Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 8K Bytes of In-System Self-programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 512 Bytes Internal SRAM
  - Up to 64K Bytes Optional External Memory Space
  - Programming Lock for Software Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Three PWM Channels
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Three Sleep Modes: Idle, Power-down and Standby
- I/O and Packages
  - 35 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega8515L
  - 4.5 - 5.5V for ATmega8515
- Speed Grades
  - 0 - 8 MHz for ATmega8515L
  - 0 - 16 MHz for ATmega8515

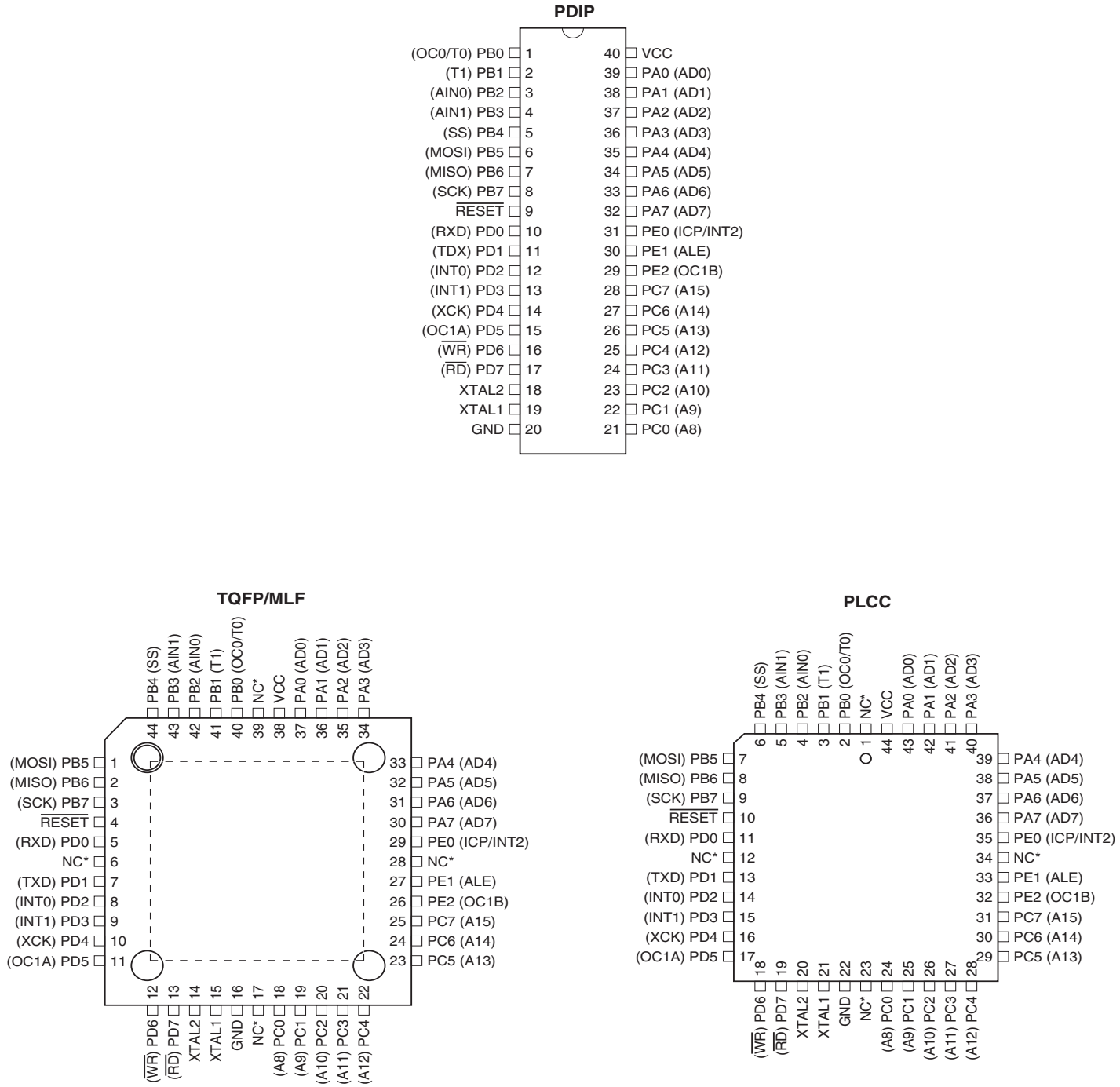
8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

ATmega8515
ATmega8515L

2512K–AVR–01/10

Pin Configurations

Figure 1. Pinout ATmega8515



NOTES:

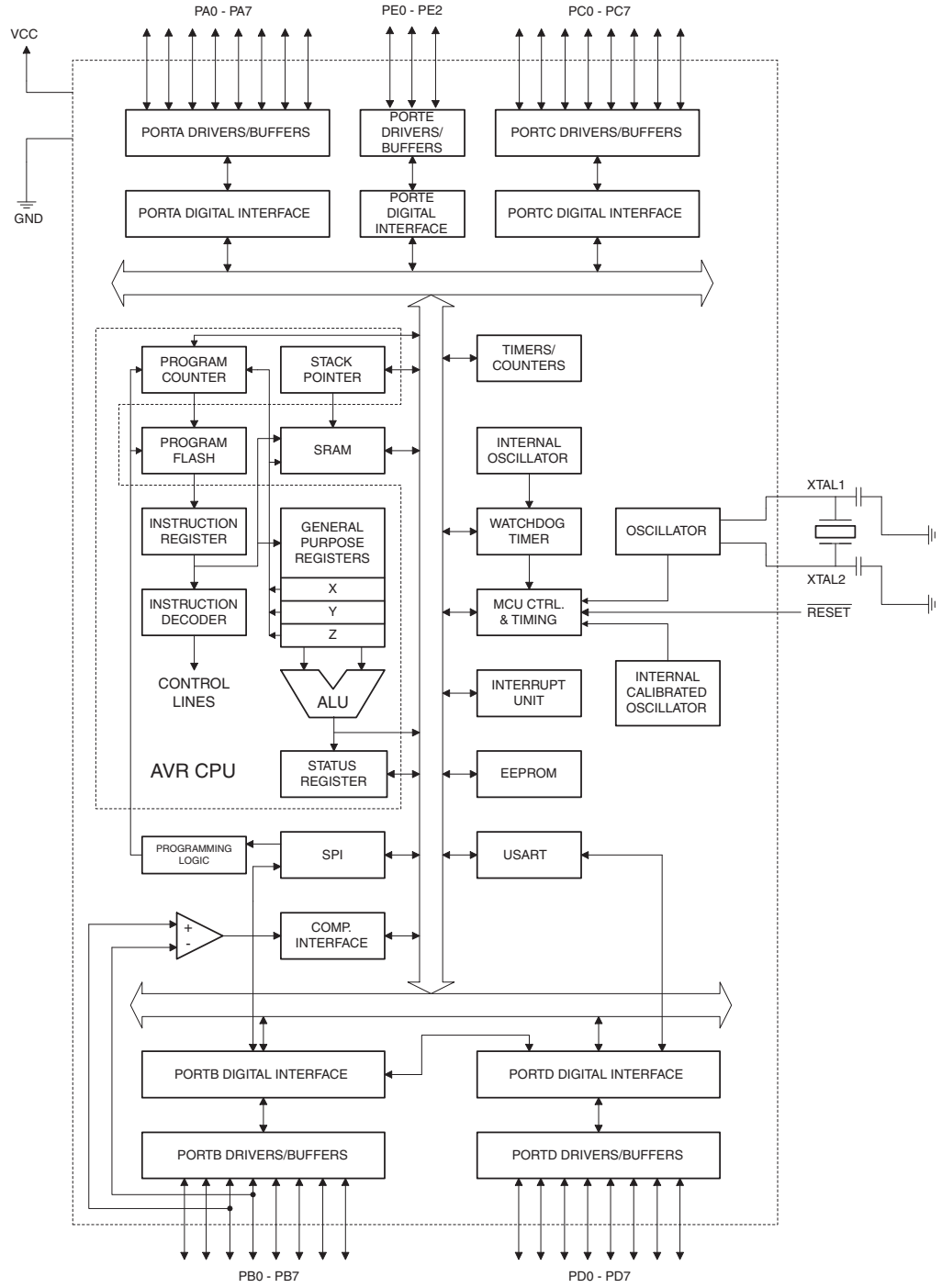
1. MLF bottom pad should be soldered to ground.
2. * NC = Do not connect (May be used in future devices)

Overview

The ATmega8515 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 ATmega8515 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



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

The ATmega8515 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 512 bytes SRAM, an External memory interface, 35 general purpose I/O lines, 32 general purpose working registers, two flexible Timer/Counters with compare modes, Internal and External interrupts, a Serial Programmable USART, a programmable Watchdog Timer with internal Oscillator, a SPI serial port, and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and Interrupt system to continue functioning. The Power-down mode saves the Register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the Program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-programmable Flash on a monolithic chip, the Atmel ATmega8515 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega8515 is supported with a full suite of program and system development tools including: C Compilers, Macro assemblers, Program debugger/simulators, In-circuit Emulators, and Evaluation kits.

Disclaimer

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

AT90S4414/8515 and ATmega8515 Compatibility

The ATmega8515 provides all the features of the AT90S4414/8515. In addition, several new features are added. The ATmega8515 is backward compatible with AT90S4414/8515 in most cases. However, some incompatibilities between the two microcontrollers exist. To solve this problem, an AT90S4414/8515 compatibility mode can be selected by programming the S8515C Fuse. ATmega8515 is 100% pin compatible with AT90S4414/8515, and can replace the AT90S4414/8515 on current printed circuit boards. However, the location of Fuse bits and the electrical characteristics differs between the two devices.

AT90S4414/8515 Compatibility Mode

Programming the S8515C Fuse will change the following functionality:

- The timed sequence for changing the Watchdog Time-out period is disabled. See "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 53 for details.
- The double buffering of the USART Receive Registers is disabled. See "AVR USART vs. AVR UART – Compatibility" on page 137 for details.
- PORTE(2:1) will be set as output, and PORTE0 will be set as input.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega8515 as listed on page 67.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega8515 as listed on page 67.

Port C (PC7..PC0) Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D (PD7..PD0) Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega8515 as listed on page 72.

Port E(PE2..PE0) Port E is an 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega8515 as listed on page 74.

$\overline{\text{RESET}}$ Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 18 on page 46. Shorter pulses are not guaranteed to generate a reset.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting Oscillator amplifier.

Resources

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

About Code Examples

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

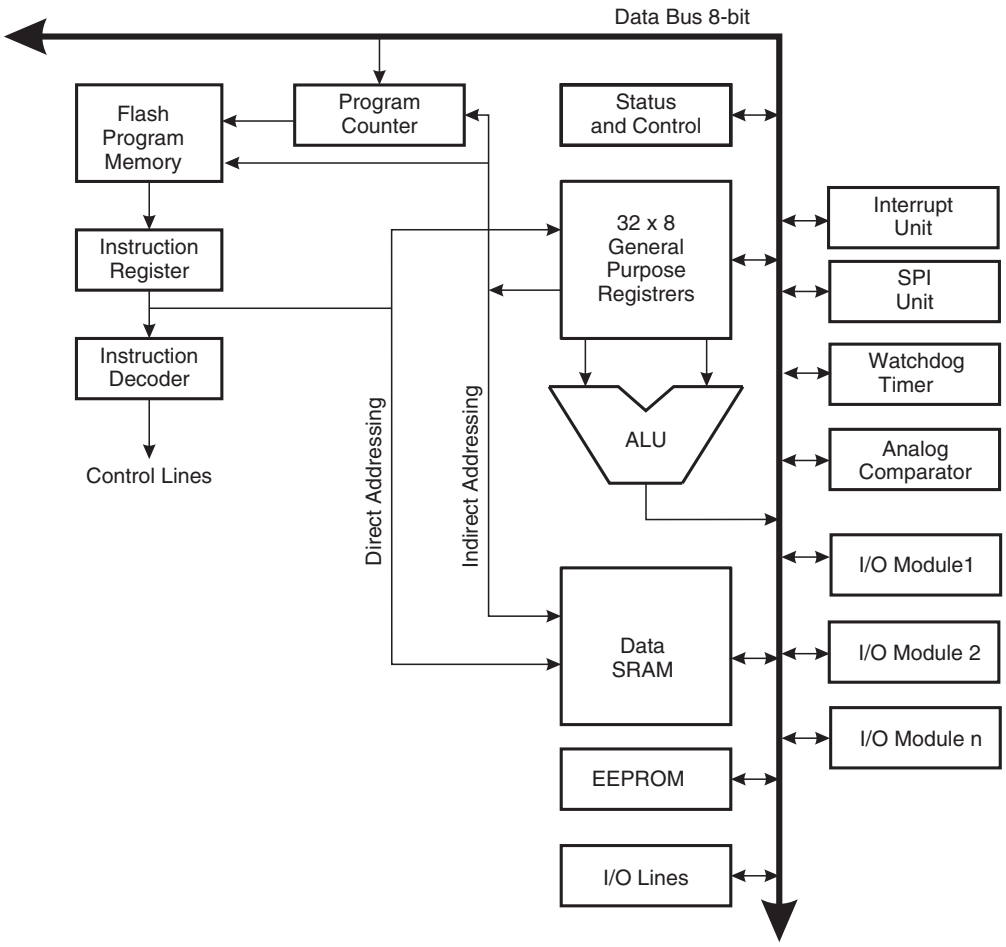
AVR CPU Core

Introduction

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

Architectural Overview

Figure 3. Block Diagram of the AVR Architecture



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

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

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

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every Program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its Control Registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate interrupt vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, $20 - $5F.

ALU – Arithmetic Logic Unit

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

Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

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

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

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate Control Registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

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

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

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

The S-bit is always an exclusive or between the Negative Flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

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

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

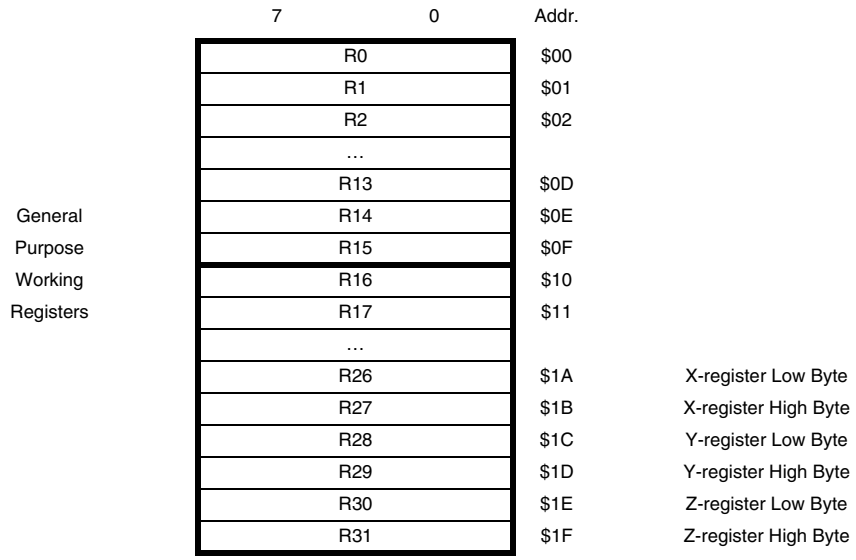
General Purpose Register File

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

Figure 4. AVR CPU General Purpose Working Registers



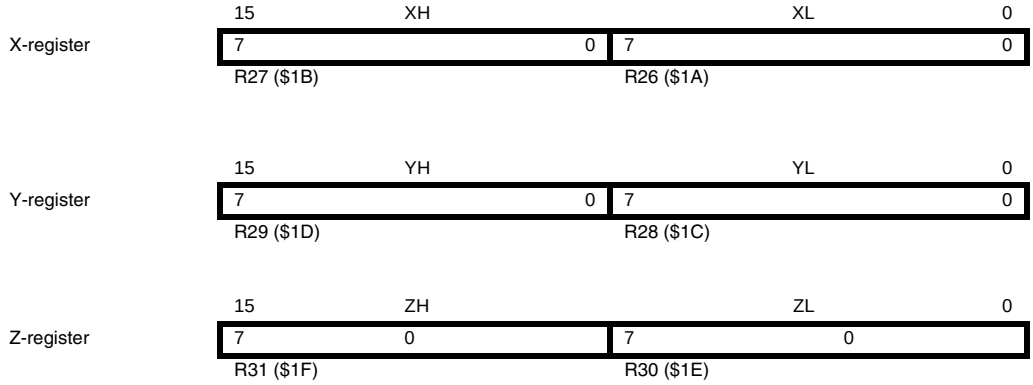
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

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

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

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



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

Stack Pointer

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

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when address is popped from the Stack with return from subroutine RET or return from interrupt RETI.

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

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

Instruction Execution Timing

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

Figure 6 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

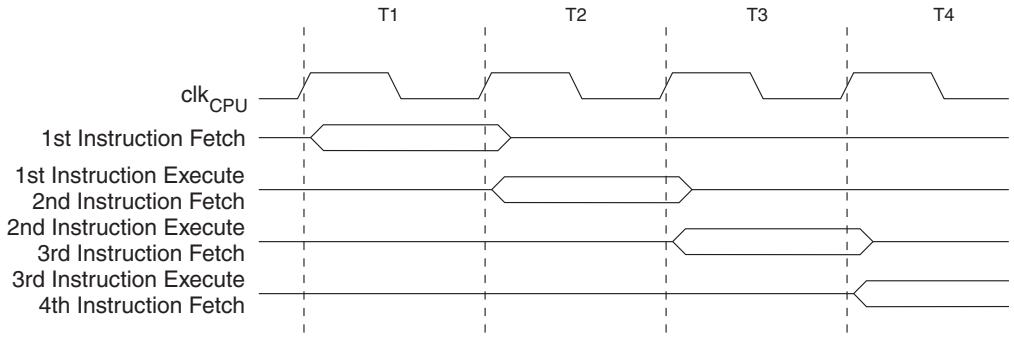
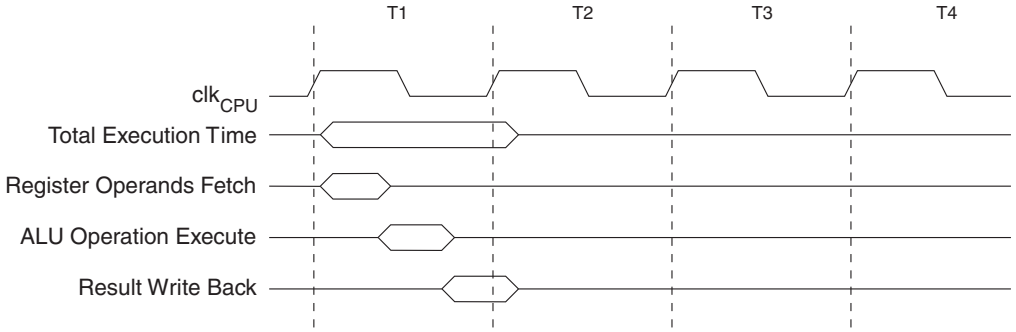


Figure 7 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



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. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section “Memory Programming” on page 179 for details.

The lowest addresses in the Program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 54. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to “Interrupts” on page 54 for more information. The Reset Vector can

also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see “Boot Loader Support – Read-While-Write Self-Programming” on page 166.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

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

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

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

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

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence..

Assembly Code Example

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

C Code Example

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

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

Assembly Code Example

```
sei  ; set global interrupt enable
sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

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

Interrupt Response Time

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

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

AVR ATmega8515 Memories

This section describes the different memories in the ATmega8515. The AVR architecture has two main memory spaces, the Data Memory and the Program memory space. In addition, the ATmega8515 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program memory

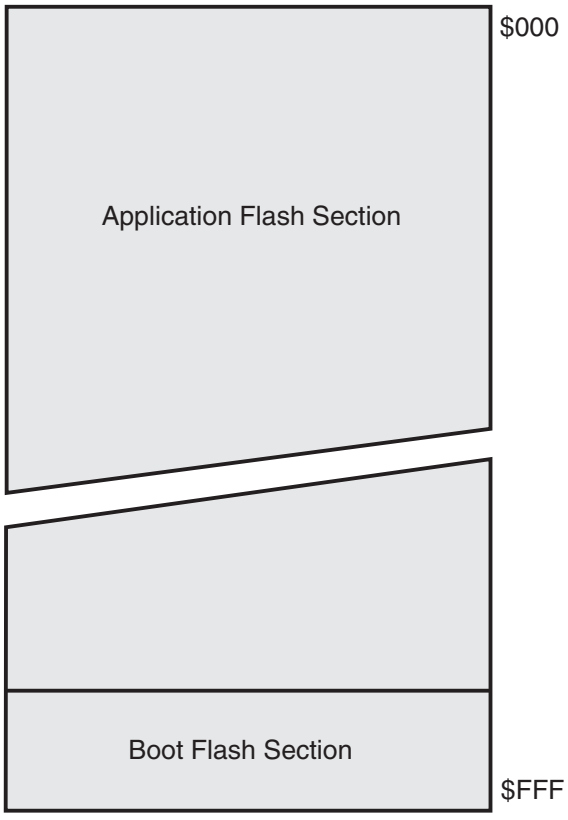
The ATmega8515 contains 8K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8515 Program Counter (PC) is 12 bits wide, thus addressing the 4K Program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in “Boot Loader Support – Read-While-Write Self-Programming” on page 166. “Memory Programming” on page 179 contains a detailed description on Flash data serial downloading using the SPI pins.

Constant tables can be allocated within the entire Program memory address space, see the LPM – Load Program memory instruction description.

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

Figure 8. Program memory Map



SRAM Data Memory

Figure 9 shows how the ATmega8515 SRAM Memory is organized.

The lower 608 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 512 locations address the internal data SRAM.

An optional external data SRAM can be used with the ATmega8515. This SRAM will occupy an area in the remaining address locations in the 64K address space. This area starts at the address following the internal SRAM. The Register File, I/O, Extended I/O and Internal SRAM occupies the lowest 608 bytes in normal mode, so when using 64KB (65536 bytes) of External Memory, 64928 Bytes of External Memory are available. See “External Memory Interface” on page 25 for details on how to take advantage of the external memory map.

When the addresses accessing the SRAM memory space exceeds the internal Data memory locations, the external data SRAM is accessed using the same instructions as for the internal Data memory access. When the internal data memories are accessed, the read and write strobe pins (PD7 and PD6) are inactive during the whole access cycle. External SRAM operation is enabled by setting the SRE bit in the MCUCR Register.

Accessing external SRAM takes one additional clock cycle per byte compared to access of the internal SRAM. This means that the commands LD, ST, LDS, STS, LDD, STD, PUSH, and POP take one additional clock cycle. If the Stack is placed in external SRAM, interrupts, subroutine calls and returns take three clock cycles extra because the two-byte Program Counter is pushed and popped, and external memory access does not take advantage of the internal pipe-line memory access. When external SRAM interface is used with wait-state, one-byte external access takes two, three, or four additional clock cycles for one, two, and three wait-states respectively. Interrupts, subroutine calls and returns will need five, seven, or nine clock cycles more than specified in the instruction set manual for one, two, and three wait-states.

The five different addressing modes for the Data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

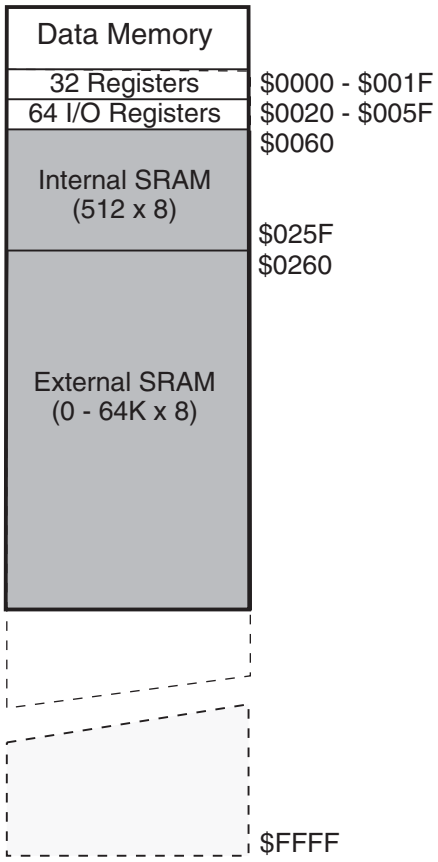
The direct addressing reaches the entire data space.

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

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

The 32 general purpose working registers, 64 I/O Registers, and the 512 bytes of internal data SRAM in the ATmega8515 are all accessible through all these addressing modes. The Register File is described in “General Purpose Register File” on page 11.

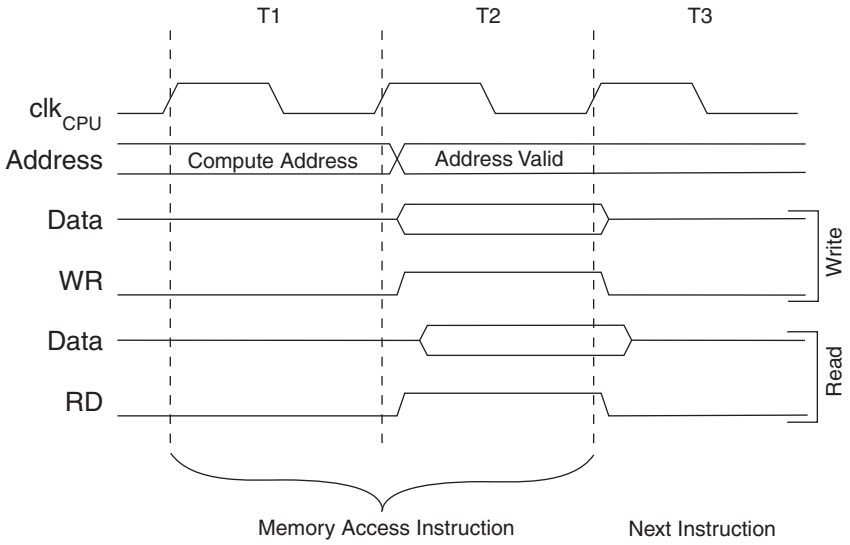
Figure 9. Data Memory Map



Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega8515 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

"Memory Programming" on page 179 contains a detailed description on EEPROM Programming in SPI or Parallel Programming mode.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See "Preventing EEPROM Corruption" on page 24. for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

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

The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	**EEAR8**	**EEARH**
	EEAR7	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	**EEARL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8515 and will always read as zero.

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

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

The EEPROM Data Register – EEDR

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

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

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

The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8515 and will always read as zero.

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

Writing EERIE to one enables the EEPROM Ready Interrupt if the I-bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

- **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEWE becomes zero.
2. Wait until SPMEN in SPMCR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See “Boot Loader Support – Read-While-Write Self-Programming” on page 166 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

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

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Table 1. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles[1]	Typ Programming Time
EEPROM Write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse settings.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r16) to data register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address and data registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

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

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from data register
  in   r16,EEDR
  ret
```

C Code Example

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

EEPROM Write During Power-down Sleep Mode

When entering Power-down Sleep mode while an EEPROM write operation is active, the EEPROM write operation will continue, and will complete before the Write Access time has passed. However, when the write operation is completed, the crystal Oscillator continues running, and as a consequence, the device does not enter Power-down entirely. It is therefore recommended to verify that the EEPROM write operation is completed before entering Power-down.

Preventing EEPROM Corruption

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

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

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

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

I/O Memory

The I/O space definition of the ATmega8515 is shown in “Register Summary” on page 239.

All ATmega8515 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O Registers as data space using LD and ST instructions, $20 must be added to these addresses.

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

Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

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

External Memory Interface

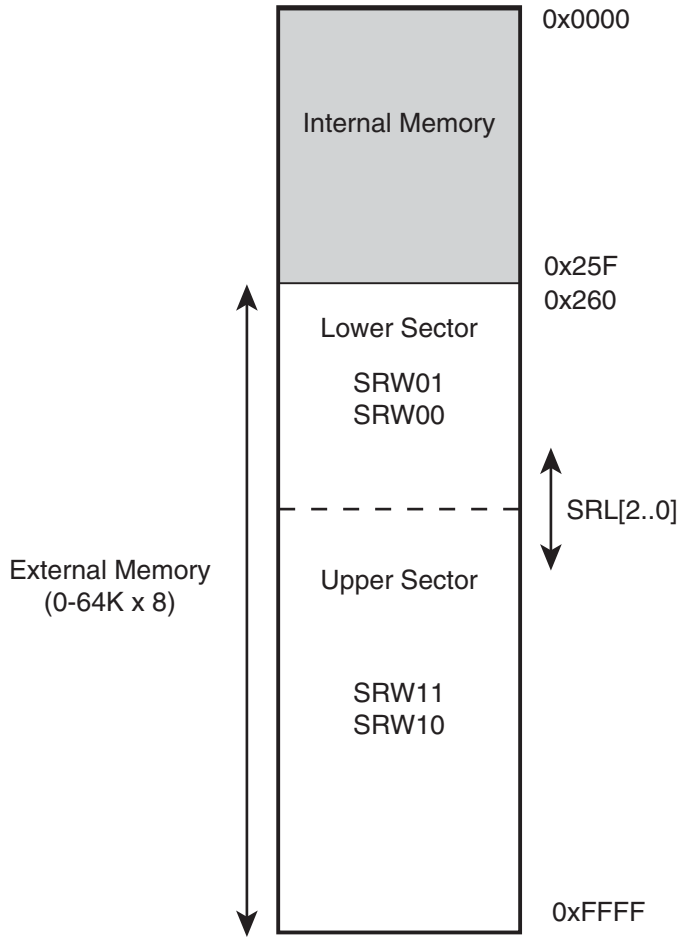
With all the features the External Memory Interface provides, it is well suited to operate as an interface to memory devices such as external SRAM and Flash, and peripherals such as LCD-display, A/D, and D/A. The main features are:

- **Four Different Wait State Settings (Including No wait State)**
- **Independent Wait State Setting for Different External Memory Sectors (Configurable Sector Size)**
- **The Number of Bits Dedicated to Address High Byte is Selectable**
- **Bus Keepers on Data Lines to Minimize Current Consumption (Optional)**

Overview

When the eXternal MEMory (XMEM) is enabled, address space outside the internal SRAM becomes available using the dedicated external memory pins (see Figure 1 on page 2, Table 26 on page 66, Table 32 on page 70, and Table 38 on page 74). The memory configuration is shown in Figure 11.

Figure 11. External Memory with Sector Select



Using the External Memory Interface

The interface consists of:

- AD7:0: Multiplexed low-order address bus and data bus
- A15:8: High-order address bus (configurable number of bits)
- ALE: Address latch enable
- $\overline{RD}$: Read strobe
- $\overline{WR}$: Write strobe

The control bits for the External Memory Interface are located in three registers, the MCU Control Register – MCUCR, the Extended MCU Control Register – EMCUCR, and the Special Function IO Register – SFIOR.

When the XMEM interface is enabled, it will override the settings in the data direction registers corresponding to the ports dedicated to the interface. For details about this port override, see the alternate functions in section “I/O Ports” on page 59. The XMEM interface will auto-detect whether an access is internal or external. If the access is external, the XMEM interface will output address, data, and the control signals on the ports according to Figure 13 (this figure shows the wave forms without wait states). When ALE goes from high to low, there is a valid address on AD7:0. ALE is low during a data transfer. When the XMEM interface is enabled, also an internal access will cause activity on address-, data-, and ALE ports, but the $\overline{RD}$ and $\overline{WR}$ strobes will not toggle during internal access. When the External Memory Interface is disabled, the normal pin and data direction settings are used. Note that when the XMEM interface is disabled, the address space above the internal SRAM boundary is not mapped into the internal SRAM. Figure 12 illustrates how to connect an external SRAM to the AVR using an octal latch (typically “74x573” or equivalent) which is transparent when G is high.

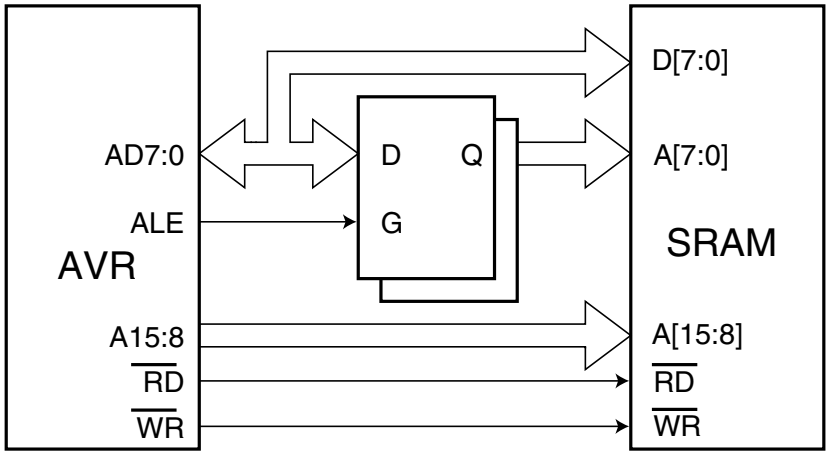
Address Latch Requirements

Due to the high-speed operation of the XRAM interface, the address latch must be selected with care for system frequencies above 8 MHz @ 4V and 4 MHz @ 2.7V. When operating at conditions above these frequencies, the typical old style 74HC series latch becomes inadequate. The external memory interface is designed in compliance to the 74AHC series latch. However, most latches can be used as long they comply with the main timing parameters. The main parameters for the address latch are:

- D to Q propagation delay (t_{pd})
- Data setup time before G low (t_{su})
- Data (address) hold time after G low ($_{th}$)

The external memory interface is designed to guaranty minimum address hold time after G is asserted low of t_h = 5 ns (refer to t_{LAXX_LD}/t_{LLAXX_ST} in Table 98 to Table 105 on page 204). The D to Q propagation delay (t_{pd}) must be taken into consideration when calculating the access time requirement of the external component. The data setup time before G low (t_{su}) must not exceed address valid to ALE low (t_{AVLLC}) minus PCB wiring delay (dependent on the capacitive load).

Figure 12. External SRAM Connected to the AVR



Pull-up and Bus Keeper

The pull-up resistors on the AD7:0 ports may be activated if the corresponding Port Register is written to one. To reduce power consumption in sleep mode, it is recommended to disable the pull-ups by writing the Port Register to zero before entering sleep.

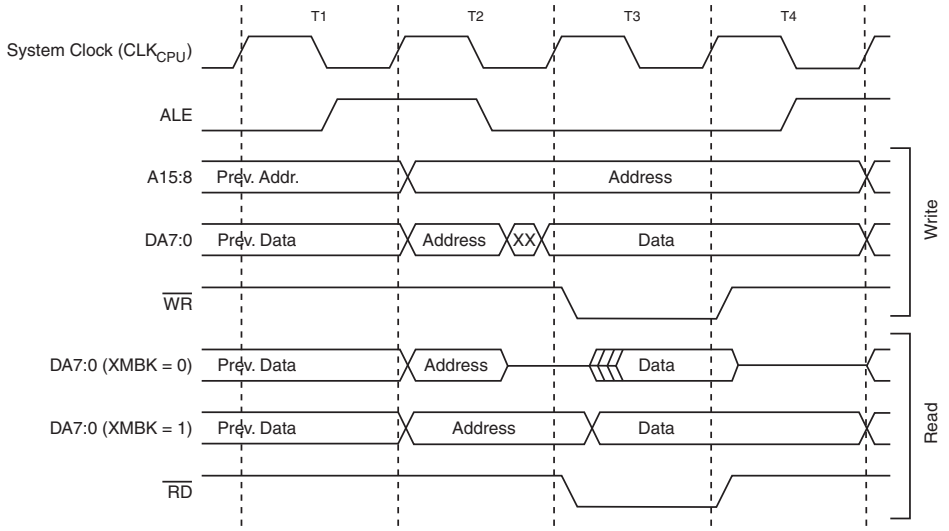
The XMEM interface also provides a bus keeper on the AD7:0 lines. The bus keeper can be disabled and enabled in software as described in "Special Function IO Register – SFIOR" on page 31. When enabled, the bus keeper will keep the previous value on the AD7:0 bus while these lines are tri-stated by the XMEM interface.

Timing

External memory devices have various timing requirements. To meet these requirements, the ATmega8515 XMEM interface provides four different wait states as shown in Table 3. It is important to consider the timing specification of the external memory device before selecting the wait state. The most important parameters are the access time for the external memory in conjunction with the set-up requirement of the ATmega8515. The access time for the external memory is defined to be the time from receiving the chip select/address until the data of this address actually is driven on the bus. The access time cannot exceed the time from the ALE pulse is asserted low until data must be stable during a read sequence (t_{LLRL}+ t_{RLRH} - t_{DVRH} in Table 98 to Table 105 on page 204). The different wait states are set up in software. As an additional feature, it is possible to divide the external memory space in two sectors with individual wait state settings. This makes it possible to connect two different memory devices with different timing requirements to the same XMEM interface. For XMEM interface timing details, please refer to Figure 89 to Figure 92, and Table 98 to Table 105.

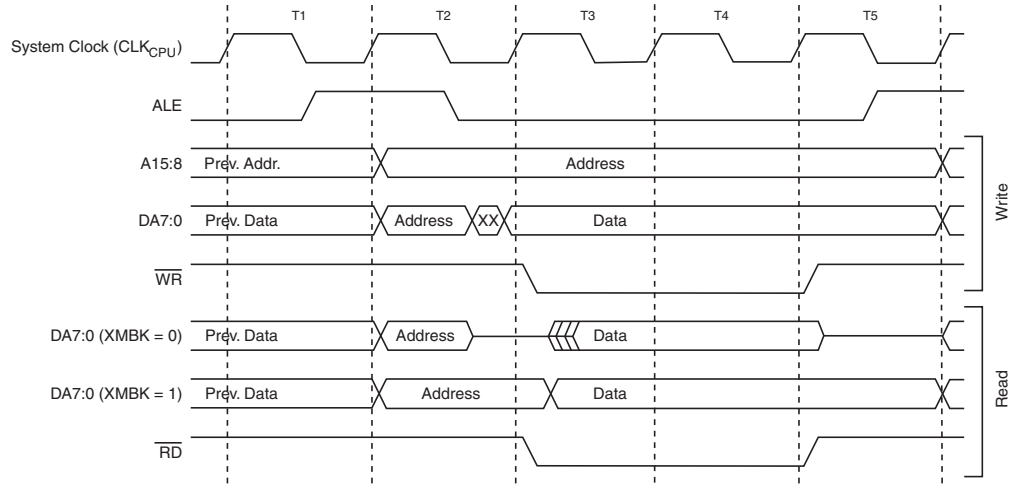
Note that the XMEM interface is asynchronous and that the waveforms in the figures below are related to the internal system clock. The skew between the Internal and External clock (XTAL1) is not guaranteed (it varies between devices, temperature, and supply voltage). Consequently, the XMEM interface is not suited for synchronous operation.

Figure 13. External Data Memory Cycles without Wait State (SRWn1 = 0 and SRWn0 = 0)[1]



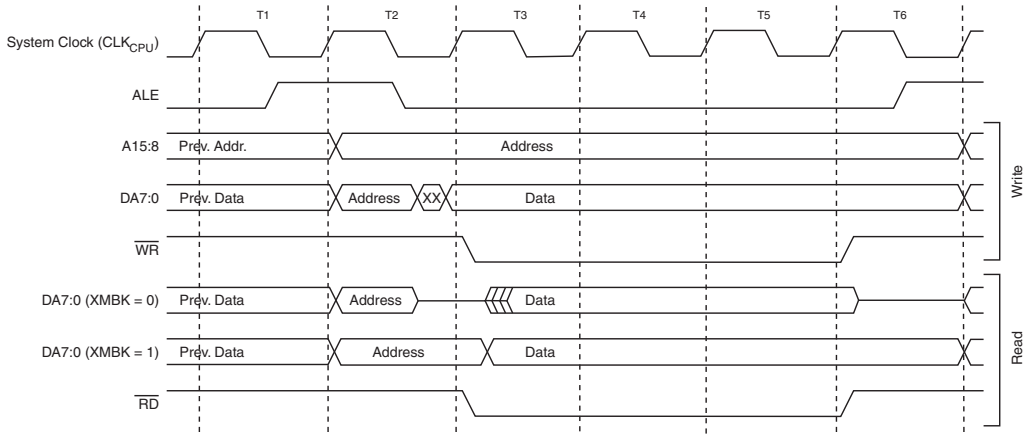
Note: 1. SRWn1 = SRW11 (upper sector) or SRW01 (lower sector), SRWn0 = SRW10 (upper sector) or SRW00 (lower sector)
The ALE pulse in period T4 is only present if the next instruction accesses the RAM (internal or external).

Figure 14. External Data Memory Cycles with SRWn1 = 0 and SRWn0 = 1[1]



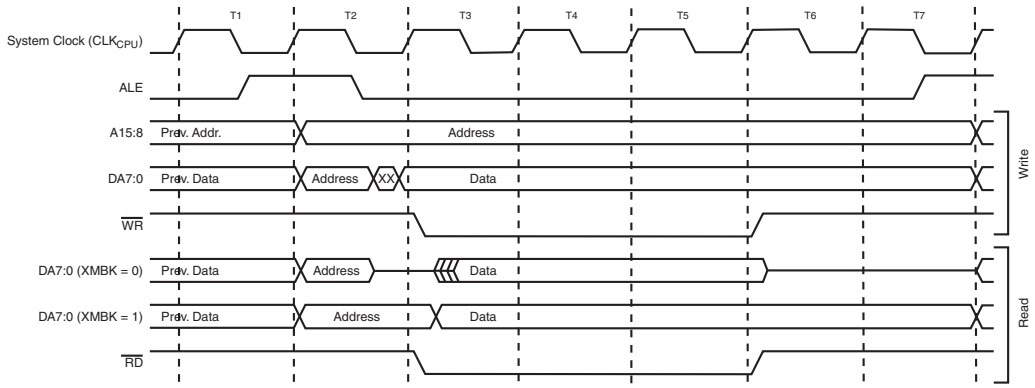
Note: 1. SRWn1 = SRW11 (upper sector) or SRW01 (lower sector), SRWn0 = SRW10 (upper sector) or SRW00 (lower sector)
The ALE pulse in period T5 is only present if the next instruction accesses the RAM (internal or external).

Figure 15. External Data Memory Cycles with SRWn1 = 1 and SRWn0 = 0[1]



Note: 1. SRWn1 = SRW11 (upper sector) or SRW01 (lower sector), SRWn0 = SRW10 (upper sector) or SRW00 (lower sector)
The ALE pulse in period T6 is only present if the next instruction accesses the RAM (internal or external).

Figure 16. External Data Memory Cycles with SRWn1 = 1 and SRWn0 = 1 (1)



Note: 1. SRWn1 = SRW11 (upper sector) or SRW01 (lower sector), SRWn0 = SRW10 (upper sector) or SRW00 (lower sector)
The ALE pulse in period T7 is only present if the next instruction accesses the RAM (internal or external).

XMEM Register Description

MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SRE: External SRAM/XMEM Enable**

Writing SRE to one enables the External Memory Interface.The pin functions AD7:0, A15:8, ALE, $\overline{WR}$, and $\overline{RD}$ are activated as the alternate pin functions. The SRE bit overrides any pin direction settings in the respective Data Direction Registers. Writing SRE to zero, disables the External Memory Interface and the normal pin and data direction settings are used.

- **Bit 6 – SRW10: Wait State Select Bit**

For a detailed description, see common description for the SRWn bits below (EMCUCR description).

Extended MCU Control Register – EMCUCR

Bit	7	6	5	4	3	2	1	0	
	SM0	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	ISC2	EMCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6..4 – SRL2, SRL1, SRL0: Wait State Sector Limit**

It is possible to configure different wait states for different external memory addresses. The External Memory address space can be divided in two sectors that have separate wait state bits. The SRL2, SRL1, and SRL0 bits select the splitting of these sectors, see Table 2 and Figure 11. By default, the SRL2, SRL1, and SRL0 bits are set to zero and the entire External Memory address space is treated as one sector. When the entire

SRAM address space is configured as one sector, the wait states are configured by the SRW11 and SRW10 bits.

Table 2. Sector Limits with Different Settings of SRL2..0

SRL2	SRL1	SRL0	Sector Limits
0	0	0	Lower sector = N/A Upper sector = 0x0260 - 0xFFFF
0	0	1	Lower sector = 0x0260 - 0x1FFF Upper sector = 0x2000 - 0xFFFF
0	1	0	Lower sector = 0x0260 - 0x3FFF Upper sector = 0x4000 - 0xFFFF
0	1	1	Lower sector = 0x0260 - 0x5FFF Upper sector = 0x6000 - 0xFFFF
1	0	0	Lower sector = 0x0260 - 0x7FFF Upper sector = 0x8000 - 0xFFFF
1	0	1	Lower sector = 0x0260 - 0x9FFF Upper sector = 0xA000 - 0xFFFF
1	1	0	Lower sector = 0x0260 - 0xBFFF Upper sector = 0xC000 - 0xFFFF
1	1	1	Lower sector = 0x0260 - 0xDFFF Upper sector = 0xE000 - 0xFFFF

- **Bit 1 and Bit 6 MCUCR – SRW11, SRW10: Wait State Select Bits for Upper Sector**

The SRW11 and SRW10 bits control the number of wait states for the upper sector of the External Memory address space, see Table 3.

- **Bit 3..2 – SRW01, SRW00: Wait State Select Bits for Lower Sector**

The SRW01 and SRW00 bits control the number of wait states for the lower sector of the External Memory address space, see Table 3.

Table 3. Wait States(1)

SRWn1	SRWn0	Wait States
0	0	No wait states.
0	1	Wait one cycle during read/write strobe.
1	0	Wait two cycles during read/write strobe.
1	1	Wait two cycles during read/write and wait one cycle before driving out new address.

Note: 1. n = 0 or 1 (lower/upper sector).
For further details of the timing and wait states of the External Memory Interface, see Figure 13 to Figure 16 how the setting of the SRW bits affects the timing.

Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	XMBK	XMM2	XMM1	XMM0	PUD	–	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – XMBK: External Memory Bus Keeper Enable**

Writing XMBK to one enables the Bus Keeper on the AD7:0 lines. When the Bus Keeper is enabled, AD7:0 will keep the last driven value on the lines even if the XMEM interface has tri-stated the lines. Writing XMBK to zero disables the Bus Keeper. XMBK is not qualified with SRE, so even if the XMEM interface is disabled, the Bus Keepers are still activated as long as XMBK is one.

- **Bit 5..3 – XMM2, XMM1, XMM0: External Memory High Mask**

When the External Memory is enabled, all Port C pins are used for the high address byte by default. If the full 64,928 bytes address space is not required to access the External Memory, some, or all, Port C pins can be released for normal Port Pin function as described in Table 4. As described in “Using all 64KB Locations of External Memory” on page 33, it is possible to use the XMMn bits to access all 64KB locations of the External Memory.

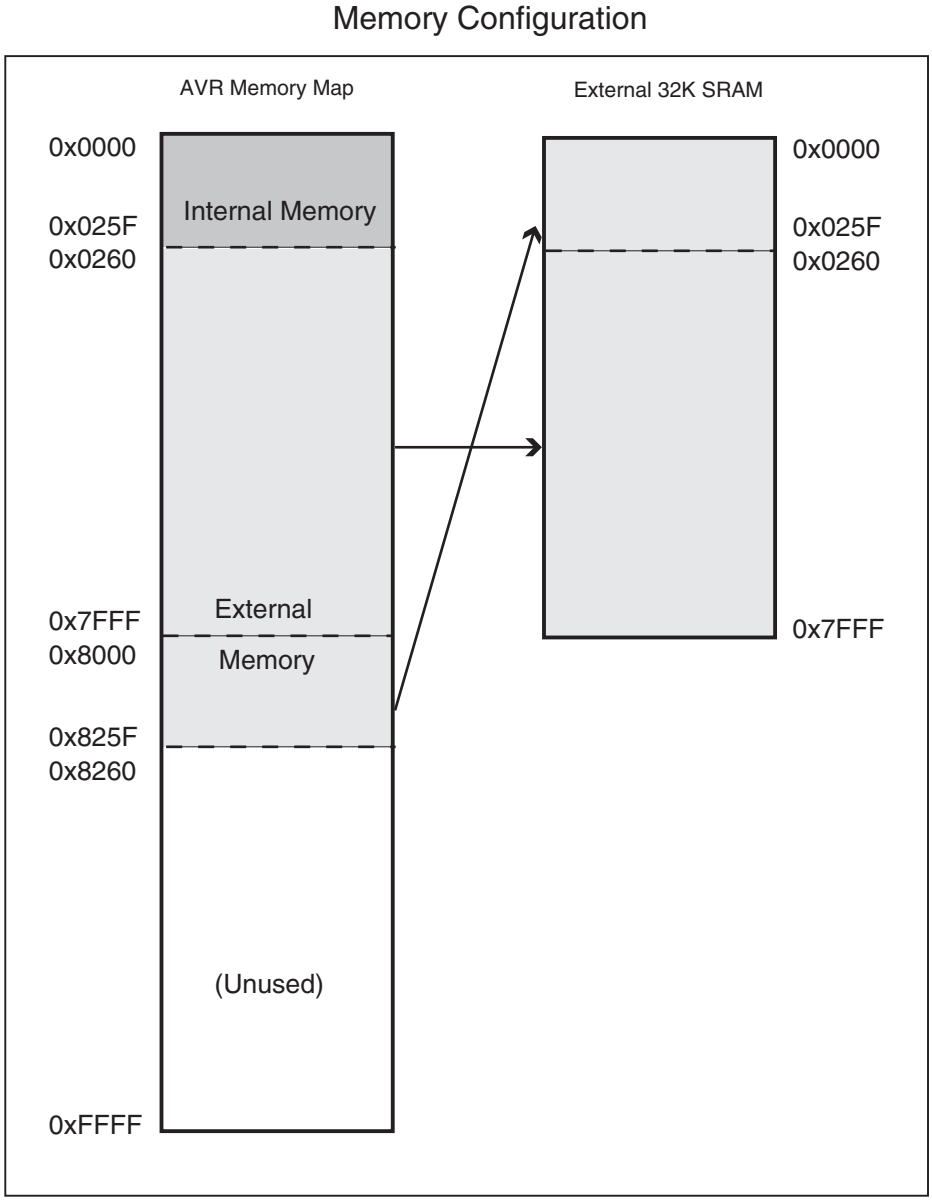
Table 4. Port C Pins Released as Normal Port Pins when the External Memory is Enabled

XMM2	XMM1	XMM0	# Bits for External Memory Address	Released Port Pins
0	0	0	8 (Full 64,928 Bytes Space)	None
0	0	1	7	PC7
0	1	0	6	PC7 - PC6
0	1	1	5	PC7 - PC5
1	0	0	4	PC7 - PC4
1	0	1	3	PC7 - PC3
1	1	0	2	PC7 - PC2
1	1	1	No Address High bits	Full Port C

Using all Locations of External Memory Smaller than 64 KB

Since the external memory is mapped after the internal memory as shown in Figure 11, the external memory is not addressed when addressing the first 608 bytes of data space. It may appear that the first 608 bytes of the external memory are inaccessible (external memory addresses 0x0000 to 0x025F). However, when connecting an external memory smaller than 64 KB, for example 32 KB, these locations are easily accessed simply by addressing from address 0x8000 to 0x825F. Since the External Memory Address bit A15 is not connected to the external memory, addresses 0x8000 to 0x825F will appear as addresses 0x0000 to 0x025F for the external memory. Addressing above address 0x825F is not recommended, since this will address an external memory location that is already accessed by another (lower) address. To the Application software, the external 32 KB memory will appear as one linear 32 KB address space from 0x0260 to 0x825F. This is illustrated in Figure 17.

Figure 17. Address Map with 32 KB External Memory



Using all 64KB Locations of External Memory

Since the External Memory is mapped after the Internal Memory as shown in Figure 11, only 64,928 bytes of External Memory is available by default (address space 0x0000 to 0x025F is reserved for Internal Memory). However, it is possible to take advantage of the entire External Memory by masking the higher address bits to zero. This can be done by using the XMMn bits and control by software the most significant bits of the address. By setting Port C to output 0x00, and releasing the most significant bits for normal Port Pin operation, the Memory Interface will address 0x0000 - 0x1FFF. See code example below.

Assembly Code Example[1]

```
; OFFSET is defined to 0x2000 to ensure
; external memory access
; Configure Port C (address high byte) to
; output 0x00 when the pins are released
; for normal Port Pin operation
ldi  r16, 0xFF
out  DDRC, r16
ldi  r16, 0x00
out  PORTC, r16
; release PC7:5
ldi  r16, (1<<XMM1)|(1<<XMM0)
out  SFIOR, r16
; write 0xAA to address 0x0001 of external
; memory
ldi  r16, 0xaa
sts  0x0001+OFFSET, r16
; re-enable PC7:5 for external memory
ldi  r16, (0<<XMM1)|(0<<XMM0)
out  SFIOR, r16
; store 0x55 to address (OFFSET + 1) of
; external memory
ldi  r16, 0x55
sts  0x0001+OFFSET, r16
```

C Code Example[1]

```
#define OFFSET 0x2000

void XRAM_example(void)
{
unsigned char *p = (unsigned char *) (OFFSET + 1);

DDRC = 0xFF;
PORTC = 0x00;

SFIOR = (1<<XMM1) | (1<<XMM0);

*p = 0xaa;

SFIOR = 0x00;

*p = 0x55;
}
```

Note: 1. See “About Code Examples” on page 7.

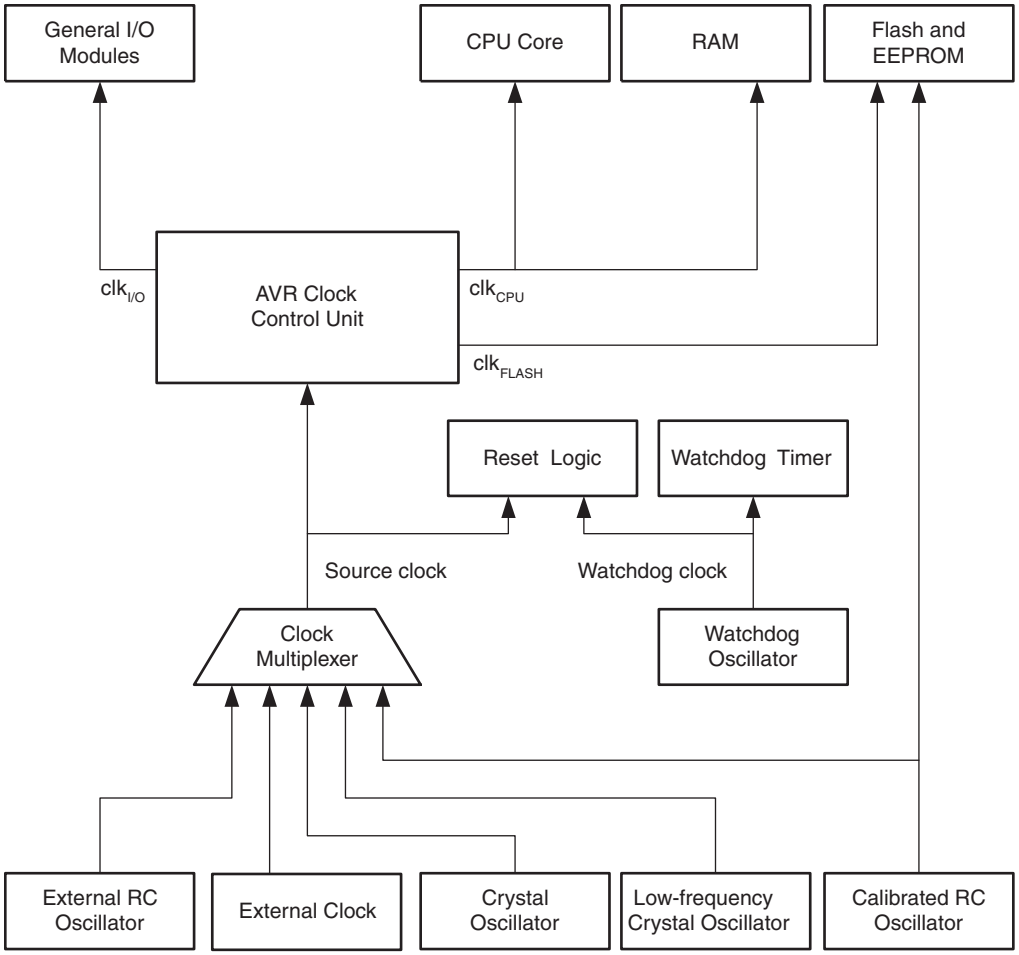
Care must be exercised using this option as most of the memory is masked away.

System Clock and Clock Options

Clock Systems and their Distribution

Figure 18 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in “Power Management and Sleep Modes” on page 41. The clock systems are detailed below.

Figure 18. Clock Distribution



CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register, and the Data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

I/O Clock – $clk_{I/O}$

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

Flash Clock – clk_{FLASH}

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

Clock Sources

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

Table 5. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

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

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 6. The frequency of the Watchdog Oscillator is voltage dependent as shown in “ATmega8515 Typical Characteristics” on page 207.

Table 6. Number of Watchdog Oscillator Cycles

Typ Time-out (V_{CC} = 5.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K 65,536)

Default Clock Source

The device is shipped with CKSEL = “0001” and SUT = “10”. The default clock source setting is therefore the Internal RC Oscillator with longest start-up time. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel Programming.

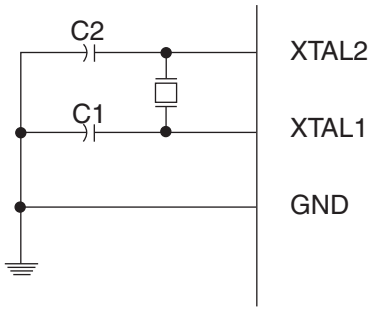
Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 19. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator

in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 7. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 19. Crystal Oscillator Connections



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

Table 7. Crystal Oscillator Operating Modes

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101(1)	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

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

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 8.

Table 8. Start-up Times for the Crystal Oscillator Clock Selection

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

Table 8. Start-up Times for the Crystal Oscillator Clock Selection (Continued)

CKSEL0	SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

Low-frequency Crystal Oscillator

To use a 32.768 kHz watch crystal as the clock source for the device, the Low-frequency Crystal Oscillator must be selected by setting the CKSEL Fuses to “1001”. The crystal should be connected as shown in Figure 19. By programming the CKOPT Fuse, the user can enable internal capacitors on XTAL1 and XTAL2, thereby removing the need for external capacitors. The internal capacitors have a nominal value of 36 pF.

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

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

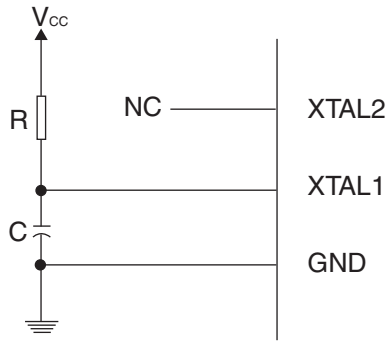
SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	1K CK(1)	4.1 ms	Fast rising power or BOD enabled
01	1K CK(1)	65 ms	Slowly rising power
10	32K CK	65 ms	Stable frequency at start-up
11	Reserved		

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

External RC Oscillator

For timing insensitive applications, the external RC configuration shown in Figure 20 can be used. The frequency is roughly estimated by the equation f = 1/(3RC). C should be at least 22 pF. By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND, thereby removing the need for an external capacitor.

Figure 20. External RC Configuration



The Oscillator can operate in four different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..0 as shown in Table 10.

Table 10. External RC Oscillator Operating Modes

CKSEL3..0	Frequency Range (MHz)
0101	0.1 - 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 11.

Table 11. Start-up Times for the External RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	18 CK	–	BOD enabled
01	18 CK	4.1 ms	Fast rising power
10	18 CK	65 ms	Slowly rising power
11	6 CK(1)	4.1 ms	Fast rising power or BOD enabled

Note: 1. This option should not be used when operating close to the maximum frequency of the device.

Calibrated Internal RC Oscillator

The calibrated internal RC Oscillator provides a fixed 1.0, 2.0, 4.0, or 8.0 MHz clock. All frequencies are nominal values at 5V and 25°C. This clock may be selected as the system clock by programming the CKSEL Fuses as shown in Table 12. If selected, it will operate with no external components. The CKOPT Fuse should always be unprogrammed when using this clock option. During reset, hardware loads the calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. At 5V, 25°C, and 1.0 MHz Oscillator frequency selected, this calibration gives a frequency within ± 3% of the nominal frequency. Using run-time calibration methods as described in application notes available at www.atmel.com/avr it is possible to achieve ± 1% accuracy at any given V_{CC} and Temperature. When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. For more information on the pre-programmed calibration value, see the section “Calibration Byte” on page 181.

Table 12. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency (MHz)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

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

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 13. XTAL1 and XTAL2 should be left unconnected (NC).

Table 13. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10(1)	6 CK	65 ms	Slowly rising power
11	Reserved		

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

Oscillator Calibration Register – OSCCAL

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

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

Writing the calibration byte to this address will trim the internal Oscillator to remove process variations from the Oscillator frequency. During Reset, the 1 MHz calibrated value which is located in the signature row High Byte (address 0x00) is automatically loaded into the OSCCAL Register. If the internal RC is used at other frequencies, the calibration values must be loaded manually. This can be done by first reading the signature row by a programmer, and then store the calibration values in the Flash or EEPROM. Then the value can be read by software and loaded into the OSCCAL Register. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register

will increase the frequency of the internal Oscillator. Writing $FF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0, 2.0, 4.0, or 8.0 MHz. Tuning to other values is not guaranteed, as indicated in Table 14.

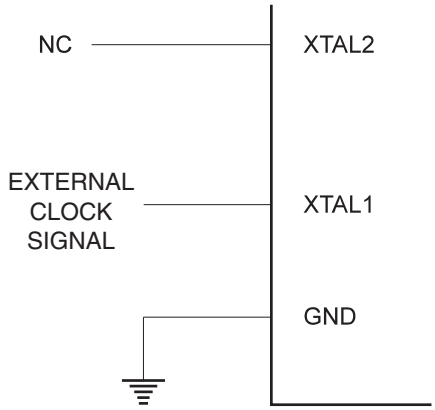
Table 14. Internal RC Oscillator Frequency Range.

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency	Max Frequency in Percentage of Nominal Frequency
$00	50%	100%
$7F	75%	150%
$FF	100%	200%

External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 21. To run the device on an external clock, the CKSEL Fuses must be programmed to “0000”. By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND.

Figure 21. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 15.

Table 15. Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10	6 CK	65 ms	Slowly rising power
11	Reserved		

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

Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the three sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM2 bit in MCUCSR, the SM1 bit in MCUCR, and the SM0 bit in the EMCUCR Register select which sleep mode (Idle, Power-down, or Standby) will be activated by the SLEEP instruction. See Table 16 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, it executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a Reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 18 on page 34 presents the different clock systems in the ATmega8515, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5 – SE: Sleep Enable**

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

- **Bit 4 – SM1: Sleep Mode Select Bit 1**

The Sleep Mode Select bits select between the three available sleep modes as shown in Table 16.

MCU Control and Status Register – MCUCSR

Bit	7	6	5	4	3	2	1	0	
	–	–	SM2	–	WDRF	BORF	EXTRF	PORF	MCUCSR
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 5 – SM2: Sleep Mode Select Bit 2**

The Sleep Mode Select bits select between the three available sleep modes as shown in Table 16.

Extended MCU Control Register – EMCUCR

Bit	7	6	5	4	3	2	1	0	
	SM0	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	ISC2	EMCUCR
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 – SM0: Sleep Mode Select Bit 0**

The Sleep Mode Select bits select between the three available sleep modes as shown in Table 16.

Table 16. Sleep Mode Select

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

Note: 1. Standby mode is only available with external crystals or resonators.

Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, USART, Analog Comparator, Timer/Counters, Watchdog, and the Interrupt System to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the Analog Comparator Control and Status Register – ACSR. This will reduce power consumption in Idle mode.

Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the external Oscillator is stopped, while the External Interrupts and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, an External level interrupt on INT0 or INT1, or an External interrupt on INT2 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to “External Interrupts” on page 77 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period, as described in “Clock Sources” on page 35.

Standby Mode

When the SM2..0 bits are written to 110, and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Table 17. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock domains			Oscillators	Wake-up Sources		
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	Main Clock Source Enabled	INT2 INT1 INT0	SPM/ EEPROM Ready	Other I/O
Idle			X	X	X	X	X
Power-down					X(2)		
Standby(1)				X	X(2)		

Notes: 1. External Crystal or resonator selected as clock source
2. Only INT2 or level interrupt INT1 and INT0

Minimizing Power Consumption

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

Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not needed. In the other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 164 for details on how to configure the Analog Comparator.

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 BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Brown-out Detection" on page 48 for details on how to configure the Brown-out Detector.

Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detector or the Analog Comparator. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to "Internal Voltage Reference" on page 50 for details on the start-up time.

Watchdog Timer

If the Watchdog Timer is not needed in the application, this module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to page 53 for details on how to configure the Watchdog Timer.

Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is to ensure that no pins drive resistive loads. In sleep modes where the I/O clock ($clk_{I/O}$) is stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section “Digital Input Enable and Sleep Modes” on page 63 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

System Control and Reset

Resetting the AVR

During Reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a RJMP instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 22 shows the reset logic. Table 18 defines the electrical parameters of the reset circuitry.

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

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in “Clock Sources” on page 35.

Reset Sources

The ATmega8515 has four sources of reset:

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

Figure 22. Reset Logic

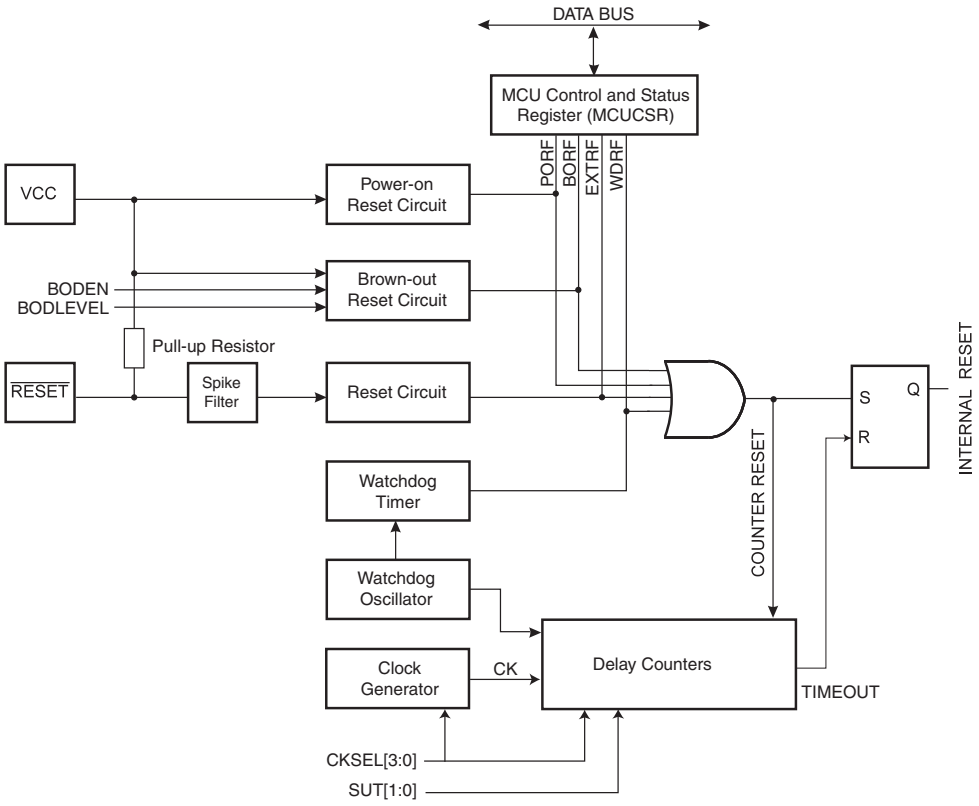


Table 18. Reset Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)[1]			1.4	2.3	V
	Power-on Reset Threshold Voltage (falling)			1.3	2.3	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.1		0.9	V_{CC}
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin				1.5	µs
V_{BOT}	Brown-out Reset Threshold Voltage[2]	BODLEVEL = 1	2.5	2.7	3.2	V
		BODLEVEL = 0	3.7	4.0	4.5	
t_{BOD}	Minimum low voltage period for Brown-out Detection	BODLEVEL = 1		2		µs
		BODLEVEL = 0		2		µs
V_{HYST}	Brown-out Detector hysteresis			130		mV

Notes: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

2. 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. The test is performed using BODLEVEL=1 for ATmega8515L and BODLEVEL=0 for ATmega8515. BODLEVEL=1 is not applicable for ATmega8515.

Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 18. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

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

Figure 23. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

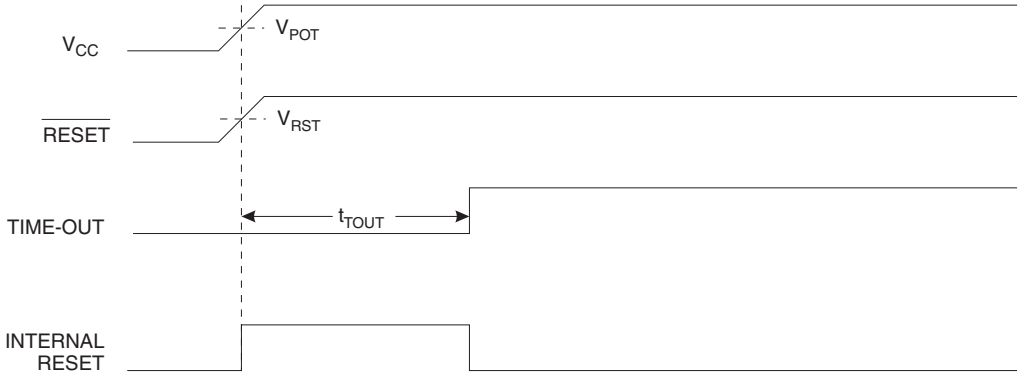
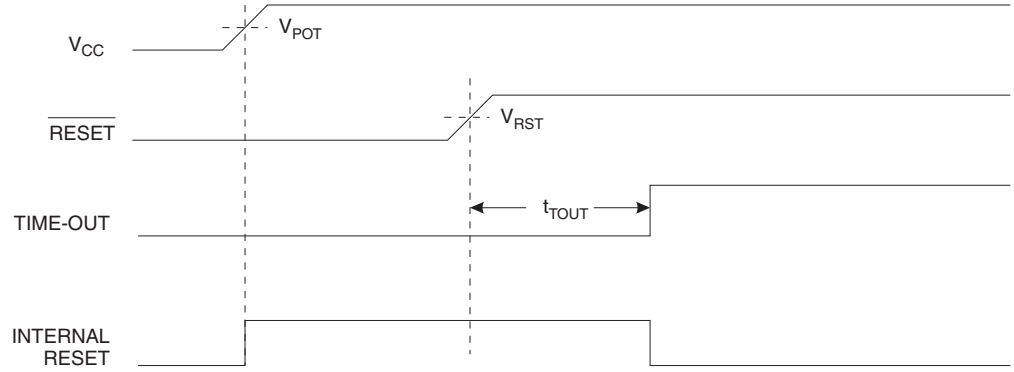


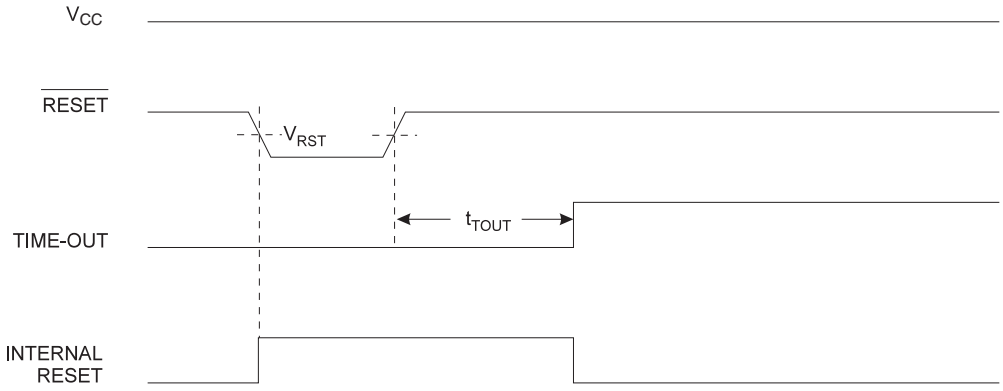
Figure 24. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 18) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

Figure 25. External Reset During Operation



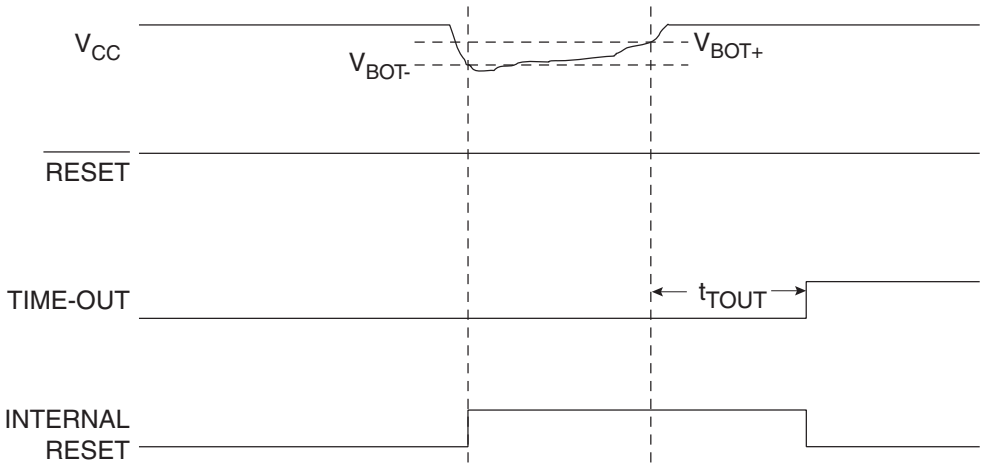
Brown-out Detection

ATmega8515 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the fuse BODLEVEL to be 2.7V (BODLEVEL unprogrammed), or 4.0V (BODLEVEL programmed). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD circuit can be enabled/disabled by the fuse BODEN. When the BOD is enabled (BODEN programmed), and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 26), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 26), the delay counter starts the MCU after the time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 18.

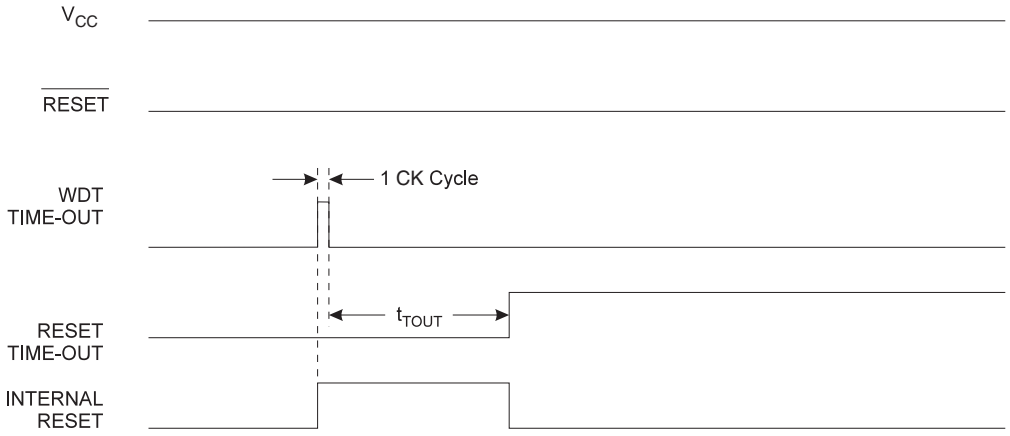
Figure 26. Brown-out Reset During Operation



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 53 for details on operation of the Watchdog Timer.

Figure 27. Watchdog Reset During Operation



MCU Control and Status Register – MCUCSR

The MCU Control and Status Register provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
	–	–	SM2	–	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- **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 MCUCSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

Internal Voltage Reference

ATmega8515 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator.

Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 19. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODEN Fuse).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).

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

Table 19. Internal Voltage Reference Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{BG}	Bandgap reference voltage	1.15	1.23	1.35	V
t_{BG}	Bandgap reference start-up time		40	70	µs
I_{BG}	Bandgap reference current consumption		10		µA

Watchdog Timer

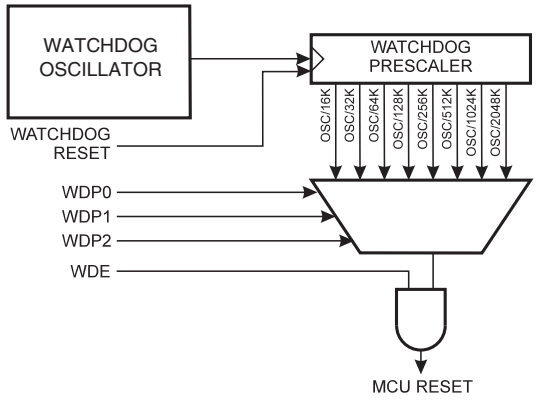
The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1 MHz. This is the typical frequency at V_{CC} = 5V. See characterization data for typical values at other V_{CC} levels. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 21 on page 52. 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. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATmega8515 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 49.

To prevent unintentional disabling of the Watchdog or unintentional change of time-out period, three different safety levels are selected by the Fuses S8515C and WDTON as shown in Table 20. Safety level 0 corresponds to the setting in AT90S4414/8515. There is no restriction on enabling the WDT in any of the safety levels. Refer to “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 53 for details.

Table 20. WDT Configuration as a Function of the Fuse Settings of S8515C and WDTON.

S8515C	WDTON	Safety Level	WDT Initial State	How to Disable the WDT	How to Change Time-out
Unprogrammed	Unprogrammed	1	Disabled	Timed sequence	Timed sequence
Unprogrammed	Programmed	2	Enabled	Always enabled	Timed sequence
Programmed	Unprogrammed	0	Disabled	Timed sequence	No restriction
Programmed	Programmed	2	Enabled	Always enabled	Timed sequence

Figure 28. Watchdog Timer



Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8515 and will always read as zero.

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

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure. In Safety Levels 1 and 2, this bit must also be set when changing the prescaler bits. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 53.

- **Bit 3 – WDE: Watchdog Enable**

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

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

In safety level 2, it is not possible to disable the Watchdog Timer, even with the algorithm described above. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 53.

- **Bits 2..0 – WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1, and 0**

The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 21.

Table 21. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

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

Assembly Code Example

```
WDT_off:
  ; Write logical one to WDCE and WDE
  ldi  r16, (1<<WDCE)|(1<<WDE)
  out  WDTCR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCR, r16
  ret
```

C Code Example

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

Timed Sequences for Changing the Configuration of the Watchdog Timer

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

Safety Level 0

This mode is compatible with the Watchdog operation found in AT90S4414/8515. The Watchdog Timer is initially disabled, but can be enabled by writing the WDE bit to 1 without any restriction. The time-out period can be changed at any time without restriction. To disable an enabled Watchdog Timer, the procedure described on page 51 (WDE bit description) must be followed.

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 changing the Watchdog Time-out period or disabling an enabled Watchdog Timer. To disable an enabled Watchdog Timer, and/or changing the Watchdog Time-out, the following procedure must be followed:

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

Safety Level 2

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

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

Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega8515. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 13.

Interrupt Vectors in ATmega8515

Table 22. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$005	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$006	TIMER1 OVF	Timer/Counter1 Overflow
8	$007	TIMER0 OVF	Timer/Counter0 Overflow
9	$008	SPI, STC	Serial Transfer Complete
10	$009	USART, RXC	USART, Rx Complete
11	$00A	USART, UDRE	USART Data Register Empty
12	$00B	USART, TXC	USART, Tx Complete
13	$00C	ANA_COMP	Analog Comparator
14	$00D	INT2	External Interrupt Request 2
15	$00E	TIMER0 COMP	Timer/Counter0 Compare Match
16	$00F	EE_RDY	EEPROM Ready
17	$010	SPM_RDY	Store Program memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see “Boot Loader Support – Read-While-Write Self-Programming” on page 166.
2. When the IVSEL bit in GICR is set, Interrupt Vectors will be moved to the start of the Boot Flash section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash section.

Table 23 shows Reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 23. Reset and Interrupt Vectors Placement[1]

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	$0000	$0001
1	1	$0000	Boot Reset Address + $0001
0	0	Boot Reset Address	$0001
0	1	Boot Reset Address	Boot Reset Address + $0001

Note: 1. The Boot Reset Address is shown in Table 78 on page 177. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega8515 is:

```
Address  Labels   Code              Comments
$000              rjmp RESET        ; Reset Handler
$001              rjmp EXT_INT0     ; IRQ0 Handler
$002              rjmp EXT_INT1     ; IRQ1 Handler
$003              rjmp TIM1_CAPT    ; Timer1 Capture Handler
$004              rjmp TIM1_COMPA   ; Timer1 Compare A Handler
$005              rjmp TIM1_COMPB   ; Timer1 Compare B Handler
$006              rjmp TIM1_OVF     ; Timer1 Overflow Handler
$007              rjmp TIM0_OVF     ; Timer0 Overflow Handler
$008              rjmp SPI_STC      ; SPI Transfer Complete Handler
$009              rjmp USART_RXC    ; USART RX Complete Handler
$00a              rjmp USART_UDRE   ; UDR0 Empty Handler
$00b              rjmp USART_TXC    ; USART TX Complete Handler
$00c              rjmp ANA_COMP     ; Analog Comparator Handler
$00d              rjmp EXT_INT2     ; IRQ2 Handler
$00e              rjmp TIM0_COMP    ; Timer0 Compare Handler
$00f              rjmp EE_RDY       ; EEPROM Ready Handler
$010              rjmp SPM_RDY      ; Store Program memory Ready
Handler

$011     RESET:   ldi  r16,high(RAMEND); Main program start
$012              out  SPH,r16      ; Set Stack Pointer to top of RAM
$013              ldi  r16,low(RAMEND)
$014              out  SPL,r16
$015              sei               ; Enable interrupts
$016              <instr>  xxx
...      ...      ...
```

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2K bytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address  Labels   Code                 Comments
$000     RESET:  ldi  r16,high(RAMEND); Main program start
$001             out  SPH,r16       ; Set Stack Pointer to top of RAM
$002             ldi  r16,low(RAMEND)
$003             out  SPL,r16
$004             sei                ; Enable interrupts
$005             <instr>  xxx
;
.org $C02
$C02             rjmp EXT_INT0      ; IRQ0 Handler
$C04             rjmp EXT_INT1      ; IRQ1 Handler
...      ....    ..                 ;
$C2A             rjmp SPM_RDY       ; Store Program memory Ready
Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 2K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address  Labels   Code               Comments
.org $002
$001             rjmp EXT_INT0      ; IRQ0 Handler
$002             rjmp EXT_INT1      ; IRQ1 Handler
...      ....    ..                 ;
$010             rjmp SPM_RDY       ; Store Program memory Ready
Handler
;
.org $C00
$C00     RESET:  ldi  r16,high(RAMEND); Main program start
$C01             out  SPH,r16       ; Set Stack Pointer to top of RAM
$C02             ldi  r16,low(RAMEND)
$C03             out  SPL,r16
$C04             sei                ; Enable interrupts
$C05             <instr>  xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 2K bytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address  Labels   Code                 Comments
.org $C00
$C00             rjmp RESET         ; Reset handler
$C01             rjmp EXT_INT0      ; IRQ0 Handler
$C02             rjmp EXT_INT1      ; IRQ1 Handler
...      ....    ..                 ;
$C10             rjmp SPM_RDY       ; Store Program memory Ready
Handler
;
$C11     RESET:  ldi  r16,high(RAMEND); Main program start
```

```
$C12            out  SPH,r16      ; Set Stack Pointer to top of RAM
$C13            ldi  r16,low(RAMEND)
$C14            out  SPL,r16
$C15            sei               ; Enable interrupts
$C16            <instr>  xxx
```

Moving Interrupts between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table.

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – IVSEL: Interrupt Vector Select**

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash section is determined by the BOOTSZ Fuses. Refer to the section “Boot Loader Support – Read-While-Write Self-Programming” on page 166 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section. Refer to the section “Boot Loader Support – Read-While-Write Self-Programming” on page 166 for details on Boot Lock bits.

- **Bit 0 – IVCE: Interrupt Vector Change Enable**

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example

```
Move_interrupts:
  ; Enable change of interrupt vectors
  ldi  r16, (1<<IVCE)
  out  GICR, r16
  ; Move interrupts to boot flash section
  ldi  r16, (1<<IVSEL)
  out  GICR, r16
  ret
```

C Code Example

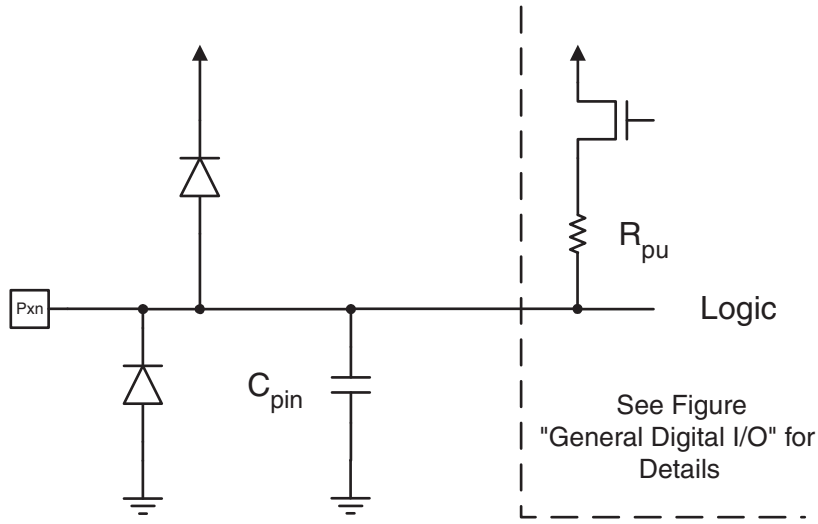
```
void Move_interrupts(void)
{
  /* Enable change of interrupt vectors */
  GICR = (1<<IVCE);
  /* Move interrupts to boot flash section */
  GICR = (1<<IVSEL);
}
```

I/O Ports

Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 29. Refer to “Electrical Characteristics” on page 197 for a complete list of parameters.

Figure 29. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in “Register Description for I/O Ports” on page 75.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all ports when set.

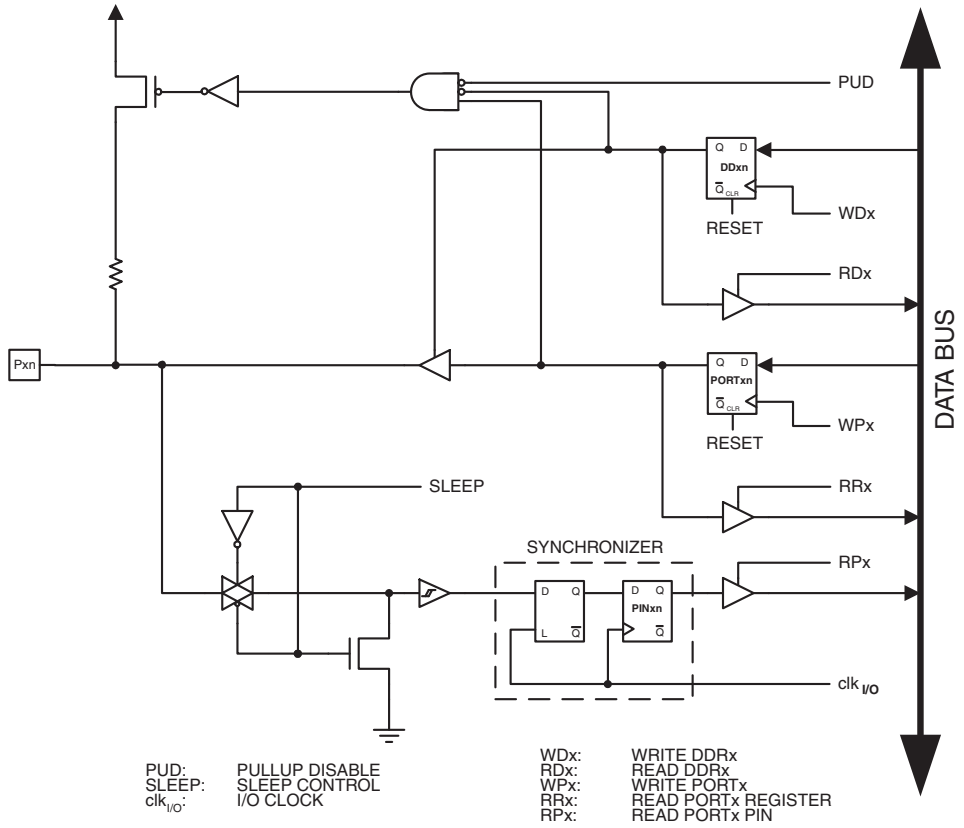
Using the I/O port as General Digital I/O is described in “Ports as General Digital I/O” on page 60. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in “Alternate Port Functions” on page 64. Refer to the individual module sections for a full description of the alternate functions.

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

Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 30 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 30. General Digital I/O(1)



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

Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O Ports" on page 75, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

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

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

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

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up

enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 24 summarizes the control signals for the pin value.

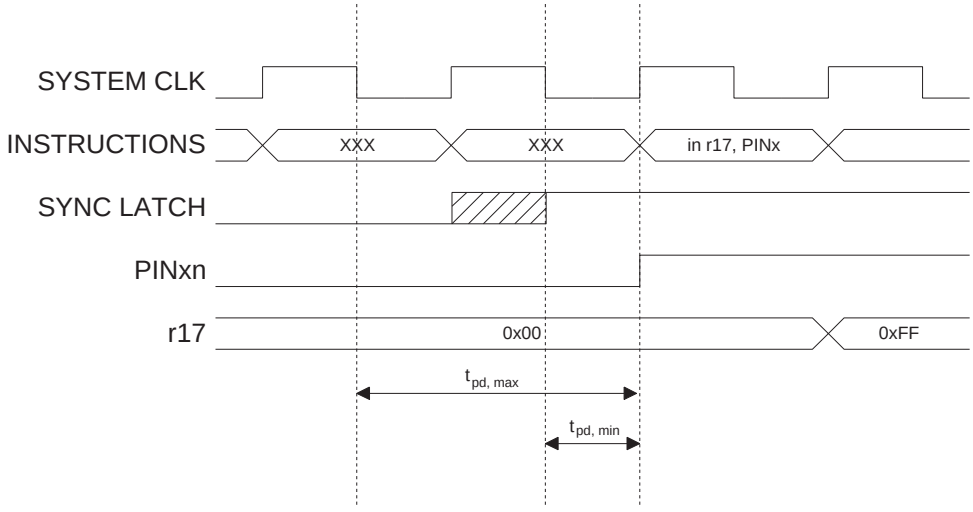
Table 24. Port Pin Configurations

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

Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 30, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 31 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 31. Synchronization when Reading an Externally Applied Pin Value

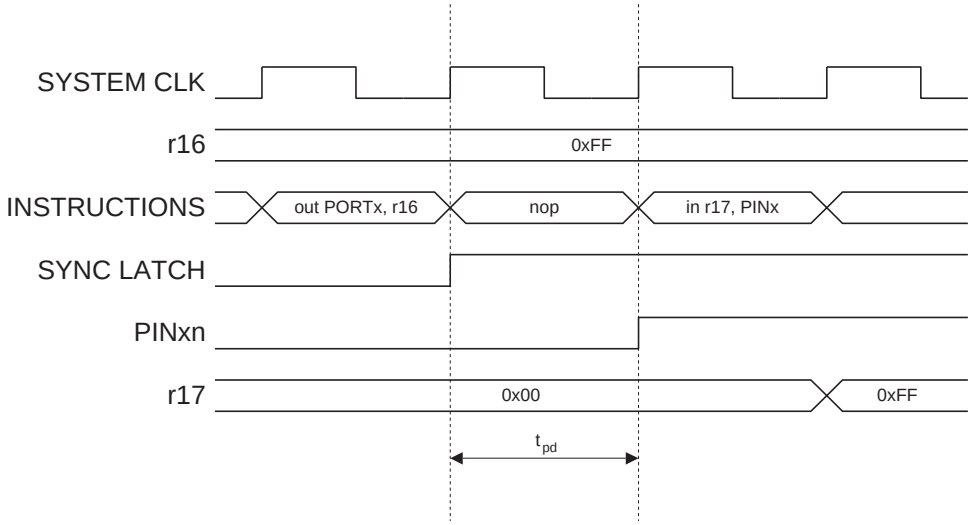


Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min}$, a single

signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a *nop* instruction must be inserted as indicated in Figure 32. The *out* instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay t_{pd} through the synchronizer is one system clock period.

Figure 32. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a *nop* instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi  r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out  PORTB,r16
out  DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in   r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINB;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

Digital Input Enable and Sleep Modes

As shown in Figure 30, the digital input signal can be clamped to ground at the input of the Schmitt Trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as External Interrupt pins. If the External Interrupt Request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in “Alternate Port Functions” on page 64.

If a logic high level (“one”) is present on an Asynchronous External Interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.

Unconnected pins

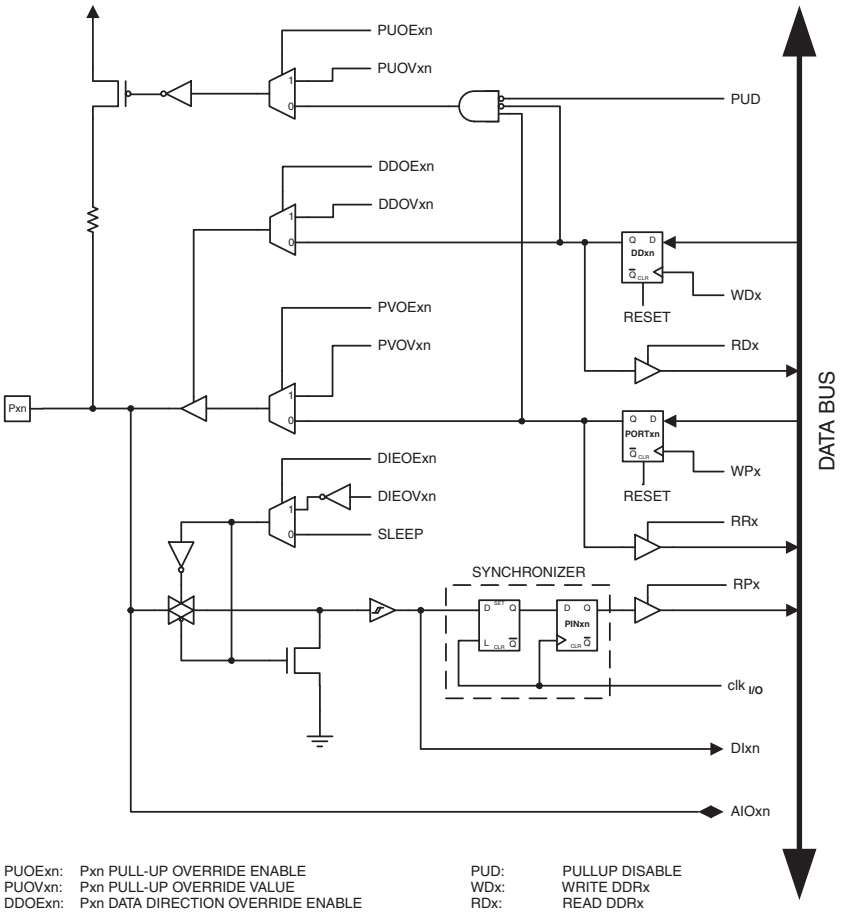
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

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

Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 33 shows how the port pin control signals from the simplified Figure 30 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 33. Alternate Port Functions(1)



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

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

Table 25 summarizes the function of the overriding signals. The pin and port indexes from Figure 33 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 25. Generic Description of Overriding Signals for Alternate Functions.

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
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 modes).
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 modes).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/output	This is the Analog Input/Output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

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

Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	XMBK	XMM2	XMM1	XMM0	PUD	–	PSR10	SFIOR
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 2 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See “Configuring the Pin” on page 60 for more details about this feature.

Alternate Functions of Port A

Port A has an alternate function as the address low byte and data lines for the External Memory Interface.

Table 26. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	AD7 (External memory interface address and data bit 7)
PA6	AD6 (External memory interface address and data bit 6)
PA5	AD5 (External memory interface address and data bit 5)
PA4	AD4 (External memory interface address and data bit 4)
PA3	AD3 (External memory interface address and data bit 3)
PA2	AD2 (External memory interface address and data bit 2)
PA1	AD1 (External memory interface address and data bit 1)
PA0	AD0 (External memory interface address and data bit 0)

Table 27 and Table 28 relate the alternate functions of Port A to the overriding signals shown in Figure 33 on page 64.

Table 27. Overriding Signals for Alternate Functions in PA7..PA4

Signal Name	PA7/AD7	PA6/AD6	PA5/AD5	PA4/AD4
PUOE	SRE	SRE	SRE	SRE
PUOV	~($\overline{WR}$ | ADA[1]) • PortA7	~($\overline{WR}$ | ADA) • PortA6	~($\overline{WR}$ | ADA) • PortA5	~($\overline{WR}$ | ADA) • PortA4
DDOE	SRE	SRE	SRE	SRE
DDOV	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A7 • ADA | D7 OUTPUT • $\overline{WR}$	A6 • ADA | D6 OUTPUT • $\overline{WR}$	A5 • ADA | D5 OUTPUT • $\overline{WR}$	A4 • ADA | D4 OUTPUT • $\overline{WR}$
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D7 INPUT	D6 INPUT	D5 INPUT	D4 INPUT
AIO	–	–	–	–

Note: 1. ADA is short for ADdress Active and represents the time when address is output. See “External Memory Interface” on page 25.

Table 28. Overriding Signals for Alternate Functions in PA3..PA0

Signal Name	PA3/AD3	PA2/AD2	PA1/AD1	PA0/AD0
PUOE	SRE	SRE	SRE	SRE
PUOV	~($\overline{WR}$ | ADA) • PortA3	~($\overline{WR}$ | ADA) • PortA2	~($\overline{WR}$ | ADA) • PortA1	~($\overline{WR}$ | ADA) • PortA0
DDOE	SRE	SRE	SRE	SRE
DDOV	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A3 • ADA | D3 OUTPUT • $\overline{WR}$	A2 • ADA | D2 OUTPUT • $\overline{WR}$	A1 • ADA | D1 OUTPUT • $\overline{WR}$	A0 • ADA | D0 OUTPUT • $\overline{WR}$
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D3 INPUT	D2 INPUT	D1 INPUT	D0 INPUT
AIO	–	–	–	–

Alternate Functions Of Port B

The Port B pins with alternate functions are shown in Table 29.

Table 29. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input)
PB2	AIN0 (Analog Comparator Positive Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) OC0 (Timer/Counter0 Output Compare Match Output)

The alternate pin configuration is as follows:

- **SCK – Port B, Bit 7**

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB7. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB7 bit.

- **MISO – Port B, Bit 6**

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB6. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB6 bit.

- **MOSI – Port B, Bit 5**

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB5. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

- **$\overline{SS}$ – Port B, Bit 4**

$\overline{SS}$: Slave Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB4. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit.

- **AIN1 – Port B, Bit 3**

AIN1, Analog Comparator Negative input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

- **AIN0 – Port B, Bit 2**

AIN0, Analog Comparator Positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

- **T1 – Port B, Bit 1**

T1, Timer/Counter1 Counter Source.

- **T0/OC0 – Port B, Bit 0**

T0, Timer/Counter0 Counter Source.

OC0, Output Compare Match output: The PB0 pin can serve as an external output for the Timer/Counter0 Compare Match. The PB0 pin has to be configured as an output (DDB0 set (one)) to serve this function. The OC0 pin is also the output pin for the PWM mode timer function.

Table 31 relate the alternate functions of Port B to the overriding signals shown in Figure 33 on page 64. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 30. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/SCK	PB6/MISO	PB5/MOSI	PB4/$\overline{SS}$
PUOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
PUOV	PORTB7 • $\overline{PUD}$	PORTB6 • $\overline{PUD}$	PORTB5 • $\overline{PUD}$	PORTB4 • $\overline{PUD}$
DDOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
DDOV	0	0	0	0
PVOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • MSTR	0
PVOV	SCK OUTPUT	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SCK INPUT	SPI MSTR INPUT	SPI SLAVE INPUT	SPI $\overline{SS}$
AIO	–	–	–	–

Table 31. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/AIN1	PB2/AIN0	PB1/T1	PB0/T0/OC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	1	0	0	0
PVOE	0	0	0	OC0 ENABLE
PVOV	0	0	0	OC0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	0	T1 INPUT	T0 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 32.

Table 32. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	A15 (External memory interface address bit 15)
PC6	A14 (External memory interface address bit 14)
PC5	A13 (External memory interface address bit 13)
PC4	A12 (External memory interface address bit 12)
PC3	A11 (External memory interface address bit 11)
PC2	A10 (External memory interface address bit 10)
PC1	A9 (External memory interface address bit 9)
PC0	A8 (External memory interface address bit 8)

- **A15 – Port C, Bit 7**

A15, External memory interface address bit 15.

- **A14 – Port C, Bit 6**

A14, External memory interface address bit 14.

- **A13 – Port C, Bit 5**

A13, External memory interface address bit 13.

- **A12 – Port C, Bit 4**

A12, External memory interface address bit 12.

- **A11 – Port C, Bit 3**

A11, External memory interface address bit 11.

- **A10 – Port C, Bit 2**

A10, External memory interface address bit 10.

- **A9 – Port C, Bit 1**

A9, External memory interface address bit 9.

- **A8 – Port C, Bit 0**

A8, External memory interface address bit 8.

Table 33 and Table 34 relate the alternate functions of Port C to the overriding signals shown in Figure 33 on page 64.

Table 33. Overriding Signals for Alternate Functions in PC7..PC4

Signal Name	PC7/A15	PC6/A14	PC5/A13	PC4/A12
PUOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PUOV	0	0	0	0
DDOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
DDOV	1	1	1	1
PVOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PVOV	A15	A14	A13	A12
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	–	–	–	–

Table 34. Overriding Signals for Alternate Functions in PC3..PC0

Signal Name	PC3/A11	PC2/A10	PC1/A9	PC0/A8
PUOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
PUOV	0	0	0	0
DDOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
DDOV	1	1	1	1
PVOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
PVOV	A11	A10	A9	A8
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	–	–	–	–

Alternate Functions of Port D

The Port D pins with alternate functions are shown in Table 35.

Table 35. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	$\overline{RD}$ (Read Strobe to External Memory)
PD6	$\overline{WR}$ (Write Strobe to External Memory)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	XCK (USART External Clock Input/Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

The alternate pin configuration is as follows:

- **$\overline{RD}$ – Port D, Bit 7**

$\overline{RD}$ is the External Data memory read control strobe.

- **$\overline{WR}$ – Port D, Bit 6**

$\overline{WR}$ is the External Data memory write control strobe.

- **OC1A – Port D, Bit 5**

OC1A, Output Compare Match A output: The PD5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **XCK – Port D, Bit 4**

XCK, USART External Clock. The Data Direction Register (DDD4) controls whether the clock is output (DDD4 set) or input (DDD4 cleared). The XCK pin is active only when USART operates in Synchronous mode.

- **INT1 – Port D, Bit 3**

INT1, External Interrupt source 1: The PD3 pin can serve as an external interrupt source.

- **INT0/XCK1 – Port D, Bit 2**

INT0, External Interrupt Source 0: The PD2 pin can serve as an external interrupt source.

XCK1, External Clock. The Data Direction Register (DDD2) controls whether the clock is output (DDD2 set) or input (DDD2 cleared).

- **TXD – Port D, Bit 1**

TXD, Transmit Data (Data output pin for USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

- **RXD – Port D, Bit 0**

RXD, Receive Data (Data input pin for USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

Table 36 and Table 37 relate the alternate functions of Port D to the overriding signals shown in Figure 33 on page 64.

Table 36. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/$\overline{RD}$	PD6/$\overline{WR}$	PD5/OC1A	PD4/XCK
PUOE	SRE	SRE	0	0
PUOV	0	0	0	0
DDOE	SRE	SRE	0	0
DDOV	1	1	0	0
PVOE	SRE	SRE	OC1A ENABLE	XCK OUTPUT ENABLE
PVOV	$\overline{RD}$	$\overline{WR}$	OC1A	XCK OUTPUT
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	XCK INPUT
AIO	–	–	–	–

Table 37. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN0	RXEN0
PUOV	0	0	0	PORTD0 • $\overline{PUD}$
DDOE	0	0	TXEN0	RXEN0
DDOV	0	0	1	0
PVOE	0	0	TXEN0	0
PVOV	0	0	TXD	0
DIEOE	INT1 ENABLE	INT0 ENABLE	0	0
DIEOV	1	1	0	0
DI	INT1 INPUT	INT0 INPUT	–	RXD
AIO	–	–	–	–

Alternate Functions of Port E

The Port E pins with alternate functions are shown in Table 38.

Table 38. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE2	OC1B (Timer/Counter1 Output Compare B Match Output)
PE1	ALE (Address Latch Enable to External Memory)
PE0	ICP (Timer/Counter1 Input Capture Pin) INT2 (External Interrupt 2 Input)

The alternate pin configuration is as follows:

- **OC1B – Port E, Bit 2**

OC1B, Output Compare Match B output: The PE2 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDE2 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **ALE – Port E, Bit 1**

ALE is the external Data memory Address Latch Enable signal.

- **ICP/INT2 – Port E, Bit 0**

ICP – Input Capture Pin: The PE0 pin can act as an Input Capture pin for Timer/Counter1.

INT2, External Interrupt Source 2: The PE0 pin can serve as an external interrupt source.

Table 39 relate the alternate functions of Port E to the overriding signals shown in Figure 33 on page 64.

Table 39. Overriding Signals for Alternate Functions PE2..PE0

Signal Name	PE2	PE1	PE0
PUOE	0	SRE	0
PUOV	0	0	0
DDOE	0	SRE	0
DDOV	0	1	0
PVOE	OC1B OVERRIDE ENABLE	SRE	0
PVOV	OC1B	ALE	0
DIEOE	0	0	INT2 ENABLED
DIEOV	0	0	1
DI	0	0	INT2 INPUT, ICP INPUT
AIO	–	–	–

Register Description for I/O Ports

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	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	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	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	

Port A Input Pins Address – PINA

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

Port B Data Register – PORTB

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

Port B Data Direction Register – DDRB

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

Port B Input Pins Address – PINB

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

Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
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	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
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	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port E Data Register – PORTE

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PORTE2	PORTE1	PORTE0	PORTE
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port E Data Direction Register – DDRE

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	DDE2	DDE1	DDE0	DDRE
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port E Input Pins Address – PINE

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PINE2	PINE1	PINE0	PINE
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

External Interrupts

The External Interrupts are triggered by the INT0, INT1, and INT2 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..2 pins are configured as outputs. This feature provides a way of generating a software interrupt. The External Interrupts can be triggered by a falling or rising edge or a low level (INT2 is only an edge triggered interrupt). This is set up as indicated in the specification for the MCU Control Register – MCUCR and Extended MCU Control Register – EMCUCR. When the External Interrupt is enabled and is configured as level triggered (only INT0/INT1), the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 and INT1 requires the presence of an I/O clock, described in "Clock Systems and their Distribution" on page 34. Low level interrupts on INT0/INT1 and the edge interrupt on INT2 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. This makes the MCU less sensitive to noise. The changed level is sampled twice by the Watchdog Oscillator clock. The period of the Watchdog Oscillator is 1 µs (nominal) at 5.0V and 25°C. The frequency of the Watchdog Oscillator is voltage dependent as shown in "Electrical Characteristics" on page 197. The MCU will wake up if the input has the required level during this sampling or if it is held until the end of the start-up time. The start-up time is defined by the SUT Fuses as described in "System Clock and Clock Options" on page 34. If the level is sampled twice by the Watchdog Oscillator clock but disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The required level must be held long enough for the MCU to complete the wake up to trigger the level interrupt.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-bit and the corresponding interrupt mask in the GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 40. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 40. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

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

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 41. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 41. Interrupt 0 Sense Control

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

Extended MCU Control Register – EMCUCR

Bit	7	6	5	4	3	2	1	0	
	SM0	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	ISC2	EMCUCR
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 0 – ISC2: Interrupt Sense Control 2**

The Asynchronous External Interrupt 2 is activated by the external pin INT2 if the SREG I-bit and the corresponding interrupt mask in GICR are set. If ISC2 is written to zero, a falling edge on INT2 activates the interrupt. If ISC2 is written to one, a rising edge on INT2 activates the interrupt. Edges on INT2 are registered asynchronously. Pulses on INT2 wider than the minimum pulse width given in Table 42 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. When changing the ISC2 bit, an interrupt can occur. Therefore, it is recommended to first disable INT2 by clearing its Interrupt Enable bit in the GICR Register. Then, the ISC2 bit can be changed. Finally, the INT2 Interrupt Flag should be cleared by writing a logical one to its Interrupt Flag bit (INTF2) in the GIFR Register before the interrupt is re-enabled.

Table 42. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The

corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

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

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU General Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

- **Bit 5 – INT2: External Interrupt Request 2 Enable**

When the INT2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control2 bit (ISC2) in the MCU Control and Status Register (MCUCSR) defines whether the external interrupt is activated on rising or falling edge of the INT2 pin. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 Interrupt Vector.

General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

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

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

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

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

- **Bit 5 – INTF2: External Interrupt Flag 2**

When an event on the INT2 pin triggers an interrupt request, INTF2 becomes set (one). If the I-bit in SREG and the INT2 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. Note that when entering some sleep modes with the INT2 interrupt disabled, the input buffer on this pin will be disabled. This may cause a logic change in internal signals which will set the INTF2 Flag. See “Digital Input Enable and Sleep Modes” on page 63 for more information.

8-bit Timer/Counter0 with PWM

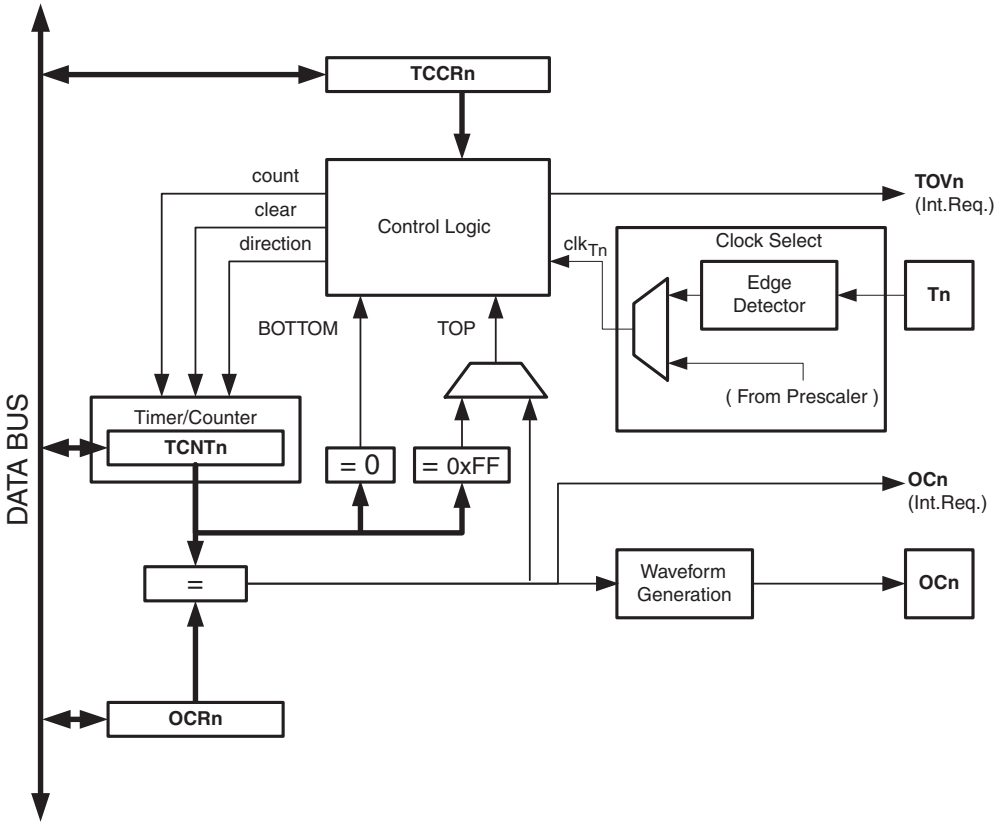
Timer/Counter0 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single Channel Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **External Event Counter**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)**

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 34. For the actual placement of I/O pins, refer to “Pinout ATmega8515” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “8-bit Timer/Counter Register Description” on page 91.

Figure 34. 8-bit Timer/Counter Block Diagram



Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

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 Register (OCR0) is compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare Pin (OC0). See “Output Compare Unit” on page 82. for details. The Compare Match event will also set the Compare Flag (OCF0) which can be used to generate an output compare interrupt request.

Definitions

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

The definitions in Table 43 are also used extensively throughout the document.

Table 43. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0 Register. The assignment is dependent on the mode of operation.

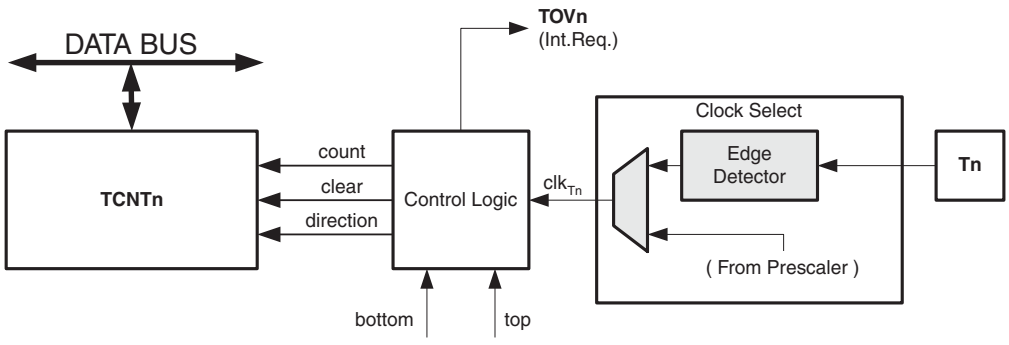
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the clock select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 95.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 35 shows a block diagram of the counter and its surroundings.

Figure 35. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT0 by 1.

direction Select between increment and decrement.

clear Clear TCNT0 (set all bits to zero).

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

top Signalize that TCNT0 has reached maximum value.

bottom Signalize that TCNT0 has reached minimum value (zero).

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

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC0. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 85.

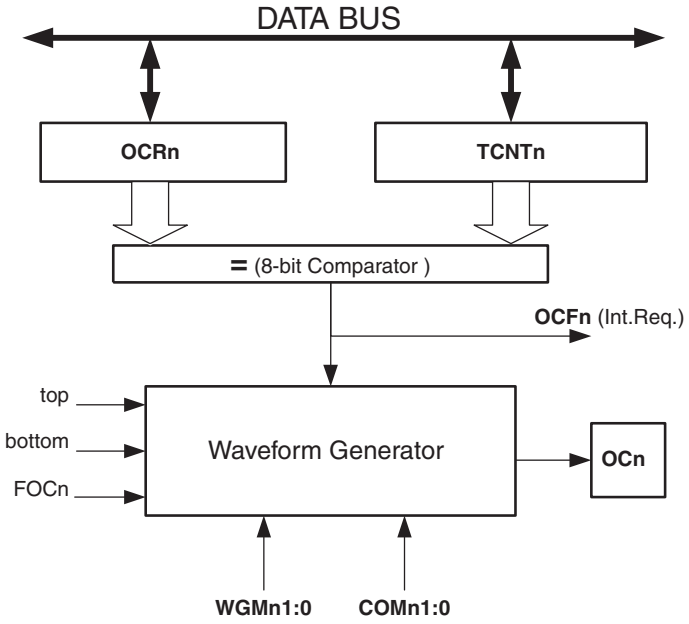
The Timer/Counter Overflow (TOV0) Flag is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0). Whenever TCNT0 equals OCR0, the comparator signals a match. A match will set the Output Compare Flag (OCF0) at the next timer clock cycle. If enabled (OCIE0 = 1 and Global Interrupt Flag in SREG is set), the Output Compare Flag generates an output compare interrupt. The OCF0 Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF0 Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM01:0 bits and Compare Output mode (COM01:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation. See “Modes of Operation” on page 85.

Figure 36 shows a block diagram of the output compare unit.

Figure 36. Output Compare Unit, Block Diagram



The OCR0 Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0 Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0 Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0 Buffer Register, and if double buffering is disabled the CPU will access the OCR0 directly.

Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0) bit. Forcing Compare Match will not set the OCF0 Flag or reload/clear the timer, but the OC0 pin will be updated as if a real Compare Match had occurred (the COM01:0 bits settings define whether the OC0 pin is set, cleared or toggled).

Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0 to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0 value, the Compare Match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is downcounting.

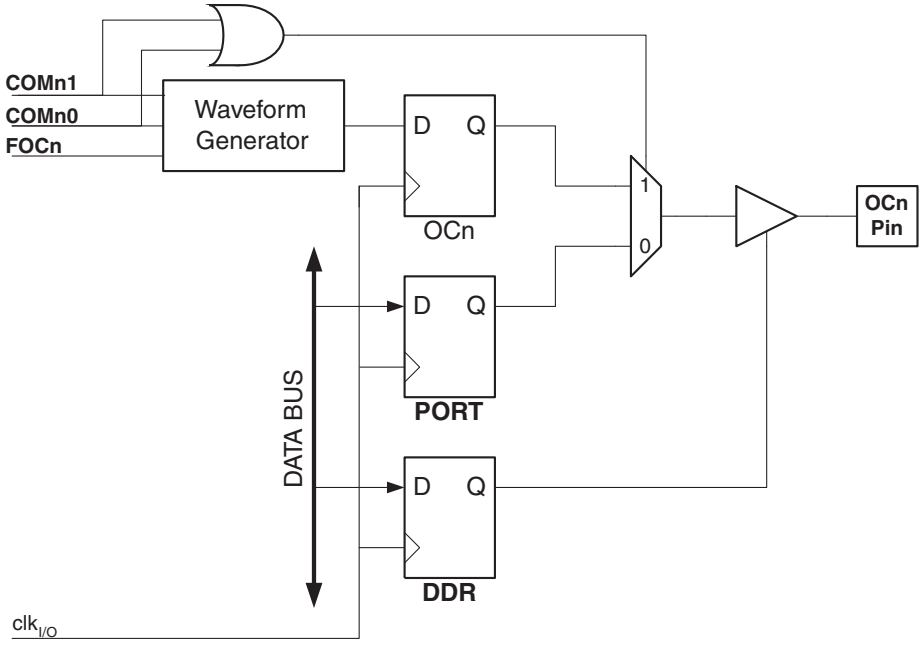
The setup of the OC0 should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0 value is to use the Force Output Compare (FOC0) strobe bits in Normal mode. The OC0 Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM01:0 bits are not double buffered together with the compare value. Changing the COM01:0 bits will take effect immediately.

Compare Match Output Unit

The Compare Output mode (COM01:0) bits have two functions. The Waveform Generator uses the COM01:0 bits for defining the Output Compare (OC0) state at the next Compare Match. Also, the COM01:0 bits control the OC0 pin output source. Figure 37 shows a simplified schematic of the logic affected by the COM01:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port Control Registers (DDR and PORT) that are affected by the COM01:0 bits are shown. When referring to the OC0 state, the reference is for the internal OC0 Register, not the OC0 pin. If a System Reset occur, the OC0 Register is reset to “0”.

Figure 37. Compare Match Output Unit, Schematics



The general I/O port function is overridden by the output compare (OC0) from the Waveform Generator if either of the COM01:0 bits are set. However, the OC0 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 OC0 pin (DDR_OC0) must be set as output before the OC0 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 OC0 state before the output is enabled. Note that some COM01:0 bit settings are reserved for certain modes of operation. See “8-bit Timer/Counter Register Description” on page 91.

Compare Output Mode and Waveform Generation

The waveform generator uses the COM01:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM01:0 = 0 tells the Waveform Generator that no action on the OC0 Register is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Table 45 on page 92. For fast PWM mode, refer to Table 46 on page 92, and for phase correct PWM refer to Table 47 on page 92.

A change of the COM01:0 bits state will have effect at the first Compare Match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0 strobe bits.

Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the Waveform Generation mode (WGM01:0) and Compare Output mode (COM01:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM01:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM01:0 bits control whether the output should be set, cleared, or toggled at a Compare Match (See “Compare Match Output Unit” on page 84.).

For detailed timing information refer to Figure 41, Figure 42, Figure 43, and Figure 44 in “Timer/Counter Timing Diagrams” on page 89.

Normal Mode

The simplest mode of operation is the Normal mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

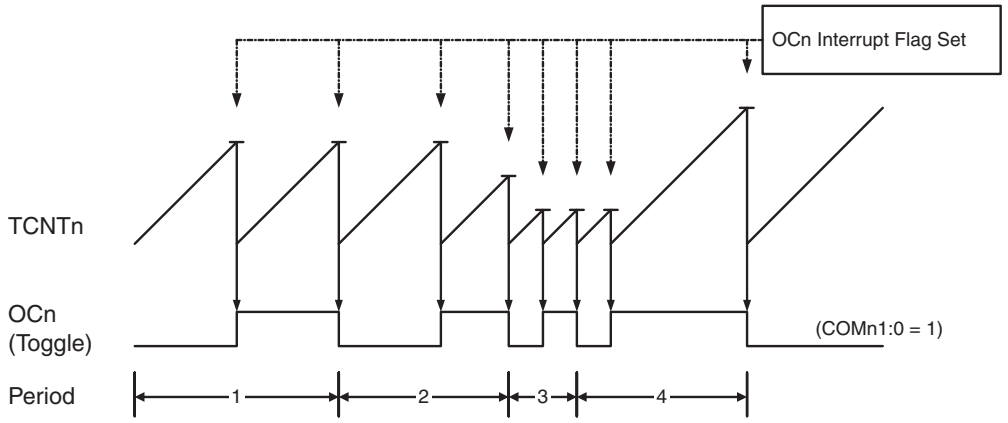
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM01:0 = 2), the OCR0 Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0. The OCR0 defines the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 38. The counter value (TCNT0) increases until a Compare Match occurs between TCNT0 and OCR0, and then counter (TCNT0) is cleared.

Figure 38. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0 Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM

when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR0 is lower than the current value of TCNT0, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the Compare Match can occur.

For generating a waveform output in CTC mode, the OC0 output can be set to toggle its logical level on each Compare Match by setting the Compare Output mode bits to toggle mode (COM01:0 = 1). The OC0 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0 is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

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

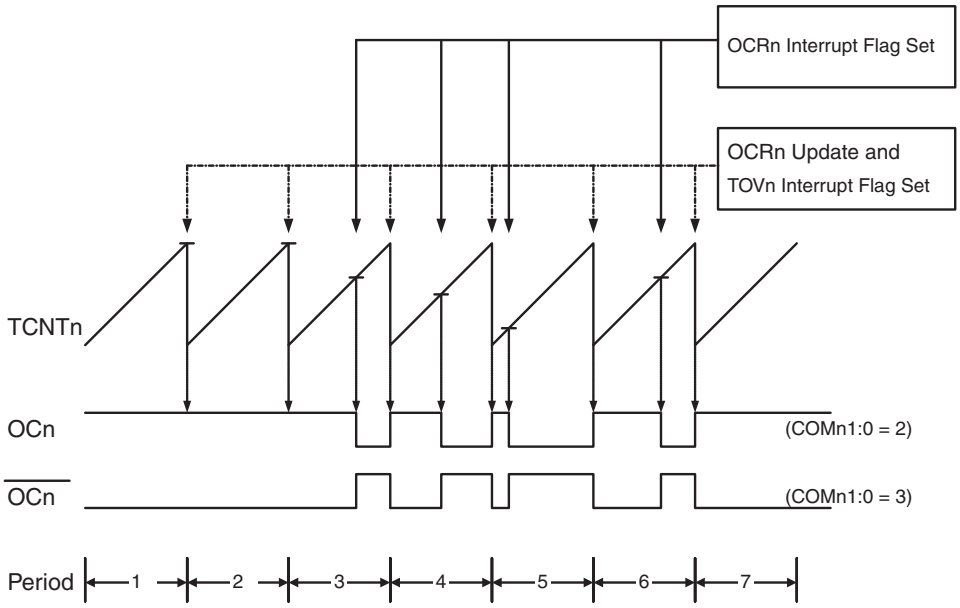
As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM01:0 = 3) 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 MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the Compare Match between TCNT0 and OCR0, and set at BOTTOM. In inverting Compare Output mode, the output is set on Compare Match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 39. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0 and TCNT0.

Figure 39. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches MAX. 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 OC0 pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM01:0 to 3 (See Table 46 on page 92). The actual OC0 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0 Register at the Compare Match between OCR0 and TCNT0, and clearing (or setting) the OC0 Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

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

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

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

The extreme values for the OCR0 Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0 is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0 equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM01:0 bits).

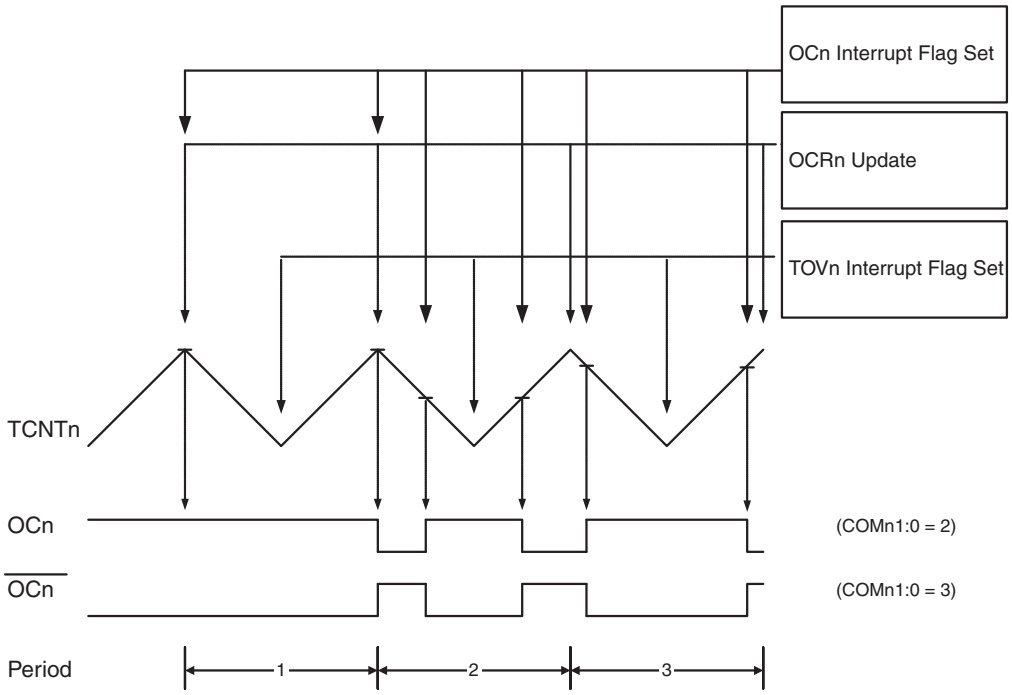
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0 to toggle its logical level on each Compare Match (COM01:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0 is set to zero. This feature is similar to the OC0 toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

Phase Correct PWM Mode

The phase correct PWM mode (WGM01:0 = 1) 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 MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the Compare Match between TCNT0 and OCR0 while upcounting, and set on the Compare Match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT0 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 40. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0 and TCNT0.

Figure 40. 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 OC0 pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM01:0 to 3 (See Table 47 on page 92). The actual OC0 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0 Register at the Compare Match between OCR0 and TCNT0 when the counter increments, and setting (or clearing) the OC0 Register at Compare Match between

OCR0 and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

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

The extreme values for the OCR0 Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0 is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 40 OCn has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

- OCR0 changes its value from MAX, like in Figure 40. When the OCR0 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 higher value than the one in OCR0, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

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 41 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 41. Timer/Counter Timing Diagram, no Prescaling

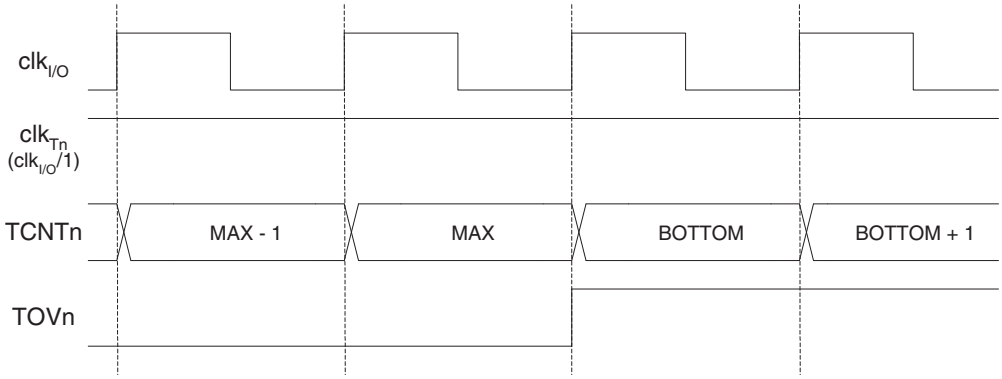


Figure 42 shows the same timing data, but with the prescaler enabled.

Figure 42. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

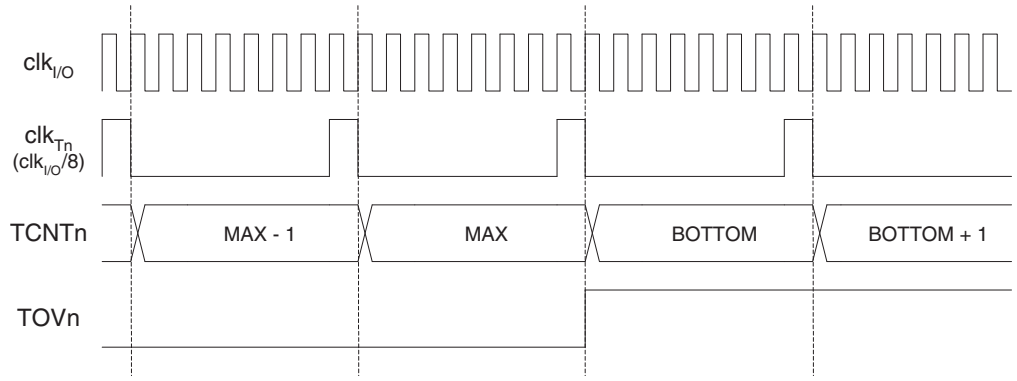


Figure 43 shows the setting of OCF0 in all modes except CTC mode.

Figure 43. Timer/Counter Timing Diagram, Setting of OCF0, with Prescaler ($f_{clk\_I/O}/8$)

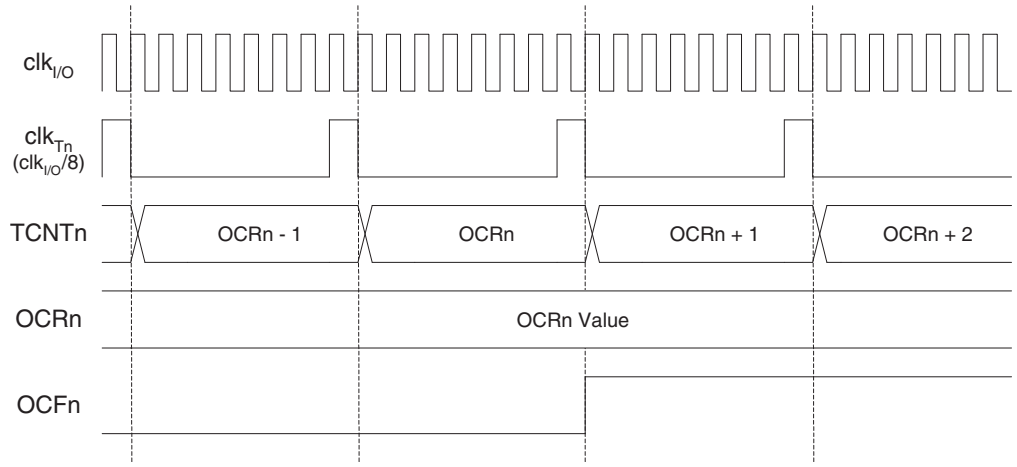
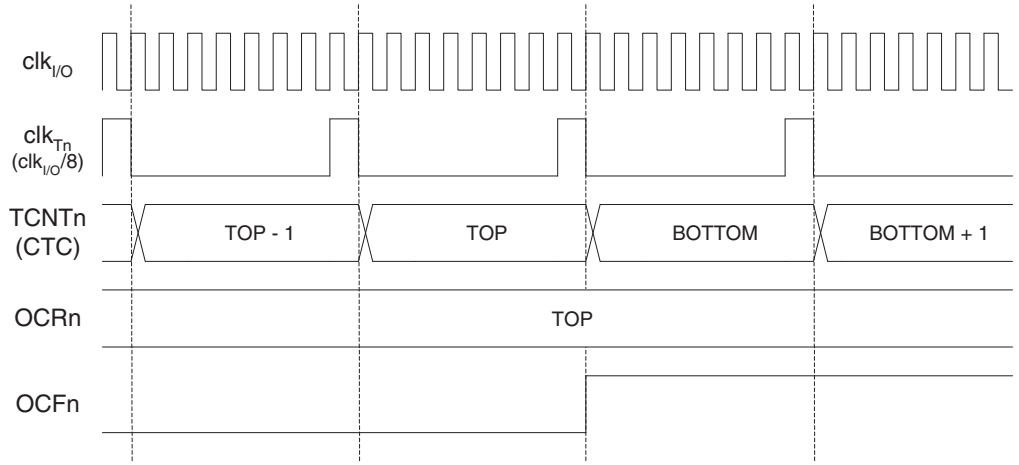


Figure 44 shows the setting of OCF0 and the clearing of TCNT0 in CTC mode.

Figure 44. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler ($f_{clk\_I/O}/8$)



8-bit Timer/Counter Register Description

Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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 – FOC0: Force Output Compare**

The FOC0 bit is only active when the WGM00 bit specifies a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0 is written when operating in PWM mode. When writing a logical one to the FOC0 bit, an immediate Compare Match is forced on the waveform generation unit. The OC0 output is changed according to its COM01:0 bits setting. Note that the FOC0 bit is implemented as a strobe. Therefore it is the value present in the COM01:0 bits that determines the effect of the forced compare.

A FOC0 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0 as TOP.

The FOC0 bit is always read as zero.

- **Bit 6, 3 – WGM01:0: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 44 and “Modes of Operation” on page 85.

Table 44. Waveform Generation Mode Bit Description[1]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

- **Bit 5:4 – COM01:0: Compare Match Output Mode**

These bits control the Output Compare pin (OC0) behavior. If one or both of the COM01:0 bits are set, the OC0 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 OC0 pin must be set in order to enable the output driver.

When OC0 is connected to the pin, the function of the COM01:0 bits depends on the WGM01:0 bit setting. Table 45 shows the COM01:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM).

Table 45. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on Compare Match.
1	0	Clear OC0 on Compare Match.
1	1	Set OC0 on Compare Match.

Table 46 shows the COM01:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 46. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on Compare Match, set OC0 at TOP (Non-Inverting).
1	1	Set OC0 on Compare Match, clear OC0 at TOP (Inverting).

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 86 for more details.

Table 47 shows the COM01:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.

Table 47. Compare Output Mode, Phase Correct PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on Compare Match when up-counting. Set OC0 on Compare Match when downcounting.
1	1	Set OC0 on Compare Match when up-counting. Clear OC0 on Compare Match when downcounting.

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Phase Correct PWM Mode” on page 88 for more details.

- **Bit 2:0 – CS02:0: Clock Select**

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

Table 48. Clock Select Bit Description

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

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

Timer/Counter Register – TCNT0

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

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0 Register.

Output Compare Register – OCR0

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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 contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0 pin.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	–	TICIE1	–	TOIE0	OCIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 0 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable**

When the OCIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OCF1B	–	ICF1	–	TOV0	OCF0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

The bit TOV0 is set (one) 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 (one), the Timer/Counter0 Overflow interrupt is executed. In phase correct PWM mode, this bit is set when Timer/Counter0 changes counting direction at $00.

- **Bit 0 – OCF0: Output Compare Flag 0**

The OCF0 bit is set (one) when a Compare Match occurs between the Timer/Counter0 and the data in OCR0 – Output Compare Register0. OCF0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0 (Timer/Counter0 Compare Match Interrupt Enable), and OCF0 are set (one), the Timer/Counter0 Compare Match Interrupt is executed.

Timer/Counter0 and Timer/Counter1 Prescalers

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

Internal Clock Source

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

Prescaler Reset

The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

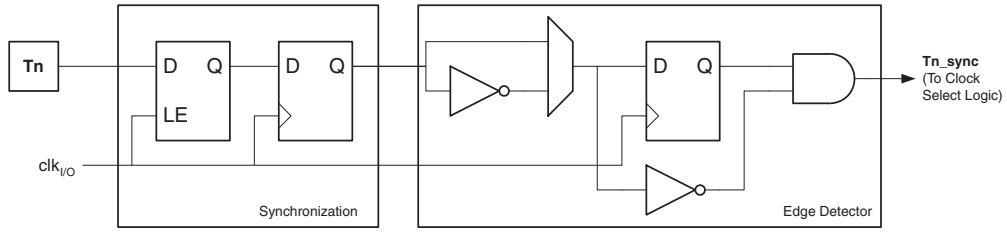
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A Prescaler Reset will affect the prescaler period for all Timer/Counters it is connected to.

External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 45 shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T1}/clk_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 45. T1/T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

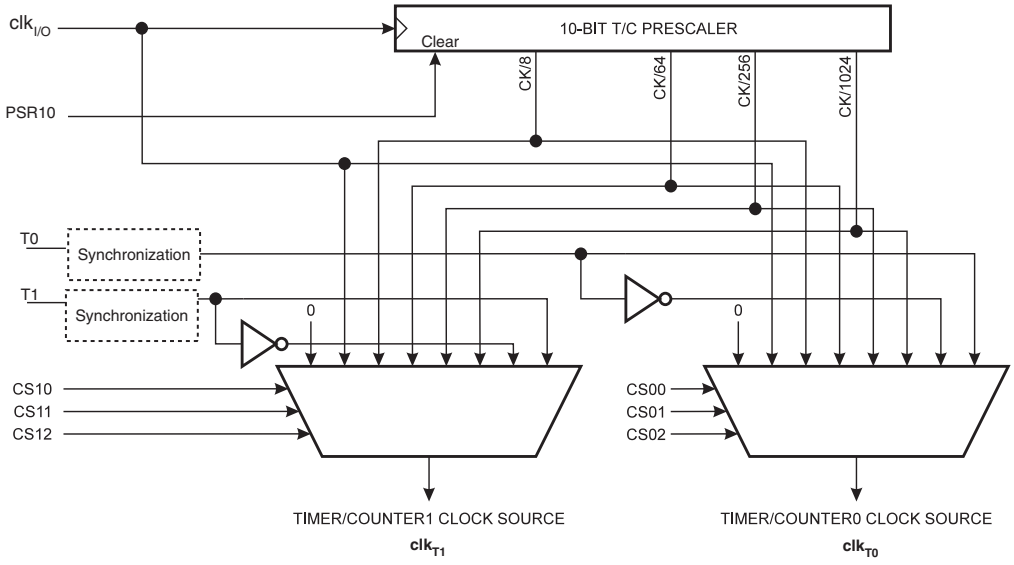
Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since

the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

Figure 46. Prescaler for Timer/Counter0 and Timer/Counter1[1]



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 45.

Special Function IO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	XMBK	XMM2	XMM1	XMM0	PUD	–	PSR10	SFIOR
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 0 – PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0**

When this bit is written to one, the Timer/Counter1 and Timer/Counter0 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers. This bit will always be read as zero.

16-bit Timer/Counter1

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

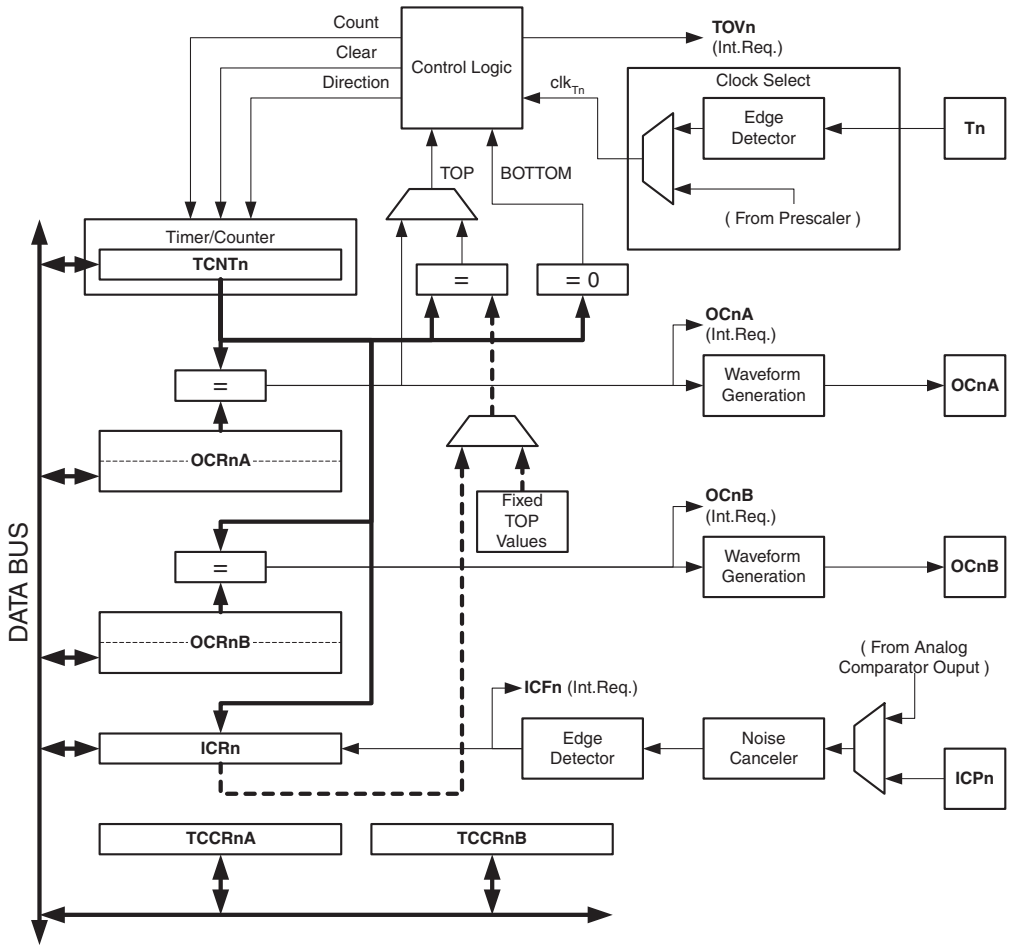
- **True 16-bit Design (i.e., allows 16-bit PWM)**
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Four Independent Interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

Overview

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

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 47. For the actual placement of I/O pins, refer to “Pin Configurations” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit location are listed in the “16-bit Timer/Counter Register Description” on page 119.

Figure 47. 16-bit Timer/Counter Block Diagram[1]



Note: 1. Refer to Figure 1 on page 2, Table 29 on page 67, and Table 35 on page 72 for Timer/Counter1 pin placement and description.

Registers

The *Timer/Counter* (TCNT1), *Output Compare Registers* (OCR1A/B), and *Input Capture Register* (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section “Accessing 16-bit Registers” on page 100. The *Timer/Counter Control Registers* (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the *Timer Interrupt Flag Register* (TIFR). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

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 time. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC1A/B). See “Output Compare Units” on page 106. The Compare Match event will

also set the Compare Match Flag (OCF1A/B) which can be used to generate an output compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture Pin (ICP1) or on the Analog Comparator pins (See “Analog Comparator” on page 164.) The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the 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 can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

Definitions

The following definitions are used extensively throughout the document:

Table 49. Definitions

BOTTOM	The counter reaches the *BOTTOM* when it becomes 0x0000.
MAX	The counter reaches its *MAX*imum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the *TOP* when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 Register. The assignment is dependent of the mode of operation.

Compatibility

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

- All 16-bit Timer/Counter related I/O Register address locations, including Timer Interrupt Registers.
- Bit locations inside all 16-bit Timer/Counter Registers, including Timer Interrupt Registers.
- Interrupt Vectors.

The following control bits have changed name, but have same functionality and register location:

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

The following bits are added to the 16-bit Timer/Counter Control Registers:

- FOC1A and FOC1B are added to TCCR1A.
- WGM13 is added to TCCR1B.

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

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 temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

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

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

Assembly Code Examples[1]

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

C Code Examples[1]

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

Note: 1. See “About 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 will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

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

Assembly Code Example[1]

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

C Code Example[1]

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

Note: 1. See “About 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 by using the same principle.

Assembly Code Example[1]

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

C Code Example[1]

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

Note: 1. See “About Code Examples” on page 7.

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

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.

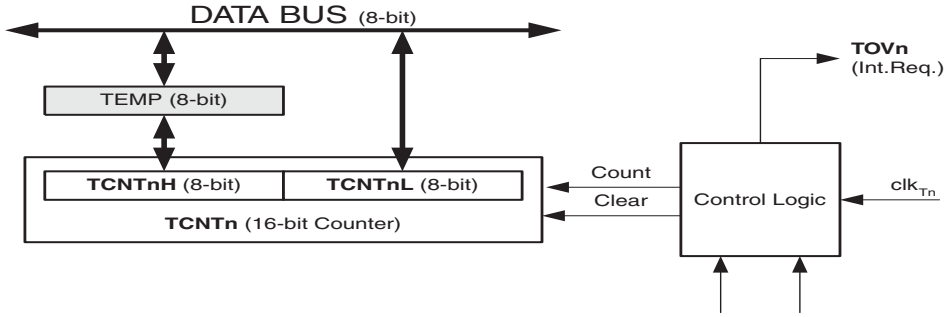
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter Control Register B* (TCCR1B). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 95.

Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 48 shows a block diagram of the counter and its surroundings.

Figure 48. Counter Unit Block Diagram



Signal description (internal signals):

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

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and *Counter Low* (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when 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 will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *Timer Clock* (clk_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, 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 (WGM13:0) 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 “Modes of Operation” on page 109.

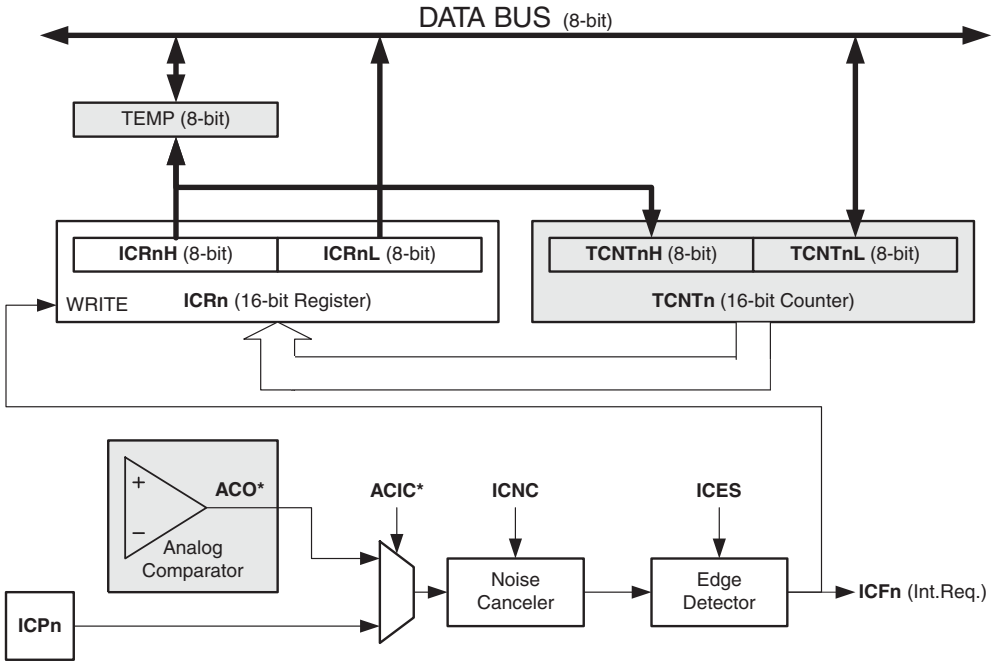
The *Timer/Counter Overflow* (TOV1) Flag is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

Input Capture Unit

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

The Input Capture unit is illustrated by the block diagram shown in Figure 49. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The small “n” in register and bit names indicates the Timer/Counter number.

Figure 49. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP1), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the *Input Capture Register* (ICR1). The *Input Capture Flag* (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (TICIE1 = 1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 Flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the *Input Capture Register* (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read the high

byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In these cases the *Waveform Generation mode* (WGM13:0) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 100.

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* (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP1) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 45 on page 95). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

Noise Canceler

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

The noise canceler is enabled by setting the *Input Capture Noise Canceler* (ICNC1) bit in *Timer/Counter Control Register B* (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

Using the Input Capture Unit

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

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

Using the Input Capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For

measuring frequency only, the clearing of the ICF1 Flag is not required (if an interrupt handler is used).

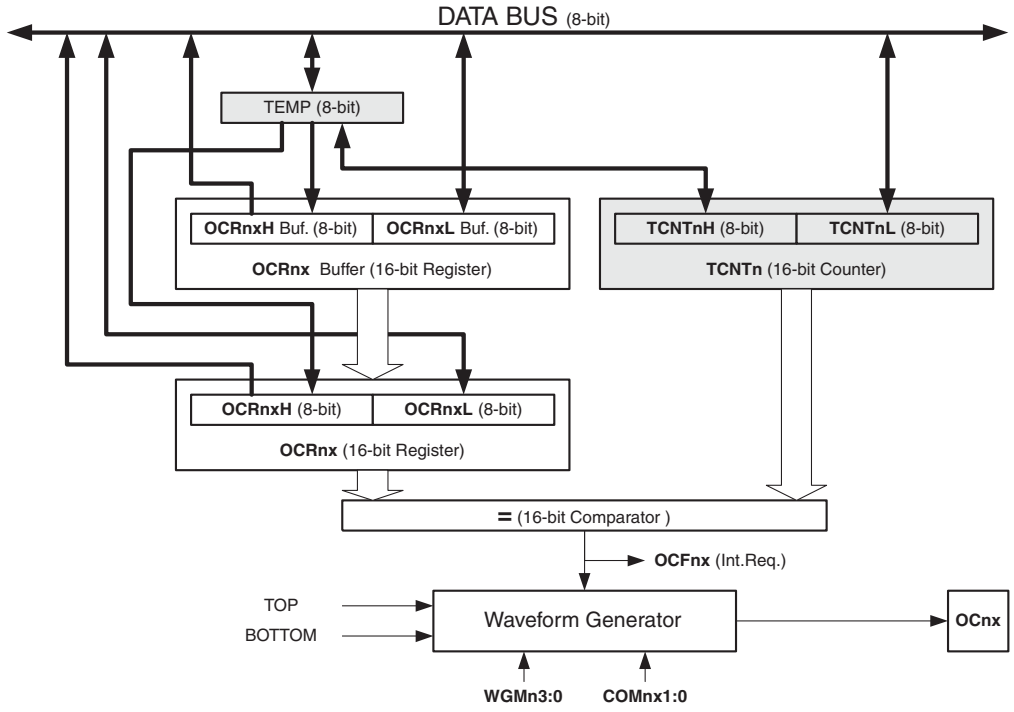
Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the *Output Compare Register* (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the *Output Compare Flag* (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the Output Compare Flag generates an output compare interrupt. The OCF1x Flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the *Waveform Generation mode* (WGM13:0) bits and *Compare Output mode* (COM1x1:0) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation. See “Modes of Operation” on page 109.

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

Figure 50 shows a block diagram of the output compare unit. The small “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter1), 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 50. Output Compare Unit, Block Diagram



The OCR1x Register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x Compare Register to either TOP or BOTTOM of the counting

sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (Buffer or Compare) Register is 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 when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR1xH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the low byte (OCR1xL) is written to the lower eight bits, the high byte will be copied into the upper eight bits of either the OCR1x Buffer or OCR1x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 100.

Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the *Force Output Compare* (FOC1x) bit. Forcing Compare Match will not set the OCF1x Flag or reload/clear the timer, but the OC1x pin will be updated as if a real Compare Match had occurred (the COM11:0 bits settings define whether the OC1x pin is set, cleared or toggled).

Compare Match Blocking by TCNT1 Write

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

Using the Output Compare Unit

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

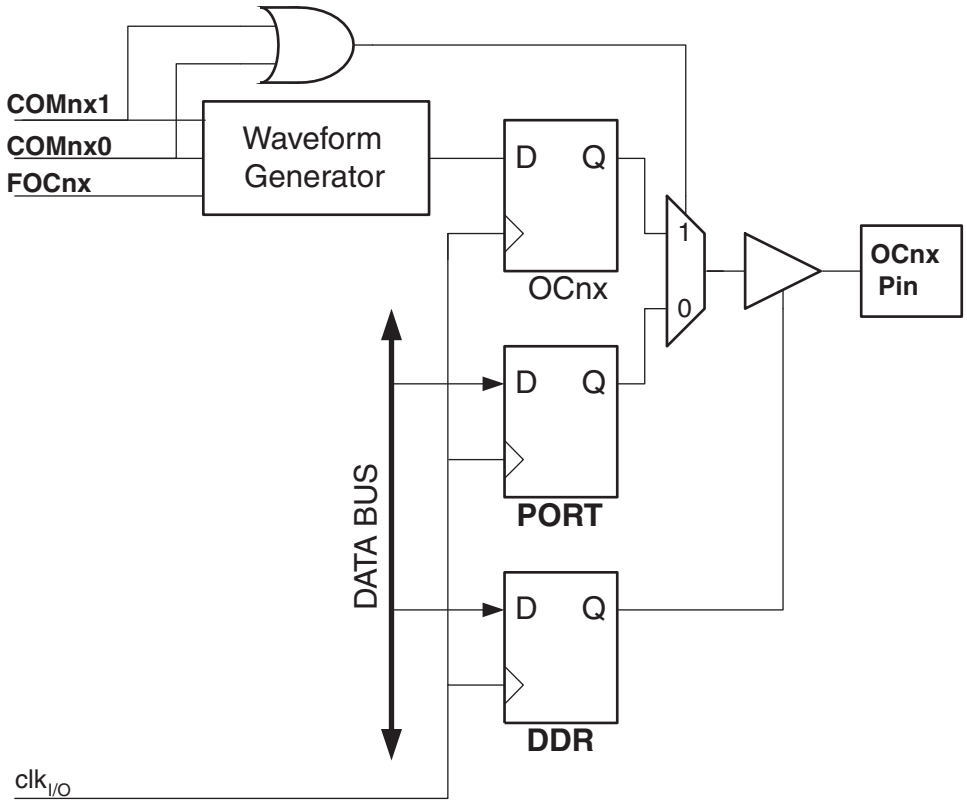
The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between Waveform Generation modes.

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

Compare Match Output Unit

The *Compare Output mode* (COM1x1:0) bits have two functions. The Waveform Generator uses the COM1x1:0 bits for defining the Output Compare (OC1x) state at the next Compare Match. Secondly the COM1x1:0 bits control the OC1x pin output source. Figure 51 shows a simplified schematic of the logic affected by the COM1x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x Register, not the OC1x pin. If a System Reset occur, the OC1x Register is reset to “0”.

Figure 51. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x1:0 bits are set. However, the OC1x pin direction (input or output) is still controlled by the *Data Direction Register* (DDR) for the port pin. The Data Direction Register bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the waveform generation mode, but there are some exceptions. Refer to Table 50, Table 51, and Table 52 for details.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. See “16-bit Timer/Counter Register Description” on page 119.

The COM1x1:0 bits have no effect on the Input Capture unit.

Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM1x1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the Waveform Generator that no action on the OC1x Register Is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Table 50 on page 119. For fast PWM mode refer to Table 51 on page 119, and for phase correct and phase and frequency correct PWM refer to Table 52 on page 120.

A change of the COM1x1:0 bits state will have effect at the first Compare Match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output mode* (COM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a Compare Match. See “Compare Match Output Unit” on page 108.

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 117.

Normal Mode

The simplest mode of operation is the *Normal* mode (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

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

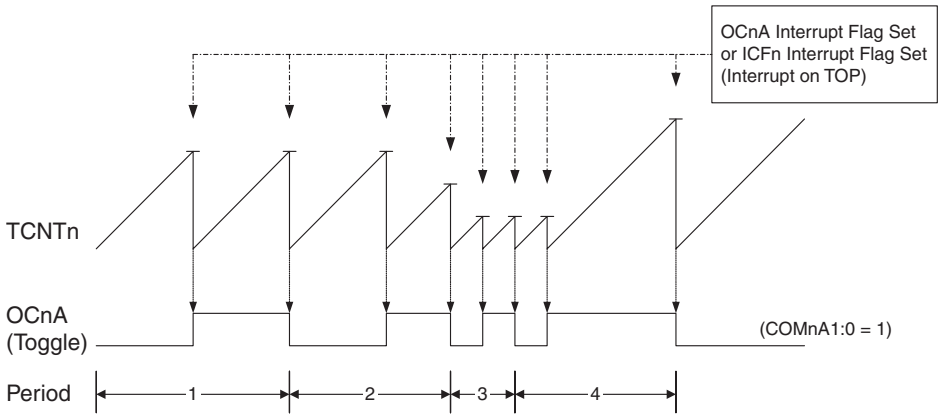
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In *clear timer on compare* or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 52. The counter value (TCNT1) increases until a Compare Match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 52. CTC Mode, Timing Diagram



An interrupt can be generated at 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 none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the Compare Match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

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

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

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

As for the normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

Fast PWM Mode

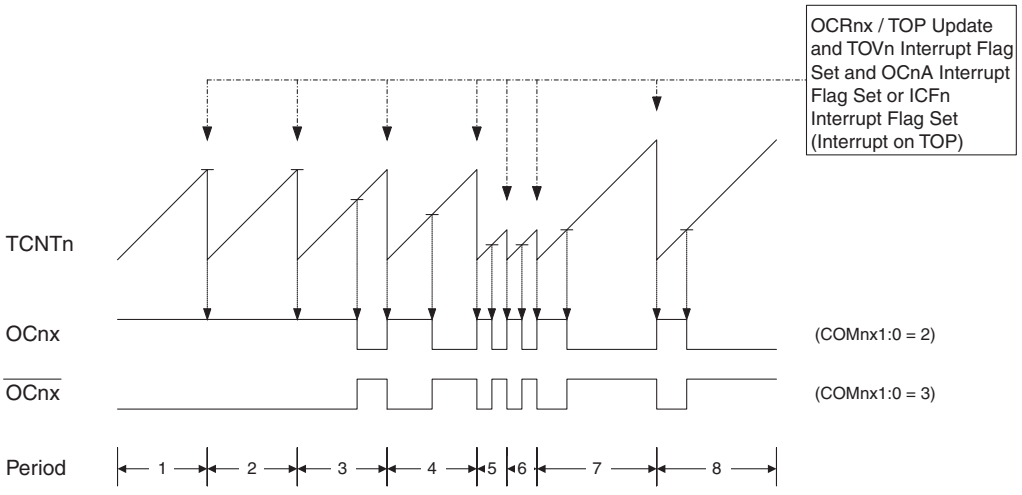
The *fast Pulse Width Modulation* or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the output compare (OC1x) is set on the Compare Match between TCNT1 and OCR1x, and cleared at BOTTOM. In inverting Compare Output mode output is cleared on Compare Match and set at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either 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 (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 53. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent Compare Matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a Compare Match occurs.

Figure 53. 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 or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a Compare Match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x Registers are written.

The procedure for updating ICR1 differs from 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 will then be that the counter will miss the Compare Match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the Compare Match can occur. The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle 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 COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table on page 119). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the Compare Match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

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

The extreme values for the 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 will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each Compare Match (COM1A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM1 = 15). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

Phase Correct PWM Mode

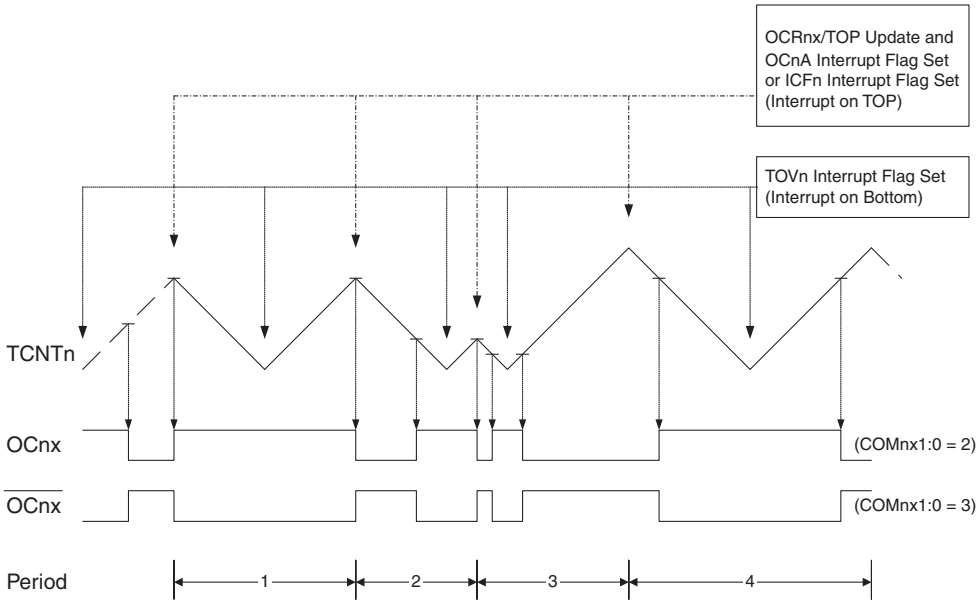
The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting compare output mode, the Output Compare (OC1x) is cleared on the Compare Match between TCNT1 and OCR1x while upcounting, and set on the Compare Match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either 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 either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 54. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent Compare Matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a Compare Match occurs.

Figure 54. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 Flag is set accordingly 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 higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a Compare Match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x Registers are written. As the third period shown in Figure 54 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR1x Register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

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

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table 52 on page 120). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the Compare Match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at Compare Match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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 will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM1 = 11) and COM1A1:0 = 1, the OC1A Output will toggle with a 50% duty cycle.

Phase and Frequency Correct PWM Mode

The *phase and frequency correct Pulse Width Modulation,* or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the Compare Match between TCNT1 and OCR1x while upcounting, and set on the Compare Match while downcounting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

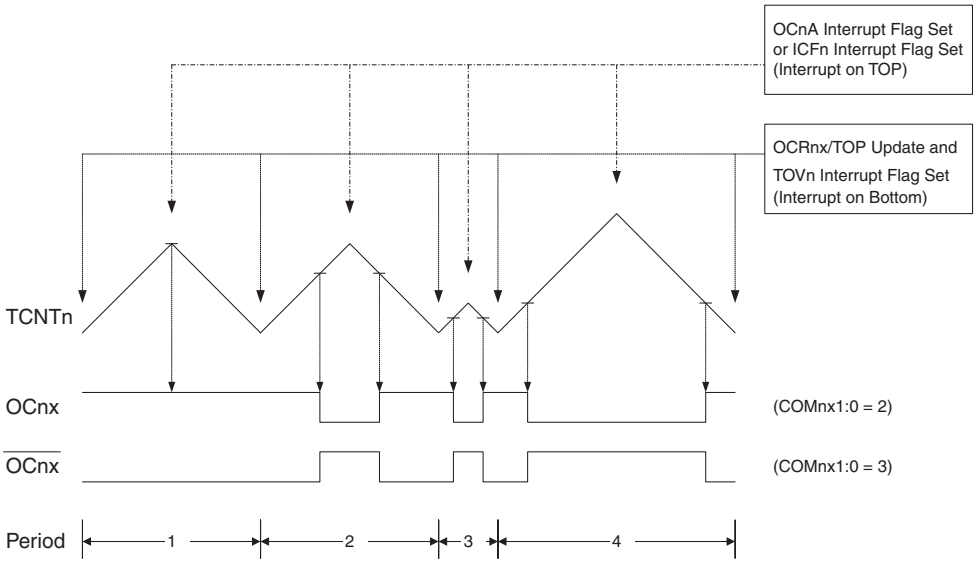
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see Figure 54 and Figure 55).

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 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 55. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent Compare Matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a Compare Match occurs.

Figure 55. 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 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 of 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 Figure 55 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

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

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3 (See Table 1 on page 120). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the Compare Match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at Compare Match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

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

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 phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM1 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

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 56 shows a timing diagram for the setting of OCF1x.

Figure 56. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling

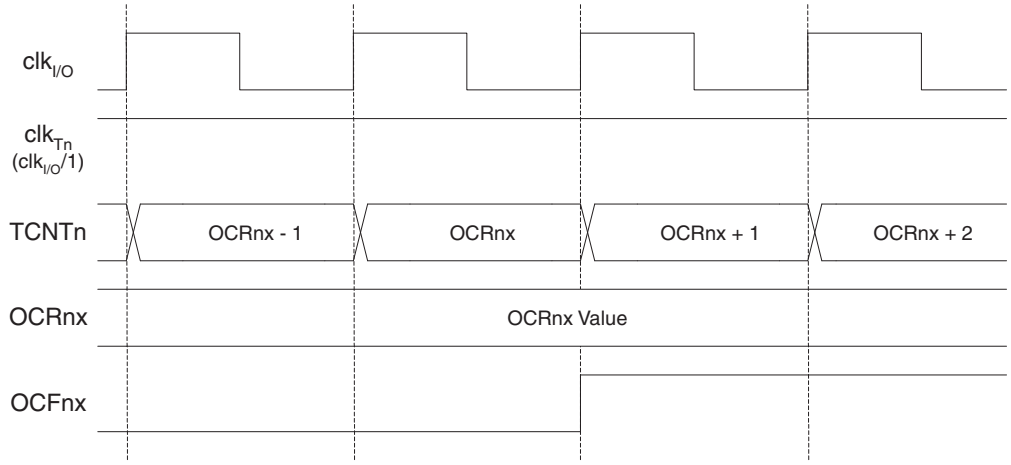


Figure 57 shows the same timing data, but with the prescaler enabled.

Figure 57. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}/8$)

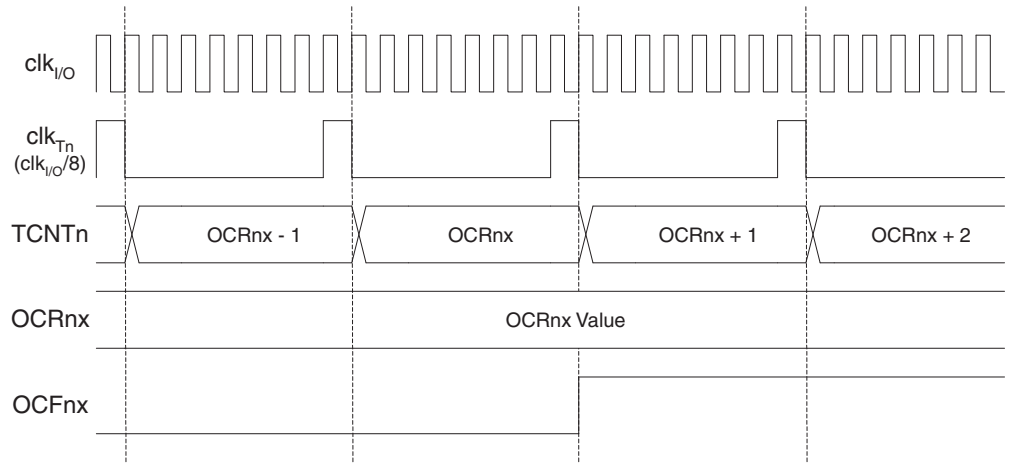


Figure 58 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

Figure 58. Timer/Counter Timing Diagram, No Prescaling

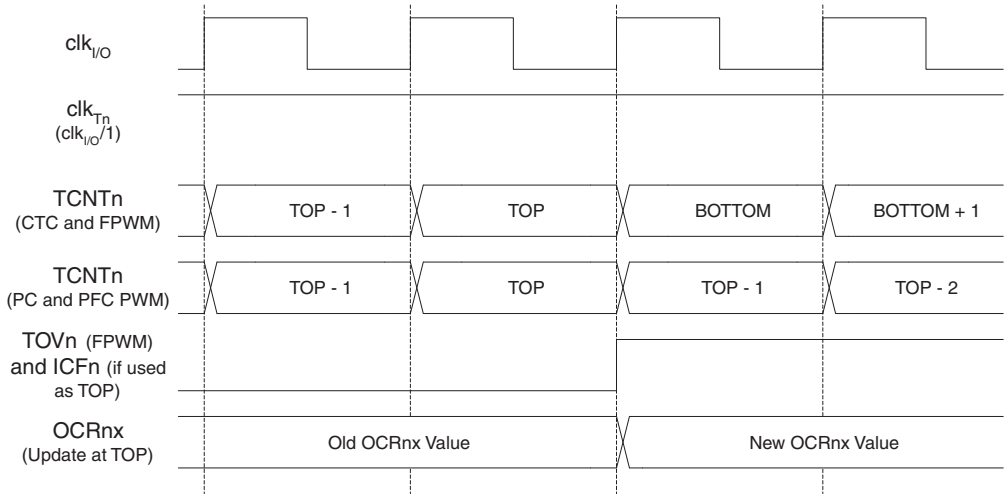
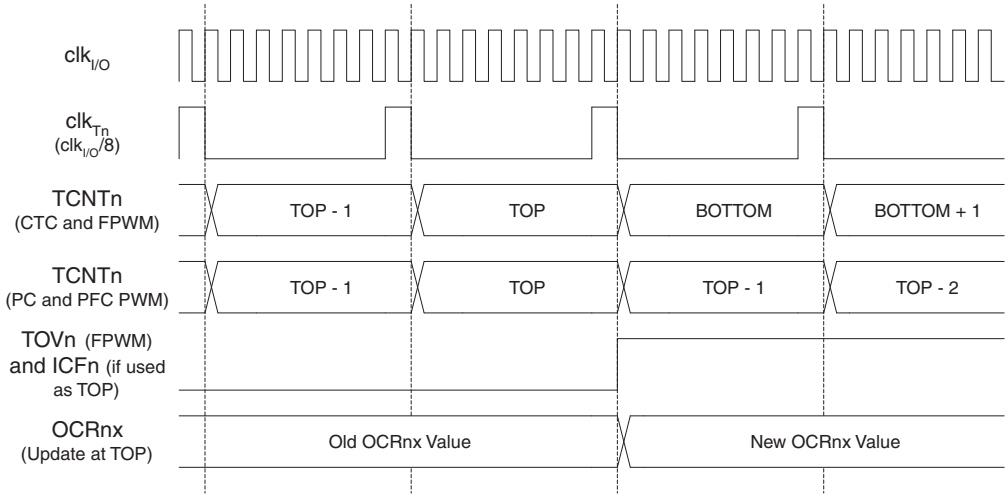


Figure 59 shows the same timing data, but with the prescaler enabled.

Figure 59. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



16-bit Timer/Counter Register Description

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM1A1:0: Compare Output Mode for Channel A**
- **Bit 5:4 – COM1B1:0: Compare Output Mode for Channel B**

The COM1A1:0 and COM1B1:0 control the Output Compare pins (OC1A and OC1B respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the *Data Direction Register* (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent of the WGM13:0 bits setting. Table 50 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 50. 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 51 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 51. Compare Output Mode, Fast PWM(1)

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (Normal port operation). For all other WGM1 setting, Normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at TOP (Non-Inverting).
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at TOP (Inverting).

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 TOP. See “Fast PWM Mode” on page 111. for more details.

Table 52 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 52. 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 = 9 or 11: Toggle OC1A on Compare Match, OC1B disconnected (Normal port operation). For all other WGM1 setting, Normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See “Phase Correct PWM Mode” on page 113. for more details.

- **Bit 3 – FOC1A: Force Output Compare for Channel A**
- **Bit 2 – FOC1B: Force Output Compare for Channel B**

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

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

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

Combined with the WGM13:2 bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 53. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. See “Modes of Operation” on page 109.

Table 53. Waveform Generation Mode Bit Description[1]

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

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

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	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 one) 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) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

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

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

- **Bit 5: Reserved Bit**

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

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

See TCCR1A Register description.

- **Bit 2:0 – CS12:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 56 and Figure 57.

Table 54. Clock Select Bit Description

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

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

Timer/Counter1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	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, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 100.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a Compare Match between TCNT1 and one of the OCR1x Registers.

Writing to the TCNT1 Register blocks (removes) the Compare Match on the following timer clock for all compare units.

Output Compare Register 1 A – OCR1AH and OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	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	

Output Compare Register 1 B – OCR1BH and OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	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-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 100.

Input Capture Register 1 – ICR1H and ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	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-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 100.

Timer/Counter Interrupt Mask Register – TIMSK[1]

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	OCIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Note: 1. This register contains interrupt control bits for several Timer/Counters, but only Timer1 bits are described in this section. The remaining bits are described in their respective timer sections.

- **Bit 7 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 overflow interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 54) 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 one, 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 “Interrupts” on page 54) 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 one, 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 “Interrupts” on page 54) is executed when the OCF1B Flag, located in TIFR, is set.

- **Bit 3 – TICIE1: Timer/Counter1, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 54) is executed when the ICF1 Flag, located in TIFR, is set.

Timer/Counter Interrupt Flag Register – TIFR[1]

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OC1FB	–	ICF1	–	TOV0	OCF0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Note: 1. This register contains flag bits for several Timer/Counters, but only Timer1 bits are described in this section. The remaining bits are described in their respective timer sections.

- **Bit 7 – TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bits setting. In Normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to Table 53 on page 121 for the TOV1 Flag behavior when using another WGM13:0 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 – 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 (FOC1A) strobe will not set the OCF1A Flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 5 – 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 (FOC1B) strobe will not set the OCF1B Flag.

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

- **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 WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value.

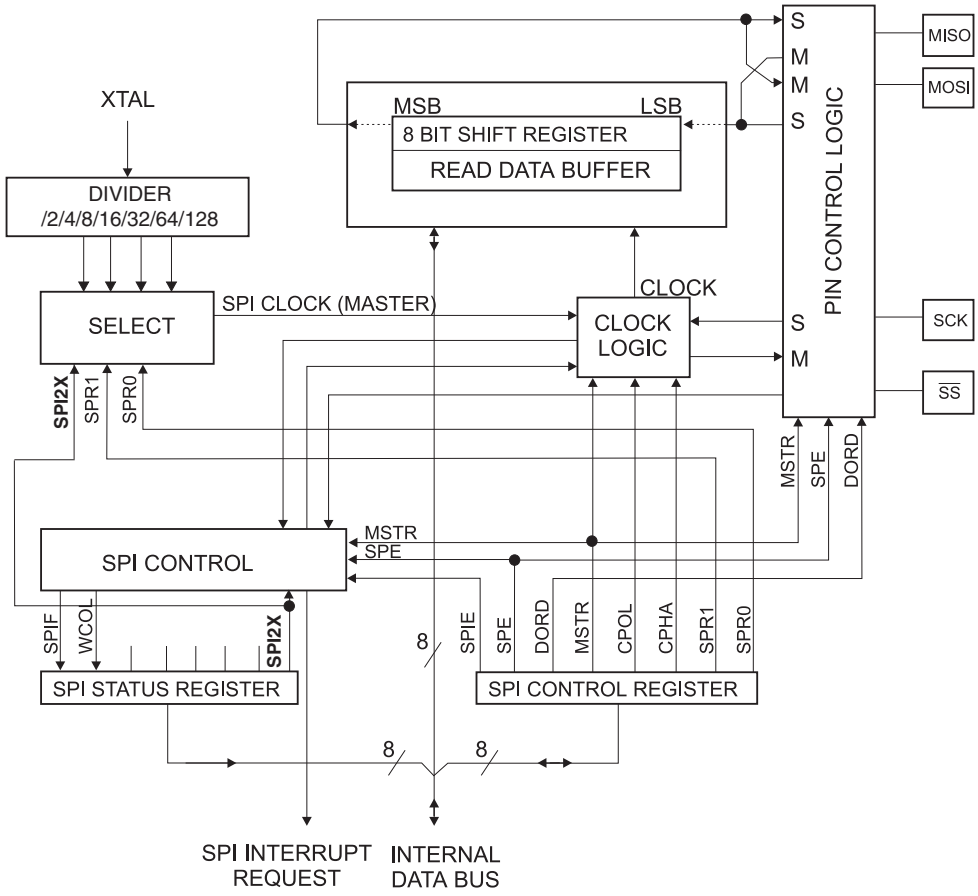
ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega8515 and peripheral devices or between several AVR devices. The ATmega8515 SPI includes the following features:

- **Full Duplex, 3-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

Figure 60. SPI Block Diagram[1]



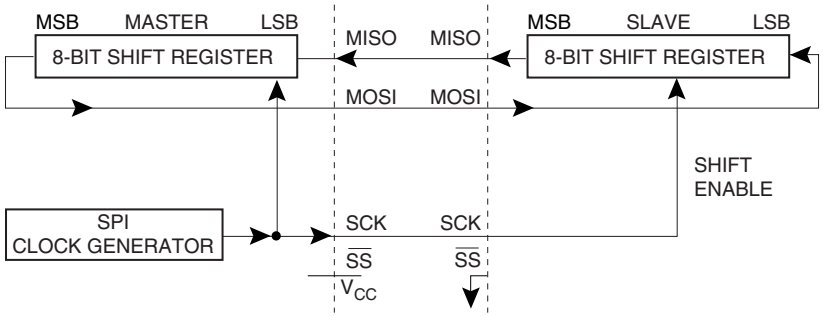
Note: 1. Refer to Figure 1 on page 2, and Table 29 on page 67 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 61. The system consists of two Shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective Shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the 8 bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 61. SPI Master-Slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be:

Low periods: Longer than 2 CPU clock cycles.

High periods: Longer than 2 CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 55. For more details on automatic port overrides, refer to “Alternate Port Functions” on page 64.

Table 55. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input

Table 55. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See “Alternate Functions Of Port B” on page 67 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. For example, if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.

Assembly Code Example[1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi  r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out  DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi  r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out  SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out  SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  sbis SPSR,SPIF
  rjmp Wait_Transmit
  ret
```

C Code Example[1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
    ;
}
```

Note: 1. See “About Code Examples” on page 7.

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example[1]

```
SPI_SlaveInit:
  ; Set MISO output, all others input
  ldi  r17,(1<<DD_MISO)
  out  DDR_SPI,r17
  ; Enable SPI
  ldi  r17,(1<<SPE)
  out  SPCR,r17
  ret

SPI_SlaveReceive:
  ; Wait for reception complete
  sbis SPSR,SPIF
  rjmp SPI_SlaveReceive
  ; Read received data and return
  in   r16,SPDR
  ret
```

C Code Example[1]

```
void SPI_SlaveInit(void)
{
  /* Set MISO output, all others input */
  DDR_SPI = (1<<DD_MISO);
  /* Enable SPI */
  SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
  /* Wait for reception complete */
  while(!(SPSR & (1<<SPIF)))
    ;
  /* Return data register */
  return SPDR;
}
```

Note: 1. See “About Code Examples” on page 7.

$\overline{SS}$ Pin Functionality

Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the Slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI Slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another Master selecting the SPI as a Slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a Slave Select, it must be set by the user to re-enable SPI Master mode.

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the Global Interrupt Enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will

be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 62 and Figure 63 for an example. The CPOL functionality is summarized below:

Table 56. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 62 and Figure 63 for an example. The CPHA functionality is summarized below:

Table 57. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the following table:

Table 58. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL: Write COLlision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8515 and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 58). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega8515 is also used for Program memory and EEPROM downloading or uploading. See page 193 for Serial Programming and verification.

SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
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	Undefined

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 62 and Figure 63. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 56 and Table 57, as done below:

Table 59. CPOL and CPHA Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)	0
CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)	1
CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)	2
CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)	3

Figure 62. SPI Transfer Format with CPHA = 0

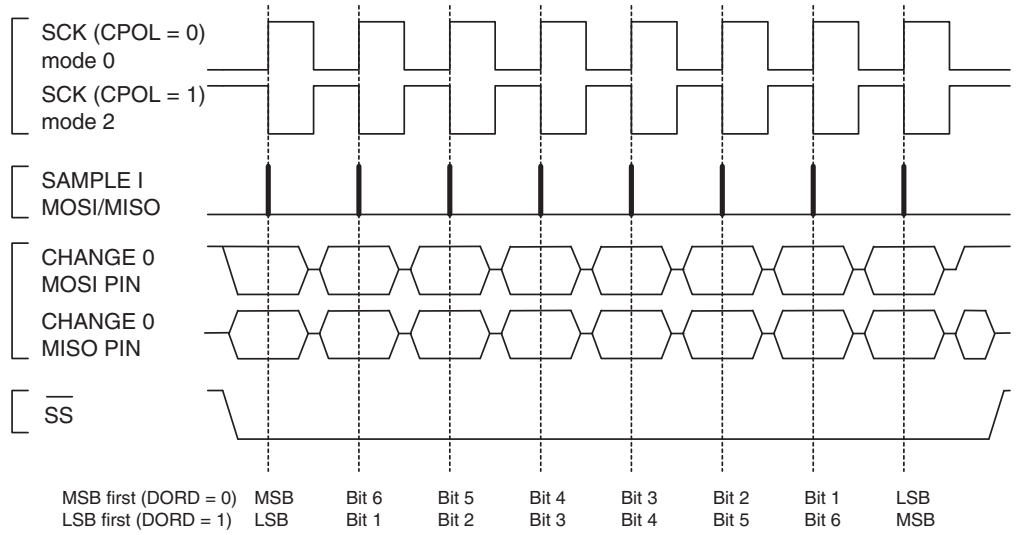
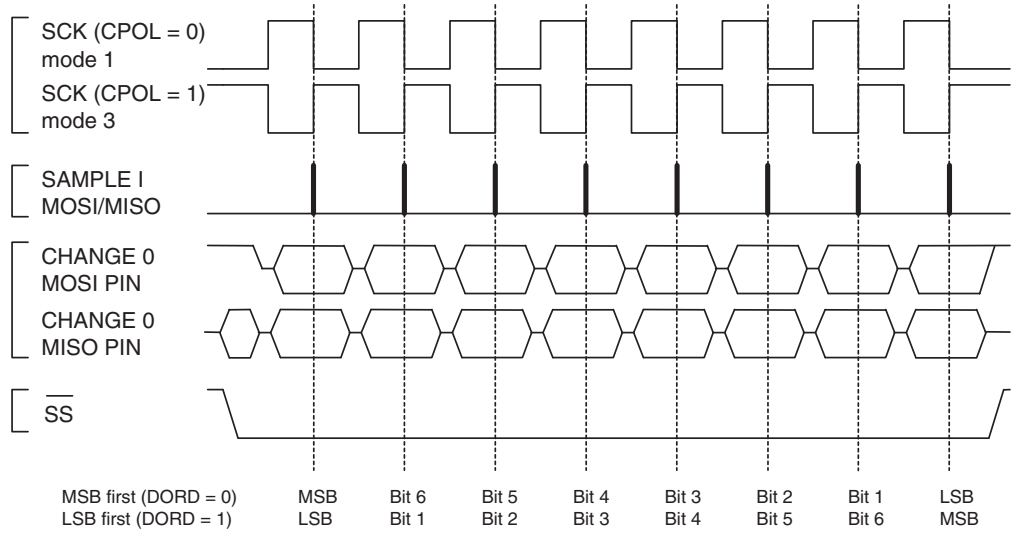


Figure 63. SPI Transfer Format with CPHA = 1



USART

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- **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**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

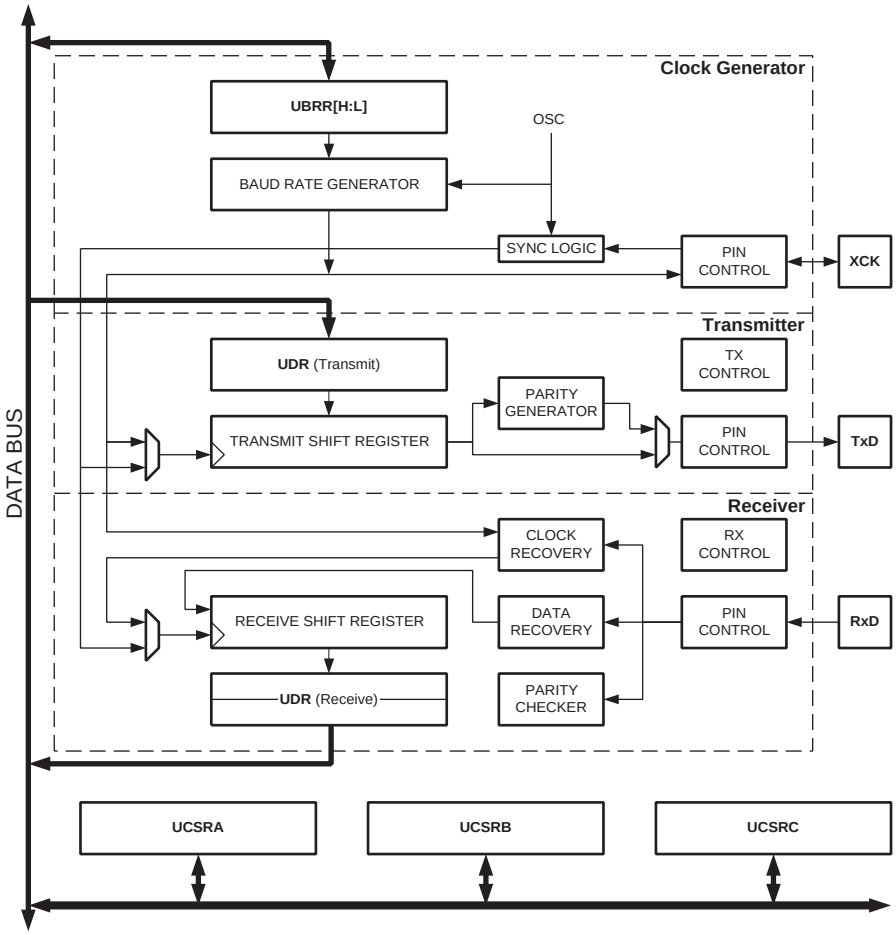
Single USART

The ATmega8515 has one USART. The functionality for the USART is described below.

Note that in AT90S4414/8515 compatibility mode, the double buffering of the USART Receive Register is disabled. For details, see “AVR USART vs. AVR UART – Compatibility” on page 137.

A simplified block diagram of the USART Transmitter is shown in Figure 64. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 64. USART Block Diagram[1]



Note: 1. Refer to Figure 1 on page 2, Table 37 on page 73, and Table 31 on page 69 for USART pin placement.

The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCK (Transfer Clock) pin is only used by Synchronous Transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any 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 (UDR). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun and Parity Errors.

AVR USART vs. AVR UART – Compatibility

The USART is fully compatible with the AVR UART regarding:

- Bit locations inside all USART Registers
- Baud Rate Generation
- Transmitter Operation
- Transmit Buffer Functionality
- Receiver Operation

However, the receive buffering has two improvements that will affect the compatibility in some special cases:

- A second Buffer Register has been added. The two Buffer Registers operate as a circular FIFO buffer. Therefore the UDR must only be read once for each incoming data. More important is the fact that the Error Flags (FE and DOR) and the ninth data bit (RXB8) are buffered with the data in the receive buffer. Therefore the status bits must always be read before the UDR Register is read. Otherwise the error status will be lost since the buffer state is lost.
- The Receiver Shift Register can now act as a third buffer level. This is done by allowing the received data to remain in the serial Shift Register (see Figure 64) if the Buffer Registers are full, until a new start bit is detected. The USART is therefore more resistant to Data OverRun (DOR) error conditions.

The following control bits have changed name, but have same functionality and register location:

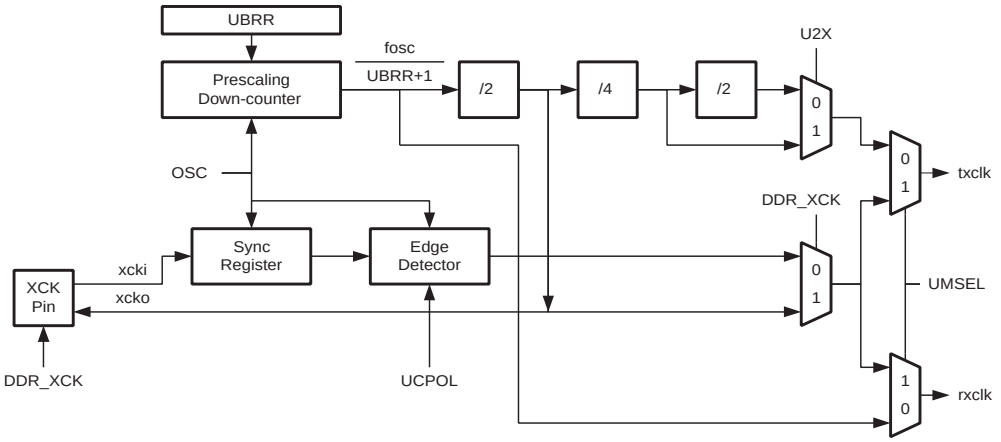
- CHR9 is changed to UCSZ2
- OR is changed to DOR

Clock Generation

The clock generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal asynchronous, Double Speed asynchronous, Master synchronous and Slave synchronous mode. The UMSEL bit in USART Control and Status Register C (UCSRC) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2X found in the UCSRA Register. When using Synchronous mode (UMSEL = 1), the Data Direction Register for the XCK pin (DDR_XCK) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCK pin is only active when using Synchronous mode.

Figure 65 shows a block diagram of the clock generation logic.

Figure 65. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock. (Internal Signal)

rxclk Receiver base clock. (Internal Signal)

xcki Input from XCK pin (internal Signal). Used for synchronous slave operation.

xcko Clock output to XCK pin (Internal Signal). Used for synchronous master operation.

fosc XTAL pin frequency (System Clock).

Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 65.

The USART Baud Rate Register (UBRR) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (fosc), is loaded with the UBRR value each time the counter has counted down to zero or when the UBRRL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= fosc/(UBRR+1)). The Transmitter divides the baud rate generator clock output by 2, 8, or 16 depending on mode. 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 mode set by the state of the UMSEL, U2X, and DDR_XCK bits.

Table 60 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.

Table 60. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{OSC} System Oscillator clock frequency

UBRR Contents of the UBRRH and UBRRL Registers, (0-4095)

Some examples of UBRR values for some system clock frequencies are found in Table 68 (see page 160).

Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 65 for details.

External clock input from the XCK pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and Receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK clock frequency is limited by the following equation:

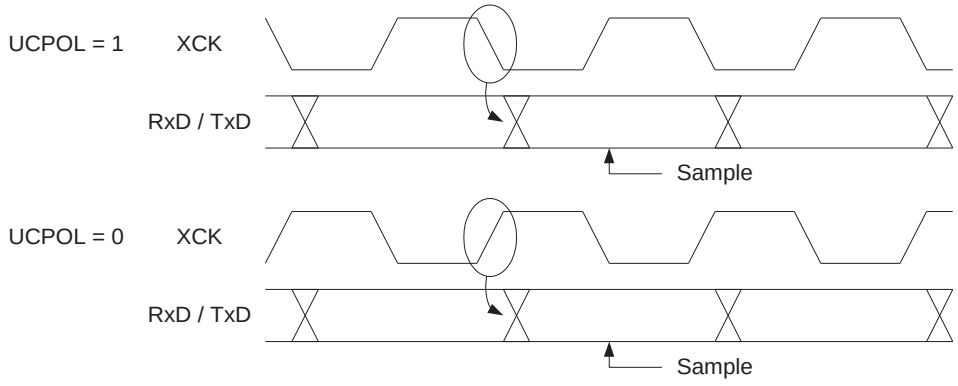
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCK pin will be 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 RxD) is sampled at the opposite XCK clock edge of the edge the data output (TxD) is changed.

Figure 66. Synchronous Mode XCK Timing.



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 66 shows, when UCPOL is zero the data will be changed at rising XCK edge and sampled at falling XCK edge. If UCPOL is set, the data will be changed at falling XCK edge and sampled at rising XCK edge.

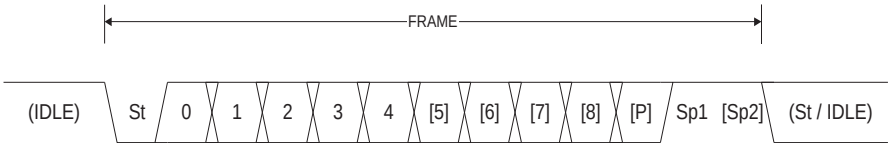
Frame Formats

A serial frame is defined to be 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:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 67 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 67. Frame Formats



St Start bit, always low

(n) Data bits (0 to 8)

P Parity bit. Can be odd or even

Sp Stop bit, always high

IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS) bit. The Receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

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::

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} Parity bit using even parity

P_{odd} Parity bit using odd parity

d_n Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

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 enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXC Flag must be cleared before each transmission (before UDR 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 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. When the function writes to the UCSRC Register, the URSEL bit (MSB) must be set due to the sharing of I/O location by UBRRH and UCSRC.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out  UBRRH, r17
  out  UBRRL, r16
  ; Enable receiver and transmitter
  ldi  r16, (1<<RXEN)|(1<<TXEN)
  out  UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi  r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
  out  UCSRC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Note: 1. See “About Code Examples” on page 7.

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. 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.

Data Transmission – The USART Transmitter

The USART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB Register. When the Transmitter is enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the Transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

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 UDR I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register 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 will transfer one complete frame at the rate given by the Baud Register, U2X bit or by XCK depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the *Data Register Empty* (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR 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 UCSRA,UDRE
  rjmp USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out  UDR,r16
  ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Note: 1. See "About Code Examples" on page 7.

The function simply waits for the transmit buffer to be empty by checking the UDRE 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.

Sending Frames with 9 Data Bits

If 9-bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the low byte of the character is written to UDR. 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)

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSRA,UDRE
  rjmp USART_Transmit
  ; Copy ninth bit from r17 to TXB8
  cbi  UCSRB,TXB8
  sbrc r17,0
  sbi  UCSRB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out  UDR,r16
  ret
```

C Code Example(1)

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Copy ninth bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. For example, only the TXB8 bit of the UCSRB Register is used after initialization.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

Transmitter Flags and Interrupts

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) Flag 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 into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRA Register.

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When

interrupt-driven data transmission is used, the Data Register Empty Interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register Empty Interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC) Flag bit is set one 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 TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like 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 (TXCIE) bit in UCSRB is set, the USART Transmit Complete Interrupt will be executed when the TXC Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC Flag, this is done automatically when the interrupt is executed.

Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the Transmitter Control Logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

Disabling the Transmitter

The disabling of the Transmitter (setting the TXEN to zero) will 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 will no longer override the TxD pin.

Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register to one. When the Receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

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 will be sampled at the baud rate or XCK clock, and shifted into the receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received (i.e., a complete serial frame is present in the Receive Shift Register), the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. 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 UCSRA, RXC
  rjmp USART_Receive
  ; Get and return received data from buffer
  in   r16, UDR
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get and return received data from buffer */
  return UDR;
}
```

Note: 1. See "About Code Examples" on page 7.

The function simply waits for data to be present in the receive buffer by checking the RXC Flag, before reading the buffer and returning the value.

Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB before reading the low bits from the UDR. This rule applies to the FE, DOR, and PE Status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR, and PE bits, which all are stored in the FIFO, will change.

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 UCSRA, RXC
  rjmp USART_Receive
  ; Get status and ninth bit, then data from buffer
  in   r18, UCSRA
  in   r17, UCSRB
  in   r16, UDR
  ; If error, return -1
  andi r18,(1<<FE)|(1<<DOR)|(1<<PE)
  breq USART_ReceiveNoError
  ldi  r17, HIGH(-1)
  ldi  r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the ninth 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 ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get status and ninth bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<PE) )
    return -1;
  /* Filter the ninth bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Note: 1. See “About Code Examples” on page 7.

The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

Receive Compete Flag and Interrupt

The USART Receiver has one flag that indicates the Receiver state.

The Receive Complete (RXC) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete Interrupt will be executed as long as the RXC 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 UDR in order to clear the RXC Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (PE). All can be accessed by reading UCSRA. Common for the error flags is that they are 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 UCSRA must be read before the receive buffer (UDR), since reading the UDR I/O location changes the buffer read location. Another equality for the error flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRA is written for upward compatibility of future USART implementations. None of the error flags can generate interrupts.

The Frame Error (FE) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE Flag is zero when the stop bit was correctly read (as one), and the FE Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE Flag is not affected by the setting of the USBS bit in UCSRC since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRA.

The Data OverRun (DOR) Flag indicates data loss due to a Receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. If the DOR Flag is set there was one or more serial frame lost between the frame last read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRA. The DOR Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (PE) Flag indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the PE bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRA. For more details see “Parity Bit Calculation” on page 140 and “Parity Checker” on page 148.

Parity Checker

The Parity Checker is active when the high USART Parity mode (UPM1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPM0 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 (PE) Flag can then be read by software to check if the frame had a parity error.

The PE bit is set if the next character that can be read from the receive buffer had a parity error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read.

Disabling the Receiver

In contrast to the Transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., the RXEN is set to zero) the Receiver will no longer override the normal function of the RxD port pin. The Receiver buffer FIFO will be flushed when the Receiver is disabled. Remaining data in the buffer will be lost

Flushing the Receive Buffer

The Receiver buffer FIFO will be flushed when the Receiver is disabled (i.e., the buffer will be emptied of its contents). Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR I/O location until the RXC Flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example[1]

```
USART_Flush:
  sbis UCSRA, RXC
  ret
  in   r16, UDR
  rjmp USART_Flush
```

C Code Example[1]

```
void USART_Flush( void )
{
  unsigned char dummy;
  while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Note: 1. See “About Code Examples” on page 7.

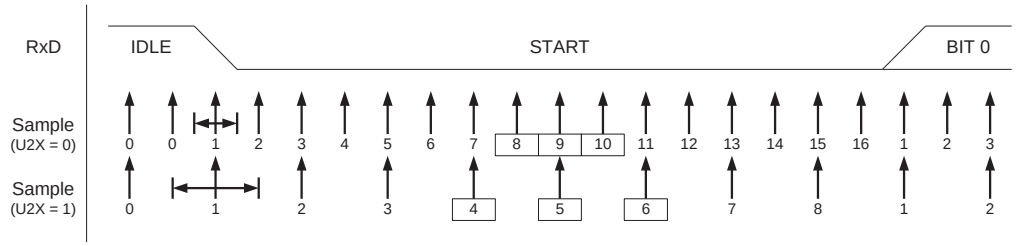
Asynchronous Data Reception

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 RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the Receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 68 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and eight times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the Double Speed mode (U2X = 1) of operation. Samples denoted zero are samples done when the RxD line is idle (i.e., no communication activity).

Figure 68. Start Bit Sampling

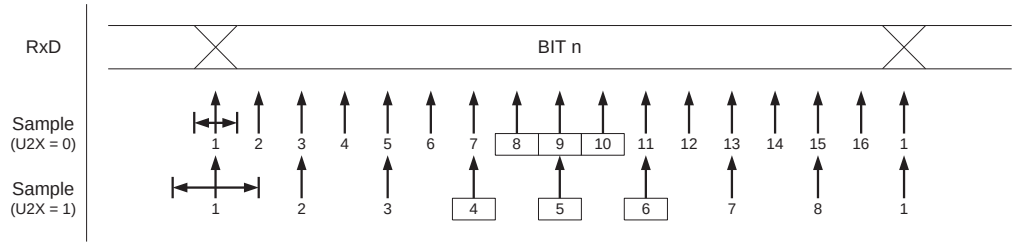


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical 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.

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 69 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.

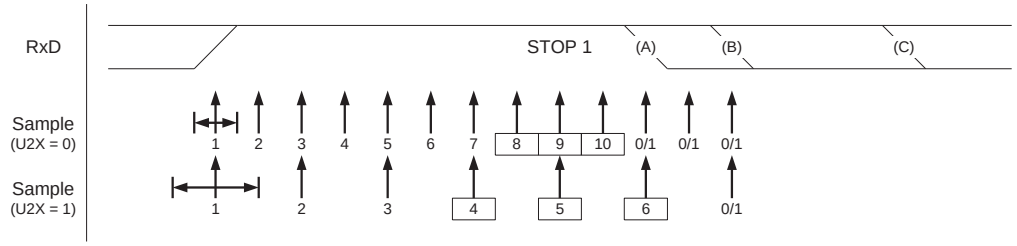
Figure 69. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure 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 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD 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 70 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 70. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE) Flag will be 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. For Normal Speed mode, the first low level sample can be at point marked (A) in Figure 70. For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar (see Table 61) base frequency, the Receiver will not be 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.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D Sum of character size and parity size (D = 5- to 10-bit).

S Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.

S_F First sample number used for majority voting. S_F = 8 for Normal Speed and S_F = 4 for Double Speed mode.

S_M Middle sample number used for majority voting. S_M = 9 for Normal Speed and S_M = 5 for Double Speed mode.

R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the Receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the Receiver baud rate.

Table 61 and Table 62 list the maximum Receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 61. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

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 62. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

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	+4.32/-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 was made under the assumption that the Receiver and Transmitter equally divides the maximum total error.

There are two possible sources for the Receiver's baud rate error. The Receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

Multi-processor Communication Mode

Setting the Multi-processor Communication mode (MPCM) bit in UCSRA enables a filtering function of incoming frames received by the USART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCM setting, but has to be used differently when it is a part of a system utilizing the Multi-processor 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 (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor 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 will receive the following data frames as normal, while the other Slave MCUs will ignore the received frames until another address frame is received.

Using MPCM

For an MCU to act as a Master MCU, it can use a 9-bit character frame format (UCSZ = 7). The ninth bit (TXB8) must be set when an address frame (TXB8 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The Slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication mode:

1. All Slave MCUs are in Multi-processor Communication mode (MPCM in UCSRA is set).
2. The Master MCU sends an address frame, and all slaves receive and read this frame. In the Slave MCUs, the RXC Flag in UCSRA will be set as normal.
3. Each Slave MCU reads the UDR Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte and keeps the MPCM setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other Slave MCUs, which still have the MPCM bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM bit and waits for a new address frame from Master. The process then repeats from 2.

Using any of the 5- to 8-bit character frame formats is possible, but impractical since the Receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the Transmitter and Receiver uses the same character size setting. If 5- to 8-bit character frames are used, the Transmitter must be set to use two stop bit (USBS = 1) since 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 MPCM bit. The MPCM bit shares the same I/O location as the TXC Flag and this might accidentally be cleared when using SBI or CBI instructions.

Accessing UBRRH/UCSRC Registers

The UBRRH Register shares the same I/O location as the UCSRC Register. Therefore some special consideration must be taken when accessing this I/O location.

Write Access

When doing a write access of this I/O location, the high bit of the value written, the USART Register Select (URSEL) bit, controls which one of the two registers that will be written. If URSEL is zero during a write operation, the UBRRH value will be updated. If URSEL is one, the UCSRC setting will be updated.

The following code examples show how to access the two registers.

Assembly Code Examples[1]

```
...
; Set UBRRH to 2
ldi r16,0x02
out UBRRH,r16
...
; Set the USBS and the UCSZ1 bit to one, and
; the remaining bits to zero.
ldi r16,(1<<URSEL)|(1<<USBS)|(1<<UCSZ1)
out UCSRC,r16
...
```

C Code Examples[1]

```
...
/* Set UBRRH to 2 */
UBRRH = 0x02;
...
/* Set the USBS and the UCSZ1 bit to one, and */
/* the remaining bits to zero. */
UCSRC = (1<<URSEL)|(1<<USBS)|(1<<UCSZ1);
...
```

Note: 1. See “About Code Examples” on page 7.

As the code examples illustrate, write accesses of the two registers are relatively unaffected of the sharing of I/O location.

Read Access

Doing a read access to the UBRRH or the UCSRC Register is a more complex operation. However, in most applications, it is rarely necessary to read any of these registers.

The read access is controlled by a timed sequence. Reading the I/O location once returns the UBRRH Register contents. If the register location was read in previous system clock cycle, reading the register in the current clock cycle will return the UCSRC contents. Note that the timed sequence for reading the UCSRC is an atomic operation. Interrupts must therefore be controlled (e.g., by disabling interrupts globally) during the read operation.

The following code example shows how to read the UCSRC Register contents.

Assembly Code Example[1]

```
USART_ReadUCSRC:
  ; Read UCSRC
  in  r16,UBRRH
  in  r16,UCSRC
  ret
```

C Code Example[1]

```
unsigned char USART_ReadUCSRC( void )
{
  unsigned char ucsrc;
  /* Read UCSRC */
  ucsrc = UBRRH;
  ucsrc = UCSRC;
  return ucsrc;
}
```

Note: 1. See “About Code Examples” on page 7.

The assembly code example returns the UCSRC value in r16.

Reading the UBRRH contents is not an atomic operation and therefore it can be read as an ordinary register, as long as the previous instruction did not access the register location.

USART Register Description

USART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (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 Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE Flag in the UCSRA Register is set. Data written to UDR when the UDRE Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read modify write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
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 – RXC: USART Receive Complete**

This flag bit is set when there are 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 will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete**

This flag bit 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 (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty**

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register Empty interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. For example, when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

- **Bit 3 – DOR: 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), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 2 – PE: 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 (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 1 – U2X: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 151.

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
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 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE Flags.

- **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed. For example, when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (character size) in a frame the Receiver and Transmitter use.

- **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8: Transmit Data Bit 8**

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with 9 data bits. Must be written before writing the low bits to UDR.

USART Control and Status Register C – UCSRC

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

The UCSRC Register shares the same I/O location as the UBRRH Register. See the “Accessing UBRRH/UCSRC Registers” on page 153 which describes how to access this register.

- **Bit 7 – URSEL: Register Select**

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

- **Bit 6 – UMSEL: USART Mode Select**

This bit selects between asynchronous and synchronous mode of operation.

Table 63. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

- **Bit 5:4 – UPM1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE Flag in UCSRA will be set.

Table 64. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 – USBS: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 65. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (character size) in a frame the Receiver and Transmitter use.

Table 66. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	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 – UCPOL: Clock Polarity**

This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 67. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

USART Baud Rate Registers – UBRRL and UBRRH

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/UCSRC Registers" on page 153 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 – Reserved Bits**

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

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 68. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 150). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 68. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 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)	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

1. UBRR = 0, Error = 0.0%

Table 69. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 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)	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

1. UBRR = 0, Error = 0.0%

Table 70. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 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)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

Table 71. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

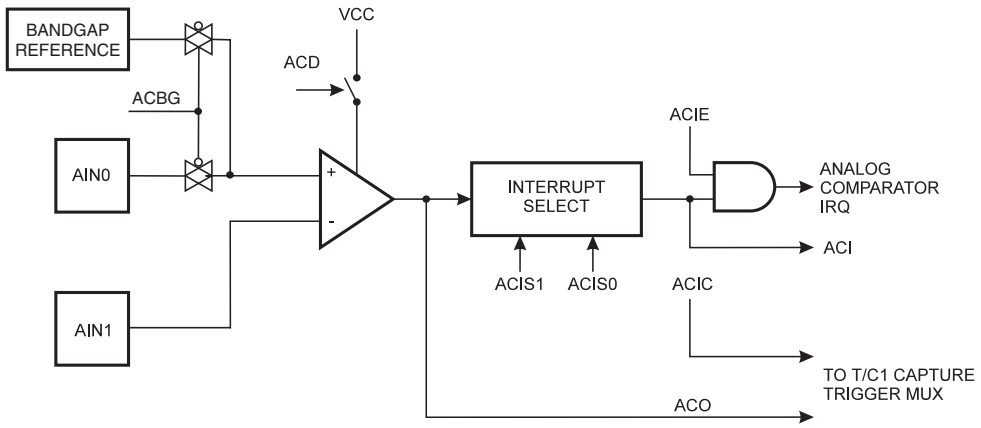
Baud Rate (bps)	f_{osc} = 16.0000 MHz				f_{osc} = 18.4320 MHz				f_{osc} = 20.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	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)	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRR = 0, Error = 0.0%

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's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 71.

Figure 71. Analog Comparator Block Diagram(1)



Note: 1. Refer to Figure 1 on page 2 and Table 29 on page 67 for Analog Comparator pin placement.

Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	**ACBG**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
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 a logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

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

When this bit is set, a fixed bandgap 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. See "Internal Voltage Reference" on page 50.

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

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

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

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

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

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – 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. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/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 TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set.

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

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 72.

Table 72. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

When changing the ACIS1/ACIS0 bits, the Analog Comparator interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

Boot Loader Support – Read-While-Write Self-Programming

The Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the Program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

Features

- **Read-While-Write Self-Programming**
- **Flexible Boot Memory Size**
- **High Security (Separate Boot Lock bits for a Flexible Protection)**
- **Separate Fuse to Select Reset Vector**
- **Optimized Page[1] Size**
- **Code Efficient Algorithm**
- **Efficient Read-Modify-Write Support**

Note: 1. A page is a section in the Flash consisting of several bytes (see Table 89 on page 183) used during programming. The page organization does not affect normal operation.

Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section (see Figure 73). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 78 on page 177 and Figure 73. These two sections can have different level of protection since they have different sets of Lock bits.

Application Section

The Application section is the section of the Flash that is used for storing the application code. The protection level for the Application section can be selected by the application Boot Lock bits (Boot Lock bits 0), see Table 74 on page 169. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

BLS – Boot Loader Section

While the Application section is used for storing the application code, the Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1), see Table 75 on page 169.

Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on which address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Table 79 on page 177 and Figure 73 on page 168. The main difference between the two sections is:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation.
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation.

Note that the user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax “Read-While-Write section” refers to which section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.

RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an on-going programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by a rcall/rjmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program memory Control Register (SPMCR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See “Store Program memory Control Register – SPMCR” on page 170. for details on how to clear RWWSB.

NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire page erase or page write operation.

Table 73. Read-While-Write Features

Which Section does the Z-pointer Address during the Programming?	Which Section Can be Read during Programming?	Is the CPU Halted?	Read-While-Write Supported?
RWW section	NRWW section	No	Yes
NRWW section	None	Yes	No

Figure 72. Read-While-Write vs. No Read-While-Write

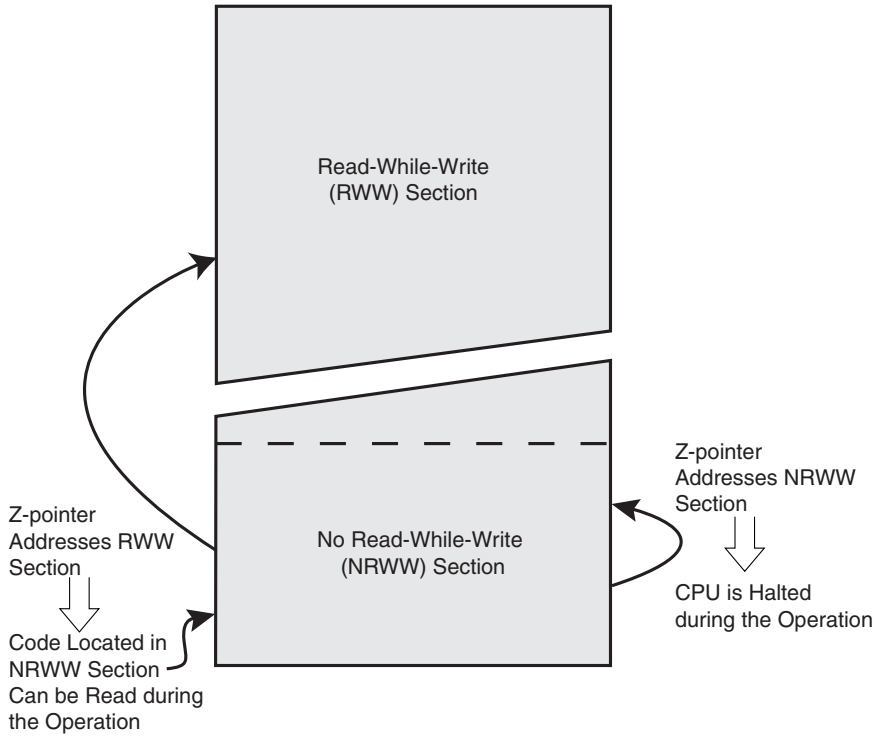
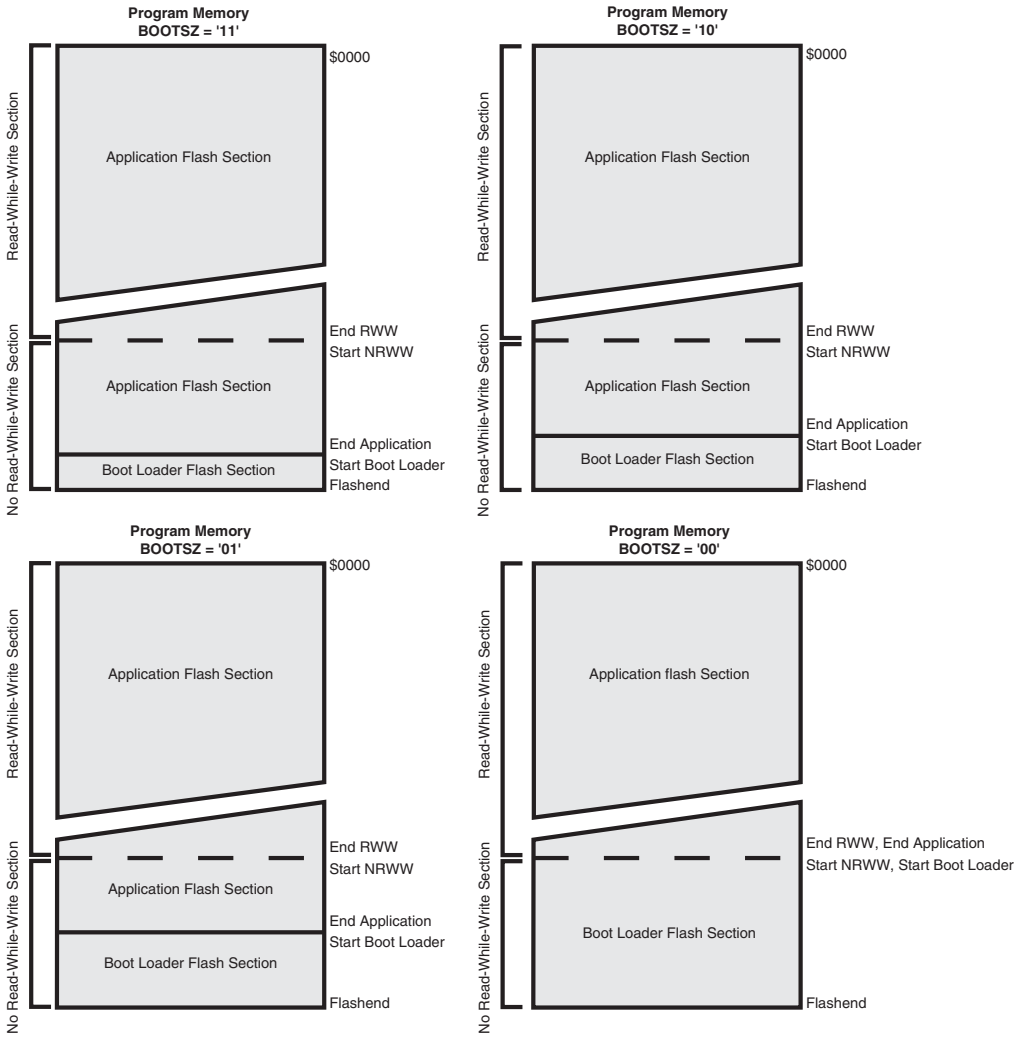


Figure 73. Memory Sections[1]



Note: 1. The parameters in the figure above are given in Table 78 on page 177.

Boot Loader Lock bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU.
- To protect only the Boot Loader Flash section from a software update by the MCU.
- To protect only the Application Flash section from a software update by the MCU.
- Allow software update in the entire Flash.

See Table 74 and Table 75 for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 1) does not control reading nor writing by LPM/SPM, if it is attempted.

Table 74. Boot Lock Bit0 Protection Modes (Application Section)[1]

BLB0 Mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

Note: 1. “1” means unprogrammed, “0” means programmed

Table 75. Boot Lock Bit1 Protection Modes (Boot Loader Section)[1]

BLB1 Mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: 1. “1” means unprogrammed, “0” means programmed

Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the Serial or Parallel Programming interface.

Table 76. Boot Reset Fuse[1]

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address $0000)
0	Reset Vector = Boot Loader Reset (see Table 78 on page 177)

Note: 1. “1” means unprogrammed, “0” means programmed

Store Program memory Control Register – SPMCR

The Store Program memory Control Register contains the control bits needed to control the Boot Loader operations.

Bit	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPMIE: SPM Interrupt Enable**

When the SPMIE bit is written to one, and the I-bit in the Status Register is set (one), the SPM ready interrupt will be enabled. The SPM ready interrupt will be executed as long as the SPMEN bit in the SPMCR Register is cleared.

- **Bit 6 – RWWSB: Read-While-Write Section Busy**

When a Self-Programming (Page Erase or Page Write) operation to the RWW section is initiated, the RWWSB will be set (one) by hardware. When the RWWSB bit is set, the RWW section cannot be accessed. The RWWSB bit will be cleared if the RWWSRE bit is written to one after a Self-Programming operation is completed. Alternatively the RWWSB bit will automatically be cleared if a page load operation is initiated.

- **Bit 5 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega8515 and always read as zero.

- **Bit 4 – RWWSRE: Read-While-Write Section Read Enable**

When programming (page erase or page write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the Flash is busy with a Page Erase or a Page Write (SPMEN is set). If the RWWSRE bit is written while the Flash is being loaded, the Flash load operation will abort and the data loaded will be lost.

- **Bit 3 – BLBSET: Boot Lock Bit Set**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets Boot Lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the Lock bit set, or if no SPM instruction is executed within four clock cycles.

An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See “Reading the Fuse and Lock bits from Software” on page 174 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a Page Write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of

the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 0 – SPMEN: Store Program memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT' or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During Page Erase and Page Write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than "10001", "01001", "00101", "00011", or "00001" in the lower five bits will have no effect.

Addressing the Flash During Self-Programming

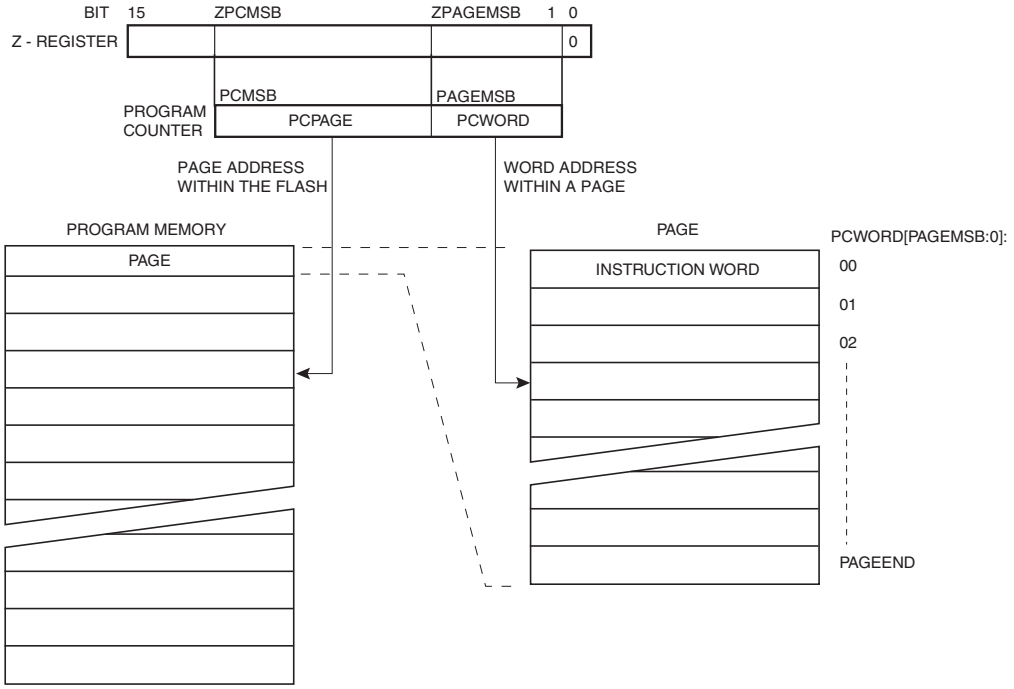
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 89 on page 183), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 74. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the Flash byte by byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 74. Addressing the Flash during SPM[1][2]



Notes: 1. The different variables used in Figure 74 are listed in Table 80 on page 178.
2. PCPAGE and PCWORD are listed in Table 89 on page 183.

Self-Programming the Flash

The Program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a Page Erase and a Page Write operation:

Alternative 1, fill the buffer before a Page Erase:

- Fill temporary page buffer.
- Perform a Page Erase.
- Perform a Page Write.

Alternative 2, fill the buffer after Page Erase:

- Perform a Page Erase.
- Fill temporary page buffer.
- Perform a Page Write.

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page. See “Simple Assembly Code Example for a Boot Loader” on page 175 for an assembly code example.

Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “X0000011” to SPMCR and execute SPM within four clock cycles after writing SPMCR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer must be written zero during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the Page Erase.
- Page Erase to the NRWW section: The CPU is halted during the operation.

Filling the Temporary Buffer (page loading)

To write an instruction word, set up the address in the Z pointer and data in R1:R0, write “00000001” to SPMCR and execute SPM within four clock cycles after writing SPMCR. 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 RWWSRE bit in SPMCR. 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.

Note: If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “X0000101” to SPMCR and execute SPM within four clock cycles after writing SPMCR. 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 zero during this operation.

- Page Write to the RWW section: The NRWW section can be read during the Page Write.
- Page Write to the NRWW section: The CPU is halted during the operation.

Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCR is cleared. This means that the interrupt can be used instead of polling the SPMCR Register in software. When using the SPM interrupt, the Interrupt Vectors should be moved to the BLS section to avoid that an interrupt is accessing the RWW section when it is blocked for reading. How to move the interrupts is described in “Interrupts” on page 54.

Consideration While Updating BLS

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit11 unprogrammed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock bit11 to protect the Boot Loader software from any internal software changes.

Prevent Reading the RWW Section During Self-Programming

During Self-Programming (either Page Erase or Page Write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the Self-Programming operation. The RWWSB in the SPMCR will be set as long as the RWW section is busy. During Self-Programming the Interrupt Vector table should be moved to the BLS as described in “Interrupts” on page 54, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See “Simple Assembly Code Example for a Boot Loader” on page 175 for an example.

Setting the Boot Loader Lock bits by SPM

To set the Boot Loader Lock bits, write the desired data to R0, write “X0001001” to SPMCR and execute SPM within four clock cycles after writing SPMCR. The only accessible Lock bits are the Boot Lock bits that may prevent the Application and Boot Loader section from any software update by the MCU.

Bit	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

See Table 74 and Table 75 for how the different settings of the Boot Loader bits affect the Flash access.

If bits 5..2 in R0 are cleared (zero), the corresponding Boot Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMEN are set in SPMCR. The Z-pointer is don’t care during this operation, but for future compatibility it is recommended to load the Z-pointer with $0001 (same as used for reading the Lock bits). For future compatibility It is also recommended to set bits 7, 6, 1, and 0 in R0 to “1” when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

EEPROM Write Prevents Writing to SPMCR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEWE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCR Register.

Reading the Fuse and Lock bits from Software

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with $0001 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within three CPU cycles after the BLBSET and SPMEN bits are set in SPMCR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLBSET and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low bits is similar to the one described above for reading the Lock bits. To read the Fuse Low bits, load the Z-pointer with $0000 and set the BLBSET and SPMEN bits in SPMCR. When an LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse Low bits (FLB) will be loaded in the destination register as shown below. Refer to Table 84 on page 181 for a detailed description and mapping of the Fuse Low bits.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the Fuse High bits, load $0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse High bits (FHB) will be loaded in the destination register as shown below. Refer to Table 83 on page 180 for detailed description and mapping of the Fuse High bits.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

Preventing Flash Corruption

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

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

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
2. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} Reset Protection circuit can be used. If a Reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
3. Keep the AVR core in Power-down Sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCR Register and thus the Flash from unintentional writes.

Programming Time for Flash when using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 77 shows the typical programming time for Flash accesses from the CPU.

Table 77. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash Write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms

Simple Assembly Code Example for a Boot Loader

```
  ;-the routine writes one page of data from RAM to Flash
  ; the first data location in RAM is pointed to by the Y pointer
  ; the first data location in Flash is pointed to by the Z pointer
  ;-error handling is not included
  ;-the routine must be placed inside the boot space
  ; (at least the Do_spm sub routine). Only code inside NRWW section
can
  ; be read during Self-Programming (page erase and page write).
  ;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
  ; loophi (r25), spmcrval (r20)
  ; storing and restoring of registers is not included in the routine
  ; register usage can be optimized at the expense of code size
  ;-It is assumed that either the interrupt table is moved to the
Boot
  ; loader section or that the interrupts are disabled.
.equ PAGESIZEB = PAGESIZE*2     ;PAGESIZEB is page size in BYTES, not
words
.org SMALLBOOTSTART
Write_page:
  ; page erase
  ldi  spmcrval, (1<<PGERS) | (1<<SPMEN)
  rcallDo_spm
```

```
  ; re-enable the RWW section
  ldi  spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  rcallDo_spm

  ; transfer data from RAM to Flash page buffer
  ldi  looplo, low(PAGESIZEB)   ;init loop variable
  ldi  loophi, high(PAGESIZEB)  ;not required for PAGESIZEB<=256
Wrloop:
  ld   r0, Y+
  ld   r1, Y+
  ldi  spmcrval, (1<<SPMEN)
  rcallDo_spm
  adiw ZH:ZL, 2
  sbiw loophi:looplo, 2         ;use subi for PAGESIZEB<=256
  brne Wrloop

  ; execute page write
  subi ZL, low(PAGESIZEB)       ;restore pointer
  sbci ZH, high(PAGESIZEB)      ;not required for PAGESIZEB<=256
  ldi  spmcrval, (1<<PGWRT) | (1<<SPMEN)
  rcallDo_spm

  ; re-enable the RWW section
  ldi  spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  rcallDo_spm

  ; read back and check, optional
  ldi  looplo, low(PAGESIZEB)   ;init loop variable
  ldi  loophi, high(PAGESIZEB)  ;not required for PAGESIZEB<=256
  subi YL, low(PAGESIZEB)       ;restore pointer
  sbci YH, high(PAGESIZEB)
Rdloop:
  lpm  r0, Z+
  ld   r1, Y+
  cpse r0, r1
  rjmp Error
  sbiw loophi:looplo, 1         ;use subi for PAGESIZEB<=256
  brne Rdloop

  ; return to RWW section
  ; verify that RWW section is safe to read
Return:
  in   temp1, SPMCR
  sbrs temp1, RWWSB             ; If RWWSB is set, the RWW section is
not
                                ; ready yet
  ret
  ; re-enable the RWW section
  ldi  spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  rcallDo_spm
  rjmp Return

Do_spm:
  ; check for previous SPM complete
Wait_spm:
  in   temp1, SPMCR
  sbrc temp1, SPMEN
  rjmp Wait_spm
```

```
  ; input: spmcrval determines SPM action
  ; disable interrupts if enabled, store status
  in   temp2, SREG
  cli
  ; check that no EEPROM write access is present
Wait_ee:
  sbic EECR, EEWE
  rjmp Wait_ee
  ; SPM timed sequence
  out  SPMCR, spmcrval
  spm
  ; restore SREG (to enable interrupts if originally enabled)
  out  SREG, temp2
  ret
```

ATmega8515 Boot Loader Parameters

In Table 78 through Table 80, the parameters used in the description of the Self-Programming are given.

Table 78. Boot Size Configuration[1]

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (start Boot Loader Section)
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF7F	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xEFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xDFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xBFF	0xC00

Note: 1. The different BOOTSZ Fuse configurations are shown in Figure 73

Table 79. Read-While-Write Limit[1]

Section	Pages	Address
Read-While-Write section (RWW)	96	0x000 - 0xBFF
No Read-While-Write section (NRWW)	32	0xC00 - 0xFFF

Note: 1. For details about these two section, see “NRWW – No Read-While-Write Section” on page 167 and “RWW – Read-While-Write Section” on page 167.

Table 80. Explanation of Different Variables used in Figure 74 and the Mapping to the Z-pointer(1)

Variable		Corresponding Z-value	Description
PCMSB	11		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	4		Most significant bit which is used to address the words within one page (32 words in a page requires five bits PC [4:0]).
ZPCMSB		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z5	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[11:5]	Z12:Z6	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[4:0]	Z5:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

Note: 1. Z15:Z13: always ignored.
Z0: should be zero for all SPM commands, byte select for the LPM instruction.
See “Addressing the Flash During Self-Programming” on page 171 for details about the use of Z-pointer during Self-Programming.

Memory Programming

Program and Data Memory Lock bits

The ATmega8515 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 82. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 81. Lock Bit Byte[1]

Lock Bit Byte	Bit no	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. “1” means unprogrammed, “0” means programmed

Table 82. Lock Bit Protection Modes[2]

Memory Lock bits			Protection Type
LB Mode	**LB2**	**LB1**	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.[1]
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.[1]
BLB0 Mode	**BLB02**	**BLB01**	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
BLB1 Mode	**BLB12**	**BLB11**	

Table 82. Lock Bit Protection Modes[2] (Continued)

Memory Lock bits			Protection Type
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Notes: 1. Program the Fuse bits before programming the Lock bits.
2. "1" means unprogrammed, "0" means programmed

Fuse bits

The ATmega8515 has two Fuse bytes. Table 83 and Table 84 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the Fuses are read as logical zero, "0", if they are programmed.

Table 83. Fuse High Byte

Fuse High Byte	Bit no	Description	Default Value
S8515C	7	AT90S4414/8515 compatibility mode	1 (unprogrammed)
WDTON	6	Watchdog Timer always on	1 (unprogrammed)
SPIEN[1][2]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[3]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 78 for details)	0 (programmed)[4]
BOOTSZ0	1	Select Boot Size (see Table 78 for details)	0 (programmed)[4]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Notes: 1. See "AT90S4414/8515 Compatibility Mode" on page 4 for details.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. The CKOPT Fuse functionality depends on the setting of the CKSEL bits. See "Clock Sources" on page 35. for details.
4. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 78 on page 177.

Table 84. Fuse Low Byte

Fuse Low Byte	Bit no	Description	Default value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

Notes: 1. The default value of SUT1..0 results in maximum start-up time. See Table 13 on page 39 for details.
2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 1 MHz. See Table 5 on page 35 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Latching of Fuses

The fuse values are latched when the device enters Programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

Signature Bytes

All Atmel microcontrollers have a 3-byte signature code which identifies the device. This code can be read in both Serial and Parallel mode, also when the device is locked. The three bytes reside in a separate address space.

For the ATmega8515 the signature bytes are:

1. $000: $1E (indicates manufactured by Atmel).
2. $001: $93 (indicates 8KB Flash memory).
3. $002: $06 (indicates ATmega8515 device when $001 is $93).

Calibration Byte

The ATmega8515 stores four different calibration values for the internal RC Oscillator. These bytes resides in the signature row high byte of the addresses 0x000, 0x0001, 0x0002, and 0x0003 for 1, 2, 4, and 8 MHz respectively. During Reset, the 1 MHz value is automatically loaded into the OSCCAL Register. If other frequencies are used, the calibration value has to be loaded manually, see “Oscillator Calibration Register – OSCCAL” on page 39 for details.

Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega8515. Pulses are assumed to be at least 250 ns unless otherwise noted.

Signal Names

In this section, some pins of the ATmega8515 are referenced by signal names describing their functionality during parallel programming, see Figure 75 and Table 85. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 87.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 88.

Figure 75. Parallel Programming

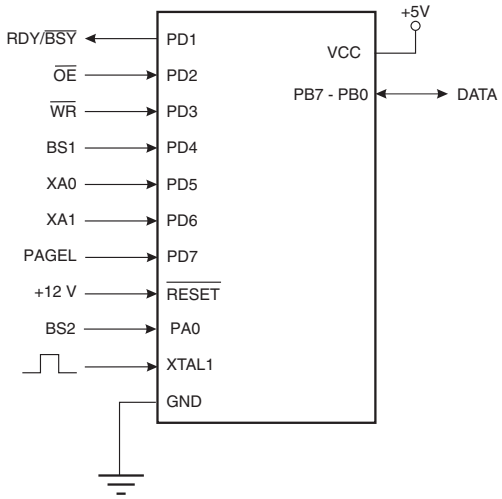


Table 85. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active low)
$\overline{WR}$	PD3	I	Write Pulse (Active low)
BS1	PD4	I	Byte Select 1 (“0” selects low byte, “1” selects high byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
PAGEL	PD7	I	Program memory and EEPROM data Page Load
BS2	PA0	I	Byte Select 2 (“0” selects low byte, “1” selects 2’nd high byte)
DATA	PB7-0	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low)

Table 86. Pin Values used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 87. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS1)
0	1	Load Data (High or Low data byte for Flash determined by BS1)
1	0	Load Command
1	1	No Action, Idle

Table 88. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature Bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

Table 89. No. of Words in a Page and No. of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11

Table 90. No. of Words in a Page and No. of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

Parallel Programming

Enter Programming Mode

The following algorithm puts the device in Parallel Programming mode:

1. Apply 4.5 - 5.5 V between V_{CC} and GND, and wait for at least 100 µs.
2. Set $\overline{\text{RESET}}$ to “0”, wait for at least 100 ns and toggle XTAL1 at least six times.
3. Set the Prog_enable pins listed in Table 86 on page 183 to “0000” and wait at least 100 ns.
4. Apply 11.5 - 12.5V to $\overline{\text{RESET}}$. Any activity on Prog_enable pins within 100 ns after +12V has been applied to $\overline{\text{RESET}}$, will cause the device to fail entering Programming mode.

Note, if External Crystal or External RC configuration is selected, it may not be possible to apply qualified XTAL1 pulses. In such cases, the following algorithm should be followed:

1. Set Prog_enable pins listed in Table 86 on page 183 to “0000”.
2. Apply 4.5 - 5.5V between V_{CC} and GND simultaneously as 11.5 - 12.5V is applied to $\overline{\text{RESET}}$.
3. Wait 100 µs.
4. Re-program the fuses to ensure that External Clock is selected as clock source (CKSEL3:0 = 0b0000) If Lock bits are programmed, a Chip Erase command must be executed before changing the fuses.
5. Exit Programming mode by power the device down or by bringing $\overline{\text{RESET}}$ pin to 0b0.
6. Entering Programming mode with the original algorithm, as described above.

Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value $FF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

Chip Erase

The Chip Erase will erase the Flash and EEPROM[1] memories plus Lock bits. The Lock bits are not reset until the Program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash or EEPROM are reprogrammed.

Note: 1. The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{\text{WR}}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{\text{BSY}}$ goes low.
6. Wait until RDY/$\overline{\text{BSY}}$ goes high before loading a new command.

Programming the Flash

The Flash is organized in pages, see Table 89 on page 183. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

B. Load Address Low byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte ($00 - $FF).
3. Give XTAL1 a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte ($00 - $FF).
4. Give XTAL1 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 77 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 76 on page 186. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a page write.

G. Load Address High byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address high byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

1. Set BS1 = “0”.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.

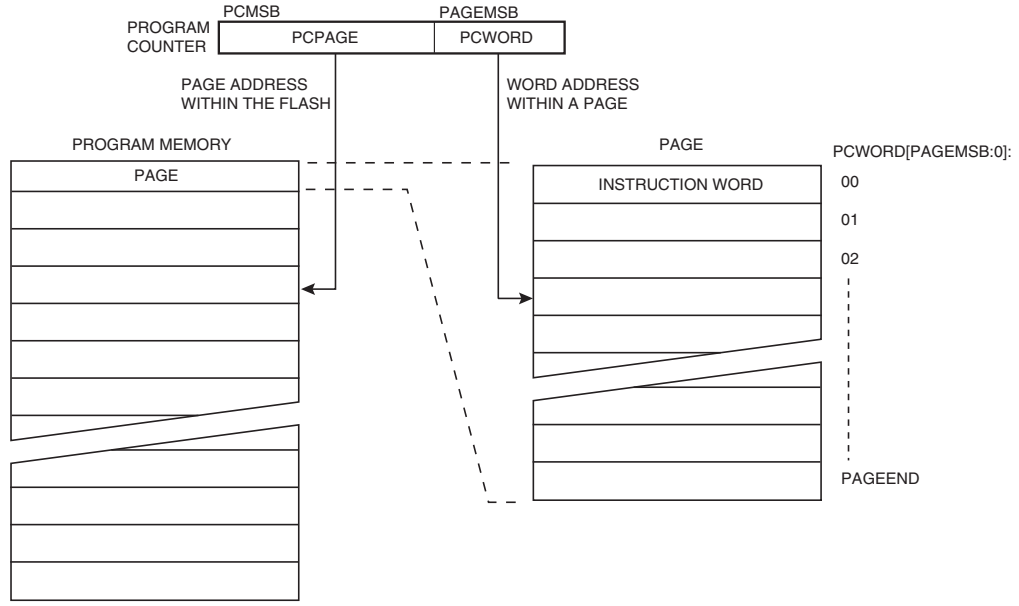
3. Wait until RDY/$\overline{\text{BSY}}$ goes high. (See Figure 77 for signal waveforms)

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

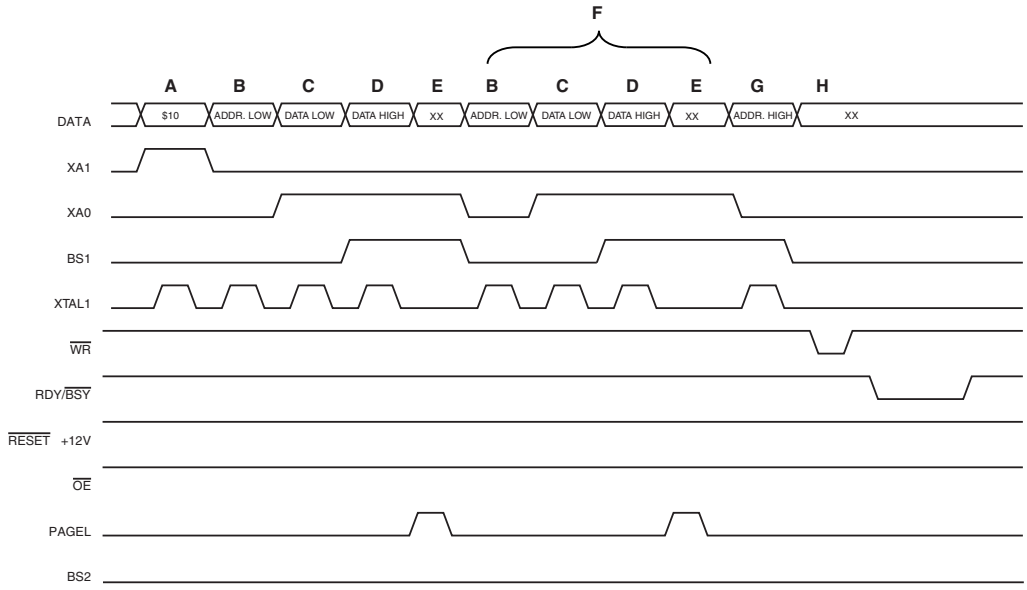
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 76. Addressing the Flash which is Organized in Pages(1)



Note: 1. PCPAGE and PCWORD are listed in Table 89 on page 183.

Figure 77. Programming the Flash Waveforms



Note: "XX" is don't care. The letters refer to the programming description above.

Programming the EEPROM

The EEPROM is organized in pages, see Table 90 on page 183. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM Data memory is as follows (refer to "Programming the Flash" on page 185 for details on Command, Address and Data loading):

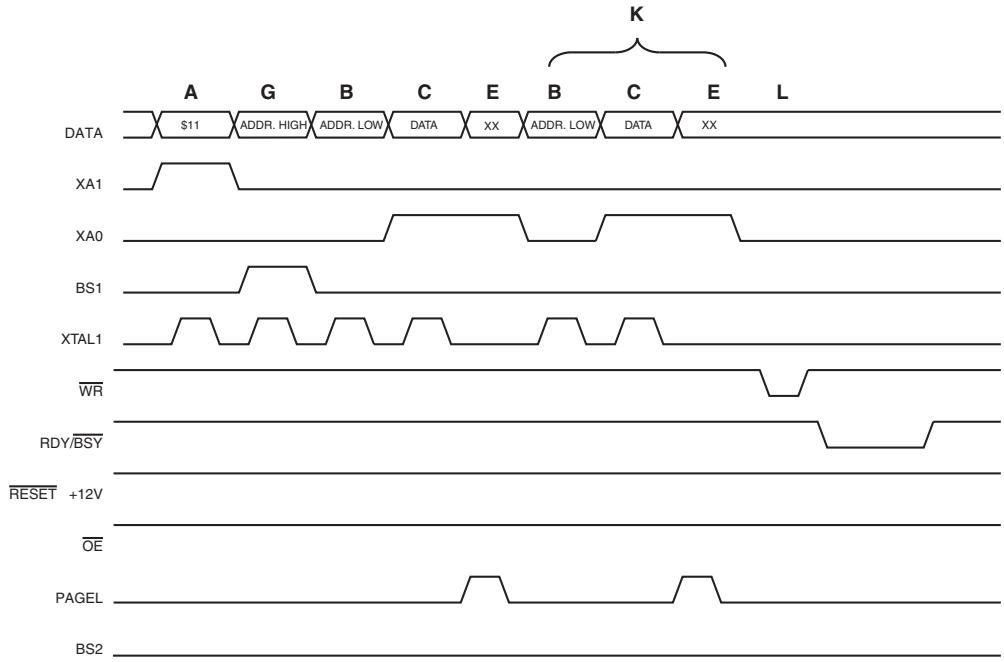
1. A: Load Command "0001 0001".
2. G: Load Address High Byte ($00 - $FF).
3. B: Load Address Low Byte ($00 - $FF).
4. C: Load Data ($00 - $FF).
5. E: Latch data (give PAGEL a positive pulse).

K: Repeat 3 through 5 until the entire buffer is filled.

L: Program EEPROM page.

1. Set BS1 to "0".
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page. (See Figure 78 for signal waveforms.)

Figure 78. Programming the EEPROM Waveforms



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 185 for details on Command and Address loading):

1. A: Load Command “0000 0010”.
2. G: Load Address High Byte ($00 - $FF).
3. B: Load Address Low Byte ($00 - $FF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The Flash word low byte can now be read at DATA.
5. Set BS1 to “1”. The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 185 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
2. G: Load Address High Byte ($00 - $FF).
3. B: Load Address Low Byte ($00 - $FF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 185 for details on Command and Data loading):

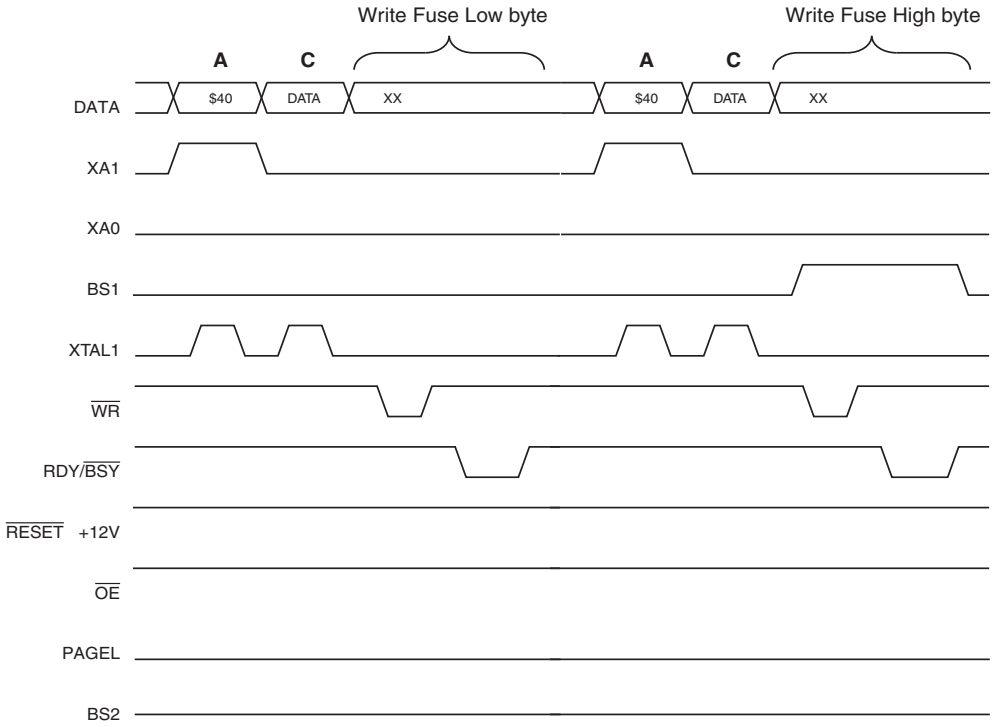
1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “0” and BS2 to “0”. This selects low data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 185 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

Figure 79. Programming the Fuses Waveforms



Programming the Lock bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 185 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

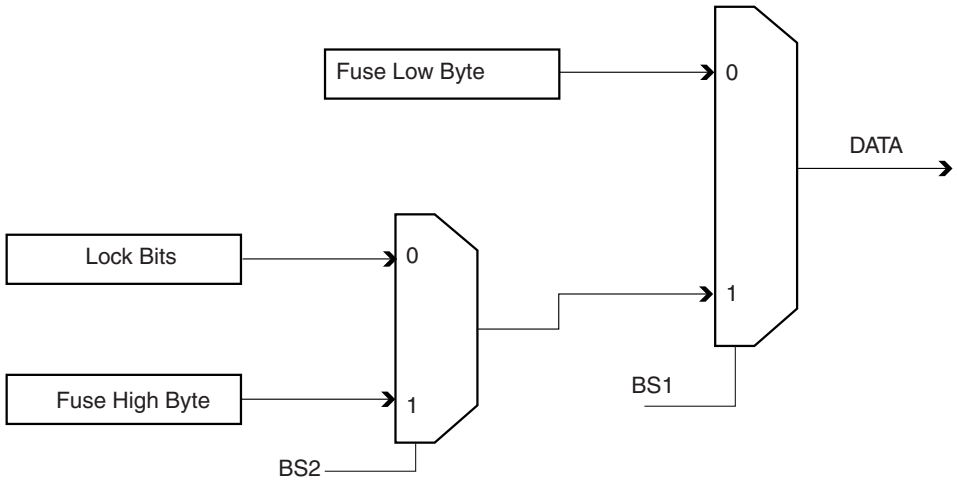
The Lock bits can only be cleared by executing Chip Erase.

Reading the Fuse and Lock bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to “Programming the Flash” on page 185 for details on Command loading):

1. A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “1”.

Figure 80. Mapping Between BS1, BS2, and the Fuse- and Lock bits During Read



Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to “Programming the Flash” on page 185 for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte ($00 - $02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to “Programming the Flash” on page 185 for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte, $00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

Parallel Programming Characteristics

Figure 81. Parallel Programming Timing, Including some General Timing Requirements

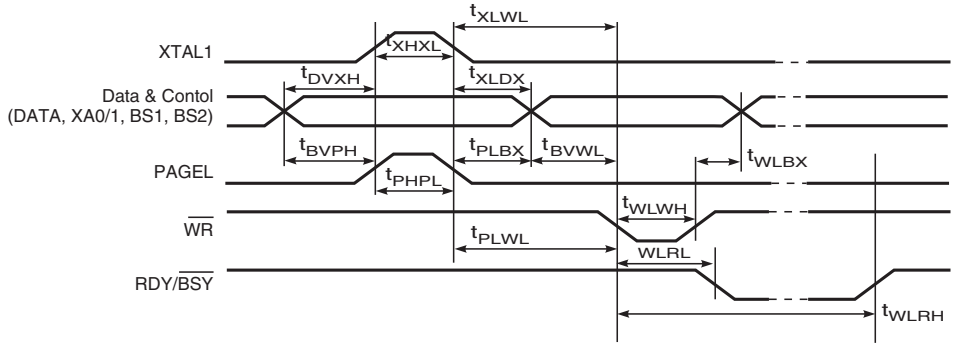
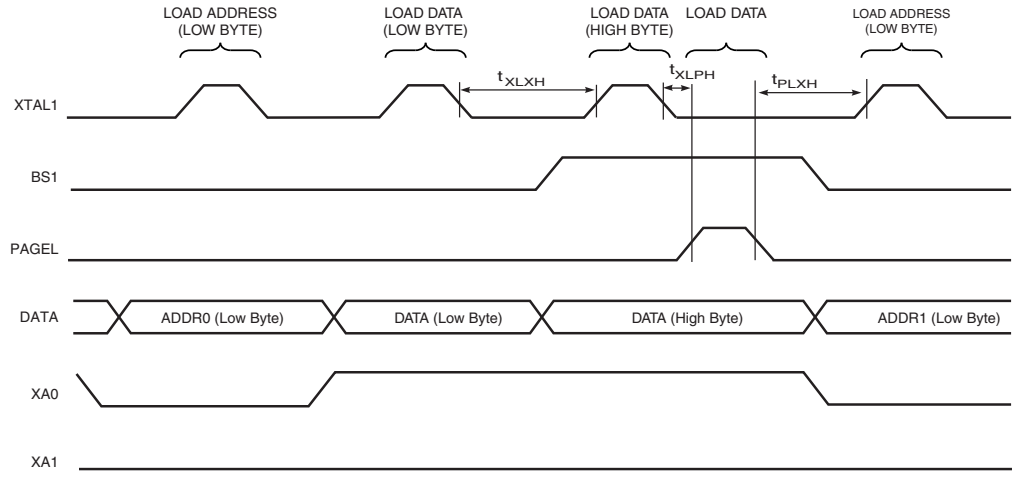
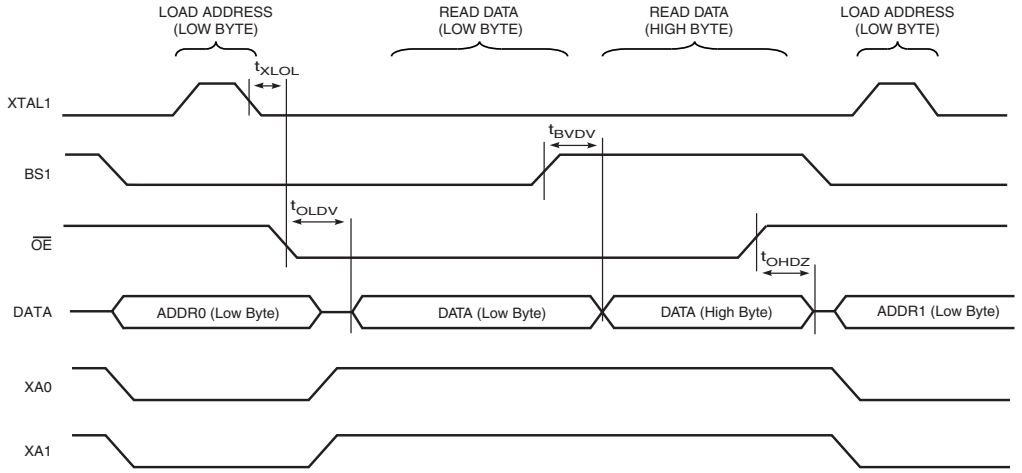


Figure 82. Parallel Programming Timing, Loading Sequence with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 81 (i.e. t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 83. Parallel Programming Timing, Reading Sequence (within the same Page) with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 81 (i.e. t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 91. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns

Table 91. Parallel Programming Characteristics, V_{CC} = 5V ± 10% (Continued)

Symbol	Parameter	Min	Typ	Max	Units
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	XTAL1 Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to XTAL1 high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[1]	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			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.

Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Note: In Table 92, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

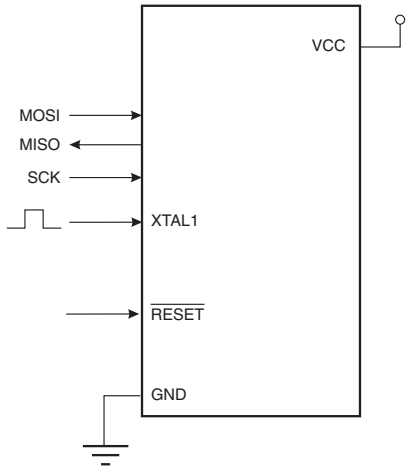
Serial Programming Pin Mapping

Table 92. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB5	I	Serial data in
MISO	PB6	O	Serial data out
SCK	PB7	I	Serial clock

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 92 on page 193, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 84. Serial Programming and Verify(1)



Note: 1. If the device is clocked by the internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into $FF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for $f_{ck} \geq$ 12 MHz

High: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for $f_{ck} \geq$ 12 MHz

Serial Programming Algorithm

When writing serial data to the ATmega8515, data is clocked on the rising edge of SCK.

When reading data from the ATmega8515, data is clocked on the falling edge of SCK. See Figure 85 for timing details.

To program and verify the ATmega8515 in the Serial Programming mode, the following sequence is recommended (See four byte instruction formats in Table 94.):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during Power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The Serial Programming instructions will not work if the communication is out of synchronization. When in synchronization, the second byte ($53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the $53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The page size is found in Table 89 on page 183. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the Load Program memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program memory Page is stored by loading the Write Program memory Page instruction with the 7 MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page, see Table 93. Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte, see Table 93. In a chip erased device, no $FFs in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

Data Polling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value $FF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value $FF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip erased device contains $FF in all locations, programming of addresses that are meant to contain $FF, can be skipped. See Table 93 for t_{WD_FLASH} value.

Data Polling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value $FF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value $FF, but the user should have the following in mind: As a chip erased device contains $FF in all locations, programming of addresses that are meant to contain $FF, can be skipped. This does not apply if the EEPROM is reprogrammed without chip-erasing the device. In this case, data polling cannot be used for the value $FF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See Table 93 for t_{WD_EEPROM} value.

Table 93. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FUSE}	4.5 ms
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	9.0 ms
t_{WD_ERASE}	9.0 ms

Figure 85. Serial Programming Waveforms

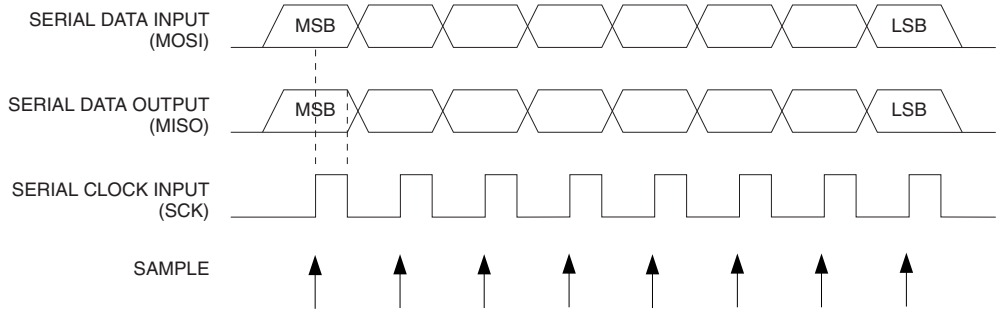


Table 94. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program memory	0010 **H**000	0000 **aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**.
Load Program memory Page	0100 **H**000	0000 xxxx	xxx**b bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program memory page at word address **b**. Data low byte must be loaded before Data high byte is applied within the same address.
Write Program memory Page	0100 1100	0000 **aaaa**	**bbb**x xxxx	xxxx xxxx	Write Program memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	00xx xxx**a**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	00xx xxx**a**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Read Lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. “0” = programmed, “1” = unprogrammed. See Table 81 on page 179 for details.
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock bits. Set bits = “0” to program Lock bits. See Table 81 on page 179 for details.
Read Signature Byte	0011 0000	00xx xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 84 on page 181 for details.
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 83 on page 180 for details.
Read Fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. “0” = programmed, “1” = unprogrammed. See Table 84 on page 181 for details.
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse high bits. “0” = programmed, “1” = unprogrammed. See Table 83 on page 180 for details.
Read Calibration Byte	0011 1000	00xx xxxx	0000 00**bb**	**oooo oooo**	Read Calibration Byte

Note: **a** = address high bits
b = address low bits
H = 0 - Low byte, 1 - High Byte
o = data out
i = data in
x = don’t care

Electrical Characteristics

Absolute Maximum Ratings*

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

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC}=2.7V - 5.5V	-0.5		0.2 V_{CC}(1)	V
V_{IH}	Input High Voltage except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC}=2.7V - 5.5V	0.6 V_{CC}(2)		V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage XTAL1 pin	V_{CC}=2.7V - 5.5V	-0.5		0.1 V_{CC}(1)	V
V_{IH1}	Input High Voltage XTAL1 pin	V_{CC}=2.7V - 5.5V	0.8 V_{CC}(2)		V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage $\overline{\text{RESET}}$ pin	V_{CC}=2.7V - 5.5V	-0.5		0.2 V_{CC}	V
V_{IH2}	Input High Voltage $\overline{\text{RESET}}$ pin	V_{CC}=2.7V - 5.5V	0.9 V_{CC}(2)		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage(3) (Ports A,B,C,D,E)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH}	Output High Voltage(4) (Ports A,B,C,D,E)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.2			V V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{pu}	I/O Pin Pull-up Resistor		20		50	kΩ

DC Characteristics (Continued)

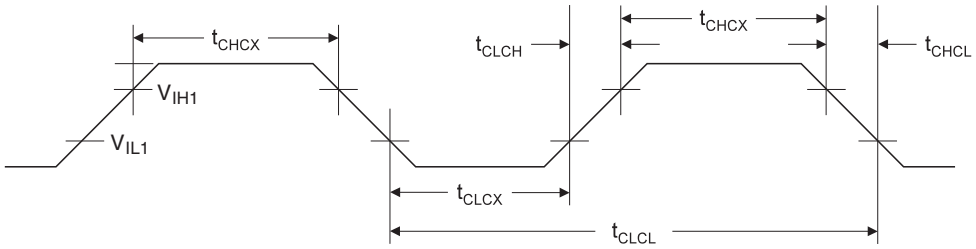
T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
I_{CC}	Power Supply Current	Active 4 MHz, V_{CC} = 3V (ATmega8515L)			4	mA
		Active 8 MHz, V_{CC} = 5V (ATmega8515)			12	mA
		Idle 4 MHz, V_{CC} = 3V (ATmega8515L)			1.5	mA
		Idle 8 MHz, V_{CC} = 5V (ATmega8515)			5.5	mA
	Power-down mode(5)	WDT enabled, V_{CC} = 3V			< 13	µA
		WDT disabled, V_{CC} = 3V			< 2	µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2			40	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2	-50		50	nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

Notes:
1. "Max" means the highest value where the pin is guaranteed to be read as low.
2. "Min" means the lowest value where the pin is guaranteed to be read as high.
3. Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 200 mA.
   2] The sum of all IOL, for ports B0 - B7, D0 - D7, and XTAL2, should not exceed 100 mA.
   3] The sum of all IOL, for ports A0 - A7, E0 - E2, and C0 - C7 should not exceed 100 mA.
4. Although each I/O port can source more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient),the following must be observed:
   1] The sum of all IOH,for all ports, should not exceed 200 mA.
   2] The sum of all IOH,for ports B0 - B7, D0 - D7, and XTAL2,should not exceed 100 mA.
   3] The sum of all IOH,for ports A0 - A7, E0 - E2, and C0 - C7 should not exceed 100 mA.
5. Minimum V_{CC} for Power-down is 2.5V.

External Clock Drive Waveforms

Figure 86. External Clock Drive Waveforms



External Clock Drive

Table 95. External Clock Drive

Symbol	Parameter	V_{CC} = 2.7 - 5.5V		V_{CC} = 4.5 - 5.5V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	8	0	16	MHz
t_{CLCL}	Clock Period	125		62.5		ns
t_{CHCX}	High Time	50		25		ns
t_{CLCX}	Low Time	50		25		ns
t_{CLCH}	Rise Time		1.6		0.5	µs
t_{CHCL}	Fall Time		1.6		0.5	µs
Δt_{CLCL}	Change in period from one clock cycle to the next(1)		2		2	%

Note: 1. Refer to “External Clock” on page 40 for details.

Table 96. External RC Oscillator, Typical Frequencies (V_{CC} = 5V)

R [kΩ](1)	C [pF]	f(2)
33	22	650 kHz
10	22	2.0 MHz

Notes: 1. R should be in the range 3 kΩ - 100 kΩ, and C should be at least 20 pF. The C values given in the table includes pin capacitance. This will vary with package type.
2. The frequency will vary with package type and board layout.

SPI Timing Characteristics

See Figure 87 and Figure 88 for details.

Table 97. SPI Timing Parameters

	Description	Mode	Min	Typ	Max	
1	SCK period	Master		See Table 58		
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		$0.5 \bullet t_{SCK}$		ns
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		
10	SCK period	Slave	$4 \bullet t_{ck}$			
11	SCK high/low(1)	Slave	$2 \bullet t_{ck}$			
12	Rise/Fall time	Slave			1.6	µs
13	Setup	Slave	10			
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		ns
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	$\overline{SS}$ low to SCK	Salve	$2 \bullet t_{ck}$			

Note: 1. In SPI Programming mode the minimum SCK high/low period is:
- 2 t_{CLCL} for f_{CK} < 12 MHz
- 3 t_{CLCL} for f_{CK} >12 MHz

Figure 87. SPI Interface Timing Requirements (Master Mode)

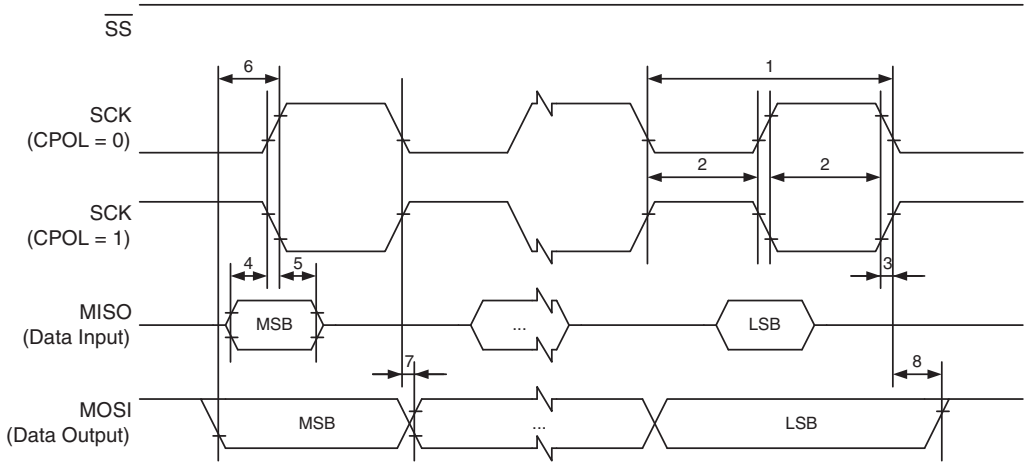
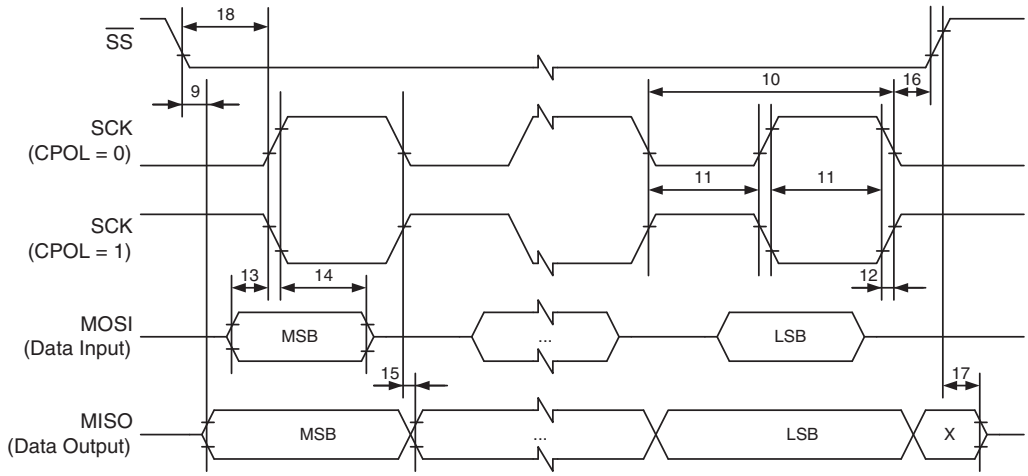


Figure 88. SPI Interface Timing Requirements (Slave Mode)



External Data Memory Timing

Table 98. External Data Memory Characteristics, 4.5 - 5.5 Volts, No Wait-state

			8 MHz Oscillator		Variable Oscillator		
	Symbol	Parameter	Min	Max	Min	Max	Unit
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	16	MHz
1	t_{LHLL}	ALE Pulse Width	115		$1.0t_{CLCL}$-10		ns
2	t_{AVLL}	Address Valid A to ALE Low	57.5		$0.5t_{CLCL}$-5(1)		ns
3a	t_{LLAX_ST}	Address Hold After ALE Low, write access	5		5		ns
3b	t_{LLAX_LD}	Address Hold after ALE Low, read access	5		5		ns
4	t_{AVLLC}	Address Valid C to ALE Low	57.5		$0.5t_{CLCL}$-5(1)		ns
5	t_{AVRL}	Address Valid to RD Low	115		$1.0t_{CLCL}$-10		ns
6	t_{AVWL}	Address Valid to WR Low	115		$1.0t_{CLCL}$-10		ns
7	t_{LLWL}	ALE Low to WR Low	47.5	67.5	$0.5t_{CLCL}$-15(2)	$0.5t_{CLCL}$+5(2)	ns
8	t_{LLRL}	ALE Low to RD Low	47.5	67.5	$0.5t_{CLCL}$-15(2)	$0.5t_{CLCL}$+5(2)	ns
9	t_{DVRH}	Data Setup to RD High	40		40		ns
10	t_{RLDV}	Read Low to Data Valid		75		$1.0t_{CLCL}$-50	ns
11	t_{RHDX}	Data Hold After RD High	0		0		ns
12	t_{RLRH}	RD Pulse Width	115		$1.0t_{CLCL}$-10		ns
13	t_{DVWL}	Data Setup to WR Low	42.5		$0.5t_{CLCL}$-20(1)		ns
14	t_{WHDX}	Data Hold After WR High	115		$1.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid to WR High	125		$1.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	115		$1.0t_{CLCL}$-10		ns

Notes: 1. This assumes 50% clock duty cycle. The half period is actually the high time of the external clock, XTAL1.
2. This assumes 50% clock duty cycle. The half period is actually the low time of the external clock, XTAL1.

Table 99. External Data Memory Characteristics, 4.5 - 5.5 Volts, 1 Cycle Wait-state

			8 MHz Oscillator		Variable Oscillator		
	Symbol	Parameter	Min	Max	Min	Max	Unit
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	16	MHz
10	t_{RLDV}	Read Low to Data Valid		200		$2.0t_{CLCL}$-50	ns
12	t_{RLRH}	RD Pulse Width	240		$2.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid to WR High	240		$2.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	240		$2.0t_{CLCL}$-10		ns

Table 100. External Data Memory Characteristics, 4.5 - 5.5 Volts, SRWn1 = 1, SRWn0 = 0

			4 MHz Oscillator		Variable Oscillator		
	Symbol	Parameter	Min	Max	Min	Max	Unit
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	16	MHz
10	t_{RLDV}	Read Low to Data Valid		325		$3.0t_{CLCL}$-50	ns
12	t_{RLRH}	RD Pulse Width	365		$3.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid to WR High	375		$3.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	365		$3.0t_{CLCL}$-10		ns

Table 101. External Data Memory Characteristics, 4.5 - 5.5 Volts, SRWn1 = 1, SRWn0 = 1

			4 MHz Oscillator		Variable Oscillator		
	Symbol	Parameter	Min	Max	Min	Max	Unit
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	16	MHz
10	t_{RLDV}	Read Low to Data Valid		325		$3.0t_{CLCL}$-50	ns
12	t_{RLRH}	RD Pulse Width	365		$3.0t_{CLCL}$-10		ns
14	t_{WHDX}	Data Hold After WR High	240		$2.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid to WR High	375		$3.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	365		$3.0t_{CLCL}$-10		ns

Table 102. External Data Memory Characteristics, 2.7 - 5.5 Volts, No Wait-state

			4 MHz Oscillator		Variable Oscillator		
	Symbol	Parameter	Min	Max	Min	Max	Unit
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	8	MHz
1	t_{LHLL}	ALE Pulse Width	235		t_{CLCL}-15		ns
2	t_{AVLL}	Address Valid A to ALE Low	115		$0.5t_{CLCL}$-10[1]		ns
3a	t_{LLAX_ST}	Address Hold After ALE Low, write access	5		5		ns
3b	t_{LLAX_LD}	Address Hold after ALE Low, read access	5		5		ns
4	t_{AVLLC}	Address Valid C to ALE Low	115		$0.5t_{CLCL}$-10[1]		ns
5	t_{AVRL}	Address Valid to RD Low	235		$1.0t_{CLCL}$-15		ns
6	t_{AVWL}	Address Valid to WR Low	235		$1.0t_{CLCL}$-15		ns
7	t_{LLWL}	ALE Low to WR Low	115	130	$0.5t_{CLCL}$-10[2]	$0.5t_{CLCL}$+5[2]	ns
8	t_{LLRL}	ALE Low to RD Low	115	130	$0.5t_{CLCL}$-10[2]	$0.5t_{CLCL}$+5[2]	ns
9	t_{DVRH}	Data Setup to RD High	45		45		ns
10	t_{RLDV}	Read Low to Data Valid		190		$1.0t_{CLCL}$-60	ns
11	t_{RHDX}	Data Hold After RD High	0		0		ns

Table 102. External Data Memory Characteristics, 2.7 - 5.5 Volts, No Wait-state (Continued)

			4 MHz Oscillator		Variable Oscillator		
	Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
12	t_{RLRH}	RD Pulse Width	235		$1.0t_{CLCL}$-15		ns
13	t_{DVWL}	Data Setup to WR Low	105		$0.5t_{CLCL}$-20[1]		ns
14	t_{WHDX}	Data Hold After WR High	235		$1.0t_{CLCL}$-15		ns
15	t_{DVWH}	Data Valid to WR High	250		$1.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	235		$1.0t_{CLCL}$-15		ns

Notes: 1. This assumes 50% clock duty cycle. The half period is actually the high time of the external clock, XTAL1.
2. This assumes 50% clock duty cycle. The half period is actually the low time of the external clock, XTAL1.

Table 103. External Data Memory Characteristics, 2.7 - 5.5 Volts, SRWn1 = 0, SRWn0 = 1

			4 MHz Oscillator		Variable Oscillator		
	Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	8	MHz
10	t_{RLDV}	Read Low to Data Valid		440		$2.0t_{CLCL}$-60	ns
12	t_{RLRH}	RD Pulse Width	485		$2.0t_{CLCL}$-15		ns
15	t_{DVWH}	Data Valid to WR High	500		$2.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	485		$2.0t_{CLCL}$-15		ns

Table 104. External Data Memory Characteristics, 2.7 - 5.5 Volts, SRWn1 = 1, SRWn0 = 0

			4 MHz Oscillator		Variable Oscillator		
	Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	8	MHz
10	t_{RLDV}	Read Low to Data Valid		690		$3.0t_{CLCL}$-60	ns
12	t_{RLRH}	RD Pulse Width	735		$3.0t_{CLCL}$-15		ns
15	t_{DVWH}	Data Valid to WR High	750		$3.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	735		$3.0t_{CLCL}$-15		ns

Table 105. External Data Memory Characteristics, 2.7 - 5.5 Volts, SRWn1 = 1, SRWn0 = 1

			4 MHz Oscillator		Variable Oscillator		
	Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
0	$1/t_{CLCL}$	Oscillator Frequency			0.0	8	MHz
10	t_{RLDV}	Read Low to Data Valid		690		$3.0t_{CLCL}$-60	ns
12	t_{RLRH}	RD Pulse Width	735		$3.0t_{CLCL}$-15		ns
14	t_{WHDX}	Data Hold After WR High	485		$2.0t_{CLCL}$-15		ns
15	t_{DVWH}	Data Valid to WR High	750		$3.0t_{CLCL}$		ns
16	t_{WLWH}	WR Pulse Width	735		$3.0t_{CLCL}$-15		ns

Figure 89. External Memory Timing (SRWn1 = 0, SRWn0 = 0

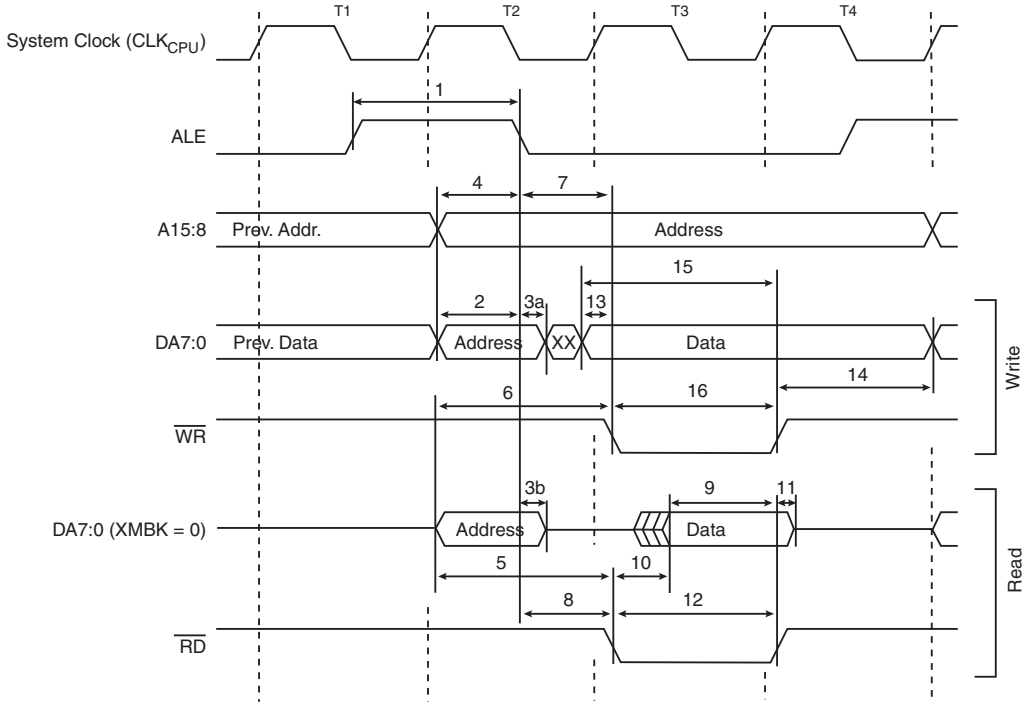


Figure 90. External Memory Timing (SRWn1 = 0, SRWn0 = 1)

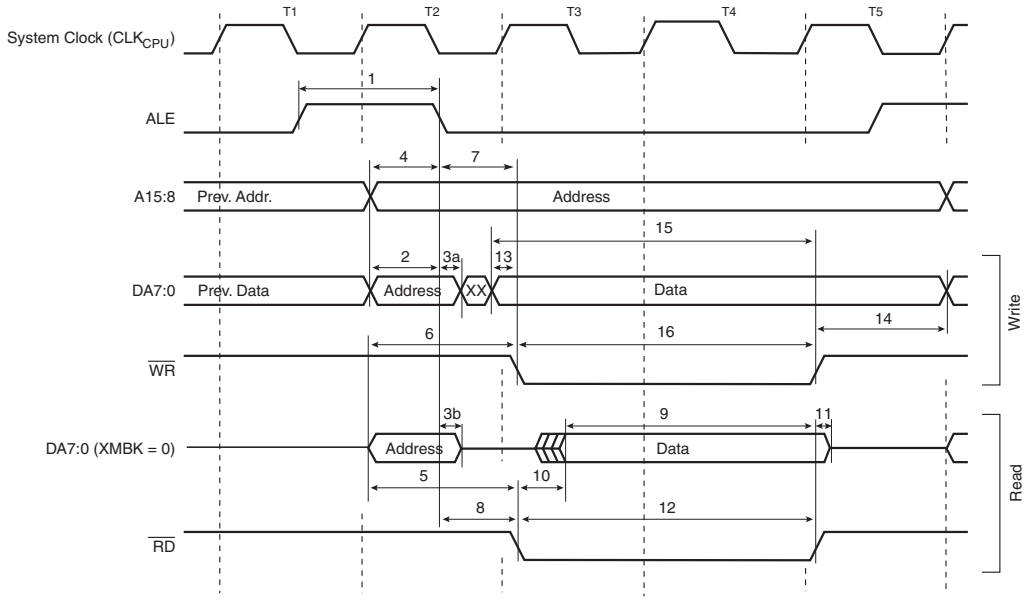


Figure 91. External Memory Timing (SRWn1 = 1, SRWn0 = 0)

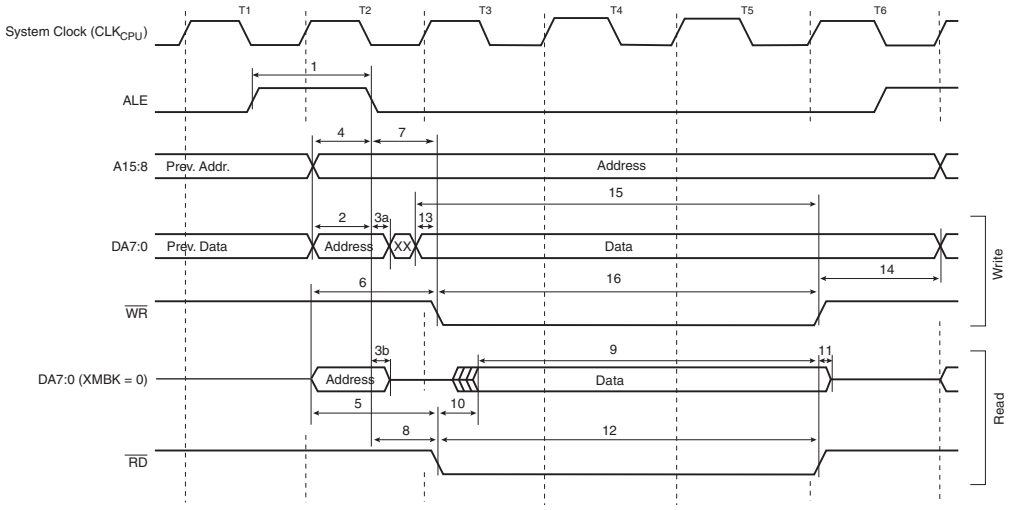
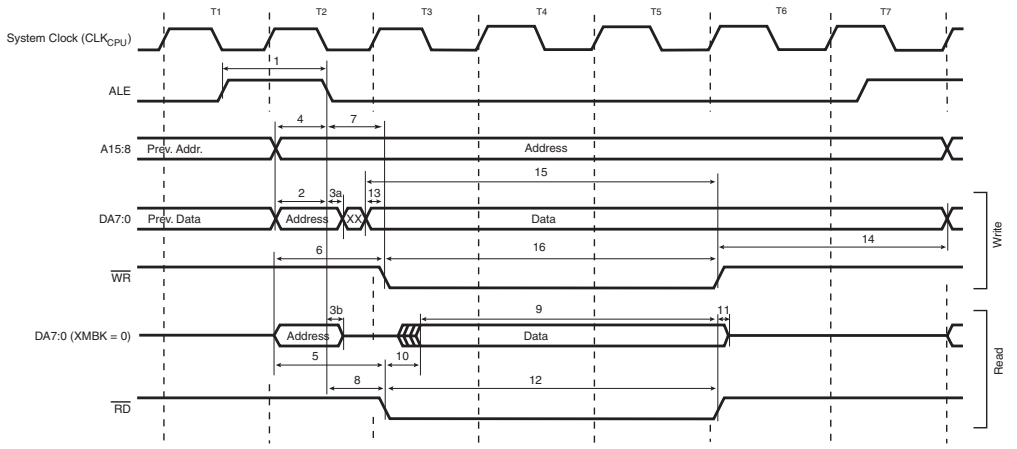


Figure 92. External Memory Timing (SRWn1 = 1, SRWn0 = 1)[1]



Note: 1. The ALE pulse in the last period (T4-T7) is only present if the next instruction accesses the RAM (internal or external).

ATmega8515 Typical Characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: Operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L*V_{CC}*f where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

Active Supply Current

Figure 93. Active Supply Current vs. Frequency (0.1 - 1.0 MHz)

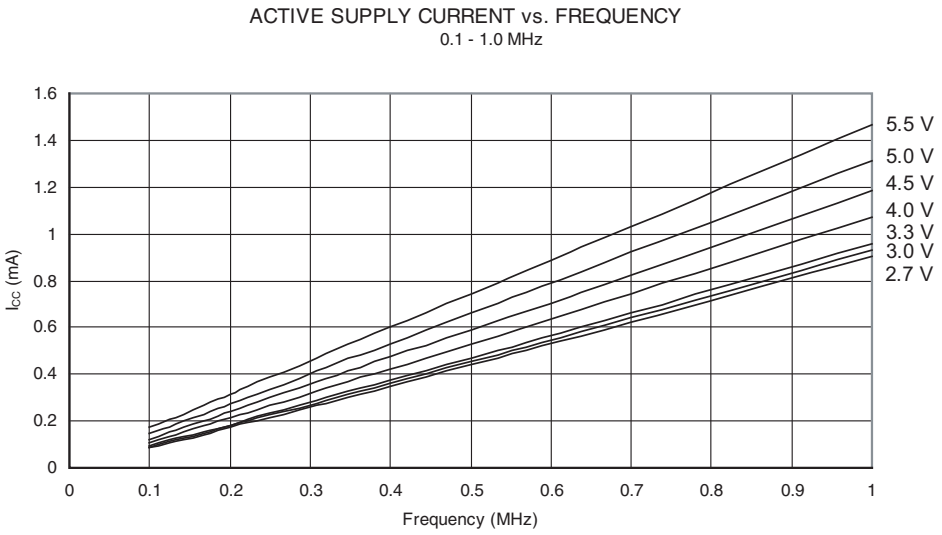


Figure 94. Active Supply Current vs. Frequency (1 - 20 MHz)

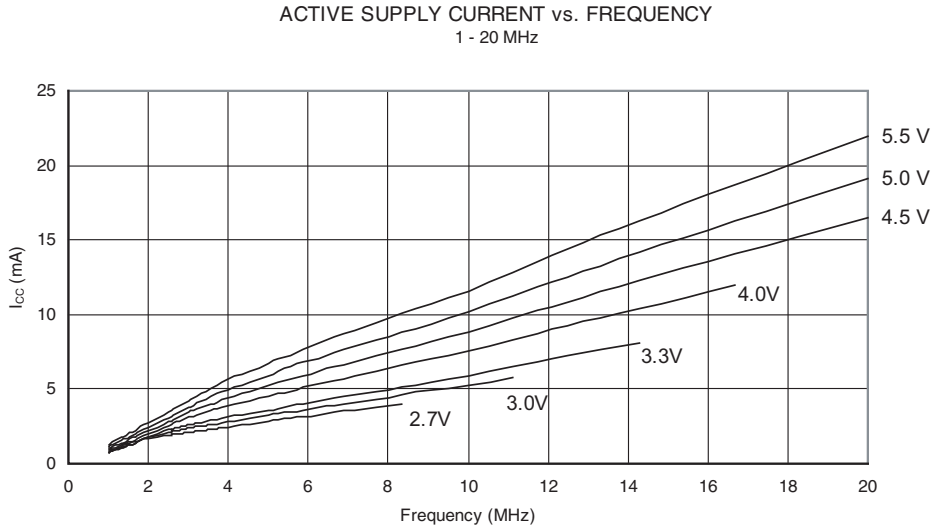


Figure 95. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

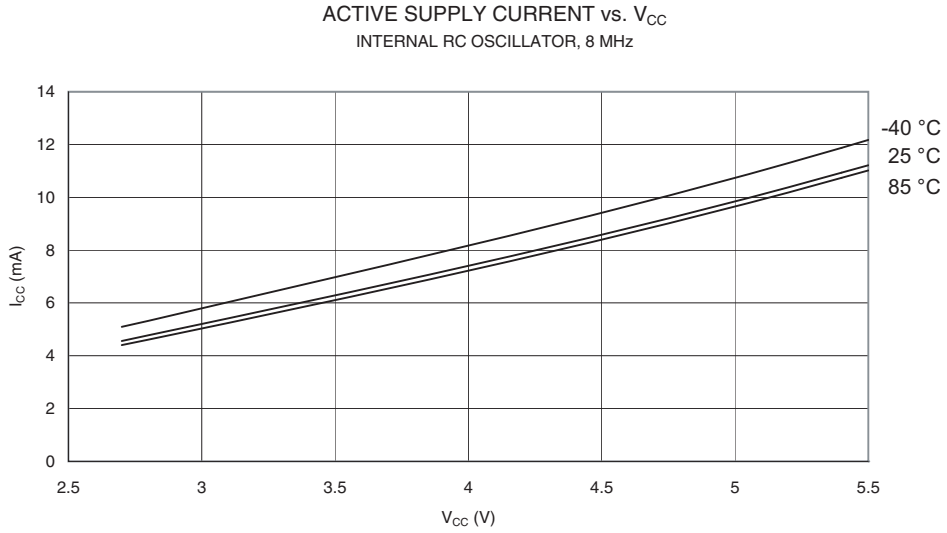


Figure 96. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

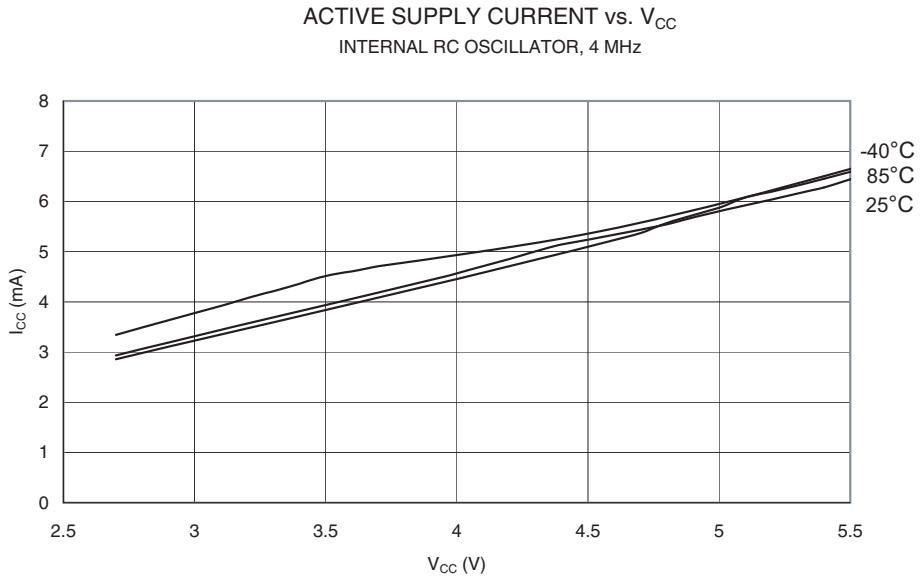


Figure 97. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

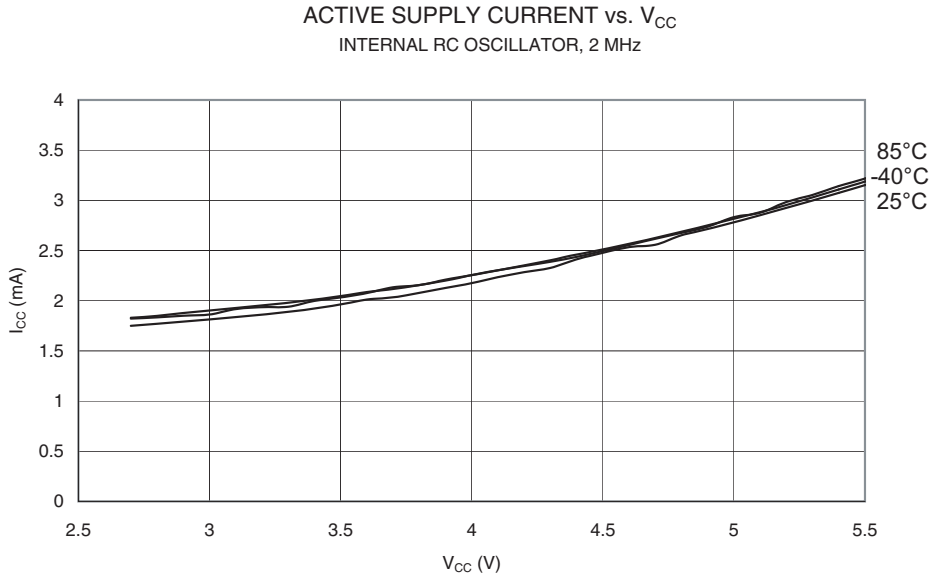


Figure 98. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

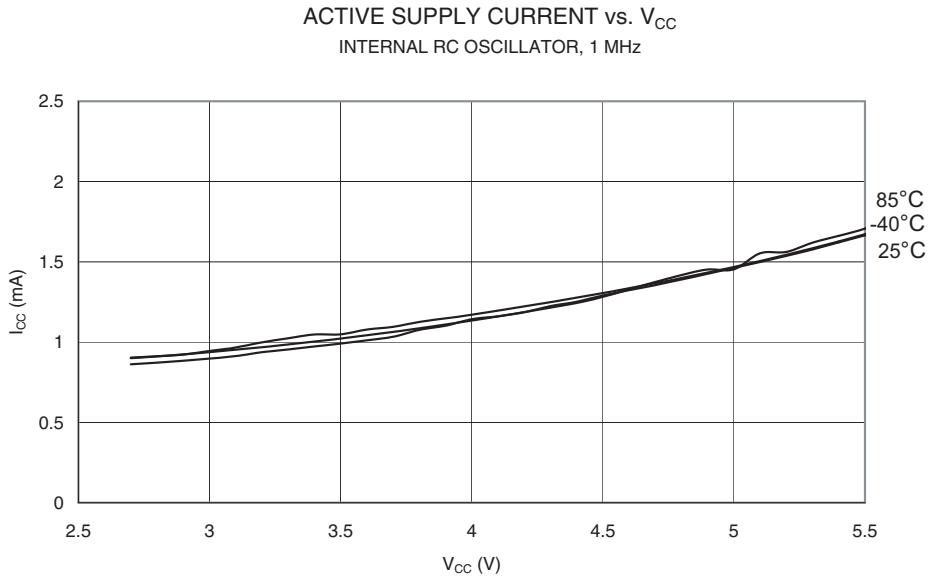
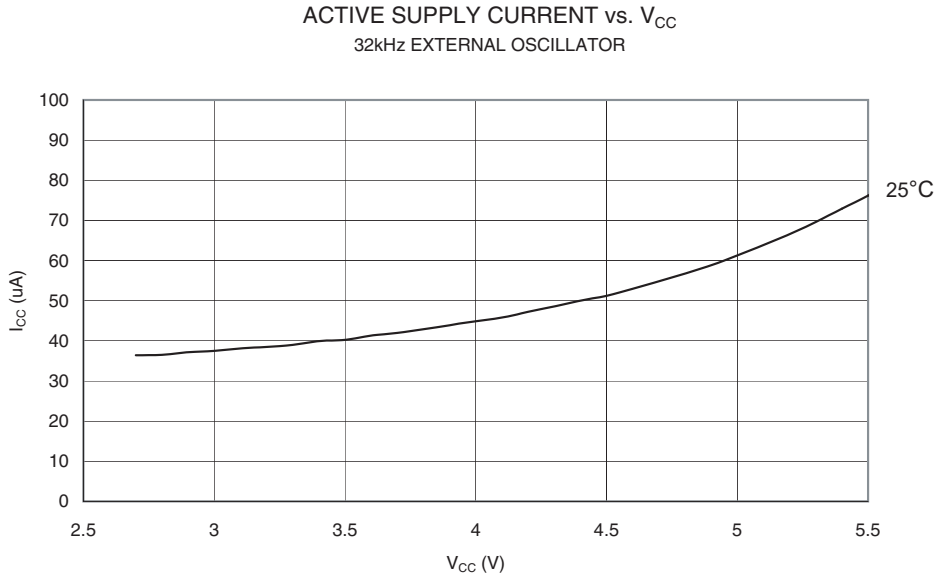


Figure 99. Active Supply Current vs. V_{CC} (32 kHz External Oscillator)



Idle Supply Current

Figure 100. Idle Supply Current vs. Frequency (0.1 - 1.0 MHz)

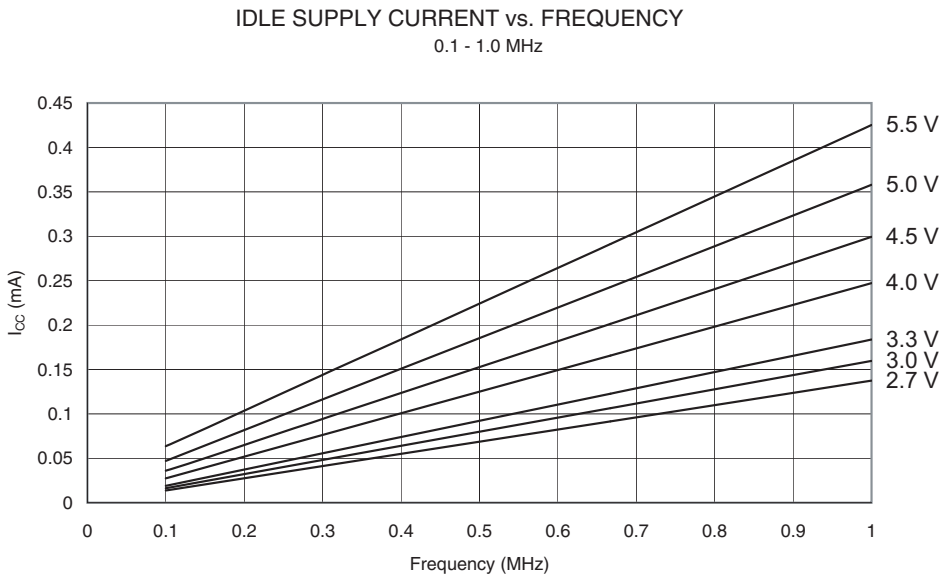


Figure 101. Idle Supply Current vs. Frequency (1 - 20 MHz)

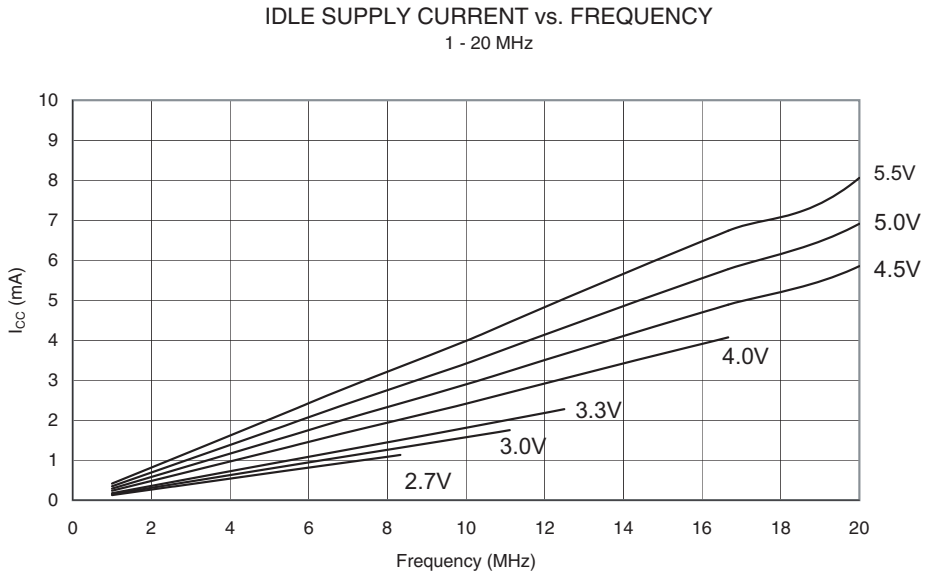


Figure 102. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

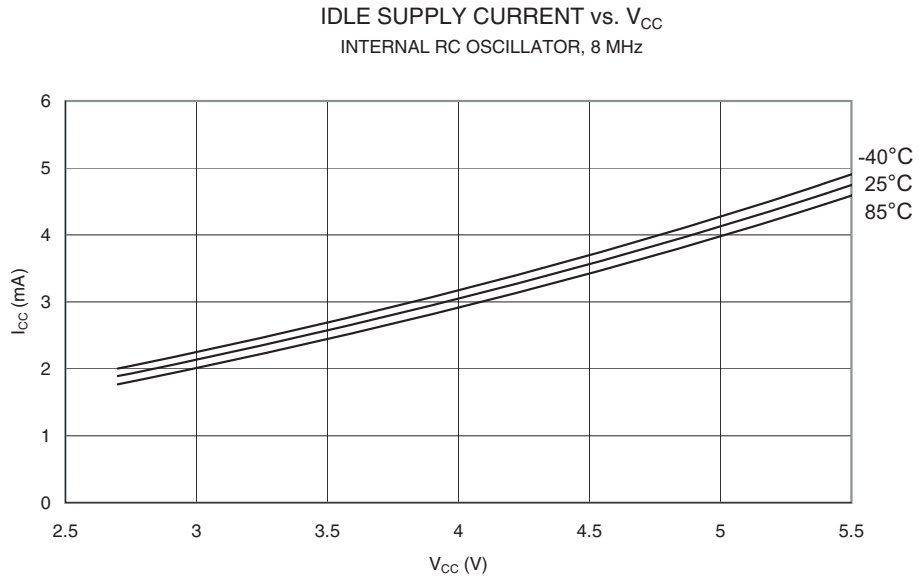


Figure 103. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 4 MHz)

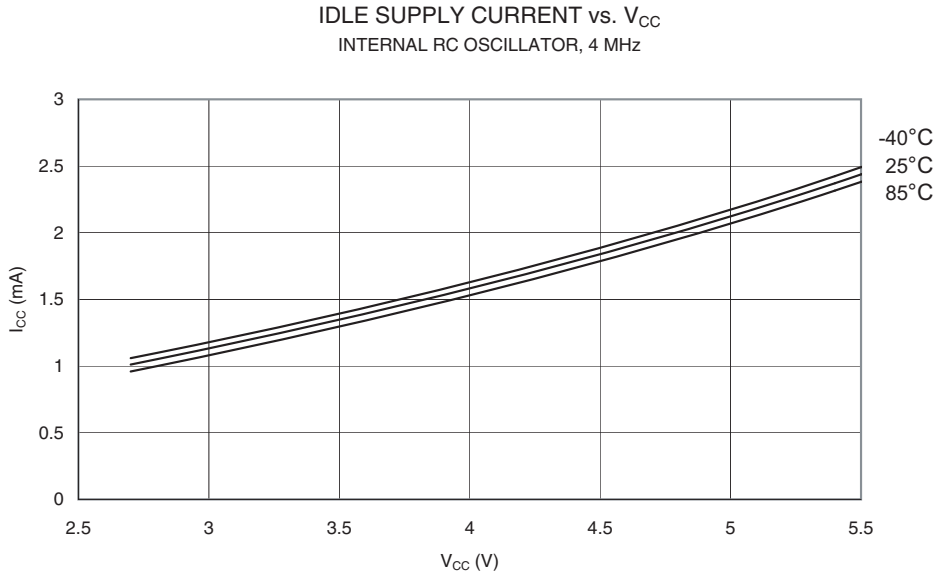


Figure 104. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 2 MHz)

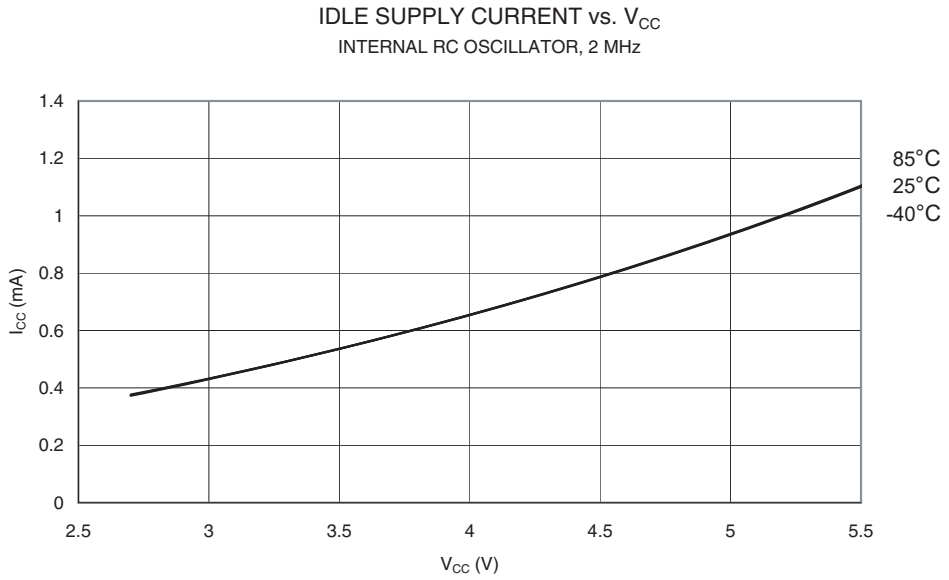


Figure 105. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

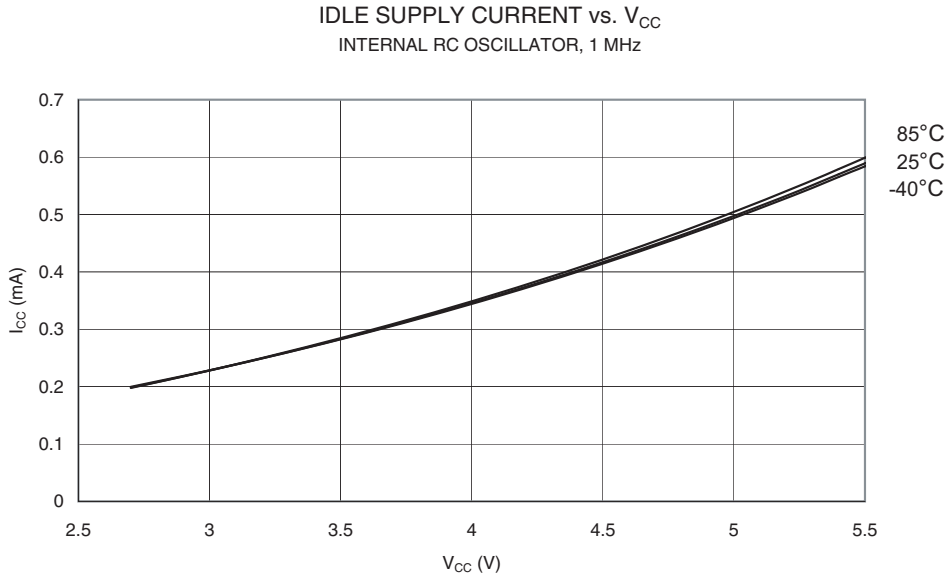
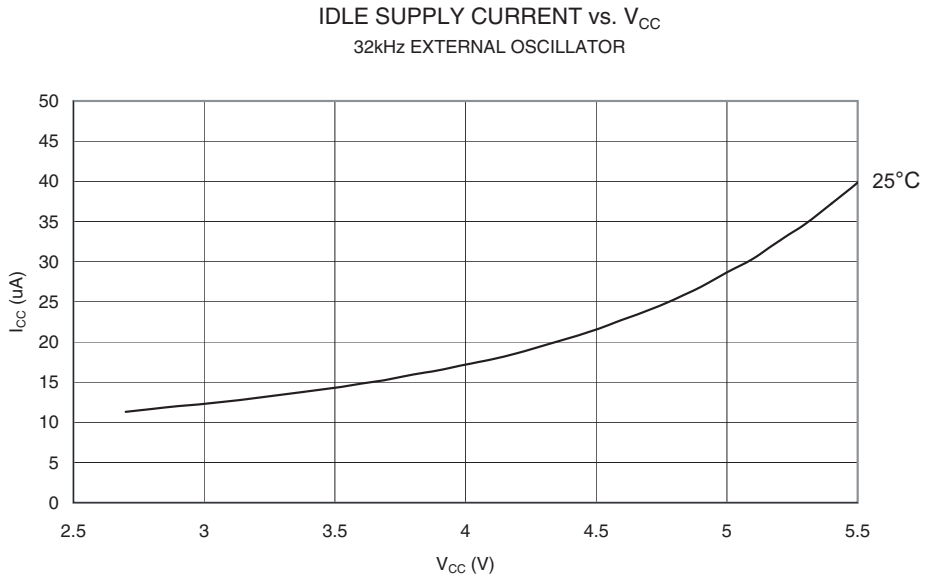


Figure 106. Idle Supply Current vs. V_{CC} (32 kHz External Oscillator)



Power-Down Supply Current

Figure 107. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

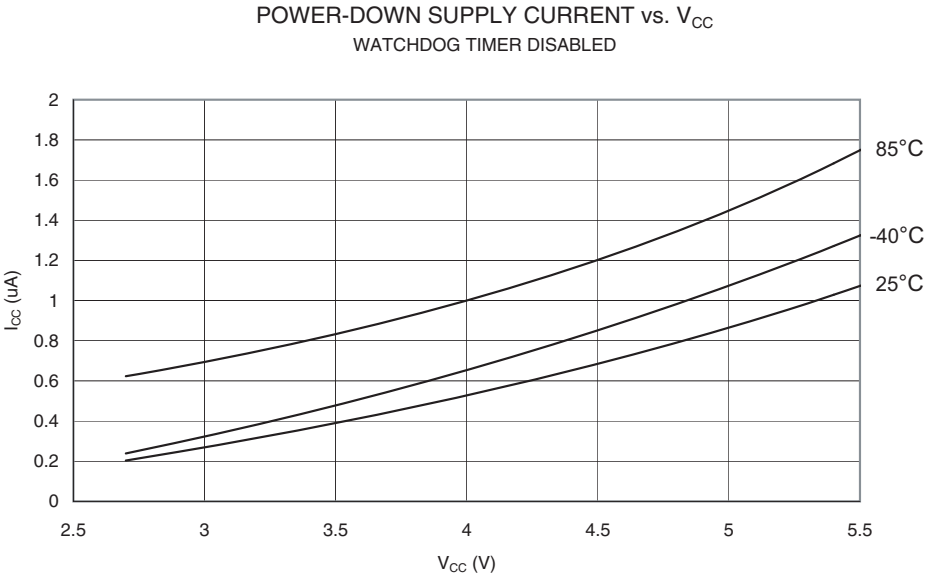
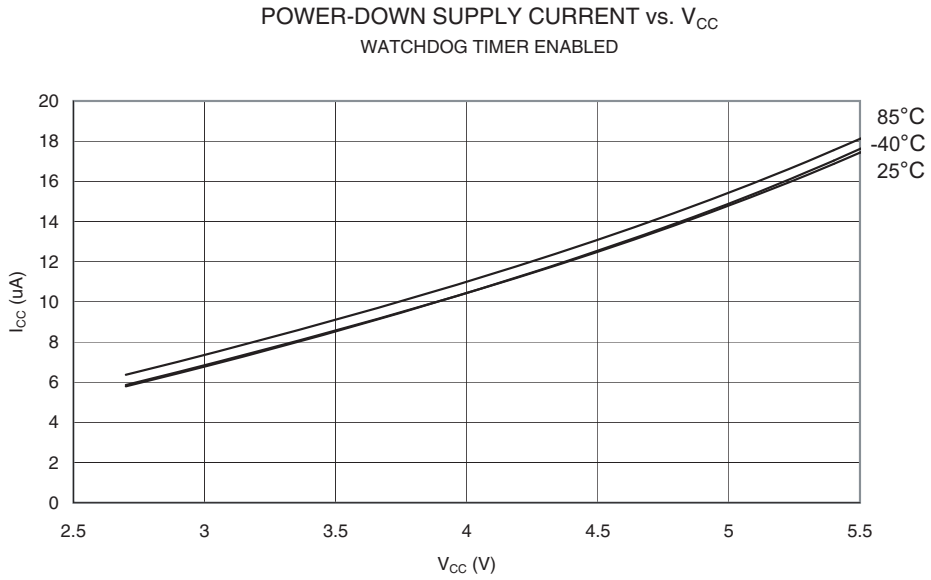


Figure 108. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



Standby Supply Current

Figure 109. Standby Supply Current vs. V_{CC} (455 kHz Resonator, Watchdog Timer Disabled)

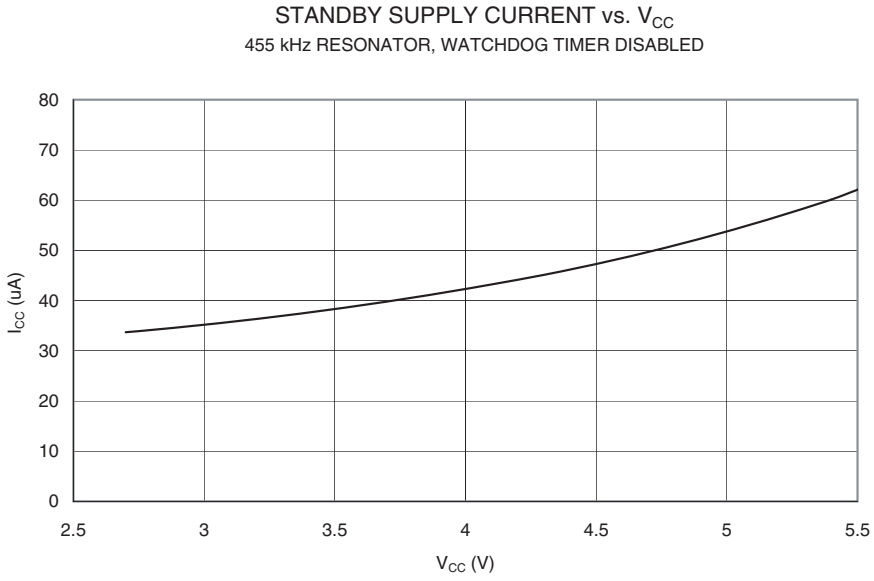


Figure 110. Standby Supply Current vs. V_{CC} (1 MHz Resonator, Watchdog Timer Disabled)

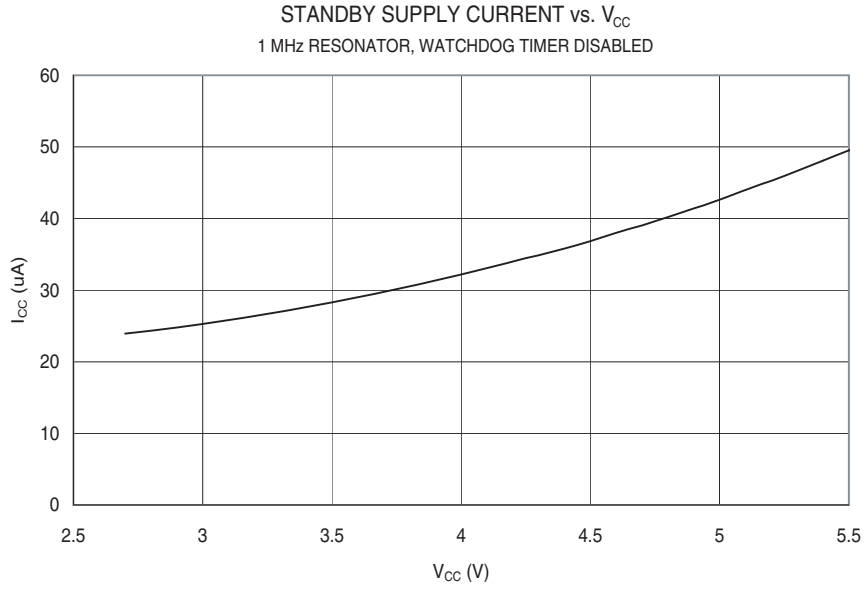


Figure 111. Standby Supply Current vs. V_{CC} (2 MHz Resonator, Watchdog Timer Disabled)

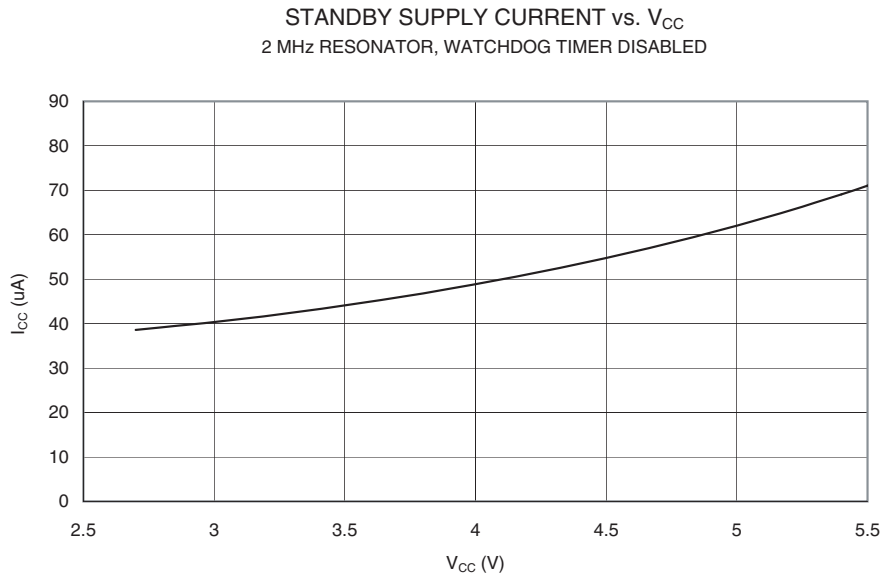


Figure 112. Standby Supply Current vs. V_{CC} (2 MHz XTAL, Watchdog Timer Disabled)

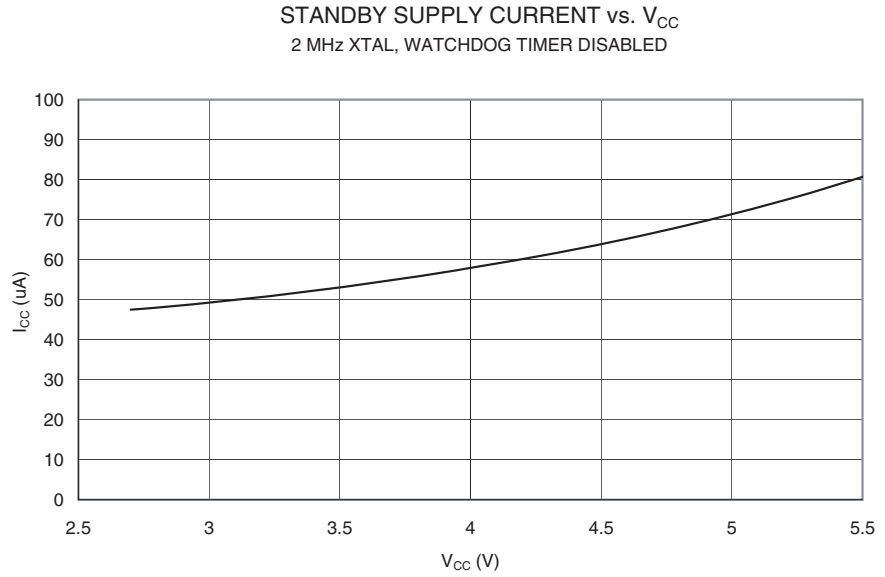


Figure 113. Standby Supply Current vs. V_{CC} (4 MHz Resonator, Watchdog Timer Disabled)

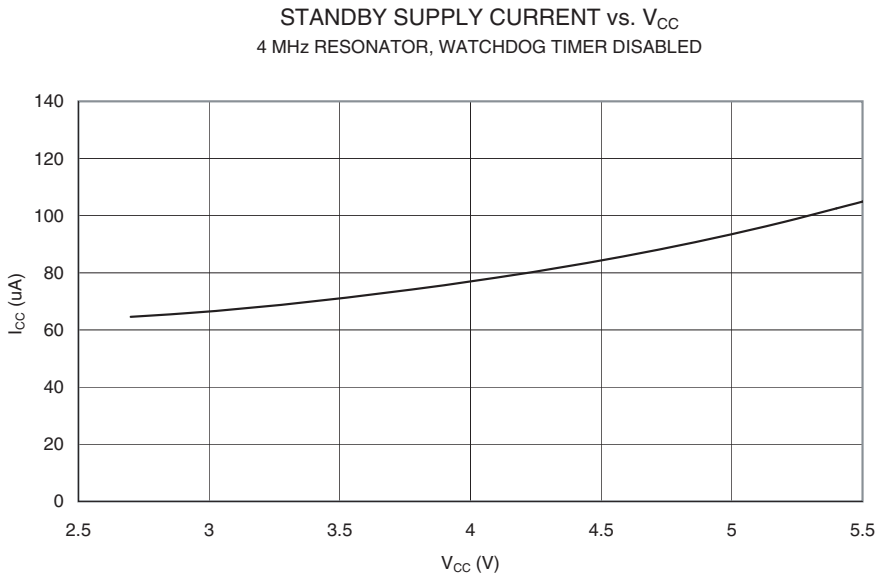


Figure 114. Standby Supply Current vs. V_{CC} (4 MHz XTAL, Watchdog Timer Disabled)

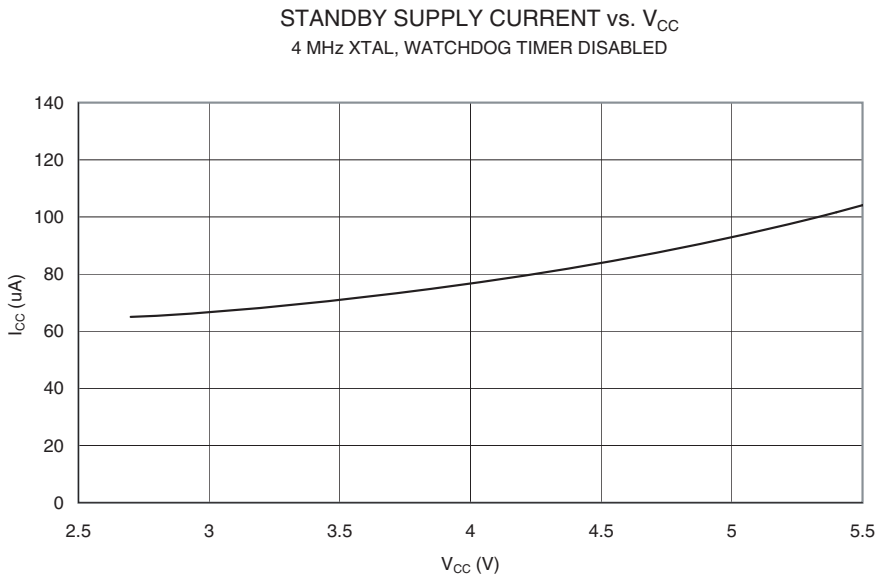


Figure 115. Standby Supply Current vs. V_{CC} (6 MHz Resonator, Watchdog Timer Disabled)

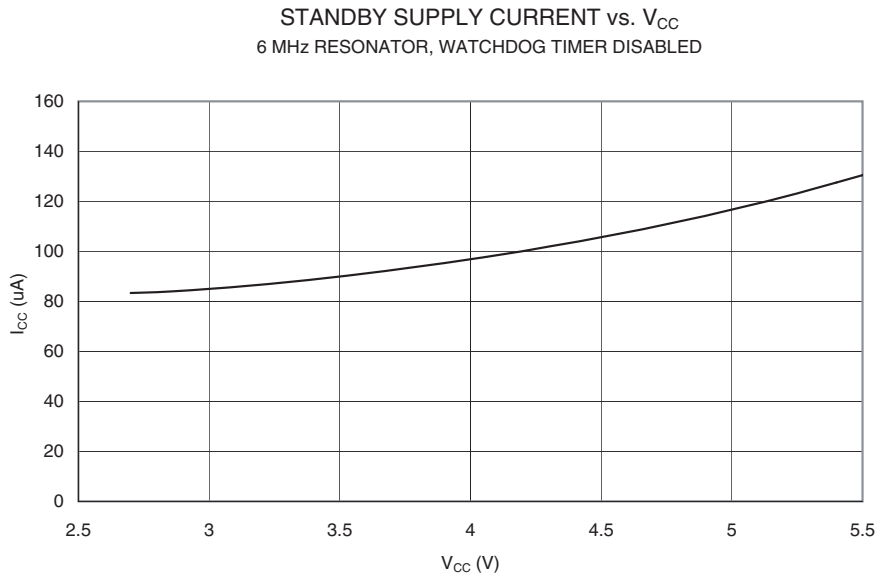
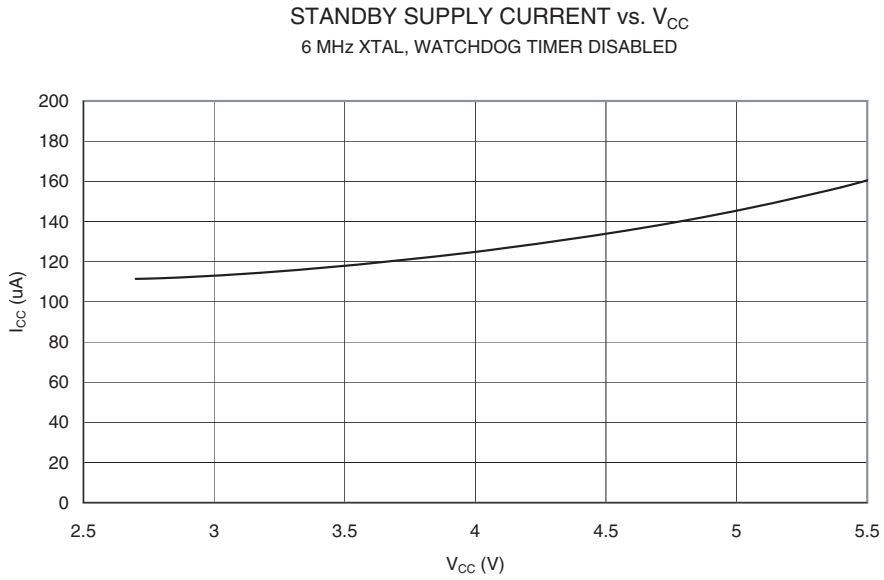


Figure 116. Standby Supply Current vs. V_{CC} (6 MHz XTAL, Watchdog Timer Disabled)



Pin Pull-up

Figure 117. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

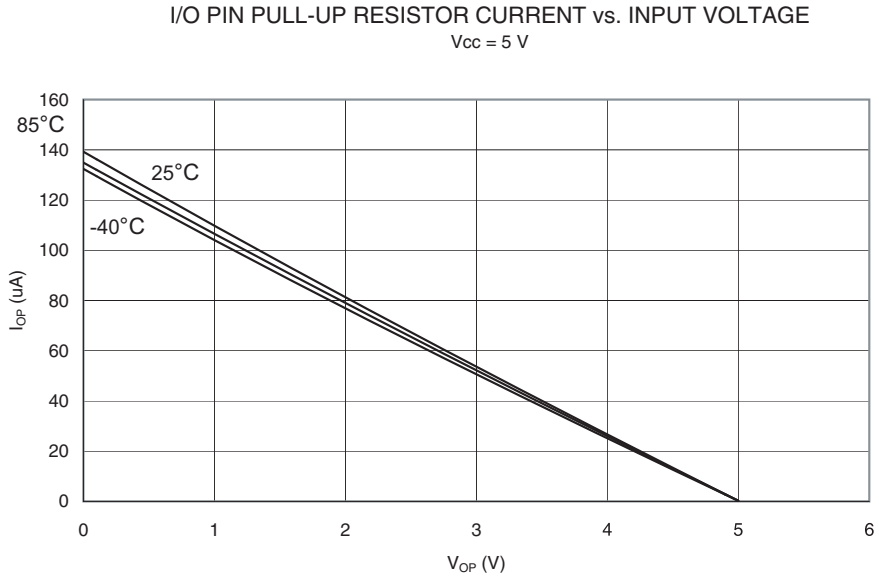


Figure 118. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

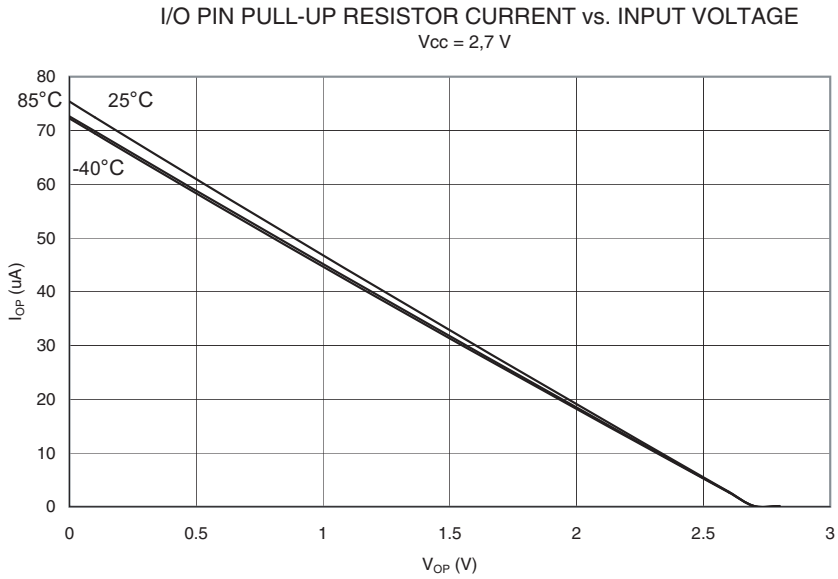


Figure 119. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)

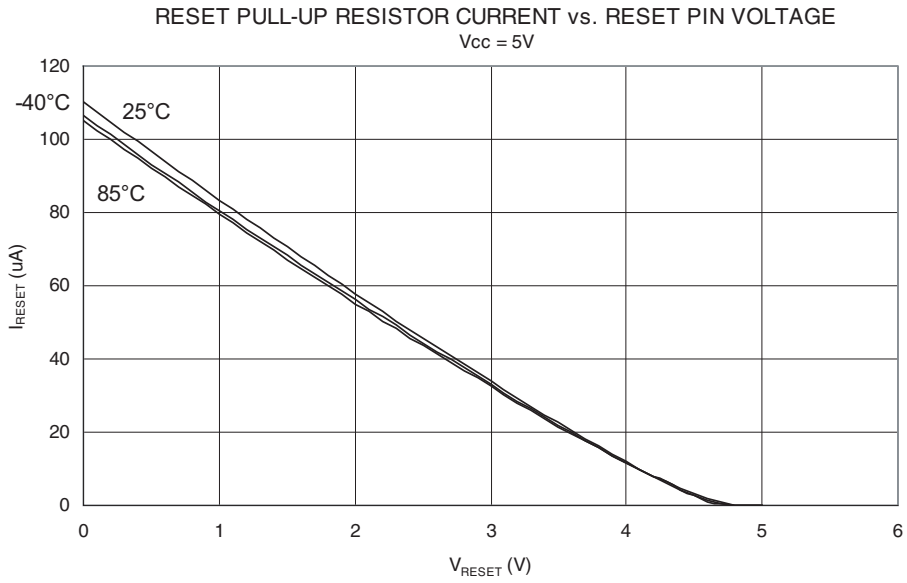
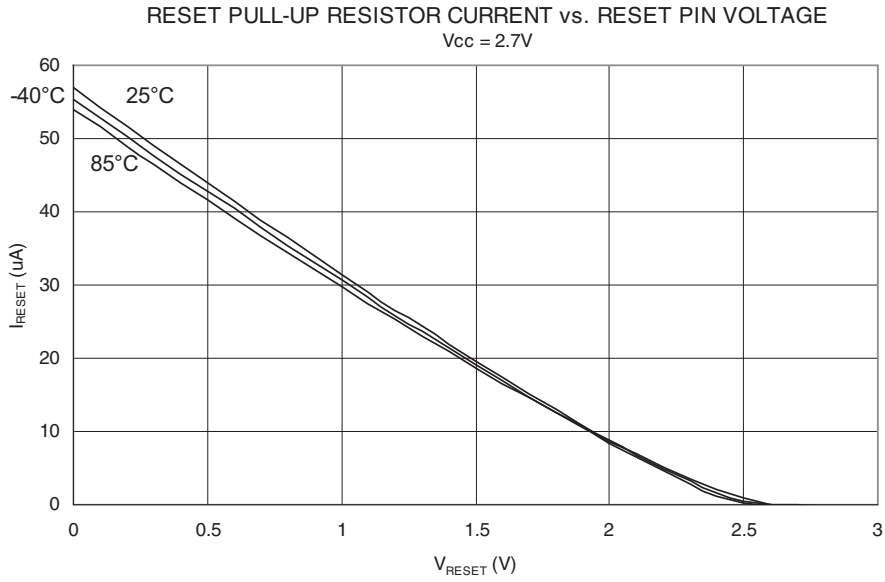


Figure 120. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)



Pin Driver Strength

Figure 121. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

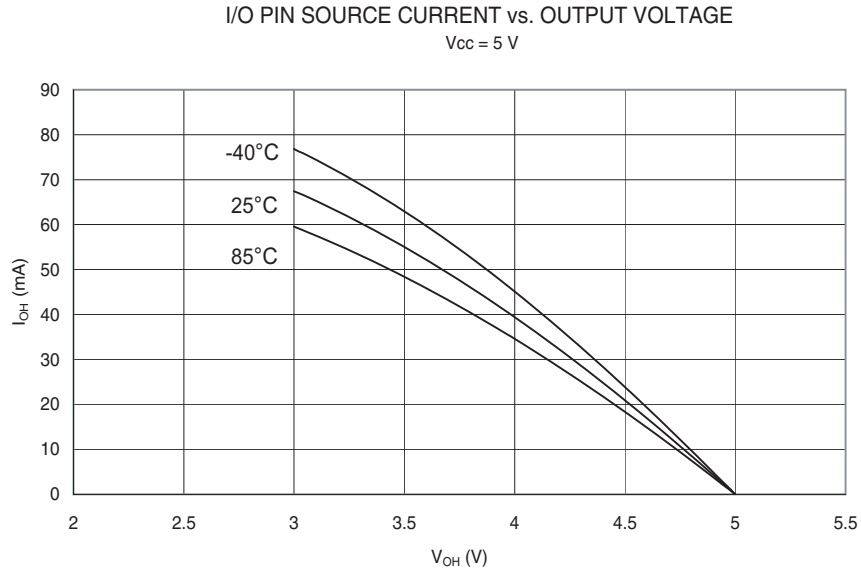


Figure 122. I/O Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

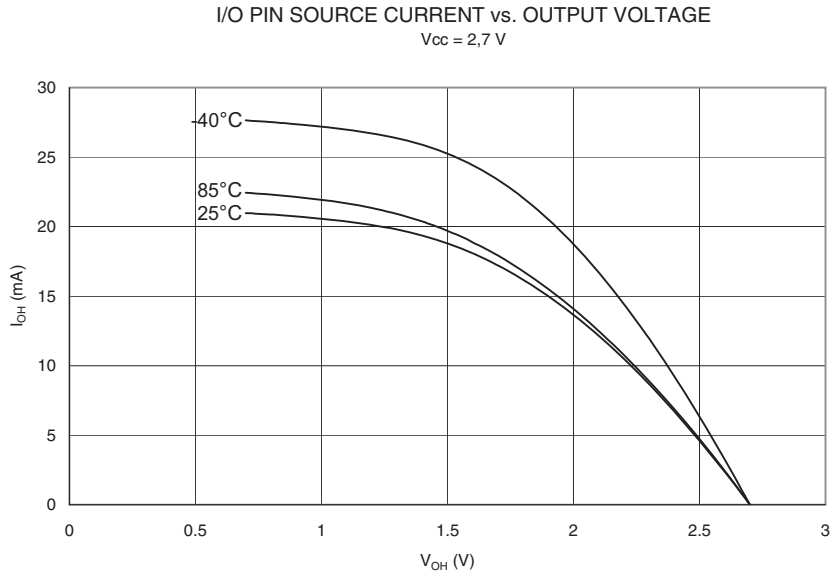


Figure 123. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

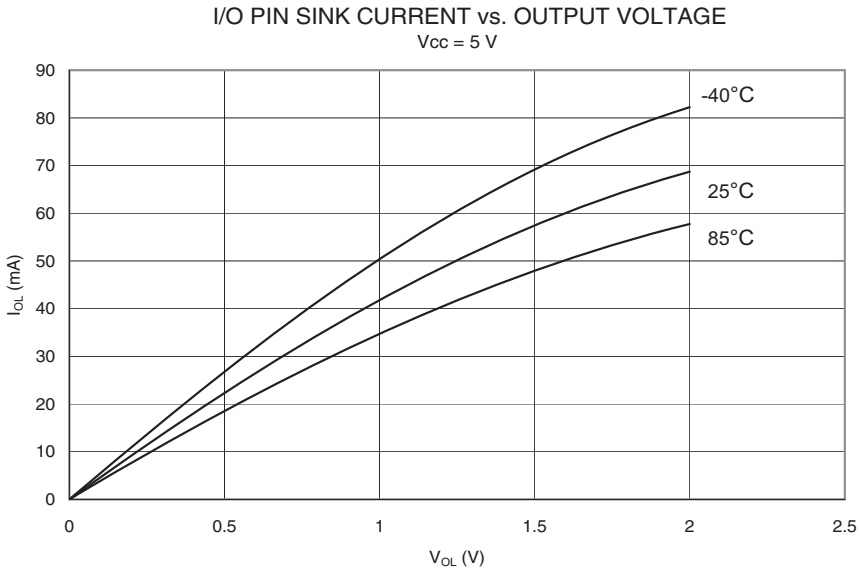
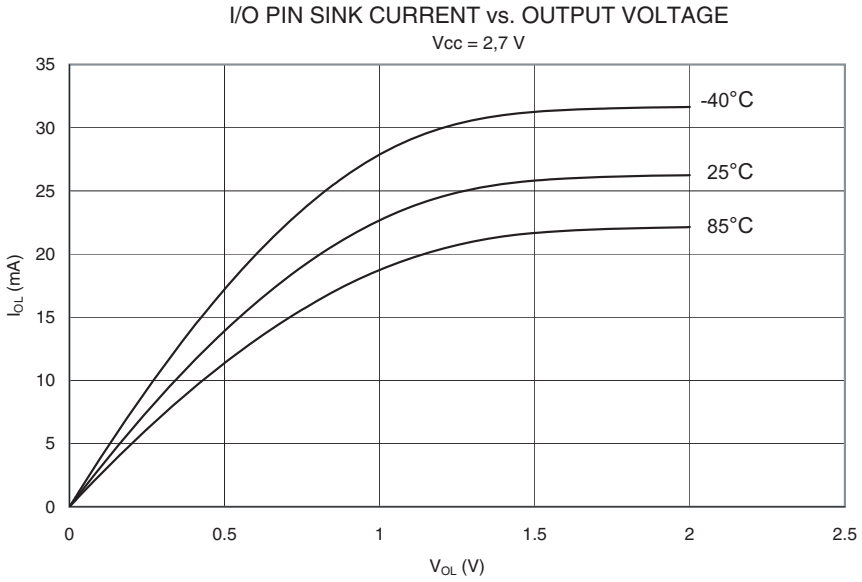


Figure 124. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)



Pin Thresholds And Hysteresis

Figure 125. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read As '1')

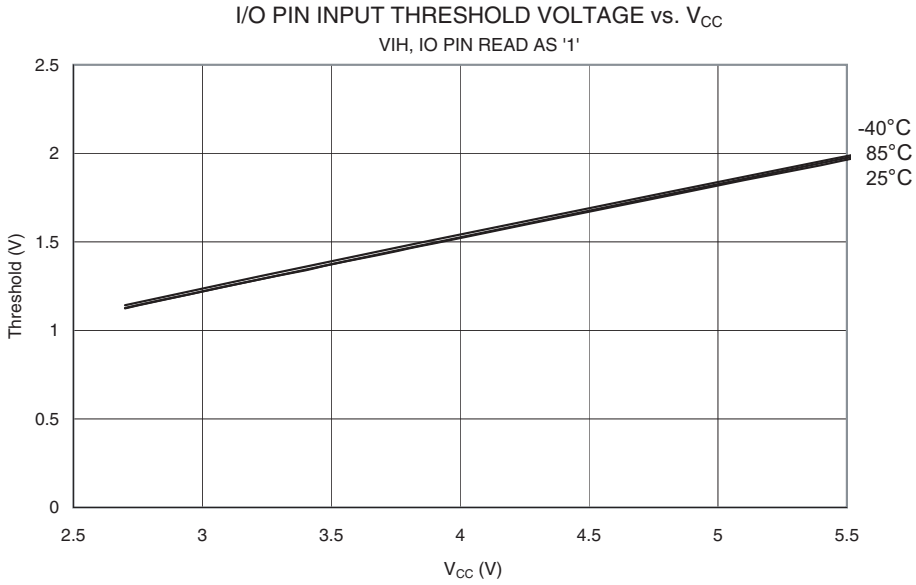


Figure 126. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read As '0')

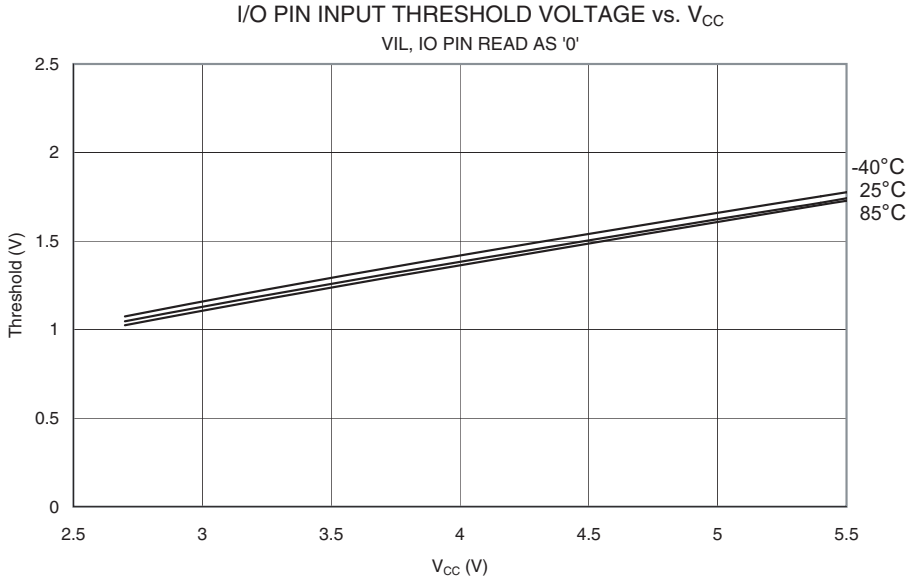


Figure 127. I/O Pin Input Hysteresis vs. V_{CC}

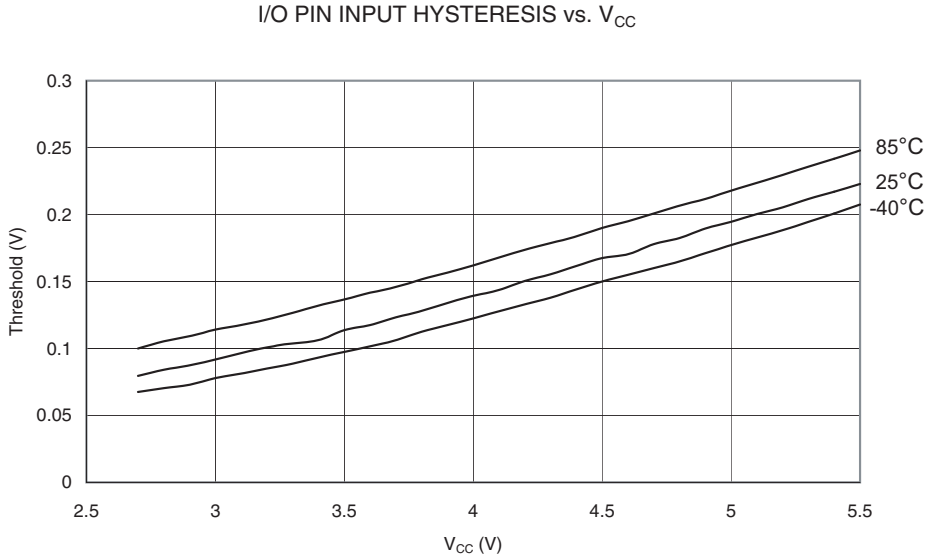


Figure 128. Reset Input Threshold Voltage vs. V_{CC} (V_{IH}, Reset Pin Read As '1')

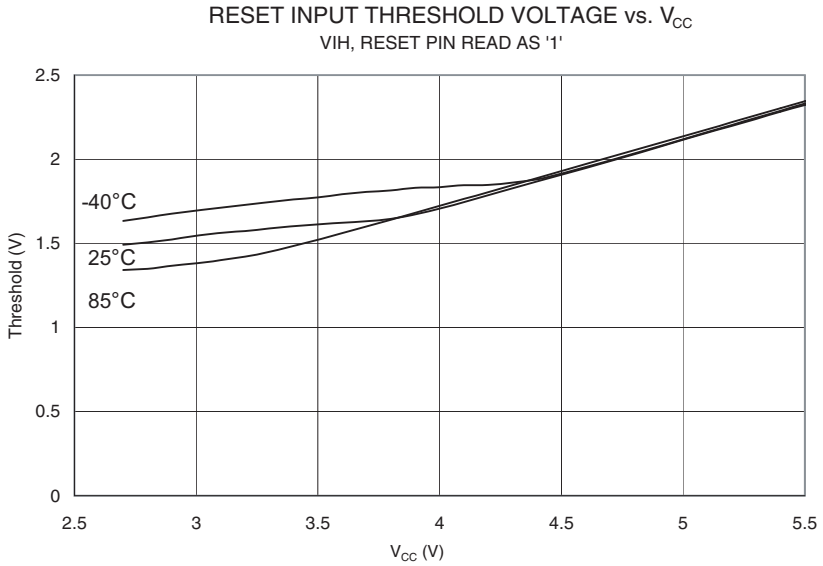


Figure 129. Reset Input Threshold Voltage vs. V_{CC} (V_{IL}, Reset Pin Read As '0')

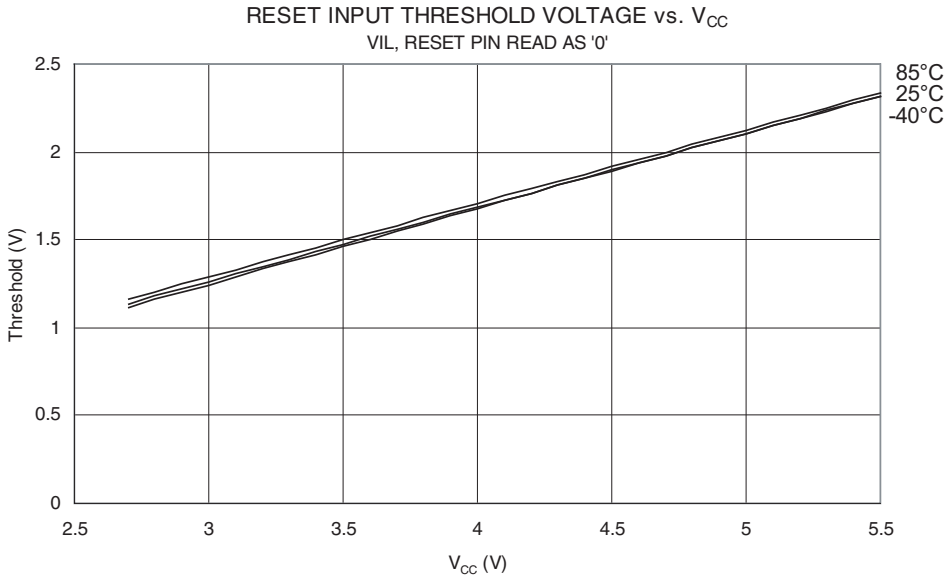
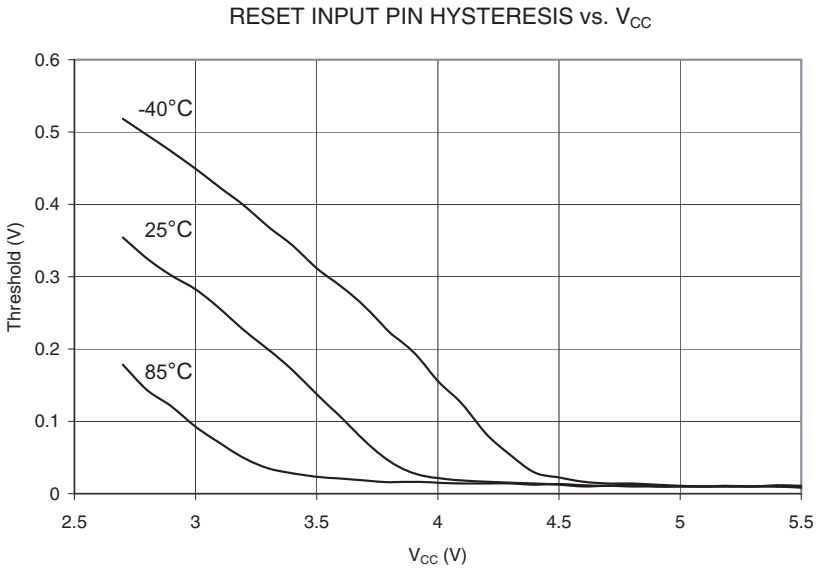


Figure 130. Reset Input Pin Hysteresis vs. V_{CC}



BOD Thresholds And Analog Comparator Offset

Figure 131. BOD Thresholds vs. Temperature (BOD Level is 4.0V)

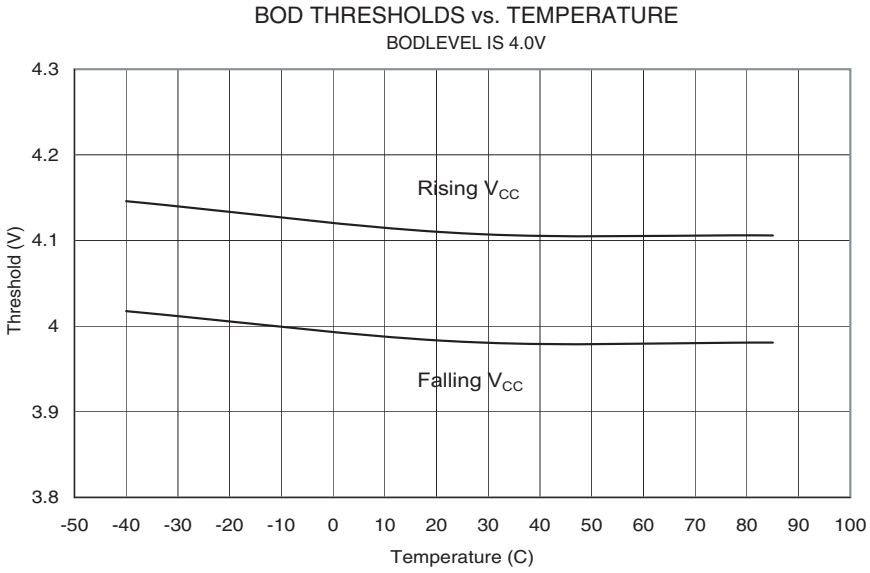


Figure 132. BOD Thresholds vs. Temperature (BOD Level is 2.7V)

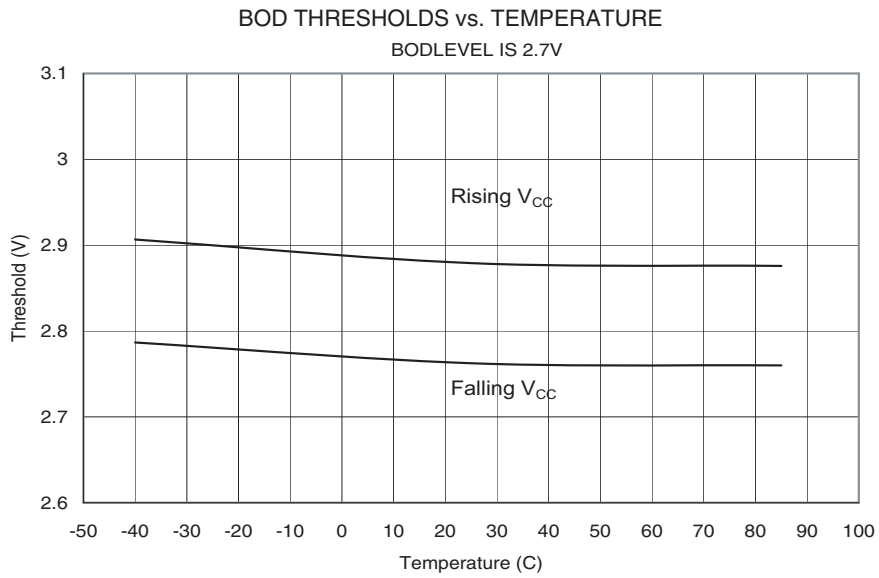


Figure 133. Bandgap Voltage vs. V_{CC}

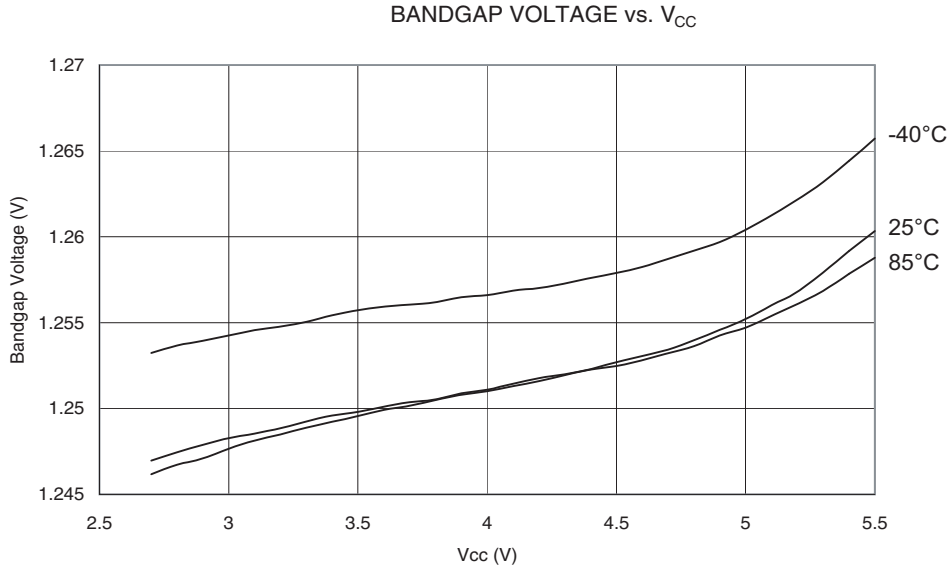


Figure 134. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 5V)

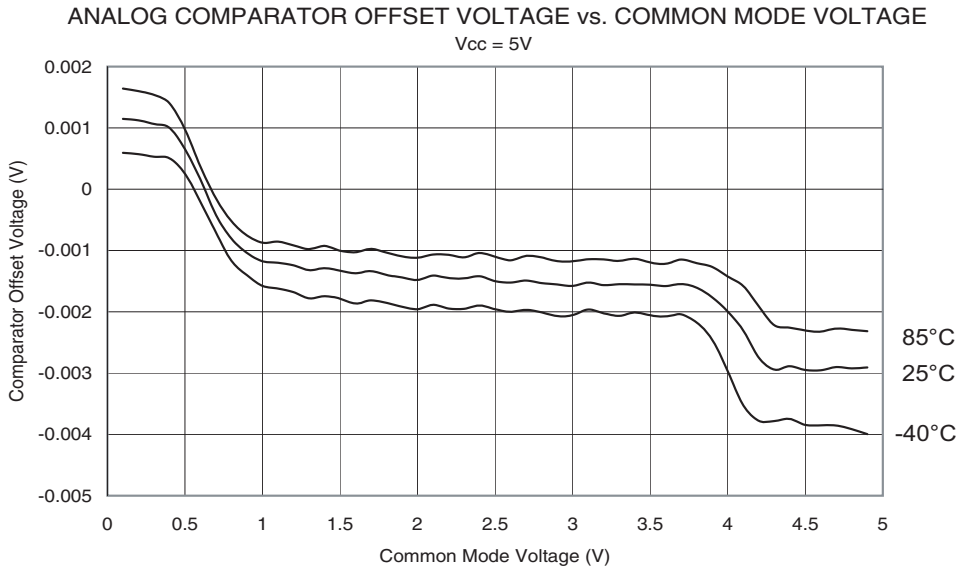
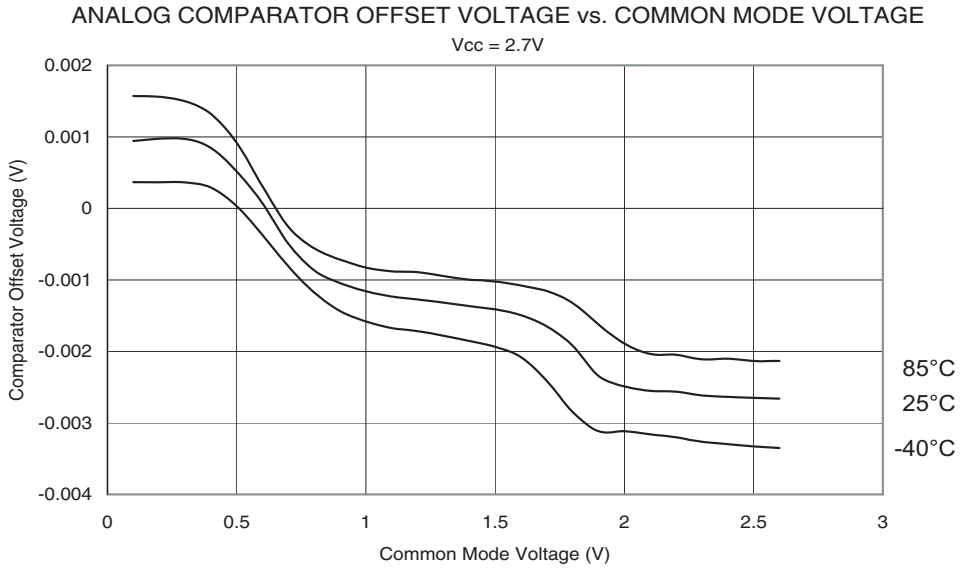


Figure 135. Analog Comparator Offset Voltage vs. Common Mode Voltage(V_{CC} = 2.7V)



Internal Oscillator Speed

Figure 136. Watchdog Oscillator Frequency vs. Temperature

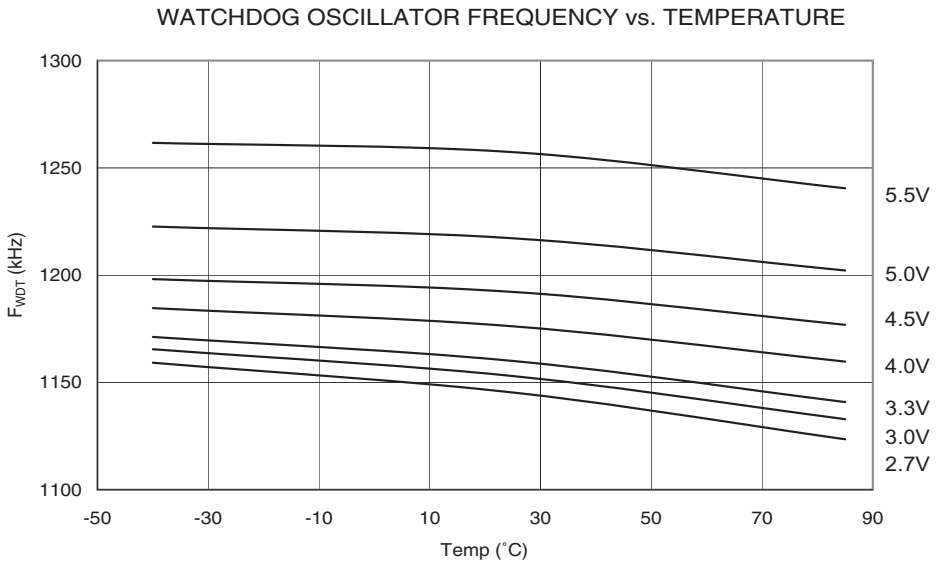


Figure 137. Watchdog Oscillator Frequency vs. V_{CC}

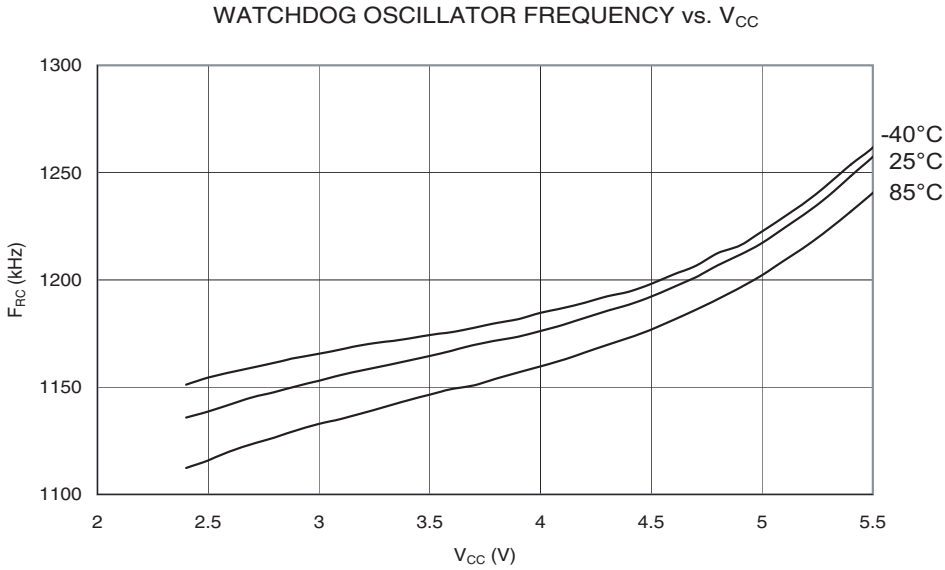


Figure 138. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

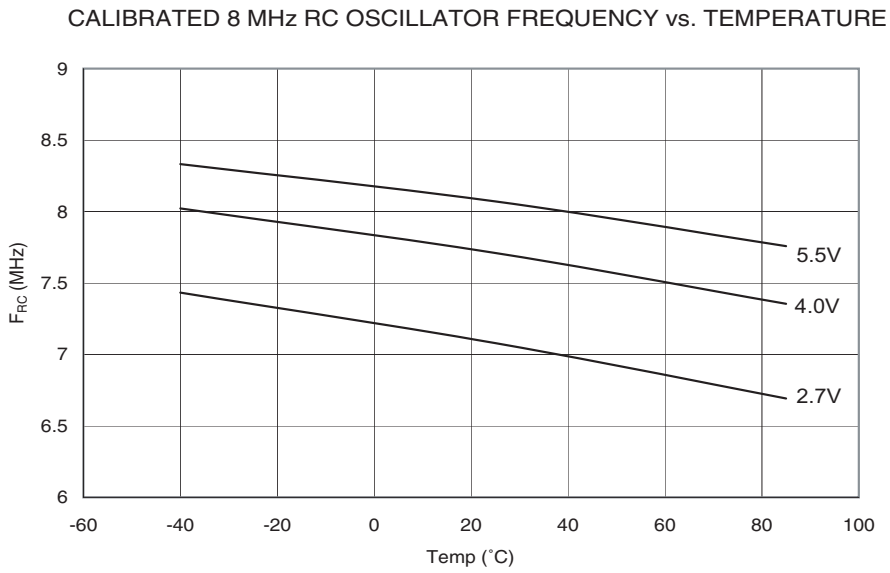


Figure 139. Calibrated 8 MHz RC Oscillator Frequency vs. V_{CC}

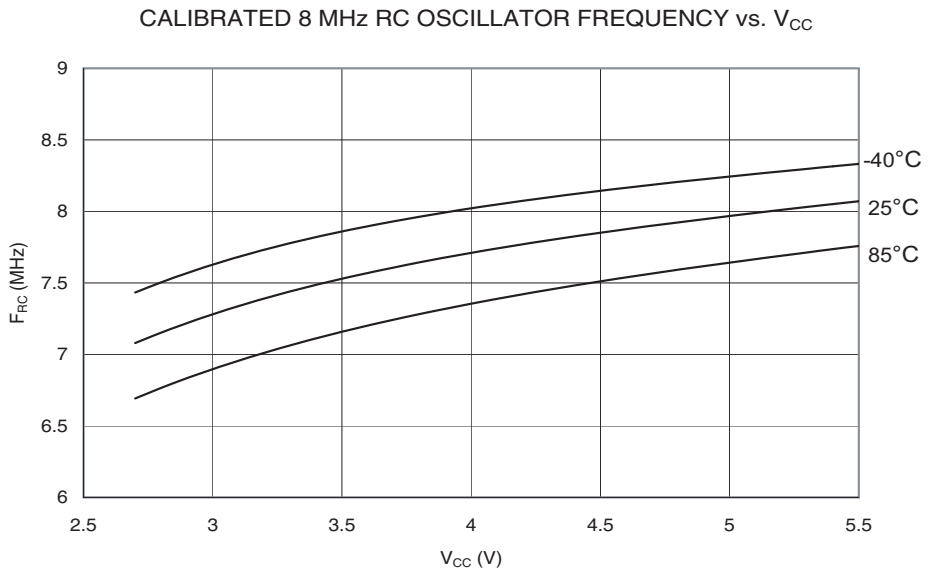


Figure 140. Calibrated 8 MHz RC Oscillator Frequency vs. Osccal Value

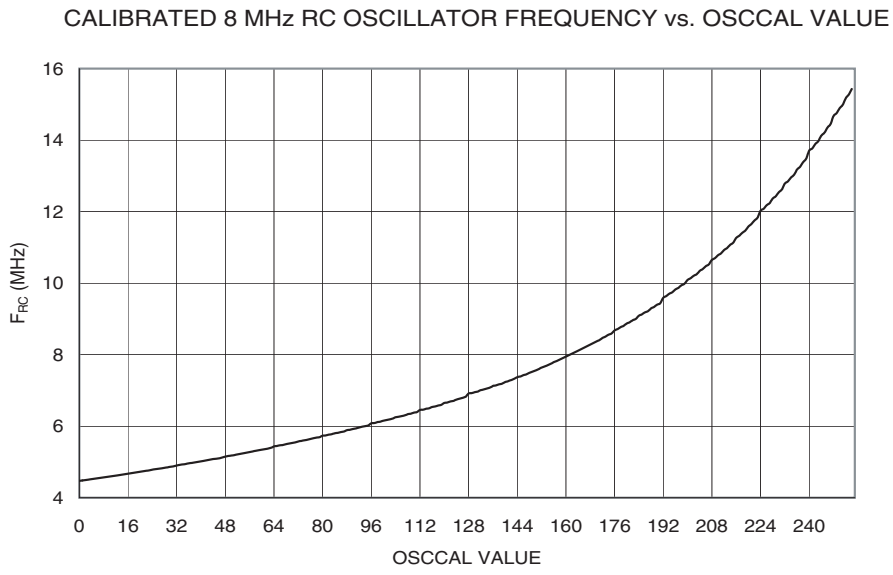


Figure 141. Calibrated 4 MHz RC Oscillator Frequency vs. Temperature

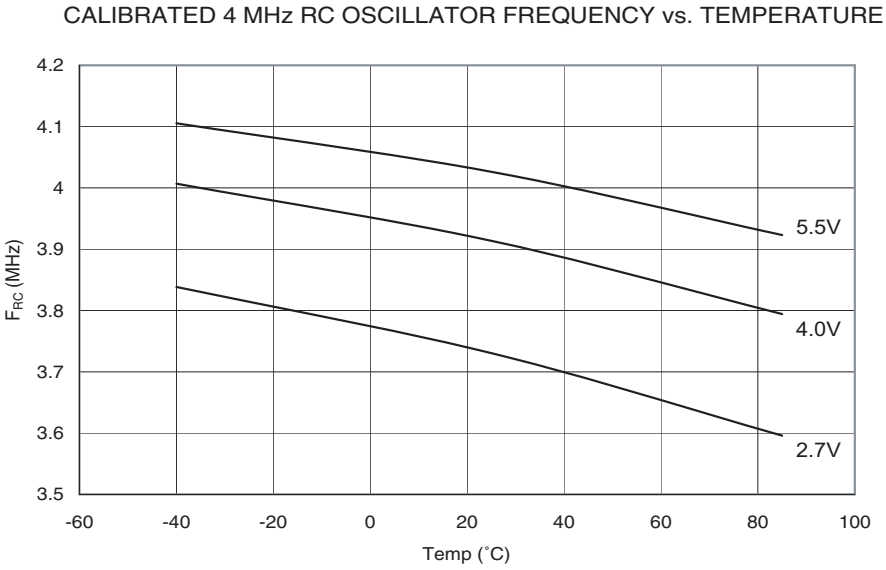


Figure 142. Calibrated 4 MHz RC Oscillator Frequency vs. V_{CC}

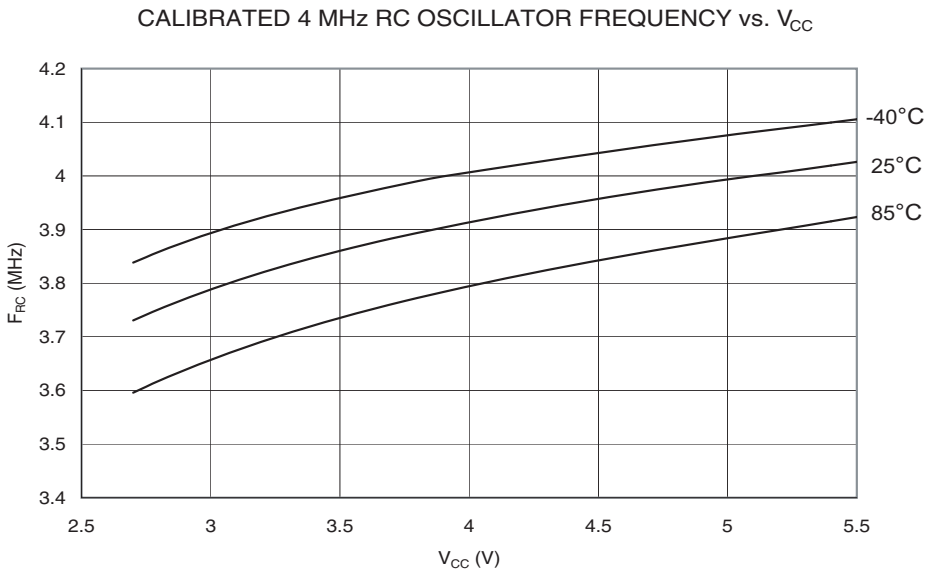


Figure 143. Calibrated 4 MHz RC Oscillator Frequency vs. Osccal Value

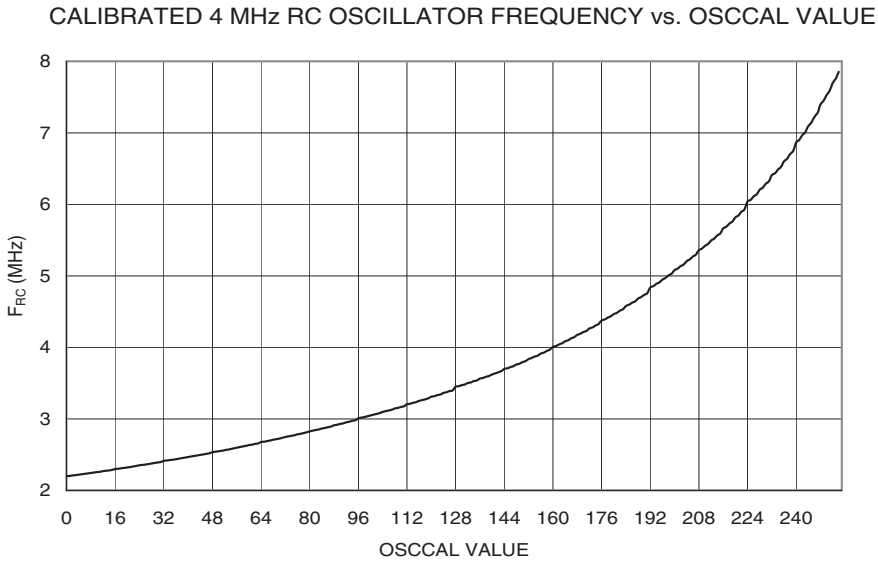


Figure 144. Calibrated 2 MHz RC Oscillator Frequency vs. Temperature

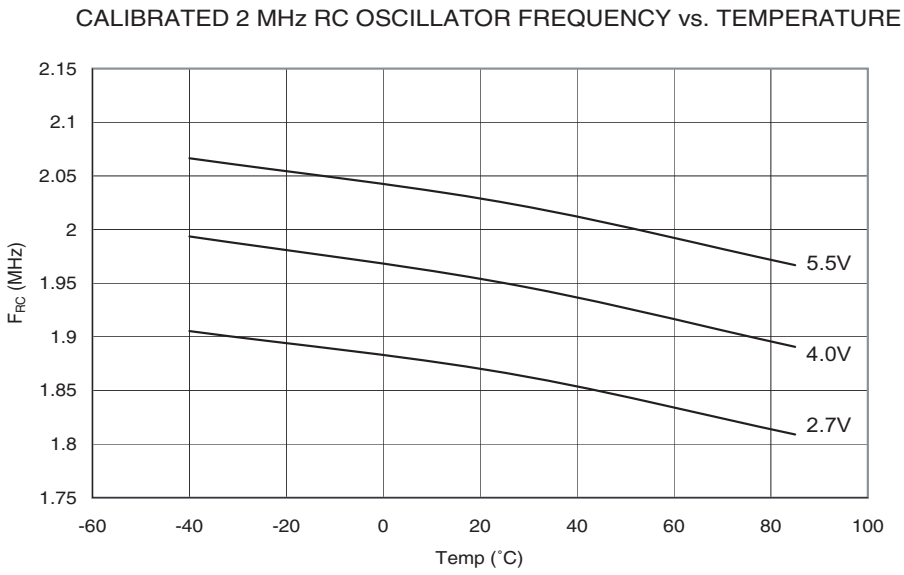


Figure 145. Calibrated 2 MHz RC Oscillator Frequency vs. V_{CC}

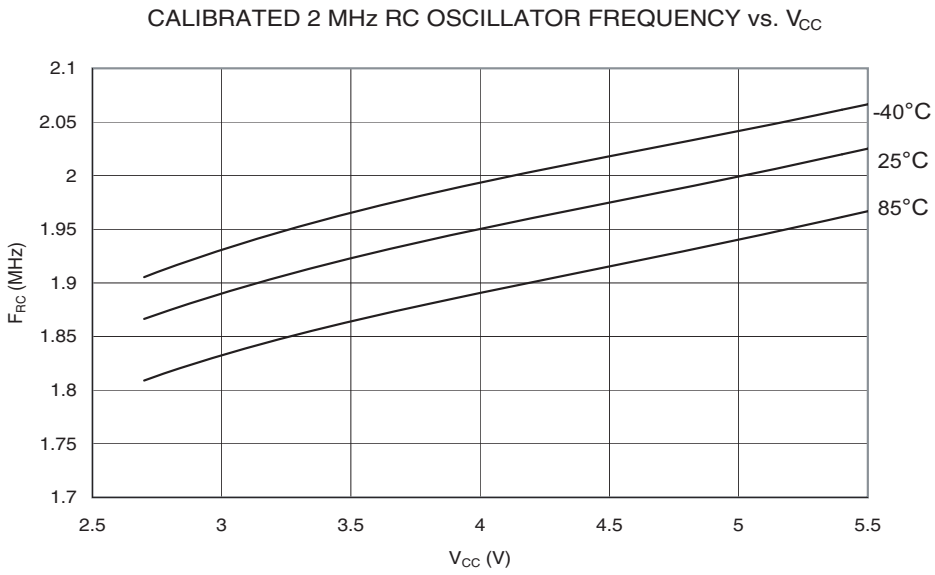


Figure 146. Calibrated 2 MHz RC Oscillator Frequency vs. Osccal Value

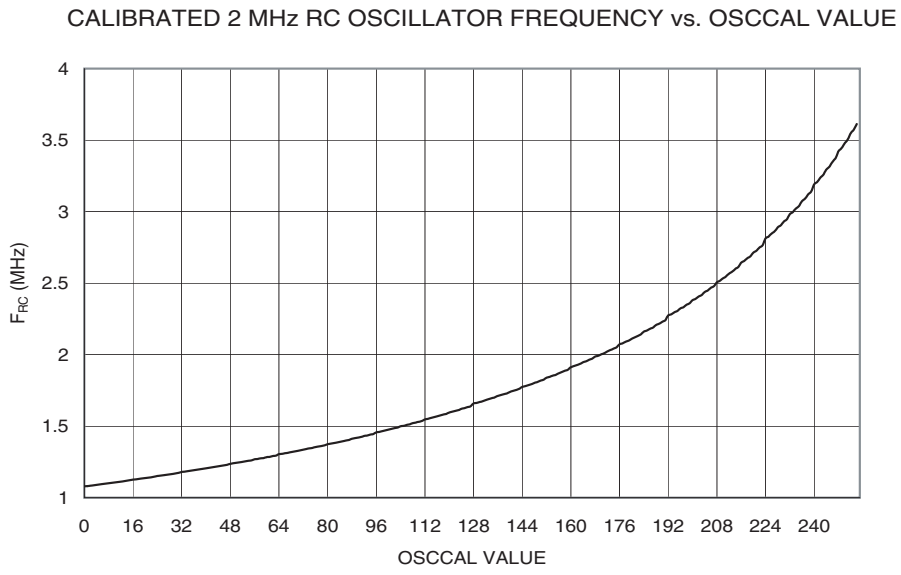


Figure 147. Calibrated 1 MHz RC Oscillator Frequency vs. Temperature

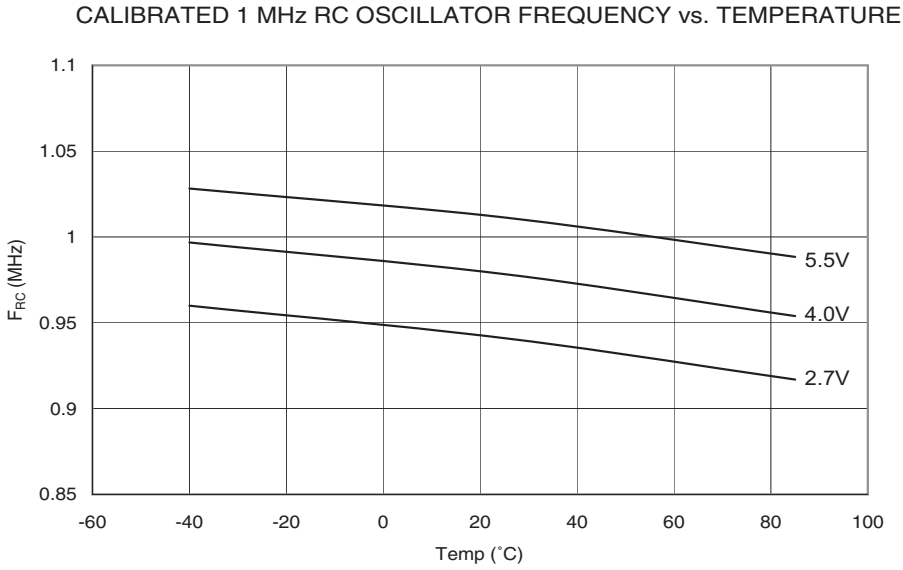


Figure 148. Calibrated 1 MHz RC Oscillator Frequency vs. V_{CC}

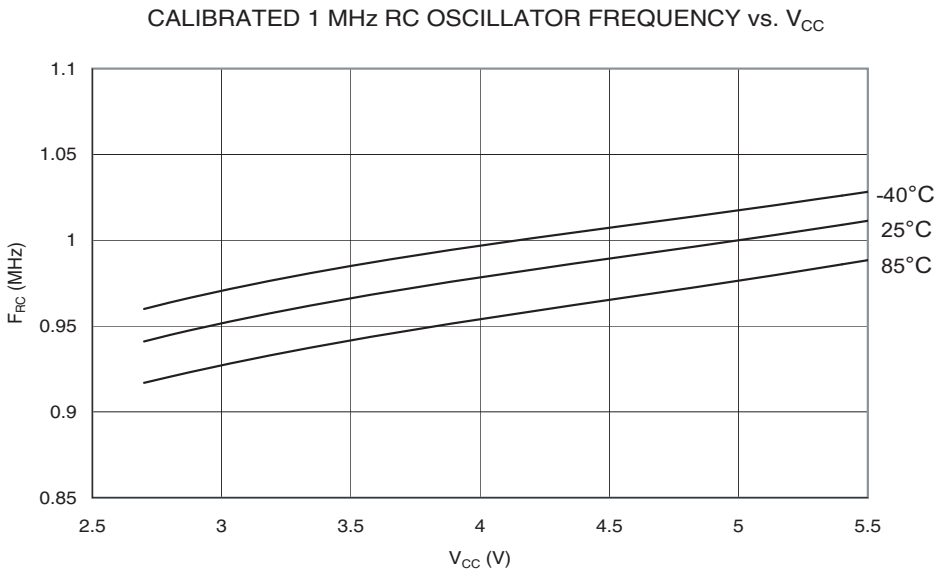
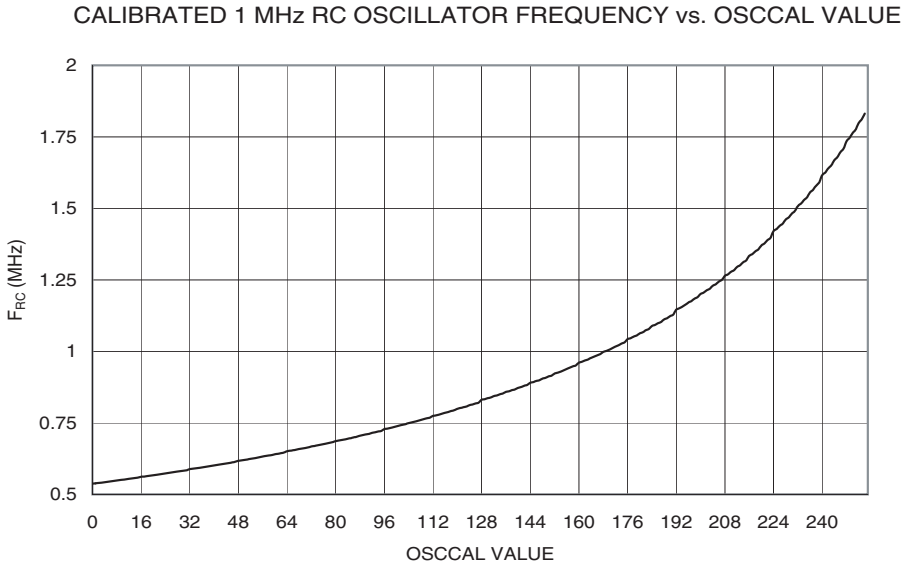


Figure 149. Calibrated 1 MHz RC Oscillator Frequency vs. Osccal Value



Current Consumption Of Peripheral Units

Figure 150. Analog Comparator Current vs. V_{CC}

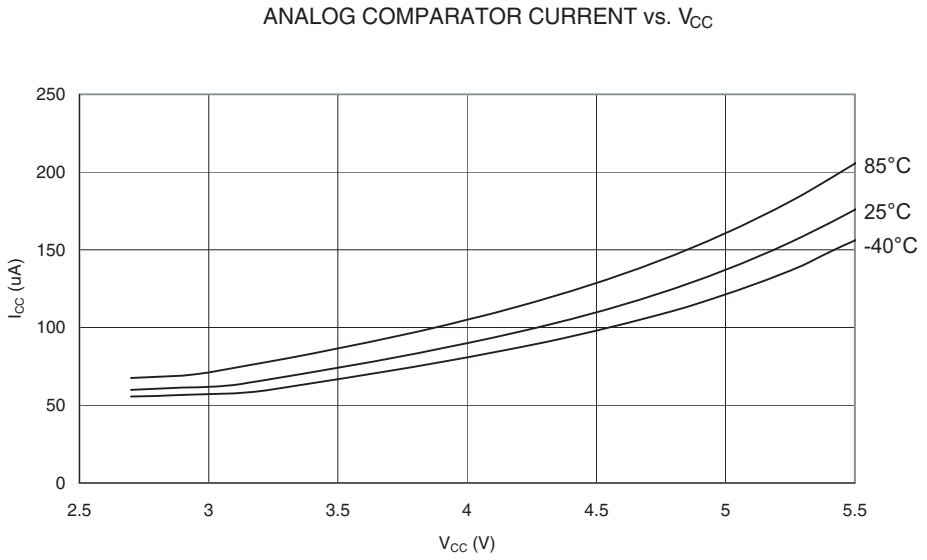


Figure 151. Brownout Detector Current vs. V_{CC}

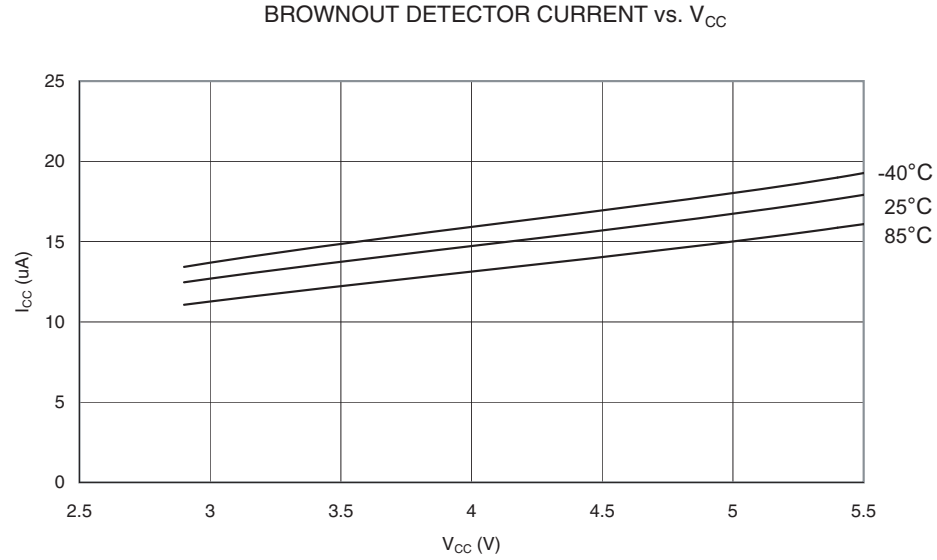
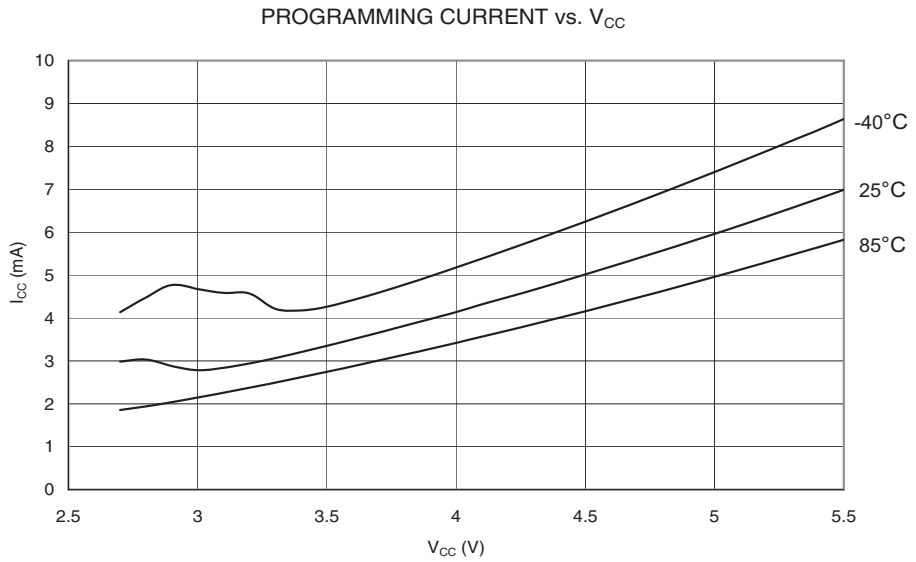


Figure 152. Programming Current vs. V_{CC}



Current Consumption In Reset And Reset Pulsewidth

Figure 153. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, Excluding Current Through The Reset Pull-up)

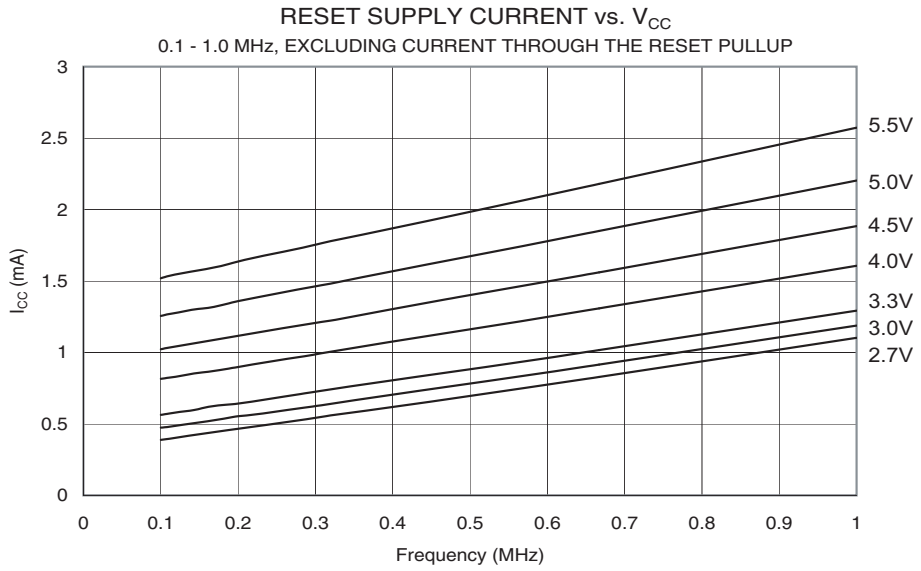


Figure 154. Reset Supply Current vs. V_{CC} (1 - 20 MHz, Excluding Current Through The Reset Pull-up)

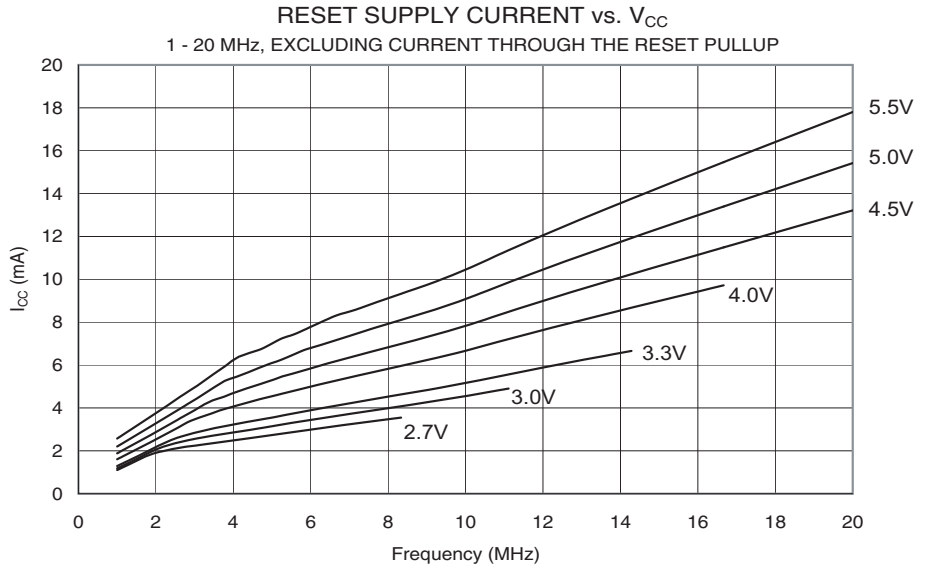
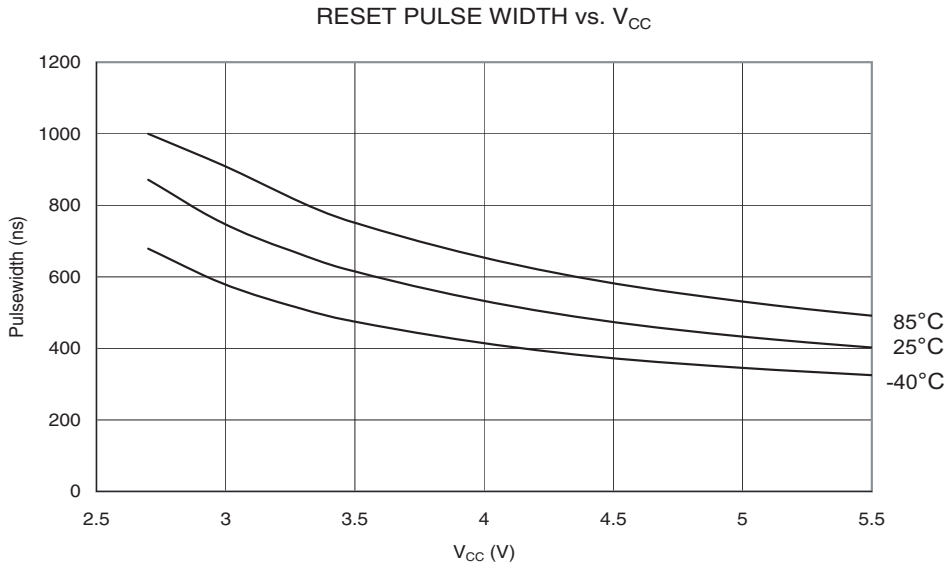


Figure 155. Reset Pulse Width vs. V_{CC}



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	10
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	12
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
$3C ($5C)	Reserved	-								
$3B ($5B)	GICR	INT1	INT0	INT2	-	-	-	IVSEL	IVCE	57, 78
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	-	-	-	-	-	79
$39 ($59)	TIMSK	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	OCIE0	93, 124
$38 ($58)	TIFR	TOV1	OCF1A	OCF1B	-	ICF1	-	TOV0	OCF0	94, 125
$37 ($57)	SPMCR	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	170
$36 ($56)	EMCUCR	SM0	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	ISC2	29,42,78
$35 ($55)	MCUCR	SRE	SRW10	SE	SM1	ISC11	ISC10	ISC01	ISC00	29,41,77
$34 ($54)	MCUCSR	-	-	SM2	-	WDRF	BORF	EXTRF	PORF	41,49
$33 ($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	91
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								93
$31 ($51)	OCR0	Timer/Counter0 Output Compare Register								93
$30 ($50)	SFIOR	-	XMBK	XMM2	XMM1	XMM0	PUD	-	PSR10	31,66,96
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	119
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	122
$2D ($4D)	TCNT1H	Timer/Counter1 - Counter Register High Byte								123
$2C ($4C)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								123
$2B ($4B)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								123
$2A ($4A)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								123
$29 ($49)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								123
$28 ($48)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								123
$27 ($47)	Reserved	-								-
$26 ($46)	Reserved	-								-
$25 ($45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								124
$24 ($44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								124
$23 ($43)	Reserved	-								-
$22 ($42)	Reserved	-								-
$21 ($41)	WDTCR	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	51
$20[1] ($40)[1]	UBRRH	URSEL	-	-	-	UBRR[11:8]				159
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	157
$1F ($3F)	EEARH	-	-	-	-	-	-	-	EEAR8	19
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								19
$1D ($3D)	EEDR	EEPROM Data Register								20
$1C ($3C)	EECR	-	-	-	-	EERIE	EEMWE	EEWE	EERE	20
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	75
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	75
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	75
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	75
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	75
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	75
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	75
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	75
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	76
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	76
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	76
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	76
$0F ($2F)	SPDR	SPI Data Register								133
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	133
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	131
$0C ($2C)	UDR	USART I/O Data Register								155
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	155
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	156
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								159
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	164
$07 ($27)	PORTE	-	-	-	-	-	PORTE2	PORTE1	PORTE0	76
$06 ($26)	DDRE	-	-	-	-	-	DDE2	DDE1	DDE0	76
$05 ($25)	PINE	-	-	-	-	-	PINE2	PINE1	PINE0	76
$04 ($24)	OSCCAL	Oscillator Calibration Register								39

Notes: 1. Refer to the USART description for details on how to access UBRRH and UCSRC.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

3. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF − Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 − Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - 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 ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd − Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd − Rr − C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd − K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

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
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					

Mnemonics	Operands	Description	Operation	Flags	#Clocks
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operation Range
8	2.7 - 5.5V	ATmega8515L-8AC	44A	Commercial (0°C to 70°C)
		ATmega8515L-8PC	40P6	
		ATmega8515L-8JC	44J	
		ATmega8515L-8MC[2]	44M1	
		ATmega8515L-8AI	44A	Industrial (-40°C to 85°C)
		ATmega8515L-8PI	40P6	
		ATmega8515L-8JI	44J	
		ATmega8515L-8MI	44M1	
		ATmega8515L-8AU[2]	44A	
		ATmega8515L-8PU[2]	40P6	
		ATmega8515L-8JU[2]	44J	
		ATmega8515L-8MU[2]	44M1	
16	4.5 - 5.5V	ATmega8515-16AC	44A	Commercial (0°C to 70°C)
		ATmega8515-16PC	40P6	
		ATmega8515-16JC	44J	
		ATmega8515-16MC	44M1	
		ATmega8515-16AI	44A	Industrial (-40°C to 85°C)
		ATmega8515-16PI	40P6	
		ATmega8515-16JI	44J	
		ATmega8515-16MI	44M1	
		ATmega8515-16AU[2]	44A	
		ATmega8515-16PU[2]	40P6	
		ATmega8515-16JU[2]	44J	
		ATmega8515-16MU[2]	44MI	

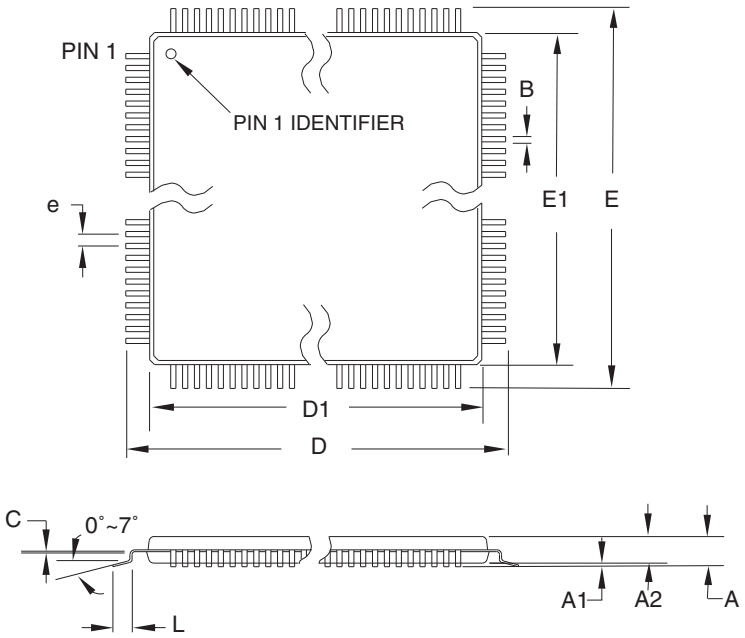
Note: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities..

2. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive).Also Halide free and fully Green.

Package Type	
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
40P6	40-lead, 0.600” Wide, Plastic Dual Inline Package (PDIP)
44J	44-lead, Plastic J-Leaded Chip Carrier (PLCC)
44M1	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

Packaging Information

44A



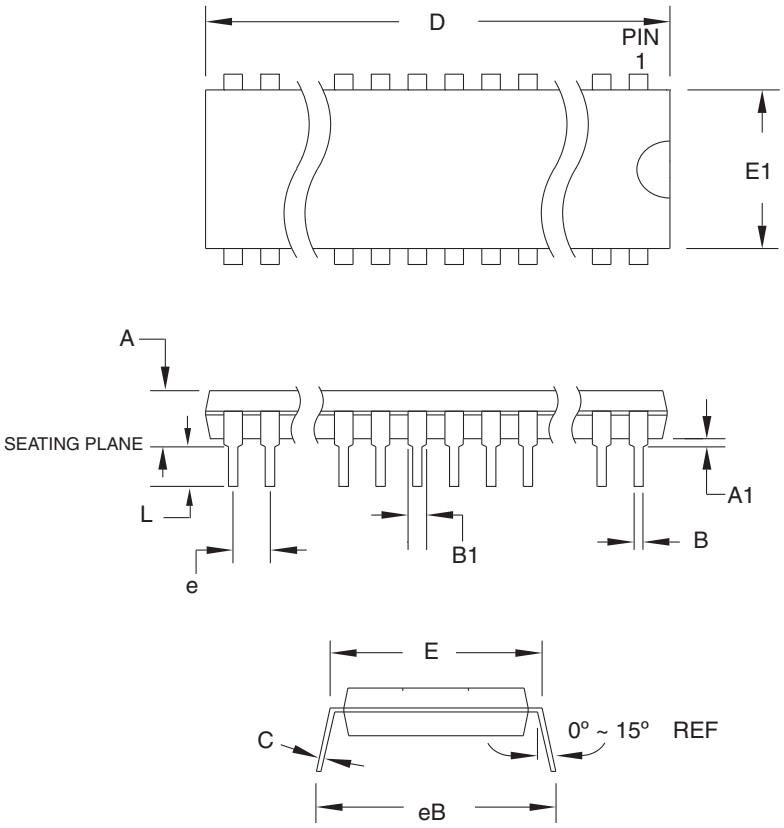
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A,** 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B



COMMON DIMENSIONS
(Unit of Measure = mm)

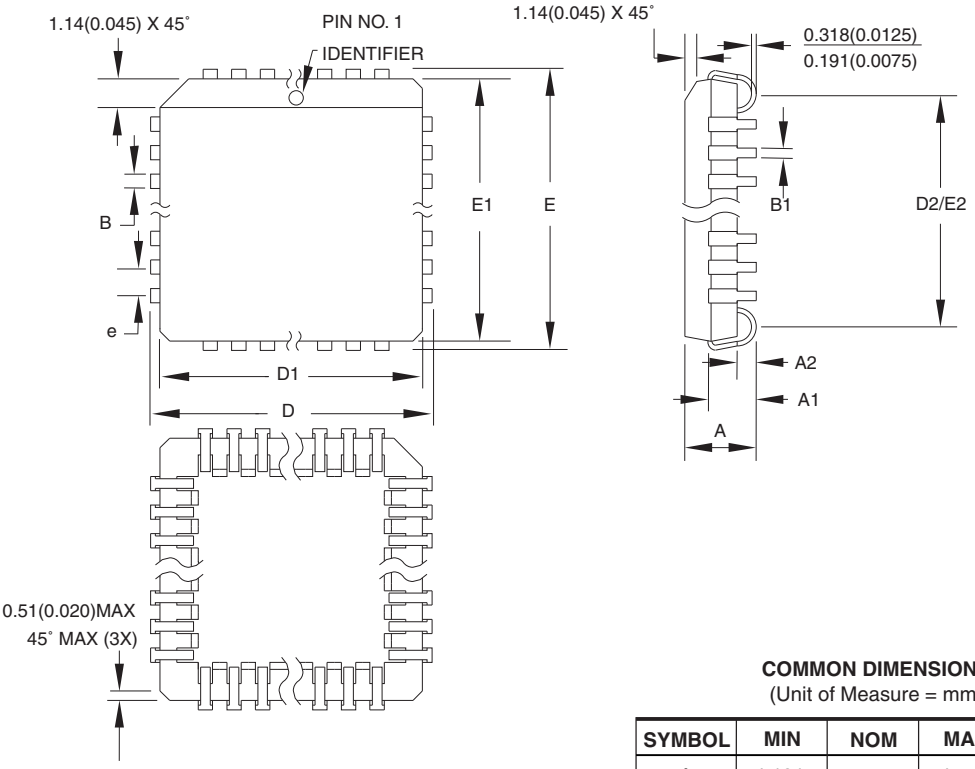
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

44J



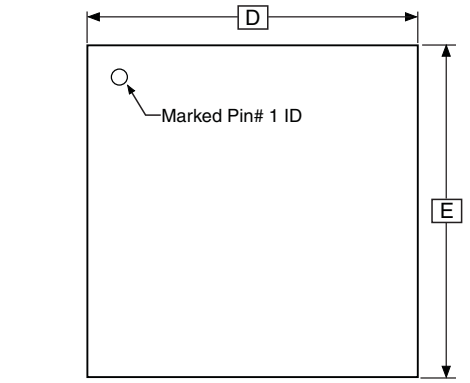
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

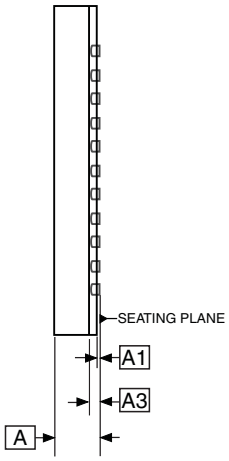
Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting line.
3. Lead coplanarity is 0.004" (0.102 mm) maximum.

10/04/01

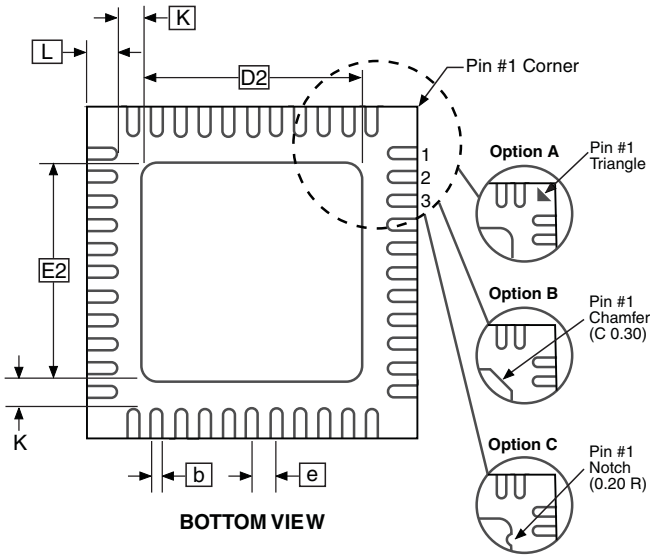
	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)	44J	B



TOP VIEW



SIDE VIEW



BOTTOM VIEW

Note: JEDEC Standard MO-220, Fig. 1 (SAW Singulation) VKKD-3.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	6.90	7.00	7.10	
D2	5.00	5.20	5.40	
E	6.90	7.00	7.10	
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.59	0.64	0.69	
K	0.20	0.26	0.41	

9/26/08

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	**44M1,** 44-pad, 7 x 7 x 1.0 mm Body, Lead Pitch 0.50 mm, 5.20 mm Exposed Pad, Thermally Enhanced Plastic Very Thin Quad Flat No Lead Package (VQFN)	ZWS	44M1	H

Errata

The revision letter in this section refers to the revision of the ATmega8515 device.

ATmega8515(L) Rev. C and D

1. **First Analog Comparator conversion may be delayed**

   If the device is powered by a slow rising VCC, the first Analog Comparator conversion will take longer than expected on some devices.

   Problem Fix/Workaround

   When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

Datasheet Revision History

Please note that the referring page numbers in this section are referring to this document. The referring revision in this section are referring to the document revision.

Changes from Rev. 2512J-10/06 to Rev. 2512K-01/10

1. **Updated Table 18 on page 46 Reset Characteristics.**

Changes from Rev. 2512I-08/06 to Rev. 2512J-10/06

1. **Updated TOP/BOTTOM description for all Timer/Counters Fast PWM mode.**
2. **Updated "Errata" on page 249.**

Changes from Rev. 2512H-04/06 to Rev. 2512I-08/06

1. **Updated "Ordering Information" on page 244.**

Changes from Rev. 2512G-03/05 to Rev. 2512H-04/06

1. **Added "Resources" on page 6.**
2. **Updated cross reference in "Phase Correct PWM Mode" on page 113.**
3. **Updated "Timer/Counter Interrupt Mask Register – TIMSK(1)" on page 124.**
4. **Updated "Serial Peripheral Interface – SPI" on page 126.**
5. **Removed obsolete section of "Calibration Byte" on page 181.**
6. **Updated Table 10 on page 38, Table 52 on page 120, Table 94 on page 196 and Table 96 on page 199.**

Changes from Rev. 2512F-12/03 to Rev. 2512G-03/05

1. **MLF-package alternative changed to "Quad Flat No-Lead/Micro Lead Frame Package QFN/MLF".**
2. **Updated "Electrical Characteristics" on page 197**
3. **Updated "Ordering Information" on page 244.**

Rev. 2512E-09/03

1. **Updated "Calibrated Internal RC Oscillator" on page 39.**

Rev. 2512E-09/03

1. **Removed "Preliminary" from the datasheet.**
2. **Updated Table 18 on page 46 and "Absolute Maximum Ratings" and "DC Characteristics" in "Electrical Characteristics" on page 197.**
3. **Updated chapter "ATmega8515 Typical Characteristics" on page 207.**

Rev. 2512D-02/03

1. **Added "EEPROM Write During Power-down Sleep Mode" on page 23.**
2. **Improved the description in "Phase Correct PWM Mode" on page 88.**
3. **Corrected OCn waveforms in Figure 53 on page 111.**

4. Added note under “Filling the Temporary Buffer (page loading)” on page 173 about writing to the EEPROM during an SPM page load.
5. Updated Table 93 on page 195.
6. Updated “Packaging Information” on page 245.

Rev. 2512C-10/02

1. Added “Using all Locations of External Memory Smaller than 64 KB” on page 31.
2. Removed all TBD.
3. Added description about calibration values for 2, 4, and 8 MHz.
4. Added variation in frequency of “External Clock” on page 40.
5. Added note about V_{BOT}, Table 18 on page 46.
6. Updated about “Unconnected pins” on page 64.
7. Updated “16-bit Timer/Counter1” on page 97, Table 51 on page 119 and Table 52 on page 120.
8. Updated “Enter Programming Mode” on page 184, “Chip Erase” on page 184, Figure 77 on page 187, and Figure 78 on page 188.
9. Updated “Electrical Characteristics” on page 197, “External Clock Drive” on page 199, Table 96 on page 199 and Table 97 on page 200, “SPI Timing Characteristics” on page 200 and Table 98 on page 202.
10. Added “Errata” on page 249.

Rev. 2512B-09/02

1. Changed the Endurance on the Flash to 10,000 Write/Erase Cycles.

Rev. 2512A-04/02

1. Initial.

Table of Contents

Features 1

Pin Configurations 2

Overview 3

Block Diagram 3
Disclaimer 4
AT90S4414/8515 and ATmega8515 Compatibility 4
Pin Descriptions 5

Resources 6

About Code Examples 7

AVR CPU Core 8

Introduction 8
Architectural Overview 8
ALU – Arithmetic Logic Unit 9
Status Register 10
General Purpose Register File 11
Stack Pointer 12
Instruction Execution Timing 13
Reset and Interrupt Handling 13

AVR ATmega8515 Memories 16

In-System Reprogrammable Flash Program memory 16
SRAM Data Memory 17
EEPROM Data Memory 19
I/O Memory 24
External Memory Interface 25
XMEM Register Description 29

System Clock and Clock Options 34

Clock Systems and their Distribution 34
Clock Sources 35
Default Clock Source 35
Crystal Oscillator 35
Low-frequency Crystal Oscillator 37
External RC Oscillator 38
Calibrated Internal RC Oscillator 39
External Clock 40

Power Management and Sleep Modes 41

Idle Mode 42
Power-down Mode 42

Standby Mode 43
Minimizing Power Consumption 43

System Control and Reset 45
Internal Voltage Reference 50
Watchdog Timer 50
Timed Sequences for Changing the Configuration of the Watchdog Timer 53

Interrupts 54
Interrupt Vectors in ATmega8515 54

I/O Ports 59
Introduction 59
Ports as General Digital I/O 60
Alternate Port Functions 64
Register Description for I/O Ports 75

External Interrupts 77

8-bit Timer/Counter0 with PWM 80
Overview 80
Timer/Counter Clock Sources 81
Counter Unit 81
Output Compare Unit 82
Compare Match Output Unit 84
Modes of Operation 85
Timer/Counter Timing Diagrams 89
8-bit Timer/Counter Register Description 91

Timer/Counter0 and Timer/Counter1 Prescalers 95

16-bit Timer/Counter1 97
Overview 97
Accessing 16-bit Registers 100
Timer/Counter Clock Sources 103
Counter Unit 103
Input Capture Unit 104
Output Compare Units 106
Compare Match Output Unit 108
Modes of Operation 109
Timer/Counter Timing Diagrams 117
16-bit Timer/Counter Register Description 119

Serial Peripheral Interface – SPI 126

$\overline{SS}$ Pin Functionality 131
Data Modes 134

USART 135
Single USART 135
Clock Generation 137
Frame Formats 140
USART Initialization 141
Data Transmission – The USART Transmitter 142
Data Reception – The USART Receiver 145
Asynchronous Data Reception 148
Multi-processor Communication Mode 151
Accessing UBRRH/UCSRC Registers 153
USART Register Description 155
Examples of Baud Rate Setting 159

Analog Comparator 164

Boot Loader Support – Read-While-Write Self-Programming 166
Features 166
Application and Boot Loader Flash Sections 166
Read-While-Write and No Read-While-Write Flash Sections 166
Boot Loader Lock bits 168
Entering the Boot Loader Program 169
Addressing the Flash During Self-Programming 171
Self-Programming the Flash 172

Memory Programming 179
Program and Data Memory Lock bits 179
Fuse bits 180
Signature Bytes 181
Calibration Byte 181
Parallel Programming Parameters, Pin Mapping, and Commands 182
Parallel Programming 184
Serial Downloading 193
Serial Programming Pin Mapping 193

Electrical Characteristics 197
External Clock Drive Waveforms 199
External Clock Drive 199
SPI Timing Characteristics 200
External Data Memory Timing 202

ATmega8515 Typical Characteristics 207

Register Summary 239

Instruction Set Summary *241*

Ordering Information *244*

Packaging Information *245*
44A 245
40P6 246
44J 247
44M1 248

Errata *249*
ATmega8515(L)
Rev. C and D 249

Datasheet Revision History *250*
Changes from Rev. 2512I-08/06 to Rev. 2512J-10/06 250
Changes from Rev. 2512H-04/06 to Rev. 2512I-08/06 250
Changes from Rev. 2512G-03/05 to Rev. 2512H-04/06 250
Changes from Rev. 2512F-12/03 to Rev. 2512G-03/05 250
Changes from Rev. 2512F-12/03 to Rev. 2512E-09/03 250
Changes from Rev. 2512D-02/03 to Rev. 2512E-09/03 250
Changes from Rev. 2512C-10/02 to Rev. 2512D-02/03 250
Changes from Rev. 2512B-09/02 to Rev. 2512C-10/02 251
Changes from Rev. 2512A-04/02 to Rev. 2512B-09/02 251

Table of Contents ..... *i*

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life

© 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR®, AVR® logo and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

2512K–AVR–01/10