits can act as a second address match register or an address mask register.

If TWAE is set to “0”, TWSAM can be loaded with a 7-bit slave address mask. Each bit in TWSAM can mask (disable) the corresponding address bit in the TWSA register. If the mask bit is “1”, the address match between the incoming address bit and the corresponding bit in TWSA is ignored. In other words, masked bits always match.

If TWAE is set to “1”, TWSAM can be loaded with a second slave address in addition to the TWSA register. In this mode, the slave matches at two unique addresses, one in TWSA and the other in TWSAM.

- **Bit 0 – TWAE: TWI Address Enable**

By default, this bit is “0” and the TWSAM bits act as an address mask to the TWSA register. If this bit is set to “1”, the slave address match logic responds to the two unique addresses in TWSA and TWSAM.

16. USI – Universal Serial Interface

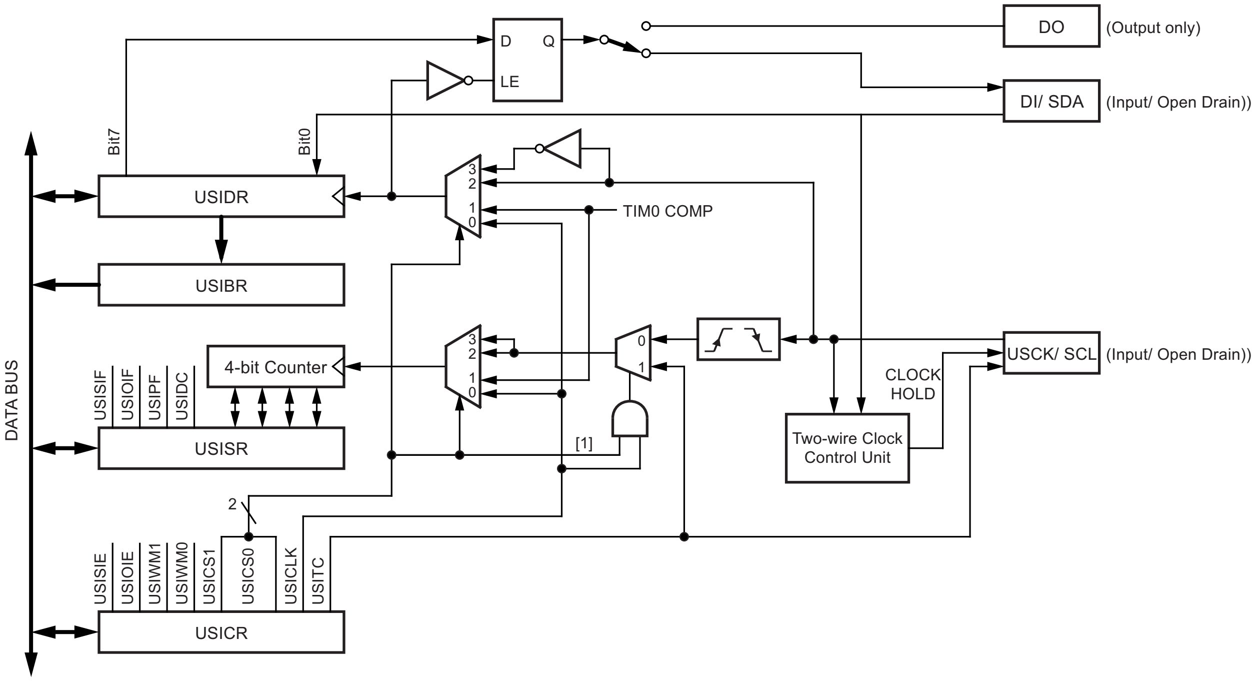
16.1 Features

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

16.2 Overview

The universal serial interface (USI) provides basic hardware resources 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 alone. The USI hardware also includes interrupts to minimize the processor load. A simplified block diagram of the USI is shown in Figure 16-1.

Figure 16-1. Universal Serial Interface, Block Diagram



Incoming and outgoing data is contained in the 8-bit USI data register (USIDR). It is directly accessible via the data bus, but a copy of the contents is also placed in the USI buffer register (USIBR) where it can be retrieved later. If USIDR is read directly, it must be done as quickly as possible to ensure that no data is lost.

Depending on the operating mode, the most significant bit of USIDR is connected to one of two output pins. A transparent latch between the output of USIDR and the output pin delays the change of data output to the opposite clock edge of the data input sampling.

Regardless of the operating mode, the serial input is always sampled from the data input (DI) pin.

The 4-bit counter can be read and written via the data bus and can generate an overflow interrupt. Both USIDR 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.

When an external clock source is selected, the counter counts both clock edges, meaning it registers the number of clock edges and not the number of data bits. The clock can be selected from three different sources:

- The USCK pin
- The Timer/Counter0 compare match
- The software

The two-wire clock control unit can be configured to generate an interrupt when a start condition has been detected on the two-wire bus. By holding the clock pin low after a start condition is detected or after the counter overflows, the unit can also be used to generate wait states.

The USI connects to I/O pins of the device as listed in Table 16-1. For I/O pin placement, see Section 1-1 “Pinout of Atmel ATtiny1634” on page 3.

Table 16-1. USI Connects to I/O Pins of the Device

Three-wire Mode	Two-wire Mode	Pin
Data input (DI)	Serial data (SDA)	PB1
Data output (DO)	–	PB2
Clock (USCK)	Serial clock (SCL)	PC1

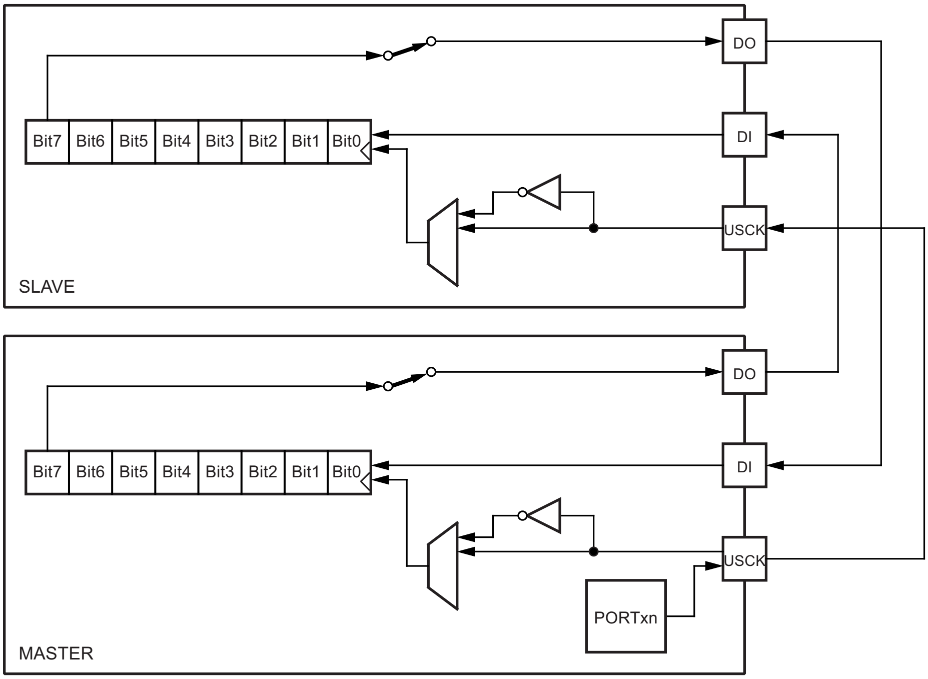
The device-specific I/O register and bit locations are listed in Section 16.7 “Register Descriptions” on page 133.

16.3 Three-wire Mode

The USI three-wire mode complies with the serial peripheral interface (SPI) mode 0 and 1 but does not have slave select (SS) pin functionality. However, this feature can be implemented in software if necessary. Pin names used in this mode are DI, DO, and USCK (see Table 16-1).

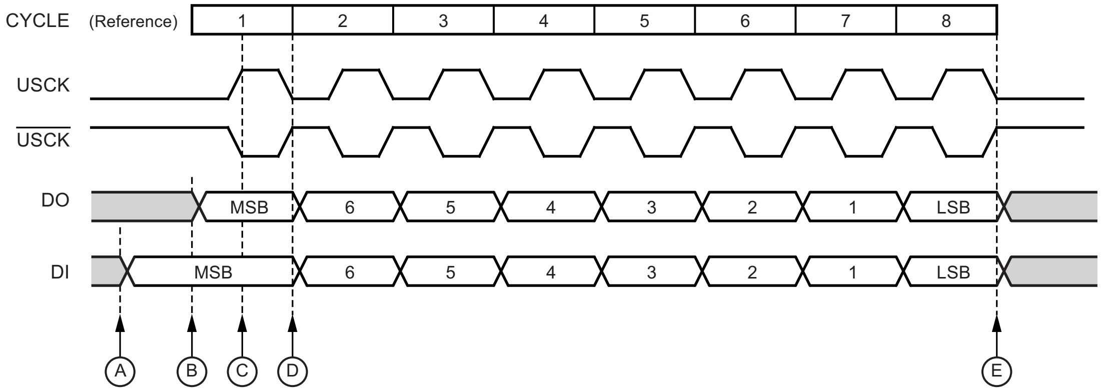
Figure 16-2 shows two USI units operating in three-wire mode, one as master and one as slave. The two USI data registers are interconnected in such a way that after eight USCK clocks, the data in each register has been interchanged. The same clock also increments the USI’s 4-bit counter. The counter overflow (interrupt) flag, or USIOIF, can thus be used to determine when a transfer has been completed. The clock is generated via the master device software by toggling the USCK pin or by writing a “1” to the USITC bit in USICR.

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



The three-wire mode timing is shown in Figure 16-3. At the top of the figure is a USCK cycle reference. One bit is shifted into the USI 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 (USIDR is shifted by one) at negative edges. In external clock mode 1 (USICS0 = 1) the opposite edges are used. In other words, data is sampled at negative and output is changed at positive edges. The USI clock modes correspond to the SPI data mode 0 and 1.

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



As shown in the timing diagram in Figure 16-3, a bus transfer involves the following steps:

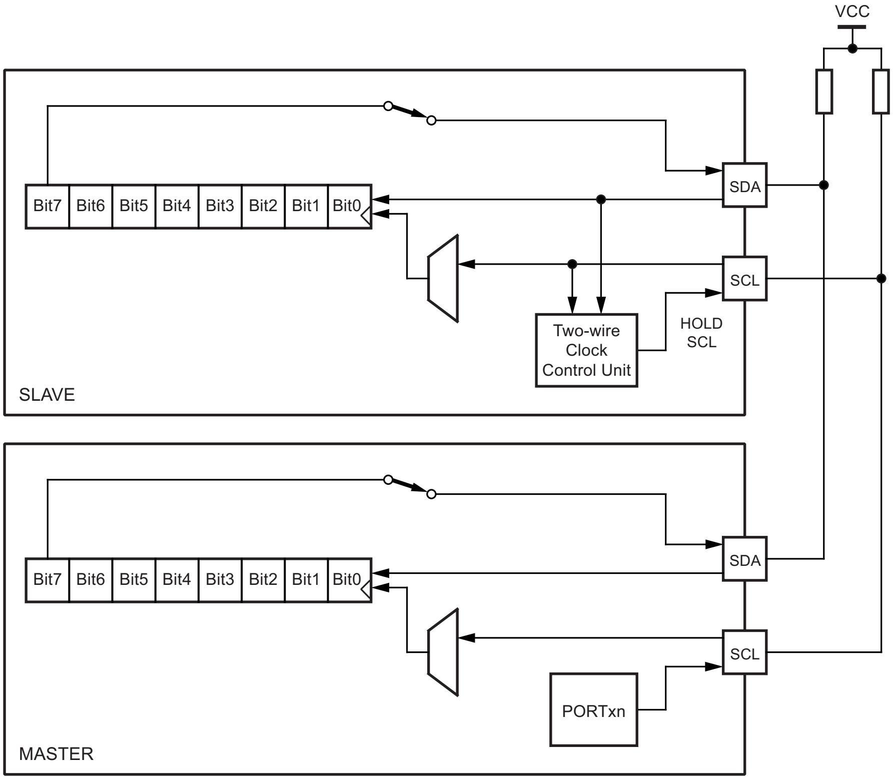
1. The slave and master devices set up their data outputs and, depending on the protocol used, enable their output drivers (mark A and B). The output is set up by writing the data to be transmitted to USIDR. The output is enabled by setting the bit corresponding to DO in the data direction register (DDRx) of the port. Note that there is not a preferred order of points A and B in the figure; both must be at least one-half USCK cycle before point C, where the data is sampled. This is in order to ensure that the data setup requirement is satisfied. The 4-bit counter is reset to “0”.
2. The master software generates a clock pulse by toggling the USCK line twice (C and D). The bit value on the data input pin (DI) 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 counts 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 overflows and indicates that the transfer has been completed. If USI buffer registers are not used, the data bytes that have been transferred must now be processed before a new transfer can be initiated. The overflow interrupt wakes up the processor if it is set to idle mode. Depending on the protocol used, the slave device can now set its output to high impedance.

16.4 Two-wire Mode

The USI two-wire mode complies with the Inter IC (TWI) bus protocol, but without slew rate limiting on outputs and without input noise filtering. Pin names used in this mode are SCL and SDA (see Table 16-1 on page 127).

Figure 16-4 on page 129 shows two USI units operating in two-wire mode, one as master and one as slave. Only the physical layer is shown because the system operation highly depends on 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. Only the slave uses the clock control unit.

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

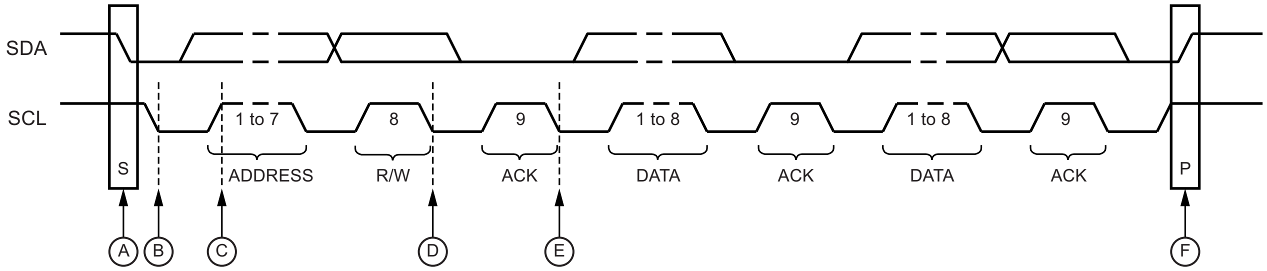


Clock generation must be implemented in the software; but the shift operation is done automatically in both devices. Note that clocking only on negative edges for shifting data is of practical use in this mode. The slave can insert wait states at the start or end of transfer by forcing the SCL clock low. This means the master must always check if the SCL line was actually released after it has generated a positive edge.

Because the clock also increments the counter, a counter overflow can be used to indicate that the transfer has been 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, such as the one used by the TWI bus, must be implemented to control data flow.

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



As shown in the timing diagram (Figure 16-5), a bus transfer involves the following steps:

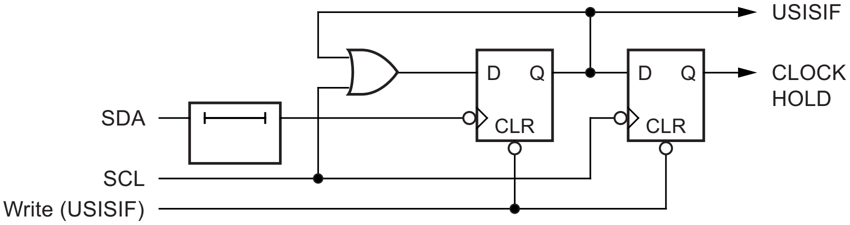
1. The start condition is generated by the master by forcing the SDA low line while keeping the SCL line high (A). SDA can be forced low either by writing a "0" to bit 7 of USIDR or by setting the corresponding bit in the PORT register to "0". Note that the data direction register (DDRx) bit must be set to "1" for the output to be enabled. The start detector logic of the slave device (see Figure 16-6 on page 130) detects the start condition and sets the USISIF flag. The flag can generate an interrupt where required.
2. The start detector holds the SCL line low after the master has forced a negative edge on this line (B). This allows the slave to wake up from sleep or complete other tasks before setting up USIDR to receive the address. This is done by clearing the start condition flag and resetting the counter.
3. The master sets the first bit to be transferred and releases the SCL line (C). The slave samples the data and shifts it into USIDR at the positive edge of the SCL clock.
4. After eight bits containing slave address and data direction (read or write) have been transferred, the slave counter overflows and the SCL line is forced low (D). If the slave is not the one the master has addressed, it releases the SCL line and waits for a new start condition.
5. When the slave is addressed, it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the USI counter register must be set to "14" before releasing SCL at (D)). The master or slave enables its output depending on the R/W bit. 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 acknowledgment (E).
6. Multiple bytes can now be transmitted—all in the same direction—until a stop condition is given by the master (F) or a new start condition is given.

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

16.4.1 Start Condition Detector

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

Figure 16-6. Start Condition Detector, Logic Diagram



The start condition detector works asynchronously and can therefore wake up the processor from power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. As a result, when using this feature, the oscillator start-up time (set by CKSEL fuses, see Section 7.2 "Clock Sources" on page 26) must also be considered. For more information, see the description of the USISIF bit in Section 19. "Analog Comparator" on page 169.

16.4.2 Clock Speed Considerations

Maximum frequency for SCL and SCK is $f_{CK}/2$. This is also the maximum data transmit and receive rate in both two- and three-wire mode. In two-wire slave mode the two-wire clock control unit holds the SCL low until the slave is ready to receive more data. This may reduce the current data rate in two-wire mode.

16.5 Alternative Use

The flexible design of the USI allows it to be used for other tasks when serial communication is not needed. Some examples are given below.

16.5.1 Half-Duplex Asynchronous Data Transfer

A more compact and higher performance UART can be implemented than through using software only by using the USI data register in three-wire mode.

16.5.2 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 increment the counter value.

16.5.3 12-bit Timer/Counter

Combining the 4-bit USI counter with one of the 8-bit Timer/Counters creates a 12-bit counter.

16.5.4 Edge-Triggered External Interrupt

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

16.5.5 Software Interrupt

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

16.6 Program Examples

16.6.1 Example: SPI Master Operation

The following code demonstrates how to use the USI module as an SPI master.

Assembly Code Example

```
SPITransfer:
       out    USIDR,r16
       ldi    r16,(1<<USIOIF)
       out    USISR,r16
       ldi    r17,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)

SPITransfer_loop:
       out    USICR,r17
       in     r16, USISR
       sbrs   r16, USIOIF
       rjmp   SPITransfer_loop
       in     r16,USIDR
       ret
```

See Section 4.2 “Code Examples” on page 7.

The code is size-optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in the data direction register (DDRx). The value stored in register r16 before 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 in register r16.

The first two instructions clear the USI counter overflow flag and the USI counter value. The next two instructions set three-wire mode, positive edge clock, count at USITC strobe, and toggle USCK. The transfer loop is then repeated 16 times.

16.6.2 Example: Full-Speed SPI Master

The following code demonstrates how to use the USI as an SPI master with maximum speed ($f_{SCK} = f_{CK}/2$).

Assembly Code Example

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

        out     USICR,r16 ;msB
        out     USICR,r17
        out     USICR,r16
        out     USICR,r17
        out     USICR,r16
        out     USICR,r17
        out     USICR,r16
        out     USICR,r17
        out     USICR,r16
        out     USICR,r17
        out     USICR,r16
        out     USICR,r17
        out     USICR,r16
        out     USICR,r17
        out     USICR,r16; LSB
        out     USICR,r17

        in      r16,USIDR
        ret
```

See Section 4.2 “Code Examples” on page 7.

16.6.3 Example: SPI Slave Operation

The following code demonstrates how to use the USI module as an SPI slave.

Assembly Code Example

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

See Section 4.2 “Code Examples” on page 7.

The code is size-optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in the port data direction register. The value stored in register r16 before 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 in the register r16.

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

16.7 Register Descriptions

16.7.1 USICR – USI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2A (0x4A)	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 USI control register includes bits for interrupt enable, setting the wire mode, selecting the clock, and clock strobe.

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

Setting this bit to “1” enables the start condition detector interrupt. If there is a pending interrupt and USISIE and the global interrupt enable flag are set to “1”, the interrupt is immediately executed (for more information, see the USISIF bit description on 135).

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

Setting this bit to “1” enables the counter overflow interrupt. If there is a pending interrupt and USIOIE and the global interrupt enable flag are set to “1”, the interrupt is immediately executed (for more information, see the USIOIF bit description on 135).

- **Bits 5:4 – USIWM[1:0]: Wire Mode**

These bits set the type of wire mode to be used as shown in Table 16-2 on page 133.

Basically, only the function of the outputs is affected by these bits. Data and clock inputs are not affected by the mode selected and 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.

Table 16-2. Relationship between USIWM[1:0] and USI Operation

USIWM1	USIWM0	Description
0	0	Outputs, clock hold, and start detector disabled. Port pins operate normally.
0	1	Three-wire mode. Uses DO, DI, and USCK pins. The data output (DO) pin overrides the corresponding bit in the PORTA register. However, the corresponding DDRA bit still controls the data direction. When the port pin is set as input, the pin pull-up is controlled by the PORTA bit. The data input (DI) and serial clock (USCK) pins do not affect normal port operation. When operating as master, clock pulses are software generated by toggling the PORTA 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 bidirectional and use open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDRA register. When the output driver is enabled for the SDA pin, the output driver forces the line SDA low if the output of the USI data register or the corresponding bit in the PORTA register is “0”. Otherwise, the SDA line is not driven (i.e., it is released). When the SCL pin output driver is enabled, the SCL line is forced low if the corresponding bit in the PORTA register is “0”, or forced low by the start detector. Otherwise, the SCL line is not driven. The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the start condition flag (USISIF) releases the line. The SDA and SCL pin inputs are not affected by enabling this mode. Pull-ups on the SDA and SCL port pin are disabled in two-wire mode.
1	1	Two-wire mode. Uses SDA and SCL pins. Same operation as in two-wire mode above, except that the SCL line is also held low when a counter overflow occurs and 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 among operating modes.

- **Bits 3:2 – USICS[1:0]: 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 an external clock source (USCK/SCL). When software strobe or Timer/Counter0 compare match clock option is selected, the output latch is transparent and the output is therefore changed immediately.

Clearing the USICS[1:0] bits enables the software strobe option. When using this option, writing a "1" 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, instead it selects between external clocking or software clocking by the USITC strobe bit.

Table 16-3 shows the relationship between the USICS[1:0] and USICLK setting and clock source used for the USI data register and the 4-bit counter.

Table 16-3. Relationship between the USICS1:0 and USICLK Setting

USICS1	USICS0	USICLK	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 A	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 "1" to this bit location strobes the USI data register to shift one step and the counter to increment by one, provided that the software clock strobe option has been selected by writing USICS[1:0] bits to "0". The output immediately changes when the clock strobe is executed, i.e., during the same instruction cycle. The value shifted into the USI data register is sampled in the previous instruction cycle.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a clock select register. In this case, setting the USICLK bit selects the USITC strobe bit as a clock source for the 4-bit counter (see Table 16-3).

The bit is read as "0".

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

Writing a "1" to this bit location toggles the USCK/SCL value either from "0" to "1", or from "1" to "0". The toggling does not depend on the setting in the data direction register; however, if the PORT value is to be shown on the pin, the corresponding DDR pin must be set as output (to "1"). This feature allows easy clock generation when implementing master devices.

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

The bit reads as "0".

16.7.2 USISR – USI Status register

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	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 "1") when a start condition has been detected. When three-wire mode or output disable mode has been selected, any edge on the SCK pin sets the flag.

If the USISIE bit in USICR and the global interrupt enable flag are set, an interrupt is generated when this flag is set. The flag is only cleared by writing a logic one to the USISIF bit. Clearing this bit releases the start detection hold of USCL in two-wire mode.

A start condition interrupt wakes up the processor from all sleep modes.

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

This flag is set ("1") when the 4-bit counter overflows (i.e., at the transition from "15" to "0"). If the USIOIE bit in USICR and the global interrupt enable flag are set, an interrupt is also generated when the flag is set. The flag is only cleared if a "1" is written to the USIOIF bit. Clearing this bit releases the counter overflow hold of SCL in two-wire mode.

A counter overflow interrupt wakes up the processor from idle sleep mode.

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

When two-wire mode is selected, the USIPF flag is set ("1") when a stop condition has been detected. This flag is cleared by writing a "1" 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 logic 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 – USICNT[3:0]: Counter Value**

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

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 on the setting of the USICS[1:0] 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 choosing an external clock source (USICS1 = 1) and writing a "1" to the USICLK bit.

Note that even when no wire mode is selected (USIWM[1:0] = 0), the external clock input (USCK/SCL) can still be used by the counter.

16.7.3 USIDR – USI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	**MSB**							**LSB**	**USIDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The USI data register can be accessed directly but a copy of the data can also be found in the USI buffer register.

Depending on the USICS[1:0] bits of the USI control register, a (left) shift operation may be performed. The shift operation can be synchronized to an external clock edge, to a Timer/Counter0 compare match, or directly to software via the USICLK bit. If a serial clock occurs in the same cycle the register is written, the register contains the value written and no shift is performed.

Note that even when no wire mode is selected (USIWM[1:0] = 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 (DO or SDA, depending on the wire mode) is connected via the output latch to the most significant bit (bit 7) of the USI data register. The output latch ensures that data input is sampled and data output is changed on opposite clock edges. The latch is open (transparent) during the first half of a serial clock cycle while an external clock source is selected (USICS1 = 1) and constantly open when an internal clock source is used (USICS1 = 0). The output is changed immediately when a new MSB is written as long as the latch is open.

Note that the data direction register bit corresponding to the output pin must be set to “1” in order to enable data output from the USI data register.

16.7.4 USIBR – USI Buffer Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	**MSB**							**LSB**	**USIBR**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Instead of reading data from the USI data register, the USI buffer register can be used. This makes controlling the USI less time-critical and gives the CPU more time to handle other program tasks. USI flags are set in a way similar to when reading the USIDR register.

The content of the USI data register is loaded to the USI buffer register when the transfer has been completed.

17. USART (USART0 and USART1)

17.1 Features

- Full duplex operation (independent serial receive and transmit registers)
- Asynchronous or synchronous operation
- Master- or slave-clocked synchronous operation
- High-resolution baud rate generator
- Supports serial frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits
- Odd or even parity generation and parity check supported by hardware
- Data overrun detection
- Framing error detection
- Noise filtering includes false start bit detection and digital low pass filter
- Three separate interrupts on TX complete, TX data register empty, and RX complete
- Multiprocessor communication mode
- Double-speed asynchronous communication mode
- Start frame detection

17.2 USART0 and USART1

The Atmel® ATtiny1634 has two universal synchronous and asynchronous serial receiver and transmitters—USART0 and USART1.

The functionality for all USARTs is described below; most register and bit references in this section are written in general form. A lowercase “n” replaces the USART number.

As shown in Section 27. “Register Summary” on page 245, USART0 and USART1 have different I/O registers.

17.3 Overview

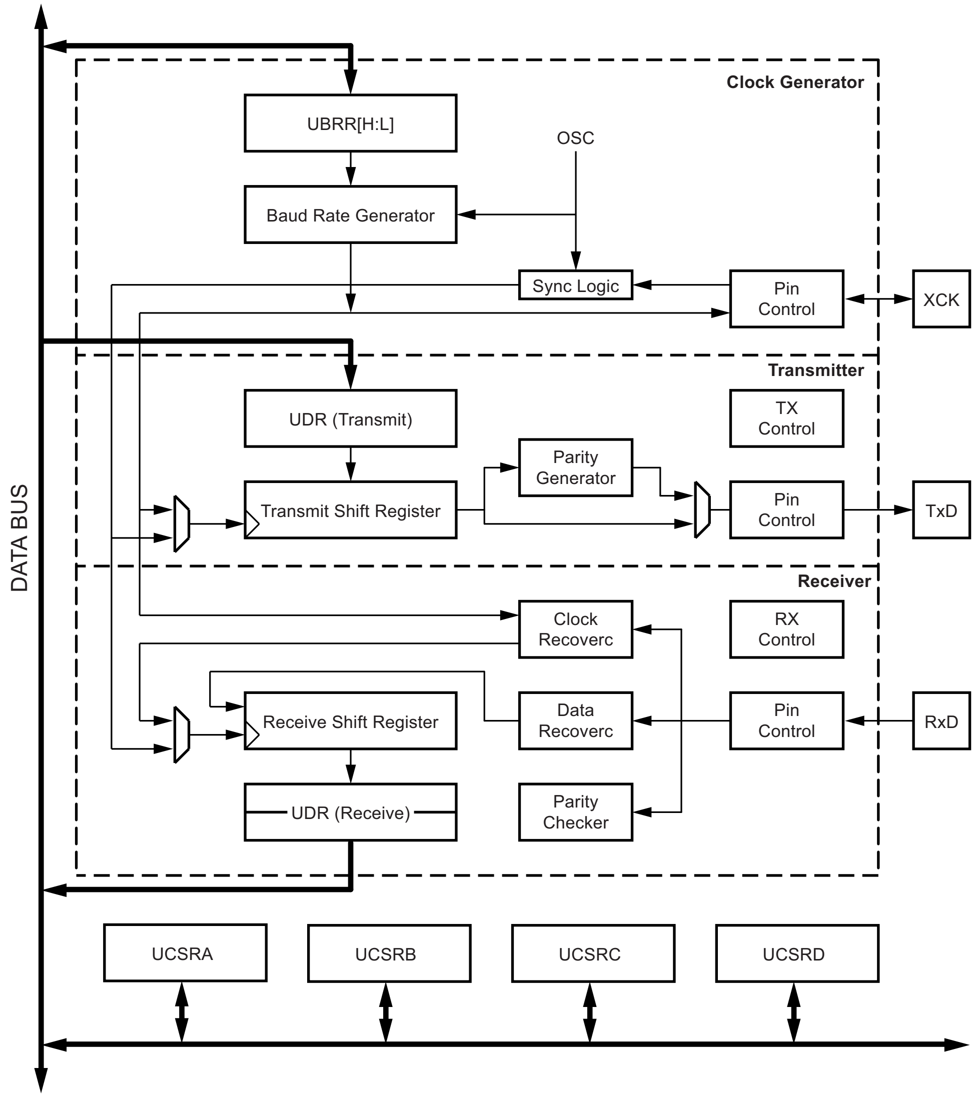
The universal synchronous and asynchronous serial receiver and transmitter (USART) is a highly flexible serial communication device.

A simplified block diagram of the USART transmitter is shown in Figure 17-1 on page 138. CPU-accessible I/O registers and I/O pins are shown in bold.

The power reduction USART0 bit, PRUSART0, in Section 8.4.2 “PRR – Power Reduction Register” on page 37 must be disabled by writing a logic zero to it.

The power reduction USART1 bit, PRUSART1, in Section 8.4.2 “PRR – Power Reduction Register” on page 37 must be disabled by writing a logic zero to it.

Figure 17-1. USART Block Diagram



For USART pin placement, see Figure 1-1 on page 3 and Section 11.3 “Alternate Port Functions” on page 59.

The dashed boxes in the block diagram of Figure 17-1 illustrate the three main parts of the USART as follows (listed from the top):

- Clock generator
- Transmitter
- Receiver

The clock generation logic consists of synchronization logic (for external clock input in synchronous slave operation) and the baud rate generator. The transfer clock pin (XCKn) is only used in synchronous transfer mode.

The transmitter consists of a single write buffer, a serial shift register, a parity generator, and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without delay between frames.

The receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a shift register, and a two-level receive buffer (UDRn). The receiver supports the same frame formats as the transmitter and can detect the following errors:

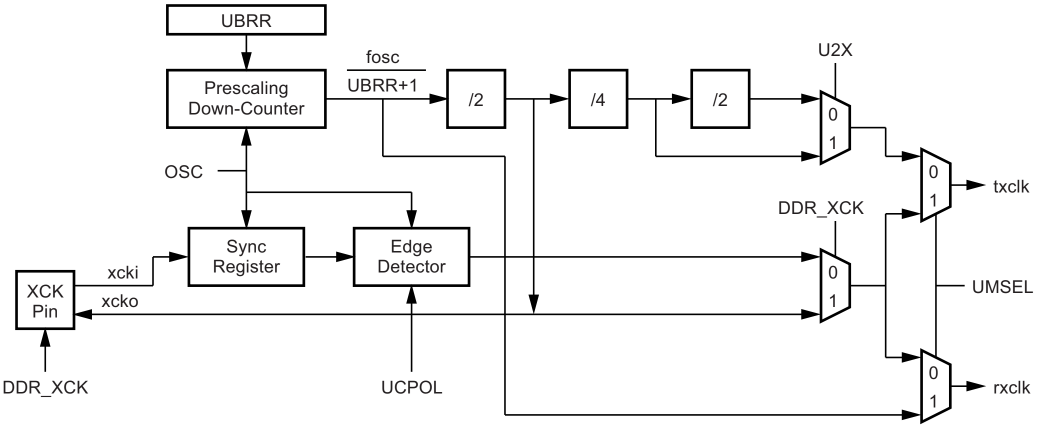
- Frame error
- Data overrun error
- Parity errors

The USARTn power reduction bit must be disabled for the USART to operate (see Section 8.4.2 “PRR – Power Reduction Register” on page 37).

17.4 Clock Generation

The clock generation logic creates the base clock for the transmitter and receiver. A block diagram of the clock generation logic is shown in Figure 17-2.

Figure 17-2. Clock Generation Logic, Block Diagram



Signal description for Figure 17-2:

txclk Transmitter clock (internal signal)
rxclk Receiver base clock (internal signal)
xcki Input from XCKn pin (internal signal). Used for synchronous slave operation
xcko Clock output to XCKn (internal signal). Used for synchronous master operation
f_{OSC} XTAL pin frequency (system clock)

The USART supports the following four clock operating modes:

- Normal asynchronous mode
- Double-speed asynchronous mode
- Master synchronous mode
- Slave synchronous mode

The UMSELn bit selects between asynchronous and synchronous operation. In asynchronous mode, the speed is controlled by the U2X bit.

In synchronous mode, the direction bit of the XCKn pin (DDR_XCKn) in the data direction register, where the XCKn pin is located (DDRx), controls whether the clock source is internal (master mode) or external (slave mode). The XCKn pin is active in synchronous mode only.

17.4.1 Internal Clock Generation – the Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master operating modes. The description in this section refers to Figure 17-2 on page 139.

The USART baud rate register (UBRRn) and the down-counter connected to it function as a programmable prescaler, or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRRn value each time the counter has counted down to “0” or when UBRRnL is written.

A clock is generated each time the counter reaches “0”. This is the baud rate generator clock output and has a frequency of f_{osc}/(UBRRn+1). Depending on the operating mode, the transmitter divides the baud rate generator clock output by 2, 8, or 16. The baud rate generator output is used directly by the receiver’s clock and data recovery units. However, the recovery units use a state machine that uses 2, 8, or 16 states, depending on the mode set by UMSELn, U2Xn, and DDR_XCKn bits.

Table 17-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each operating mode using an internally generated clock source.

Table 17-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Baud Rate[1]	UBRR Value
Asynchronous normal mode (U2Xn = 0)	$\mathrm{BAUD} = \dfrac{f_{OSC}}{16(\mathrm{UBRRn}+1)}$	$\mathrm{UBRRn} = \dfrac{f_{OSC}}{16\mathrm{BAUD}} - 1$
Asynchronous double-speed mode (U2Xn = 1)	$\mathrm{BAUD} = \dfrac{f_{OSC}}{8(\mathrm{UBRRn}+1)}$	$\mathrm{UBRRn} = \dfrac{f_{OSC}}{8\mathrm{BAUD}} - 1$
Synchronous master mode	$\mathrm{BAUD} = \dfrac{f_{OSC}}{2(\mathrm{UBRRn}+1)}$	$\mathrm{UBRRn} = \dfrac{f_{OSC}}{2\mathrm{BAUD}} - 1$

Note: 1. The baud rate is defined as the transfer rate in bits per second (bps).

Signal description for Table 17-1:

BAUD	Baud rate (in bits per second, bps)
f_{OSC}	System oscillator clock frequency
UBRR	Contents of the UBRRHn and UBRRLn registers, (0-4095)

Some examples of UBRRn values for selected system clock frequencies are shown in Section 17.11 “Examples of Baud Rate Setting” on page 154.

17.4.2 Double-Speed Operation

The transfer rate can be doubled by setting the U2Xn bit. Setting this bit only has an effect in asynchronous operating mode. In synchronous operating mode this bit should be cleared.

Setting this bit reduces the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note, however, that in this case the receiver only uses half the number of samples. In double-speed mode, the number of data and clock recovery samples are reduced from 16 to 8, and therefore a more accurate baud rate setting and system clock are required.

There are no downsides for the transmitter.

17.4.3 External Clock

External clocking is used in synchronous slave operating modes. To minimize the chance of meta-stability, the external clock input from the XCK pin is sampled by a synchronization register. The output from the synchronization register then passes through an edge detector before it is used by the transmitter and receiver. This process introduces a delay of two CPU clocks and the maximum external clock frequency is thus limited by the following equation:

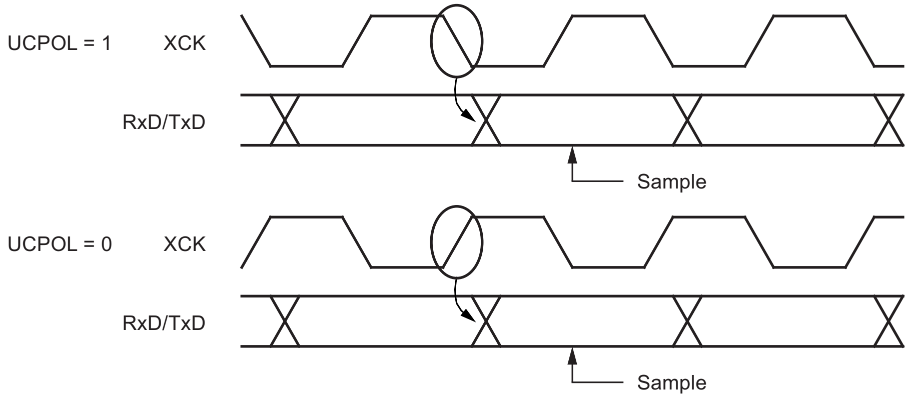
$$f_{XCKn} < \frac{f_{OSC}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible data loss due to frequency variations.

17.4.4 Synchronous Clock Operation

When synchronous mode is used (UMSELn = 1), the XCKn pin is used as either clock input (slave) or clock output (master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxDn) is sampled at the opposite XCKn clock edge of the edge where the data output (TxDn) is changed.

Figure 17-3. Synchronous Mode XCKn Timing



The UCPOLn bit UCRSC selects which XCKn clock edge is used for data sampling and which is used for data change. As Figure 17-3 shows, when UCPOLn is "0", the data is changed at the rising XCKn edge and sampled at the falling XCKn edge. If UCPOLn is set, the data is changed at the falling XCKn edge and sampled at the rising XCKn edge.

17.5 Frame Formats

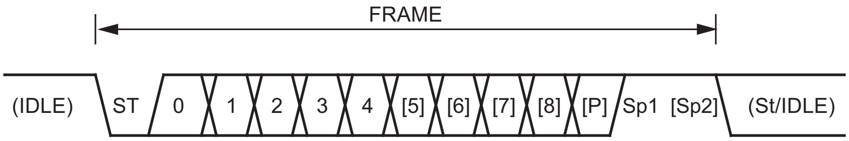
A serial frame is defined as one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- Start bit: 1
- Data bits: 5, 6, 7, 8, or 9
- Parity bit: no, even, or odd parity
- Stop bits: 1, or 2

A frame begins with the start bit followed by the least significant data bit. The other data bits then follow, the last one being the most significant bit. If enabled, the parity bit is inserted after the data bits before the stop bits. When a complete frame has been transmitted, it can be followed directly by a new frame or the communication line can be set to an idle (high) state.

Figure 17-4 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 17-4. Frame Formats



Signal description for Figure 17-4:

St	Start bit (always low)
(n)	Data bits (0 to 4/5/6/7/8)
P	Parity bit if enabled (odd or even)
Sp	Stop bit (always high)
IDLE	No transfers on the communication line (RxDn or TxDn) (high)

The frame format used by the USART is set by the UCSZn, UPMn, and USBSn bits as follows:

- The USART character size bits (UCSZn) select the number of data bits in the frame
- The USART parity mode bits (UPMn) choose the type of parity bit
- The selection between one or two stop bits is done by the USART stop bit select bit (USBSn). The receiver ignores the second stop bit. A frame error (FE) is therefore only detected in cases where the first stop bit is "0".

The receiver and transmitter use the same setting. Note that changing the setting of any of these bits corrupts all ongoing communication for both the receiver and transmitter.

17.5.1 Parity Bit Calculation

The parity bit is calculated by doing an "exclusive or" of all the data bits. If odd parity is used, the result of the "exclusive or" is inverted. The relation between the parity bit and data bits is as follows:

$$P_{EVEN} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{ODD} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

... where:

$\mathbf{P_{EVEN}}$	Parity bit using even parity
$\mathbf{P_{ODD}}$	Parity bit using odd parity
$\mathbf{d_n}$	Data bit n of the character

If used, the parity bit is located between the last data bit and the first stop bit of a serial frame.

17.6 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and, depending on the method of use, enabling the transmitter or the receiver. For interrupt driven USART operation, the global interrupt flag should be cleared and the USART interrupts disabled.

Before re-initializing baud rate or frame format, it should be checked that there are no ongoing transmissions during the period the registers are changed. The TXCn flag can be used to check that the transmitter has completed all transfers, and the RXCn flag can be used to check that there are no unread data in the receive buffer. Note that, if used, the TXCn flag must be cleared before each transmission (before UDRn is written).

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example[1]

```
USART_Init:
        ; Set baud rate
        out     UBRRnH, r17
        out     UBRRnL, r16
        ; Enable receiver and transmitter
        ldi     r16, (1<<RXENn)|(1<<TXENn)
        out     UCSRnB,r16
        ; Set frame format: 8data, 2stop bit
        ldi     r16, (1<<USBSn)|(3<<UCSZn0)
        out     UCSRnC,r16
        ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
        /* Set baud rate */
        UBRRnH = (unsigned char)(baud>>8);
        UBRRnL = (unsigned char)baud;
        /* Enable receiver and transmitter */
        UCSRnB = (1<<RXENn)|(1<<TXENn);
        /* Set frame format: 8data, 2stop bit */
        UCSRnC = (1<<USBSn)|(3<<UCSZn0);
}
```

Note: 1. See Section 4.2 “Code Examples” on page 7.

More advanced initialization routines can be done that include frame format as parameters, disable interrupts, etc. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine or be combined with initialization code for other I/O modules.

17.7 Data Transmission – the USART Transmitter

The USART transmitter is enabled by setting the transmit enable bit (TXENn). When the transmitter is enabled, the normal port operation of the TxDn pin is overridden by the USART and given the transmitter’s serial output function. The baud rate, operating mode, and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCKn pin is overridden and used as the transmission clock.

17.7.1 Sending Frames with 5 to 8 Data Bits

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDRn register. The buffered data in the transmit buffer is moved to the shift register when it is ready to send a new frame. The shift register is loaded with new data if it is in idle state (no ongoing transmission), or immediately after the last stop bit of the previous frame is transmitted. When the shift register is loaded with new data, it transfers one complete frame at the rate given by the baud rate register, the U2Xn bit, or by XCKn, depending on the operating mode.

The following code examples show a simple USART transmit function based on polling of the data register empty flag (UDREn). When using frames with less than eight bits, the most significant bits written to UDRn are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register R16.

Assembly Code Example[1]

```
USART_Transmit:
        ; Wait for empty transmit buffer
        sbis   UCSRnA,UDREn
        rjmp   USART_Transmit
        ; Put data (r16) into buffer, sends the data
        out    UDRn,r16
        ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRnA & (1<<UDREn)) )
                                ;
        /* Put data into buffer, sends the data */
        UDRn = data;
}
```

Note: 1. See Section 4.2 “Code Examples” on page 7.

The function simply waits for the transmit buffer to be empty by checking the UDREn flag before loading it with new data to be transmitted. If the data register empty interrupt is utilized, the interrupt routine writes the data into the buffer.

17.7.2 Sending Frames with 9 Data Bits

If 9-bit characters are used, the ninth bit must be written to the TXB8 bit in UCSRnB before the low byte of the character is written to UDRn. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

Assembly Code Example(1)(2)

```
USART_Transmit:
        ; Wait for empty transmit buffer
        sbis    UCSRnA,UDREn
        rjmp    USART_Transmit
        ; Copy 9th bit from r17 to TXB8
        cbi     UCSRnB,TXB8
        sbrc    r17,0
        sbi     UCSRnB,TXB8
        ; Put LSB data (r16) into buffer, sends the data
        out     UDRn,r16
        ret
```

C Code Example(1)(2)

```
void USART_Transmit( unsigned int data )
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRnA & (1<<UDREn))) )
                                    ;
        /* Copy 9th bit to TXB8 */
        UCSRnB &= ~(1<<TXB8);
        if ( data & 0x0100 )
                UCSRnB |= (1<<TXB8);
        /* Put data into buffer, sends the data */
        UDRn = data;
}
```

Notes: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB register is used after initialization.

2. See Section 4.2 “Code Examples” on page 7.

The ninth bit can be used for indicating an address frame when using multiprocessor communication mode or for other protocol handling such as synchronization.

17.7.3 Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART data register empty (UDREn) and transmit complete (TXCn). Both flags can be used for generating interrupts.

The data register empty flag (UDREn) indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty and cleared when the transmit buffer contains data to be transmitted that has not yet been moved to the shift register. For compatibility with future devices, always write this bit to “0”.

When the data register empty interrupt enable bit (UDRIEn) is set, the USART data register empty interrupt is executed as long as UDREn is set (provided that global interrupts are enabled). UDREn is cleared by writing UDRn. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to UDRn in order to clear UDREn or disable the data register empty interrupt. Otherwise, a new interrupt occurs once the interrupt routine terminates.

The transmit complete flag (TXCn) is set when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXCn flag is automatically cleared when a transmit complete interrupt is executed or it can be cleared by writing a “1” to its location. The TXCn flag is useful in half-duplex communication interfaces (such as the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the transmit compete interrupt enable bit (TXCIEn) is set, the USART transmit complete interrupt is executed when the TXCn flag is set (and provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXCn flag, because this is done automatically when the interrupt is executed.

17.7.4 Parity Generator

The parity generator calculates the parity bit for the serial frame data. When the parity bit is enabled (UPMn1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame sent.

17.7.5 Disabling the Transmitter

Clearing TXENn disables the transmitter, but the change does not become effective until all ongoing and pending transmissions are completed, i.e., not before the data to be transmitted has been cleared from the transmit shift register and transmit buffer register. When disabled, the transmitter no longer overrides the TxDn pin.

17.8 Data Reception – the USART Receiver

The USART receiver is enabled by writing the receive enable bit (RXENn). When the receiver is enabled, the normal operation of the RxDn pin is overridden by the USART and given the receiver's serial input function. The baud rate, operating mode, and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCKn pin is used as the transfer clock.

17.8.1 Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit is sampled at the baud rate, or XCKn clock, and then shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit is ignored by the receiver. When the first stop bit is received, i.e., a complete serial frame is present in the receive shift register, its contents are moved to the receive buffer. The receive buffer can then be read by reading UDRn.

The following code example shows a simple USART receive function based on polling of the receive complete flag (RXCn). When using frames with less than eight bits, the most significant bits of the data read from the UDRn are masked to "0". The USART has to be initialized before the function can be used.

Assembly Code Example[1]

```
USART_Receive:
       ; Wait for data to be received
       sbis   UCSRnA, RXCn
       rjmp   USART_Receive
       ; Get and return received data from buffer
       in     r16, UDRn
       ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
       /* Wait for data to be received */
       while ( !(UCSRnA & (1<<RXCn)) )
                           ;
       /* Get and return received data from buffer */
       return UDRn;
}
```

Note: 1. See Section 4.2 "Code Examples" on page 7.

The function simply waits for data to be present in the receive buffer by checking the RXCn flag before reading the buffer and returning the value.

17.8.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used, the ninth bit must be read from the RXB8n bit before reading the low bits from UDRn. This rule applies to the FEn, DORn, and UPEn status flags as well. Status bits must be read before data from UDRn, because reading UDRn changes the state of the receive buffer FIFO and as a result, the state of TXB8n, FE, DORn, and UPEn bits.

The following code example shows a simple USART receive function that handles both 9-bit characters and the status bits.

Assembly Code Example[1]

```
USART_Receive:
       ; Wait for data to be received
       sbis                UCSRnA, RXCn
       rjmp                USART_Receive
       ; Get status and 9th bit, then data from buffer
       in                  r18, UCSRnA
       in                  r17, UCSRnB
       in                  r16, UDRn
       ; If error, return -1
       andi                r18,(1<<FEn)|(1<<DORn)|(1<<UPEn)
       breq                USART_ReceiveNoError
       ldi                 r17, HIGH(-1)
       ldi                 r16, LOW(-1)
USART_ReceiveNoError:
       ; Filter the 9th bit, then return
       lsr                 r17
       andi                r17, 0x01
       ret
```

C Code Example[1]

```
unsigned int USART_Receive( void )
{
       unsigned char status, resh, resl;
       /* Wait for data to be received */
       while ( !(UCSRnA & (1<<RXCn)) )
                           ;
       /* Get status and 9th bit, then data */
       /* from buffer */
       status = UCSRnA;
       resh = UCSRnB;
       resl = UDRn;
       /* If error, return -1 */
       if ( status & (1<<FEn)|(1<<DORn)|(1<<UPEn) )
              return -1;
       /* Filter the 9th bit, then return */
       resh = (resh >> 1) & 0x01;
       return ((resh << 8) | resl);
}
```

Note: 1. See Section 4.2 “Code Examples” on page 7.

The receive function example reads all I/O registers into the register file before any computation is done. This results in optimal receive buffer utilization because the buffer location read is free to accept new data as early as possible.

17.8.3 Receive Compete Flag and Interrupt

The USART receiver has one flag that indicates the receiver state.

The receive complete flag (RXCn) indicates if there are unread data present in the receive buffer. This flag is set when unread data exist in the receive buffer. The flag is cleared when the receive buffer is empty (i.e., it does not contain any unread data). If the receiver is disabled (RXENn = 0), the receive buffer is flushed, causing the RXCn bit to become “0”.

When the receive complete interrupt enable (RXCIEn) is set, the USART receive complete interrupt is executed as long as the RXCn flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDRn in order to clear the RXCn flag. Otherwise, a new interrupt occurs once the interrupt routine terminates.

17.8.4 Receiver Error Flags

The USART receiver has three error flags: frame error (FEn), data overrun (DORn), and parity error (UPEn). All can be accessed by reading UCSRnA. The error flags are commonly located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the error flags, the UCSRnA must be read before the receive buffer (UDRn), because reading the UDRn I/O location changes the buffer read location. Another attribute of error flags is that they cannot be altered by software writing to the flag location. However, to ensure upward compatibility of future USART implementations, all flags must be set to “0” when the UCSRnA is written. None of the error flags can generate interrupts.

The frame error (FEn) flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FEn flag is “0” when the stop bit was read correctly (as “1”) and the FEn flag is “1” when the stop bit was incorrect (“0”). This flag can be used for detecting out-of-sync conditions, break conditions, and protocol handling. The FEn flag is not affected by the setting of the USBSn bit in UCSRnC because the receiver ignores all stop bits except for the first stop bits. For compatibility with future devices, always set this bit to “0” when writing to UCSRnA.

The data overrun (DORn) flag indicates data loss due to a receiver buffer full condition. A data overrun occurs when the receive buffer is full (two characters), a new character is waiting in the receive shift register, or a new start bit is detected. If the DORn flag is set, there was one or more serial frame lost between the frame last read from UDRn and the next frame read from UDRn. For compatibility with future devices, always write this bit to “0” when writing to UCSRnA. The DORn flag is cleared when the frame received was successfully moved from the shift register to the receive buffer.

The parity error (UPEn) flag indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled, the UPEn bit always reads as “0”. For compatibility with future devices, always set this bit to “0” when writing to UCSRnA. For more information, see Section 17.5.1 “Parity Bit Calculation” on page 142 and Section 17.8.5 “Parity Checker” on page 148.

17.8.5 Parity Checker

The parity checker is active when the high USART parity mode bit (UPMn1) is set. The type of parity check to be performed (odd or even) is selected by the UPMn0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The parity error flag (UPEn) can then be read by software to check if the frame had a parity error.

If parity checking is enabled, the UPEn bit is set if the next character that can be read from the receive buffer had a parity error when received. This bit is valid until the receive buffer (UDRn) is read.

17.8.6 Disabling the Receiver

Unlike the transmitter, the receiver is disabled immediately and any data from ongoing receptions is lost. When disabled (RXENn = 0), the receiver no longer overrides the normal function of the RxDn port pin and the FIFO buffer is flushed with any remaining data in the buffer lost.

17.8.7 Flushing the Receive Buffer

The receiver buffer FIFO is flushed when the receiver is disabled, i.e., the buffer is emptied of its contents. Unread data is lost. To flush the buffer during normal operation, for example, due to an error condition, read the UDRn until the RXCn flag is cleared.

The following code example shows how to flush the receive buffer.

Assembly Code Example(1)

```
USART_Flush:
        sbis   UCSRnA, RXCn
        ret
        in     r16, UDRn
        rjmp   USART_Flush
```

C Code Example(1)

```
void USART_Flush( void )
{
        unsigned char dummy;
        while (UCSRnA & (1<<RXCn)) dummy = UDRn;
}
```

Note: 1. See Section 4.2 “Code Examples” on page 7.

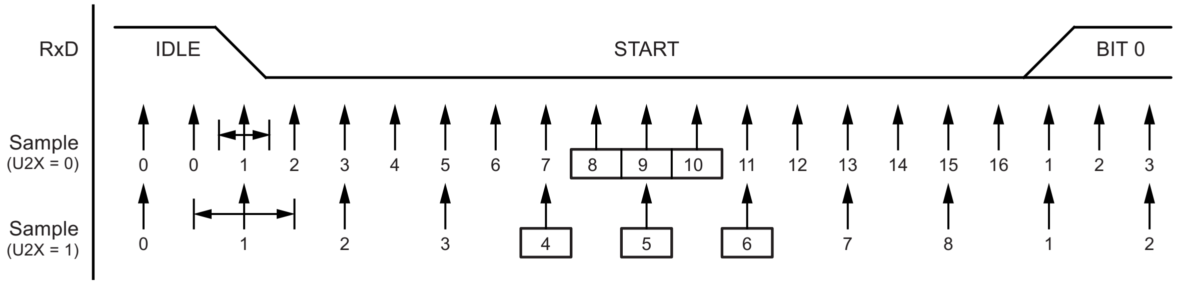
17.9 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxDn pin. The data recovery logic samples and low-pass filters each incoming bit, improving the noise immunity of the receiver. The operational range of asynchronous reception depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

17.9.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes the internal clock to the incoming serial frames. Figure 17-5 illustrates the sampling process of the start bit of an incoming frame. In normal mode the sample rate is 16 times the baud rate, in double-speed mode eight times. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double-speed operating mode (U2Xn = 1). Samples denoted “0” are samples done when the RxDn line is idle (i.e., no communication activity).

Figure 17-5. Start Bit Sampling

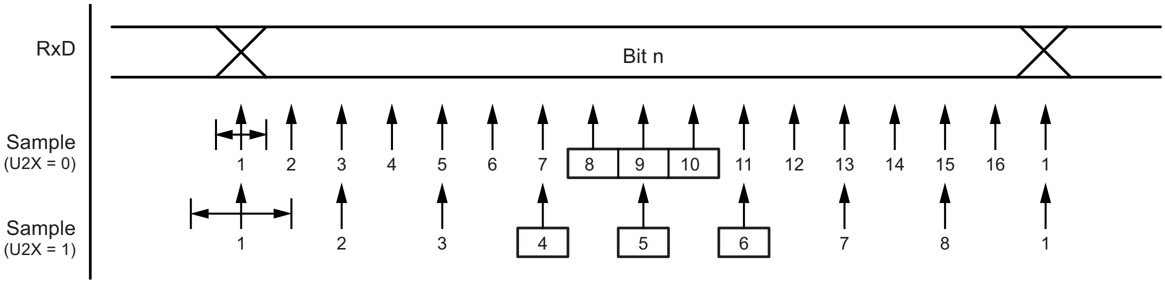


When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. In Figure 17-5 samples are indicated with numbers inside boxes and sample number 1 denotes the first “0” sample. The clock recovery logic then uses samples 8, 9, and 10 (in normal mode), or samples 4, 5, and 6 (in double-speed mode) to decide if a valid start bit is received. If two or more of these three samples have logic high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high-to-low transition. If, however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

17.9.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and eight states for each bit in double-speed mode. Figure 17-6 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

Figure 17-6. Sampling of Data and Parity Bit

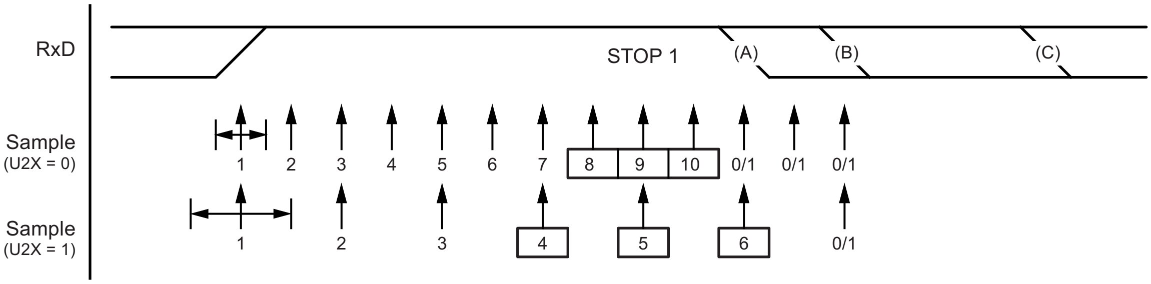


The decision of the logic level of the received bit is made by doing a majority voting of the logic value to the three samples in the center of the received bit. In the figure, the center samples are emphasized by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic one. If two or all three samples have low levels, the received bit is registered to be a logic zero. This majority voting process acts as a low pass filter for the incoming signal on the RxDn pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit.

Note that the receiver only uses the first stop bit of a frame.

Figure 17-7 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 17-7. Stop Bit Sampling and Next Start Bit Sampling



The stop bit is subject to the same majority voting as the other bits in the frame. If the stop bit is registered to have a logic low value, the frame error flag (FEn) is set.

A new high-to-low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. In normal speed mode, the first low level sample can be at the point marked (A) in Figure 17-7. In double-speed mode the first low level must be delayed to (B). Point (C) marks the full length of a stop bit.

The early start bit detection influences the operational range of the receiver.

17.9.3 Asynchronous Operational Range

The operational range of the receiver depends on the mismatch between the received bit rate and the internally generated baud rate. If the transmitter is sending frames excessively fast or slow bit rates, or the internally generated baud rate of the receiver does not have a similar base frequency (see Table 17-2 on page 151), the receiver is not able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\times S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

... where:

D Sum of character size and parity size (D = 5 to 10 bit)

S Samples per bit, 16 for normal speed mode, or 8 for double-speed mode

S_F First sample number used for majority voting, 8 (normal speed), or 4 (double)

S_M Middle sample number for majority voting, 9 (normal speed), or 5 (double speed)

R_{slow} The ratio of the slowest incoming data rate that can be accepted with respect to the receiver baud rate

R_{fast} The ratio of the fastest incoming data rate that can be accepted with respect to the receiver baud rate

Table 17-2 and Table 17-3 list the maximum receiver baud rate error that can be tolerated. Note that normal speed mode has higher toleration of baud rate variations.

Table 17-2. Recommended Maximum Receiver Baud Rate Error in Normal Speed Mode

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/–6.8	±3.0
6	94.12	105.79	+5.79/–5.88	±2.5
7	94.81	105.11	+5.11/–5.19	±2.0
8	95.36	104.58	+4.58/–4.54	±2.0
9	95.81	104.14	+4.14/–4.19	±1.5
10	96.17	103.78	+3.78/–3.83	±1.5

Table 17-3. Recommended Maximum Receiver Baud Rate Error in Double-Speed Mode

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/–5.88	±2.5
6	94.92	104.92	+4.92/–5.08	±2.0
7	95.52	104,35	+4.35/–4.48	±1.5
8	96.00	103.90	+3.90/–4.00	±1.5
9	96.39	103.53	+3.53/–3.61	±1.5
10	96.70	103.23	+3.23/–3.30	±1.0

The recommendations of the maximum receiver baud rate error are made under the assumption that the receiver and transmitter divide the maximum total error equally.

There are two possible sources for the receiver baud rate error:

- The system clock of the receiver always has some minor instability over the supply voltage range and the temperature range.
- The second source for error is more controllable. The baud rate generator cannot always do an exact division of the system frequency to get the desired baud rate. In this case, an UBRR value that produces an acceptable low error should be used if possible.

17.9.4 Start Frame Detection

The USART start frame detector can wake up the MCU from power-down, standby, or ADC noise reduction sleep mode when it detects a start bit.

When a high-to-low transition is detected on RxDn, the internal 8MHz oscillator is powered up and the USART clock is enabled. After start-up, the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the internal 8MHz oscillator start-up time. Start-up time of the internal 8MHz oscillator varies with supply voltage and temperature.

The USART start frame detection works both in asynchronous and synchronous modes. It is enabled by writing the start frame detection enable bit (SFDEn). If the USART start interrupt enable (RXSIE) bit is set, the USART receive start interrupt is generated immediately when start is detected.

When using the feature without start interrupt, the start detection logic activates the internal 8MHz oscillator and the USART clock only while the frame is being received. Other clocks remain stopped until the receive complete interrupt wakes up the MCU.

The maximum baud rate in synchronous mode depends on the sleep mode the device is woken up from as specified here:

- Idle or ADC noise reduction sleep mode: system clock frequency divided by four
- Standby or power-down: 500kb/s

The maximum baud rate in asynchronous mode depends on the sleep mode the device is woken up from as specified here:

- Idle sleep mode: the same as in active mode.
- Other sleep modes: see Table 17-4 and Table 17-5.

Table 17-4. Maximum Total Baud Rate Error in Normal Speed Mode

Baud Rate	Frame Size					
	5	6	7	8	9	10
0 to 28.8kb/s	+6.67 –5.88	+5.79 –5.08	+5.11 –4.48	+4.58 –4.00	+4.14 –3.61	+3.78 –3.30
38.4kb/s	+6.63 –5.88	+5.75 –5.08	+5.08 –4.48	+4.55 –4.00	+4.12 –3.61	+3.76 –3.30
57.6kb/s	+6.10 –5.88	+5.30 –-5.08	+4.69 –4.48	+4.20 –4.00	+3.80 –3.61	+3.47 –3.30
76.8kb/s	+5.59 –5.88	+4.85 –5.08	+4.29 –4.48	+3.85 –4.00	+3.48 –3.61	+3.18 –3.30
115.2kb/s	+4.57 –5.88	+3.97 –5.08	+3.51 –4.48	+3.15 –4.00	+2.86 –3.61	+2.61 –3.30

Table 17-5. Maximum Total Baud Rate Error in Double-Speed Mode

Baud Rate	Frame Size					
	5	6	7	8	9	10
0 to 57.6kb/s	+5.66 –4.00	+4.92 –3.45	+4.35 –3.03	+3.90 –2.70	+3.53 –2.44	+3.23 –2.22
76.8kb/s	+5.59 –4.00	+4.85 –3.45	+4.29 –3.03	+3.85 –2.70	+3.48 –2.44	+3.18 –2.22
115.2kb/s	+4.57 –4.00	+3.97 –3.45	+3.51 –3.03	+3.15 –2.70	+2.86 –2.44	+2.61 –2.22

17.10 Multiprocessor Communication Mode

Setting the multiprocessor communication mode bit (MPCMn) enables a filtering function of incoming frames received by the USART receiver. Frames that do not contain address information are ignored and not put into the receive buffer. In a system with multiple MCUs that communicate via the same serial bus this effectively reduces the number of incoming frames that has to be handled by the CPU. The transmitter is unaffected by the MPCMn bit, but has to be used differently when it is a part of a system utilizing the multiprocessor communication mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the receiver is set up for frames with nine data bits, then the ninth bit (RXB8n) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is “1”, the frame contains an address. When the frame type bit is “0”, the frame is a data frame.

The multiprocessor communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it receives the following data frames as normal whereas the other slave MCUs ignore the received frames until another address frame is received.

For an MCU to act as a master MCU, it can use a 9-bit character frame format. The ninth bit (TXB8) must be set when an address frame is transmitted and cleared when a data frame is transmitted. In this case, the slave MCUs must be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in multiprocessor communication mode:

1. All slave MCUs are set to multiprocessor communication mode (MPCMn = 1).
2. The master MCU sends an address frame and all slaves receive and read this frame. In the slave MCUs the RXCn flag is set as normal.
3. Each slave MCU reads UDRn and determines if it has been selected. If so, it clears the MPCMn bit. If not, it waits for the next address byte and keeps the MPCMn setting.
4. The addressed MCU receives all data frames until a new address frame is received. The other slave MCUs, which still have the MPCMn bit set, ignore the data frames.
5. When the last data frame is received by the addressed MCU, it sets the MPCMn bit and waits for a new address frame from master. The process then repeats from step 2.

Though possible, it is impractical to use any of the 5- to 8-bit character frame formats due to the fact that the receiver would then have to alternate between using n and n+1 character frame formats. This makes full-duplex operation difficult because the transmitter and receiver use the same character size setting. If 5- to 8-bit character frames are used, the transmitter must be set to use two stop bits as the first stop bit is used for indicating the frame type.

Do not use read-modify-write instructions (SBI and CBI) to set or clear the MPCMn bit. The MPCMn bit shares the same I/O location as the TXCn flag and this might accidentally be cleared when using SBI or CBI instructions.

17.11 Examples of Baud Rate Setting

Commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Section 17.11 "Examples of Baud Rate Setting" on page 154. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate are shown in bold. Higher error ratings are acceptable, but the receiver has less noise resistance when the error ratings are high—especially for large serial frames (see Section 17.9.3 "Asynchronous Operational Range" on page 151). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \times 100\%$$

Table 17-6. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	–7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	–3.5%	7	0.0%	15	0.0%	8	–3.5%	16	2.1%
19.2k	2	8.5%	6	–7.0%	5	0.0%	11	0.0%	6	–7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	–3.5%
38.4k	1	–18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	–7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	–18.6%	1	–25.0%	2	0.0%	1	–18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max.[1]	62.5kb/s		125kb/s		115.2kb/s		230.4kb/s		125kb/s		250kb/s	

Note: 1. UBRR = 0, error = 0.0%

Table 17-7. **Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)**

Baud Rate (bps)	f_{osc} = 3.6864MHz				f_{osc} = 1.0000MHz				f_{osc} = 7.3728MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	–0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	–3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	–7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	–3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	–7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	–7.8%	1	–7.8%	0	0.0%	1	0.0%	1	–7.8%	3	–7.8%
0.5M	–	–	0	–7.8%	–	–	0	0.0%	0	–7.8%	1	–7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	–7.8%
Max.[1]	230.4kb/s		460.8kb/s		250kb/s		0.5Mb/s		460.8kb/s		921.6kb/s	

Note: 1. UBRR = 0, error = 0.0%

Table 17-8. **Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)**

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	–0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	–0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	–0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	–3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	–7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	–3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	–7.8%	5	–7.8%	3	–7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	–7.8%	1	–7.8%	3	–7.8%
1M	–	–	0	0.0%	–	–	–	–	0	–7.8%	1	–7.8%
Max.[1]	0.5Mb/s		1Mb/s		691.2kb/s		1.3824Mb/s		921.6kb/s		1.8432Mb/s	

Note: 1. UBRR = 0, error = 0.0%

Table 17-9. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	–0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	–0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	–0.1%	79	0.0%	159	0.0%	86	–0.2%	173	–0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	–0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	–0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	–1.4%	64	0.2%
57.6k	16	2.1%	34	–0.8%	19	0.0%	39	0.0%	21	–1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	–1.4%
115.2k	8	–3.5%	16	2.1%	9	0.0%	19	0.0%	10	–1.4%	21	–1.4%
230.4k	3	8.5%	8	–3.5%	4	0.0%	9	0.0%	4	8.5%	10	–1.4%
250k	3	0.0%	7	0.0%	4	–7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.[1]	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mb/s	

Note: 1. UBRR = 0, error = 0.0%

17.12 Register Description

17.12.1 UDRn – USART I/O Data Register

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	RXB[7:0]								**UDR0 (Read)**
0x20 (0x40)	TXB[7:0]								**UDR0 (Write)**
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	5	4	3	2	1	0	
(0x73)	RXB[7:0]								**UDR1 (Read)**
(0x73)	TXB[7:0]								**UDR1 (Write)**
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 USART transmit data buffer and USART receive data buffer registers share the same I/O address, referred to as USART data register or UDRn. Data written to UDRn goes to the transmit data buffer register (TXB). Reading UDR returns the contents of the receive data buffer register (RXB).

For 5-, 6-, or 7-bit characters, the upper unused bits are ignored by the transmitter and set to “0” by the receiver.

The transmit buffer can only be written when the UDREn flag is set. Data written to UDRn when the UDREn flag is not set is ignored. When the transmitter is enabled and data is written to the transmit buffer, the transmitter loads the data into the transmit shift register when it is empty. The data is then serially transmitted on the TxDn pin.

The receive buffer consists of a two-level FIFO. The FIFO changes its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, read-modify-write instructions (SBI and CBI) should not be used to access this location. Care should also be taken when using bit test instructions (SBIC and SBIS) because these also change the state of the FIFO.

17.12.2 UCSRnA – USART Control and Status register A

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	**RXC0**	**TXC0n**	**UDRE0n**	**FE0**	**DOR0**	**UPE0**	**U2X0**	**MPCM0**	**UCSR0A**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x79)	**RXC1**	**TXC1**	**UDRE1**	**FE1**	**DOR1**	**UPE1**	**U2X1**	**MPCM1**	**UCSR1A**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCn: USART Receive Complete**

This flag is set when there is unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer is flushed causing the RXCn flag to become “0”. The flag can be used to generate a receive complete interrupt (see RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag is set when the entire frame in the transmit shift register has been shifted out and there is no new data currently present in the transmit buffer (UDRn). The TXCn flag bit is automatically cleared when a transmit complete interrupt is executed or it can be cleared by writing a “1” to its bit location. The flag can generate a transmit complete interrupt (see TXCIEn bit).

- **Bit 5 – UDREn: USART Data Register Empty**

This flag indicates that the transmit buffer (UDRn) is ready to receive new data. If UDREn is “1”, the buffer is empty and therefore ready to be written to. The UDREn flag can generate a data register empty interrupt (see UDRIEn bit).

The UDREn flag is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FEn: Frame Error**

This flag is set if the next character in the receive buffer had a frame error when received (i.e., when the first stop bit of the next character in the receive buffer is “0”). This bit is valid until the receive buffer (UDRn) is read. The FEn bit is “0” when the stop bit of received data is “1”.

Always set this bit to “0” when writing the register.

- **Bit 3 – DORn: Data Overrun**

This bit is set if a data overrun condition is detected. A data overrun occurs when the receive buffer is full (two characters), a new character is waiting in the receive shift register, or a new start bit is detected. This bit is valid until the receive buffer (UDRn) is read.

Always set this bit to “0” when writing the register.

- **Bit 2 – UPEn: USART Parity Error**

This bit is set if the next character in the receive buffer had a parity error when received and the parity checking was enabled at that point (UPMn1 = 1). This bit is valid until the receive buffer (UDRn) is read.

Always set this bit to “0” when writing the register.

- **Bit 1 – U2Xn: Double the USART Transmission Speed**

This bit only has effect for asynchronous operation. Write this bit to “0” when using synchronous operation.

Writing this bit to “1” reduces the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCMn: Multiprocessor Communication Mode**

This bit enables multiprocessor communication mode. When the bit is written to “1”, all the incoming frames received by the USART receiver that do not contain address information are ignored. The transmitter is unaffected by the MPCMn bit. For more information, see Section 17.10 “Multiprocessor Communication Mode” on page 153.

17.12.3 UCSRnB – USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	UCSR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x78)	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	UCSR1B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable**

Writing this bit to “1” enables interrupt on the RXCn flag.

A USART receive complete interrupt is generated only if the RXCIEn bit, the global interrupt flag, and the RXCn bits are set.

- **Bit 6 – TXCIEn: TX Complete Interrupt Enable**

Writing this bit to “1” enables interrupt on the TXCn flag.

A USART receive complete interrupt is generated only if the TXCIEn bit, the global interrupt flag, and the TXCn bits are set.

- **Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable**

Writing this bit to “1” enables interrupt on the UDREn flag.

A data register empty interrupt is generated only if the UDRIEn bit is written to “1”, the global interrupt flag in SREG is written to “1”, and the UDREn bit in UCSRnA is set.

- **Bit 4 – RXENn: Receiver Enable**

Writing this bit to “1” enables the USART receiver. When enabled, the receiver overrides normal port operation for the RxDn pin.

Writing this bit to “0” disables the receiver. Disabling the receiver flushes the receive buffer, invalidating the FEn, DORn, and UPEn flags.

- **Bit 3 – TXENn: Transmitter Enable**

Writing this bit to “0” enables the USART transmitter. When enabled, the transmitter overrides normal port operation for the TxDn pin.

Writing this bit to “0” disables the transmitter. Disabling the transmitter becomes effective after ongoing and pending transmissions are completed, i.e., when the transmit shift register and transmit buffer register do not contain data to be transmitted. When disabled, the transmitter no longer overrides the TxDn port.

- **Bit 2 – UCSZn2: Character Size**

The UCSZn2 bit combined with the UCSZn[1:0] bits set the number of data bits (character size) in the frame the receiver and transmitter use.

- **Bit 1 – RXB8n: Receive Data Bit 8**

RXB8n is the ninth data bit of the received character when operating with serial frames with nine data bits. It must be read before reading the low bits from UDRn.

- **Bit 0 – TXB8n: Transmit Data Bit 8**

TXB8n is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. It must be written before writing the low bits to UDRn.

17.12.4 UCSRnC – USART Control and Status register C

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	UCSR0C
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Bit	7	6	5	4	3	2	1	0	
(0x77)	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	UCSR1C
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bits 7:6 – UMSELn[1:0]: USART Mode Select**

These bits select the operating mode of the USART as shown in Table 17-10.

Table 17-10. UMSELn Bit Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	Reserved
1	1	Master SPI (MSPIM)[1]

Note: 1. For a complete description of master SPI mode (MSPIM) operation, see Section 18. "USART in SPI Mode" on page 162.

- **Bits 5:4 – UPMn1:0: Parity Mode**

These bits enable and set the type of parity generation and check. If enabled, the transmitter automatically generates and sends the parity of the transmitted data bits within each frame. The receiver generates a parity value for the incoming data and compares it to the UPMn setting. If a mismatch is detected, the UPEn flag is set.

Table 17-11. Parity Mode Selection

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, even parity
1	1	Enabled, odd parity

- **Bit 3 – USBSn: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the transmitter. The receiver ignores this setting.

Table 17-12. USBSn Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

- **Bits 2:1 – UCSZn[1:0]: Character Size**

The UCSZn[1:0] bits combined with the UCSZn2 bit in UCSRnB set the number of data bits (character size) in a frame the receiver and transmitter use (see Table 17-13).

Table 17-13. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOLn: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to “0” when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample and the synchronous clock (XCKn).

Table 17-14. Clock Polarity Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCK edge	Falling XCK edge
1	Falling XCK edge	Rising XCK edge

17.12.5 UCSRnD – USART Control and Status register D

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**RXSIE0**	**RXS0**	**SFDE0**	**–**	**–**	**–**	**–**	**–**	**UCSR0D**
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	1	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x76)	**RXSIE1**	**RXS1**	**SFDE1**	**–**	**–**	**–**	**–**	**–**	**UCSR1D**
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXSIEn: USART RX Start Interrupt Enable**

Writing this bit to “1” enables the interrupt on the RXSn flag. In sleep modes this bit enables the start frame detector that can wake up the MCU when a start condition is detected on the RxDn line.

The USART RX start interrupt is generated only if the RXSIEn bit, the global interrupt enable flag, and RXSn are set.

- **Bit 6 – RXSn: USART RX Start**

This flag is set when a start condition is detected on the RxDn line. If the RXSIEn bit and the global interrupt enable flag are set, an RX start interrupt is generated when this flag is set. The flag can only be cleared by writing a logic one to the RXSn bit location.

If the start frame detector is enabled and the global interrupt enable flag is set, the RX start interrupt wakes up the MCU from all sleep modes.

- **Bit 5 – SFDEn: Start Frame Detection Enable**

Writing this bit to “1” enables the USART start frame detection mode. The start frame detector is able to wake up the MCU from sleep mode when a start condition, i.e., a high (IDLE) to low (START) transition, is detected on the RxDn line.

Table 17-15. USART Start Frame Detection Modes

SFDEn	RXSIEn	RXCIEn	Description
0	X	X	Start frame detector disabled
1	0	0	Reserved
1	0	1	Start frame detector enabled. The RXCn flag wakes up MCU from all sleep modes.
1	1	0	Start frame detector enabled. The RXSn flag wakes up MCU from all sleep modes.
1	1	1	Start frame detector enabled. Both RXCn and RXSn wake up the MCU from all sleep modes.

For more information, see Section 17.9.4 “Start Frame Detection” on page 152.

- **Bits 4:0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny1634 and are always read as “0”.

17.12.6 UBRRnL and UBRRnH – USART Baud Rate Registers

Initial Value	0	0	0	0	0	0	0	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Bit	15	14	13	12	11	10	9	8	
0x22 (0x42)	–	–	–	–	UBRR0[11:8]				UBRR0H
0x21 (0x41)	UBRR0[7:0]								UBRR0L
Bit	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	
Initial Value	0	0	0	0	0	0	0	0	

Initial Value	0	0	0	0	0	0	0	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Bit	15	14	13	12	11	10	9	8	
(0x75)	–	–	–	–	UBRR1[11:8]				UBRR1H
(0x74)	UBRR1[7:0]								UBRR1L
Bit	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	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 15:12 – Res: Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bits must be written to “0” when UBRRnH is written.

- **Bits 11:0 – UBRR[11:0]: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. UBRRnH contains the four most significant bits and UBRRnL contains the eight least significant bits of the USART baud rate.

Writing to UBRRnL triggers an immediate update of the baud rate prescaler. Ongoing transmissions by the transmitter and receiver are corrupted when the baud rate is changed.

18. USART in SPI Mode

18.1 Features

- Full duplex, three-wire synchronous data transfer
- Master operation
- Supports all four SPI operating modes (mode 0, 1, 2, and 3)
- LSB first or MSB first data transfer (configurable data order)
- Queued operation (double-buffered)
- High-resolution baud rate generator
- High-speed operation (fXCKmax = fCK/2)
- Flexible interrupt generation

18.2 Overview

The universal synchronous and asynchronous serial receiver and transmitter (USART) can be set to an operating mode compliant with master SPI.

Setting both UMSELn[1:0] bits to “1” enables the USART in MSPIM logic. In this operating mode the SPI master control logic takes direct control of the USART resources. These resources include the transmitter and receiver shift registers and buffers and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic are disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both operating modes.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

18.3 Clock Generation

The clock generation logic generates the base clock for the transmitter and receiver. For the USART MSPIM operating mode, only internal clock generation (i.e., master operation) is supported. Therefore, for the USART in MSPIM to operate correctly, the data direction register (DDRx) where the XCK pin is located must be configured to set the pin as output (DDR_XCKn = 1). It is advisable to set up the DDR_XCKn before the USART in MSPIM is enabled (i.e., before TXENn and RXENn bits are set).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRR setting can therefore be calculated using the same equations (see Table 18-1).

Table 18-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Calculating Baud Rate[1]	Calculating UBRR Value
Synchronous Master mode	$\mathrm{BAUD} = \dfrac{f_{\mathrm{OSC}}}{2(\mathrm{UBRRn}+1)}$	$\mathrm{UBRRn} = \dfrac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$

Note: 1. The baud rate is defined as the transfer rate in bits per second (bps).

BAUD	Baud rate (in bits per second, bps)
f_{OSC}	System oscillator clock frequency
UBRRn	Contents of UBRRnH and UBRRnL, (0-4095)

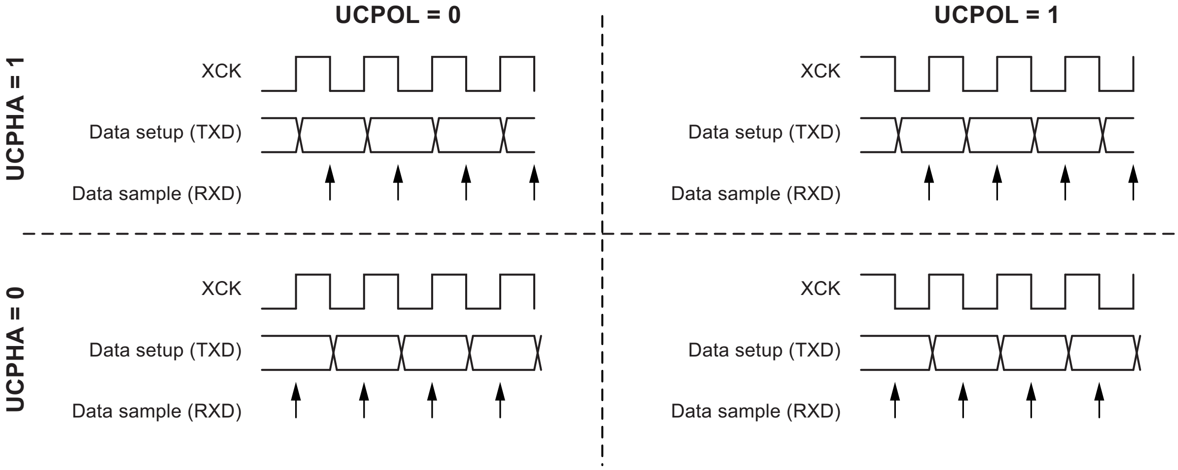
18.4 SPI Data Modes and Timing

With respect to serial data, there are four combinations of XCKn (SCK) phase and polarity which are determined by the UCPHAn and UCPOLn control bits. The data transfer timing diagrams are shown in Figure 18-1. Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAn functionality is summarized in Table 18-2. Note that changing the setting of any of these bits corrupts all ongoing communication for both the receiver and transmitter.

Table 18-2. UCPOLn and UCPHAn Functionality

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
0	0	0	Sample (rising)	Setup (falling)
0	1	1	Setup (rising)	Sample (falling)
1	0	2	Sample (falling)	Setup (rising)
1	1	3	Setup (falling)	Sample (rising)

Figure 18-1. UCPHAn and UCPOLn Data Transfer Timing Diagrams



18.5 Frame Formats

A serial frame for the MSPIM is defined as one character of eight data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. After this follows the next data bits up to a total of eight, ending with the most or least significant bit. When a complete frame is transmitted, a new frame can follow it directly or the communication line can be set to an idle (high) state.

The UDORDn bit sets the frame format used by the USART in MSPIM mode. The receiver and transmitter use the same setting. Note that changing the setting of any of these bits corrupts all ongoing communication for both the receiver and transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A USART transmit complete interrupt then signals that the 16-bit value has been shifted out.

18.5.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting the master operating mode, setting the frame format, and enabling the transmitter and receiver. Only the transmitter can operate independently. For interrupt-driven USART operation, the global interrupt flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: The baud-rate register (UBRRn) must be “0” when the transmitter is enabled to ensure immediate initialization of the XCKn output. Unlike the normal USART operating mode, the UBRRn must then be written to the desired value after the transmitter is enabled but before the first transmission is started. Setting UBRRn to “0” before enabling the transmitter is not necessary if the initialization is done immediately after a reset because UBRRn is reset to “0”.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXCn flag can be used to check that the transmitter has completed all transfers, and the RXCn flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in registers R17:R16.

Assembly Code Example[1]

```
USART_Init:
        clr r18
        out UBRRnH,r18
        out UBRRnL,r18
        ; Setting the XCKn port pin as output, enables master mode.
        sbi XCKn_DDR, XCKn
        ; SetmsPI mode of operation and SPI data mode 0.
        ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
        out UCSRnC,r18
        ; Enable receiver and transmitter.
        ldi r18, (1<<RXENn)|(1<<TXENn)
        out UCSRnB,r18
        ; Set baud rate.
        ; IMPORTANT: The Baud Rate must be set after the transmitter is
                        enabled!
        out UBRRnH, r17
        out UBRRnL, r18
        ret
```

C Code Example[1]

```
void USART_Init(unsigned int baud)
{
        UBRRn = 0;
        /* Setting the XCKn port pin as output, enables master mode. */
        XCKn_DDR |= (1<<XCKn);
        /* SetmsPI mode of operation and SPI data mode 0. */
        UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
        /* Enable receiver and transmitter. */
        UCSRnB = (1<<RXENn)|(1<<TXENn);
        /* Set baud rate. */
        /* IMPORTANT: The Baud Rate must be set after the transmitter
                        is enabled   */
        UBRRn = baud;
}
```

Note: 1. See Section 4.2 “Code Examples” on page 7.

18.6 Data Transfer

Using the USART in MSPI mode requires the transmitter to be enabled, i.e., the TXENn bit to be set. When the transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the receiver's serial input. In both cases the XCKn is used as the transfer clock.

After initialization the USART is ready to transfer data. A data transfer is initiated by writing to UDRn. This is the case for both sending and receiving data because the transmitter controls the transfer clock. The data written to UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, UDRn must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e., if an overflow occurs, the characters last received are lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn are each not read before all transfers are completed, then byte 3 to be received is lost, not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the data register empty flag (UDREn) and the receive complete flag (RXCn). The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register R16 and the data received is available in the same register (R16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDREn flag before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXCn flag before reading the buffer and returning the value.

Assembly Code Example(1)

```
USART_MSPIM_Transfer:
        ; Wait for empty transmit buffer
        sbis UCSRnA, UDREn
        rjmp USART_MSPIM_Transfer
        ; Put data (r16) into buffer, sends the data
        out UDRn,r16
        ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
        sbis UCSRnA, RXCn
        rjmp USART_MSPIM_Wait_RXCn
        ; Get and return received data from buffer
        in r16, UDRn
        ret
```

C Code Example(1)

```
unsigned char USART_Receive( void )
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRnA & (1<<UDREn)) );
        /* Put data into buffer, sends the data */
        UDRn = data;
        /* Wait for data to be received */
        while ( !(UCSRnA & (1<<RXCn)) );
        /* Get and return received data from buffer */
        return UDRn;
}
```

Note: 1. See Section 4.2 "Code Examples" on page 7.

18.6.1 Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREn flags and corresponding interrupts in USART in MSPIM mode are identical in function to normal USART operation. However, the receiver error status flags (FEn, DORn, and PEn) are not in use and always read as "0".

18.6.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to normal USART operation.

18.7 Compatibility with AVR SPI

The USART in MSPIM mode is fully compatible with the AVR® SPI regarding:

- Master mode timing diagram
- The UCPOLn bit functionality is identical to the SPI CPOL bit
- The UCPHAn bit functionality is identical to the SPI CPHA bit
- The UDORDn bit functionality is identical to the SPI DORD bit

However, because the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer.
- The USART in MSPIM mode receiver includes an additional buffer level.
- The SPI WCOL (write collision) bit is not included in USART in MSPIM mode.
- The SPI double-speed mode (SPI2X) bit is not included. However, the same effect is achieved by choosing a corresponding UBRRn setting.
- Interrupt timing is not compatible.
- Pin control differs due to the master only operation of the USART in MSPIM mode.

A comparison of the USART in MSPIM mode and the SPI pins is shown in Table 18-3.

Table 18-3. Comparison of USART in MSPIM Mode and SPI Pins

USART_MSPIM	SPI	Comment
TxDn	MOSI	Master out, only
RxDn	MISO	Master in, only
XCKn	SCK	Functionally identical
(N/A)	$\overline{\text{SS}}$	Not supported by USART in MSPIM

18.8 Register Description

The following section describes the registers used for SPI operation using the USART.

18.8.1 UDRn – USARTmsPIM I/O Data Register

The function and bit description of the USART data register (UDRn) in MSPI mode is identical to normal USART operation (see Section 17.12.1 "UDRn – USART I/O Data Register" on page 156).

18.8.2 UCSRnA – USARTmsPIM Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	**TXCn**	**UDREn**	**–**	**–**	**–**	**–**	**–**	**UCSRnA**
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 – RXCn: USART Receive Complete**

This flag is set when there is unread data in the receive buffer. The flag is cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer is flushed, causing the flag to become “0”.

The flag can be used to generate a receive complete interrupt (see RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag is set when the entire frame in the transmit shift register has been shifted out and there is no new data in the transmit buffer (UDRn). The flag is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a “1” to its bit location.

This flag can generate a transmit complete interrupt (see TXCIEn bit).

- **Bit 5 – UDREn: USART Data Register Empty**

This flag indicates the transmit buffer (UDRn) is ready to receive new data. If the flag is “1”, the buffer is empty and ready to be written. The flag is set after a reset to indicate that the transmitter is ready.

The flag can generate a data register empty interrupt (see UDRIEn bit).

- **Bits 4:0 – Reserved Bits in MSPI Mode**

In MSPI mode these bits are reserved for future use. For compatibility with future devices, these bits must be written as “0”.

18.8.3 UCSRnB – USARTmsPIM Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIE	RXENn	TXENn	–	–	–	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable**

Writing this bit to “1” enables interrupt on the RXCn flag. A USART receive complete interrupt will be generated only if the RXCIEn bit is written to “1”, the global interrupt flag in SREG is written to “1”, and the RXCn bit is set.

- **Bit 6 – TXCIEn: TX Complete Interrupt Enable**

Writing this bit to “1” enables interrupt on the TXCn flag. A USART transmit complete interrupt will be generated only if the TXCIEn bit is written to “1”, the global interrupt flag in SREG is written to “1”, and the TXCn bit is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to “1” enables interrupt on the UDREn flag. A data register empty interrupt is generated only if the UDRIEn bit is written to “1”, the global interrupt flag in SREG is written to “1”, and the UDREn bit is set.

- **Bit 4 – RXENn: Receiver Enable**

Writing this bit to “1” enables the USART receiver in MSPIM mode. When enabled, the receiver overrides normal port operation for the RxDn pin.

Disabling the receiver flushes the receive buffer.

Enabling the receiver only and leaving the transmitter disabled has no meaning in MSPI mode, because only master mode is supported and the transmitter controls the transfer clock.

- **Bit 3 – TXENn: Transmitter Enable**

Writing this bit to “1” enables the USART transmitter. When enabled, the transmitter overrides normal port operation for the TxDn pin.

Disabling the transmitter does not become effective until ongoing and pending transmissions are completed, i.e., when the transmit shift register and transmit buffer register do not contain data to be transmitted. When disabled, the transmitter no longer overrides the TxDn pin.

- **Bits 2:0 – Reserved Bits in MSPI Mode**

In MSPI mode these bits are reserved for future use. For compatibility with future devices, these bits must be written as “0”.

18.8.4 UCSRnC – USARTmsPIM Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	–	–	–	UDORDn	UCPHAn	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – UMSELn[1:0]: USART Mode Select**

These bits select the operating mode of the USART as shown in Table 18-4. The MSPIM is enabled when both UMSEL bits are set to “1”.

Table 18-4. UMSELn Bit Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(reserved)
1	1	Master SPI (MSPIM)

For a full description of normal USART operation, see Section 17.12.4 “UCSRnC – USART Control and Status register C” on page 159.

Bits UDORDn, UCPHAn, and UCPOLn may be set in the same write operation where the MSPIM is enabled.

- **Bits 5:3 – Reserved Bits in MSPI Mode**

In MSPI mode these bits are reserved for future use. For compatibility with future devices, these bits must be written as “0”.

- **Bit 2 – UDORDn: Data Order**

When set, the LSB of the data word is transmitted first.

When cleared, the MSB of the data word is transmitted first.

For more information, see Section 18.5 “Frame Formats” on page 163.

- **Bit 1 – UCPHAn: Clock Phase**

This bit determines if data is sampled on the leading (first), or trailing (last) edge of XCKn.

For more information, see Section 18.4 “SPI Data Modes and Timing” on page 163.

- **Bit 0 – UCPOLn: Clock Polarity**

This bit sets the polarity of the XCKn clock. The combination of UCPOLn and UCPHAn bits determine the timing of the data transfer.

For more information, see Table 18-2 on page 163.

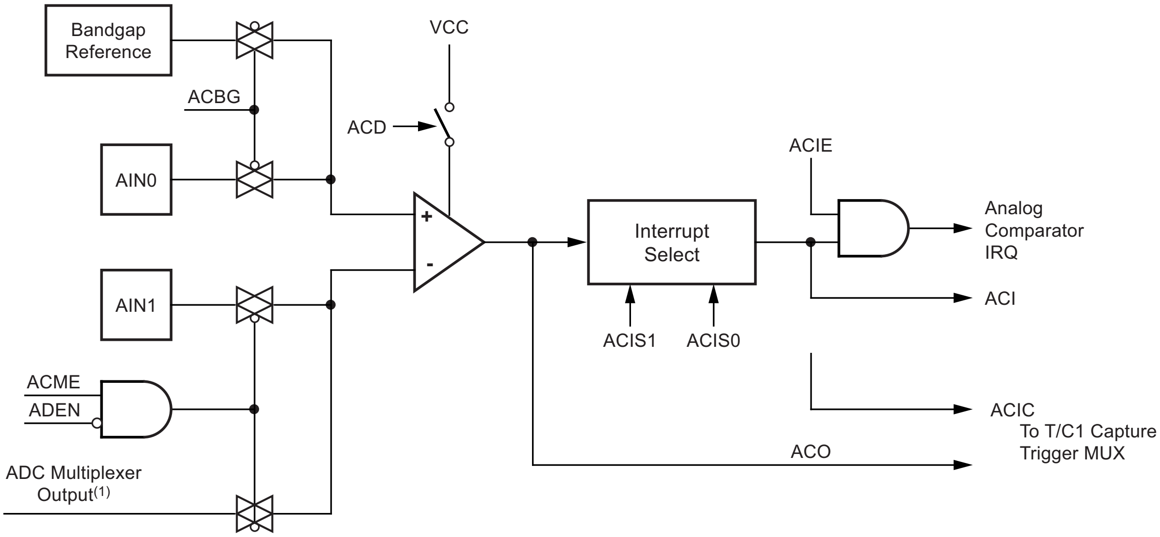
18.8.5 UBRRnL and UBRRnH – USARTmsPIM Baud Rate Registers

The function and bit description of the baud rate registers in MSPI mode are identical to normal USART operation (see Section 17.12.6 “UBRRnL and UBRRnH – USART Baud Rate Registers” on page 161).

19. Analog Comparator

The analog comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the analog comparator output (ACO) is set. The comparator can trigger a separate interrupt that excludes the analog comparator. The user can select interrupt triggering on comparator output rise, fall, or toggle. A block diagram of the comparator and its related logic is shown in Figure 19-1.

Figure 19-1. Analog Comparator Block Diagram



Note: 1. See Table 19-1 on page 169.

For information about pin placements, see Figure 1-1 on page 3.

The ADC power reduction bit (PRADC) must be disabled in order to use the ADC input multiplexer. This is done by clearing the PRADC bit in the power reduction register (PRR). For more information, see Section 8.4.2 “PRR – Power Reduction Register” on page 37.

19.1 Analog Comparator Multiplexed Input

When the analog to digital converter (ADC) is configured as a single-ended input channel, it is possible to select any of the ADC[11:0] pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input; the ADC must therefore 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 “0”), MUX[3:0], in ADMUX select the input pin to replace the negative input to the analog comparator as shown in Table 19-1. If ACME is cleared or ADEN is set, AIN1 is applied to the analog comparator negative input.

Table 19-1. Analog Comparator Multiplexed Input

ACME	ADEN	Analog Comparator Negative Input
0	X	AIN1
1	0	ADC multiplexer. See Table 20-4 on page 183
1	1	AIN1

19.2 Register Description

19.2.1 ACSRA – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	ACD	ACBG	ACO	ACI	ACIE	ACIC	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 reduces 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 Band-gap Select**

When this bit is set, a fixed internal band-gap reference voltage replaces the positive input to the analog comparator. When this bit is cleared, AIN0 is applied to the positive input of the analog comparator.

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

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

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

This bit is set by the 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 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the analog comparator. In this case, the comparator output is connected directly to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 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/Counter1 input capture interrupt, the ICIE1 bit in the timer interrupt mask register (TIMSK) must be set.

- **Bits 1:0 – ACIS[1:0]: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events trigger the analog comparator interrupt. The different settings are shown in Table 19-2.

Table 19-2. ACIS1/ACIS0 Settings

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

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

19.2.2 ACSRB – Analog Comparator Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	HSEL	HLEV	ACLP	–	ACCE	ACME	ACIRS1	ACIRS0	ACSRB
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	

- **Bit 7 – HSEL: Hysteresis Select**

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

- **Bit 6 – HLEV: Hysteresis Level**

When enabled via the HSEL bit, the level of hysteresis can be set using the HLEV bit as shown in Table 19-3.

Table 19-3. Selecting Level of Analog Comparator Hysteresis

HSEL	HLEV	Hysteresis of Analog Comparator
0	X	Not enabled
1	0	20mV
	1	50mV

- **Bit 5 – ACLP**

This bit is reserved for QTouch and always writes as “0”.

- **Bit 4 – Reserved**

This bit is reserved and always reads as “0”.

- **Bit 3 – ACCE**

This bit is reserved for QTouch and always writes as “0”.

- **Bit 2 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is “0”), 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 19.1 “Analog Comparator Multiplexed Input” on page 169.

- **Bit 1 – ACIRS1**

This bit is reserved for QTouch and always writes as “0”.

- **Bit 0 – ACIRS0**

This bit is reserved for QTouch and always writes as “0”.

19.2.3 DIDR0 – Digital Input Disable Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	AIN1D	AIN0D	AREFD	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 2:1 – AIN1D, AIN0D: AIN1 and AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN register bit always reads as “0” when this bit is set. When used as an analog input but not required as a digital input, the power consumption in the digital input buffer can be reduced by writing this bit to logic one.

20. Analog to Digital Converter

20.1 Features

- 10-bit resolution
- 1 LSB integral nonlinearity
- ±2 LSB absolute accuracy
- 13 clocks conversion time
- 15kSPS at maximum resolution
- 12 multiplexed single-ended input channels
- Temperature sensor input channel
- Optional left adjustment for ADC result readout
- 0 - V_{CC} ADC input voltage range
- 1.1V ADC reference voltage
- Free-running or single-conversion mode
- ADC start conversion by auto triggering on interrupt sources
- Interrupt on ADC conversion complete
- Sleep mode noise canceler

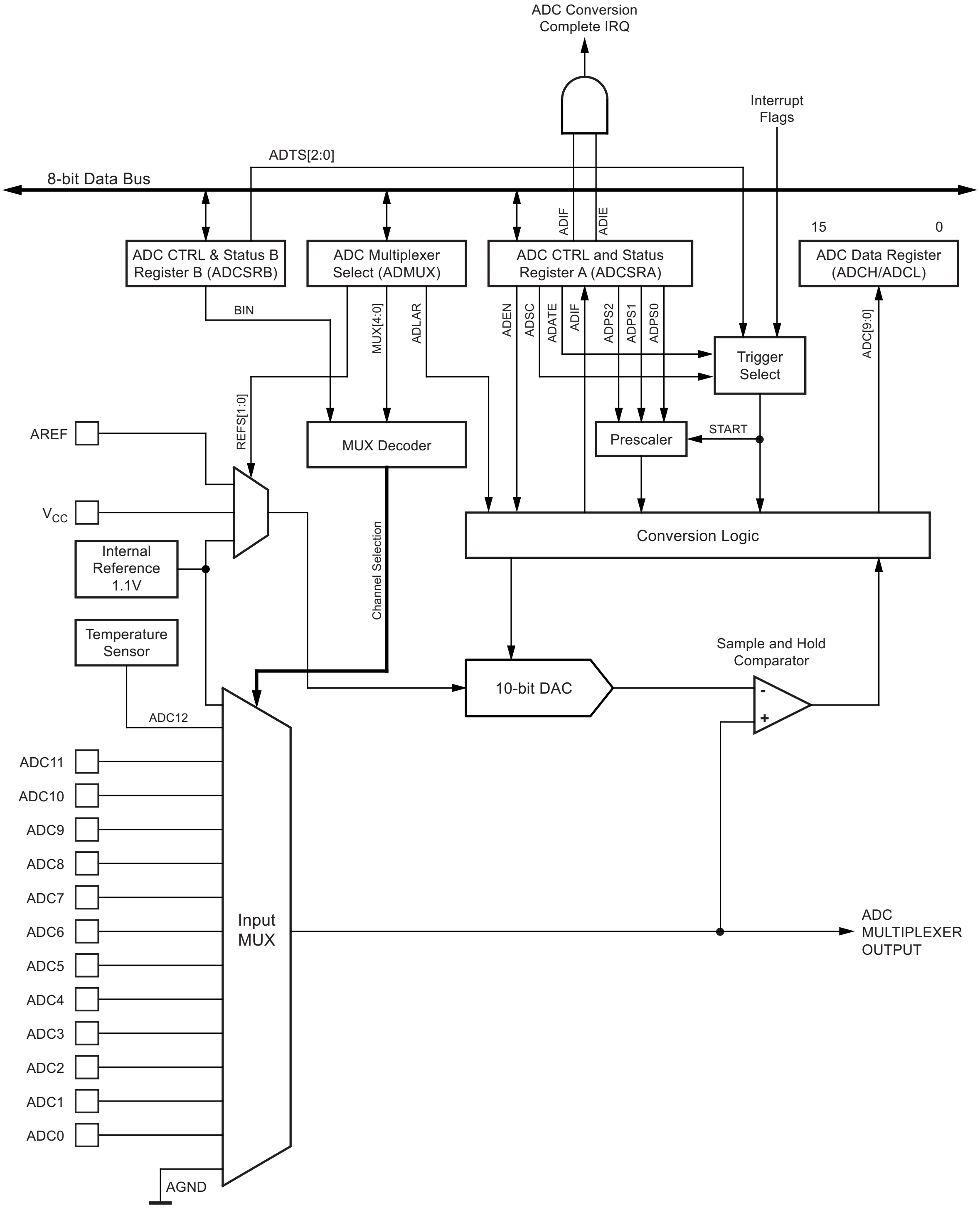
20.2 Overview

The Atmel® ATtiny1634 features a 10-bit, successive approximation analog-to-digital converter (ADC). The ADC is wired to a 13-channel analog multiplexer, which allows the ADC to measure the voltage at 12 single-ended input pins or from one internal single-ended voltage channel coming from the internal temperature sensor. Single-ended voltage inputs are referred 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 20-1 on page 173.

Internal reference voltage of nominally 1.1V is provided on-chip. 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.

Figure 20-1. Analog to Digital Converter Block Schematic



20.3 Operation

To enable use of ADC, the power reduction bit (PRADC) in the power reduction register must be disabled. This is done by clearing the PRADC bit. For more information, see Section 8.4.2 “PRR – Power Reduction Register” on page 37.

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

The ADC converts an analog input voltage to a 10-bit digital value using successive approximation. The minimum value represents GND and the maximum value represents the reference voltage. The ADC voltage reference is selected by writing the REFS[1:0] bits in the ADMUX register. Alternatives are the V_{CC} supply pin, the AREF pin, and the internal 1.1V voltage reference.

The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the ADC input pins can be selected as single-ended inputs to the ADC.

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

If the result is left-adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH only. 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 triggers even if the result is lost.

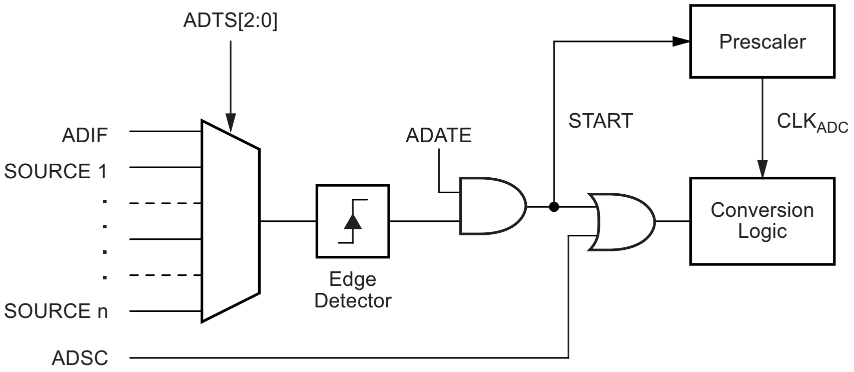
20.4 Starting a Conversion

Make sure the ADC is powered by clearing the ADC power reduction bit (PRADC) in the power reduction register (PRR). For more information, see Section 8.4.2 “PRR – Power Reduction Register” on page 37.

A single conversion is started by writing a logic one to the ADC start conversion bit (ADSC). This bit stays high as long as the conversion is in progress and is cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC finishes 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 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 is still set when the conversion completes, a new conversion is not started. If another positive edge occurs on the trigger signal during conversion, the edge is ignored. Note that an interrupt flag is 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 20-2. ADC Auto Trigger Logic



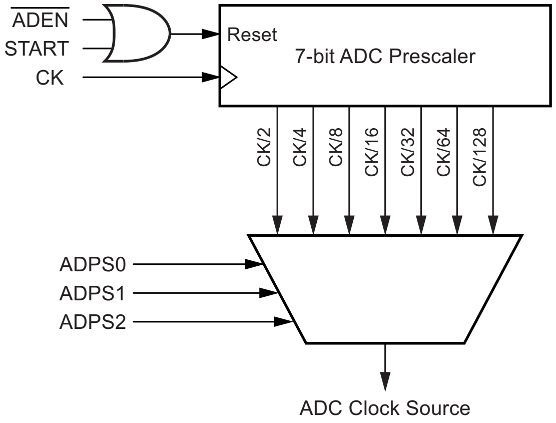
Using the ADC interrupt flag as a trigger source forces the ADC to start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free-running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logic one to the ADSC bit in ADCSRA. In this mode the ADC performs successive conversions regardless 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 "1". ADSC can also be used to determine if a conversion is in progress. The ADSC bit is read as "1" during a conversion and does not depend on how the conversion was started.

20.5 Prescaling and Conversion Timing

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

Figure 20-3. ADC Prescaler

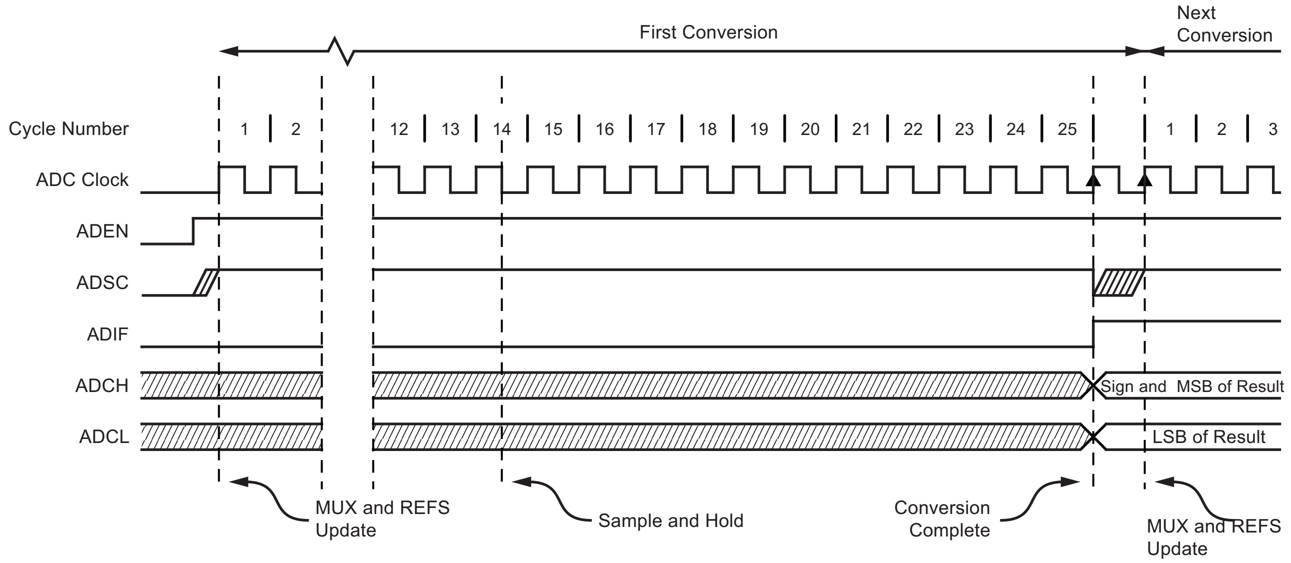


The ADC module contains a prescaler, as illustrated in Figure 20-3, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in 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 next rising edge of the ADC clock cycle.

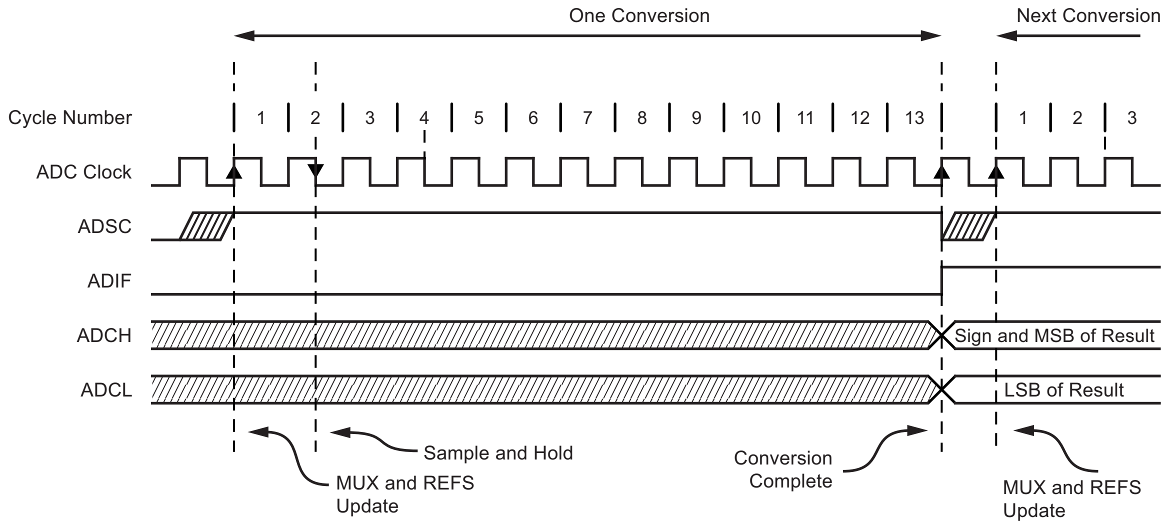
A normal conversion takes 13 ADC clock cycles as summarized in Table 20-1 on page 177. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) and takes 25 ADC clock cycles in order to initialize the analog circuitry as shown in Figure 20-4 below.

Figure 20-4. ADC Timing Diagram, First Conversion (Single-Conversion Mode)



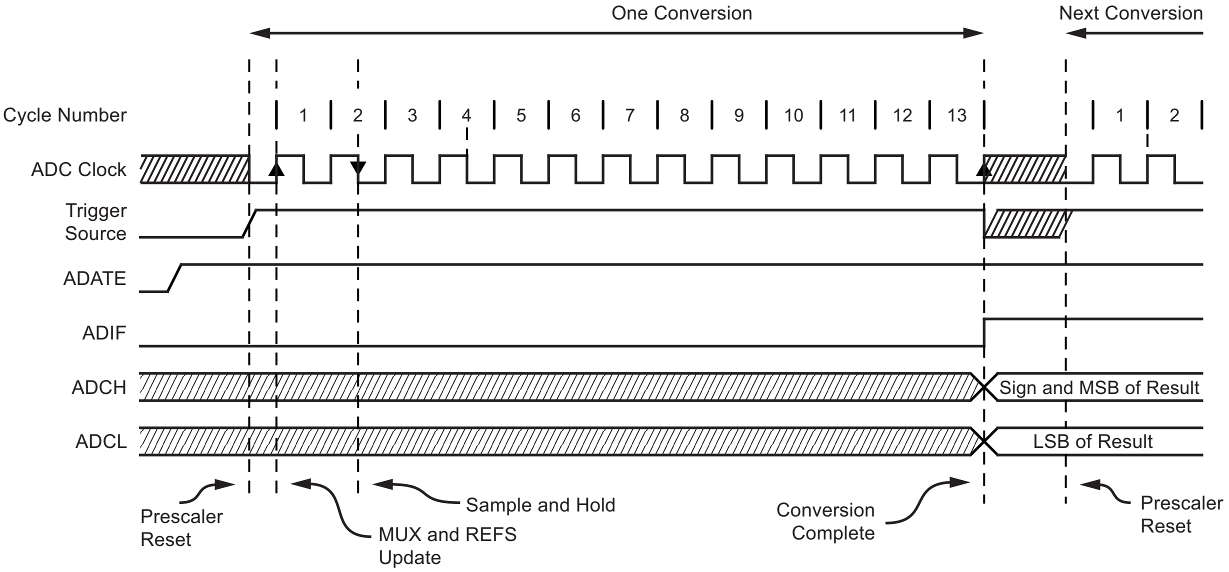
The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of a first conversion (see Figure 20-5). When a conversion is complete, the result is written to the ADC data registers and ADIF is set. In single-conversion mode ADSC is cleared simultaneously. The software may then set ADSC again and a new conversion is initiated on the first rising ADC clock edge.

Figure 20-5. ADC Timing Diagram, Single Conversion



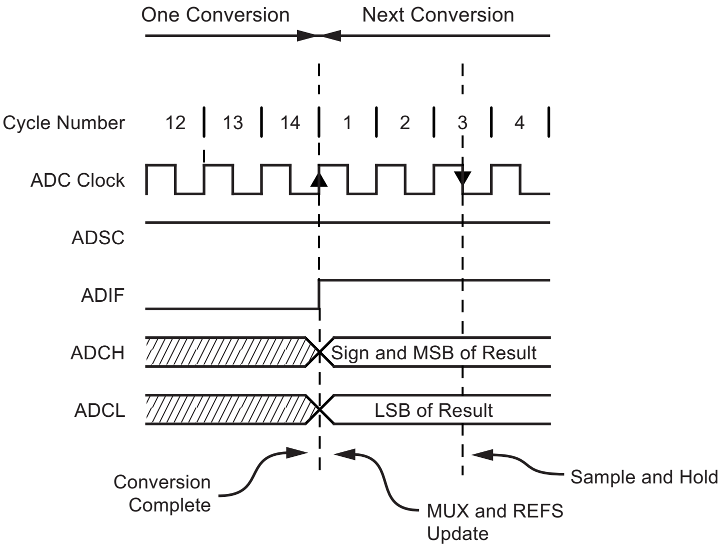
When auto triggering is used, the prescaler is reset when the trigger event occurs as shown in Figure 20-6 on page 177. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

Figure 20-6. ADC Timing Diagram, Auto Triggered Conversion



In free-running mode a new conversion will be started immediately after the conversion completes while ADSC remains high (see Figure 20-7).

Figure 20-7. ADC Timing Diagram, Free-Running Conversion



For a summary of conversion times, see Table 20-1.

Table 20-1. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions	1.5	13
Auto triggered conversions	2	13.5
Free-running conversion	2.5	14

20.6 Changing Channel or Reference Selection

The MUX[3:0] and REFS[1:0] 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 there is 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. For this reason, the user is 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 to check what conversion is affected by the new settings.

If both ADATE and ADEN are written to “1”, an interrupt event can occur at any time. If the ADMUX register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

- When ADATE or ADEN are cleared
- During conversion, at least one ADC clock cycle after the trigger event
- After a conversion, before the interrupt flag used as the trigger source is cleared

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

20.6.1 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 “1” 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 “1” to ADSC. However, the simplest method is to wait for the first conversion to complete and then change the channel selection. Because the next conversion has already started automatically, the next result reflects the previous channel selection. Subsequent conversions reflect the new channel selection.

20.6.2 ADC Voltage Reference

The ADC reference voltage (V_{REF}) indicates the conversion range for the ADC. Single-ended channels that exceed V_{REF} result in codes close to 0x3FF. V_{REF} can be selected as V_{CC}, internal 1.1V reference, or external AREF pin. The internal 1.1V reference is generated from the internal band-gap reference (V_{BG}) through an internal amplifier.

The first ADC conversion result after switching reference voltage source may be inaccurate; the user is advised to discard this result.

20.7 ADC Noise Canceler

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

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

Note that the ADC is not automatically turned off when entering sleep modes other than idle mode and ADC noise reduction mode. The user is advised to write “0” to ADEN before entering such sleep modes to avoid excessive power consumption.

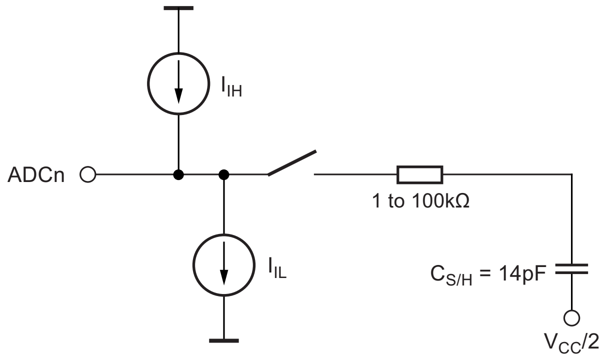
20.8 Analog Input Circuitry

The analog input circuitry for single-ended channels is illustrated in Figure 20-8. 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 or not. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

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

In order to avoid distortion from unpredictable signal convolution, signal components higher than the Nyquist frequency (f_{ADC}/2) should not be present. The user is advised to remove high-frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 20-8. Analog Input Circuitry



Note: The capacitor in the figure depicts the total capacitance, including the sample/hold capacitor and any stray or parasitic capacitance inside the device. The value given is worst case.

20.9 Noise Canceling Techniques

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

- Keep analog signal paths as short as possible.
- Make sure analog tracks run over the analog ground plane.
- Keep analog tracks well away from high-speed switching digital tracks.
- If any port pin is used as a digital output, it should never switch while a conversion is in progress.
- Place bypass capacitors as close to V_{CC} and GND pins as possible.

Where high ADC accuracy is required, it is recommended to use ADC noise reduction mode as described in Section 20.7 “ADC Noise Canceler” on page 178. This is especially the case when system clock frequency is above 1MHz or when the ADC is used for reading the internal temperature sensor as described in Section 20.12 “Temperature Measurement” on page 182. A good system design with properly placed external bypass capacitors reduces the need for using ADC noise reduction mode.

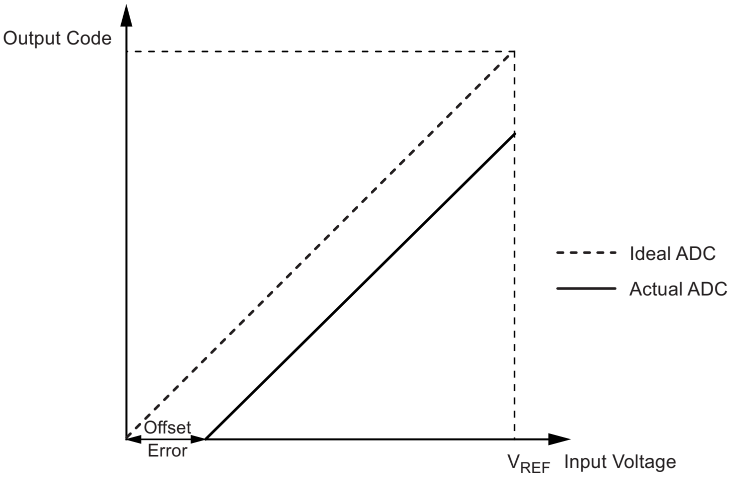
20.10 ADC Accuracy Definitions

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

Several parameters describe the deviation from the ideal behavior as follows:

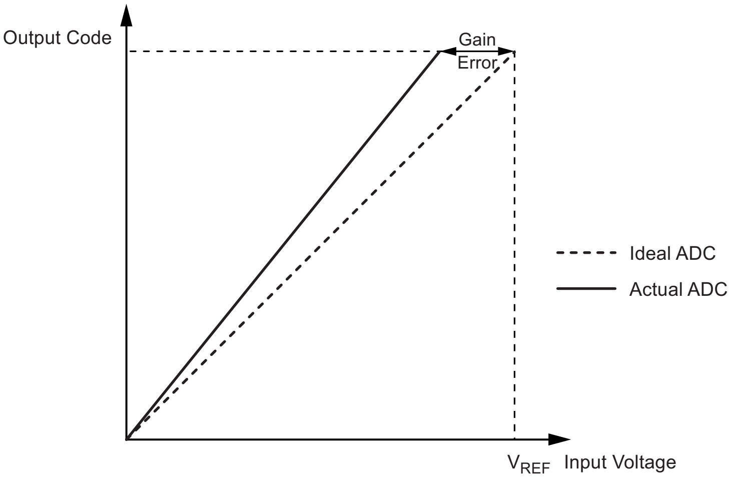
- Offset: the deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 20-9. Offset Error



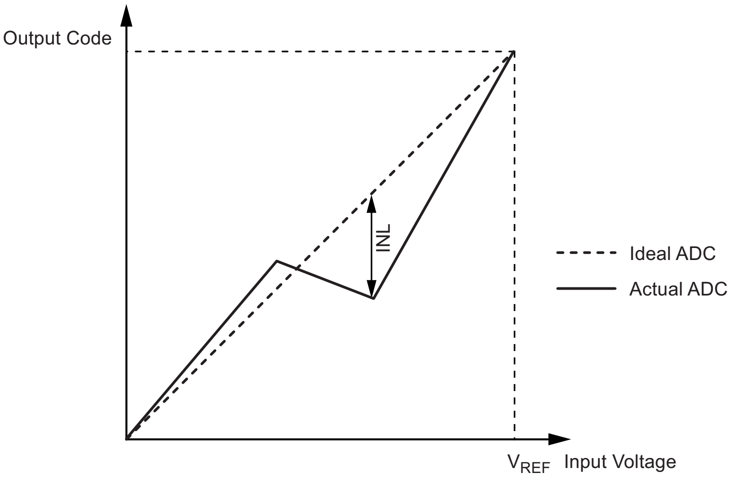
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 20-10. Gain Error



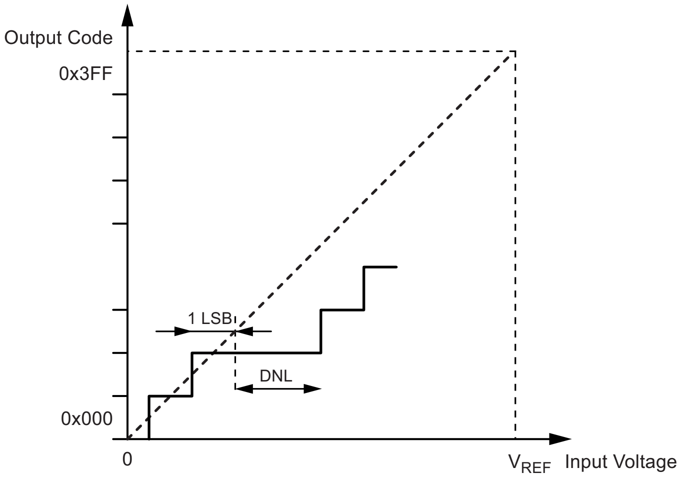
- Integral nonlinearity (INL): after adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 20-11. Integral Nonlinearity (INL)



- Differential nonlinearity (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 20-12. Differential Nonlinearity (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) codes 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, nonlinearity, and quantization error. Ideal value: ±0.5 LSB.

20.11 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC data registers (ADCL, ADCH).

For single-ended conversion, the result is

$$\mathrm{ADC} = \frac{V_{IN} \times 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 20-3 on page 183 and Table 20-4 on page 183). 0x000 represents analog ground and 0x3FF represents the selected reference voltage minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

20.12 Temperature Measurement

Temperature measurement is based on an on-chip sensor, coupled to a single-ended ADC channel. The temperature sensor is enabled when channel ADC12 is selected from the ADMUX register. When measuring temperature, the internal voltage reference must be selected as the ADC reference source. When 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 temperature as shown in Table 20-2. The sensitivity is approximately 1 LSB/°C and the accuracy depends on the method of user calibration. The temperature sensor should be calibrated by firmware in order to achieve reasonable accuracy. Typically, the measurement accuracy after a single temperature calibration is ±10°C assuming calibration at room temperature. Better accuracies are achieved by using two temperature points for calibration.

Table 20-2. Temperature versus Sensor Output Voltage (Typical)

Temperature	–40°C	+25°C	+85°C
ADC	235 LSB	300 LSB	360 LSB

The values described in Table 20-2 are typical values. Due to process variation, however, the output voltage of the temperature sensor varies from one chip to another. To achieve more accurate results, temperature measurements can be calibrated in the application software. The software calibration can be done using the equation:

$T = k \times [(ADCH << 8) \mid ADCL] + T_{OS}$

where ADCH and ADCL are the ADC data registers, k is the fixed slope coefficient, and T_{OS} is the temperature sensor offset. Typically, k is very close to 1.0 and in single-point calibration the coefficient may be omitted.

20.13 Register Description

20.13.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	**REFS1**	**REFS0**	**REFEN**	**ADC0EN**	**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	

- **Bits 7:6 – REFS[1:0]: Reference Selection Bits**

These bits select the voltage reference for the ADC as shown in Table 20-3 on page 183.

Table 20-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	V_{CC} used as analog reference, disconnected from PA0 (AREF)
0	1	External voltage reference at PA0 (AREF) pin
1	0	Internal 1.1V voltage reference
1	1	Reserved

If these bits are changed during a conversion, the change does not go into effect until this conversion is complete (ADIF in ADCSR is set). Also note that when these bits are changed, the next conversion takes 25 ADC clock cycles.

It is recommended to force the ADC to perform a long conversion when changing multiplexer or voltage reference settings. This can be done by first turning off the ADC, then changing reference settings, and then turning on the ADC. Alternatively, the first conversion results after changing reference settings should be discarded.

Internal voltage reference options should not be used if an external voltage is being applied to the AREF pin.

- **Bit 5 – REFEN**

This bit is reserved for QTouch and always writes as “0”.

- **Bit 4 – ADC0EN**

This bit is reserved for QTouch and always writes as “0”.

- **Bits 3:0 – MUX[3:0]: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC as shown in Table 20-4. Selecting the channel ADC12 enables the temperature measurement (see Table 20-4).

Table 20-4. Single-Ended Input channel Selections.

MUX[3:0]	Single-Ended Input	Pin
0000	ADC0	PA3
0001	ADC1	PA4
0010	ADC2	PA5
0011	ADC3	PA6
0100	ADC4	PA7
0101	ADC5	PB0
0110	ADC6	PB1
0111	ADC7	PB2
1000	ADC8	PB3
1001	ADC9	PC0
1010	ADC10	PC1
1011	ADC11	PC2
1100	Ground	GND
1101	Internal 1.1V reference(1)	(internal)
1110	Temperature sensor(2)	(internal)
1111	Reserved	Not connected

Notes: 1. After switching to internal voltage reference, the ADC requires a settling time of 1ms before measurements are stable. Conversions starting before this may not be reliable. The ADC must be enabled during the settling time.

2. See Section 20.12 “Temperature Measurement” on page 182.

If these bits are changed during a conversion, the change does not go into effect until the conversion is complete (ADIF in ADCSRA is set).

20.13.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	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 “1” enables the ADC. Writing it to “0” turns ADC off. Turning the ADC off while a conversion is in progress terminates this conversion.

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

In single-conversion mode write this bit to “1” to start each conversion. In free-running mode write this bit to “1” 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. It takes 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC reads as “1” as long as a conversion is in progress. When the conversion is complete, it returns to “0”. Writing “0” to this bit has no effect.

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

When this bit is written to “1”, auto triggering of the ADC is enabled. The ADC starts 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 the hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logic one to the flag. Beware that doing a read-modify-write on ADCSRA may cause a pending interrupt to be disabled. This also applies if the SBI instruction is used.

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

When this bit is written to “1” and the I bit in SREG is set, the ADC conversion complete interrupt is activated.

- **Bits 2:0 – ADPS[2:0]: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 20-5. ADC Prescaler Selections

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

20.13.3 ADCL and ADCH – ADC Data Register

20.13.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
0x01 (0x21)	–	–	–	–	–	–	ADC9	ADC8	ADCH
0x00 (0x20)	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	

20.13.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
0x01 (0x21)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x00 (0x20)	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. If the result is left-adjusted and no more than 8-bit precision is required, it is therefore sufficient to read ADCH. Otherwise, ADCL must be read first and then ADCH.

The ADLAR bit in ADCSRB and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left-adjusted. If ADLAR is cleared (default), the result is right-adjusted.

- **ADC[9:0]: ADC Conversion Result**

These bits represent the result from the conversion as detailed in Section 20.11 “ADC Conversion Result” on page 182.

20.13.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	VDEN	VDPD	–	–	ADLAR	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – VDEN**

This bit is reserved for QTouch and always writes as “0”.

- **Bit 6 – VDPD**

This bit is reserved for QTouch and always writes as “0”.

- **Bits 5:4 – Res: Reserved Bits**

These are reserved bits in Atmel® ATtiny1634. For compatibility with future devices, always write these bits to “0”.

- **Bit 3 – ADLAR: ADC Left-Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC data register. Write “1” to ADLAR to left-adjust the result. Otherwise, the result is right-adjusted. Changing the ADLAR bit affects the ADC data register immediately, regardless of any ongoing conversions. For a complete description of this bit, see Section 20.13.3 “ADCL and ADCH – ADC Data Register” on page 185.

- **Bits 2:0 – ADTS[2:0]: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to “1”, the value of these bits selects what source triggers an ADC conversion. If ADATE is cleared, the ADTS[2:0] settings have no effect. A conversion is 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 generates a positive edge on the trigger signal. If ADEN in ADCSRA is set, this starts a conversion. Switching to free-running mode (ADTS[2:0]=0) does not cause a trigger event even if the ADC interrupt flag is set.

Table 20-6. ADC Auto Trigger Source Selections

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

20.13.5 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
(0x60)	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	AIN1D	AIN0D	AREFD	**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 7:3 – ADC4D:ADC0D: ADC[4:0] Digital Input Disable**

When a bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN register bit always reads as “0” when this bit is set. When an analog signal is applied to the ADC[7:0] 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.

20.13.6 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
(0x61)	–	–	–	–	ADC8D	ADC7D	ADC6D	ADC5D	**DIDR1**
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 3:0 – ADC8D:ADC5D: ADC[8:5] Digital Input Disable**

When a bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN register bit always reads as “0” when this bit is set. When an analog signal is applied to the ADC[8:5] 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.

20.13.7 DIDR2 – Digital Input Disable Register 2

Bit	7	6	5	4	3	2	1	0	
(0x62)	–	–	–	–	–	**ADC11D**	**ADC10D**	**ADC9D**	**DIDR2**
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2:0 – ADC11D:ADC9D: ADC[11:9] Digital Input Disable**

When a bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN register bit always reads as “0” when this bit is set. When an analog signal is applied to the ADC[11:9] 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.

21. debugWIRE On-chip Debug System

21.1 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 nonvolatile memories

21.2 Overview

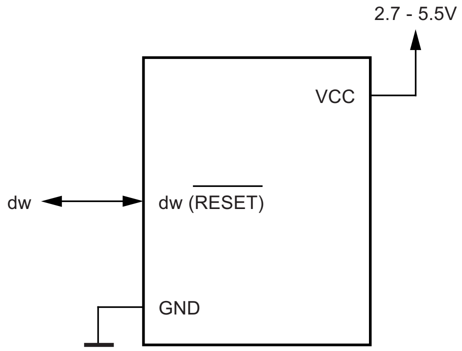
The debugWIRE on-chip debug system uses a one-wire bidirectional interface to control the program flow, execute AVR® instructions in the CPU, and program the different nonvolatile memories.

21.3 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) bidirectional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 21-1 shows the schematic of a target MCU with debugWIRE enabled and the emulator connector. The system clock is not affected by debugWIRE and is always the clock source selected by the CKSEL fuses.

Figure 21-1. The debugWIRE Setup



When designing a system where debugWIRE is used, the following must be observed:

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

21.4 Software Break Points

debugWIRE supports program memory break points by the AVR® break instruction. Setting a break point in AVR Studio® inserts a BREAK instruction in the program memory. The instruction replaced by the BREAK instruction is stored. When program execution is continued, the stored instruction is 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 reprogrammed each time a break point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of break points thus reduces the Flash data retention. Devices used for debugging purposes should not be shipped to end customers.

21.5 Limitations of debugWIRE

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

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

The debugWIRE interface is asynchronous, which means that the debugger needs to synchronize to the system clock. If the system clock is changed by software (e.g., by writing CLKPS bits), communication via debugWIRE may fail. In addition, clock frequencies below 100kHz may cause communication problems.

A programmed DWEN fuse enables some parts of the clock system to run in all sleep modes. This increases the power consumption while in sleep. The DWEN fuse should therefore be disabled when debugWIRE is not used.

21.6 Register Description

The following section describes the registers used with the debugWIRE.

21.6.1 DWDR – debugWIRE Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	**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 via the debugWIRE and therefore cannot be used as a general purpose register in normal operations.

22. Self Programming

22.1 Features

- Self programming enables MCU to erase, write, and reprogram application memory
- Efficient read-modify-write support
- Lock bits allow application memory to be securely closed for further access

22.2 Overview

The device provides a self programming mechanism for downloading and uploading the program code by the MCU itself. Self programming can use any available data interface and associated protocol to read code and write (program) that code into program memory.

22.3 Lock Bits

Program memory can be protected from internal or external access (see Section 23.1 "Lock Bits" on page 196).

22.4 Self Programming the Flash

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 "4-Page Erase" command or between a 4-page erase and a page write operation:

1. Either, fill the buffer before a 4-page erase:
   a. Fill temporary page buffer
   b. Perform a 4-page erase
   c. Perform a page write
4. Or, fill the buffer after a 4-page erase:
   a. Perform a 4-page erase
   b. Fill temporary page buffer
   c. Perform a page write

The "4-Page Erase" command erases four program memory pages at the same time. If only part of this section needs to be changed, the rest must be stored before the erase and then re-written.

The temporary page buffer can be accessed in a random sequence.

The SPM instruction is disabled by default but it can be enabled by programming the SELFPRGEN fuse (to "0").

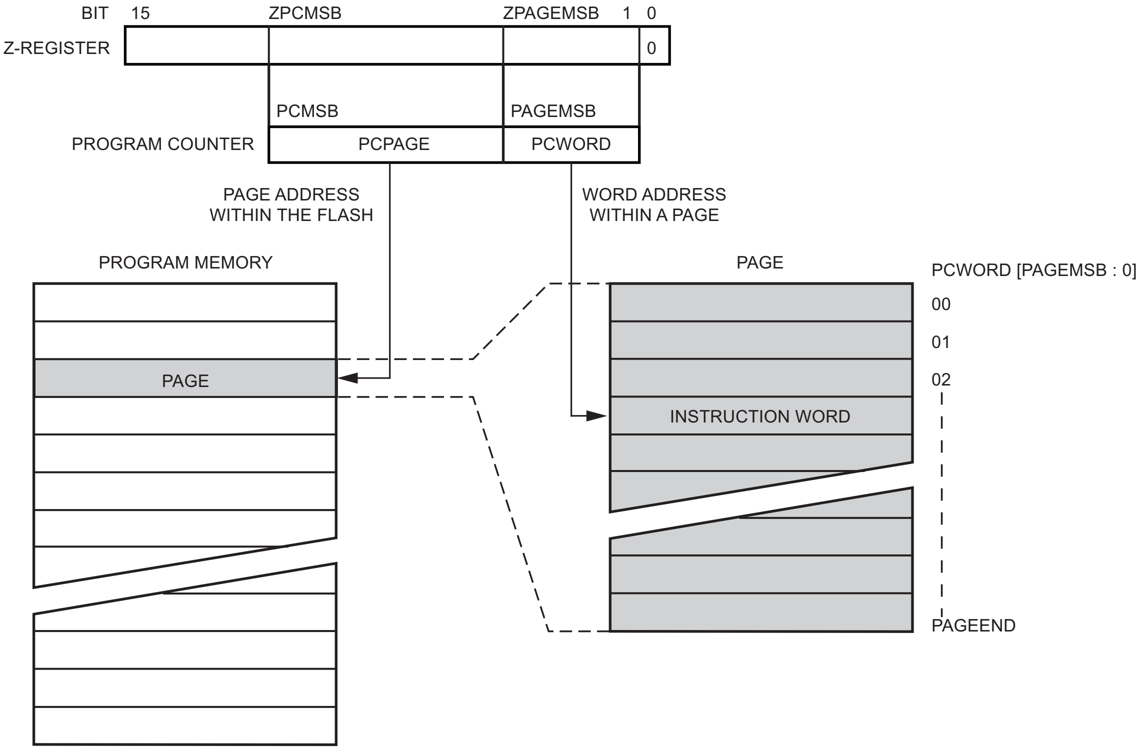
22.4.1 Addressing the Flash During Self Programming

The Z pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

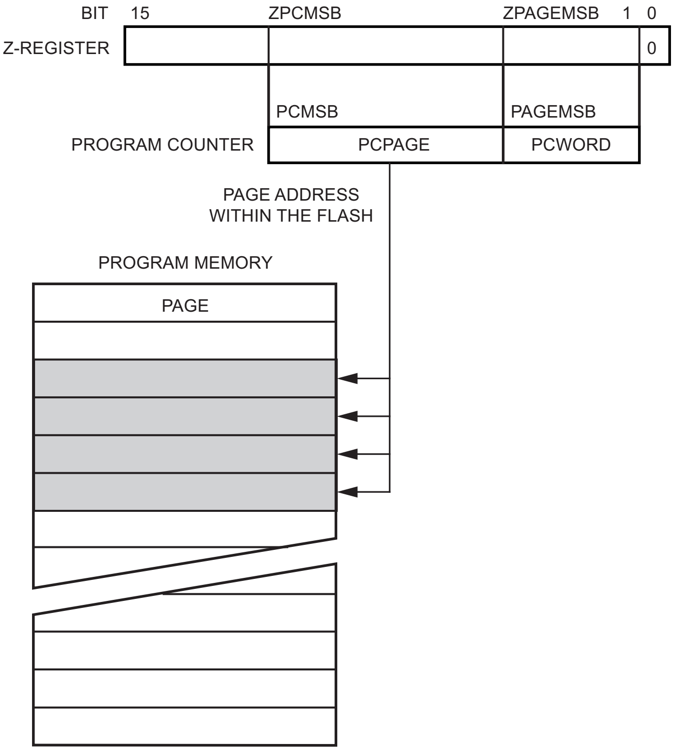
Because the Flash is organized in pages (see Table 24-1 on page 202), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, addresses the words within a page while the most significant bits address the pages. This is shown in Figure 22-1.

Figure 22-1. Addressing the Flash During SPM Load and Write Operations



The “4-Page Erase” command addresses several program memory pages simultaneously as shown in Figure 22-2.

Figure 22-2. Addressing the Flash During SPM 4-Page Erase



Variables used in the figures above are explained in Table 22-1.

Table 22-1. Variables Used in Flash Addressing

Variable	Description
PCPAGE	Program counter page address. Selects the program memory page for page load and page write commands. Selects a block of program pages for the 4-page erase operation. See Table 24-1 on page 202.
PCMSB	The most significant bit of the program counter. See Table 24-1 on page 202.
ZPCMSB	The bit in the Z register that is mapped to PCMSB. Because Z[0] is not used, ZPCMSB = PCMSB + 1. Z register bits above ZPCMSB are ignored.
PCWORD	Program counter word address. Selects the word within a page. This is used for filling the temporary buffer and must be “0” during page write operations. See Table 24-1 on page 202.
PAGEMSB	The most significant bit used to address the word within one page.
ZPAGEMSB	The bit in the Z register that is mapped to PAGEMSB. Because Z[0] is not used, ZPAGEMSB = PAGEMSB + 1.

Note that 4-page erase and page write operations address memory independently. The software must therefore ensure the page write command addresses a page previously erased by the “4-Page Erase” command.

Although the least significant bit of the Z register (Z0) should be “0” for SPM, it should be noted that the LPM instruction addresses the Flash byte-by-byte and uses Z0 as a byte select bit.

Once a programming operation is initiated, the address is latched and the Z pointer can be used for other operations.

22.4.2 4-Page Erase

This command erases four pages of program memory. To execute 4-page erase:

- Set up the address in the Z pointer.
- Write “00000011” to SPMCSR.
- Execute an SPM instruction 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. PCPAGE[1:0] is ignored, as are other bits in the Z pointer.

The CPU is stopped during the 4-page erase operation.

22.4.3 Page Load

To write an instruction word:

- Set up the address in the Z pointer.
- Set up the data in R1:R0.
- Write “00000001” to SPMCSR.
- Execute an SPM instruction 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.

22.4.4 Page Write

To execute page write:

- Set up the address in the Z pointer.
- Write “00000101” to SPMCSR.
- Execute an SPM instruction 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 “0” during this operation.

The CPU is stopped during the page write operation.

22.4.5 SPMCSR Cannot Be Written when EEPROM Is Programmed

Note that an EEPROM write operation blocks all software programming to Flash. Reading fuses and lock bits from software is also prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in EECR and verifies that it is cleared before writing to SPMCSR.

22.5 Preventing Flash Corruption

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

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, if the supply voltage for executing instructions is too low, the CPU itself can execute instructions incorrectly.

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 is 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 keeps the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR register, and thus the Flash from unintentional writes.

22.6 Programming Time for Flash when Using SPM

Flash access is timed using the internal calibrated 8MHz oscillator. Typical Flash programming times for the CPU are shown in Table 22-2.

Table 22-2. SPM Programming Time

Operation	Min[1]	Max [2]
SPM: Flash 4-page erase, Flash page write, and lock bit write	3.7ms	4.5ms

Note: 1. Min. and max. programming times are per individual operation.

22.7 Register Description

22.7.1 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)	**–**	**–**	**RSIG**	**CTPB**	**RFLB**	**PGWRT**	**PGERS**	**SPMEN**	**SPMCSR**
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 and always read as “0”.

- **Bit 5 – RSIG: Read Device Signature Imprint Table**

Issuing an LPM instruction within three cycles after RSIG and SPMEN bits have been set returns the selected data (depending on Z pointer value) from the device signature imprint table into the destination register (see Section 23.3 “Device Signature Imprint Table” on page 199).

- **Bit 4 – CTPB: Clear Temporary Page Buffer**

If the CTPB bit is written while filling the temporary page buffer, the temporary page buffer is cleared and the data is 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 reads either the lock bits or the fuse bits (depending on Z0 in the Z pointer) into the destination register. For more information, see Section 22.4.5 “SPMCSR Cannot Be Written when EEPROM Is Programmed” on page 193.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to “1” at the same time as SPMEN, the next SPM instruction within four clock cycles executes page write while storing the data 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 auto clears upon completion of a page write or if no SPM instruction is executed within four clock cycles. The CPU is stopped during the entire page write operation.

- **Bit 1 – PGERS: Page Erase**

An SPM instruction within four clock cycles of PGERS and SPMEN have been set starts 4-page erase. The page address is taken from the high part of the Z pointer. Data in R1 and R0 are ignored. This bit auto clears upon completion of a 4-page erase or if no SPM instruction is executed within four clock cycles. The CPU is stopped during the entire 4-page erase operation.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If set to “1” together with RSIG, CTPB, RFLB, PGWRT, or PGERS, the following LPM/SPM instruction has a special meaning as described elsewhere.

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 auto clears upon completion of an SPM instruction or if no SPM instruction is executed within four clock cycles. During 4-page erase and page write, the SPMEN bit remains high until the operation is completed.

23. Lock Bits, Fuse Bits and Device Signature

23.1 Lock Bits

The Atmel® ATtiny1634 provides the program and data memory lock bits listed in Table 23-1.

Table 23-1. Lock Bit Byte

Lock Bit Byte	Bit No	Description	See	Default Value[1]
–	7	–		1 (unprogrammed)
–	6	–		1 (unprogrammed)
–	5	–		1 (unprogrammed)
–	4	–		1 (unprogrammed)
–	3	–		1 (unprogrammed)
–	2	–		1 (unprogrammed)
LB2	1	Lock bit	Below	1 (unprogrammed)
LB1	0			1 (unprogrammed)

Note: 1. “1” means unprogrammed, “0” means programmed.

Lock bits can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 23-2.

Table 23-2. Lock Bit Protection Modes

Lock Bits [1]		
LB2	LB1	Mode of Protection
1	1	No memory lock features enabled.
1	0	Further programming of Flash and EEPROM is disabled in parallel and serial programming mode. Fuse bits are locked in both serial and parallel programming mode[2].
0	1	Reserved
0	0	Further reading and programming of Flash and EEPROM is disabled in parallel and serial programming mode. Fuse bits are locked in both serial and parallel programming mode[2].

Notes: 1. “1” means unprogrammed, “0” means programmed.

2. Program fuse bits before programming LB1 and LB2.

When programming the lock bits, the mode of protection can be increased only. Writing the same, or lower, mode of protection automatically results in maximum protection.

Lock bits can be erased to “1” with the chip erase command only.

The Atmel ATtiny1634 has no separate boot loader section. The SPM instruction is enabled for the whole Flash if the SELFPRGEN fuse is programmed (“0”), otherwise it is disabled.

23.2 Fuse Bits

Fuse bits are described in Table 23-3, Table 23-4, and Table 23-5. Note that programmed fuses read as “0”.

Table 23-3. Extended Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	–	–		1 (unprogrammed)
6	–	–		1 (unprogrammed)
5	–	–		1 (unprogrammed)
4	BODPD1	Sets BOD operating mode when device is in sleep modes other than idle	Table 9-2 on page 42	0 (unprogrammed)
3	BODPD0			1 (unprogrammed)
2	BODACT1	Sets BOD operating mode when device is active or idle	Table 9-1 on page 42	1 (unprogrammed)
1	BODACT0			0 (unprogrammed)
0	SELFPRGEN	Enables SPM instruction	Section 22. on page 190	1 (unprogrammed)

Table 23-4. High Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	RSTDISBL	Disables external reset[1]	Section 9.2.2 on page 40	1 (unprogrammed)
6	DWEN	Enables debugWIRE[1]	Section 21. on page 188	1 (unprogrammed)
5	SPIEN	Enables serial programming and downloading of data to device[2]		0 (programmed)[3]
4	WDTON	Sets watchdog timer permanently on	Section 9.5.2 on page 45	1 (unprogrammed)
3	EESAVE	Preserves EEPROM memory during chip erase operation	Section 24.2.3 on page 204	1 (unprogrammed)[4]
2	BODLEVEL2	Sets BOD trigger level	Table 25-7 on page 219	1 (unprogrammed)
1	BODLEVEL1			0 (programmed)
0	BODLEVEL0			1 (unprogrammed)

Notes:
1. Programming this fuse bit changes the functionality of the $\overline{\text{RESET}}$ pin and renders further programming via the serial interface impossible. The fuse bit can be unprogrammed using the parallel programming algorithm (see Section 24.2 “Parallel Programming” on page 202).
2. This fuse bit is not accessible in serial programming mode.
3. This setting enables SPI programming.
4. This setting does not preserve EEPROM.

Table 23-5. Low Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	CKDIV8	Divides clock by 8[1]	Section 7.3 on page 28	0 (programmed)
6	CKOUT	Outputs system clock on port pin	Section 7.4 on page 28	1 (unprogrammed)
5	–	–		1 (unprogrammed)
4	SUT	Sets system start-up time	Table 7-2 on page 29	0 (programmed)[2]
3	CKSEL3	Selects clock source	Table 7-3 on page 30	0 (programmed)[3]
2	CKSEL2			0 (programmed)[3]
1	CKSEL1			1 (unprogrammed)[3]
0	CKSEL0			0 (programmed)[3]

Note:
1. Unprogramming this fuse at low voltages may result in overclocking. See Section 25.3 "Speed" on page 217 for device speed versus supply voltage.
2. This setting results in maximum start-up time for the default clock source.
3. This setting selects the clock source described in Section 7.2.2 "Calibrated Internal 8MHz Oscillator" on page 27.

Fuse bits are locked when lock bit 1 (LB1) is programmed. Therefore, fuse bits must be programmed before lock bits.

Fuse bits are not affected by a chip erase.

23.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values have no effect until the part leaves the programming mode. This does not apply to the EESAVE fuse, which takes effect once it is programmed. The fuses are also latched on power-up in normal mode.

23.3 Device Signature Imprint Table

The device signature imprint table is a dedicated memory area used for storing miscellaneous device information such as the device signature and oscillator calibration data. Most of this memory segment is reserved for internal use as outlined in Table 23-6. Byte addresses are used when the device itself reads the data with the LPM command. External programming devices must use word addresses.

Table 23-6. Contents of Device Signature Imprint Table

Word Address (External)	Byte Address (Internal)	Description
0x00	0x00	Signature byte 0[(1)]
	0x01	Calibration data for internal 8MHz oscillator (OSCCAL0)[(2)]
0x01	0x02	Signature byte 1[(1)]
	0x03	Oscillator temperature calibration data (OSCTCAL0A)
0x02	0x04	Signature byte 2[(1)]
	0x05	Oscillator temperature calibration data (OSCTCAL0B)
0x03	0x06	Reserved
	0x07	Calibration data for internal 32kHz oscillator (OSCCAL1)[(2)]
0x04...0x3F	...	Reserved
	...	Reserved

Notes: 1. For more information, see Section 23.3.1 “Signature Bytes” on page 199.

2. For more information, see Section 23.3.2 “Calibration Bytes” on page 199.

23.3.1 Signature Bytes

All Atmel® microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked.

Signature bytes can also be read by the device firmware. For more information, see Section 23.4 “Reading Lock, Fuse, and Signature Data from Software” on page 200.

The three signature bytes reside in a separate address space called the device signature imprint table. The signature data for the Atmel ATtiny1634 is given in Table 23-7.

Table 23-7. Device Signature Bytes

Part	Signature Byte 0	Signature Byte 1	Signature Byte 0
Atmel ATtiny1634	0x1E	0x94	0x12

23.3.2 Calibration Bytes

The device signature imprint table of the Atmel ATtiny1634 contains calibration data for the internal oscillators as shown in Table 23-6 on page 199. During reset, calibration data is automatically copied to the calibration registers (OSCCAL0, OSCCAL1) to ensure correct frequency of the calibrated oscillators (see Section 7.5.3 “OSCCAL0 – Oscillator Calibration Register” on page 32, and Section 7.5.6 “OSCCAL1 – Oscillator Calibration Register” on page 33).

Calibration bytes can also be read by the device firmware. For more information, see Section 23.4 “Reading Lock, Fuse, and Signature Data from Software” on page 200.

23.4 Reading Lock, Fuse, and Signature Data from Software

Fuse and lock bits can be read by the device firmware. Programmed fuse and lock bits read as “0”, unprogrammed fuse and lock bits read as “1” (see Section 23.1 “Lock Bits” on page 196 and Section 23.2 “Fuse Bits” on page 197).

In addition, firmware can also read data from the device signature imprint table (see Section 23.3 “Device Signature Imprint Table” on page 199).

23.4.1 Lock Bit Read

Lock bit values are returned in the destination register after an LPM instruction has been issued within three CPU cycles after RFLB and SPMEN bits have been set in SPMCSR (see Section 22.7.1 “SPMCSR – Store Program Memory Control and Status Register” on page 194). The RFLB and SPMEN bits automatically clear upon completion of reading the lock bits, or if no LPM instruction is executed within three CPU cycles, or if no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared, LPM functions normally.

Follow the procedure below to read the lock bits:

1. Load the Z pointer with 0x0001.
2. Set RFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the lock bits from the LPM destination register.

If successful, the contents of the destination register are as follows:

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	**LB2**	**LB1**

For more information, see Section 23.1 “Lock Bits” on page 196.

23.4.2 Fuse Bit Read

The algorithm for reading fuse bytes is similar to the one described above for reading lock bits, only the addresses are different.

Follow the procedure below to read the fuse low byte (FLB):

1. Load the Z pointer with 0x0000.
2. Set RFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the FLB from the LPM destination register.

If successful, the contents of the destination register are as follows:

Bit	7	6	5	4	3	2	1	0
Rd	**FLB7**	**FLB6**	**FLB5**	**FLB4**	**FLB3**	**FLB2**	**FLB1**	**FLB0**

For a detailed description and mapping of the fuse low byte, see Table 23-5 on page 198.

To read the fuse high byte (FHB), replace the address in the Z pointer with 0x0003 and repeat the procedure above. If successful, the contents of the destination register are as follows:

Bit	7	6	5	4	3	2	1	0
Rd	**FHB7**	**FHB6**	**FHB5**	**FHB4**	**FHB3**	**FHB2**	**FHB1**	**FHB0**

For a detailed description and mapping of the fuse high byte, see Table 23-4 on page 197.

To read the fuse extended byte (FEB), replace the address in the Z pointer with 0x0002 and repeat the previous procedure. If successful, the contents of the destination register are as follows:

Bit	7	6	5	4	3	2	1	0
Rd	**FEB7**	**FEB6**	**FEB5**	**FEB4**	**FEB3**	**FEB2**	**FEB1**	**FEB0**

For a detailed description and mapping of the fuse extended byte, see Table 23-3 on page 197.

23.4.3 Device Signature Imprint Table Read

Follow the procedure below to read the contents of the device signature imprint table:

1. Load the Z pointer with the table index.
2. Set RSIG and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read table data from the LPM destination register.

If successful, the contents of the destination register are as described in Section 23.3 “Device Signature Imprint Table” on page 199. See program example below.

Assembly Code Example

```
DSIT_read:
        ; Uses Z-pointer as table index
        ldi     ZH, 0
        ldi     ZL, 1
        ; Preload SPMCSR bits into R16, then write to SPMCSR
        ldi     r16, (1<<RSIG)|(1<<SPMEN)
        out     SPMCSR, r16
        ; Issue LPM. Table data will be returned into r17
        lpm     r17, Z
        ret
```

Note: See Section 4.2 “Code Examples” on page 7.

24. External Programming

This section describes how to program and verify Flash memory, EEPROM, lock bits, and fuse bits in the Atmel® ATtiny1634.

24.1 Memory Parametrics

Flash memory parametrics are summarized in Table 24-1.

Table 24-1. Flash Parametrics

Device	Flash Size	Page Size	PCWORD(1)	Pages	PCPAGE(1)	PCMSB(1)
ATtiny1634	8K words (16KB)	16 words	PC[3:0]	512	PC[12:4]	12

Note: 1. See Table 22-1 on page 192.

EEPROM parametrics are summarized in Table 24-2 below.

Table 24-2. EEPROM Parametrics

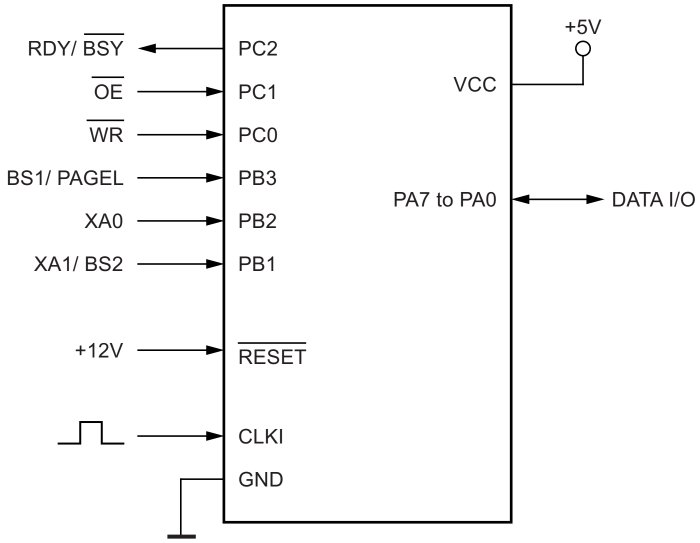
Device	EEPROM Size	Page Size	PCWORD(1)	Pages	PCPAGE(1)	EEAMSB
ATtiny1634	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7

Note: 1. See Table 22-1 on page 192.

24.2 Parallel Programming

Parallel programming signals and connections are illustrated in Figure 24-1.

Figure 24-1. Parallel Programming Signals



Signals are described in Table 24-3. Pins not listed in the table are referenced by pin names.

Table 24-3. Pin and Signal Names Used in Programming Mode

Signal Name	Pin(s)	I/O	Function
RDY/$\overline{\text{BSY}}$	PC2	O	0: the device is busy programming 1: the device is ready for new command
$\overline{\text{OE}}$	PC1	I	Output enable (active low)
$\overline{\text{WR}}$	PC0	I	Write pulse (active low)
BS1/PAGEL	PB3	I	Byte select 1 (0: low byte, 1: high byte) / Program memory and EEPROM data page load
XA0	PB2	I	XTAL action bit 0
XA1/BS2	PB1	I	XTAL action bit 1/ Byte Select 2 (0: low byte, 1: 2nd high byte)
DATA I/O	PA[7:0]	I/O	Bidirectional data bus. Output when $\overline{\text{OE}}$ is low

Pulses are assumed to be at least 250ns, unless otherwise noted.

Table 24-4. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
$\overline{\text{WR}}$	Prog_enable[3]	0
BS1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
XA1	Prog_enable[0]	0

The XA1 and XA0 pins determine the action when CLKI is given a positive pulse as shown in Table 24-5.

Table 24-5. XA1 and XA0 Coding

XA1	XA0	Action When CLKI 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

When pulsing $\overline{\text{WR}}$ or $\overline{\text{OE}}$, the command loaded determines the action executed. The different command options are shown in Table 24-6.

Table 24-6. Command Byte Bit Coding

Command Byte	Command
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

24.2.1 Enter Programming Mode

The following algorithm puts the device in parallel (high-voltage) programming mode:

1. Set Prog_enable pins (see Table 24-4 on page 203) to “0000”, $\overline{\text{RESET}}$ pin to 0V and V_{CC} to 0V.
2. Apply 4.5V to 5.5V between V_{CC} and GND. Ensure that V_{CC} reaches at least 1.8V within the next 20µs.
3. Wait 20µs to 60µs and apply 11.5V to 12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10µs after the high voltage has been applied to ensure the Prog_enable signature has been latched.
5. Wait at least 300µs before giving any parallel programming commands.
6. Exit programming mode by powering the device down or by bringing $\overline{\text{RESET}}$ pin to 0V.

If the rise time of the V_{CC} is unable to fulfill the requirements listed above, the following alternative algorithm can be used:

1. Set Prog_enable pins (Table 24-4 on page 203) to “0000”, $\overline{\text{RESET}}$ pin to 0V and V_{CC} to 0V.
2. Apply 4.5V to 5.5V between V_{CC} and GND.
3. Monitor V_{CC}, and as soon as V_{CC} reaches 0.9V to 1.1V apply 11.5V to 12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10µs after the high voltage has been applied to ensure the Prog_enable signature has been latched.
5. Wait until V_{CC} actually reaches 4.5V to 5.5V before giving any parallel programming commands.
6. Exit programming mode by powering the device down or by bringing $\overline{\text{RESET}}$ pin to 0V.

24.2.2 Considerations for Efficient Programming

Loaded commands and addresses are retained in the device during programming. For efficient programming, the following should be considered:

- When writing or reading multiple memory locations, the command only needs to be loaded once.
- Do not write the data value 0xFF, because this is already the contents of the entire Flash and EEPROM (unless the EESAVE fuse is programmed) after a chip erase.
- Address high byte only needs to be loaded before programming or reading a new 256-word window in Flash or 256-byte EEPROM. This also applies to reading signature bytes.

24.2.3 Chip Erase

A chip erase must be performed before the Flash and/or EEPROM are reprogrammed. The chip erase command will erase all Flash and EEPROM plus lock bits. If the EESAVE fuse is programmed, the EEPROM is not erased.

Lock bits are not reset until the program memory has been completely erased. Fuse bits are not changed.

The chip erase command is loaded as follows:

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the chip erase command.
4. Give CLKI 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.

24.2.4 Programming the Flash

Flash is organized in pages as shown in Table 24-1 on page 202. When programming Flash, the program data is first 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 the “Write Flash” command

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.
4. Give CLKI a positive pulse. This loads the command.

B. Load the address low byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects the low address.
3. Set DATA = address low byte (0x00 – 0xFF).
4. Give CLKI a positive pulse. This loads the address low byte.

C. Load the data low byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = data low byte (0x00 – 0xFF).
3. Give CLKI a positive pulse. This loads the data byte.

D. Load the 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 CLKI a positive pulse. This loads the data byte.

E. Latch data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes (see Figure 24-3 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 24-2 on page 206. Note that if less than eight bits are required to address words in the page (page size < 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 the high address.
3. Set DATA = address high byte (0x00 – 0xFF).
4. Give CLKI a positive pulse. This loads the address high byte.

H. Program page

1. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{\text{BSY}}$ goes low.
2. Wait until RDY/$\overline{\text{BSY}}$ goes high (see Figure 24-3 for signal waveforms).

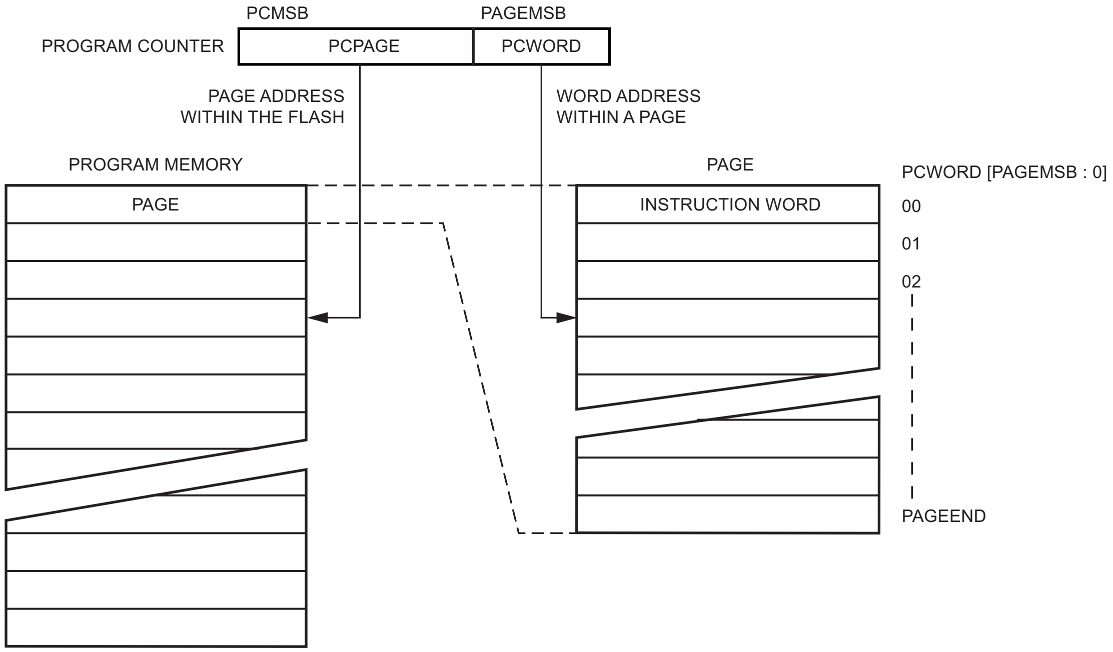
I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. Ending 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 CLKI a positive pulse. This loads the command and the internal write signals are reset.

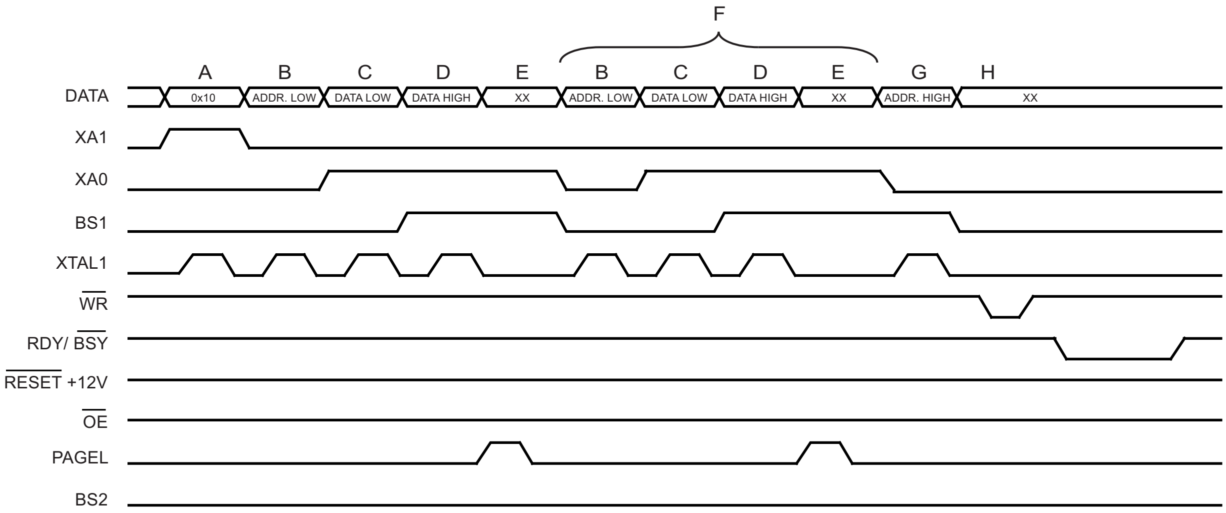
Flash page addressing is illustrated in Figure 24-2. Symbols used are described in Table 22-1 on page 192.

Figure 24-2. Addressing the Flash which Is Organized in Pages



Flash programming waveforms are illustrated in Figure 24-3, where XX means “don’t care” and letters refer to the programming steps described above.

Figure 24-3. Flash Programming Waveforms



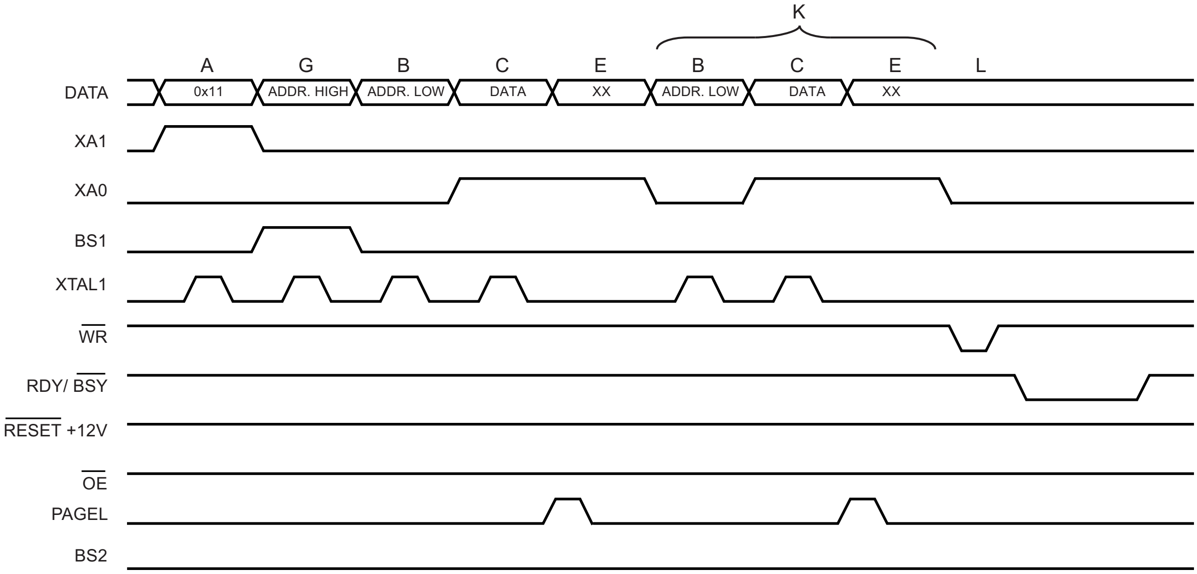
24.2.5 Programming the EEPROM

The EEPROM is organized in pages (see Table 24-2 on page 202). 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 (for more information on loading the command, address and data, see Section 24.2.4 "Programming the Flash" on page 205):

- A: Load command "0001 0001"
- G: Load address high byte (0x00 – 0xFF)
- B: Load address low byte (0x00 – 0xFF)
- C: Load data (0x00 – 0xFF)
- E: Latch data (give PAGEL a positive pulse)
- K: Repeat steps B, C, and E until the entire buffer is filled.
- L: Programming the EEPROM page:
  - Set BS1 to "0".
  - Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{\text{BSY}}$ goes low.
  - Wait until RDY/$\overline{\text{BSY}}$ goes high before programming the next page (see Figure 24-4 for signal waveforms)

EEPROM programming waveforms are illustrated in Figure 24-4, where XX means "don't care" and letters refer to the programming steps described above.

Figure 24-4. EEPROM Programming Waveforms



24.2.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (for more information on loading the command and address, see Section 24.2.4 "Programming the Flash" on page 205):

- A: Load command "0000 0010"
- G: Load address high byte (0x00 – 0xFF)
- B: Load address low byte (0x00 – 0xFF)
- Set $\overline{\text{OE}}$ to "0", and BS1 to "0". The Flash word low byte can now be read at DATA.
- Set BS1 to "1". The Flash word high byte can now be read at DATA.
- Set $\overline{\text{OE}}$ to "1".

24.2.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (for more information on loading the command and address, see Section 24.2.4 “Programming the Flash” on page 205):

- A: Load command “0000 0011”
- G: Load address high byte (0x00 – 0xFF)
- B: Load address low byte (0x00 – 0xFF)
- Set $\overline{\text{OE}}$ to “0”, and BS1 to “0”. The EEPROM data byte can now be read at DATA.
- Set $\overline{\text{OE}}$ to “1”.

24.2.8 Programming Low Fuse Bits

The algorithm for programming the low fuse bits is as follows (for more information on loading the command and data, see Section 24.2.4 “Programming the Flash” on page 205):

- A: Load command “0100 0000”.
- C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit
- Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

24.2.9 Programming High Fuse Bits

The algorithm for programming the high fuse bits is as follows (for more information on loading the command and data, see Section 24.2.4 “Programming the Flash” on page 205):

- A: Load command “0100 0000”.
- C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit
- Set BS1 to “1” and BS2 to “0”. This selects high data byte.
- Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.
- Set BS1 to “0”. This selects low data byte.

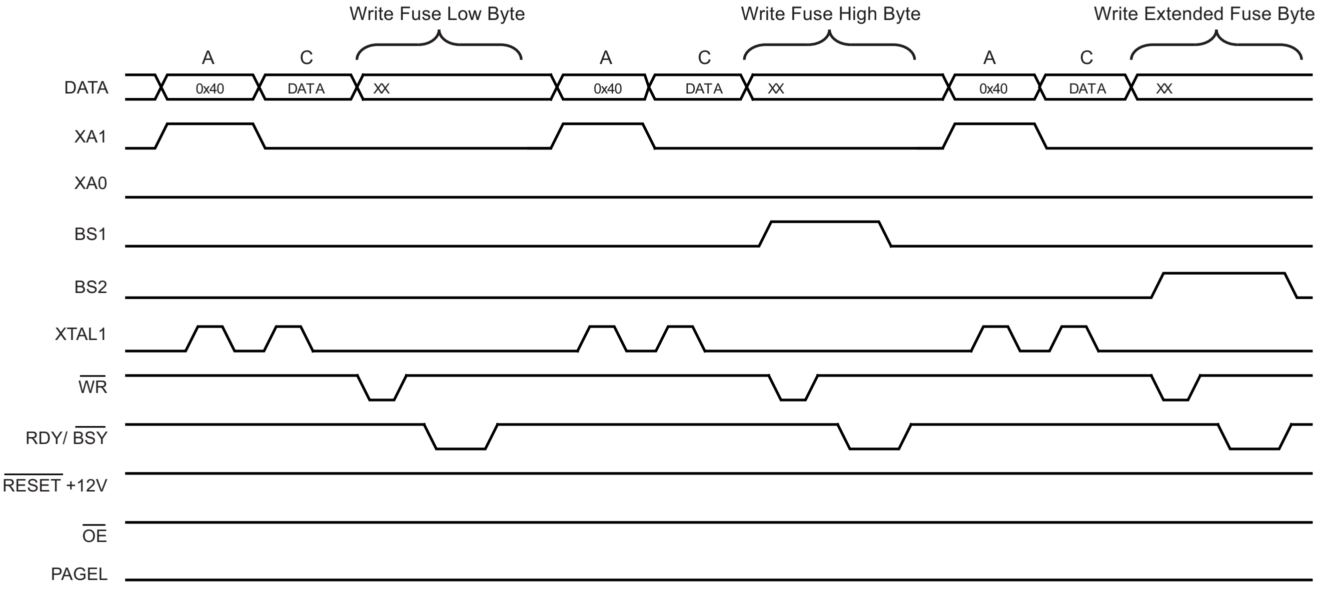
24.2.10 Programming Extended Fuse Bits

The algorithm for programming the extended fuse bits is as follows (for more information on loading the command and data, see Section 24.2.4 “Programming the Flash” on page 205):

- A: Load command “0100 0000”.
- C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit
- Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
- Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.
- Set BS2 to “0”. This selects low data byte.

Fuse programming waveforms are illustrated in Figure 24-5, where XX means "don't care" and letters refer to the programming steps described above.

Figure 24-5. Fuses Programming Waveforms



24.2.11 Programming the Lock Bits

The algorithm for programming the lock bits is as follows (for more information on loading the command and data, see Section 24.2.4 "Programming the Flash" on page 205):

- A: Load command "0010 0000".
- C: Load data low byte. Bit n = "0" programs the lock bit. If LB1 and LB2 have been programmed, it is not possible to program the lock bits by any external programming mode.
- Give WR a negative pulse and wait for RDY/BSY to go high.

Lock bits can only be cleared by executing chip erase.

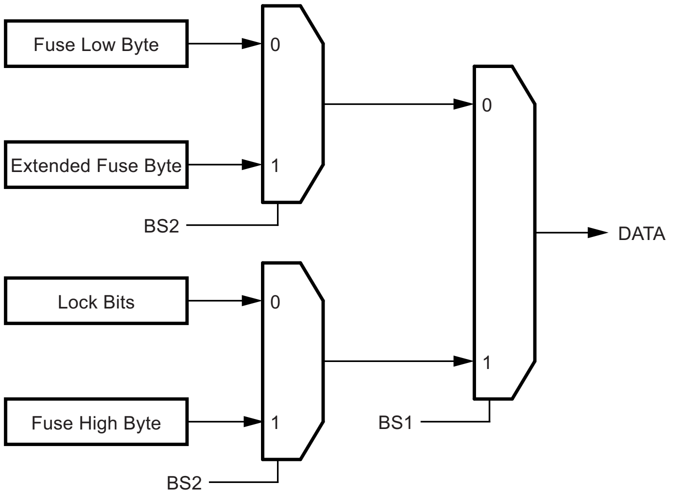
24.2.12 Reading Fuse and Lock Bits

The algorithm for reading fuse and lock bits is as follows (for more information on loading the command and data, see Section 24.2.4 "Programming the Flash" on page 205):

- A: Load command "0000 0100".
- Set OE to "0", BS2 to "0" and BS1 to "0". Low fuse bits can now be read at DATA ("0" means programmed).
- Set OE to "0", BS2 to "1" and BS1 to "1". High fuse bits can now be read at DATA ("0" means programmed).
- Set OE to "0", BS2 to "1", and BS1 to "0". Extended fuse bits can now be read at DATA ("0" means programmed).
- Set OE to "0", BS2 to "0" and BS1 to "1". Lock bits can now be read at DATA ("0" means programmed).
- Set OE to "1".

Fuse and lock bit mapping is illustrated in Figure 24-6.

Figure 24-6. Mapping Between BS1, BS2, and the Fuse and Lock Bits During Read



24.2.13 Reading Signature Bytes

The algorithm for reading the signature bytes is as follows (for more information on loading the command and address, see Section 24.2.4 “Programming the Flash” on page 205):

1. A: Load command “0000 1000”.
2. B: Load address low byte (0x00 – 0x02).
3. Set $\overline{\text{OE}}$ to “0”, and BS1 to “0”. The selected signature byte can now be read at DATA.
4. Set $\overline{\text{OE}}$ to “1”.

24.2.14 Reading the Calibration Byte

The algorithm for reading the calibration byte is as follows (for more information on loading the command and address, see Section 24.2.4 “Programming the Flash” on page 205):

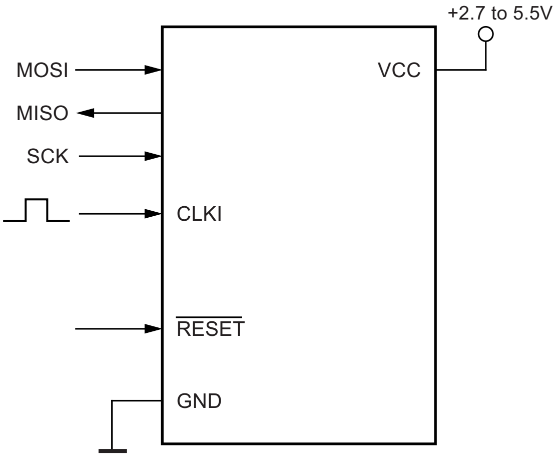
1. A: Load command “0000 1000”.
2. B: Load address low byte, 0x00.
3. Set $\overline{\text{OE}}$ to “0”, and BS1 to “1”. The calibration byte can now be read at DATA.
4. Set $\overline{\text{OE}}$ to “1”.

24.3 Serial Programming

Flash and EEPROM memory arrays can both be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of the SCK, MOSI (input), and MISO (output) pins. After $\overline{\text{RESET}}$ is set low, the programming enable instruction needs to be executed before program/erase operations can be executed.

Serial programming signals and connections are illustrated in Figure 24-7 below. The pin mapping is listed in Table 24-7 on page 211.

Figure 24-7. Serial Programming Signals



Note: If the device is clocked by the internal oscillator, there is no need to connect a clock source to the CLKI pin.

When programming the EEPROM, an auto-erase cycle is built into the self timed programming operation and there is no need to first execute the chip erase instruction. This applies to serial programming mode only.

The chip erase operation turns the content of every memory location in Flash 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:

- Minimum low period of serial clock:
  - 2 CPU clock cycles
- Minimum high period of serial clock:
  - 2 CPU clock cycles

24.3.1 Pin Mapping

The pin mapping is listed in Table 24-7. Note that not all parts use the SPI pins dedicated for the internal SPI interface.

Table 24-7. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB1	I	Serial data in
MISO	PB2	O	Serial data out
SCK	PC1	I	Serial clock

24.3.2 Programming Algorithm

When writing serial data to the Atmel® ATtiny1634, data is clocked on the rising edge of SCK. When reading data from the Atmel ATtiny1634, data is clocked on the falling edge of SCK. For more information on timing, see Figure 25-7 on page 223 and Figure 25-8 on page 223.

To program and verify the Atmel ATtiny1634 in the serial programming mode, the following sequence is recommended (see Table 24-8 on page 213):

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 cannot guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse after SCK has been set to "0". The duration of the pulse must be at least t_{RST} plus two CPU clock cycles. See Table 25-5 on page 218 for the definition of minimum pulse width on $\overline{\text{RESET}}$ pin, t_{RST}.
2. Wait for at least 20ms and then enable serial programming by sending the programming enable serial instruction to the MOSI pin.
3. The serial programming instructions do not work if the communication is out of synchronization. When in sync, the second byte (0x53) echoes back when issuing the third byte of the programming enable instruction.
    - Regardless of whether the echo is correct or not, all four bytes of the instruction must be transmitted
    - If the 0x53 does 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 6 LSB of the address and data together with the load program memory page instruction.
    - To ensure correct loading of the page, data low byte must be loaded before data high byte for a given address is applied.
    - The program memory page is stored by loading the write program memory page instruction with the 7msB 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 24-9). Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. The EEPROM can be programmed one byte or one page at a time.
    - **A**: Byte programming. The EEPROM array is programmed one byte at a time by supplying the address and data together with the write instruction. EEPROM memory locations are 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 24-9). In a chip-erased device, no 0xFFs in the data file(s) need to be programmed.
    - **B**: Page programming (the EEPROM array is programmed one page at a time). The memory page is loaded one byte at a time by supplying the 6 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 7msB of the address. When using EEPROM, page access only byte locations loaded with the Load EEPROM memory page instruction are altered and the remaining locations remain unchanged. 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 24-9). In a chip-erased device, no 0xFF in the data file(s) needs to be programmed.
6. Any memory location can be verified by using the read instruction which returns the content at the selected address at the serial output pin (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 required): set $\overline{\text{RESET}}$ to "1", and turn V_{CC} power off.

24.3.3 Programming Instruction Set

The instruction set for serial programming is described in Table 24-8 and Figure 24-8 on page 214.

Table 24-8. 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	$00	adr LSB	high data byte in
Load Program Memory Page, Low byte	$40	$00	adr LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	$00	0000 000aa[2]	data byte in
Read Instructions				
Read Program Memory, High byte	$28	adrmsB	adr LSB	high data byte out
Read Program Memory, Low byte	$20	adrmsB	adr LSB	low data byte out
Read EEPROM Memory	$A0	0000 00aa[2]	aaaa aaaa[2]	data byte out
Read Lock bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	0000 000aa[2]	data byte out
Read Fuse bits	$50	$00	$00	data byte out
Read Fuse High bits	$58	$08	$00	data byte out
Read Fuse Extended Bits	$50	$08	$00	data byte out
Read Calibration Byte	$38	$00	$00	data byte out
Write Instructions[3]				
Write Program Memory Page	$4C	adrmsB[4]	adr LSB[4]	$00
Write EEPROM Memory	$C0	0000 00aa[2]	aaaa aaaa[2]	data byte in
Write EEPROM Memory Page (page access)	$C2	0000 00aa[2]	aaaa aa00[2]	$00
Write Lock bits[5]	$AC	$E0	$00	data byte in
Write Fuse bits[5]	$AC	$A0	$00	data byte in
Write Fuse High bits[5]	$AC	$A8	$00	data byte in
Write Fuse Extended Bits[5]	$AC	$A4	$00	data byte in

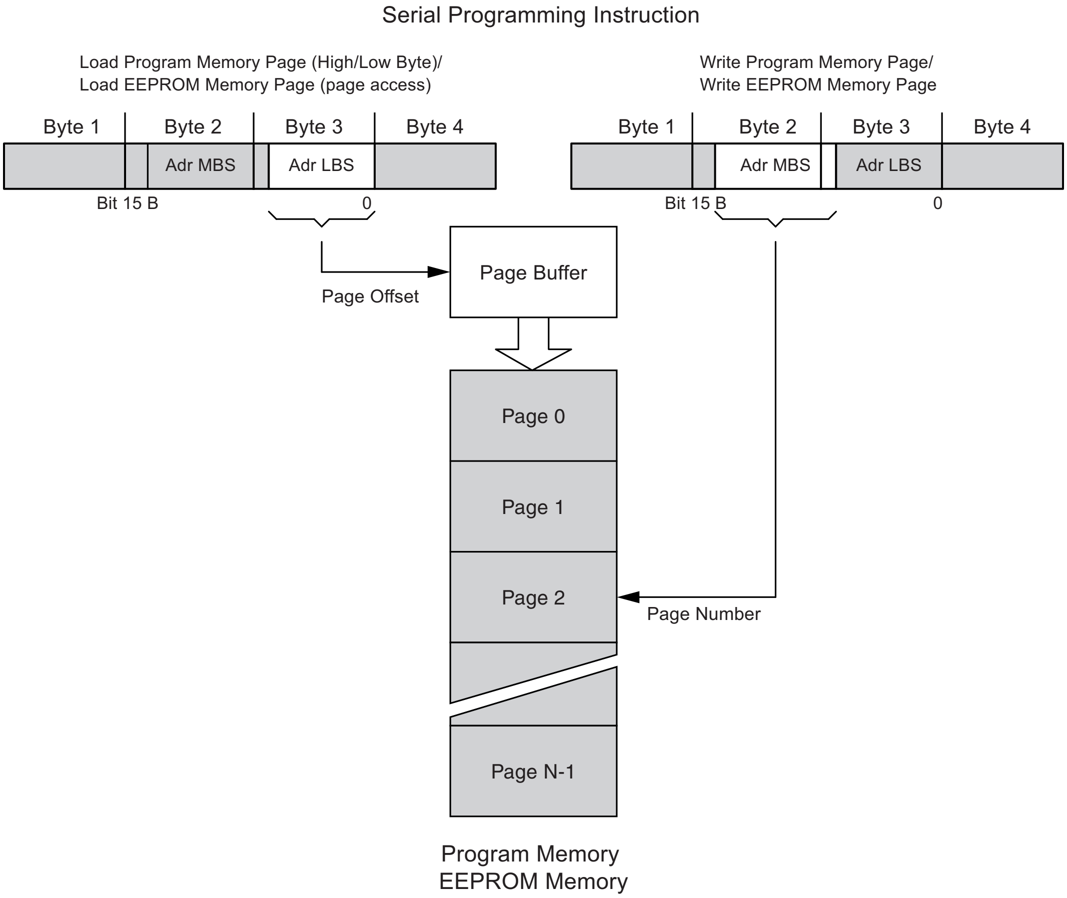
Notes:
1. Not all instructions are applicable for all parts.
2. a = address.
3. Instructions accessing program memory use a word address. This address may be random within the page range.
4. Word addressing.
5. To ensure future compatibility, unused fuses and lock bits should be unprogrammed (“1”).

If the LSB of RDY/$\overline{\text{BSY}}$ 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 24-8 on page 214).

Figure 24-8. Serial Programming Instruction Example



24.4 Programming Time for Flash and EEPROM

Flash and EEPROM wait times are listed in Table 24-9.

Table 24-9. Typical Wait Delays before Next Flash or EEPROM Location Can Be Written

Operation	Minimum Wait Delay
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	3.6ms
t_{WD_ERASE}	9.0ms

25. Electrical Characteristics

25.1 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.	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 DC current V_{CC} and GND pins		40 200	mA

25.2 DC Characteristics

Table 25-1. DC Characteristics. T_A = –40°C to +125°C

Parameter	Condition	Symbol	Min	Typ[1]	Max	Unit
Input low voltage	V_{CC} = 2.7V to 5.5V	V_{IL}			$0.3V_{CC}$[2]	V
Input low-voltage, $\overline{\text{RESET}}$ pin as reset[4]	V_{CC} = 2.7V to 5.5V				$0.2V_{CC}$[2]	V
Input high-voltage except $\overline{\text{RESET}}$ pin	V_{CC} = 2.7V to 5.5V	V_{IH}	$0.6V_{CC}$[3]		V_{CC} +0.5	V
Input high-voltage $\overline{\text{RESET}}$ pin as reset[4]	V_{CC} = 2.7V to 5.5V		$0.9V_{CC}$[3]		V_{CC} +0.5	V
Output low-voltage[5] Except $\overline{\text{RESET}}$ pin[7]	Standard I/O: I_{OL} = 8mA, V_{CC} = 5V	V_{OL}			0.8	V
	High-sink I/O: I_{OL} = 20mA, V_{CC} = 5V					
	Standard I/O: I_{OL} = 5mA, V_{CC} = 3V				0.7	V
	High-sink I/O: I_{OL} = 10mA, V_{CC} = 3V					

Notes:
1. Typical values at +25°C.
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. Not tested in production.
5. 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 sum of all I_{OL} (for all ports) should not exceed 100mA. If I_{OL} exceeds the test conditions, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
6. 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 sum of all I_{OH} (for all ports) should not exceed 100mA. If I_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
7. The $\overline{\text{RESET}}$ pin must tolerate high voltages when entering and operating in programming modes and, as a consequence, has a weak drive strength as compared to regular I/O pins (see Section 26.7 “Output Driver Strength” on page 234).
8. These are test limits, which account for leakage currents of the test environment. Actual device leakage currents are lower.
9. Values are with external clock using methods described in Section 8.3 “Minimizing Power Consumption” on page 35. Power reduction is enabled (PRR = 0xFF) and there is no I/O drive.
10. Bod disabled.

Table 25-1. DC Characteristics. T_A = –40°C to +125°C (Continued)

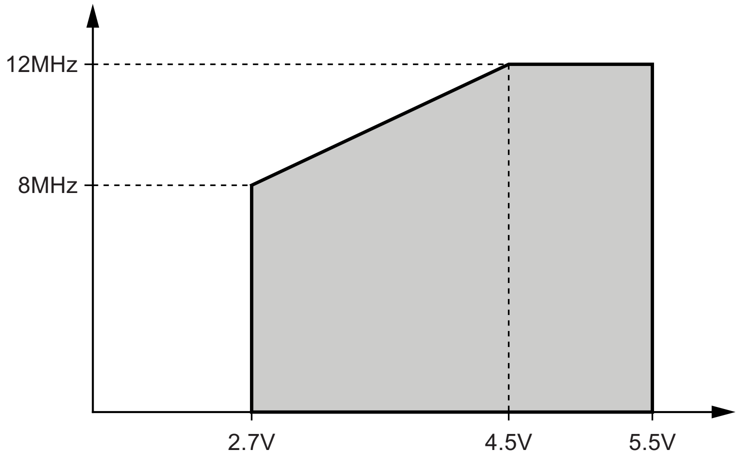
Parameter	Condition	Symbol	Min	Typ[1]	Max	Unit
Output high-voltage[6] except $\overline{\text{RESET}}$ pin[7]	I_{OH} = –10mA, V_{CC} = 5V	V_{OH}	4.3			V
	I_{OH} = –5mA, V_{CC} = 3V		2.5			V
Input leakage current I/O pin	Vcc = 5.5V, pin low (absolute value)	I_{LIL}		< 0.05	1[8]	µA
Input leakage current I/O pin	Vcc = 5.5V, pin high (absolute value)	I_{LIH}		< 0.05	1[8]	µA
Pull-up resistor, I/O pin	V_{CC} = 5.5V, input low	R_{PU}	20		50	kΩ
Pull-up resistor, reset pin	V_{CC} = 5.5V, input low		30		60	kΩ
Supply current, active mode[9]	f = 4MHz, V_{CC} = 3V	I_{CC}		1.2	1.8	mA
	f = 8MHz, V_{CC} = 5V			4.2	8	mA
	f = 12MHz, V_{CC} = 5V			5.9	10	mA
Supply current, idle mode[9]	f = 4MHz, V_{CC} = 3V			0.23	0.75	mA
	f = 8MHz, V_{CC} = 5V			1.1	1.5	
	f = 12MHz, V_{CC} = 5V			1.7	2.2	
Supply current, power-down mode[10]	WDT enabled, V_{CC} = 3V			1.93	30	µA
	WDT disabled, V_{CC} = 3V			0.11	27	µA
	WDT enabled, V_{CC} = 5V			3.85	45	µA
	WDT disabled, V_{CC} = 5V			0.15	40	µA

Notes:
1. Typical values at +25°C.
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. Not tested in production.
5. 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 sum of all I_{OL} (for all ports) should not exceed 100mA. If I_{OL} exceeds the test conditions, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
6. 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 sum of all I_{OH} (for all ports) should not exceed 100mA. If I_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
7. The $\overline{\text{RESET}}$ pin must tolerate high voltages when entering and operating in programming modes and, as a consequence, has a weak drive strength as compared to regular I/O pins (see Section 26.7 “Output Driver Strength” on page 234).
8. These are test limits, which account for leakage currents of the test environment. Actual device leakage currents are lower.
9. Values are with external clock using methods described in Section 8.3 “Minimizing Power Consumption” on page 35. Power reduction is enabled (PRR = 0xFF) and there is no I/O drive.
10. Bod disabled.

25.3 Speed

The maximum operating frequency of the device depends on the supply voltage (V_{CC}). The relationship between supply voltage and maximum operating frequency is piece-wise linear as shown in Figure 25-1.

Figure 25-1. Maximum Frequency versus V_{CC}



25.4 Clock

25.4.1 Accuracy of Calibrated 8MHz Oscillator

It is possible to manually calibrate the internal 8MHz oscillator to be more accurate than default factory calibration. Note that the oscillator frequency depends on temperature and voltage. For voltage and temperature characteristics, see Section 26-39 "Calibrated Oscillator Frequency (Nominal = 1MHz) versus V_{CC}" on page 243 and Section 26-40 "Calibrated Oscillator Frequency (Nominal = 1MHz) versus Temperature" on page 244.

Table 25-2. Calibration Accuracy of Internal 8MHz Oscillator

Calibration Method	Target Frequency	V_{CC}	Temperature	Accuracy
Factory calibration	8.0MHz	3.0V	25°C	±1.5%(1)
		2.7V to 5.5V	–40°C to +125°C	±10%

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

25.4.2 Accuracy of Calibrated 32kHz Oscillator

It is possible to manually calibrate the internal 32kHz oscillator to be more accurate than default factory calibration. Note that the oscillator frequency depends on temperature and voltage. For voltage and temperature characteristics, see Section 26-41 "ULP Oscillator Frequency (Nominal = 32kHz) versus V_{CC}" on page 244 and Section 26-41 "ULP Oscillator Frequency (Nominal = 32kHz) versus V_{CC}" on page 244.

Table 25-3. Calibration Accuracy of Internal 32kHz Oscillator

Calibration Method	Target Frequency	V_{CC}	Temperature	Accuracy
Factory calibration	32kHz	2.7V to 5.5V	–40°C to +125°C	±33%

25.4.3 External Clock Drive

Figure 25-2. External Clock Drive Waveform

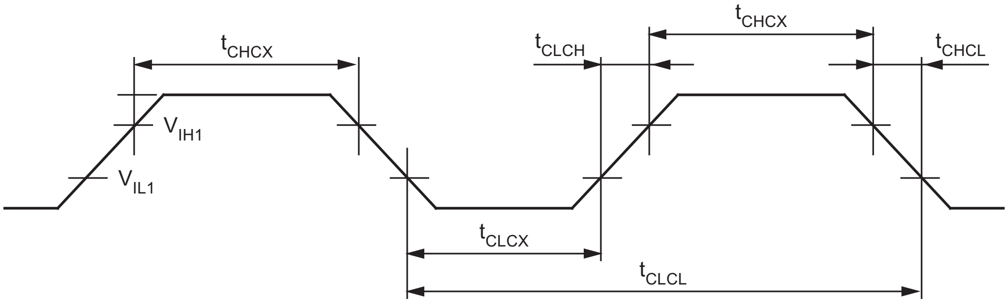


Table 25-4. External Clock Drive Characteristics

Parameter	Symbol	V_{CC} = 2.7V to 5.5V		V_{CC} = 4.5V to 5.5V		Unit
		Min.	Max.	Min.	Max.	
Clock frequency	$1/t_{CLCL}$	0	8	0	12	MHz
Clock period	t_{CLCL}	125		83		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 next	Δt_{CLCL}		2		2	%

25.5 System and Reset

Table 25-5. Reset, Brown-out, and Internal Voltage Characteristics

Parameter	Condition	Symbol	Min	Typ	Max	Unit
$\overline{\text{RESET}}$ pin threshold voltage		V_{RST}	$0.2V_{CC}$		$0.9V_{CC}$	V
Minimum pulse width on $\overline{\text{RESET}}$ pin	V_{CC} = 3V V_{CC} = 5V	t_{RST}	2000(1)	700 400		ns
Brown-out detector hysteresis		V_{HYST}		50		mV
Minimum pulse width on brown-out reset		t_{BOD}		2		μs
Internal band-gap reference voltage	V_{CC} = 2.7V T_A = 25°C	V_{BG}	1.0	1.1	1.2	V
Internal band-gap reference start-up time	V_{CC} = 2.7V T_A = 25°C	t_{BG}		40	70	μs
Internal band-gap reference current consumption	V_{CC} = 2.7V T_A = 25°C	I_{BG}		15		μA

Note: 1. Minimum pulse width to guarantee a reset under all usage conditions.

25.5.1 Power-on Reset

Table 25-6. 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.

25.5.2 Brown-out Detection

Table 25-7. V_{BOT} versus BODLEVEL Fuse Coding

BODLEVEL[2:0] Fuses	Min[1]	Typ[1]	Max[1]	Unit
11X	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.0	4.3	4.6	
0XX	Reserved			

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.

25.6 Two-wire Serial Interface

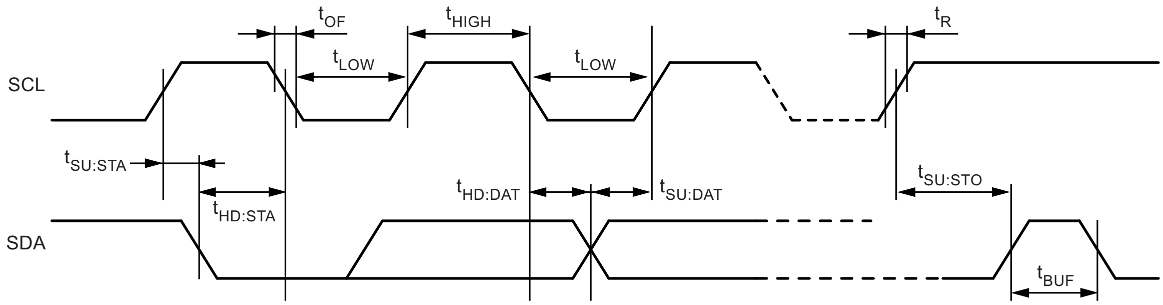
The following data is based on simulations and characterizations. Parameters listed in Table 25-8 are not tested in production. Symbols refer to Figure 25-3.

Table 25-8. Two-wire Serial Interface Characteristics

Parameter	Condition	Symbol	Min	Max	Unit
Input low voltage		V_{IL}	–0.5	0.3V_{CC}	V
Input high voltage		V_{IH}	0.7V_{CC}	V_{CC} + 0.5	V
Hysteresis of Schmitt trigger inputs	$V_{CC} \geq 2.7V$	V_{HYS}	0.05V_{CC}	–	V
	$V_{CC} < 2.7V$		0		
Output low voltage	3mA sink current	V_{OL}	0	0.4	V
Spikes suppressed by input filter		t_{SP}	0	50	ns
SCL clock frequency[1]	f_{CK} > max(16f_{SCL}, 250kHz)	f_{SCL}	0	400	kHz
Hold time (repeated) START condition		$t_{HD:STA}$	0.6	–	µs
Low period of SCL clock		t_{LOW}	1.3	–	µs
High period of SCL clock		t_{HIGH}	0.6	–	µs
Setup time for repeated START condition		$t_{SU:STA}$	0.6	–	µs
Data hold time		$t_{HD:DAT}$	0	0.9	µs
Data setup time		$t_{SU:DAT}$	100	–	ns
Setup time for STOP condition		$t_{SU:STO}$	0.6	–	µs
Bus free time between STOP and START condition		t_{BUF}	1.3	–	µs

Note: 1. f_{CK} = CPU clock frequency.

Figure 25-3. Two-wire Serial Bus Timing



25.7 Analog-To-Digital Converter

Table 25-9. ADC Characteristics, Single-Ended Channels. T = –40°C to +125°C

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Resolution					10	Bits
Total Unadjusted Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz	TUE		5.0	10	LSB
	V_{REF} = 4V, V_{CC} = 4V 200kHz noise canceler			4.5	8	LSB
Integral Non Linearity (INL) (accuracy after offset and gain calibration)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz	INL		0.6	2.0	LSB
	V_{REF} = Int 1.1V Ref, V_{CC} = 4V, ADC clock = 200kHz.			0.8	2.5	LSB
Differential Non Linearity (DNL)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz	DNL		±0.5	±0.9	LSB
	V_{REF} = Int 1.1V Ref, V_{CC} = 4V, ADC clock = 200kHz.			±0.5	±0.95	LSB
Gain error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		–1.0	0.5	1.0	%
Offset error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200kHz		–40	+20	+40	mV
	V_{REF} = Int 1.1V Ref, V_{CC} = 4V, ADC clock = 200kHz.		–70	+30	+70	mV
Conversion time	Free-running conversion		13		25	Clocks
Clock frequency			50		200	kHz
Input voltage		V_{IN}	GND		V_{REF}	V
Input bandwidth				38.5		kHz
External voltage reference		A_{REF}	2.0		V_{CC}	V
Internal voltage reference		V_{INT}	1.0	1.1	1.2	V
Reference input resistance at 5V		R_{REF}	22.4	32	41.6	kΩ
Analog input resistance		R_{AIN}		100		MΩ
ADC conversion output			0		1023	LSB

25.8 Analog Comparator

Table 25-10. Analog Comparator Characteristics, T_A = –40°C to +125°C

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Input offset voltage	GND < V_{IN} < V_{CC}	V_{AIO}	–80	10	80	mV
Input leakage current	V_{CC} = 5V, V_{IN} = V_{CC} / 2	I_{LAC}	–50		50	nA
Analog propagation delay (from saturation to slight overdrive)	V_{CC} = 2.7V	t_{APD}		750[1]		ns
	V_{CC} = 4.0V			500[1]		
Analog propagation delay (large step change)	V_{CC} = 2.7V			100[1]		
	V_{CC} = 4.0V			75[1]		
Digital propagation delay	V_{CC} = 2.7V to 5.5V	t_{DPD}		1[1]	2[1]	CLK

Note: 1. Values derived from design simulation.

25.9 Parallel Programming

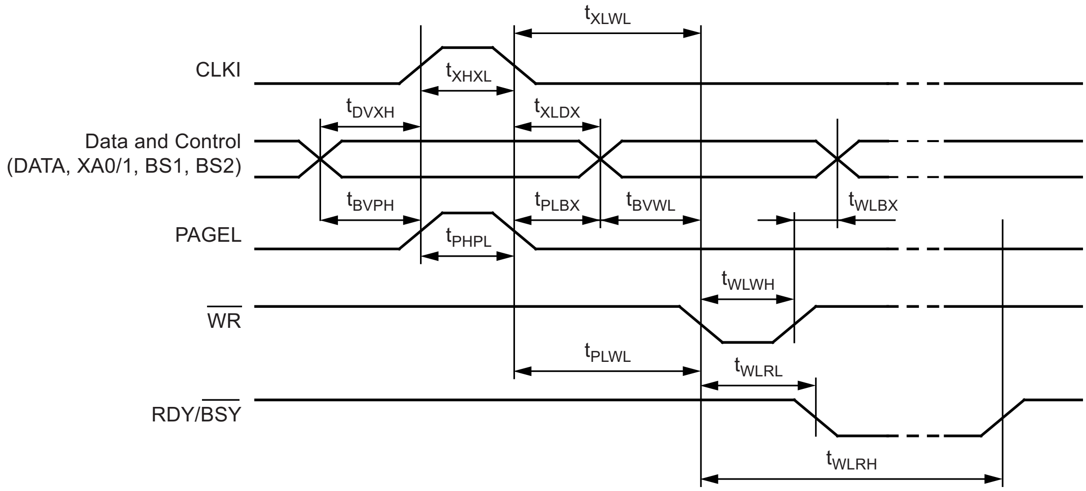


Figure 25-4. Parallel Programming Timing, Including Some General Timing Requirements

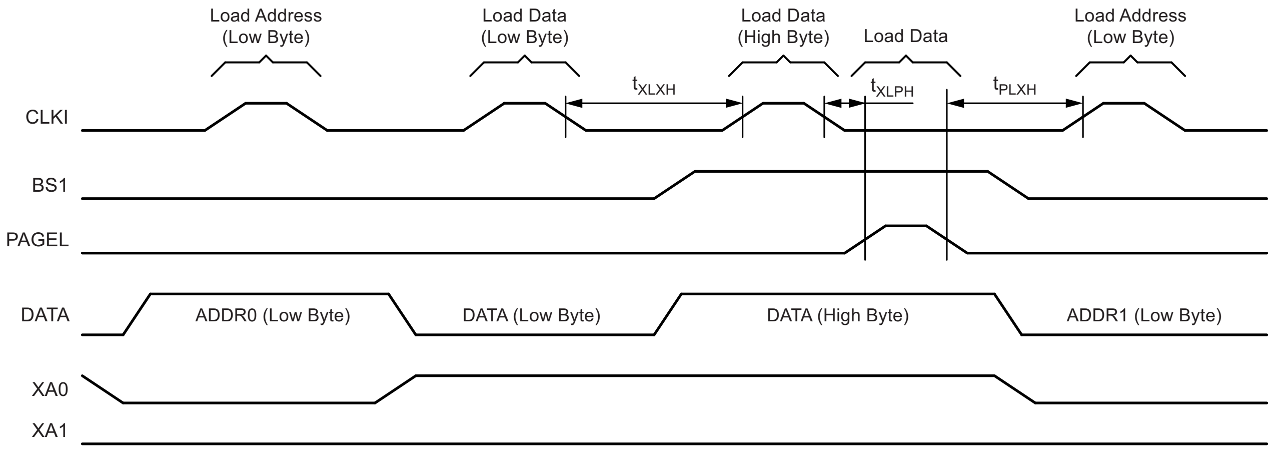
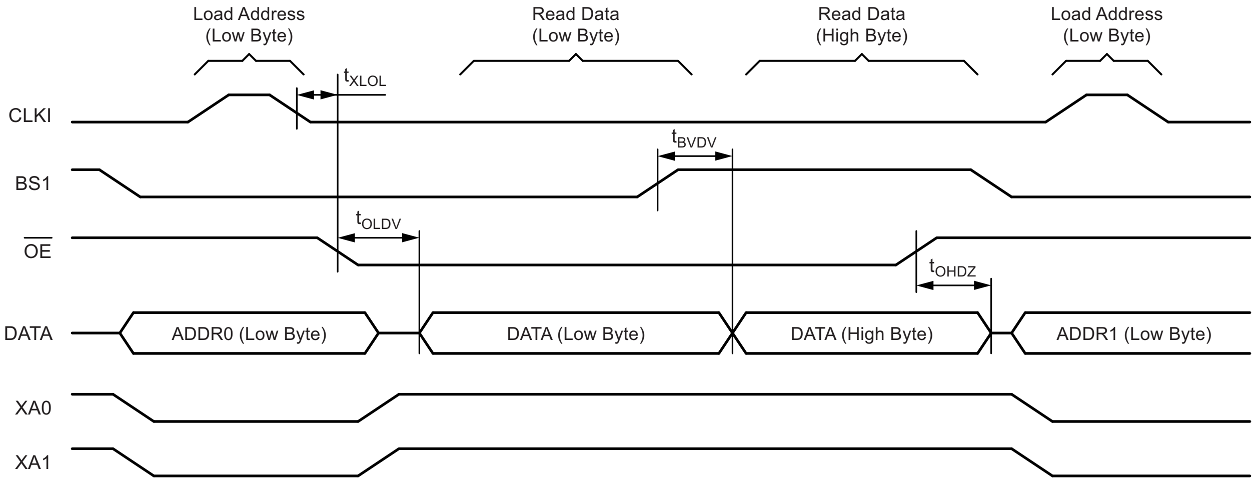


Figure 25-5. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]

Note: 1. The timing requirements shown in Figure 25-4 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 25-6. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 25-4 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 25-11. Parallel Programming Characteristics, T_A = 25°C, V_{CC} = 5V

Parameter	Symbol	Min	Typ	Max	Units
Programming enable voltage	V_{PP}	11.5		12.5	V
Programming enable current	I_{PP}			250	µA
Data and control valid before CLKI high	t_{DVXH}	67			ns
CLKI low to CLKI high	t_{XLXH}	200			ns
CLKI pulse width high	t_{XHXL}	150			ns
Data and control hold after CLKI low	t_{XLDX}	67			ns
CLKI low to $\overline{WR}$ low	t_{XLWL}	0			ns
CLKI low to PAGEL high	t_{XLPH}	0			ns
PAGEL low to CLKI high	t_{PLXH}	150			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}	3.7		9	ms
CLKI 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.

25.10 Serial Programming

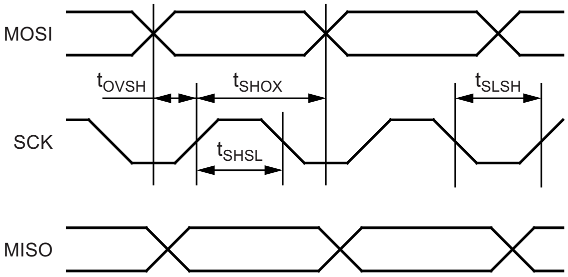


Figure 25-7. Serial Programming Timing

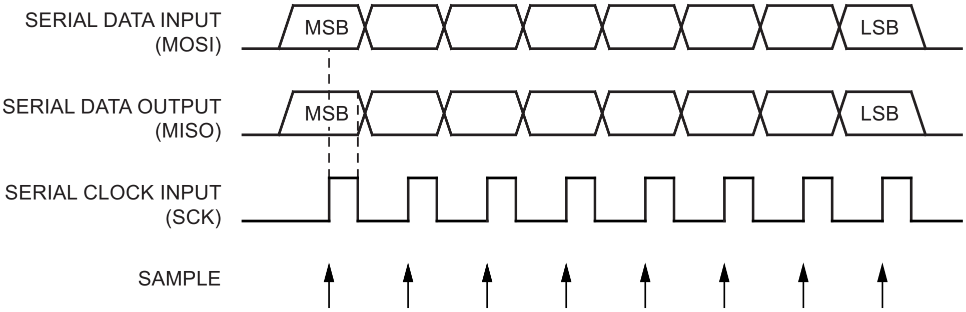


Figure 25-8. Serial Programming Waveform

Table 25-12. Serial Programming Characteristics, T_A = –40°C to +125°C

Parameter	Symbol	Min	Typ	Max	Unit
Oscillator frequency at V_{CC} = 2.7V to 5.5V	$1/t_{CLCL}$	0		1	MHz
Oscillator period at V_{CC} = 2.7V to 5.5V	t_{CLCL}	1000			ns
Oscillator frequency at V_{CC} = 4.5V to 5.5V	$1/t_{CLCL}$	0		6	MHz
Oscillator period at V_{CC} = 4.5V to 5.5V	t_{CLCL}	167			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

26. Typical Characteristics

The data contained in this section is largely based on simulations and characterization of similar devices in the same process and design methods. Thus, the data should be treated as indications of how the part will behave.

The following charts show typical behavior. These figures are not tested during manufacturing. During characterisation, devices are operated at frequencies higher than test limits but they are not guaranteed to function properly at frequencies higher than the ordering code indicates.

All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. Current consumption is a function of several factors such as operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed, and ambient temperature. The dominating factors are operating voltage and frequency.

A sine wave generator with rail-to-rail output is used as the clock source, but current consumption in power-down mode does not depend on clock selection. The difference between current consumption in power-down mode with watchdog timer enabled and power-down mode with watchdog timer disabled represents the differential current drawn by the watchdog timer.

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

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

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

26.1 Current Consumption in Active Mode

Figure 26-1. Active Supply Current versus Low Frequency (0.1MHz to 1.0MHz), –40°C up to +125°C

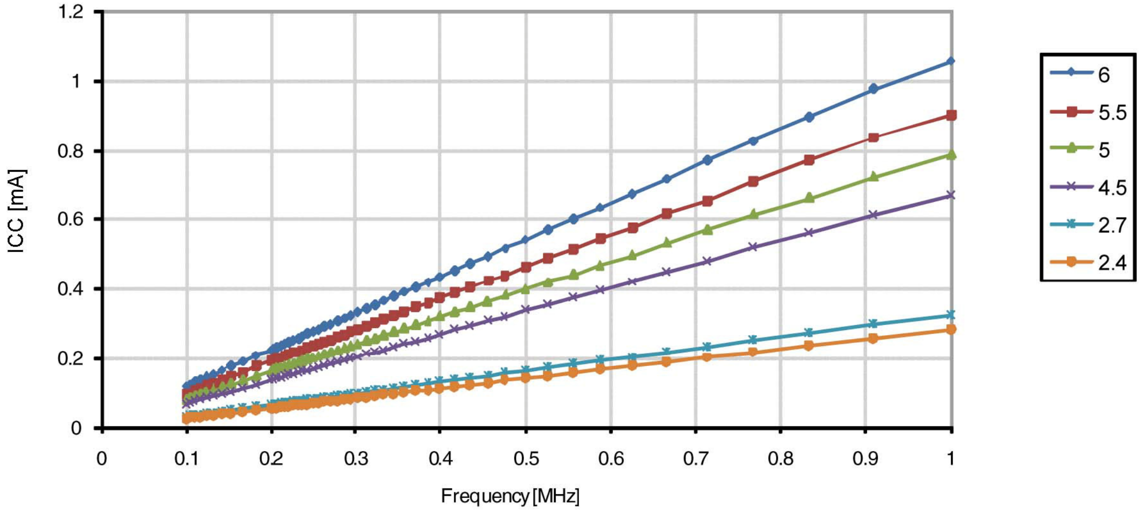


Figure 26-2. Active Supply Current versus Frequency (1MHz to 12MHz), –40°C up to +125°C

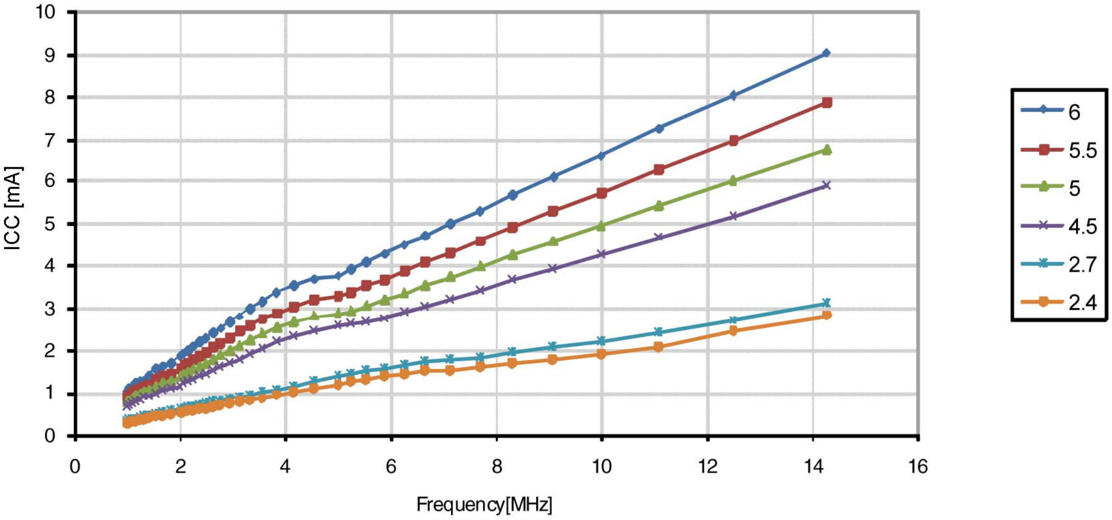


Figure 26-3. Active Supply Current versus V_{CC} (Internal Oscillator, 8MHz)

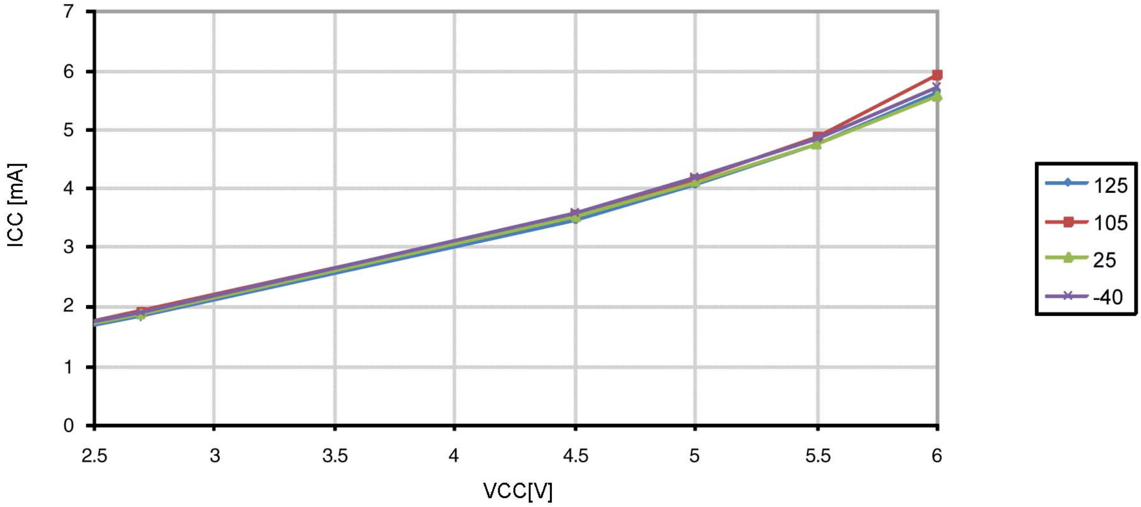


Figure 26-4. Active Supply Current versus V_{CC} (Internal Oscillator, 1MHz)

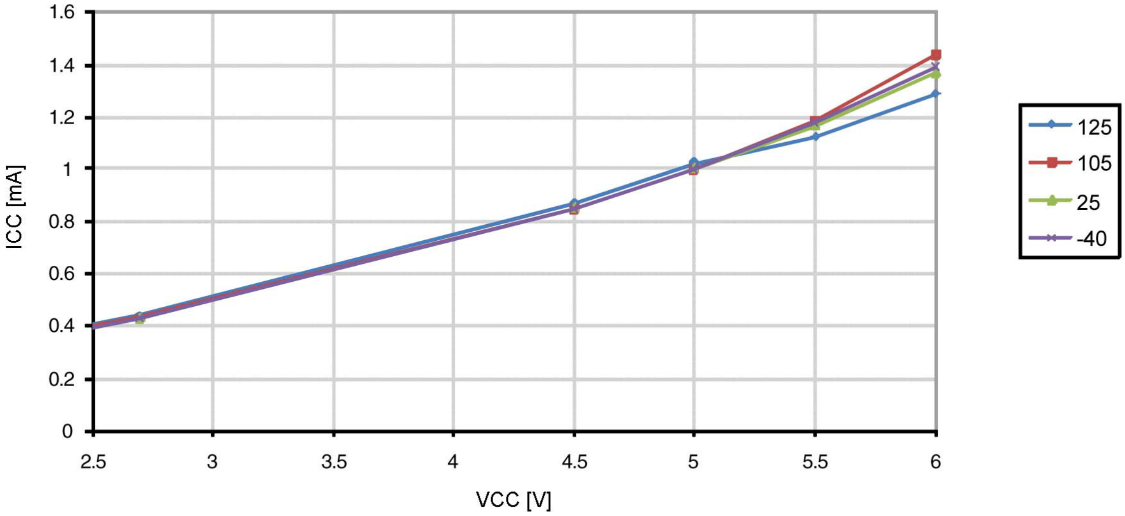
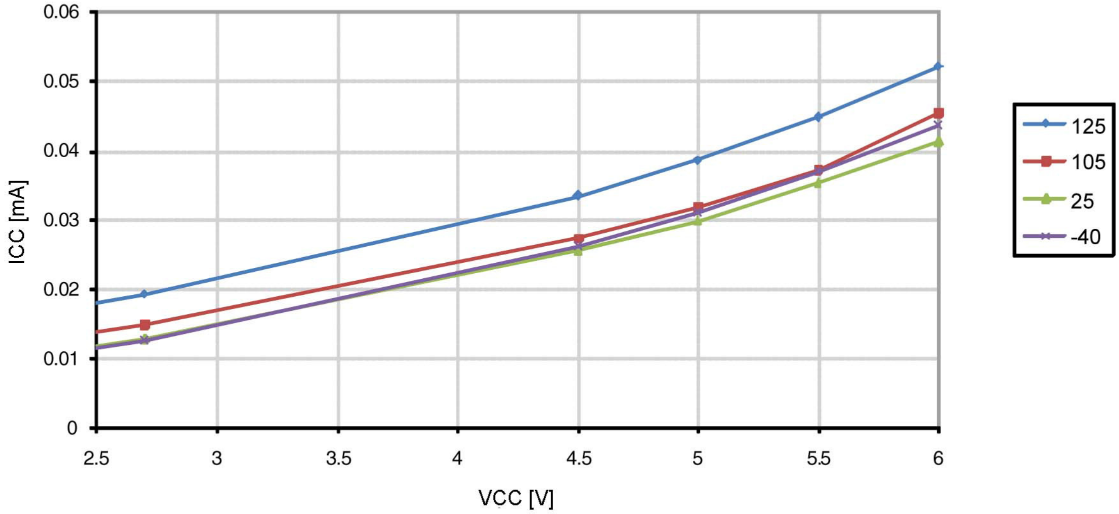


Figure 26-5. Active Supply Current versus V_{CC} (Internal Oscillator, 32kHz)



26.2 Current Consumption in Idle Mode

Figure 26-6. Idle Supply Current versus Low Frequency (0.1MHz - 1.0MHz), –40°C up to +125°C

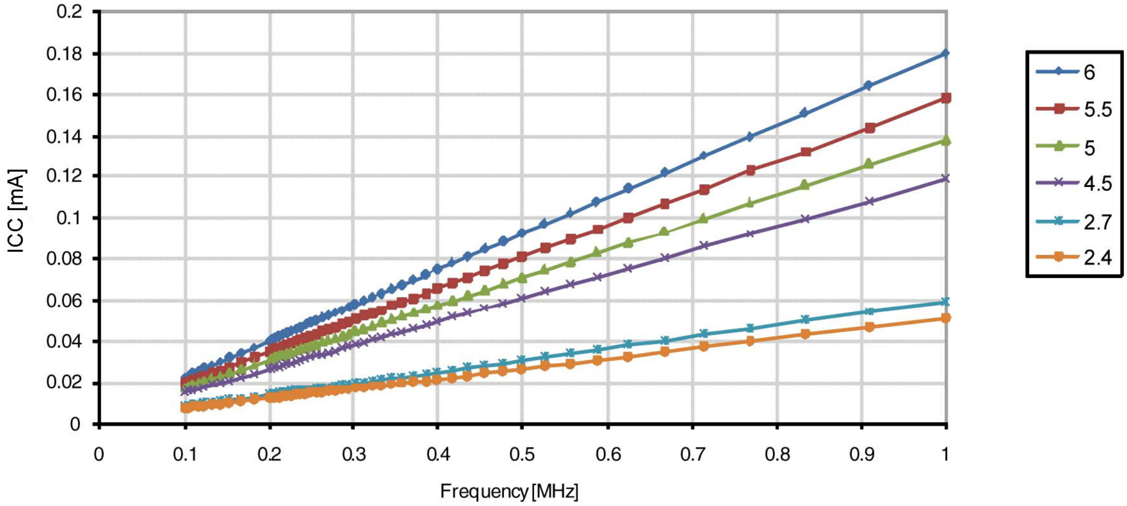


Figure 26-7. Idle Supply Current versus Frequency (1MHz - 12MHz), –40°C up to +125°C

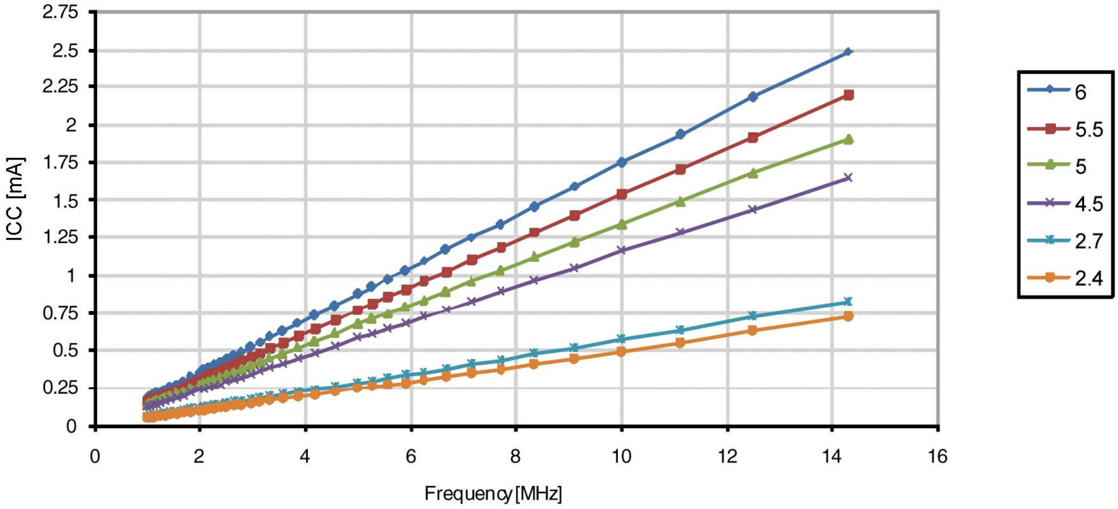


Figure 26-8. Idle Supply Current versus V_{CC} (Internal Oscillator, 8MHz)

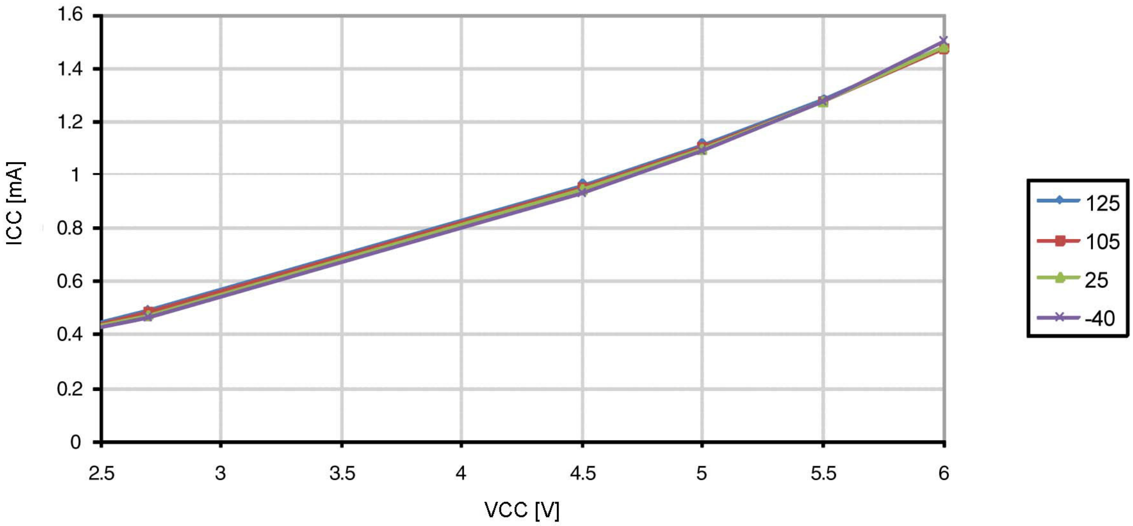


Figure 26-9. Idle Supply Current versus V_{CC} (Internal Oscillator, 1MHz)

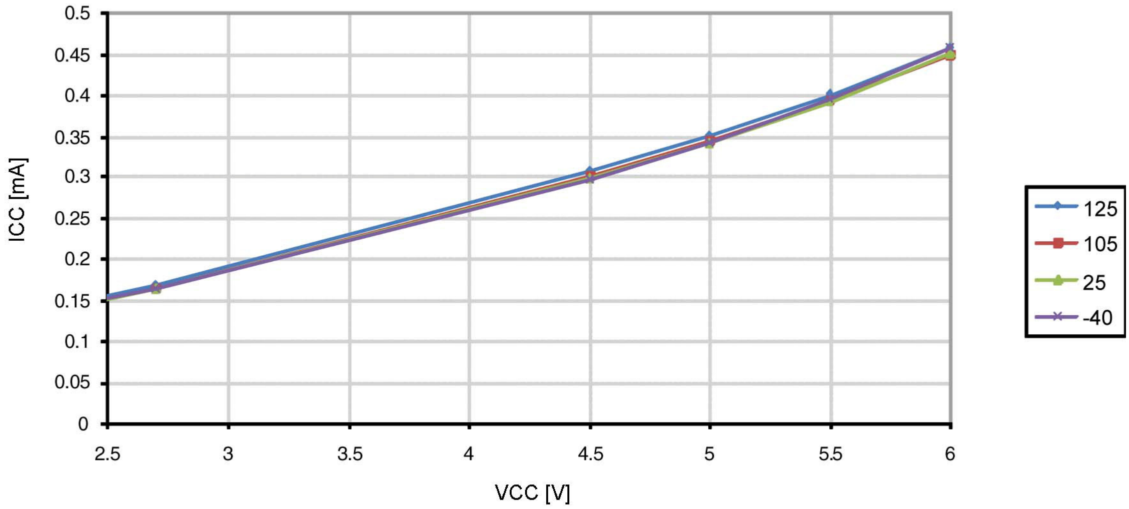
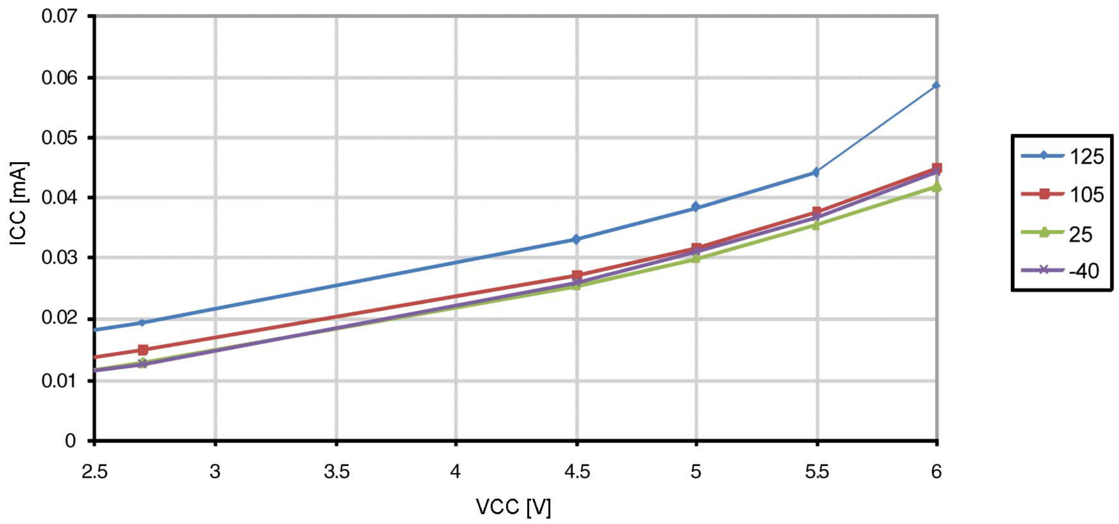


Figure 26-10. Idle Supply Current versus V_{CC} (Internal Oscillator, 32kHz)



26.3 Current Consumption in Power-Down Mode

Figure 26-11. Power-Down Supply Current versus V_{CC} (Watchdog Timer Disabled)

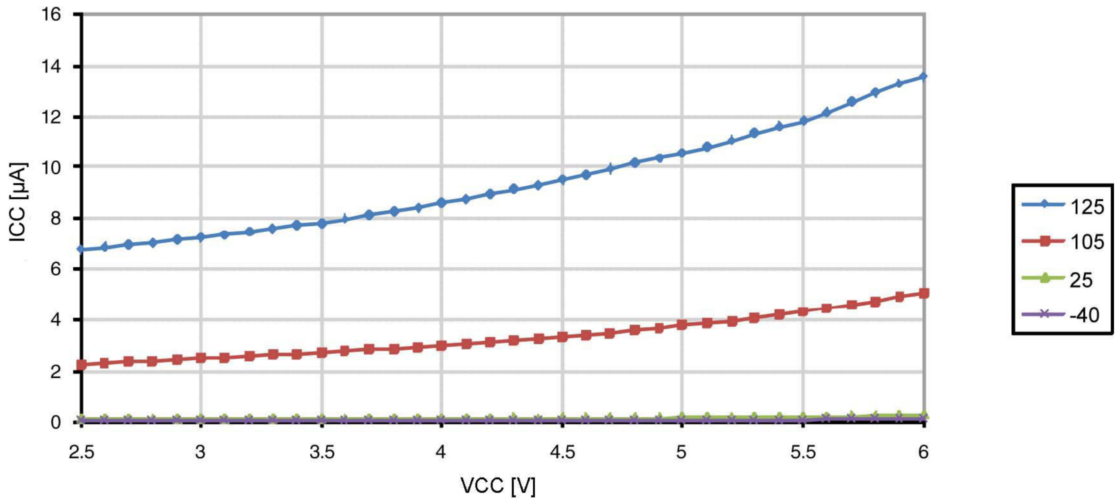
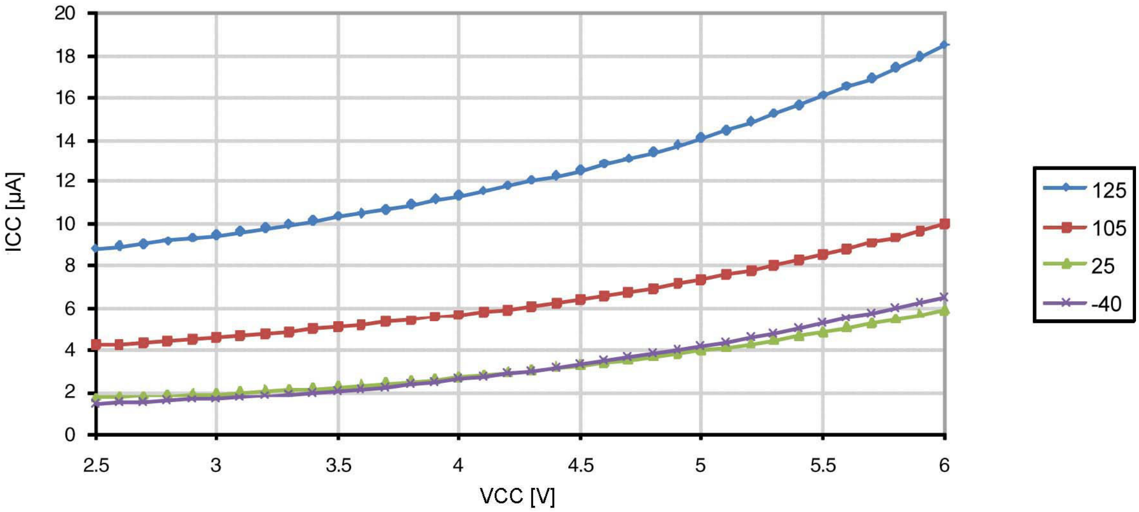


Figure 26-12. Power-Down Supply Current versus V_{CC} (Watchdog Timer Enabled)



26.4 Current Consumption of Peripheral Units

Figure 26-13. Reset Current versus V_{CC}, Excluding Current through the Reset Pull-Up and No Clock

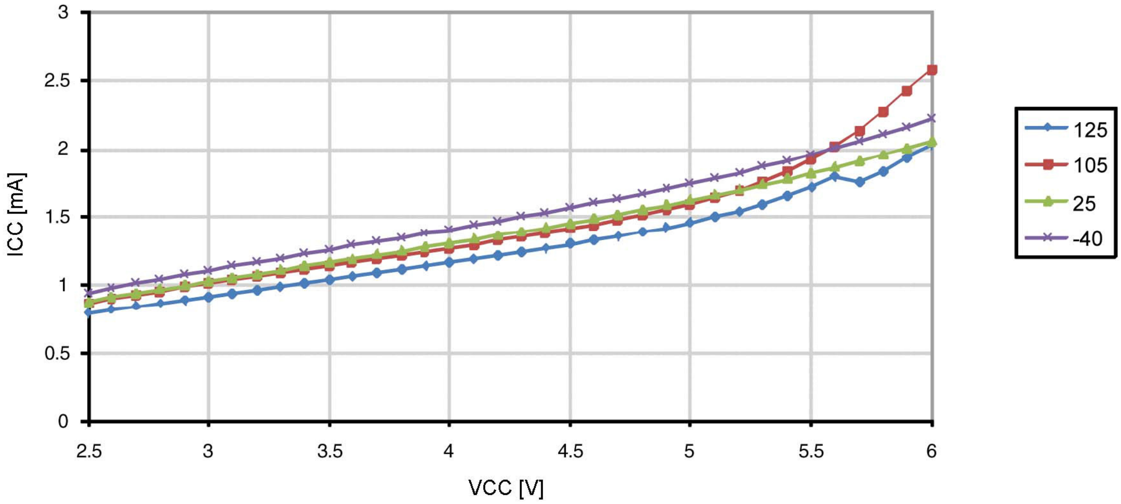


Figure 26-14. Brown-out Detector Current versus V_{CC}, Normal BOD

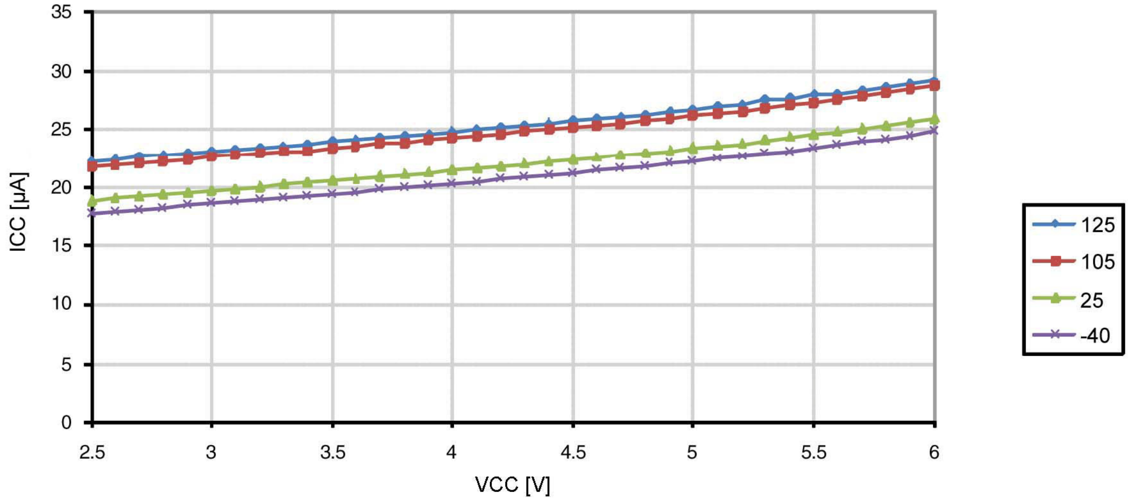
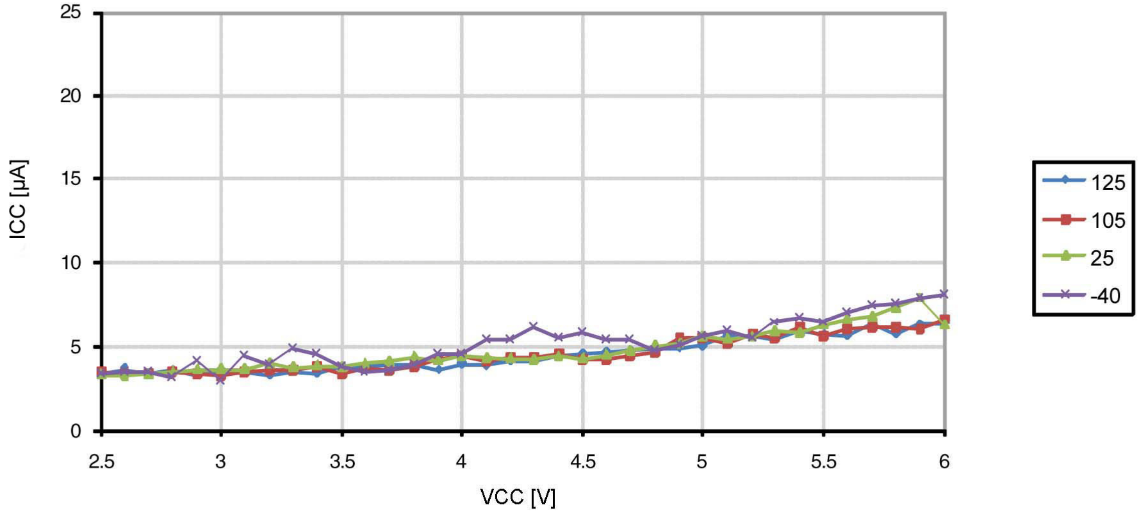
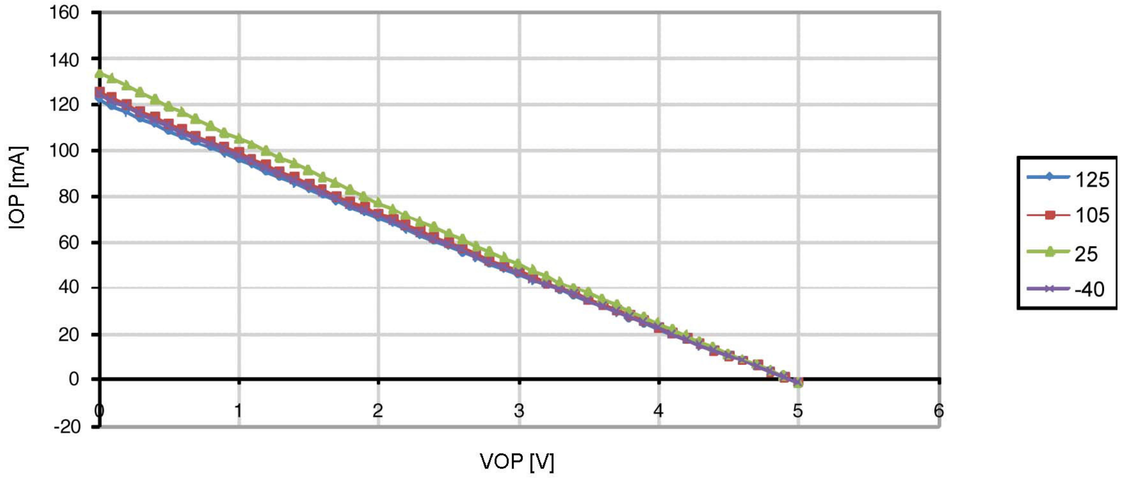


Figure 26-15. Sampled Brown-out Detector Current versus V_{CC}, Sampled BOD, Average of Five Current Measurements



26.5 Pull-Up Resistors

Figure 26-16. I/O Pin Pull-Up Resistor Current versus Input Voltage (V_{CC} = 5V)



26.6 Input Thresholds

Figure 26-17. V_{IH}: Input Threshold Voltage versus V_{CC} (I/O Pin, Read as “1”)

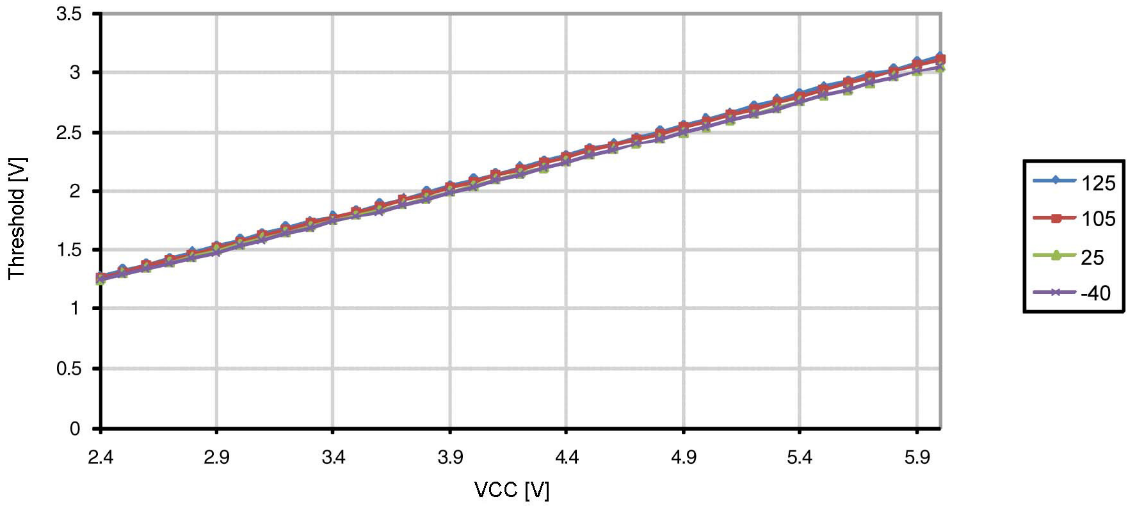


Figure 26-18. V_{IL}: Input Threshold Voltage versus V_{CC} (I/O Pin, Read as "0")

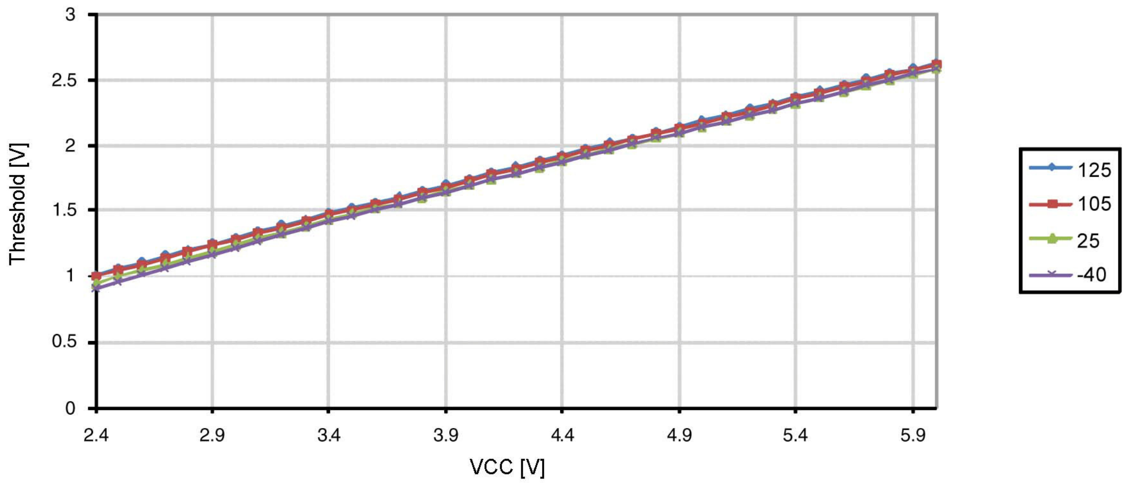
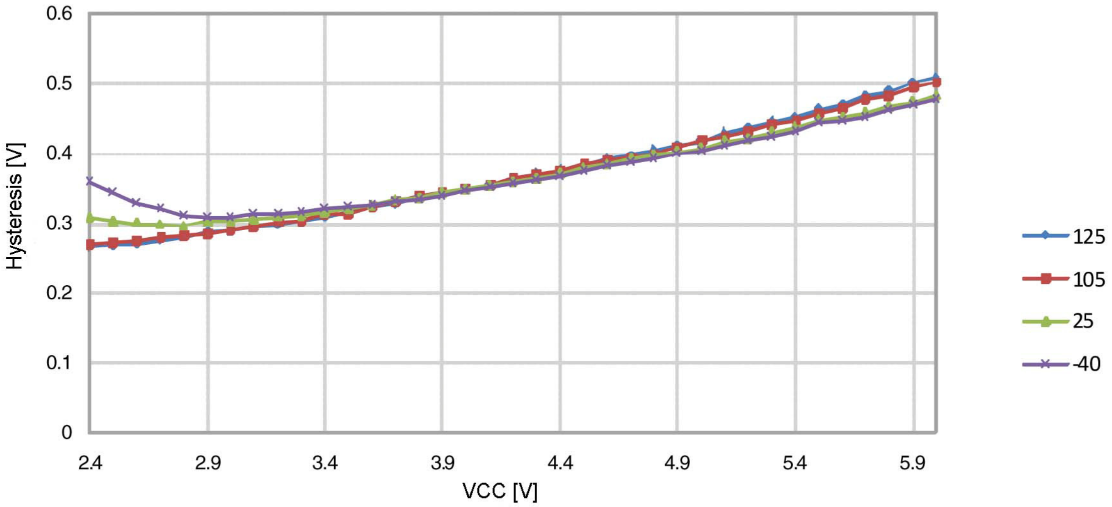


Figure 26-19. V_{IH}-V_{IL}: Input Hysteresis versus V_{CC} (Reset Pin as I/O)



26.7 Output Driver Strength

Figure 26-20. V_{OH}: I/O Pin Output Voltage versus Source Current, Low-power Pins (V_{CC} = 3V)

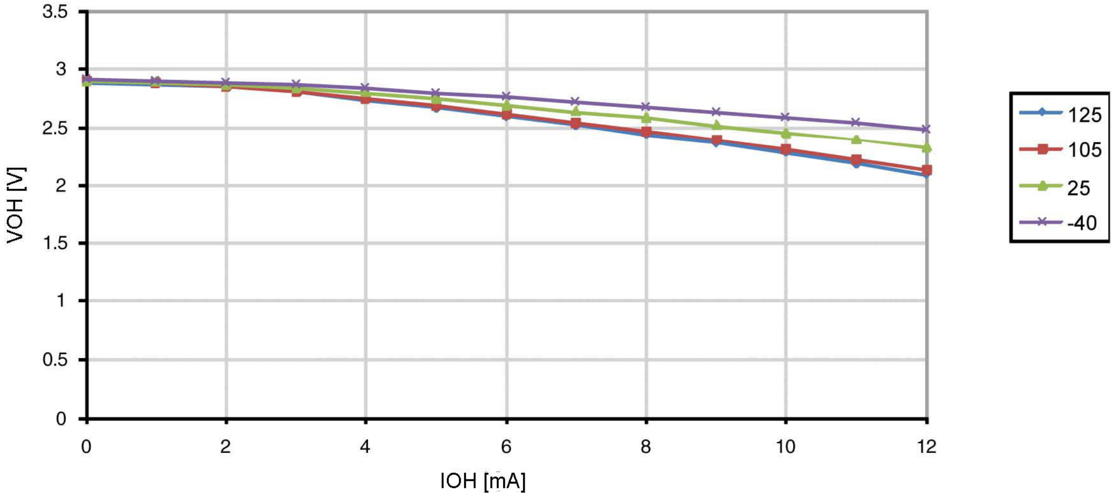


Figure 26-21. V_{OH}: I/O Pin Output Voltage versus Source Current, Low-power Pins (V_{CC} = 5V)

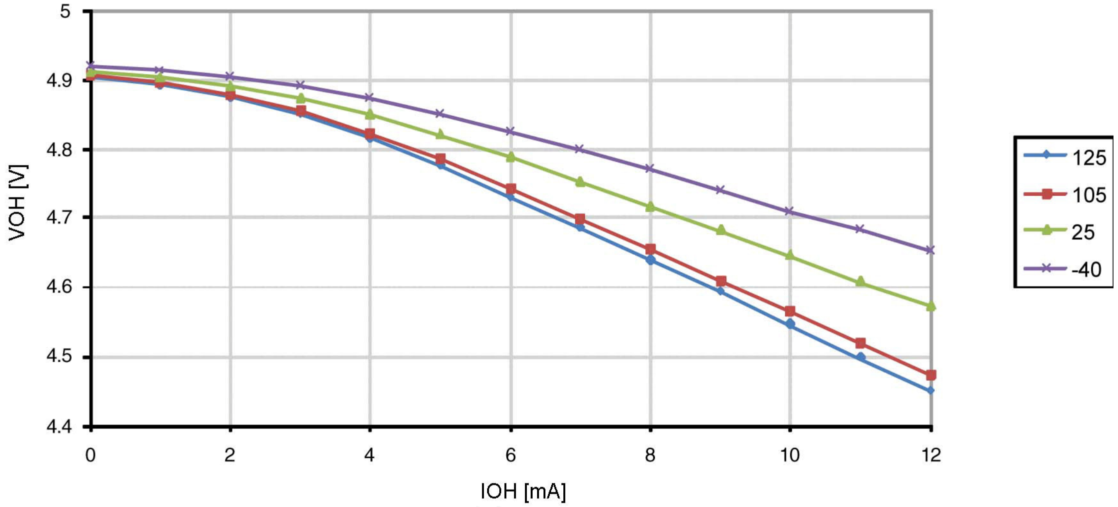


Figure 26-22. V_{OL}: I/O Pin Output Voltage versus Sink Current, Low-power Pins (V_{CC} = 3V)

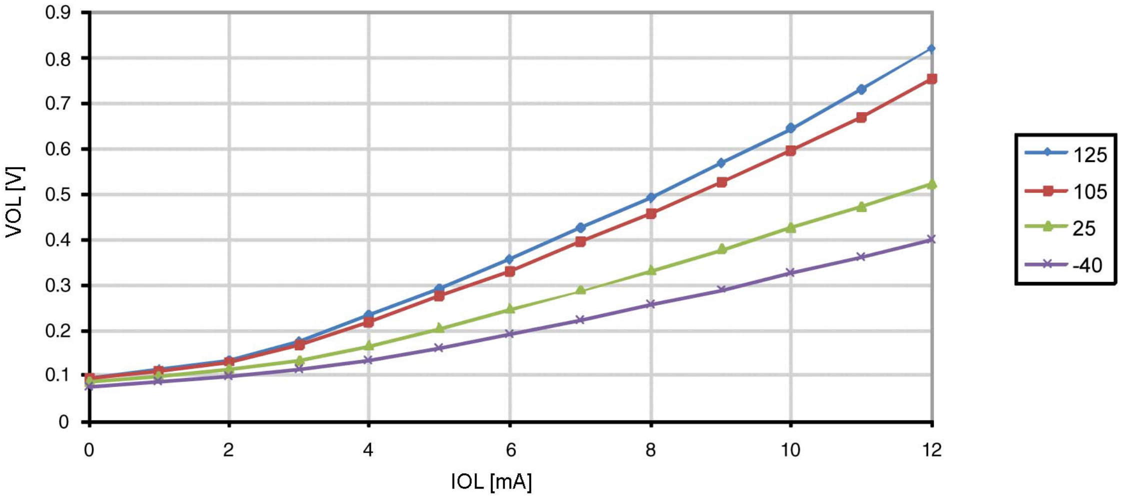


Figure 26-23. V_{OL}: I/O Pin Output Voltage versus Sink Current, Low-power Pins (V_{CC} = 5V)

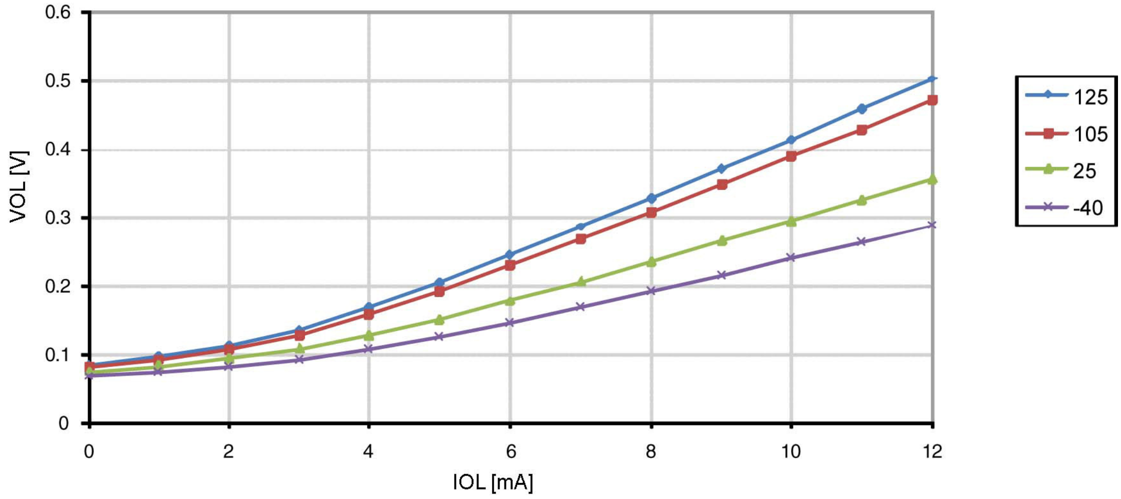


Figure 26-24. V_{OH}: **Reset Pin Output Voltage versus Sink Current (Reset Pin as I/O,** V_{CC} **= 3V)**

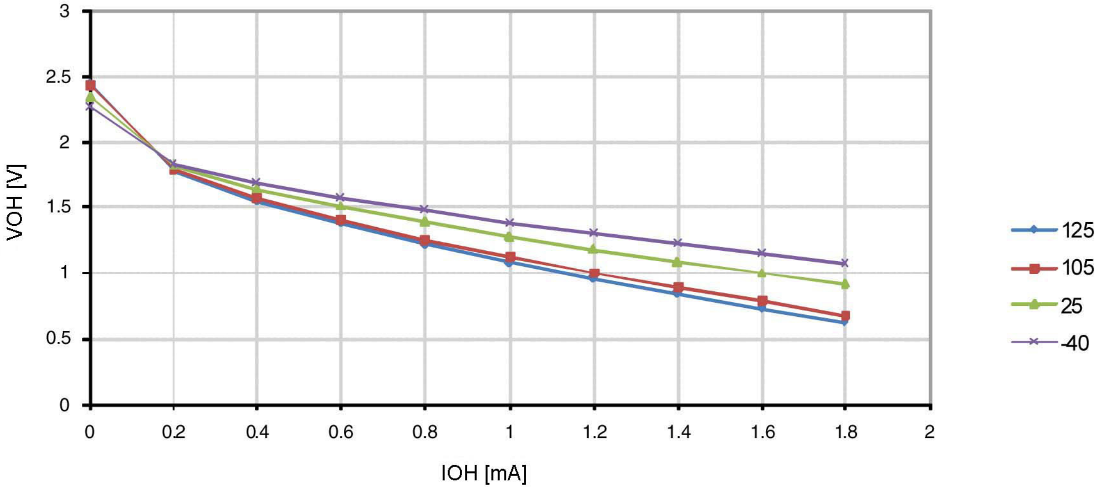


Figure 26-25. V_{OH}: **Reset Pin Output Voltage versus Sink Current (Reset Pin as I/O,** V_{CC} **= 5V)**

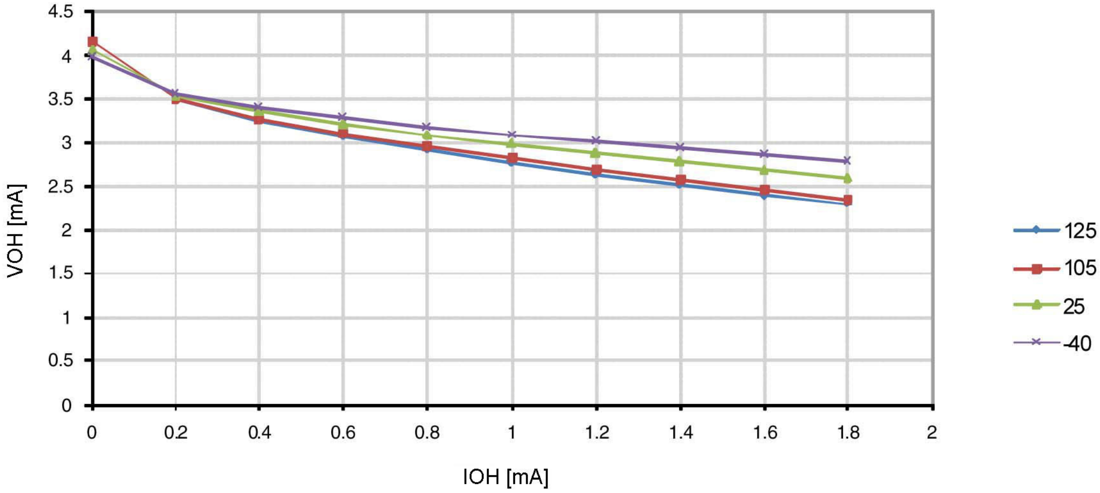


Figure 26-26. V_{OL}: Reset Pin Output Voltage versus Sink Current (Reset Pin as I/O, V_{CC} = 5V)

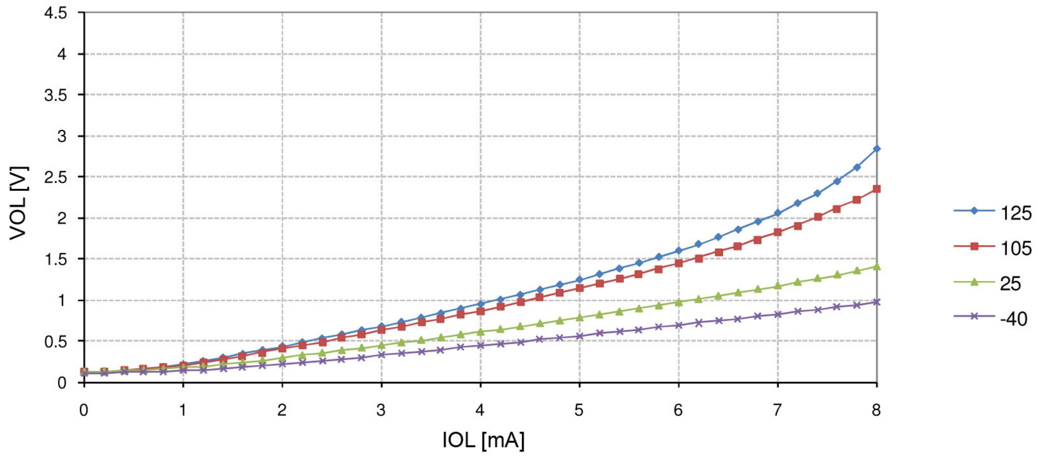
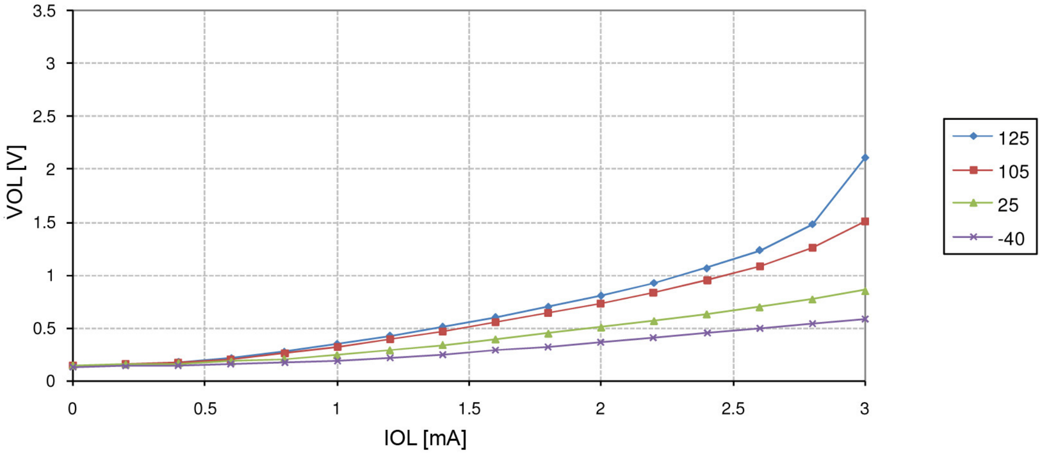


Figure 26-27. V_{OL}: Reset Pin Output Voltage versus Sink Current (Reset Pin as I/O, V_{CC} = 3V)



26.8 BOD

Figure 26-28. BOD Threshold versus Temperature, (BODLEVEL = 4.3V)

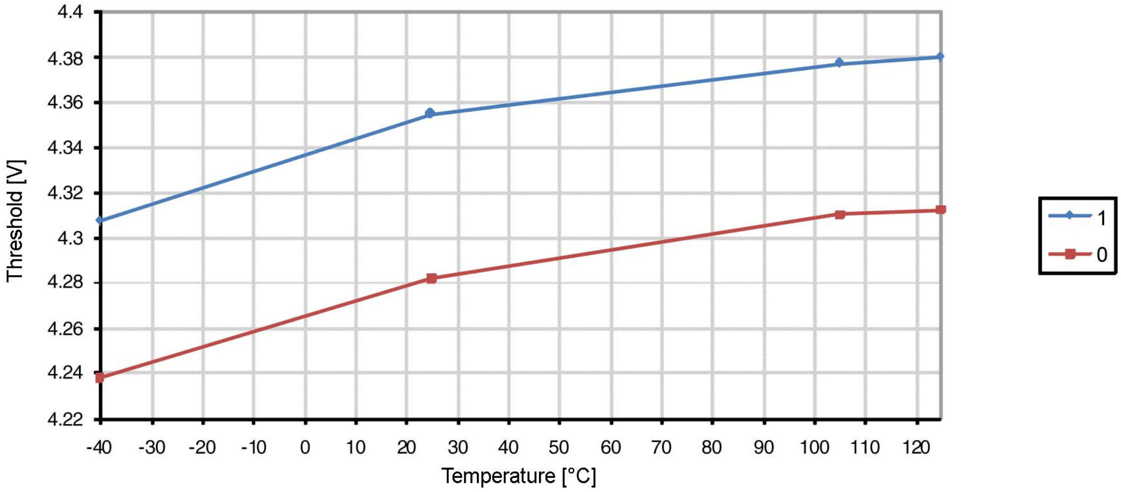


Figure 26-29. BOD Threshold versus Temperature (BODLEVEL = 2.7V)

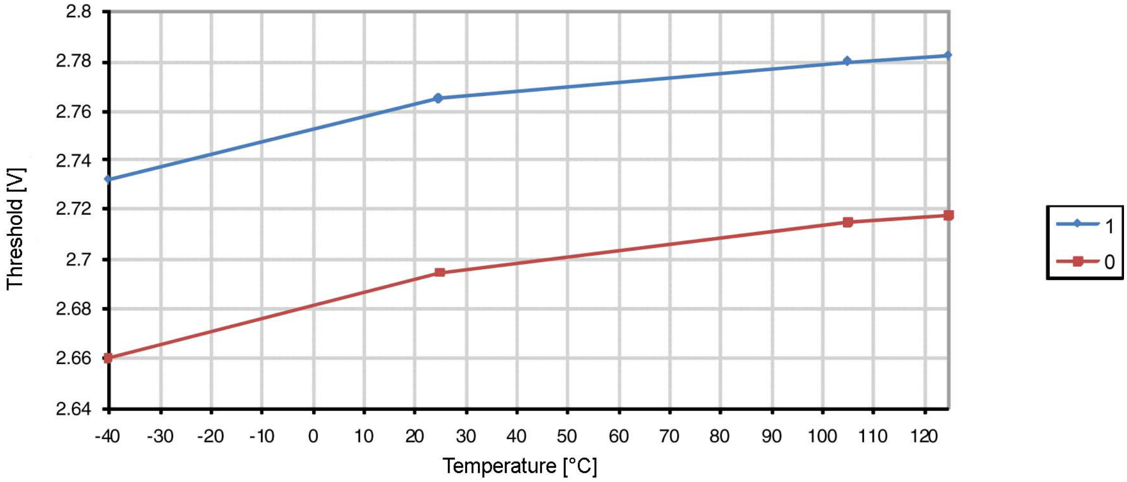


Figure 26-30. Sampled BOD Threshold versus Temperature (BODLEVEL = 4.3V)

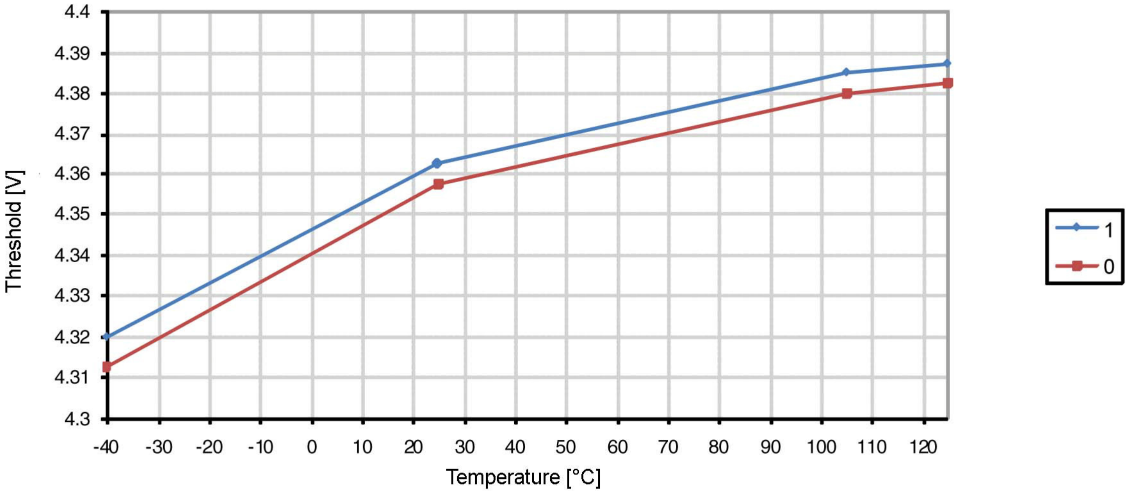
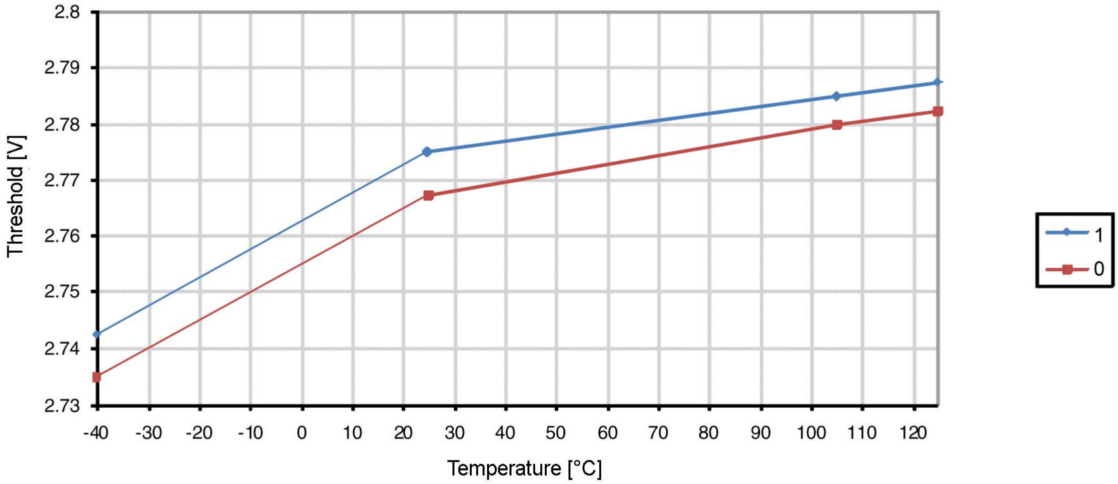
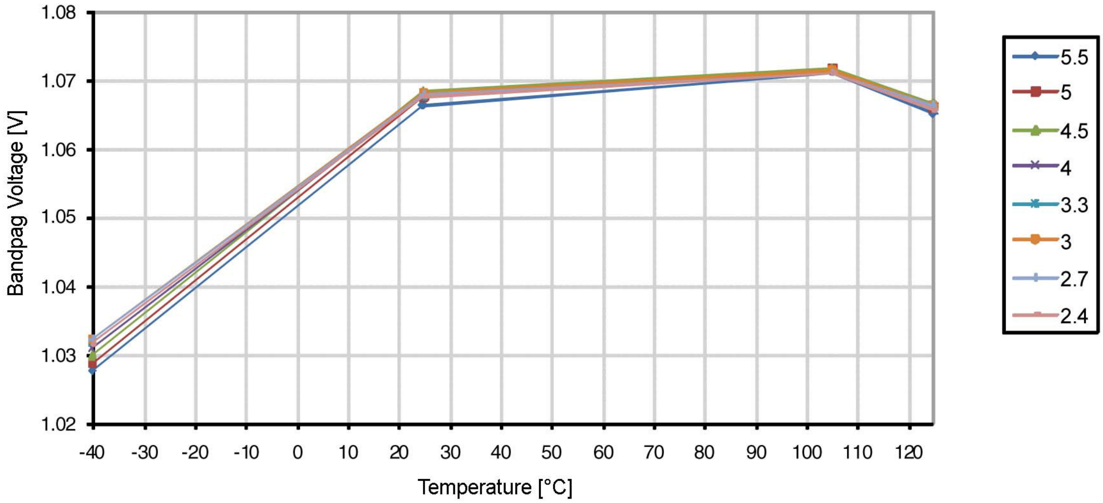


Figure 26-31. Sampled BOD Threshold versus Temperature (BODLEVEL = 2.7V)



26.9 Band-gap Voltage

Figure 26-32. Band-gap Voltage versus Temperature



26.10 Reset

Figure 26-33. V_{IH}: Input Threshold Voltage versus V_{CC} (Reset Pin, Read as "1")

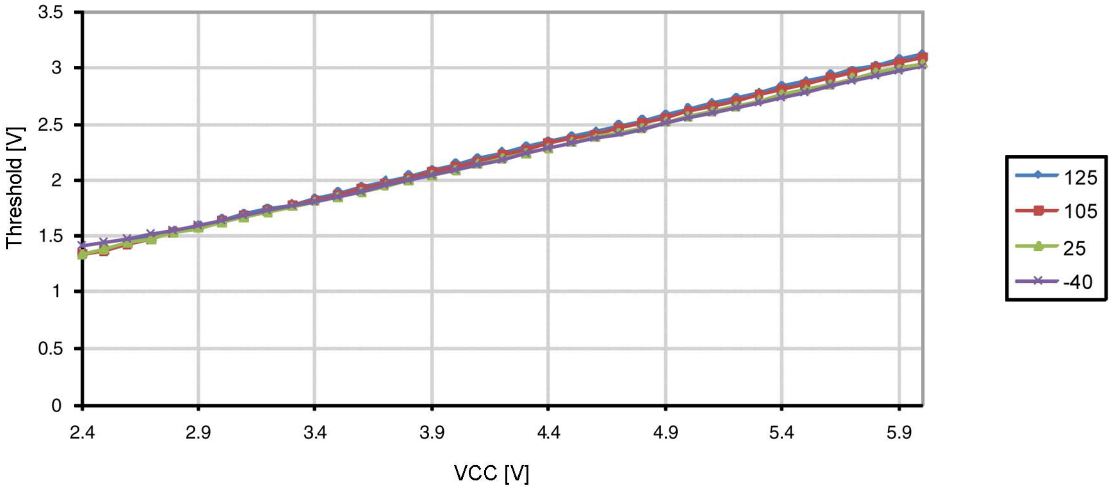


Figure 26-34.V_{IL}: Input Threshold Voltage versus V_{CC} (Reset Pin, Read as “0”)

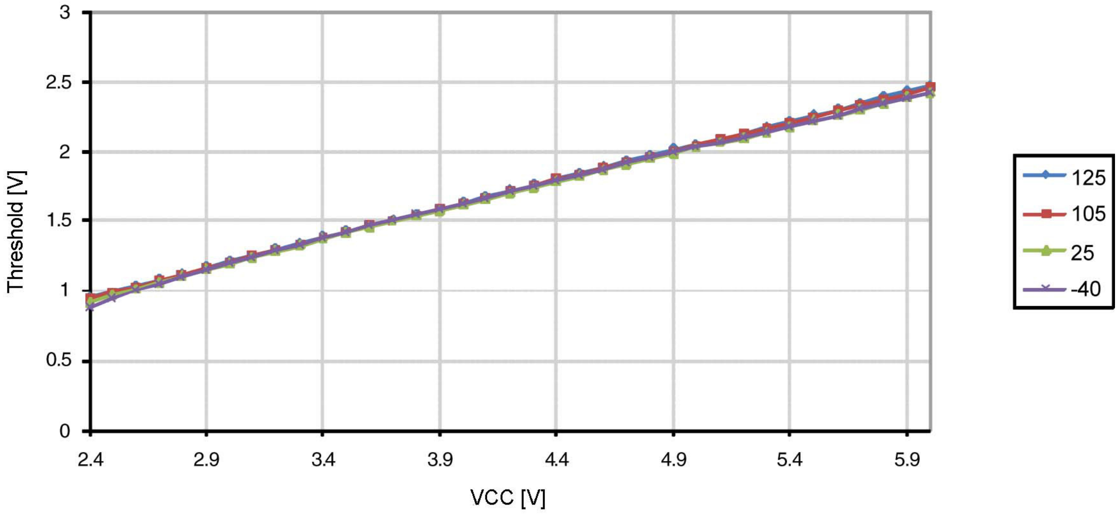
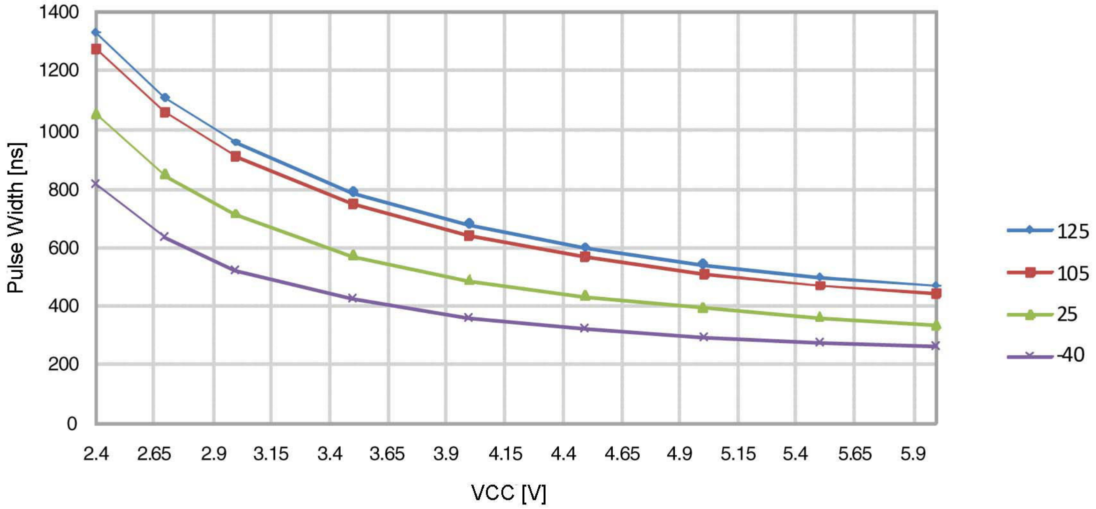


Figure 26-35. Minimum Reset Pulse Width versus V_{CC}



26.11 Internal Oscillator Speed

Figure 26-36. Calibrated Oscillator Frequency (Nominal = 8MHz) versus V_{CC}

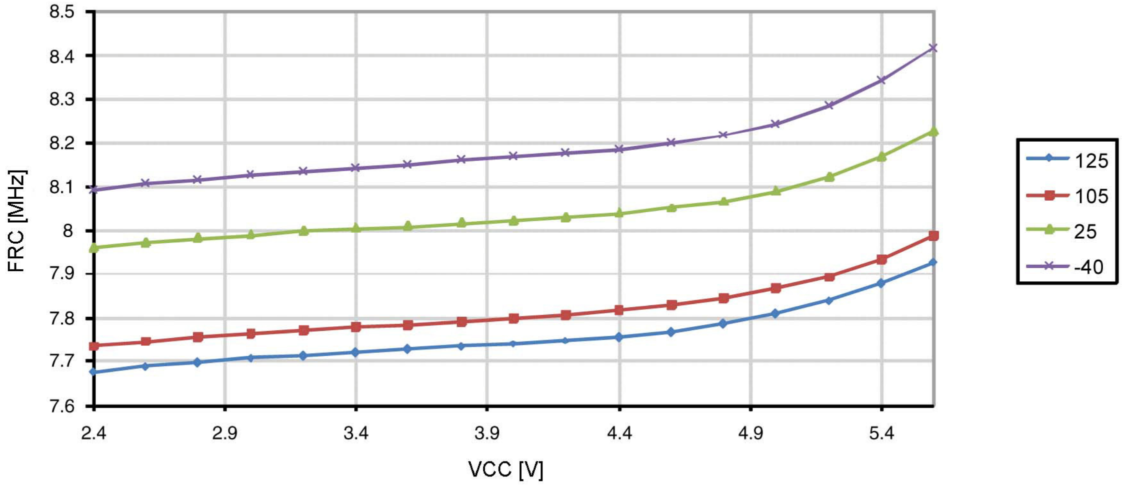


Figure 26-37. Calibrated Oscillator Frequency (Nominal = 8MHz) versus Temperature

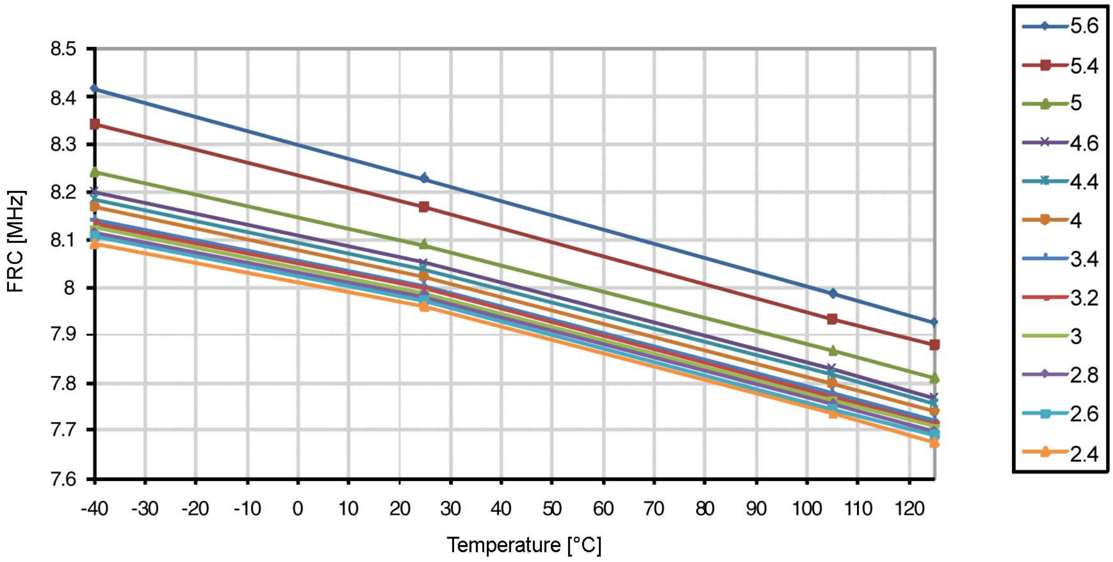


Figure 26-38. Calibrated Oscillator Frequency (Nominal = 8MHz) versus OSCCAL Value

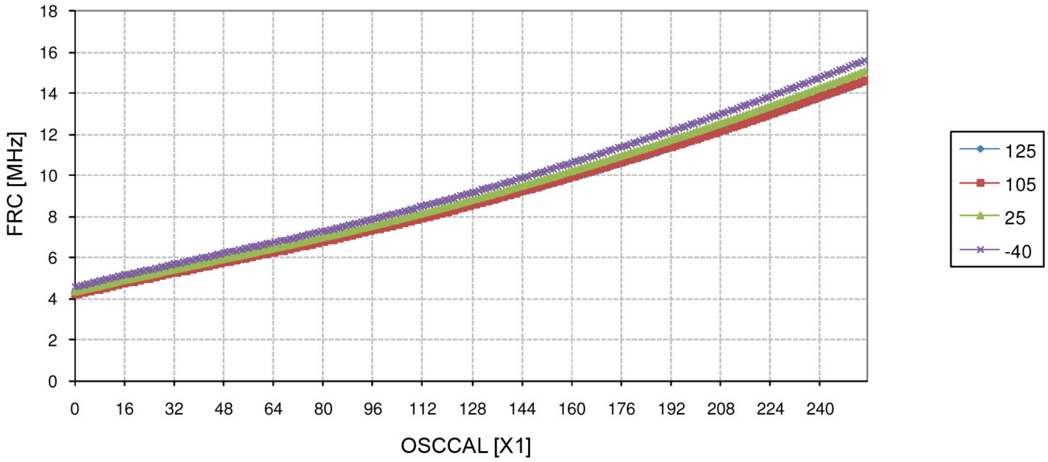


Figure 26-39. Calibrated Oscillator Frequency (Nominal = 1MHz) versus V_{CC}

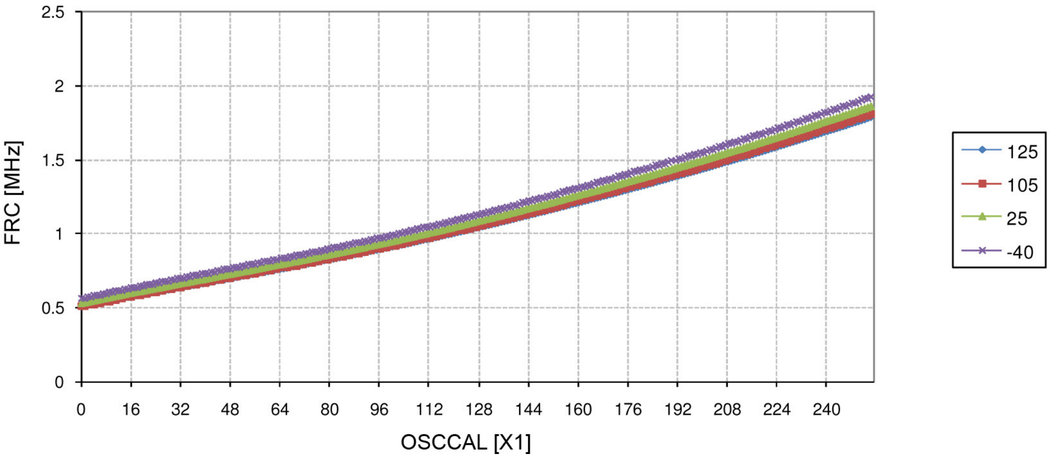


Figure 26-40. Calibrated Oscillator Frequency (Nominal = 1MHz) versus Temperature

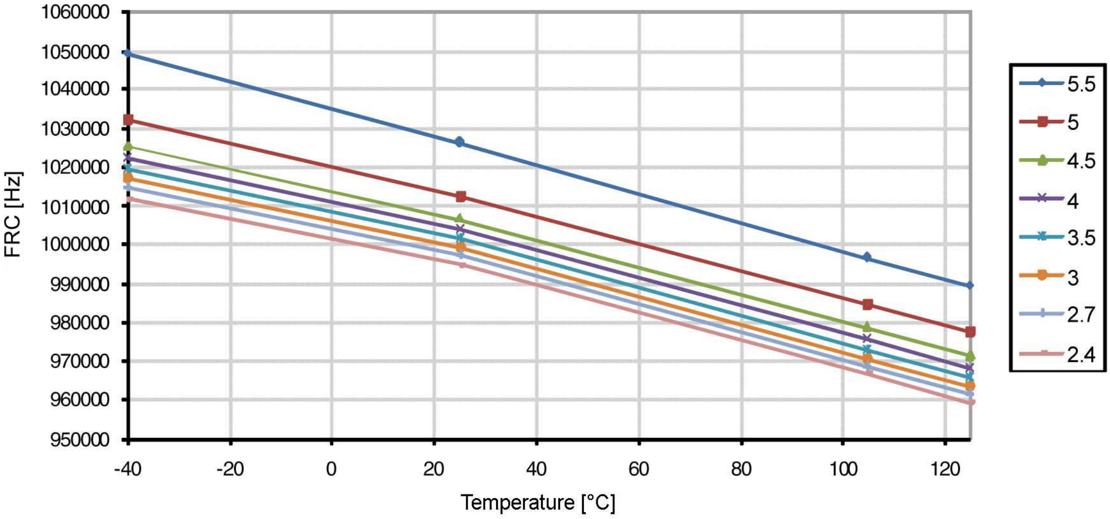
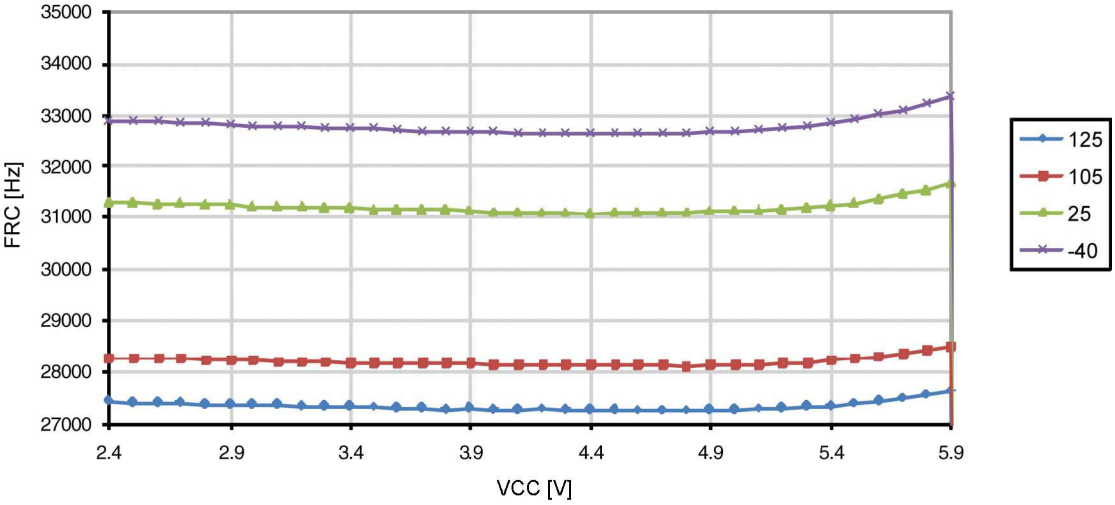


Figure 26-41. ULP Oscillator Frequency (Nominal = 32kHz) versus V_{CC}



27. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page(s)
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	Reserved	–	–	–	–	–	–	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	Reserved	–	–	–	–	–	–	–	–	
(0xFA)	Reserved	–	–	–	–	–	–	–	–	
(0xF9)	Reserved	–	–	–	–	–	–	–	–	
...	...	...	...	...	...	...	...	...	...	...
(0x85)	Reserved	–	–	–	–	–	–	–	–	
(0x84)	Reserved	–	–	–	–	–	–	–	–	
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	Reserved	–	–	–	–	–	–	–	–	
(0x81)	Reserved	–	–	–	–	–	–	–	–	
(0x80)	Reserved	–	–	–	–	–	–	–	–	
(0x7F)	TWSCRA	TWSHE	–	TWDIE	TWASIE	TWEN	TWSIE	TWPME	TWSME	121
(0x7E)	TWSCRB						TWAA	TWCMD[1:0]		122
(0x7D)	TWSSRA	TWDIF	TWASIF	TWCH	TWRA	TWC	TWBE	TWDIR	TWAS	123
(0x7C)	TWSA	TWI Slave Address Register								124
(0x7B)	TWSAM	TWI Slave Address Mask Register								125
(0x7A)	TWSD	TWI Slave Data Register								124
(0x79)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	157
(0x78)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	158
(0x77)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM01	USBS1	UCSZ11	UCSZ10	UCPOL1	159
(0x76)	UCSR1D	RXSIE1	RXS1	SFDE1						160
(0x75)	UBRR1H	USART1 Baud Rate Register High Byte								161
(0x74)	UBRR1L	USART1 Baud Rate Register Low Byte								161
(0x73)	UDR1	USART1 I/O Data Register								156
(0x72)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	106
(0x71)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	108
(0x70)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	109
(0x6F)	TCNT1H	Timer/Counter1 – Counter Register High Byte								109
(0x6E)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								109
(0x6D)	OCR1AH	Timer/Counter1 – Compare Register A High Byte								109
(0x6C)	OCR1AL	Timer/Counter1 – Compare Register A Low Byte								109
(0x6B)	OCR1BH	Timer/Counter1 – Compare Register B High Byte								110
(0x6A)	OCR1BL	Timer/Counter1 – Compare Register B Low Byte								110
(0x69)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								110
(0x68)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								110

Notes:

1. For compatibility with future devices, reserved bits should be written to “0” 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 logic one to them. Note that, unlike most other AVR®s, the CBI and SBI instructions only operate 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.

27. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page(s)
(0x67)	GTCCR	TSM	–	–	–	–	–	–	PSR10	113
(0x66)	OSCCAL1	–	–	–	–	–	–	CAL11	CAL10	33
(0x65)	OSCTCAL0B	Oscillator Temperature Compensation Register B								33
(0x64)	OSCTCAL0A	Oscillator Temperature Compensation Register A								32
(0x63)	OSCCAL0	CAL07	CAL06	CAL05	CAL04	CAL03	CAL02	CAL01	CAL00	32
(0x62)	DIDR2	–	–	–	–	–	ADC11D	ADC10D	ADC9D	187
(0x61)	DIDR1	–	–	–	–	ADC8D	ADC7D	ADC6D	ADC5D	186
(0x60)	DIDR0	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	AIN1D	AIN0D	AREFD	171, 186
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	15
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	14
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	14
0x3C (0x5C)	GIMSK	–	INT0	PCIE2	PCIE1	PCIE0	–	–	–	50
0x3B (0x5B)	GIFR	–	INTF0	PCIF2	PCIF1	PCIF0	–	–	–	51
0x3A (0x5A)	TIMSK	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	86, 110
0x39 (0x59)	TIFR	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	86, 111
0x38 (0x58)	QTCSR	QTouch Control and Status Register								7
0x37 (0x57)	SPMCSR	–	–	RSIG	CTPB	RFLB	PGWRT	PGERS	SPMEN	194
0x36 (0x56)	MCUCR	–	SM1	SM0	SE	–	–	ISC01	ISC00	37, 50
0x35 (0x55)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	45
0x34 (0x54)	PRR	–	PRTWI	PRTIM0	PRTIM0	PRUSI	PRUSART1	PRUSART0	PRADC	37
0x33 (0x53)	CLKPR	–	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	31
0x32 (0x52)	CLKSR	OSCRDY	CSTR	CKOUT_IO	SUT	CKSEL3	CKSEL2	CKSEL1	CKSEL0	29
0x31 (0x51)	Reserved	–	–	–	–	–	–	–	–	
0x30 (0x50)	WDTCSR	WDIF	WDIE	WDP3	–	WDE	WDP2	WDP1	WDP0	45
0x2F (0x4F)	CCP	CPU Change Protection Register								14
0x2E (0x4E)	DWDR	DWDR[7:0]								189
0x2D (0x4D)	USIBR	USI Buffer Register								136
0x2C (0x4C)	USIDR	USI Data Register								136
0x2B (0x4B)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	135
0x2A (0x4A)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	133
0x29 (0x49)	PCMSK2	–	–	PCINT17	PCINT16	PCINT15	PCINT14	PCINT13	PCINT12	51
0x28 (0x48)	PCMSK1	–	–	–	–	PCINT11	PCINT10	PCINT9	PCINT8	52
0x27 (0x47)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	52
0x26 (0x46)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM	157
0x25 (0x45)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	158
0x24 (0x44)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	159
0x23 (0x43)	UCSR0D	RXCIE0	RXS0	SFDE0	–	–	–	–	–	160
0x22 (0x42)	UBRR0H	–	–	–	–	USART0 Baud Rate Register High Byte				161

Notes: 1. For compatibility with future devices, reserved bits should be written to “0” 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 logic one to them. Note that, unlike most other AVR®s, the CBI and SBI instructions only operate 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.

27. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page(s)
0x21 (0x41)	UBRR0L	USART0 Baud Rate Register Low Byte								161
0x20 (0x40)	UDR0	USART0 I/O Data Register								156
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	–	
0x1E (0x3E)	EEARL	EEAR[7:0]								22
0x1D (0x3D)	EEDR	EEPROM Data Register								23
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	23
0x1B (0x3B)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	82
0x1A (0x3A)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	84
0x19 (0x39)	TCNT0	Timer/Counter0								85
0x18 (0x38)	OCR0A	Timer/Counter0 – Compare Register A								85
0x17 (0x37)	OCR0B	Timer/Counter0 – Compare Register B								85
0x16 (0x36)	GPIOR2	General Purpose Register 2								24
0x15 (0x35)	GPIOR1	General Purpose Register 1								24
0x14 (0x34)	GPIOR0	General Purpose Register 0								24
0x13 (0x33)	PORTCR	–	–	–	–	–	BBMC	BBMB	BBMA	69
0x12 (0x32)	PUEA	PUEA7	PUEA6	PUEA5	PUEA4	PUEA3	PUEA2	PUEA1	PUEA0	69
0x11 (0x31)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	69
0x10 (0x30)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	69
0x0F (0x2F)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	70
0x0E (0x2E)	PUEB	–	–	–	–	PUEB3	PUEB2	PUEB1	PUEB0	70
0x0D (0x2D)	PORTB	–	–	–	–	PORTB3	PORTB2	PORTB1	PORTB0	70
0x0C (0x2C)	DDRB	–	–	–	–	DDB3	DDB2	DDB1	DDB0	70
0x0B (0x2B)	PINB	–	–	–	–	PINB3	PINB2	PINB1	PINB0	70
0x0A (0x2A)	PUEC	–	–	PUEC5	PUEC4	PUEC3	PUEC2	PUEC1	PUEC0	70
0x09 (0x29)	PORTC	–	–	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	70
0x08 (0x28)	DDRC	–	–	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	71
0x07 (0x27)	PINC	–	–	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	71
0x06 (0x26)	ACSRA	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	170
0x05 (0x25)	ACSRB	HSEL	HLEV	ACLP	–	ACCE	ACME	ACIRS1	ACIRS0	171
0x04 (0x24)	ADMUX	REFS1	REFS0	REFEN	ADC0EN	MUX3	MUX2	MUX1	MUX0	182
0x03 (0x23)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	184
0x02 (0x22)	ADCSRB	VDEN	VDPD	–	–	ADLAR	ADTS2	ADTS1	ADTS0	185
0x01 (0x21)	ADCH	ADC Data Register High Byte								185
0x00 (0x20)	ADCL	ADC Data Register Low Byte								185

Notes:

1. For compatibility with future devices, reserved bits should be written to "0" 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 logic one to them. Note that, unlike most other AVR®s, the CBI and SBI instructions only operate 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.

28. 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 register	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					
JMP	k	Direct jump	PC ← k	None	3
RJMP	k	Relative jump	PC ← PC + k + 1	None	2
IJMP		Indirect jump to (Z)	PC ← Z	None	2
CALL	k	Direct subroutine	PC ← k	None	4
RCALL	k	Relative subroutine call	PC ← PC + k + 1	None	3
ICALL		Indirect call to (Z)	PC ← Z	None	3
RET		Subroutine return	PC ← STACK	None	4
RETI		Interrupt return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, skip if equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare register with immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if bit in register cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if bit in register is set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if bit in I/O register cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if bit in I/O register is set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if status flag set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if status flag cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if (C = 0) then PC ← PC + k + 1	None	1/2

28. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRSH	k	Branch if same or higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if greater or equal, signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if less than zero, signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag is cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
Bit and Bit Test Instructions					
SBI	P,b	Set bit in I/O register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear bit in I/O register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical shift left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical shift right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate left through carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate right through carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic shift right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit store from register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to register	Rd(b) ← T	None	1
SEC		Set carry	C ← 1	C	1
CLC		Clear carry	C ← 0	C	1
SEN		Set negative flag	N ← 1	N	1
CLN		Clear negative flag	N ← 0	N	1
SEZ		Set zero flag	Z ← 1	Z	1
CLZ		Clear zero flag	Z ← 0	Z	1
SEI		Global interrupt enable	I ← 1	I	1
CLI		Global interrupt disable	I ← 0	I	1
SES		Set signed test flag	S ← 1	S	1
CLS		Clear signed test flag	S ← 0	S	1
SEV		Set two’s complement overflow.	V ← 1	V	1
CLV		Clear two’s complement overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set half carry flag in SREG	H ← 1	H	1
CLH		Clear half carry flag in SREG	H ← 0	H	1

28. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
Data Transfer Instructions					
MOV	Rd, Rr	Move between registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy register word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load immediate	Rd ← K	None	1
LD	Rd, X	Load indirect	Rd ← (X)	None	2
LD	Rd, X+	Load indirect and post-inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load indirect and pre-dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load indirect and post-inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load indirect and pre-dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load indirect with displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load indirect and post-inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load indirect and pre-dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load indirect with displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store indirect	(X) ← Rr	None	2
ST	X+, Rr	Store indirect and post-inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store indirect and pre-dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store indirect and post-inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store indirect and pre-dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store indirect with displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store indirect and post-inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store indirect and pre-dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store indirect with displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store direct to SRAM	(k) ← Rr	None	2
LPM		Load program memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load program memory and post-inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store program memory	(z) ← R1:R0	None	
IN	Rd, P	IN port	Rd ← P	None	1
OUT	P, Rr	OUT port	P ← Rr	None	1
PUSH	Rr	Push register on stack	STACK ← Rr	None	2
POP	Rd	Pop register from stack	Rd ← STACK	None	2
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep	(see specific 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

29. Ordering Information

Table 29-1. Ordering Information Atmel ATtiny1634

Speed (MHz)[1]	Supply Voltage (V)	Temperature Range	Package[2]	Ordering Code
12	2.7 to 5.5	Automotive (–40°C to +125°C)	6G	TINY1634-15XZ
			PC	TINY1634-15MZ

Notes: 1. For speed versus supply voltage, see Section 25.3 “Speed” on page 217.

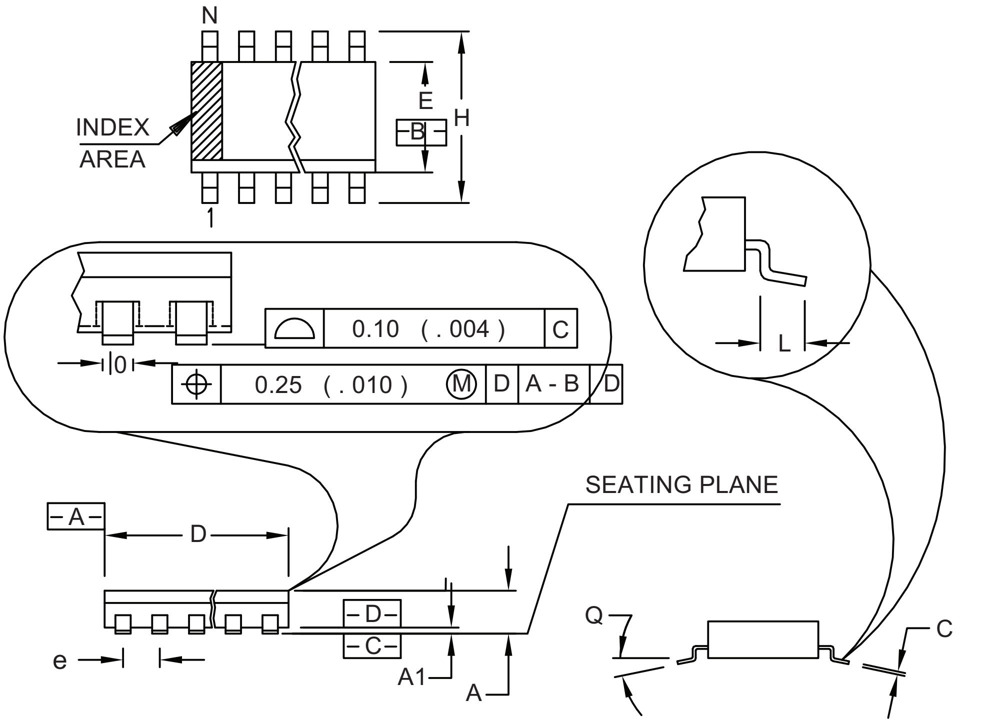
2. All packages are Pb-free, halide-free and fully green, and they comply with the European directive for restriction of hazardous substances (RoHS).

30. Packaging Information

Table 30-1. Atmel ATtiny1634 Package Types

Package Types	
6G	6G - 20 lead - 4.4mmx6.5mm body - 0.65mm pitch - lead length:0.6mm - thin shrink small outline
PC	PC - 20 lead - 4.0mmx4.0mm body - 0.50mm pitch - quad flat no-lead package

Figure 30-1. 6G

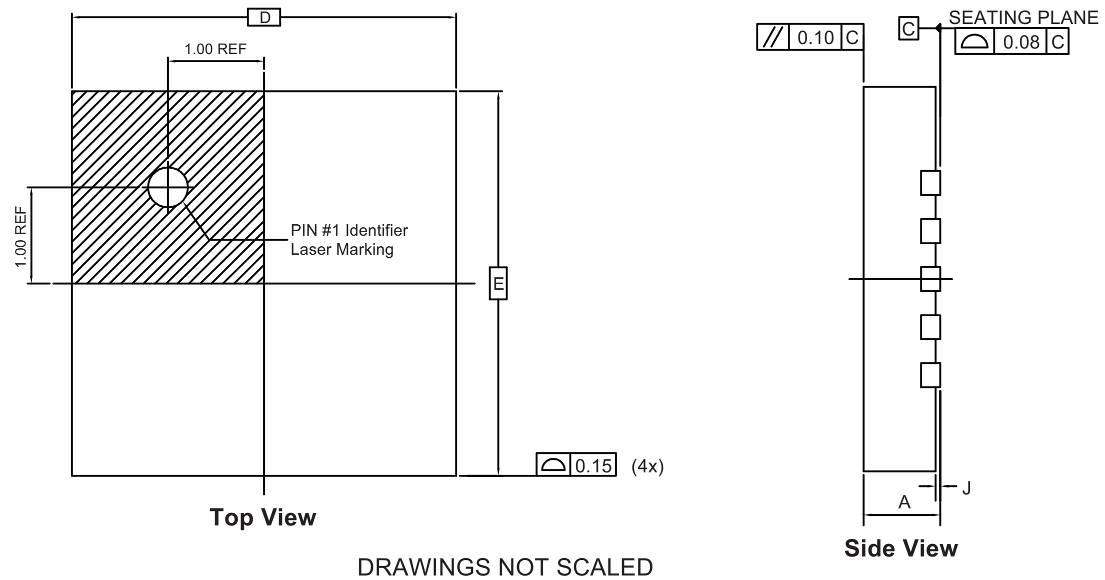


	MM		INCH	
A	----	1.10	----	.043
A1	0.05	0.15	.002	.006
b	0.19	0.30	.007	.012
C	0.09	0.20	.003	.008
D	6.40	6.60	.252	.260
E	4.30	4.50	.169	.177
e	0.65	BSC	.026	BSC
H	6.40 BSC		.252 BSC	
L	0.50	0.70	.020	.028
N	20		20	
Q	0° ~8°		0° ~8°	

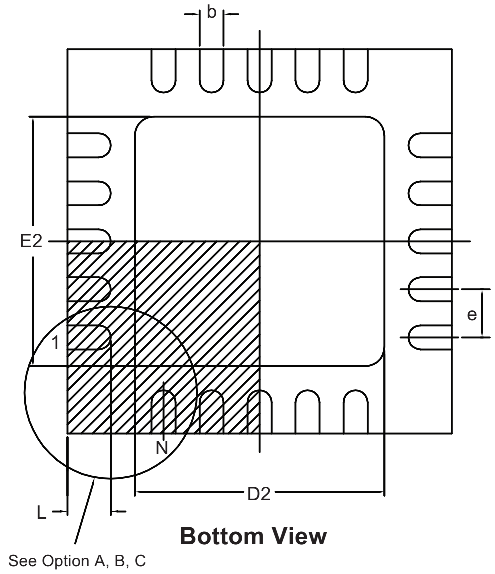
20/12/07

Atmel **Package Drawing Contact:** packagedrawings@atmel.com	TITLE 6G, 20 Leads - 4.4x6.5mm Body - 0.65mm Pitch - Lead length: 0.6mm THIN SHRINK SMALL OUTLINE	GPC	DRAWING NO. 6G	REV. A

Figure 30-2. PC

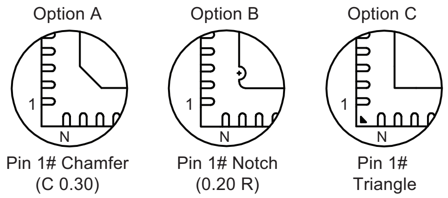


DRAWINGS NOT SCALED



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

	TITLE	GPC	DRAWING NO.	REV.
Atmel **Package Drawing Contact:** packagedrawings@atmel.com	PC, 20-Lead - 4.0x4.0mm Body, 0.50mm Pitch Quad Flat No Lead Package (QFN)		PC	I

31. Errata

The revision letters in this section refer to the revision of the corresponding Atmel® ATtiny1634 device.

31.1 Atmel ATtiny1634

31.1.1 Rev. B

- Port pin should not be used as input when ULP oscillator is disabled.

1. Port pin should not be used as input when ULP oscillator is disabled.

   Port pin PB3 is not guaranteed to perform as a reliable input when the ultra-low-power (ULP) oscillator is not running. In addition, the pin is pulled down internally when the ULP oscillator is disabled.

   Problem Fix/Workaround

   The ULP oscillator is automatically activated when required. To use PB3 as an input, activate the watchdog timer. The watchdog timer automatically enables the ULP oscillator.

32. 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
9296C-AVR-07/14	• Section 20-1 “Features on page 172 updated • Section 25.2 “DC Characteristics” on pages 215 to 216 updated • Table 25-2 “Calibration Accuracy of Internal 8MHz Oscillator” on page 217 updated • Table 25-3 “Calibration Accuracy of Internal 32kHz Oscillator” on page 217 updated • Table 25-5 “Reset, Brown-out, and Internal Voltage Characteristics” on page 218 updated • Section 25.7 “Analog-To-Digital Converter” on page 220 updated • Section 25.8 “Analog Comparator” on page 221 updated • Put datasheet in the latest template
9296B-AVR-12/13	• Section 25.2 “DC Characteristics” on page 215 updated

33. Table of Contents

Features 1

1. Pin Configurations 3
   1.1 Pin Descriptions 4

2. Overview 5

3. Automotive Quality Grade 7

4. General Information 7
   4.1 Resources 7
   4.2 Code Examples 7
   4.3 Capacitive Touch Sensing 7
   4.4 Data Retention 7

5. CPU Core 8
   5.1 Architectural Overview 8
   5.2 ALU – Arithmetic Logic Unit 9
   5.3 Status Register 9
   5.4 General Purpose Register File 9
   5.5 Stack Pointer 11
   5.6 Instruction Execution Timing 11
   5.7 Reset and Interrupt Handling 12
   5.8 Register Description 14

6. Memories 16
   6.1 Program Memory (Flash) 16
   6.2 Data Memory (SRAM) and Register Files 17
   6.3 Data Memory (EEPROM) 18
   6.4 Register Description 22

7. Clock System 25
   7.1 Clock Subsystems 25
   7.2 Clock Sources 26
   7.3 System Clock Prescaler 28
   7.4 Clock Output Buffer 28
   7.5 Register Description 29

8. Power Management and Sleep Modes 34
   8.1 Sleep Modes 34
   8.2 Power Reduction Register 35
   8.3 Minimizing Power Consumption 35
   8.4 Register Description 37

9. System Control and Reset 39
   9.1 Resetting the AVR 39
   9.2 Reset Sources 39
   9.3 Internal Voltage Reference 43
   9.4 Watchdog Timer 43
   9.5 Register Description 45

10. Interrupts 47
   10.1 Interrupt Vectors 47
   10.2 External Interrupts 48
   10.3 Register Description 50

11. I/O Ports 53
11.1 Overview 53
11.2 Ports as a General Digital I/O 54
11.3 Alternate Port Functions 59
11.4 Register Description 69

12. 8-bit Timer/Counter0 with PWM 72
12.1 Features 72
12.2 Overview 72
12.3 Clock Sources 73
12.4 Counter Unit 73
12.5 Output Compare Unit 74
12.6 Compare Match Output Unit 75
12.7 Operating Modes 76
12.8 Timer/Counter Timing Diagrams 80
12.9 Register Description 82

13. 16-bit Timer/Counter1 87
13.1 Features 87
13.2 Overview 87
13.3 Timer/Counter Clock Sources 89
13.4 Counter Unit 90
13.5 Input Capture Unit 91
13.6 Output Compare Units 92
13.7 Compare Match Output Unit 94
13.8 Operating Modes 95
13.9 Timer/Counter Timing Diagrams 101
13.10 Accessing 16-bit Registers 103
13.11 Register Description 106

14. Timer/Counter Prescaler 112
14.1 Prescaler Reset 112
14.2 External Clock Source 113
14.3 Register Description 113

15. I2C-Compatible, Two-wire Slave Interface 114
15.1 Features 114
15.2 Overview 114
15.3 General TWI Bus Concepts 114
15.4 TWI Slave Operation 120
15.5 Register Description 121

16. USI – Universal Serial Interface 126
16.1 Features 126
16.2 Overview 126
16.3 Three-wire Mode 127
16.4 Two-wire Mode 128
16.5 Alternative Use 131
16.6 Program Examples 131
16.7 Register Descriptions 133

17. USART (USART0 and USART1) 137
17.1 Features 137
17.2 USART0 and USART1 137
17.3 Overview 137
17.4 Clock Generation 139
17.5 Frame Formats 141

17.6 USART Initialization 142
17.7 Data Transmission – the USART Transmitter 143
17.8 Data Reception – the USART Receiver 146
17.9 Asynchronous Data Reception 149
17.10 Multiprocessor Communication Mode 153
17.11 Examples of Baud Rate Setting 154
17.12 Register Description 156

18. USART in SPI Mode 162
18.1 Features 162
18.2 Overview 162
18.3 Clock Generation 162
18.4 SPI Data Modes and Timing 163
18.5 Frame Formats 163
18.6 Data Transfer 165
18.7 Compatibility with AVR SPI 166
18.8 Register Description 166

19. Analog Comparator 169
19.1 Analog Comparator Multiplexed Input 169
19.2 Register Description 170

20. Analog to Digital Converter 172
20.1 Features 172
20.2 Overview 172
20.3 Operation 174
20.4 Starting a Conversion 174
20.5 Prescaling and Conversion Timing 175
20.6 Changing Channel or Reference Selection 178
20.7 ADC Noise Canceler 178
20.8 Analog Input Circuitry 179
20.9 Noise Canceling Techniques 179
20.10 ADC Accuracy Definitions 180
20.11 ADC Conversion Result 182
20.12 Temperature Measurement 182
20.13 Register Description 182

21. debugWIRE On-chip Debug System 188
21.1 Features 188
21.2 Overview 188
21.3 Physical Interface 188
21.4 Software Break Points 189
21.5 Limitations of debugWIRE 189
21.6 Register Description 189

22. Self Programming 190
22.1 Features 190
22.2 Overview 190
22.3 Lock Bits 190
22.4 Self Programming the Flash 190
22.5 Preventing Flash Corruption 193
22.6 Programming Time for Flash when Using SPM 194
22.7 Register Description 194

23. Lock Bits, Fuse Bits and Device Signature 196
23.1 Lock Bits 196
23.2 Fuse Bits 197
23.3 Device Signature Imprint Table 199
23.4 Reading Lock, Fuse, and Signature Data from Software 200

24. External Programming 202
24.1 Memory Parametrics 202
24.2 Parallel Programming 202
24.3 Serial Programming 211
24.4 Programming Time for Flash and EEPROM 214

25. Electrical Characteristics 215
25.1 Absolute Maximum Ratings 215
25.2 DC Characteristics 215
25.3 Speed 217
25.4 Clock 217
25.5 System and Reset 218
25.6 Two-wire Serial Interface 219
25.7 Analog-To-Digital Converter 220
25.8 Analog Comparator 221
25.9 Temperature Sensor 221
25.10 Parallel Programming 221
25.11 Serial Programming 223

26. Typical Characteristics 224
26.1 Current Consumption in Active Mode 224
26.2 Current Consumption in Idle Mode 227
26.3 Current Consumption in Power-Down Mode 229
26.4 Current Consumption of Peripheral Units 230
26.5 Pull-Up Resistors 232
26.6 Input Thresholds 232
26.7 Output Driver Strength 234
26.8 BOD 238
26.9 Band-gap Voltage 240
26.10 Reset 241
26.11 Internal Oscillator Speed 242

27. Register Summary 245

28. Instruction Set Summary 248

29. Ordering Information 251

30. Packaging Information 251

31. Errata 254
31.1 Atmel ATtiny1634 254

32. Revision History 254

33. Table of Contents 255

Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.