Features

- High-performance, Low-power AVR® 8-bit Microcontroller
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 8 MIPS Throughput at 8 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
- Self-programming In-System Programmable Flash Memory
  - 16K Bytes with Optional Boot Block (256 - 2K Bytes)
    Endurance: 1,000 Write/Erase Cycles
  - Boot Section Allows Reprogramming of Program Code without External Programmer
  - Optional Boot Code Section with Independent Lock Bits
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1024 Bytes Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Clock with Separate Oscillator and Counter Mode
  - Three PWM Channels
  - 8-channel, 10-bit ADC
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial UART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Four Sleep Modes: Idle, ADC Noise Reduction, Power-save, and Power-down
- Power Consumption at 4 MHz, 3.0V, 25°C
  - Active 5.0 mA
  - Idle Mode 1.9 mA
  - Power-down Mode < 1 µA
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP and 44-pin TQFP
- Operating Voltages
  - 2.7 - 5.5V for ATmega163L
  - 4.0 - 5.5V for ATmega163
- Speed Grades
  - 0 - 4 MHz for ATmega163L
  - 0 - 8 MHz for ATmega163

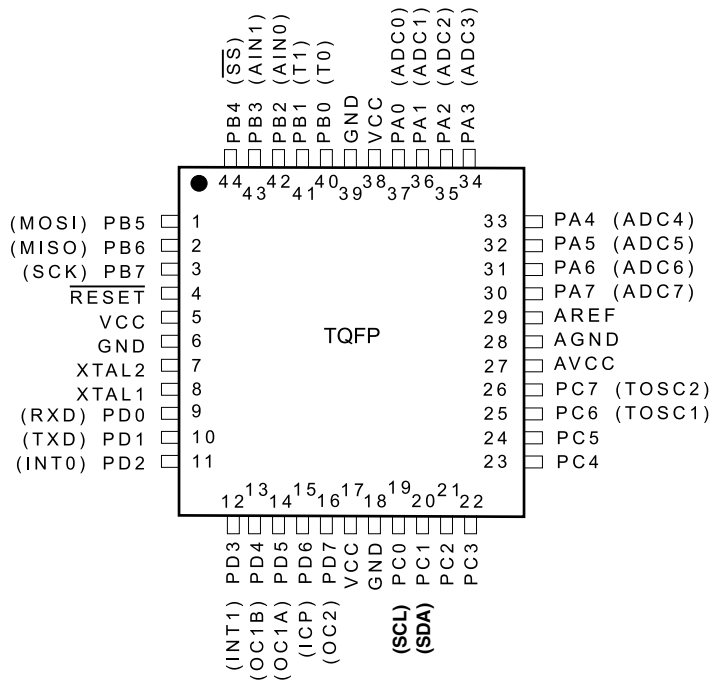
8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

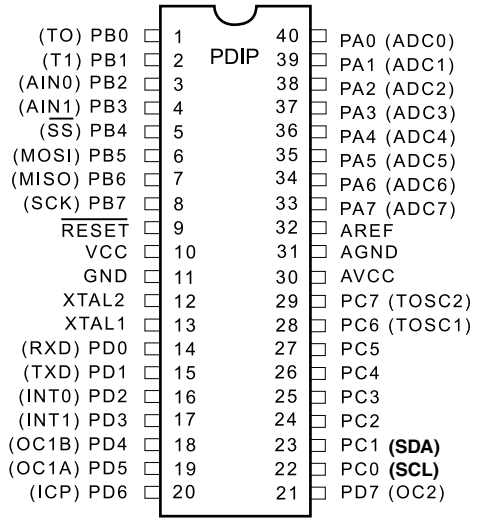
ATmega163
ATmega163L

Not Recommend for New Designs. Use ATmega16.

Rev. 1142D–AVR–09/02

Pin Configurations



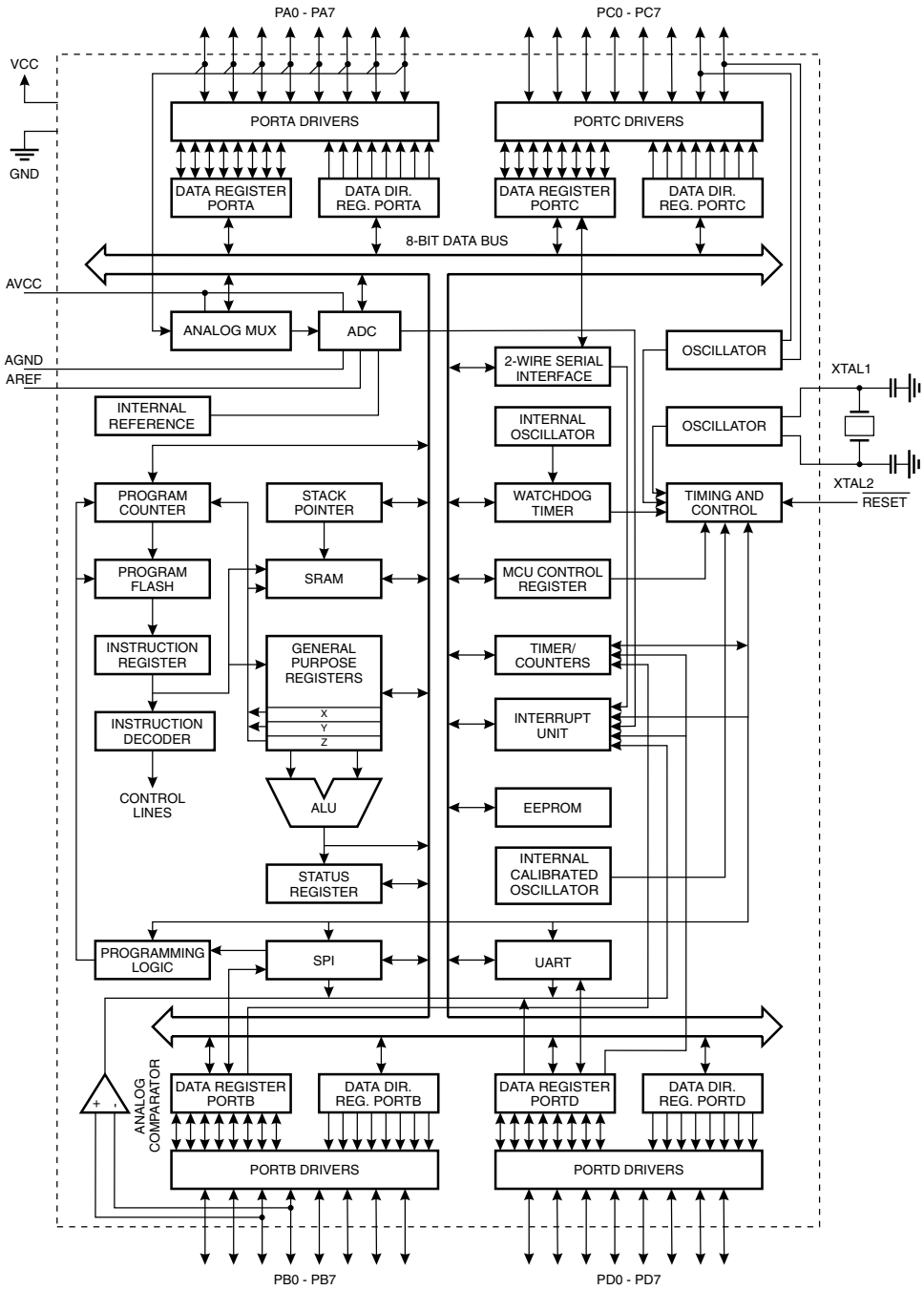


Description

The ATmega163 is a low-power CMOS 8-bit microcontroller based on the AVR architecture. By executing powerful instructions in a single clock cycle, the ATmega163 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock

cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega163 provides the following features: 16K bytes of In-System Self-Programmable Flash, 512 bytes EEPROM, 1024 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC, a programmable Watchdog Timer with internal Oscillator, a programmable serial UART, an SPI serial port, and four 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 Power-save mode, the asynchronous Timer Oscillator continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions.

The On-chip ISP Flash can be programmed through an SPI serial interface or a conventional programmer. By installing a Self-Programming Boot Loader, the microcontroller can be updated within the application without any external components. The Boot Program can use any interface to download the application program in the Application Flash memory. By combining an 8-bit CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega163 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega163 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

GND Digital ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers can sink 20mA and can drive LED displays directly. 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 tristated when a reset condition becomes active, even if the clock is not running.

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 can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. Port B also serves the functions of various special features of the ATmega83/163 as listed on page 117. The Port B pins are tristated when a reset condition becomes active, even if the clock is not running.

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 can sink 20 mA. 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 tristated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of various special features of the ATmega163 as listed on page 124.

Port D (PD7..PD0) Port D is an 8-bit bidirectional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. Port D also serves the functions of various special features of the ATmega163 as listed on page 128. The Port D pins are tristated when a reset condition becomes active, even if the clock is not running.

$\overline{\text{RESET}}$ Reset input. A low level on this pin for more than 500 ns will generate a Reset, even if the clock is not running. 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.

AVCC This is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. See page 105 for details on operation of the ADC.

AREF AREF is the analog reference input pin for the A/D Converter. For ADC operations, a voltage in the range 2.5V to AVCC can be applied to this pin.

AGND Analog ground. If the board has a separate analog ground plane, this pin should be connected to this ground plane. Otherwise, connect to GND.

Clock Options

The device has the following clock source options, selectable by Flash Fuse bits as shown:

Table 1. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001 - 1000
External RC Oscillator	0111 - 0101
Internal RC Oscillator	0100 - 0010
External Clock	0001 - 0000

Note: 1. “1” means unprogrammed, “0” means programmed.

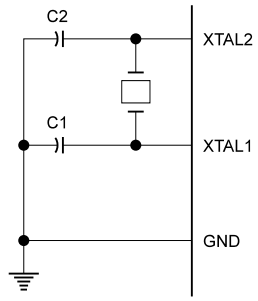
The various choices for each clocking option give different start-up times as shown in Table 5 on page 25.

Internal RC Oscillator The internal RC Oscillator option is an On-chip Oscillator running at a fixed frequency of nominally 1 MHz. If selected, the device can operate with no external components. The device is shipped with this option selected. See “EEPROM Read/Write Access” on page 62 for information on calibrating this Oscillator.

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 2. Either a quartz crystal or a ceramic resonator may be used.

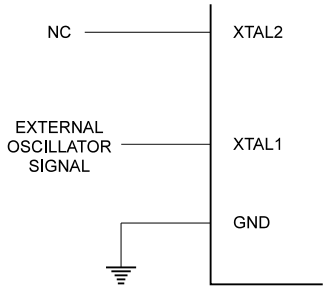
Figure 2. Oscillator Connections



External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 3.

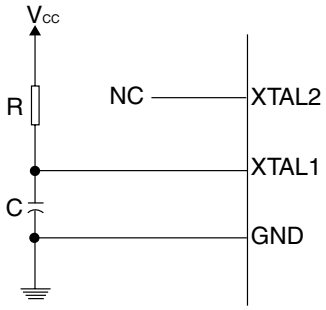
Figure 3. External Clock Drive Configuration



External RC Oscillator

For timing insensitive applications, the external RC configuration shown in Figure 4 can be used. For details on how to choose R and C, see Table 64 on page 162.

Figure 4. External RC Configuration



Timer Oscillator

For the Timer Oscillator pins, TOSC1 and TOSC2, the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32,768 Hz watch crystal. Applying an external clock source to the TOSC1 pin is not recommended.

Architectural Overview

The fast-access Register File concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one Arithmetic Logic Unit (ALU) operation is executed. 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.

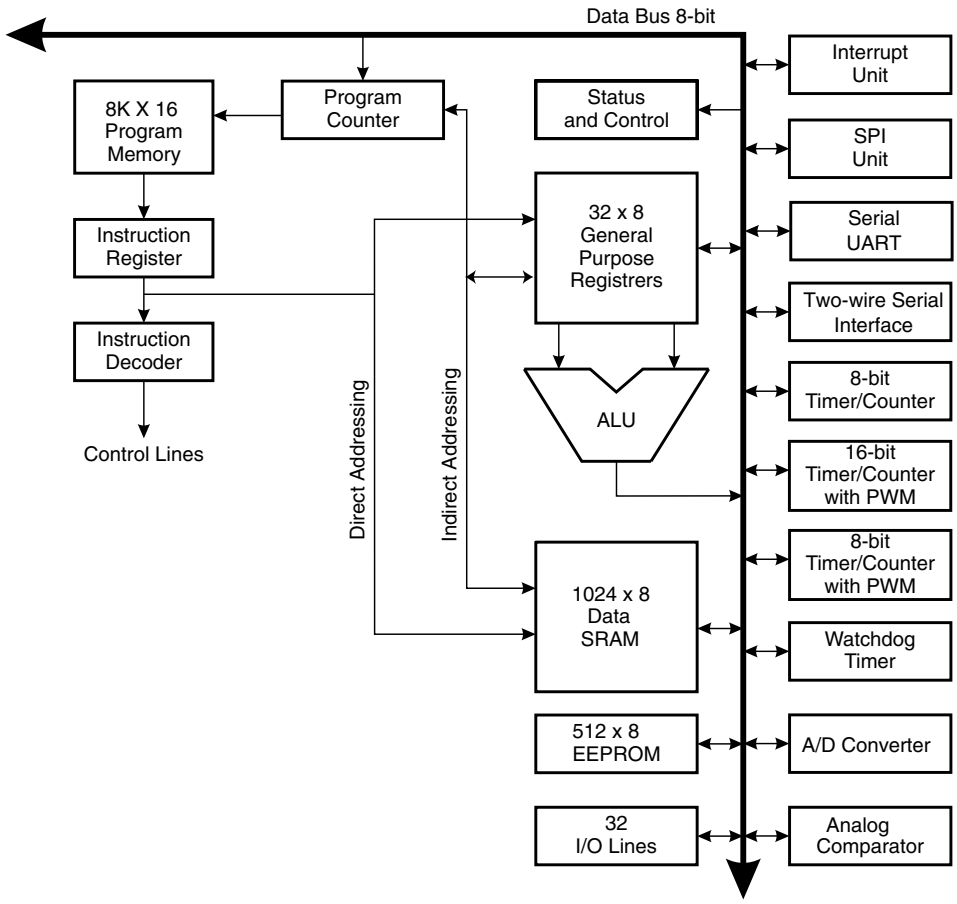
Six of the 32 registers can be used as three 16-bits indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the three address pointers is also used as the address pointer for look-up tables in Flash Program memory. These added function registers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

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 5 shows the ATmega163 AVR Enhanced RISC microcontroller architecture.

In addition to the register operation, the conventional Memory Addressing modes can be used on the Register File as well. This is enabled by the fact that the Register File is assigned the 32 lowest Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O Memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D-converters, 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.

Figure 5. The ATmega163 *AVR* RISC Architecture



The AVR uses a Harvard architecture concept – with separate memories and buses for program and data. The Program memory is executed with a two stage pipeline. 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.

With the jump and call instructions, the whole 8K word address space is directly accessed. 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 (256 to 2,048 bytes, see page 134) 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 is allowed only in the Boot Program section.

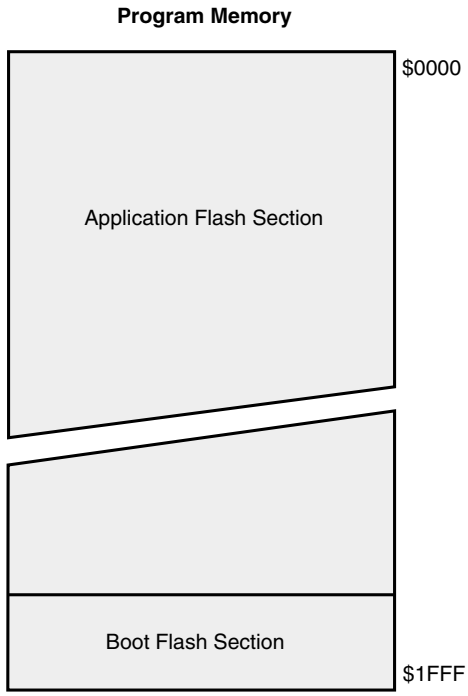
During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 11-bit Stack Pointer SP is read/write accessible in the I/O space.

The 1,024 bytes data SRAM can be easily 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 at the beginning of the Program memory. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

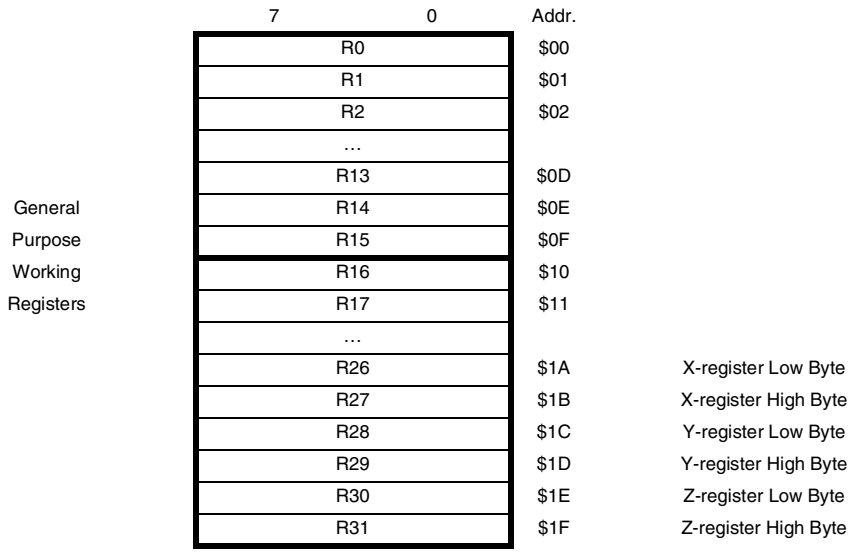
Figure 6. Memory Maps



The General Purpose Register File

Figure 7 shows the structure of the 32 general purpose working registers in the CPU.

Figure 7. AVR CPU General Purpose Working Registers



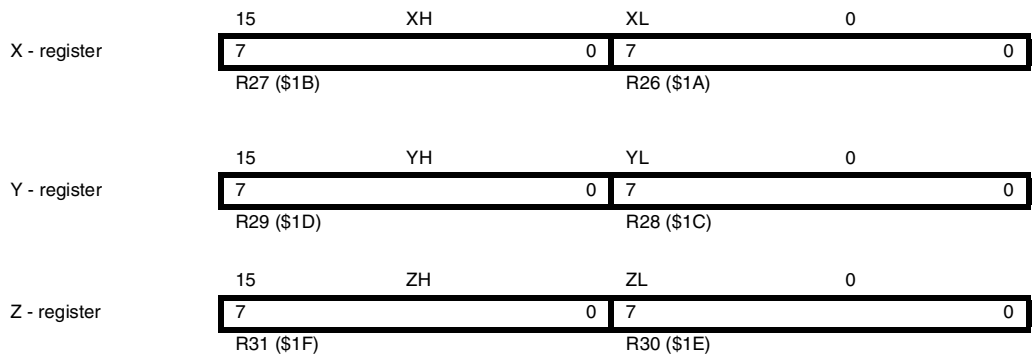
All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI, and ORI between a constant and a register, and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the Register File – R16..R31. The general SBC, SUB, CP, AND, and OR and all other operations between two registers or on a single register apply to the entire Register File.

As shown in Figure 7, 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-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 address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as:

Figure 8. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

The 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, ALU operations between registers in the Register File are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. ATmega163 also provides a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the Instruction Set section for a detailed description.

The In-System Self-Programmable Flash Program Memory

The ATmega163 contains 16K bytes On-chip In-System Self-Programmable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 8K x 16. The Flash Program memory space is divided in two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 1,000 write/erase cycles. The ATmega163 Program Counter (PC) is 13 bits wide, thus addressing the 8,192 Program Memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail on page 134. See also page 154 for a detailed description on Flash data serial downloading.

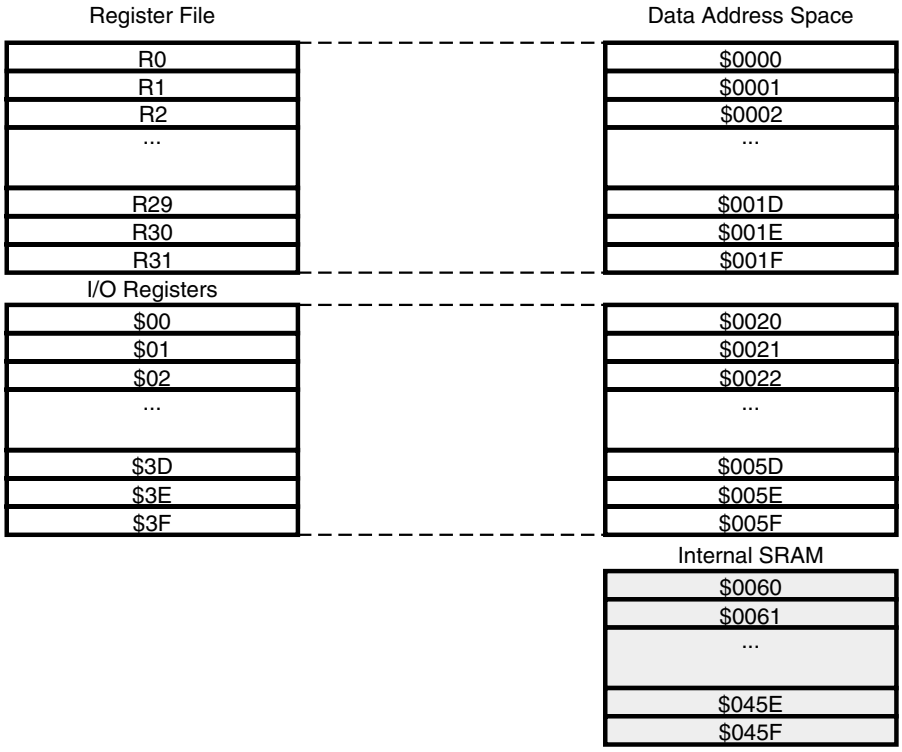
Constant tables can be allocated within the entire Program Memory address space (see the LPM – Load Program Memory instruction description).

See also page 12 for the different Program Memory Addressing modes.

The SRAM Data Memory

Figure 9 shows how the ATmega163 SRAM Memory is organized.

Figure 9. SRAM Organization



The lower 1,120 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File + I/O Memory, and the next 1,024 locations address the internal data SRAM.

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

The direct addressing reaches the entire data space.

The Indirect with Displacement mode features a 63 address locations reach 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 and incremented.

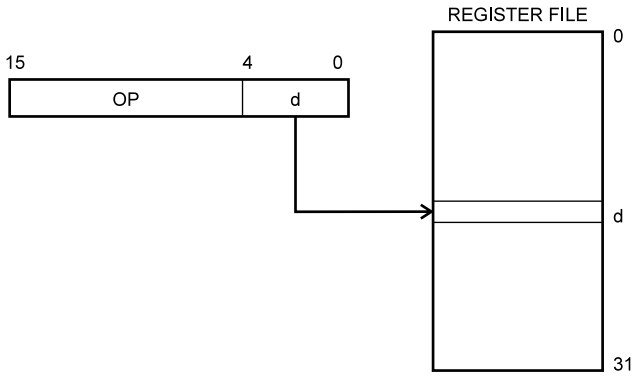
The 32 general purpose working registers, 64 I/O Registers, and the 1,024 bytes of internal data SRAM in the ATmega163 are all accessible through all these addressing modes.

The Program and Data Addressing Modes

The ATmega163 AVR Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the Program Memory (Flash) and Data Memory (SRAM, Register File, and I/O Memory). This section describes the different addressing modes supported by the AVR architecture. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

Register Direct, Single Register Rd

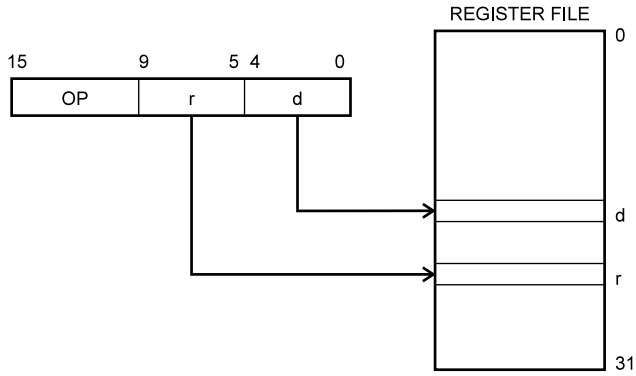
Figure 10. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd And Rr

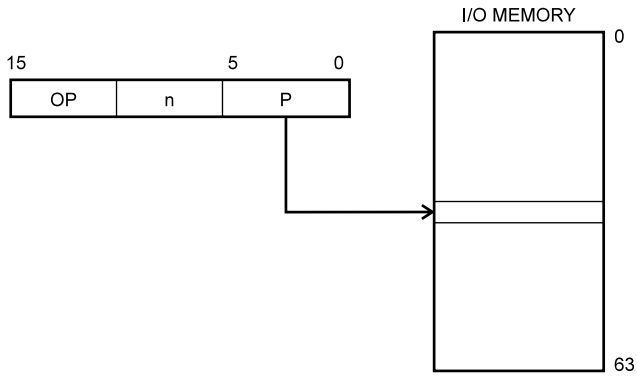
Figure 11. Direct Register Addressing, Two Registers



Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O Direct

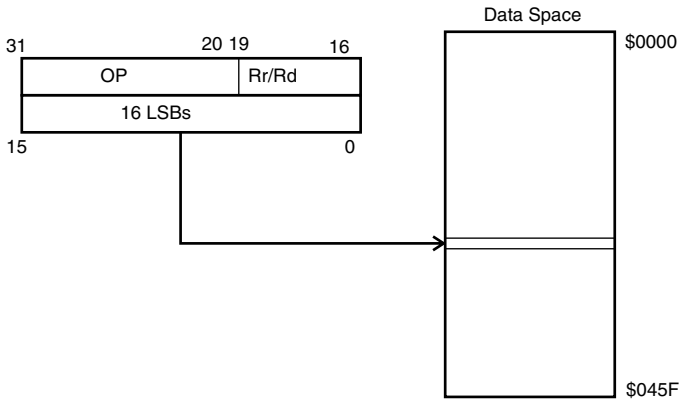
Figure 12. I/O Direct Addressing



Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

Data Direct

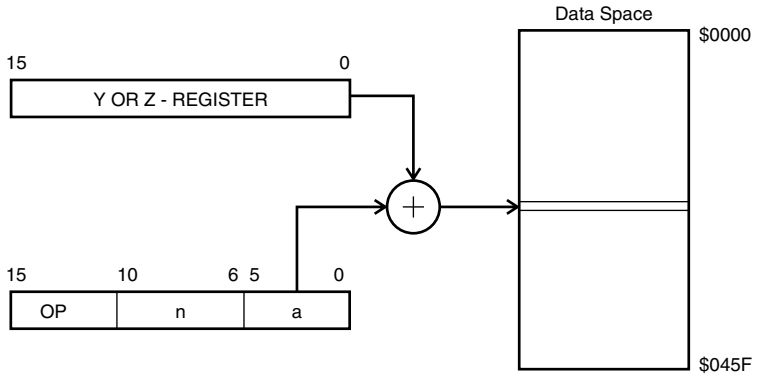
Figure 13. Direct Data Addressing



A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

Data Indirect with Displacement

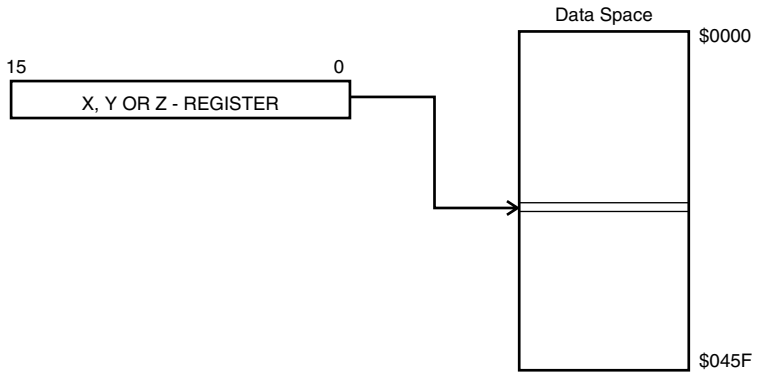
Figure 14. Data Indirect with Displacement



Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word.

Data Indirect

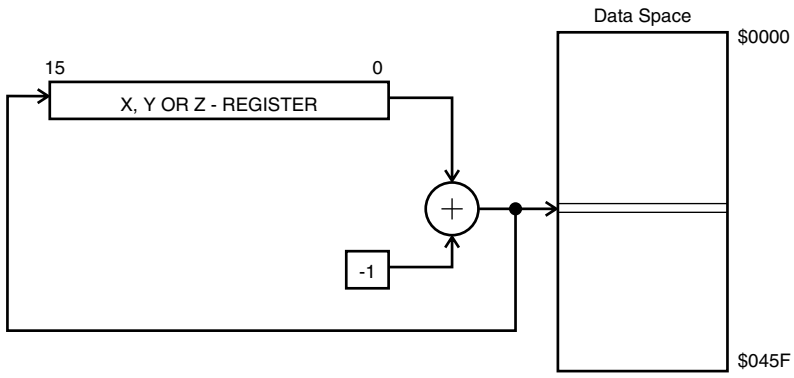
Figure 15. Data Indirect Addressing



Operand address is the contents of the X-, Y-, or the Z-register.

Data Indirect with Pre-decrement

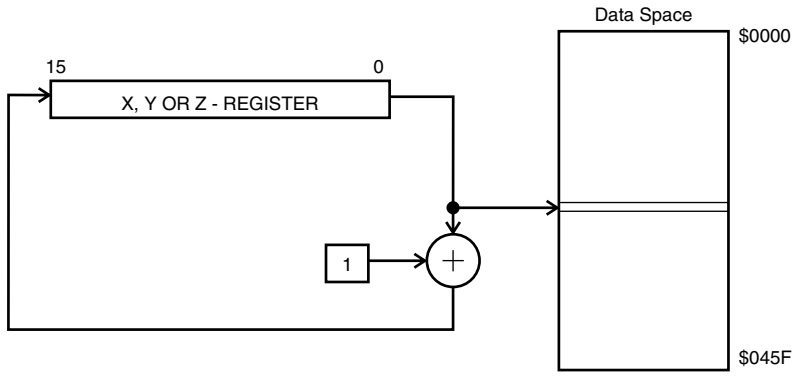
Figure 16. Data Indirect Addressing with Pre-decrement



The X-, Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.

Data Indirect with Post-increment

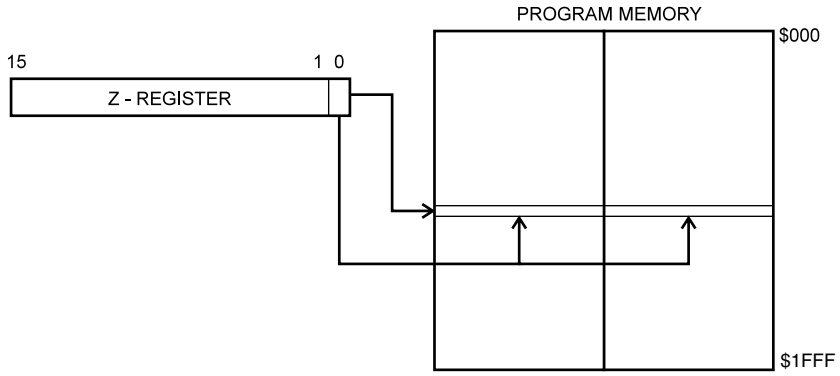
Figure 17. Data Indirect Addressing with Post-increment



The X-, Y-, or the Z-register is incremented after the operation. Operand address is the content of the X-, Y-, or the Z-register prior to incrementing.

Constant Addressing Using The LPM and SPM Instructions

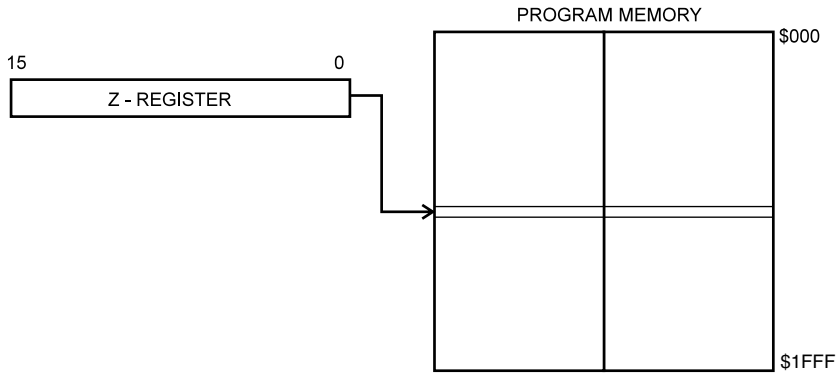
Figure 18. Code Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 8K). For LPM, the LSB selects Low Byte if cleared (LSB = 0) or High Byte if set (LSB = 1). For SPM, the LSB should be cleared.

Indirect Program Addressing, IJMP and ICALL

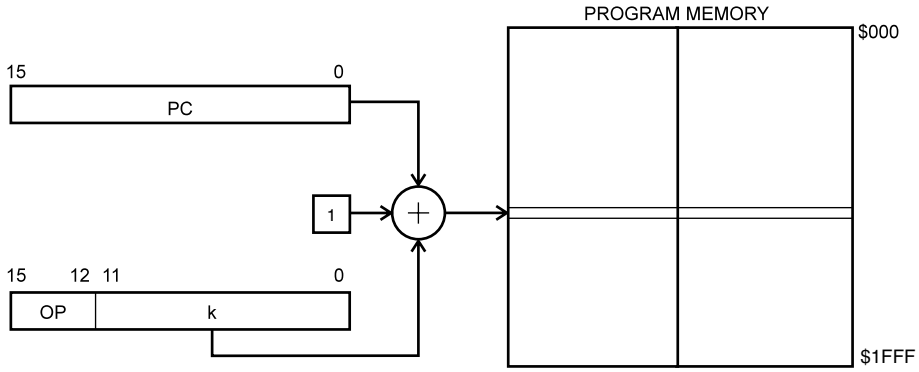
Figure 19. Indirect Program Memory Addressing



Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

Relative Program Addressing, RJMP and RCALL

Figure 20. Relative Program Memory Addressing



Program execution continues at address PC + k + 1.
The relative address k is from -2,048 to 2,047.

The EEPROM Data Memory

The ATmega163 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 on page 62 specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For the SPI data downloading, see page 154 for a detailed description.

Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the System Clock Ø, directly generated from the main Oscillator for the chip. No internal clock division is used.

Figure 21 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 21. The Parallel Instruction Fetches and Instruction Executions

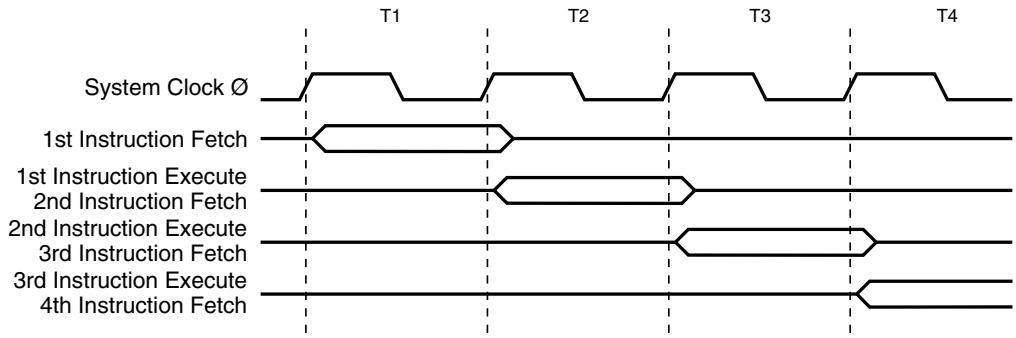
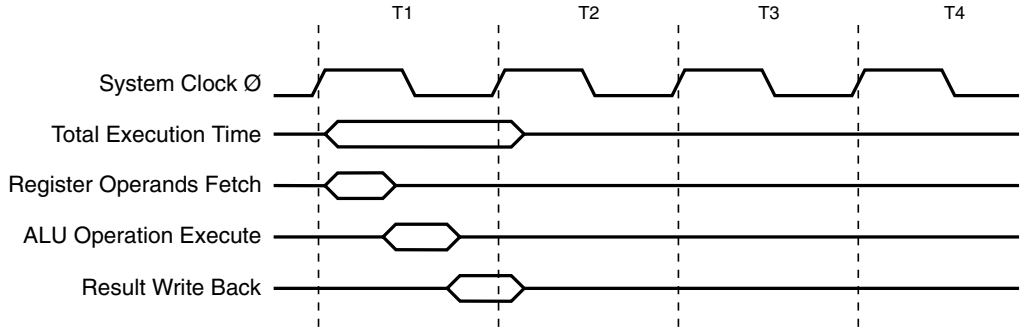


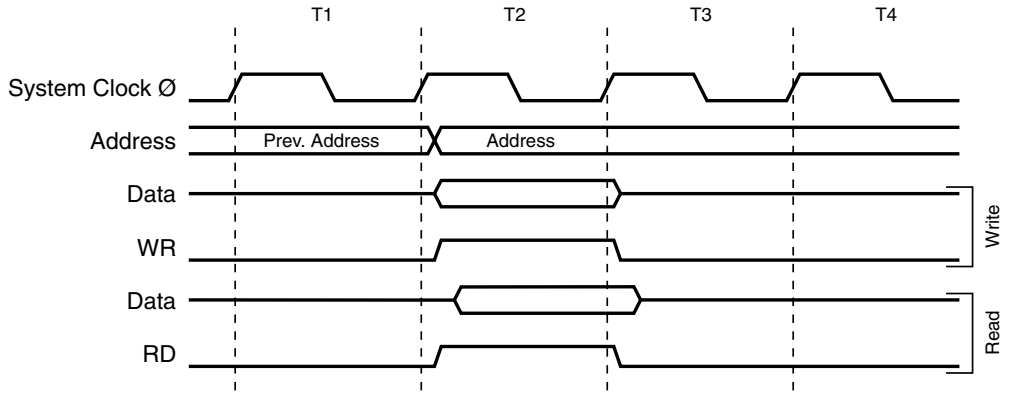
Figure 22 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 22. Single Cycle ALU Operation



The internal data SRAM access is performed in two System Clock cycles as described in Figure 23.

Figure 23. On-chip Data SRAM Access Cycles



I/O Memory

The I/O space definition of the ATmega163 is shown in the following table:

Table 2. ATmega163 I/O Space [1]

I/O Address (SRAM Address)	Name	Function
$3F ($5F)	SREG	Status REGister
$3E ($5E)	SPH	Stack Pointer High
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK Register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK Register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag Register
$37 ($57)	SPMCR	SPM Control Register
$36 ($56)	TWCR	Two-wire Serial Interface Control Register
$35 ($55)	MCUCR	MCU general Control Register
$34 ($54)	MCUSR	MCU general Status Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register

Table 2. ATmega163 I/O Space (Continued)[1]

I/O Address (SRAM Address)	Name	Function
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$31 ($51)	OSCCAL	Oscillator Calibration Register
$30 ($50)	SFIOR	Special Function I/O Register
$2F ($4F)	TCCR1A	Timer/Counter1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter1 High-byte
$2C ($4C)	TCNT1L	Timer/Counter1 Low-byte
$2B ($4B)	OCR1AH	Timer/Counter1 Output Compare Register A High-byte
$2A ($4A)	OCR1AL	Timer/Counter1 Output Compare Register A Low-byte
$29 ($49)	OCR1BH	Timer/Counter1 Output Compare Register B High-byte
$28 ($48)	OCR1BL	Timer/Counter1 Output Compare Register B Low-byte
$27 ($47)	ICR1H	T/C 1 Input Capture Register High-byte
$26 ($46)	ICR1L	T/C 1 Input Capture Register Low-byte
$25 ($45)	TCCR2	Timer/Counter2 Control Register
$24 ($44)	TCNT2	Timer/Counter2 (8-bit)
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register
$22 ($42)	ASSR	Asynchronous Mode Status Register
$21 ($41)	WDTCR	Watchdog Timer Control Register
$20 ($40)	UBRRHI	UART Baud Rate Register High-byte
$1F ($3F)	EEARH	EEPROM Address Register High-byte
$1E ($3E)	EEARL	EEPROM Address Register Low-byte
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$1B ($3B)	PORTA	Data Register, Port A
$1A ($3A)	DDRA	Data Direction Register, Port A
$19 ($39)	PINA	Input Pins, Port A
$18 ($38)	PORTB	Data Register, Port B
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$15 ($35)	PORTC	Data Register, Port C
$14 ($34)	DDRC	Data Direction Register, Port C
$13 ($33)	PINC	Input Pins, Port C
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D

Table 2. ATmega163 I/O Space (Continued)[1]

I/O Address (SRAM Address)	Name	Function
$0F ($2F)	SPDR	SPI I/O Data Register
$0E ($2E)	SPSR	SPI Status Register
$0D ($2D)	SPCR	SPI Control Register
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	UCSRA	UART Control and Status Register A
$0A ($2A)	UCSRB	UART Control and Status Register B
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register
$07 ($27)	ADMUX	ADC Multiplexer Select Register
$06 ($26)	ADCSR	ADC Control and Status Register
$05 ($25)	ADCH	ADC Data Register High
$04 ($24)	ADCL	ADC Data Register Low
$03 ($23)	TWDR	Two-wire Serial Interface Data Register
$02 ($22)	TWAR	Two-wire Serial Interface (Slave) Address Register
$01 ($21)	TWSR	Two-wire Serial Interface Status Register
$00 ($20)	TWBR	Two-wire Serial Interface Bit Rate Register

Note: 1. Reserved and unused locations are not shown in the table.

All ATmega163 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 chapter 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 SRAM, $20 must be added to these addresses. All I/O Register addresses throughout this document are shown with the SRAM address in parentheses.

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 the following sections.

The Status Register – SREG

The *AVR* Status Register – SREG – at I/O space location $3F ($5F) is defined as:

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	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 (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in the Interrupt Mask Registers. If the Global Interrupt Enable Register is cleared (zero), none of the interrupts are enabled independent of the values of the Interrupt Mask Registers. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source and 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. 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.

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

The Stack Pointer – SP

The ATmega163 Stack Pointer is implemented as two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the ATmega163 data memory has $460 locations, 11 bits are used.

Bit	15	14	13	12	11	10	9	8	
$3E ($5E)	–	–	–	–	–	SP10	SP9	SP8	SPH
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	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	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

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 and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

Reset and Interrupt Handling

The ATmega163 provides 17 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 set (one) together with the I-bit in the Status Register in order to enable the interrupt.

The lowest addresses in the Program Memory space are automatically defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in Table 3. 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, etc.

Table 3. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	UART, RXC	UART, Rx Complete

Table 3. Reset and Interrupt Vectors (Continued)

Vector No.	Program Address	Source	Interrupt Definition
13	$018	UART, UDRE	UART Data Register Empty
14	$01A	UART, TXC	UART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface

Note: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see “Boot Loader Support” on page 134.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega163 is:

```
Address   Labels  Code            Comments
$000              jmp     RESET       ; Reset Handler
$002              jmp     EXT_INT0    ; IRQ0 Handler
$004              jmp     EXT_INT1    ; IRQ1 Handler
$006              jmp     TIM2_COMP   ; Timer2 Compare Handler
$008              jmp     TIM2_OVF    ; Timer2 Overflow Handler
$00a              jmp     TIM1_CAPT   ; Timer1 Capture Handler
$00c              jmp     TIM1_COMPA  ; Timer1 Compare A Handler
$00e              jmp     TIM1_COMPB  ; Timer1 Compare B Handler
$010              jmp     TIM1_OVF    ; Timer1 Overflow Handler
$012              jmp     TIM0_OVF    ; Timer0 Overflow Handler
$014              jmp     SPI_STC     ; SPI Transfer Complete Handler
$016              jmp     UART_RXC    ; UART RX Complete Handler
$018              jmp     UART_DRE    ; UDR Empty Handler
$01a              jmp     UART_TXC    ; UART TX Complete Handler
$01c              jmp     ADC     ; ADC Conversion Complete Interrupt Handler
$01e              jmp     EE_RDY      ; EEPROM Ready Handler
$020              jmp     ANA_COMP    ; Analog Comparator Handler
$022              jmp     TWI     ; Two-wire Serial Interface Interrupt Handler
;
$024    MAIN:     ldi     r16,high(RAMEND) ; Main program start
$025              out     SPH,r16     ; Set stack pointer to top of RAM
$026              ldi     r16,low(RAMEND)
$027              out     SPL,r16
...     ...       ...
```

When the BOOTRST Fuse is programmed and the Boot section size set to 512 bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega163 is:

```
Address   Labels  Code              Comments
$002              jmp     EXT_INT0    ; IRQ0 Handler
...      ...      ...
$022              jmp     TWI     ; Two-wire Serial Interface Interrupt Handler
;
$024      MAIN:   ldi     r16,high(RAMEND) ; Main program start
$025              out     SPH,r16     ; Set stack pointer to top of RAM
$026              ldi     r16,low(RAMEND)
$027              out     SPL,r16
$028              <instr> xxx
;
.org $1f00
$1f00             jmp     RESET       ; Reset Handler
```

Reset Sources

The ATmega163 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 more than 500 ns.
- 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}).

During Reset, all I/O Registers are set to their initial values, and the program starts execution from address $000 (unless the BOOTRST Fuse is programmed, as explained above). The instruction placed in this address location must be a JMP – absolute jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 24 shows the Reset Logic. Table 4 and Table 5 define the timing and electrical parameters of the reset circuitry.

Figure 24. Reset Logic

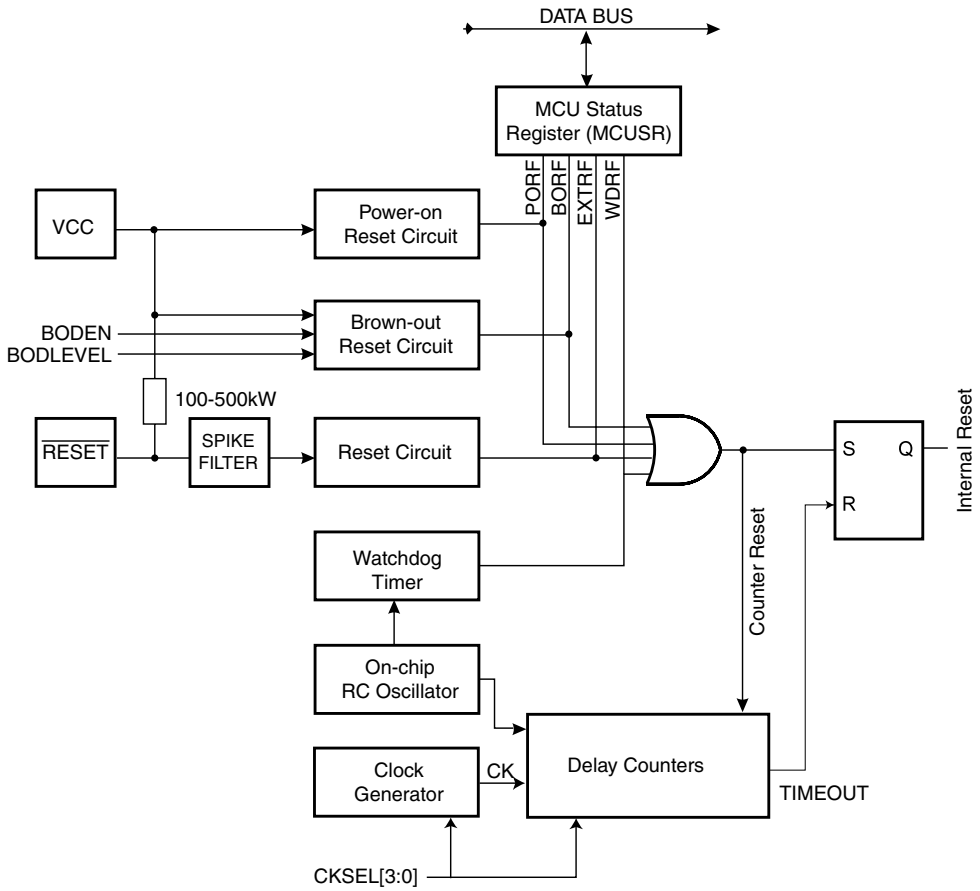


Table 4. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)		1.0	1.4	1.8	V
	Power-on Reset Threshold Voltage (falling)[1]		0.4	0.6	0.8	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		–	–	0.85 V_{CC}	V
V_{BOT}	Brown-out Reset Threshold Voltage	(BODLEVEL = 1)	2.4	2.7	3.2	V
		(BODLEVEL = 0)	3.5	4.0	4.5	

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

Table 5. Reset Delay Selections[1]

CKSEL[2]	Start-up Time, V_{CC} = 2.7V, BODLEVEL Unprogrammed	Start-up Time, V_{CC} = 4.0V, BODLEVEL Programmed	Recommended Usage[3]
0000	4.2 ms + 6 CK	5.8 ms + 6 CK	Ext. Clock, fast rising power
0001	30 µs + 6 CK[4]	10 µs + 6 CK[5]	Ext. Clock, BOD enabled
0010[6]	67 ms + 6 CK	92 ms + 6 CK	Int. RC Oscillator, slowly rising power
0011	4.2 ms + 6 CK	5.8 ms + 6 CK	Int. RC Oscillator, fast rising power
0100	30 µs + 6 CK[4]	10 µs + 6 CK[5]	Int. RC Oscillator, BOD enabled
0101	67 ms + 6 CK	92 ms + 6 CK	Ext. RC Oscillator, slowly rising power
0110	4.2 ms + 6 CK	5.8 ms + 6 CK	Ext. RC Oscillator, fast rising power
0111	30 µs + 6 CK[4]	10 µs + 6 CK[5]	Ext. RC Oscillator, BOD enabled
1000	67ms + 32K CK	92 ms + 32K CK	Ext. Low-frequency Crystal
1001	67 ms + 1K CK	92 ms + 1K CK	Ext. Low-frequency Crystal
1010	67 ms + 16K CK	92 ms + 16K CK	Crystal Oscillator, slowly rising power
1011	4.2 ms + 16K CK	5.8 ms + 16K CK	Crystal Oscillator, fast rising power
1100	30 µs + 16K CK[4]	10 µs + 16K CK[5]	Crystal Oscillator, BOD enabled
1101	67 ms + 1K CK	92 ms + 1K CK	Ceramic Resonator/Ext. Clock, slowly rising power
1110	4.2 ms + 1K CK	5.8 ms + 1K CK	Ceramic Resonator, fast rising power
1111	30 µs + 1K CK[4]	10 µs + 1K CK[5]	Ceramic Resonator, BOD enabled

Notes:
1. On power-up, the start-up time is increased with typ. 0.6 ms.
2. “1” means unprogrammed, “0” means programmed.
3. For possible clock selections, see “Clock Options” on page 5.
4. When BODEN is programmed, add 100 µs.
5. When BODEN is programmed, add 25 µs.
6. Default value.

Table 5 shows the Start-up Times from Reset. When the CPU wakes up from Power-down or Power-save, only the clock counting part of the start-up time is used. The Watchdog Oscillator is used for timing the 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 the Electrical Characteristics section. The device is shipped with CKSEL = “0010” (Int. RC Oscillator, slowly rising power).

Table 6. Number of Watchdog Oscillator Cycles[1]

BODLEVEL	V_{CC} Condition	Time-out	Number of Cycles
Unprogrammed	2.7V	30 µs	8
Unprogrammed	2.7V	130 µs	32
Unprogrammed	2.7V	4.2 ms	1K
Unprogrammed	2.7V	67 ms	16K
Programmed	4.0V	10 µs	8
Programmed	4.0V	35 µs	32
Programmed	4.0V	5.8 ms	4K
Programmed	4.0V	92 ms	64K

Note: 1. The Bodlevel Fuse can be used to select start-up times even if the Brown-out Detection is disabled (BODEN Fuse unprogrammed).

Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 4. 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 a delay counter, which determines the delay, for which the device is kept in RESET after V_{CC} rise. The Time-out Period of the delay counter can be defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in Table 5. The RESET signal is activated again, without any delay, when the V_{CC} decreases below detection level.

Figure 25. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}.

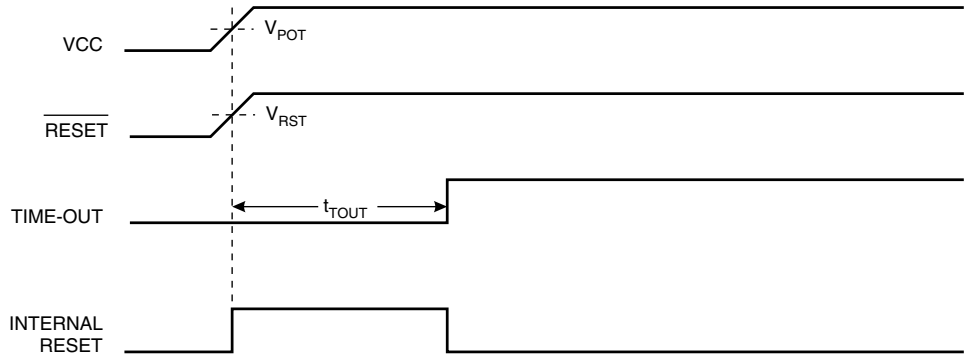
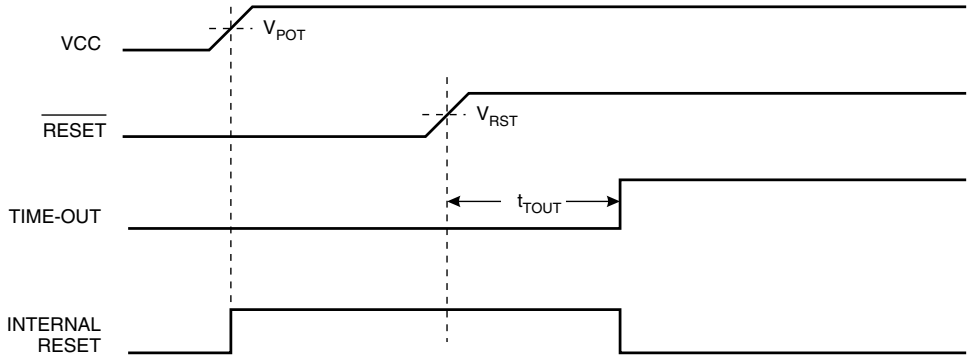


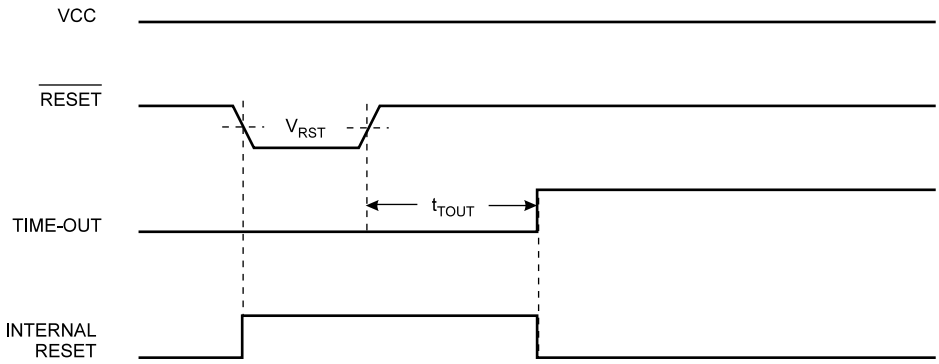
Figure 26. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 500 ns 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 timer starts the MCU after the Time-out Period t_{TOUT} has expired.

Figure 27. External Reset During Operation

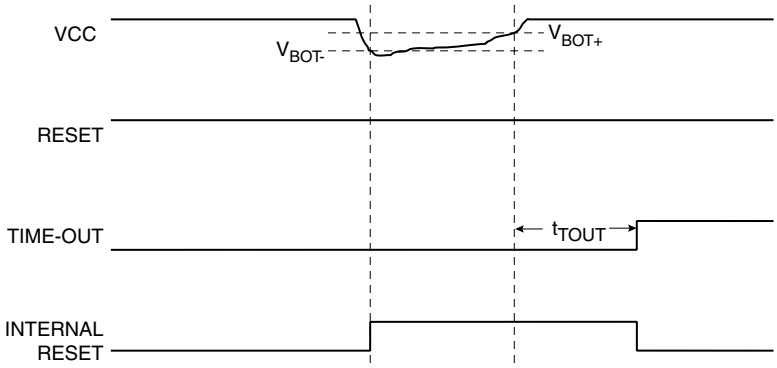


Brown-out Detection

ATmega163 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during the operation. 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, the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level, the Brown-out Reset is deactivated after a delay. The delay is defined by the user in the same way as the delay of POR signal, in Table 5. 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 of 50 mV to ensure spike free Brown-out Detection.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than 9 µs for trigger level 4.0V, 21 µs for trigger level 2.7V (typical values).

Figure 28. Brown-out Reset During Operation

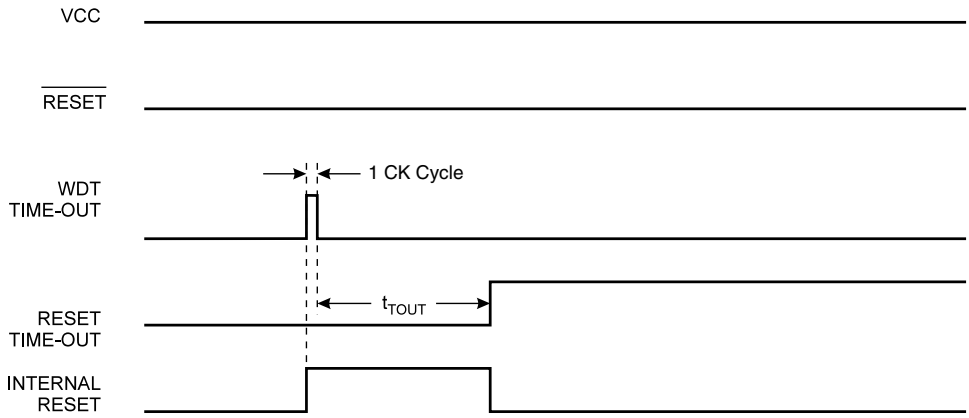


The hysteresis on V_{BOT}: $V_{BOT+} = V_{BOT} + 25$ mV, $V_{BOT-} = V_{BOT} - 25$ mV

Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out Period t_{TOUT}. Refer to page 60 for details on operation of the Watchdog Timer.

Figure 29. Watchdog Reset During Operation



MCU Status Register – MCUSR

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

Bit	7	6	5	4	3	2	1	0	
$34 ($54)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the Flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the Flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

Internal Voltage Reference

ATmega163 features an internal bandgap reference with a nominal voltage of 1.22V. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator and ADC. The 2.56V reference to the ADC is also generated from the internal bandgap reference.

Voltage Reference Enable Signals and Start-up Time

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).
3. When the ADC is enabled.

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. The bandgap reference uses typically 10 µA, and to reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

Interrupt Handling

The ATmega163 has two 8-bit Interrupt Mask Control Registers: GIMSK – General Interrupt Mask Register and TIMSK – Timer/Counter Interrupt Mask Register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software must set (one) the I-bit to enable nested interrupts. The I-bit is set (one) when a Return from Interrupt instruction – RETI – is executed.

When the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the 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 when the corresponding interrupt enable bit is cleared (zero), the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software.

If one or more interrupt conditions occur when the Global Interrupt Enable bit is cleared (zero), the corresponding interrupt flag(s) will be set and remembered until the Global Interrupt Enable bit is set (one), and will be executed by order of priority.

Note that external level interrupt does not have a flag, and will only be remembered for as long as the interrupt condition is present.

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

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 (13 bits) 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.

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

The General Interrupt Mask Register – GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	**INT1**	**INT0**	–	–	–	–	–	–	**GIMSK**
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	x	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 activated. 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 program memory address $004. See also “External Interrupts”.

- **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 activated. 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 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 Program Memory address $002. See also “External Interrupts.”

- **Bits 5 – Res: Reserved Bits**

This bit is reserved in the ATmega163 and the read value is undefined.

- **Bits 4..0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

The General Interrupt Flag Register – GIFR

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

- **Bit 7 – INTF1: External Interrupt Flag1**

When an edge on the INT1 pin triggers an interrupt request, the correspnding Interrupt Flag, INTF1, becomes set (one). If the I-bit in SREG and the corresponding Interrupt Enable bit, INT1 in GIMSK are set (one), the MCU will jump to the 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 Flag0**

When an event on the INT0 pin triggers an interrupt request, the corresponding Interrupt Flag, INTF0 becomes set (one). If the I-bit in SREG and the corresponding Interrupt Enable bit, INT0 in GIMSK are set (one), the MCU will jump to the 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.

- **Bits 5..0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

The Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
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 – OCIE2: Timer/Counter2 Output Compare Match Interrupt Enable**

When the OCIE2 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match Interrupt is enabled. The corresponding interrupt (at vector $006) is executed if a Compare Match in Timer/Counter2 occurs, i.e., when the OCF2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 6 – TOIE2: Timer/Counter2 Overflow Interrupt Enable**

When the TOIE2 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter2 Overflow Interrupt is enabled. The corresponding interrupt (at vector $008) is executed if an overflow in Timer/Counter2 occurs, i.e., when the TOV2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 5 – TICIE1: Timer/Counter1 Input Capture Interrupt Enable**

When the TICIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Input Capture Event Interrupt is enabled. The corresponding interrupt

(at vector $00A) is executed if a capture triggering event occurs on PD6 (ICP), i.e., when the ICF1 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 4 – OCIE1A: Timer/Counter1 Output CompareA Match Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Compare A Match Interrupt is enabled. The corresponding interrupt (at vector $00C) is executed if a Compare A Match in Timer/Counter1 occurs, i.e., when the OCF1A bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 3 – OCIE1B: Timer/Counter1 Output CompareB Match Interrupt Enable**

When the OCIE1B bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Compare B Match Interrupt is enabled. The corresponding interrupt (at vector $00E) is executed if a Compare B Match in Timer/Counter1 occurs, i.e., when the OCF1B bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

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

When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow Interrupt is enabled. The corresponding interrupt (at vector $010) is executed if an overflow in Timer/Counter1 occurs, i.e., when the TOV1 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and always reads as zero.

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

When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow Interrupt is enabled. The corresponding interrupt (at vector $012) 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.

The Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
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	x	0	

- **Bit 7 – OCF2: Output Compare Flag2**

The OCF2 bit is set (one) when a Compare Match occurs between the Timer/Counter2 and the data in OCR2 – Output Compare Register2. OCF2 is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, OCF2 is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE2 (Timer/Counter2 Compare Match Interrupt Enable), and the OCF2 are set (one), the Timer/Counter2 Compare Match Interrupt is executed.

- **Bit 6 – TOV2: Timer/Counter2 Overflow Flag**

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding inteRrupt Handling Vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the SREG I-bit, and TOIE2 (Timer/Counter2 Overflow Interrupt Enable), and TOV2 are set (one), the

Timer/Counter2 Overflow Interrupt is executed. In up/down PWM mode, this bit is set when Timer/Counter2 changes counting direction at $00.

- **Bit 5 – ICF1: Input Capture Flag1**

The ICF1 bit is set (one) to Flag an Input Capture Event, indicating that the Timer/Counter1 value has been transferred to the Input Capture Register – ICR1. ICF1 is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, ICF1 is cleared by writing a logic one to the flag.

- **Bit 4 – OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when a Compare Match occurs between the Timer/Counter1 and the data in OCR1A – Output Compare Register 1A. OCF1A is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, OCF1A is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE1A (Timer/Counter1 Compare Match Interrupt A Enable), and the OCF1A are set (one), the Timer/Counter1A Compare Match Interrupt is executed.

- **Bit 3 – OCF1B: Output Compare Flag 1B**

The OCF1B bit is set (one) when a Compare Match occurs between the Timer/Counter1 and the data in OCR1B – Output Compare Register 1B. OCF1B is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, OCF1B is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE1B (Timer/Counter1 Compare Match Interrupt B Enable), and the OCF1B are set (one), the Timer/Counter1B Compare Match Interrupt is executed.

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

The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, TOV1 is cleared by writing a logic one to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In up/down PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and the read value is undefined.

- **Bit 0 – 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, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

External Interrupts

The external interrupts are triggered by the INT0 and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0/INT1 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. This is set up as indicated in the specification for the MCU Control Register – MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	–	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and always reads as zero.

- **Bit 6 – SE: Sleep Enable**

The SE bit must be set (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 set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

- **Bits 5, 4 – SM1/SM0: Sleep Mode Select Bits 1 and 0**

These bits select between the three available sleep modes as shown in Table 7.

Table 7. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	ADC Noise Reduction
1	0	Power-down
1	1	Power-save

- **Bits 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-Flag and the corresponding interrupt mask in the GIMSK are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 8. 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 8. 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 9. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 9. 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.

Sleep Modes

To enter any of the four sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. The SM1 and SM0 bits in the MCUCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, or Power-save) will be activated by the SLEEP instruction. See Table 7 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, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File, SRAM, and I/O memory 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.

Idle Mode

When the SM1/SM0 bits are set to 00, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, UART, Analog Comparator, ADC, Two-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating (if enabled). This enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and UART Receive 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. If the ADC is enabled, a conversion starts automatically when this mode is entered.

ADC Noise Reduction Mode

When the SM1/SM0 bits are set to 01, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the external interrupts, the Two-wire Serial Interface address watch, Timer/Counter2 and the Watchdog to continue operating (if enabled). This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC Conversion Complete interrupt, only an External Reset, a Watchdog Reset (if enabled), a Brown-out Reset, a Two-wire Serial Interface address match interrupt, or an external level interrupt can wake up the MCU from ADC Noise Reduction Mode. A Timer/Counter2 Output Compare or overflow event will wake up the MCU, but will not generate an interrupt unless Timer/Counter2 is clocked asynchronously.

In future devices this is subject to change. It is recommended for future code compatibility to disable Timer/Counter2 interrupts during ADC Noise Reduction mode if the Timer/Counter2 is clocked synchronously.

Power-down Mode

When the SM1/SM0 bits are 10, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the Two-wire Serial Interface address match, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, or an external level interrupt can wake up the MCU.

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, and if the input has the required level during this time, the MCU will wake up. 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 the Electrical Characteristics section.

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 seen in Table 5 on page 25.

Power-save Mode

When the SM1/SM0 bits are 11, the SLEEP instruction forces the MCU into the Power-save mode. This mode is identical to Power-down, with one exception:

If Timer/Counter2 is clocked asynchronously, i.e., the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK, and the global interrupt enable bit in SREG is set.

If the asynchronous timer is NOT clocked asynchronously, Power-down mode is recommended instead of Power-save mode because the contents of the registers in the asynchronous timer should be considered undefined after wake-up in Power-save mode if AS2 is 0.

Calibrated Internal RC Oscillator

The calibrated internal Oscillator provides a fixed 1 MHz (nominal) clock at 5V and 25°C. This clock may be used as the system clock. See the section “Clock Options” on page 5 for information on how to select this clock as the system clock. This Oscillator can be calibrated by writing the calibration byte to the OSCCAL Register. 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. At 5V and 25ºC, the pre-programmed calibration byte gives a frequency within ± 1% of the nominal frequency. For details on how to use the pre-programmed calibration value, see “Calibration Byte” on page 144.

Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0
$31 ($51)	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
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 – 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. 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 operation may fail. Note that the Oscillator is intended for calibration to 1.0MHz, thus tuning to other values is not guaranteed.

Table 10. Internal RC Oscillator Frequency Range.

OSCCAL Value	Min Frequency (MHz)	Max Frequency (MHz)
$00	0.5	1.0
$7F	0.7	1.5
$FF	1.0	2.0

Special Function I/O Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
$30 ($50)	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

- **Bit 3 – ACME: Analog Comparator Multiplexer Enable**

When this bit is set (one) and the ADC is switched off (ADEN in ADCSR is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is cleared (zero), AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see “Analog Comparator Multiplexed Input” on page 104.

- **Bit 2 – PUD: Pull-up Disable**

When this bit is set (one), all pull-ups on all ports are disabled. If the bit is cleared (zero), the pull-ups can be individually enabled as described in the chapter “I/O Ports” on page 115.

- **Bit 1 – PSR2: Prescaler Reset Timer/Counter2**

When this bit is set (one) the Timer/Counter2 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. This bit will always be read as zero if Timer/Counter2 is clocked by the internal CPU clock. If this bit is written when Timer/Counter2 is operating in asynchronous mode. The bit will remain one until the prescaler has been reset. See “Asynchronous Operation of Timer/Counter2” on page 58 for a detailed description of asynchronous operation.

- **Bit 0 – PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0**

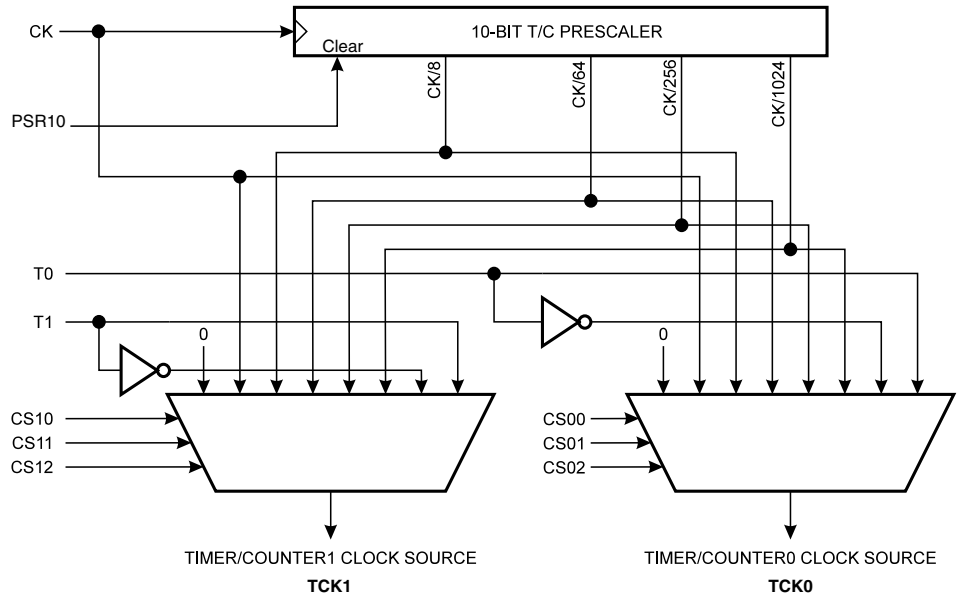
When this bit is set (one) the Timer/Counter1 and Timer/Counter0 Prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler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

Timer/Counters

The ATmega163 provides three general purpose Timer/Counters – two 8-bit T/Cs and one 16-bit T/C. Timer/Counter2 can optionally be asynchronously clocked from an external Oscillator. This Oscillator is optimized for use with a 32.768 kHz watch crystal, enabling use of Timer/Counter2 as a Real Time Clock (RTC). Timer/Counter0 and Timer/Counter1 have individual prescaling selection from the same 10-bit prescaler. Timer/Counter2 has its own prescaler. Both these prescalers can be reset by setting the corresponding control bits in the Special Functions I/O Register (SFIOR). These Timer/Counters can either be used as a timer with an internal clock time-base or as a counter with an external pin connection which triggers the counting.

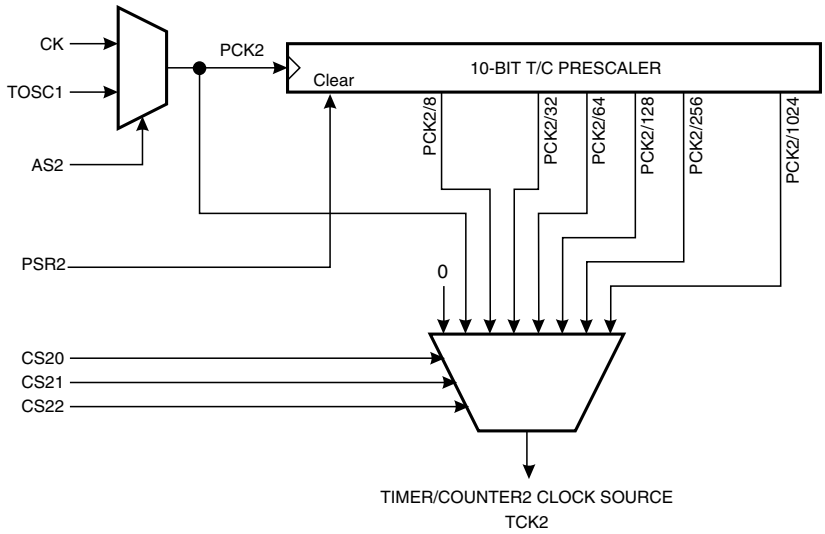
Timer/Counter Prescalers

Figure 30. Prescaler for Timer/Counter0 and Timer/Counter1



For Timer/Counter0 and Timer/Counter1, the four different prescaled selections are: CK/8, CK/64, CK/256, and CK/1024, where CK is the Oscillator clock. For the two Timer/Counter0 and Timer/Counter1, CK, external source, and stop can also be selected as clock sources. Setting the PSR10 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a Prescaler Reset will affect both Timer/Counters.

Figure 31. Prescaler for Timer/Counter2



The clock source for Timer/Counter2 is named PCK2. PCK2 is by default connected to the main system clock CK. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the PC6(TOSC1) pin. This enables use of Timer/Counter2 as a Real Time Clock (RTC). When AS2 is set, pins PC6(TOSC1) and PC7(TOSC2) are disconnected from Port C. A crystal can then be connected between the PC6(TOSC1) and PC7(TOSC2) pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768 kHz crystal. Applying an external clock source to TOSC1 is not recommended. Setting the PSR2 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler.

8-bit Timer/Counter0

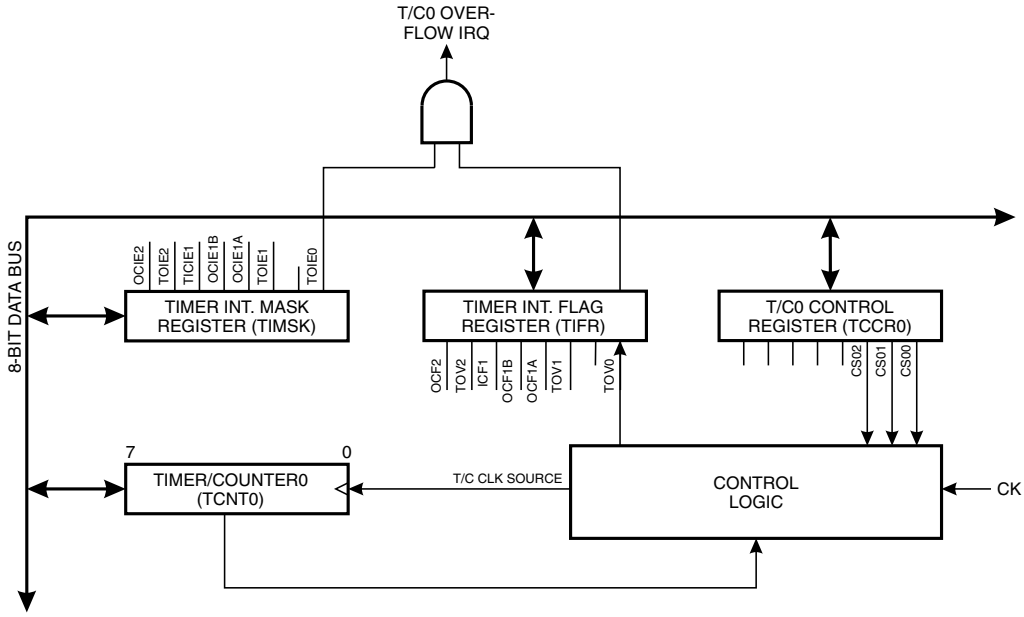
Figure 32 shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in “Timer/Counter0 Control Register – TCCR0” on page 41. The overflow Status Flag is found in “The Timer/Counter Interrupt Flag Register – TIFR” on page 32. Control signals are found in the Timer/Counter0 Control Register – TCCR0. The interrupt enable/disable settings for Timer/Counter0 are found in “The Timer/Counter Interrupt Mask Register – TIMSK” on page 31.

When Timer/Counter0 is externally clocked, the external signal is synchronized with the Oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

Figure 32. Timer/Counter0 Block Diagram



Timer/Counter0 Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

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

The Clock Select0 bits 2,1, and 0 define the prescaling source of Timer0.

Table 11. Clock0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The prescaled CK modes are scaled directly from the CK Oscillator clock. If the external pin modes are used for Timer/Counter0, transitions on PB0/(T0) will clock the counter even if the pin is configured as an output. This feature can give the user SW control of the counting.

Timer/Counter 0 – TCNT0

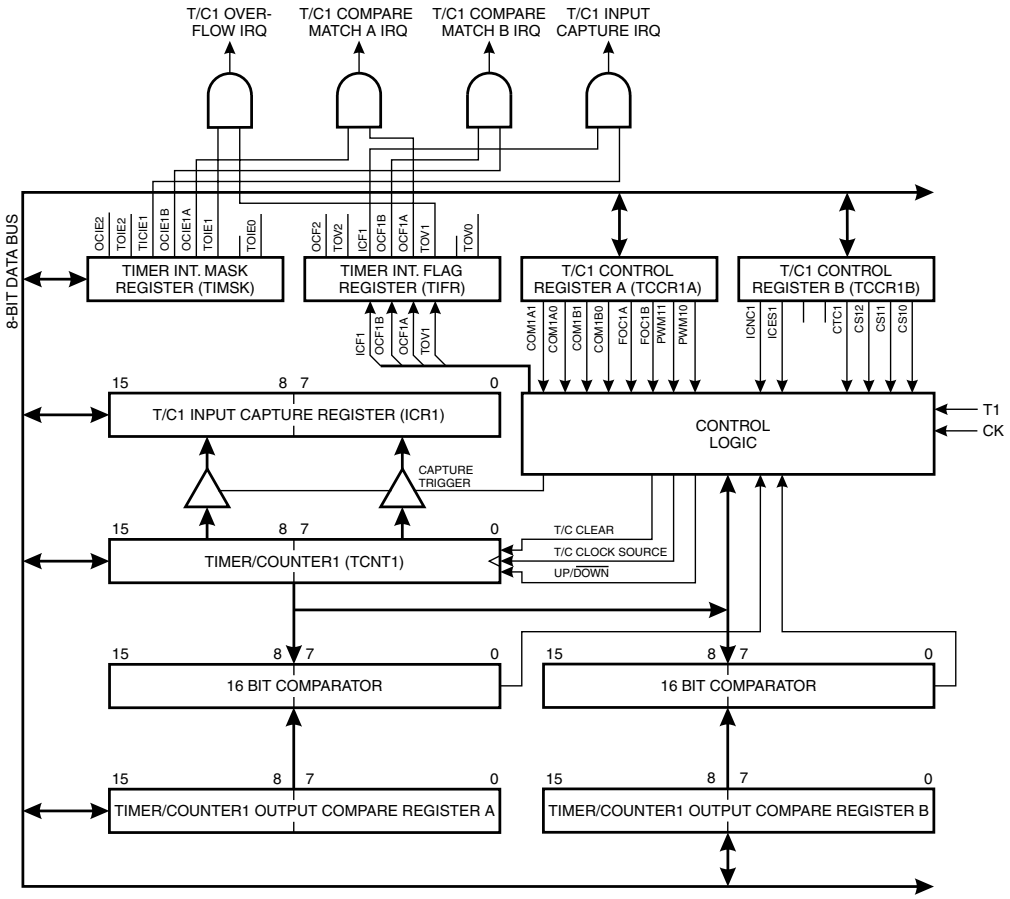
Bit	7	6	5	4	3	2	1	0	
$34 ($54)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter0 is implemented as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the clock cycle following the write operation.

16-bit Timer/Counter1

Figure 33 shows the block diagram for Timer/Counter1.

Figure 33. Timer/Counter1 Block Diagram



The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in section “Timer/Counter1 Control Register B – TCCR1B” on page 45. The different Status Flags (Overflow, Compare Match, and Capture Event) are found in the Timer/Counter Interrupt Flag Register – TIFR. Control signals are found in the Timer/Counter1 Control Registers – TCCR1A and TCCR1B. The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register – TIMSK.

When Timer/Counter1 is externally clocked, the external signal is synchronized with the Oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU

clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

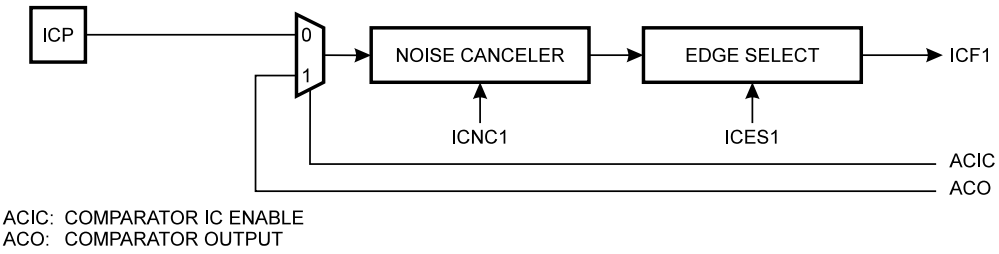
The 16-bit Timer/Counter1 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B (OCR1A and OCR1B) as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions includes optional clearing of the counter on Compare A Match, and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as an 8-, 9-, or 10-bit Pulse Width Modulator (PWM). In this mode the counter and the OCR1A/OCR1B Registers serve as a dual glitch-free stand-alone PWM with centered pulses. Alternatively, the Timer/Counter1 can be configured to operate at twice the speed in PWM mode, but without centered pulses. Refer to page 48 for a detailed description of this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register – ICR1, triggered by an external event on the Input Capture Pin – ICP. The actual capture event settings are defined by the Timer/Counter1 Control Register – TCCR1B. In addition, the Analog Comparator can be set to trigger the Input Capture. Refer to the section, “The Analog Comparator” on page 102, for details on this. The ICP pin logic is shown in Figure 34.

Figure 34. ICP Pin Schematic Diagram



If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over four samples, and all four must be equal to activate the Capture Flag.

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 6 – COM1A1, COM1A0: Compare Output Mode1A, Bits 1 and 0**

The COM1A1 and COM1A0 control bits determine any output pin action following a Compare Match in Timer/Counter1. Any output pin actions affect pin OC1A – Output Compare A. This is an alternative function to an I/O Port, and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 10.

- **Bits 5, 4 – COM1B1, COM1B0: Compare Output Mode1B, Bits 1 and 0**

The COM1B1 and COM1B0 control bits determine any output pin action following a Compare Match in Timer/Counter1. Any output pin actions affect pin OC1B – Output Compare B. This is an alternative function to an I/O port, and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 10.

Table 12. Compare 1 Mode Select[1]

COM1X1	COM1X0	Description
0	0	Timer/Counter1 disconnected from output pin OC1X
0	1	Toggle the OC1X output line.
1	0	Clear the OC1X output line (to zero).
1	1	Set the OC1X output line (to one).

Note: 1. X = A or B.

In PWM mode, these bits have a different function. Refer to Table 14 for a detailed description.

- **Bit 3 – FOC1A: Force Output Compare1A**

Writing a logical one to this bit, forces a change in the Compare Match Output pin PD5 according to the values already set in COM1A1 and COM1A0. If the COM1A1 and COM1A0 bits are written in the same cycle as FOC1A, the new settings will not take effect until next Compare Match or Forced Compare Match occurs. The Force Output Compare bit can be used to change the output pin without waiting for a Compare Match in the Timer. The automatic action programmed in COM1A1 and COM1A0 happens as if a Compare Match had occurred, but no interrupt is generated and it will not clear the timer even if CTC1 in TCCR1B is set. The corresponding I/O pin must be set as an output pin for the FOC1A bit to have effect on the pin. The FOC1A bit will always be read as zero. The setting of the FOC1A bit has no effect in PWM mode.

- **Bit 2 – FOC1B: Force Output Compare1B**

Writing a logical one to this bit, forces a change in the Compare Match Output pin PD4 according to the values already set in COM1B1 and COM1B0. If the COM1B1 and COM1B0 bits are written in the same cycle as FOC1B, the new settings will not take effect until next Compare Match or Forced Compare Match occurs. The Force Output Compare bit can be used to change the output pin without waiting for a Compare Match

in the Timer. The automatic action programmed in COM1B1 and COM1B0 happens as if a Compare Match had occurred, but no interrupt is generated. The corresponding I/O pin must be set as an output pin for the FOC1B bit to have effect on the pin. The FOC1B bit will always be read as zero. The setting of the FOC1B bit has no effect in PWM mode.

- **Bits 1..0 – PWM11, PWM10: Pulse Width Modulator Select Bits**

These bits select PWM operation of Timer/Counter1 as specified in Table 11. This mode is described on page 48.

Table 13. PWM Mode Select

PWM11	PWM10	Description
0	0	PWM operation of Timer/Counter1 is disabled
0	1	Timer/Counter1 is an 8-bit PWM
1	0	Timer/Counter1 is a 9-bit PWM
1	1	Timer/Counter1 is a 10-bit PWM

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	–	–	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: Input Capture1 Noise Canceler (4 CKs)**

When the ICNC1 bit is cleared (zero), the Input Capture trigger noise canceler function is disabled. The Input Capture is triggered at the first rising/falling edge sampled on the ICP – Input Capture Pin – as specified. When the ICNC1 bit is set (one), four successive samples are measures on the ICP – Input Capture Pin, and all samples must be high/low according to the Input Capture trigger specification in the ICES1 bit. The actual sampling frequency is XTAL clock frequency.

- **Bit 6 – ICES1: Input Capture1 Edge Select**

While the ICES1 bit is cleared (zero), the Timer/Counter1 contents are transferred to the Input Capture Register – ICR1 – on the falling edge of the Input Capture Pin – ICP. While the ICES1 bit is set (one), the Timer/Counter1 contents are transferred to the Input Capture Register – ICR1 – on the rising edge of the Input Capture Pin – ICP.

- **Bits 5, 4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

- **Bit 3 – CTC1: Clear Timer/Counter1 on Compare Match**

When the CTC1 control bit is set (one), the Timer/Counter1 is Reset to $0000 in the clock cycle after a Compare A Match. If the CTC1 control bit is cleared, Timer/Counter1 continues counting and is unaffected by a Compare Match. When a prescaling of 1 is used, and the Compare A Register is set to C, the timer will count as follows if CTC1 is set:

... | C-1 | C | 0 | 1 |...

When the prescaler is set to divide by eight, the Timer will count like this:

... | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, C, C, C, C, C, C, C | 0, 0, 0, 0, 0, 0, 0, 0 |1,1,1,1,1,1,1,1|...

In PWM mode, this bit has a different function. If the CTC1 bit is cleared in PWM mode, the Timer/Counter1 acts as an up/down counter. If the CTC1 bit is set (one), the Timer/Counter wraps when it reaches the TOP value. Refer to page 48 for a detailed description.

- **Bits 2..0 – CS12, CS11, CS10: Clock Select1, Bit 2, 1, and 0**

The Clock Select1 bits 2, 1, and 0 define the prescaling source of Timer/Counter1.

Table 14. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T1, falling edge
1	1	1	External Pin T1, rising edge

The Stop condition provides a Timer Enable/Disable function. The prescaled modes are scaled directly from the CK Oscillator clock. If the external pin modes are used for Timer/Counter1, transitions on PB1/(T1) will clock the counter even if the pin is configured as an output. This feature can give the user SW control of the counting.

Timer/Counter1 – TCNT1H and TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LSB	TCNT1L
	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	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. 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 register (TEMP). This temporary register is also used when accessing OCR1A, OCR1B, and ICR1. If the main program and also interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program and interrupt routines.

TCNT1 Timer/Counter1 Write

When the CPU writes to the high byte TCNT1H, the written data is placed in the TEMP Register. Next, when the CPU writes the low byte TCNT1L, this byte of data is combined with the byte data in the TEMP Register, and all 16 bits are written to the TCNT1 Timer/Counter1 Register simultaneously. Consequently, the high byte TCNT1H must be accessed first for a full 16-bit register write operation.

TCNT1 Timer/Counter1 Read

When the CPU reads the low byte TCNT1L, the data of the Low Byte TCNT1L is sent to the CPU and the data of the High Byte TCNT1H is placed in the TEMP Register. When the CPU reads the data in the High Byte TCNT1H, the CPU receives the data in the TEMP Register. Consequently, the Low Byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

Timer/Counter1 Output Compare Register – OCR1AH and OCR1AL

Bit	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	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	

Timer/Counter1 Output Compare Register – OCR1BH and OCR1BL

Bit	15	14	13	12	11	10	9	8	
$29 ($49)	MSB								OCR1BH
$28 ($48)								LSB	OCR1BL
	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	

The Output Compare Registers are 16-bit read/write registers.

The Timer/Counter1 Output Compare Registers contain the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status Register. A software write to the Timer/Counter Register blocks compare matches in the next Timer/Counter clock cycle. This prevents immediate interrupts when initializing the Timer/Counter.

A Compare Match will set the compare interrupt flag in the CPU clock cycle following the compare event.

Since the Output Compare Registers – OCR1A and OCR1B – are 16-bit registers, a temporary register TEMP is used when OCR1A/B are written to ensure that both bytes are updated simultaneously. When the CPU writes the high byte, OCR1AH or OCR1BH, the data is temporarily stored in the TEMP Register. When the CPU writes the Low Byte, OCR1AL or OCR1BL, the TEMP Register is simultaneously written to OCR1AH or OCR1BH. Consequently, the high byte OCR1AH or OCR1BH must be written first for a full 16-bit register write operation.

The TEMP Register is also used when accessing TCNT1 and ICR1. If the main program and also interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program and interrupt routines.

Timer/Counter1 Input Capture Register – ICR1H and ICR1L

Bit	15	14	13	12	11	10	9	8	
$27 ($47)	MSB								ICR1H
$26 ($46)								LSB	ICR1L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The Input Capture Register is a 16-bit read-only register.

When the rising or falling edge (according to the input capture edge setting – ICES1) of the signal at the Input Capture Pin – ICP – is detected, the current value of the Timer/Counter1 Register – TCNT1 – is transferred to the Input Capture Register – ICR1. At the same time, the Input Capture Flag – ICF1 – is set (one).

Since the Input Capture Register – ICR1 – is a 16-bit register, a temporary register TEMP is used when ICR1 is read to ensure that both bytes are read simultaneously. When the CPU reads the Low Byte ICR1L, the data is sent to the CPU and the data of the High Byte ICR1H is placed in the TEMP Register. When the CPU reads the data in the High Byte ICR1H, the CPU receives the data in the TEMP Register. Consequently, the Low Byte ICR1L must be accessed first for a full 16-bit register read operation.

The TEMP Register is also used when accessing TCNT1, OCR1A, and OCR1B. If the main program and also interrupt routines accesses registers using TEMP, interrupts must be disabled during access from the main program and interrupt routines.

Timer/Counter1 in PWM Mode

When the PWM mode is selected, Timer/Counter1 and the Output Compare Register1A – OCR1A and the Output Compare Register1B – OCR1B, form a dual 8,- 9-, or 10-bit, free-running, glitch-free, and phase correct PWM with outputs on the PD5 (OC1A) and PD4(OC1B) pins. In this mode, the Timer/Counter1 acts as an up/down counter, counting up from $0000 to TOP (see Table 16), where it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the 8, 9, or 10 least significant bits (depending on resolution) of OCR1A or OCR1B, the PD5(OC1A)/PD4(OC1B) pins are set or cleared according to the settings of the COM1A1/COM1A0 or COM1B1/COM1B0 bits in the Timer/Counter1 Control Register TCCR1A. Refer to Table 12 on page 44 for details.

Alternatively, the Timer/Counter1 can be configured to a PWM that operates at twice the speed as in the mode described above. Then the Timer/Counter1 and the Output Compare Register1A – OCR1A and the Output Compare Register1B – OCR1B, form a dual 8-, 9-, or 10-bit, free-running and glitch-free PWM with outputs on the PD5(OC1A) and PD4(OC1B) pins.

Table 15. Timer TOP Values and PWM Frequency

CTC1	PWM11	PWM10	PWM Resolution	Timer TOP Value	Frequency
0	0	1	8-bit	$00FF (255)	$f_{TCK1}/510$
0	1	0	9-bit	$01FF (511)	$f_{TCK1}/1022$
0	1	1	10-bit	$03FF(1023)	$f_{TCK1}/2046$
1	0	1	8-bit	$00FF (255)	$f_{TCK1}/256$
1	1	0	9-bit	$01FF (511)	$f_{TCK1}/512$
1	1	1	10-bit	$03FF(1023)	$f_{TCK1}/1024$

As shown in Table 15, the PWM operates at either 8, 9, or 10 bits resolution. Note the unused bits in OCR1A, OCR1B, and TCNT1 will automatically be written to zero by hardware. For example, bit 9 to 15 will be set to zero in OCR1A, OCR1B, and TCNT1 if the 9-bit PWM resolution is selected. This makes it possible for the user to perform read-modify-write operations in any of the three resolution modes and the unused bits will be treated as don't care.

Table 16. Timer TOP Values and PWM Frequency

PWM Resolution	Timer TOP Value	Frequency
8-bit	$00FF (255)	$f_{TC1}/510$
9-bit	$01FF (511)	$f_{TC1}/1022$
10-bit	$03FF(1023)	$f_{TC1}/2046$

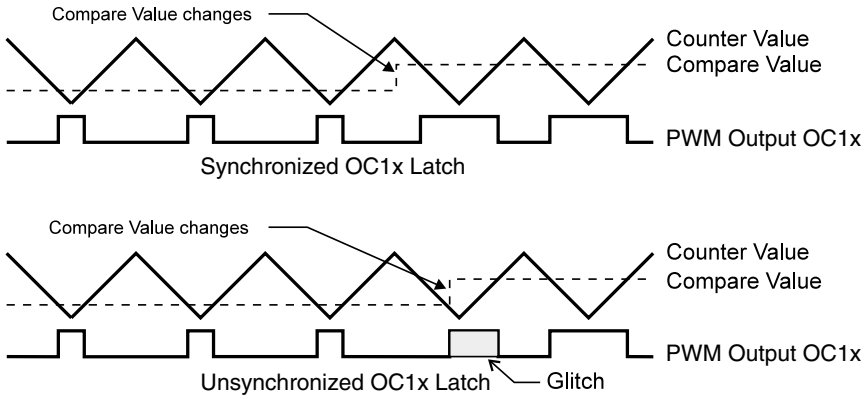
Table 17. Compare1 Mode Select in PWM Mode(1)

CTC1	COM1X 1	COM1X 0	Effect on OCX1
0	0	0	Not connected
0	0	1	Not connected
0	1	0	Cleared on Compare Match, up-counting. Set on Compare Match, down-counting (non-inverted PWM).
0	1	1	Cleared on Compare Match, down-counting. Set on Compare Match, up-counting (inverted PWM).
1	0	0	Not connected
1	0	1	Not connected
1	1	0	Cleared on Compare Match, set on overflow.
1	1	1	Set on Compare Match, cleared on overflow.

Note: 1. X = A or B

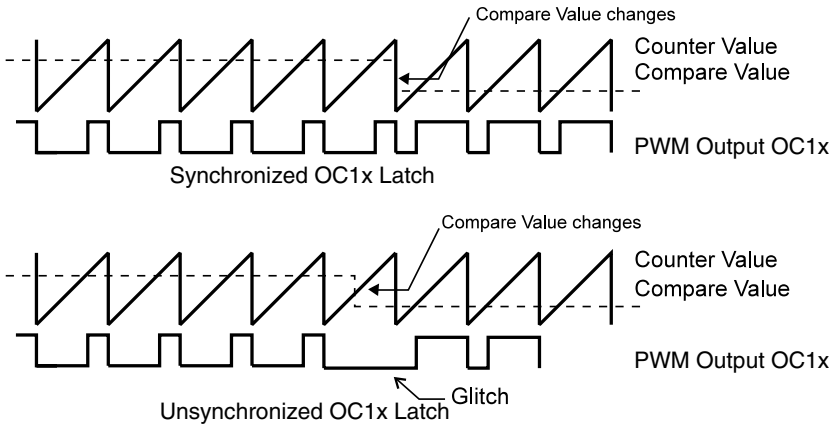
Note that in the PWM mode, the 8, 9, or 10 least significant OCR1A/OCR1B bits (depending on resolution), when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 35 and Figure 36 for an example in each mode.

Figure 35. Effects of Unsynchronized OCR1 Latching.



Note: x = A or B

Figure 36. Effects of Unsynchronized OCR1 Latching in Overflow Mode.



Note: X = A or B

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

When the OCR1X contains $0000 or TOP, and the up/down PWM mode is selected, the output OC1A/OC1B is updated to low or high on the next Compare Match according to the settings of COM1A1/COM1A0 or COM1B1/COM1B0. This is shown in Table 18. In overflow PWM mode, the output OC1A/OC1B is held low or high only when the Output Compare Register contains TOP.

Table 18. PWM Outputs OCR1X = $0000 or TOP[1]

COM1X1	COM1X0	OCR1X	Output OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

Note: 1. X = A or B

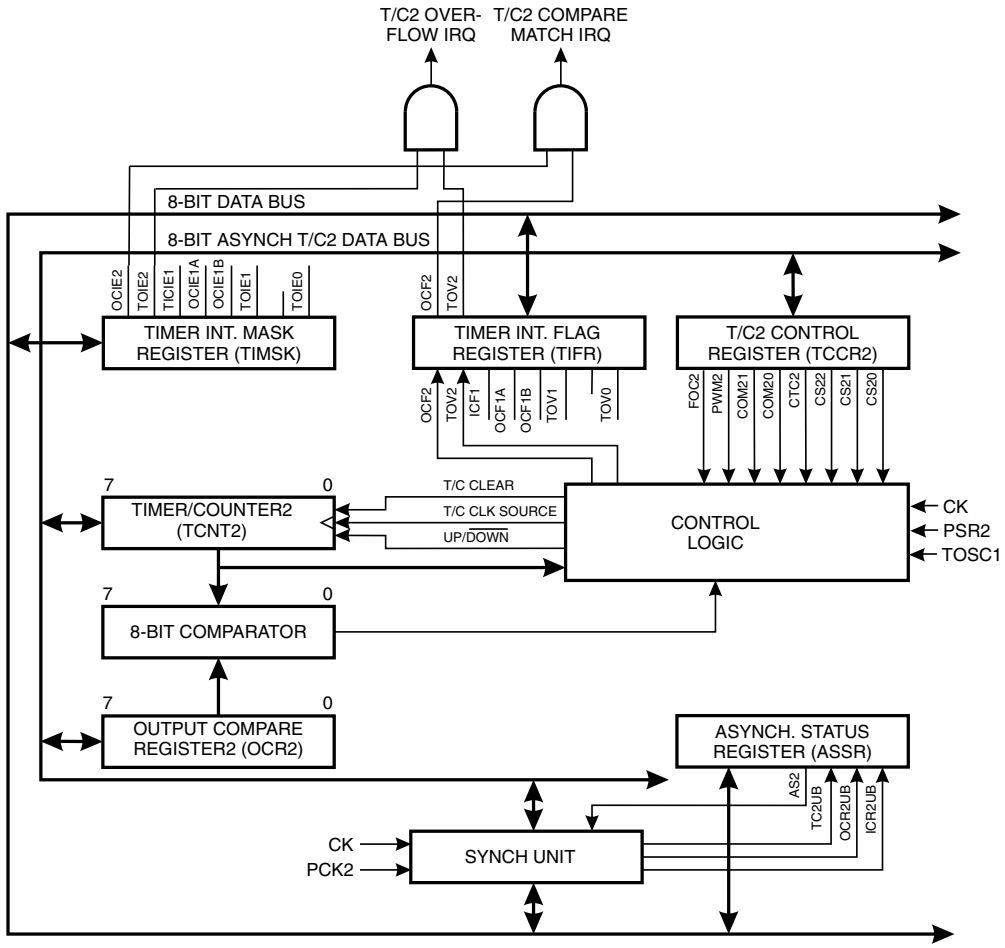
In overflow PWM mode, the table above is only valid for OCR1X = TOP.

In PWM mode, the Timer Overflow Flag1, TOV1, is set when the counter advances from $0000. In overflow PWM mode, the Timer Overflow Flag is set as in Normal Timer/Counter mode. Timer Overflow Interrupt1 operates exactly as in Normal Timer/Counter mode, i.e., it is executed when TOV1 is set provided that Timer Overflow Interrupt1 and global interrupts are enabled. This also applies to the Timer Output Compare1 Flags and interrupts.

8-bit Timer/Counter 2

Figure 37 shows the block diagram for Timer/Counter2.

Figure 37. Timer/Counter2 Block Diagram



The 8-bit Timer/Counter2 can select clock source from CK, prescaled CK, or external crystal input TOSC1. It can also be stopped as described in the section “Timer/Counter2 Control Register – TCCR2” on page 52.

The Status Flags (Overflow and Compare Match) are found in the Timer/Counter Interrupt Flag Register – TIFR. Control signals are found in the Timer/Counter Control Register TCCR2. The interrupt enable/disable settings are found in “The Timer/Counter Interrupt Mask Register – TIMSK” on page 31.

When Timer/Counter2 is externally clocked, the external signal is synchronized with the Oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU

clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

This module features a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make this unit useful for lower speed functions or exact timing functions with infrequent actions.

Timer/Counter2 can also be used as an 8-bit Pulse Width Modulator. In this mode, Timer/Counter2 and the Output Compare Register serve as a glitch-free, stand-alone PWM with centered pulses. Refer to page 57 for a detailed description on this function.

Timer/Counter2 Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
$25 ($45)	FOC2	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	TCCR2
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 – FOC2: Force Output Compare**

Writing a logical one to this bit, forces a change in the Compare Match output pin PD7 (OC2) according to the values already set in COM21 and COM20. If the COM21 and COM20 bits are written in the same cycle as FOC2, the new settings will not take effect until next compare match or Forced Output Compare Match occurs. The Force Output Compare bit can be used to change the output pin without waiting for a Compare Match in the Timer. The automatic action programmed in COM21 and COM20 happens as if a Compare Match had occurred, but no interrupt is generated, and the Timer/Counter will not be cleared even if CTC2 is set. The corresponding I/O pin must be set as an output pin for the FOC2 bit to have effect on the pin. The FOC2 bit will always be read as zero. Setting the FOC2 bit has no effect in PWM mode.

- **Bit 6 – PWM2: Pulse Width Modulator Enable**

When set (one) this bit enables PWM mode for Timer/Counter2. This mode is described on page 43.

- **Bits 5, 4 – COM21, COM20: Compare Output Mode, Bits 1 and 0**

The COM21 and COM20 control bits determine any output pin action following a Compare Match in Timer/Counter2. Output pin actions affect pin PD7(OC2). This is an alternative function to an I/O port, and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 19.

Table 19. Compare Mode Select[1]

COM21	COM20	Description
0	0	Timer/Counter disconnected from output pin OC2
0	1	Toggle the OC2 output line.
1	0	Clear the OC2 output line (to zero).
1	1	Set the OC2 output line (to one).

Note: 1. In PWM mode, these bits have a different function. Refer to Table 21 on page 55 for a detailed description.

- **Bit 3 – CTC2: Clear Timer/Counter on Compare Match**

When the CTC2 control bit is set (one), Timer/Counter2 is Reset to $00 in the CPU clock cycle following a Compare Match. If the control bit is cleared, the Timer/Counter2 continues counting and is unaffected by a Compare Match. When a prescaling of 1 is used, and the Compare Register is set to C, the Timer will count as follows if CTC2 is set:

... | C-1 | C | 0 | 1 |...

When the prescaler is set to divide by eight, the Timer will count like this:

... | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, C, C, C, C, C, C, C | 0, 0, 0, 0, 0, 0, 0, 0 | 1, 1, 1, ...

In PWM mode, this bit has a different function. If the CTC2 bit is cleared in PWM mode, the Timer/Counter acts as an up/down counter. If the CTC2 bit is set (one), the Timer/Counter wraps when it reaches $FF. Refer to page 54 for a detailed description.

- **Bits 2, 1, 0 – CS22, CS21, CS20: Clock Select Bits 2, 1, and 0**

The Clock Select bits 2, 1, and 0 define the prescaling source of Timer/Counter2.

Table 20. Timer/Counter2 Prescale Select

CS22	CS21	CS20	Description
0	0	0	Timer/Counter2 is stopped.
0	0	1	PCK2
0	1	0	PCK2/8
0	1	1	PCK2/32
1	0	0	PCK2/64
1	0	1	PCK2/128
1	1	0	PCK2/256
1	1	1	PCK2/1024

The Stop condition provides a Timer Enable/Disable function. The prescaled modes are scaled directly from the PCK2 clock.

Timer/Counter2 – TCNT2

Bit	7	6	5	4	3	2	1	0	
$24 ($44)	MSB							LSB	TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

This 8-bit register contains the value of Timer/Counter2.

Timer/Counters2 is implemented as an up or up/down (in PWM mode) counter with read and write access. If the Timer/Counter2 is written to and a clock source is selected, it continues counting in the timer clock cycle following the write operation.

Timer/Counter2 Output Compare Register – OCR2

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	MSB							LSB	OCR2
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 is an 8-bit read/write register.

The Timer/Counter Output Compare Register contains the data to be continuously compared with Timer/Counter2. Actions on compare matches are specified in TCCR2. A software write to the Timer/Counter2 Register blocks compare matches in the next Timer/Counter2 clock cycle. This prevents immediate interrupts when initializing the Timer/Counter2.

A Compare Match will set the Compare Interrupt Flag in the CPU clock cycle following the compare event.

Timer/Counter2 in PWM Mode

When PWM mode is selected, the Timer/Counter2 either wraps (overflows) when it reaches $FF or it acts as an up/down counter.

If the up/down mode is selected, the Timer/Counter2 and the Output Compare Register – OCR2 form an 8-bit, free-running, glitch-free, and phase correct PWM with outputs on the PD7(OC2) pin.

If the overflow mode is selected, the Timer/Counter2 and the Output Compare Register – OCR2 form an 8-bit, free-running, and glitch-free PWM, operating with twice the speed of the up/down counting mode.

PWM Modes (Up/Down and Overflow)

The two different PWM modes are selected by the CTC2 bit in the Timer/Counter Control Register – TCCR2.

If CTC2 is cleared and PWM mode is selected, the Timer/Counter acts as an up/down counter, counting up from $00 to $FF, where it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the Output Compare Register, the PD7(OC2) pin is set or cleared according to the settings of the COM21/COM20 bits in the Timer/Counter Control Register TCCR2.

If CTC2 is set and PWM mode is selected, the Timer/Counters will wrap and start counting from $00 after reaching $FF. The PD7(OC2) pin will be set or cleared according to the settings of COM21/COM20 on a Timer/Counter overflow or when the counter value matches the contents of the Output Compare Register. Refer to Table 21 for details.

Table 21. Compare Mode Select in PWM Mode

CTC2	COM21	COM20	Effect on Compare Pin	Frequency
0	0	0	Not connected	
0	0	1	Not connected	
0	1	0	Cleared on Compare Match, up-counting. Set on Compare Match, down-counting (non-inverted PWM).	$f_{TCK0/2}/510$
0	1	1	Cleared on Compare Match, down-counting. Set on Compare Match, up-counting (inverted PWM).	$f_{TCK0/2}/510$
1	0	0	Not connected	
1	0	1	Not connected	
1	1	0	Cleared on compare match, set on overflow.	$f_{TCK0/2}/256$
1	1	1	Set on compare match, cleared on overflow.	$f_{TCK0/2}/256$

Note that in PWM mode, the value to be written to the Output Compare Register is first transferred to a temporary location, and then latched into OCR2 when the Timer/Counter reaches $FF. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR2 write. See Figure 38 for examples.

Figure 38. Effects of Unsynchronized OCR Latching

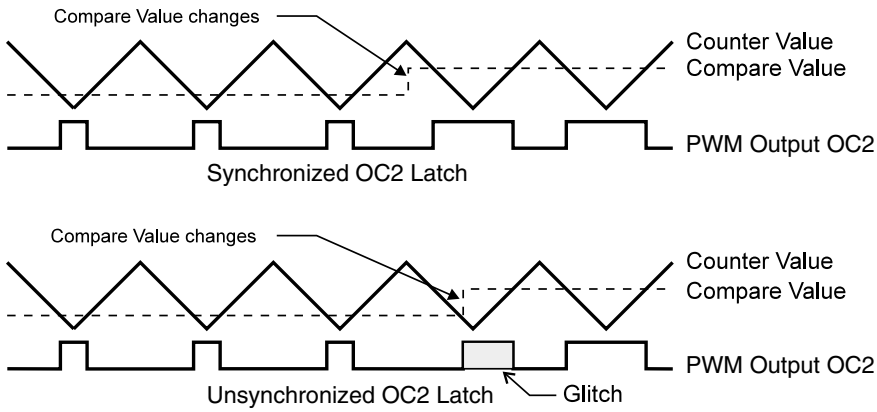
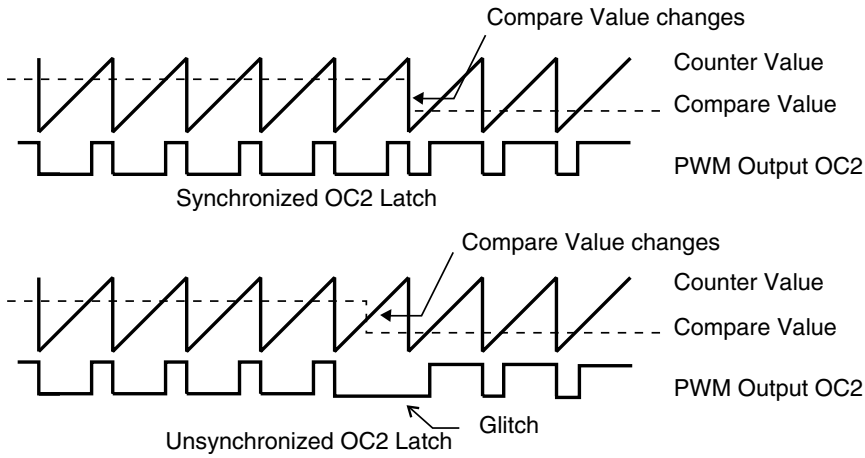


Figure 39. Effects of Unsynchronized OCR Latching in Overflow Mode.



During the time between the write and the latch operation, a read from OCR2 will read the contents of the temporary location. This means that the most recently written value always will read out of OCR2.

When the Output Compare Register contains $00 or $FF, and the up/down PWM mode is selected, the output PD7(OC2) is updated to low or high on the next compare match according to the settings of COM21/COM20. This is shown in Table 22. In overflow PWM mode, the output PD7(OC2) is held low or high only when the Output Compare Register contains $FF.

Table 22. PWM Outputs OCR2 = $00 or $FF

COM21	COM20	OCR2	Output OC2
1	0	$00	L
1	0	$FF	H
1	1	$00	H
1	1	$FF	L

In up/down PWM mode, the Timer Overflow Flag – TOV2, is set when the counter changes direction at $00. In overflow PWM mode, the Timer Overflow Flag is set as in normal Timer/Counter mode. The Timer Overflow Interrupt operates exactly as in normal Timer/Counter mode, i.e., it is executed when TOV2 is set provided that Timer Overflow Interrupt and Global Interrupts are enabled. This also applies to the Timer Output Compare Flag and Interrupt.

The frequency of the PWM will be Timer Clock Frequency divided by 510.

Asynchronous Status Register – ASSR

Bit	7	6	5	4	3	2	1	0	
$22 ($22)	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

- **Bit 3 – AS2: Asynchronous Timer/Counter2**

When AS2 is cleared (zero), Timer/Counter2 is clocked from the internal system clock, CK. When AS2 is set (one), Timer/Counter2 is clocked from the PC6(TOSC1) pin. Pins PC6 and PC7 are connected to a crystal Oscillator and cannot be used as general I/O pins. When the value of this bit is changed, the contents of TCNT2, OCR2, and TCCR2 might be corrupted.

- **Bit 2 – TCN2UB: Timer/Counter2 Update Busy**

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set (one). When TCNT2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

- **Bit 1 – OCR2UB: Output Compare Register2 Update Busy**

When Timer/Counter2 operates asynchronously and OCR2 is written, this bit becomes set (one). When OCR2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical zero in this bit indicates that OCR2 is ready to be updated with a new value.

- **Bit 0 – TCR2UB: Timer/Counter Control Register2 Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2 is written, this bit becomes set (one). When TCCR2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical zero in this bit indicates that TCCR2 is ready to be updated with a new value.

If a write is performed to any of the three Timer/Counter2 registers while its update busy flag is set (one), the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2, and TCCR2 are different. When reading TCNT2, the actual timer value is read. When reading OCR2 or TCCR2, the value in the temporary storage register is read.

Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2, and TCCR2 might be corrupted. A safe procedure for switching clock source is:
  1. Disable the Timer/Counter2 interrupts by clearing OCIE2 and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2, and TCCR2.
  4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
  5. Clear the Timer/Counter2 Interrupt Flags.
  6. Enable interrupts, if needed.
- The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock to the TOSC1 pin may result in incorrect Timer/Counter2 operation. The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2, or TCCR2, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2 write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save mode after having written to TCNT2, OCR2, or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare2 interrupt is used to wake up the device, since the output compare function is disabled during writing to OCR2 or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the OCR2UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save mode, precautions must be taken if the user wants to re-enter Power-save mode: The interrupt logic needs one TOSC1 cycle to be Reset. If the time between wake-up and re-entering Power-save mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether the time before re-entering Power-save is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
  1. Write a value to TCCR2, TCNT2, or OCR2.
  2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
  3. Enter Power-save mode.
- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down mode. After a Power-up Reset or Wake-up from Power-down, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after Power-up or wake-up from Power-down. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down due to unstable clock signal upon startup.

- Description of wake-up from Power-save mode when the Timer is clocked asynchronously: When the interrupt condition is met, the wake-up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four clock cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the Interrupt Flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.
- After waking up from Power-save mode with the asynchronous timer enabled, there will be a short interval in which TCNT2 will read as the same value as before Power-save mode was entered. After an edge on the asynchronous clock, TCNT2 will read correctly (The compare and overflow functions of the Timer are not affected by this behavior.). Safe procedure to ensure that the correct value is read:
  1. Write any value to either of the registers OCR2 or TCCR2.
  2. Wait for the corresponding Update Busy Flag to be cleared.
  3. Read TCNT2.

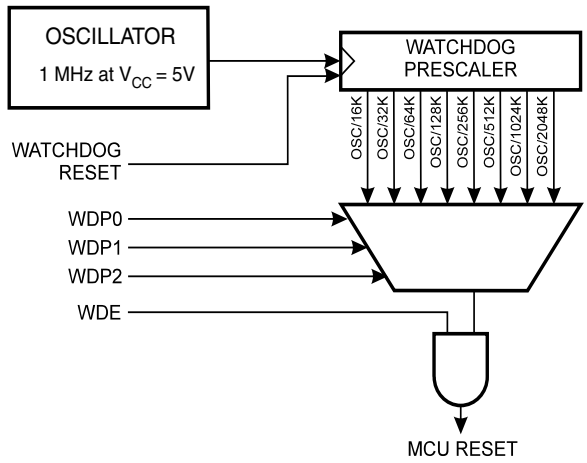
  Note that OCR2 and TCCR2 are never modified by hardware, and will always read correctly.

Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1 MHz. This is the typical value 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 23 on page 61. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATmega163 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 28.

To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

Figure 40. Watchdog Timer



The Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	–	–	–	WDTOE	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 ATmega163 and will always read as zero.

- **Bit 4 – WDTOE: Watchdog Turn-off Enable**

This bit must be set (one) when the WDE bit is cleared. Otherwise, the Watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set(one). To disable an enabled Watchdog Timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDTOE and WDE. A logical 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 logical 0 to WDE. This disables the Watchdog.

- **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 23.

Table 23. 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 cycles	47 ms	15 ms
0	0	1	32K cycles	94 ms	30 ms
0	1	0	64K cycles	0.19 s	60 ms
0	1	1	128K cycles	0.38 s	0.12 s
1	0	0	256K cycles	0.75 s	0.24 s
1	0	1	512K cycles	1.5 s	0.49 s
1	1	0	1,024K cycles	3.0 s	0.97 s
1	1	1	2,048K cycles	6.0 s	1.9 s

EEPROM Read/Write Access

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

The write access time is in the range of 1.9 - 3.8 ms, depending on the V_{CC} voltages. See Table 24 for details. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains code that writes 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. CPU operation under these conditions is likely to cause the Program Counter to perform unintentional jumps and potentially execute the EEPROM write code. To secure EEPROM integrity, the user is advised to use an External under-voltage Reset circuit or the internal BOD in this case.

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	
$1F ($3F)	–	–	–	–	–	–	–	**EEAR8**	**EEARH**
$1E ($3E)	**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 ATmega163 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	
$1D ($3D)	**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	
$1C ($3C)	–	–	–	–	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	0	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and will always read as zero.

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

When the I bit in SREG and EERIE are set (one), the EEPROM Ready Interrupt is enabled. When cleared (zero), the interrupt is disabled. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared (zero).

- **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(one) setting EEWE will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (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 set to write the value into the EEPROM. The EEMWE bit must be set when the 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 2 and 3 is not essential):

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

Caution: An interrupt between step 4 and step 5 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 the four last steps to avoid these problems.

When the write access time (see Table 24) has elapsed, the EEWE bit is cleared (zero) 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 four 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 set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR Register. The EEPROM read access takes one instruction, and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two 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 not possible to set the EERE bit, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 24 lists the typical programming time for EEPROM access from the CPU

Table 24. EEPROM Programming Time.

Symbol	Number of Calibrated RC Oscillator Cycles	Min Programmingn Time	Max Programming Time
EEPROM write (from CPU)	2048	1.9 ms	3.8 ms

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 the 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 for executing instructions is too low.

EEPROM data corruption can easily be avoided by following these design recommendations (one is sufficient):

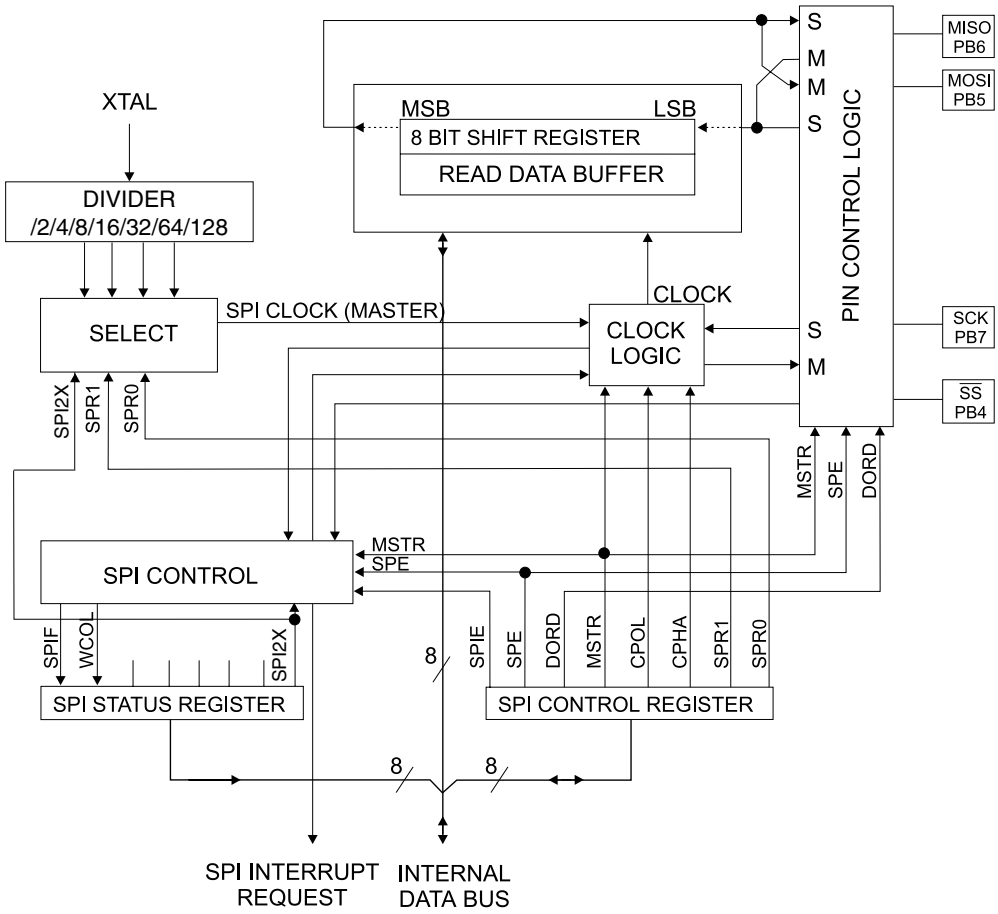
1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} Reset Protection circuit can be used. If a Reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply is voltage is sufficient.
2. Keep the AVR core in Power-down Sleep Mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the EEPROM Registers from unintentional writes.
3. Store constants in Flash memory if the ability to change memory contents from software is not required. Flash memory can not be updated by the CPU unless the boot loader software supports writing to the Flash and the Boot Lock bits are configured so that writing to the Flash memory from CPU is allowed. See “Boot Loader Support” on page 134 for details.

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega163 and peripheral devices or between several AVR devices. The ATmega163 SPI includes the following features:

- **Full-duplex, Three-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**

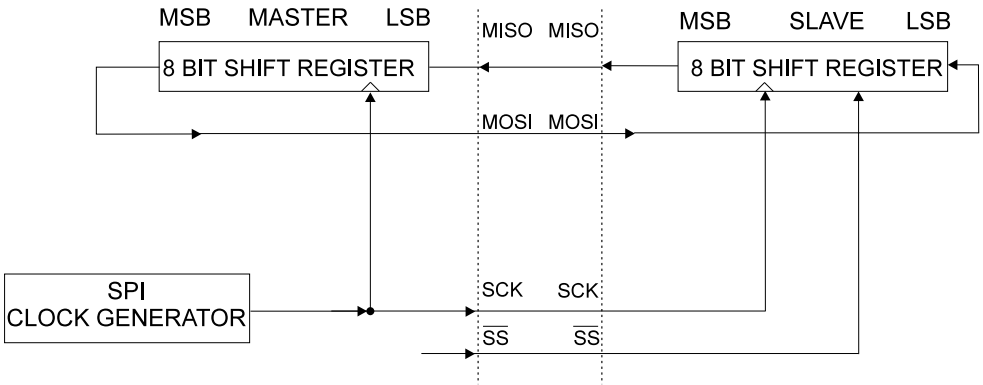
Figure 41. SPI Block Diagram



The interconnection between Master and Slave CPUs with SPI is shown in Figure 42. The PB7(SCK) pin is the clock output in the Master mode and the clock input in the Slave mode. Writing to the SPI Data Register of the Master CPU starts the SPI clock generator, and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the Slave CPU. 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 Slave Select input, PB4($\overline{SS}$), is set low to select an individual Slave SPI device. The two Shift Registers in the Master and the Slave can be considered as one distributed 16-bit circular Shift Register. This is shown in Figure 42. When data is shifted from the Master to the Slave, data is also shifted in

the opposite direction, simultaneously. During one shift cycle, data in the Master and the Slave is interchanged.

Figure 42. 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.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 25.

Table 25. SPI Pin Overrides[1]

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

Note: 1. See “Alternate Functions Of PORTB” on page 118 for a detailed description of how to define the direction of the user defined SPI pins.

$\overline{SS}$ Pin Functionality

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. 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.

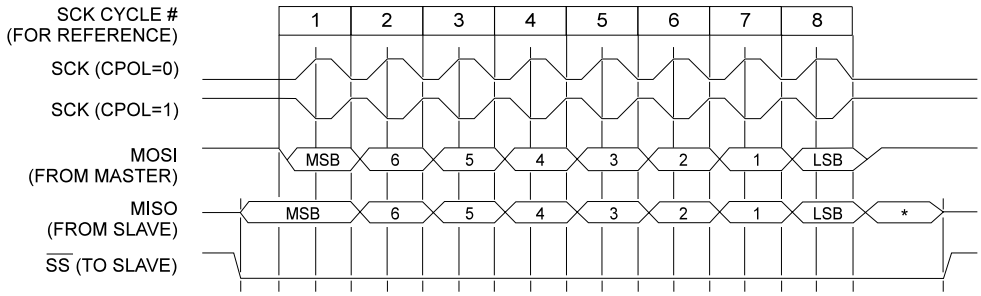
Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master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

When the SPI is configured as a Slave, the $\overline{SS}$ 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. If the $\overline{SS}$ pin is driven high during a transmission, the SPI will stop sending and receiving immediately and both data received and data sent must be considered as lost.

Data Modes

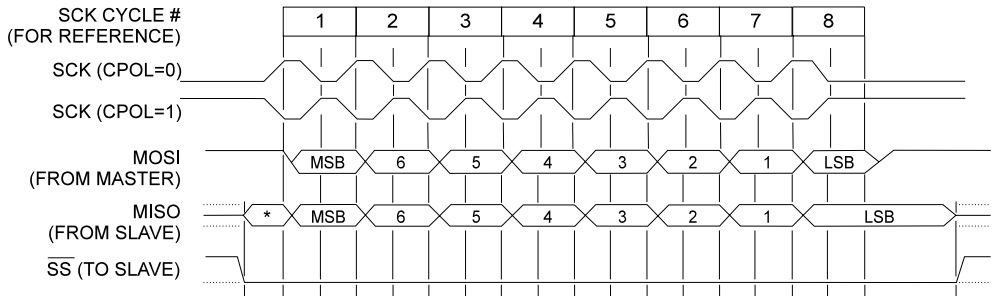
There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 43 and Figure 44.

Figure 43. SPI Transfer Format with CPHA = 0 and DORD = 0



* Not defined but normally MSB of character just received

Figure 44. SPI Transfer Format with CPHA = 1 and DORD = 0



* Not defined but normally LSB of previously transmitted character

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
$0D ($2D)	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 set (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 set (one), the LSB of the data word is transmitted first.

When the DORD bit is cleared (zero), the MSB of the data word is transmitted first.

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

This bit selects Master SPI mode when set (one), and Slave SPI mode when cleared (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 set (one), SCK is high when idle. When CPOL is cleared (zero), SCK is low when idle. Refer to Figure 43 and Figure 44 for additional information.

- **Bit 2 – CPHA: Clock Phase**

Refer to Figure 43 and Figure 44 for the functionality of this bit.

- **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_{ck} is shown in the following table:

Table 26. Relationship Between SCK and the Oscillator Frequency

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

The SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
$0E ($2E)	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 bit is set (one) and an interrupt is generated if SPIE in SPCR is set (one) 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 (one), 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 (zero) by first reading the SPI Status Register with WCOL set (one), and then accessing the SPI Data Register.

- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is set (one) the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 26). 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 $f_{ck}/4$ or lower.

The SPI interface on the ATmega163 is also used for Program memory and EEPROM downloading or uploading. See page 155 for Serial Programming and verification.

The SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
$0F ($2F)	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.

UART

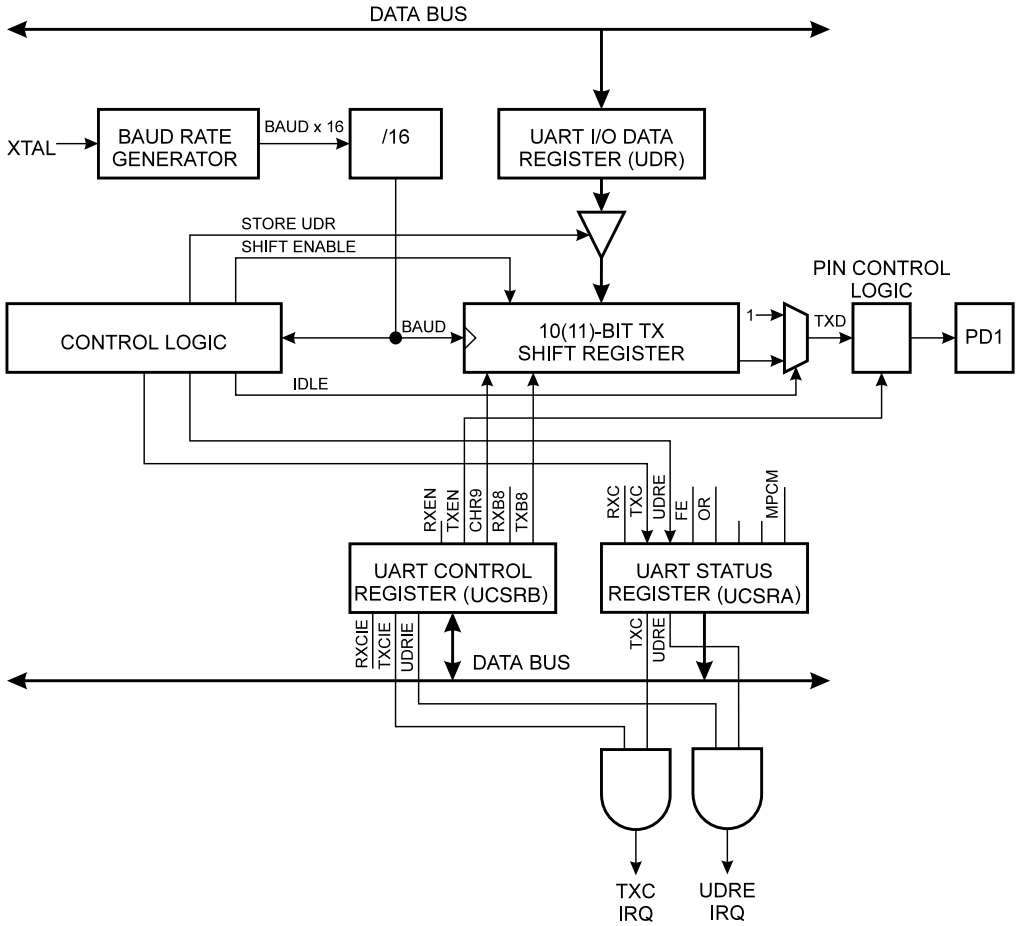
The ATmega163 features a full duplex (separate Receive and Transmit Registers) Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- **Baud Rate Generator Generates any Baud Rate**
- **High Baud Rates at Low XTAL Frequencies**
- **8 or 9 Bits Data**
- **Noise Filtering**
- **OverRun Detection**
- **Framing Error Detection**
- **False Start Bit Detection**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed UART Mode**

Data Transmission

A block schematic of the UART transmitter is shown in Figure 45.

Figure 45. UART Transmitter



Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register, UDR. Data is transferred from UDR to the Transmit Shift Register when:

- A new character has been written to UDR after the stop bit from the previous character has been shifted out. The Shift Register is loaded immediately.
- A new character has been written to UDR before the stop bit from the previous character has been shifted out. The Shift Register is loaded when the stop bit of the character currently being transmitted has been shifted out.

When data is transferred from UDR to the Shift Register, the UDRE (UART Data Register Empty) bit in the UART Status Register, USR, is set. When this bit is set (one), the UART is ready to receive the next character. At the same time as the data is transferred from UDR to the 10(11)-bit Shift Register, bit 0 of the Shift Register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9-bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the TXB8 bit in UCR is transferred to bit 9 in the Transmit Shift Register.

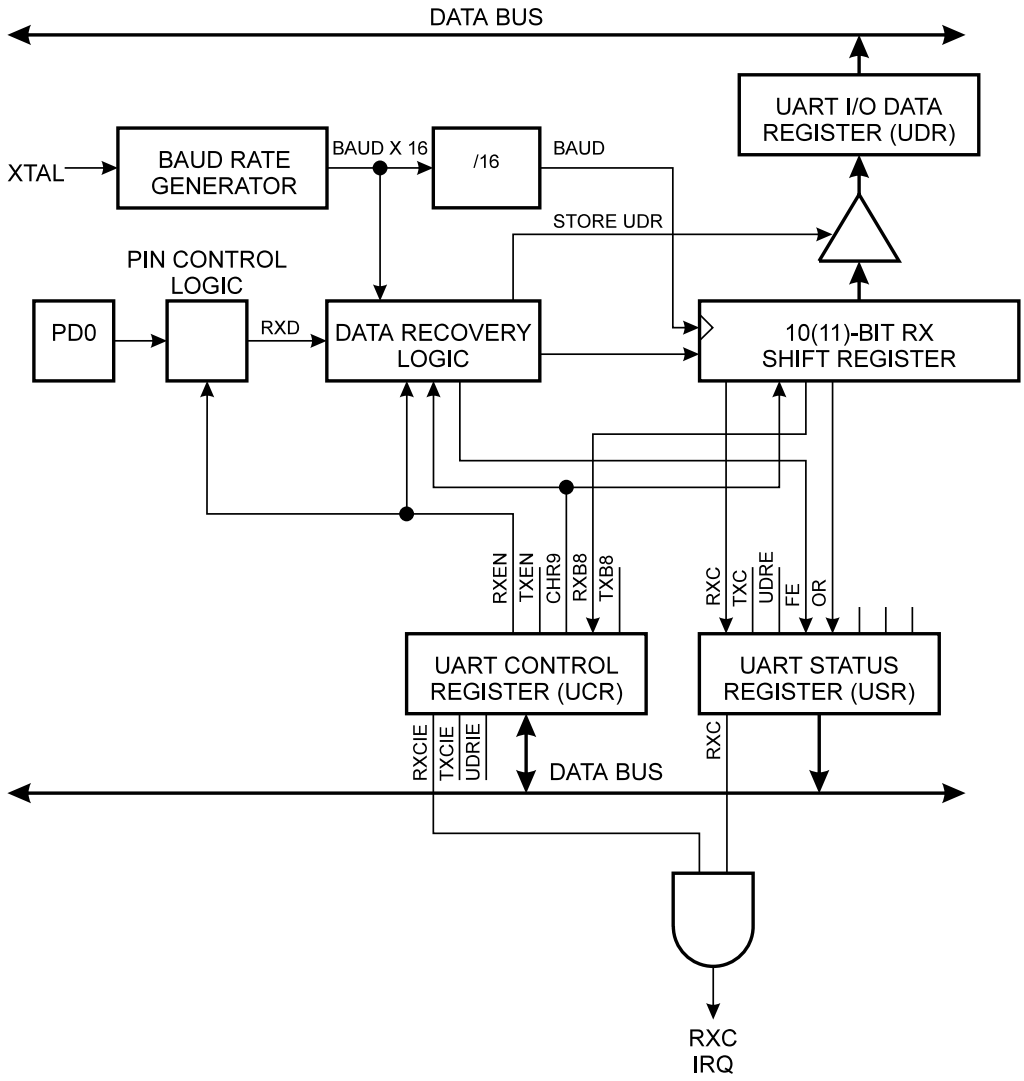
On the Baud Rate clock following the transfer operation to the Shift Register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the Shift Register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR Register to send when the stop bit is shifted out, the UDRE Flag will remain set until UDR is written again. When no new data has been written, and the stop bit has been present on TXD for one bit length, the Transmit Complete Flag, TXC, in USR is set.

The TXEN bit in UCR enables the UART transmitter when set (one). When this bit is cleared (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to PD1, which is forced to be an output pin regardless of the setting of the DDD1 bit in DDRD.

Data Reception

Figure 46 shows a block diagram of the UART Receiver

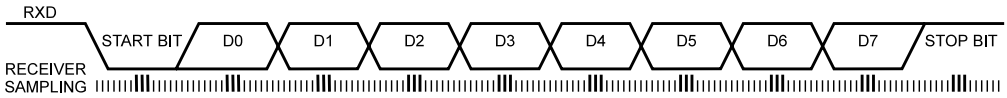
Figure 46. UART Receiver



The Receiver front-end logic samples the signal on the RXD pin at a frequency 16 times the baud rate. While the line is idle, one single sample of logical zero will be interpreted as the falling edge of a start bit, and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1 to 0-transition, the receiver samples the RXD pin at samples 8, 9, and 10. If two or more of these three samples are found to be logical ones, the start bit is rejected as a noise spike and the receiver starts looking for the next 1 to 0-transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 8, 9, and 10. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the Transmitter Shift Register as they are sampled. Sampling of an incoming character is shown in Figure 47. Note that the description above is not valid when the UART transmission speed is doubled. See “Double Speed Transmission” on page 78 for a detailed description.

Figure 47. Sampling Received Data[1]



Note: 1. This figure is not valid when the UART speed is doubled. See “Double Speed Transmission” on page 78 for a detailed description.

When the stop bit enters the Receiver, the majority of the three samples must be one to accept the stop bit. If two or more samples are logical zeros, the Framing Error (FE) Flag in the UART Status Register (USR) is set. Before reading the UDR Register, the user should always check the FE bit to detect Framing Errors.

Whether or not a valid stop bit is detected at the end of a character reception cycle, the data is transferred to UDR and the RXC Flag in USR is set. UDR is in fact two physically separate registers, one for transmitted data and one for received data. When UDR is read, the Receive Data Register is accessed, and when UDR is written, the Transmit Data Register is accessed. If 9-bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the RXB8 bit in UCR is loaded with bit nine in the Transmit Shift Register when data is transferred to UDR.

If, after having received a character, the UDR Register has not been read since the last receive, the OverRun (OR) Flag in UCR is set. This means that the last data byte shifted into to the Shift Register could not be transferred to UDR and has been lost. The OR bit is buffered, and is updated when the valid data byte in UDR is read. Thus, the user should always check the OR bit when reading the UDR Register in order to detect any overruns if the baud rate is high or CPU load is high.

When the RXEN bit in the UCR Register is cleared (zero), the receiver is disabled. This means that the PD0 pin can be used as a general I/O pin. When RXEN is set, the UART Receiver will be connected to PD0, which is forced to be an input pin regardless of the setting of the DDD0 bit in DDRD. When PD0 is forced to input by the UART, the PORTD0 bit can still be used to control the pull-up resistor on the pin.

When the CHR9 bit in the UCR Register is set, transmitted and received characters are 9-bit long plus start and stop bits. The ninth data bit to be transmitted is the TXB8 bit in UCR Register. This bit must be set to the wanted value before a transmission is initated by writing to the UDR Register. The 9th data bit received is the RXB8 bit in the UCR Register.

It is important that the Status Register (USR) always is read before the Data Register (UDR). The Data Register should be read only once for each received byte. Otherwise, the Status Register (USR) might get updated with incorrect values.

Multi-processor Communication Mode

The Multi-Processor Communication mode enables several Slave MCUs to receive data from a Master MCU. This is done by first decoding an address byte to find out which MCU has been addressed. If a particular Slave MCU has been addressed, it will receive the following data bytes as normal, while the other Slave MCUs will ignore the data bytes until another address byte is received.

For an MCU to act as a Master MCU, it should enter 9-bit transmission mode (CHR9 in UCSRB set). The ninth bit must be one to indicate that an address byte is being transmitted, and zero to indicate that a data byte is being transmitted.

For the Slave MCUs, the mechanism appears slightly differently for 8-bit and 9-bit reception mode. In 8-bit reception mode (CHR9 in UCSRB cleared), the stop bit is one for an address byte and zero for a data byte. In 9-bit reception mode (CHR9 in UCSRB

set), the 9th bit is one for an address byte and zero for a data byte, whereas the stop bit is always high.

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 byte, and all slaves receive and read this byte. 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.
4. For each received data byte, the receiving MCU will set the Receive Complete Flag (RXC in UCSRA). In 8-bit mode, the receiving MCU will also generate a Framing Error (FE in UCSRA set), since the stop bit is zero. The other slave MCUs, which still have the MPCM bit set, will ignore the data byte. In this case, the UDR Register and the RXC or FE flags will not be affected.
5. After the last byte has been transferred, the process repeats from step 2.

UART Control

UART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	MSB							LSB	UDR
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 UDR Register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data Register is written. When reading from UDR, the UART Receive Data Register is read.

UART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC	TXC	UDRE	FE	OR	–	U2X	MPCM	UCSRA
Read/Write	r	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXC: UART Receive Complete**

This bit is set (one) when a received character is transferred from the Receiver Shift Register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set(one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

- **Bit 6 – TXC: UART Transmit Complete**

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift Register has been shifted out and no new data has been written to UDR. This Flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCR is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical one to the bit.

- **Bit 5 – UDRE: UART Data Register Empty**

This bit is set (one) when a character written to UDR is transferred to the Transmit Shift Register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIE bit in UCR is set, the UART Transmit Complete interrupt to be executed as long as UDRE is set. UDRE is cleared by writing UDR. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDR in order to clear UDRE, otherwise a new interrupt will occur once the interrupt routine terminates.

UDRE is set (one) during reset to indicate that the transmitter is ready.

- **Bit 4 – FE: Framing Error**

This bit is set if a Framing Error condition is detected, i.e. when the stop bit of an incoming character is zero.

The FE bit is cleared when the stop bit of received data is one.

- **Bit 3 – OR: OverRun**

This bit is set if an Overrun condition is detected, i.e., when a character already present in the UDR Register is not read before the next character has been shifted into the Receiver Shift Register. The OR bit is buffered, which means that it will be set once the valid data still in UDR is read.

The OR bit is cleared (zero) when data is received and transferred to UDR.

- **Bit 2 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and will always read as zero.

- **Bits 1 – U2X: Double the UART Transmission Speed**

Setting this bit will reduce the division of the baud rate generator clock from 16 to 8, effectively doubling the transfer speed at the expense of robustness. For a detailed description, see “Double Speed Transmission” on page 78.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit is used to enter Multi-Processor Communication mode. The bit is set when the slave MCU waits for an address byte to be received. When the MCU has been addressed, the MCU switches off the MPCM bit, and starts data reception.

For a detailed description, see “Multi-processor Communication Mode” on page 73.

UART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	W	
Initial Value	0	0	0	0	0	0	1	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

When this bit is set (one), a setting of the RXC bit in USR will cause the Receive Complete interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

When this bit is set (one), a setting of the TXC bit in USR will cause the Transmit Complete interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 5 – UDRIE: UART Data Register Empty Interrupt Enable**

When this bit is set (one), a setting of the UDRE bit in USR will cause the UART Data Register Empty interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 4 – RXEN: Receiver Enable**

This bit enables the UART Receiver when set (one). When the Receiver is disabled, the RXC, OR, and FE Status Flags cannot become set. If these flags are set, turning off RXEN does not cause them to be cleared.

- **Bit 3 – TXEN: Transmitter Enable**

This bit enables the UART Transmitter when set (one). When disabling the Transmitter while transmitting a character, the Transmitter is not disabled before the character in the Shift Register plus any following character in UDR has been completely transmitted.

- **Bit 2 – CHR9: 9-bit Characters**

When this bit is set (one) transmitted and received characters are 9-bit long plus start and stop bits. The ninth bit is read and written by using the RXB8 and TXB8 bits in UCR, respectively. The ninth data bit can be used as an extra stop bit or a parity bit.

- **Bit 1 – RXB8: Receive Data Bit 8**

When CHR9 is set (one), RXB8 is the ninth data bit of the received character.

- **Bit 0 – TXB8: Transmit Data Bit 8**

When CHR9 is set (one), TXB8 is the ninth data bit in the character to be transmitted.

Baud Rate Generator

The Baud Rate generator is a frequency divider which generates baud-rates according to the following equation:

$$BAUD = \frac{f_{CK}}{16(UBR + 1)}$$

- BAUD = Baud Rate
- f_{CK}= Crystal Clock frequency
- UBR = Contents of the UBRRHI and UBRR Registers, (0 - 4095)

- Note that this equation is not valid when the UART transmission speed is doubled. See “Double Speed Transmission” on page 78 for a detailed description.

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBR settings in Table 27. UBR values which yield an actual baud rate differing less than 2% from the target baud rate, are bold in the table. However, using baud rates that have more than 1% error is not recommended. High error ratings give less noise resistance.

Table 27. UBR Settings at Various Crystal Frequencies

Baud Rate	1 MHz	%Error	1,84 MHz	%Error	2 MHz	%Error	2,458 MHz	%Error
2400	UBR= **25**	**0,2**	UBR= **47**	**0,0**	UBR= **51**	**0,2**	UBR= **63**	**0,0**
4800	UBR= **12**	**0,2**	UBR= **23**	**0,0**	UBR= **25**	**0,2**	UBR= **31**	**0,0**
9600	UBR= 6	7,5	UBR= **11**	**0,0**	UBR= **12**	**0,2**	UBR= **15**	**0,0**
14400	UBR= 3	7,8	UBR= **7**	**0,0**	UBR= 8	3,7	UBR= 10	3,1
19200	UBR= 2	7,8	UBR= **5**	**0,0**	UBR= 6	7,5	UBR= **7**	**0,0**
28800	UBR= 1	7,8	UBR= **3**	**0,0**	UBR= 3	7,8	UBR= 4	6,3
38400	UBR= 1	22,9	UBR= **2**	**0,0**	UBR= 2	7,8	UBR= **3**	**0,0**
57600	UBR= 0	7,8	UBR= **1**	**0,0**	UBR= 1	7,8	UBR= 2	12,5
76800	UBR= 0	22,9	UBR= 1	33,3	UBR= 1	22,9	UBR= **1**	**0,0**
115200	UBR= 0	84,3	UBR= **0**	**0,0**	UBR= 0	7,8	UBR= 0	25,0

Baud Rate	3,28 MHz	%Error	3,69 MHz	%Error	4 MHz	%Error	4,608 MHz	%Error
2400	UBR= **84**	**0,4**	UBR= **95**	**0,0**	UBR= **103**	**0,2**	UBR= **119**	**0,0**
4800	UBR= **42**	**0,8**	UBR= **47**	**0,0**	UBR= **51**	**0,2**	UBR= **59**	**0,0**
9600	UBR= **20**	**1,6**	UBR= **23**	**0,0**	UBR= **25**	**0,2**	UBR= **29**	**0,0**
14400	UBR= **13**	**1,6**	UBR= **15**	**0,0**	UBR= 16	2,1	UBR= **19**	**0,0**
19200	UBR= 10	3,1	UBR= **11**	**0,0**	UBR= **12**	**0,2**	UBR= **14**	**0,0**
28800	UBR= **6**	**1,6**	UBR= **7**	**0,0**	UBR= 8	3,7	UBR= **9**	**0,0**
38400	UBR= 4	6,3	UBR= **5**	**0,0**	UBR= 6	7,5	UBR= 7	6,7
57600	UBR= 3	12,5	UBR= **3**	**0,0**	UBR= 3	7,8	UBR= **4**	**0,0**
76800	UBR= 2	12,5	UBR= **2**	**0,0**	UBR= 2	7,8	UBR= 3	6,7
115200	UBR= 1	12,5	UBR= **1**	**0,0**	UBR= 1	7,8	UBR= 2	20,0

Baud Rate	7,37 MHz	%Error	8 MHz	%Error
2400	UBR= **191**	**0,0**	UBR= **207**	**0,2**
4800	UBR= **95**	**0,0**	UBR= **103**	**0,2**
9600	UBR= **47**	**0,0**	UBR= **51**	**0,2**
14400	UBR= **31**	**0,0**	UBR= **34**	**0,8**
19200	UBR= **23**	**0,0**	UBR= **25**	**0,2**
28800	UBR= **15**	**0,0**	UBR= 16	2,1
38400	UBR= **11**	**0,0**	UBR= **12**	**0,2**
57600	UBR= **7**	**0,0**	UBR= 8	3,7
76800	UBR= **5**	**0,0**	UBR= 6	7,5
115200	UBR= **3**	**0,0**	UBR= 3	7,8

UART Baud Rate Registers – UBRR and UBRRHI

Bit	15	14	13	12	11	10	9	8	
$20 ($40)	–	–	–	–	MSB			LSB	UBRRHI
$09 ($29)	MSB							LSB	UBRR
	7	6	5	4	3	2	1	0	
Read/Write	R	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	

This is a 12-bit register which contains the UART Baud Rate according to the equation on the previous page. The UBRRHI contains the four most significant bits, and the UBRR contains the eight least significant bits of the UART Baud Rate.

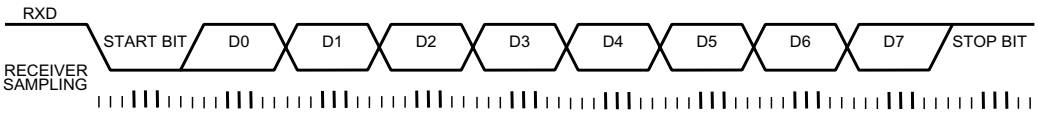
Double Speed Transmission

The ATmega163 provides a separate UART mode which allows the user to double the communication speed. By setting the U2X bit in the UART Control and Status Register UCSRA, the UART speed will be doubled. Note, however, that the receiver will in this case only use half the number of samples (only 8 instead of 16) for data sampling and clock recovery, and therefore requires more accurate baud rate setting and system clock.

The data reception will differ slightly from Normal mode. Since the speed is doubled, the Receiver front-end logic samples the signals on RXD pin at a frequency eight times the baud rate. While the line is idle, one single sample of logical zero will be interpreted as the falling edge of a start bit, and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1 to 0-transition, the Receiver samples the RXD pin at samples 4, 5, and 6. If two or more of these three samples are found to be logical ones, the start bit is rejected as a noise spike and the receiver starts looking for the next 1 to 0-transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 4, 5, and 6. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the Transmitter Shift Register as they are sampled. Sampling of an incoming character is shown in Figure 48.

Figure 48. Sampling Received Data When the Transmission Speed is Doubled



The Baud Rate Generator in Double UART Speed Mode

Note that the baud-rate equation is different from the equation on page 78 when the UART speed is doubled:

$$BAUD = \frac{f_{CK}}{8(UBR + 1)}$$

- BAUD = Baud Rate
- f_{CK}= Crystal Clock frequency
- UBR = Contents of the UBRRHI and UBRR Registers, (0 - 4095)
- Note that this equation is only valid when the UART Transmission Speed is doubled.

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBR settings in Table 28. UBR values which yield an actual baud rate differing less than 1.5% from the target baud rate, are bold in the table. However, since the

number of samples are reduced, and the system clock might have some variance (this applies especially when using resonators), it is recommended that the baud rate error is less than 0.5%.

Table 28. UBR Settings at Various Crystal Frequencies in Double Speed Mode

1.0000 MHz	% Error	**1.8432 MHz**	% Error	**2.0000 MHz**	% Error
UBR = **51**	**0.2**	UBR = **95**	**0.0**	UBR = **103**	**0.2**
UBR = **25**	**0.2**	UBR = **47**	**0.0**	UBR = **51**	**0.2**
UBR = **12**	**0.2**	UBR = **23**	**0.0**	UBR = **25**	**0.2**
UBR = 8	3.7	UBR = **15**	**0.0**	UBR = 16	2.1
UBR = 6	7.5	UBR = **11**	**0.0**	UBR = **12**	**0.2**
UBR = 3	7.8	UBR = **7**	**0.0**	UBR = 8	3.7
UBR = 2	7.8	UBR = **5**	**0.0**	UBR = 6	7.5
UBR = 1	7.8	UBR = **3**	**0.0**	UBR = 3	7.8
UBR = 1	22.9	UBR = **2**	**0.0**	UBR = 2	7.8
UBR = 0	84.3	UBR = **1**	**0.0**	UBR = 1	7.8
-	-	UBR = **0**	**0.0**	-	-

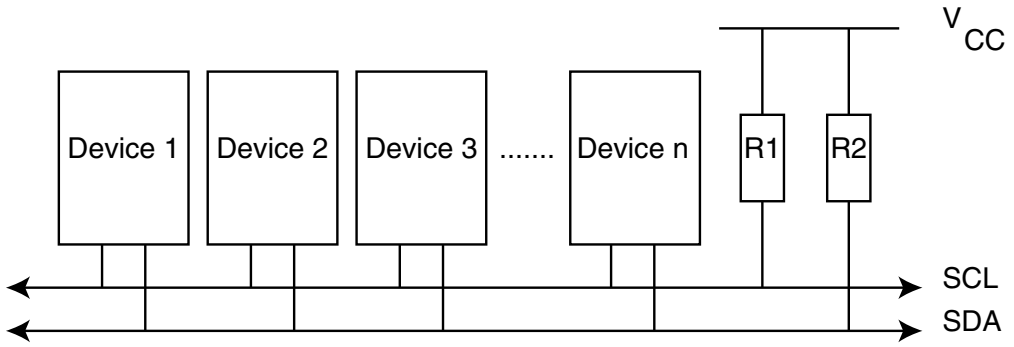
3.2768 MHz	% Error	**3.6864 MHz**	% Error	**4.0000 MHz**	% Error
UBR = **170**	**0.2**	UBR = **191**	**0.0**	UBR = **207**	**0.2**
UBR = **84**	**0.4**	UBR = **95**	**0.0**	UBR = **103**	**0.2**
UBR = **42**	**0.8**	UBR = **47**	**0.0**	UBR = **51**	**0.2**
UBR = 27	1.6	UBR = **31**	**0.0**	UBR = **34**	**0.8**
UBR = 20	1.6	UBR = **23**	**0.0**	UBR = **25**	**0.2**
UBR = 13	1.6	UBR = **15**	**0.0**	UBR = 16	2.1
UBR = 10	3.1	UBR = **11**	**0.0**	UBR = **12**	**0.2**
UBR = 6	1.6	UBR = **7**	**0.0**	UBR = 8	3.7
UBR = 4	6.2	UBR = **5**	**0.0**	UBR = 6	7.5
UBR = 3	12.5	UBR = **3**	**0.0**	UBR = 3	7.8
UBR = 1	12.5	UBR = **1**	**0.0**	UBR = 1	7.8
UBR = 0	12.5	UBR = **0**	**0.0**	UBR = 0	7.8

7.3728 MHz	% Error	**8.0000 MHz**	% Error
UBR = **383**	**0.0**	UBR = **416**	**0.1**
UBR = **191**	**0.0**	UBR = **207**	**0.2**
UBR = **95**	**0.0**	UBR = **103**	**0.2**
UBR = **63**	**0.0**	UBR = **68**	**0.6**
UBR = **47**	**0.0**	UBR = **51**	**0.2**
UBR = **31**	**0.0**	UBR = **34**	**0.8**
UBR = **23**	**0.0**	UBR = **25**	**0.2**
UBR = 15	**0.0**	UBR = 16	2.1
UBR = **11**	**0.0**	UBR = 12	**0.2**
UBR = **7**	**0.0**	UBR = 8	3.7
UBR = **3**	**0.0**	UBR = 3	7.8
UBR = **1**	**0.0**	UBR = 1	7.8
UBR = **0**	**0.0**	UBR = 0	7.8

Two-wire Serial Interface (Byte Oriented)

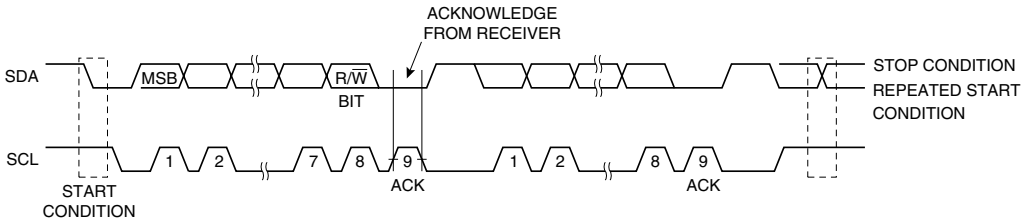
The Two-wire Serial Interface supports bi-directional serial communication. It is designed primarily for simple but efficient integrated circuit (IC) control. The system is comprised of two lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them. Various communication configurations can be designed using this bus. Figure 49 shows a typical Two-wire Serial Bus configuration. Any device connected to the bus can be master or slave. Note that all AVR devices connected to the bus must be powered to allow any bus operation.

Figure 49. Two-wire Serial Bus Configuration



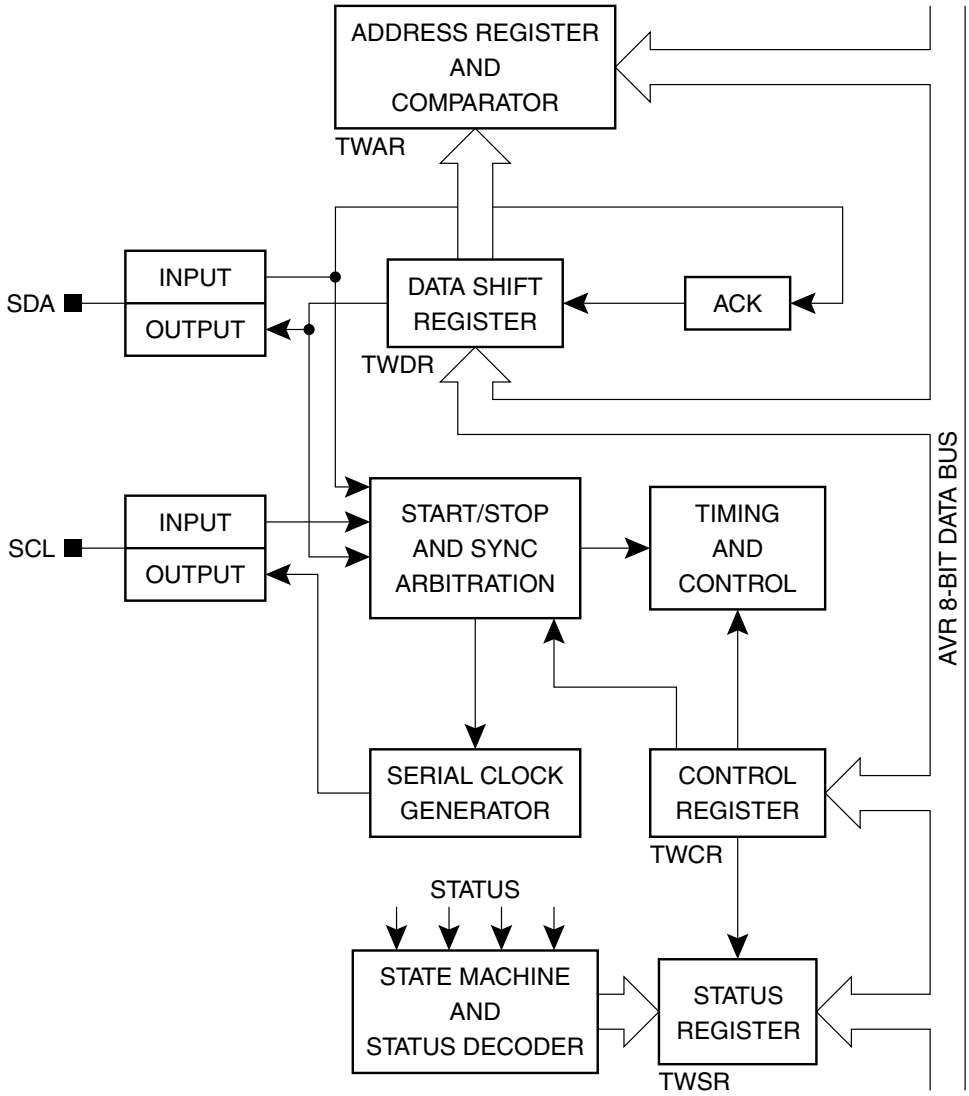
The Two-wire Serial Interface supports Master/Slave and Transmitter/Receiver operation at up to 400 kHz bus clock rate. The Two-wire Serial Interface has hardware support for 7-bit addressing, but is easily extended to, e.g., a 10-bit addressing format in software. When the Two-wire Serial Interface is enabled (TWEN in TWCR is set), a glitch filter is enabled for the input signals from the pins PC0 (SCL) and PC1 (SDA), and the output from these pins is slew-rate controlled. The Two-wire Serial Interface is byte oriented. The operation of the Two-wire Serial Bus is shown as a pulse diagram in Figure 50, including the START and STOP conditions and generation of ACK signal by the bus receiver.

Figure 50. Two-wire Serial Bus Timing Diagram



The block diagram of the Two-wire Serial Interface is shown in Figure 51.

Figure 51. Block Diagram of the Two-wire Serial Interface



The CPU interfaces with the Two-wire Serial Interface via the following five I/O Registers: the Two-wire Serial Interface Bit Rate Register (TWBR), the Two-wire Serial Interface Control Register (TWCR), the Two-wire Serial Interface Status Register (TWSR), the Two-wire Serial Interface Data Register (TWDR), and the Two-wire Serial Interface Address Register (TWAR, used in Slave mode).

The Two-wire Serial Interface Bit Rate Register – TWBR

Bit	7	6	5	4	3	2	1	0	
$00 ($20)	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
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 – Two-wire Serial Interface Bit Rate Register**

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes according to the following equation:

$$\text{Bit Rate} = \frac{f_{CK}}{16 + 2(\text{TWBR}) + t_A f_{CK}}$$

- Bit Rate = SCL frequency
- f_{CK} = CPU Clock frequency
- TWBR = Contents of the Two-wire Serial Interface Bit Rate Register
- t_A = Bus alignment adjustion

Note: Both the Receiver and the Transmitter can stretch the low period of the SCL line when waiting for user response, thereby reducing the average bit rate.

TWBR should be set to a value higher than seven to ensure correct Two-wire Serial Bus functionality. The bus alignment adjustion is automatically inserted by the Two-wire Serial Interface, and ensures the validity of setup and hold times on the bus for any TWBR value higher than seven. This adjustment may vary from 200 ns to 600 ns depending on bus loads and drive capabilities of the devices connected to the bus.

The Two-wire Serial Interface Control Register – TWCR

Bit	7	6	5	4	3	2	1	0	
$36 ($56)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TWINT: Two-wire Serial Interface Interrupt Flag**

This bit is set by hardware when the Two-wire Serial Interface has finished its current job and expects application software response. If the I-bit in the SREG and TWIE in the TWCR Register are set (one), the MCU will jump to the Interrupt Vector at address $22. While the TWINT Flag is set, the bus SCL clock line low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automaticaly cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the Two-wire Serial Interface, so all accesses to the Two-wire Serial Interface Address Register – TWAR, Two-wire Serial Interface Status Register – TWSR, and Two-wire Serial Interface Data Register – TWDR must be complete before clearing this flag.

- **Bit 6 – TWEA: Two-wire Serial Interface Enable Acknowledge Flag**

TWEA Flag controls the generation of the acknowledge pulse. If the TWEA bit is set, the ACK pulse is generated on the Two-wire Serial Bus if the following conditions are met:

1. The device's own slave address has been received.
2. A general call has been received, while the TWGCE bit in the TWAR is set.
3. A data byte has been received in Master Receiver or Slave Receiver mode.

By setting the TWEA bit low, the device can be virtually disconnected from the Two-wire Serial Bus temporarily. Address recognition can then be resumed by setting the TWEA bit again.

- **Bit 5 – TWSTA: Two-wire Serial Bus START Condition Flag**

The TWSTA Flag is set by the application when it desires to become a Master on the Two-wire Serial Bus. The Two-wire Serial Interface hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the Two-wire Serial Interface waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status.

- **Bit 4 – TWSTO: Two-wire Serial Bus STOP Condition Flag**

TWSTO is a Stop Condition Flag. In Master mode setting the TWSTO bit in the Control Register will generate a STOP condition on the Two-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode setting the TWSTO bit can be used to recover from an error condition. No stop condition is generated on the bus then, but the Two-wire Serial Interface returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC: Two-wire Serial Bus Write Collision Flag**

The TWWC bit is set when attempting to write to the Two-wire Serial Interface Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: Two-wire Serial Interface Enable Bit**

The TWEN bit enables Two-wire Serial Interface operation. If this bit is cleared (zero), the bus outputs SDA and SCL are set to high impedance state, and the input signals are ignored. The interface is activated by setting this bit (one).

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and will always read as zero.

- **Bit 0 – TWIE: Two-wire Serial Interface Interrupt Enable**

When this bit is enabled, and the I-bit in SREG is set, the Two-wire Serial Interface interrupt will be activated for as long as the TWINT Flag is high.

The TWCR is used to control the operation of the Two-wire Serial Interface. It is used to enable the Two-wire Serial Interface, to initiate a Master access by applying a START condition to the bus, to generate a receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

The Two-wire Serial Interface Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
$01 ($21)	TWS7	TWS6	TWS5	TWS4	TWS3	–	–	–	TWSR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7..3 – TWS: Two-wire Serial Interface Status**

These five bits reflect the status of the Two-wire Serial Interface logic and the Two-wire Serial Bus.

- **Bits 2..0 – Res: Reserved bits**

These bits are reserved in ATmega163 and will always read as zero

The TWSR is read only. It contains a status code which reflects the status of the Two-wire Serial Interface logic and the Two-wire Serial Bus. There are 26 possible status codes. When TWSR contains $F8, no relevant state information is available and no Two-wire Serial Interface interrupt is requested. A valid status code is available in TWSR one CPU clock cycle after the Two-wire Serial Interface Interrupt Flag (TWINT) is set by hardware and is valid until one CPU clock cycle after TWINT is cleared by software. Table 32 to Table 36 give the status information for the various modes.

The Two-wire Serial Interface Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	
$03 ($23)	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- **Bits 7..0 – TWD: Two-wire Serial Interface Data Register**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the Two-wire Serial Bus.

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writeable while the Two-wire Serial Interface is not in the process of shifting a byte. This occurs when the Two-wire Serial Interface Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remain stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from ADC Noise Reduction mode, Power-down mode, or Power-save mode by the Two-wire Serial Interface interrupt. For example, in the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK flag is controlled automatically by the Two-wire Serial Interface logic, the CPU cannot access the ACK bit directly.

The Two-wire Serial Interface (Slave) Address Register – TWAR

Bit	7	6	5	4	3	2	1	0	
$02 ($22)	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

- **Bits 7..1 – TWA: Two-wire Serial Interface (Slave) Address Register**

These seven bits constitute the slave address of the Two-wire Serial Bus unit.

- **Bit 0 – TWGCE: Two-wire Serial Interface General Call Recognition Enable Bit**

This bit enables, if set, the recognition of the General Call given over the Two-wire Serial Bus.

The TWAR should be loaded with the 7-bit slave address (in the seven most significant bits of TWAR) to which the Two-wire Serial Interface will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. The LSB of TWAR is used to enable recognition of the general call address ($00). There is an associated address comparator that looks for the slave address (or generall call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

Two-wire Serial Interface Modes

The Two-wire Serial Interface can operate in four different modes:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfer in each mode of operation is shown in Figure 52 to Figure 55. These figures contain the following abbreviations:

S: START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

$\overline{A}$: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

In Figure 52 to Figure 55, circles are used to indicate that the Two-wire Serial Interface Interrupt Flag is set. The numbers in the circles show the status code held in TWSR. At these points, actions must be taken by the application to continue or complete the Two-wire Serial Bus transfer. The Two-wire Serial Bus transfer is suspended until the Two-wire Serial Interface interrupt flag is cleared by software.

The Two-wire Serial Interface Interrupt Flag is not automatically cleared by hardware when executing the interrupt routine. Software has to clear the flag to continue the Two-wire transfer. Also note that the Two-wire Serial Interface starts execution as soon as this bit is cleared, so that all access to TWAR, TWDR, and TWSR must have been completed before clearing this flag.

When the Two-wire Serial Interface Interrupt Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in Table 32 to Table 36.

Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 52). Before Master Transmitter mode can be entered, the TWCR must be initialized as follows:

Table 29. TWCR: Master Transmitter Mode Initialization

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	X	0	0	0	1	0	X

TWEN must be set to enable the Two-wire Serial Interface, TWSTA and TWSTO must be cleared.

The Master Transmitter mode may now be entered by setting the TWSTA bit. The Two-wire Serial Interface logic will then test the Two-wire Serial Bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the Two-wire Serial Interface Interrupt Flag (TWINT) is set by hardware, and the status code in TWSR will be $08. TWDR must then be loaded with the slave address and the data direction bit (SLA+W). Clearing the TWINT bit in software will continue the transfer. The TWINT Flag is cleared by writing a logic one to the flag.

When the slave address and the direction bit have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are $18, $20, or $38. The appropriate action to be taken for each of these status codes is detailed in Table 32. The data must be loaded when TWINT is high only. If not, the access will be discarded, and the Write Collision bit – TWWC will be set in the TWCR Register. This scheme is repeated until the last byte is sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by setting TWSTO, a repeated START condition is generated by setting TWSTA and TWSTO.

After a repeated START condition (state $10) the Two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without loosing control over the bus.

Assembly code illustrating operation of the Master Transmitter mode is given at the end of the TWI section.

Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (see Figure 53). The transfer is initialized as in the Master Transmitter mode. When the START condition has been transmitted, the TWINT Flag is set by hardware. The software must then load TWDR with the 7-bit slave address and the Data Direction bit (SLA+R). The transfer will then continue when the TWINT Flag is cleared by software.

When the slave address and the direction bit have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are $40, $48, or $38. The appropriate action to be taken for each of these status codes is detailed in Table 52. Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received and a STOP condition is transmitted by writing a logic one to the TWSTO bit in the TWCR Register.

After a repeated START condition (state $10), the Two-wire Serial Interface may switch to the Master Transmitter mode by loading TWDR with SLA+W or access a new Slave as Master Receiver or Transmitter.

Assembly code illustrating operation of the Master Receiver mode is given at the end of the TWI section.

Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 54). To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

Table 30. TWAR: Slave Receiver Mode Initialization

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value	Device's Own Slave Address							

The upper seven bits are the address to which the Two-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the Two-wire Serial Interface will respond to the general call address ($00), otherwise it will ignore the general call address.

Table 31. WCR: Slave Receiver Mode Initialization

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	1	0	0	0	1	0	X

TWEN must be set to enable the Two-wire Serial Interface. The TWEA bit must be set to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be cleared.

When TWAR and TWCR have been initialized, the Two-wire Serial Interface waits until it is addressed by its own slave address (or the general call address if enabled) followed by the Data Direction bit which must be "0" (write) for the Two-wire Serial Interface to operate in the Slave Receiver mode. After its own slave address and the write bit have been received, the Two-wire Serial Interface Interrupt Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 34. The Slave Receiver mode may also be entered if arbitration is lost while the Two-wire Serial Interface is in the Master mode (see states $68 and $78).

If the TWEA bit is reset during a transfer, the Two-wire Serial Interface will return a "Not Acknowledge" ("1") to SDA after the next received data byte. While TWEA is Reset, the Two-wire Serial Interface does not respond to its own slave address. However, the Two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the Two-wire Serial Interface from the Two-wire Serial Bus.

In ADC Noise Reduction mode, Power-down mode, and Power-save mode, the clock system to the Two-wire Serial Interface is turned off. If the Slave Receive mode is enabled, the interface can still acknowledge a general call and its own slave address by using the Two-wire Serial Bus clock as a clock source. The part will then wake-up from sleep and the Two-wire Serial Interface will hold the SCL clock wil low during the wake-up and until the TWINT Flag is cleared.

Note that the Two-wire Serial Interface Data Register – TWDR – does not reflect the last byte present on the bus when waking up from these sleep modes.

Assembly code illustrating operation of the Slave Receiver mode is given at the end of the TWI section.

Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 55). The transfer is initialized as in the Slave Receiver mode. When TWAR and TWCR have been initialized, the Two-wire Serial Interface waits until it is addressed by its own slave address (or the general call address if enabled) followed by the Data Direction bit which must be “1” (read) for the Two-wire Serial Interface to operate in the Slave Transmitter mode. After its own slave address and the read bit have been received, the Two-wire Serial Interface Interrupt Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 35. The slave transmitter mode may also be entered if arbitration is lost while the Two-wire Serial Interface is in the Master mode (see state $B0).

If the TWEA bit is reset during a transfer, the Two-wire Serial Interface will transmit the last byte of the transfer and enter state $C0 or state $C8. the Two-wire Serial Interface is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all “1” as serial data. While TWEA is reset, the Two-wire Serial Interface does not respond to its own slave address. However, the Two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the Two-wire Serial Interface from the Two-wire Serial Bus.

Assembly code illustrating operation of the Slave Receiver mode is given at the end of the TWI section.

Miscellaneous States

There are two status codes that do not correspond to a defined Two-wire Serial Interface state, see Table 36.

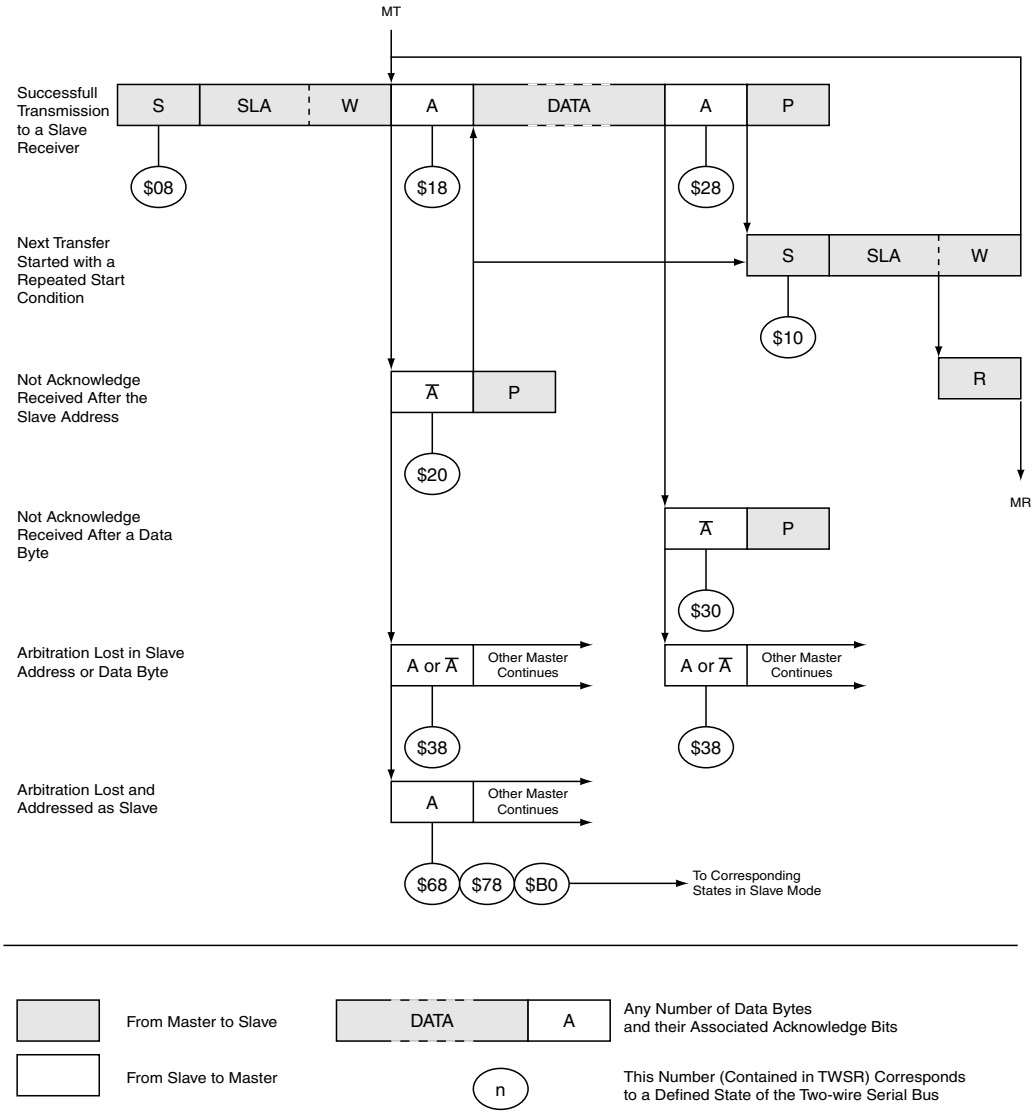
Status $F8 indicates that no relevant information is available because the Two-wire Serial Interface Interrupt Flag (TWINT) is not set yet. This occurs between other states, and when the Two-wire Serial Interface is not involved in a serial transfer.

Status $00 indicates that a bus error has occured during a Two-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the Two-wire Serial Interface to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released and no STOP condition is transmitted.

Table 32. Status Codes for Master Transmitter Mode

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+W	X	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+W or	X	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	X	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
$18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
$28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
$30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
$38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free

Figure 52. Formats and States in the Master Transmitter Mode



Assembly Code Example – Master Transmitter Mode

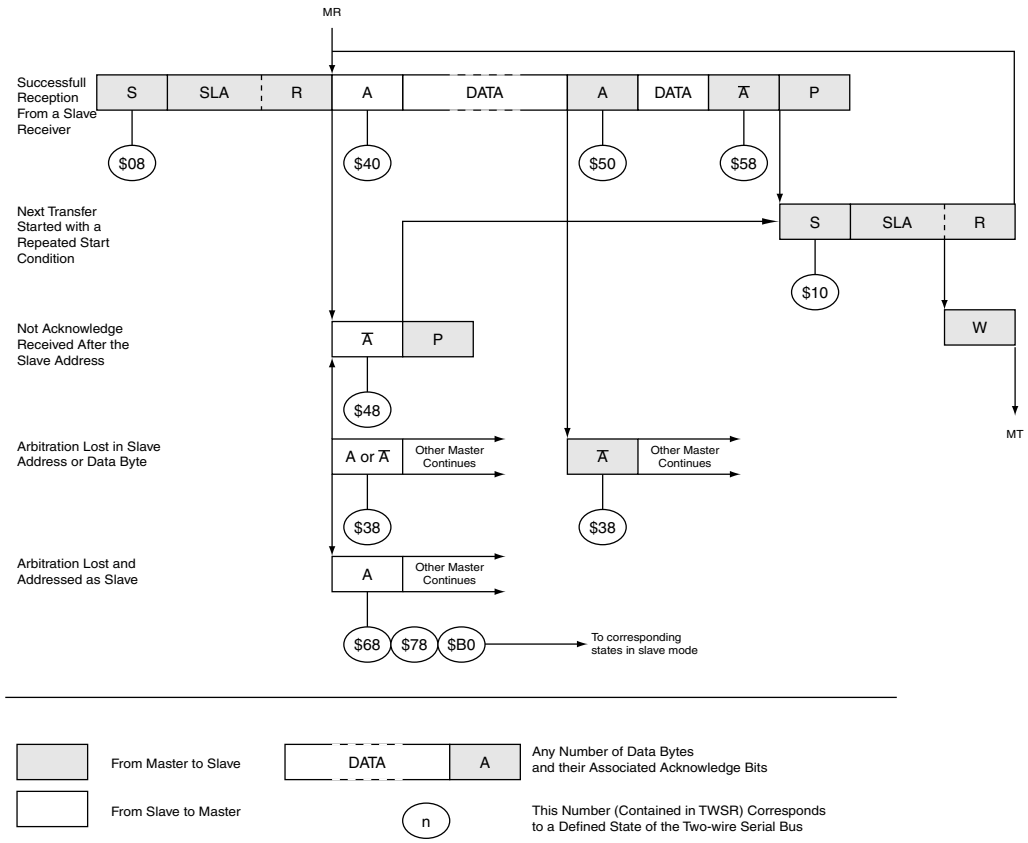
```
;The Slave being addressed has address 0x64. The code examples also assumes
some sort of error handling routine named ERROR.
;Part specific include file and TWI include file must be included.

; <Initialize registers, including TWAR, TWBR and TWCR>

      ldi   r16, (1<<TWSTA) | (1<<TWEN)
      out   TWCR, r16     ; Send START condition

wait1: in    r16,TWCR     ; Wait for TWINT Flag set. This indicates that
      sbrs  r16,TWINT     ; the START condition has been transmitted
      rjmp  wait1

      in    r16, TWSR     ; Check value of TWI Status Register.
      cpi   r16, START    ; If status different from START go to ERROR
```

```
      brne  ERROR

    ldi     r16, 0xc8    ; Load SLA+W into TWDR Register
    out     TWDR, r16
    ldi     r16, (1<<TWINT) | (1<<TWEN)
    out     TWCR, r16    ; Clear TWINT bit in TWCR to start transmission of
address

wait2:in    r16, TWCR    ; Wait for TWINT Flag set. This indicates that
      sbrs    r16, TWINT  ; SLA+W has been transmitted, and ACK/NACK has
      rjmp    wait2       ; been received

      in      r16, TWSR   ; Check value of TWI Status Register. If status
      cpi     r16, MT_SLA_ACK; different from MT_SLA_ACK, go to ERROR
      brne    ERROR

      ldi     r16, 0x33   ; Load data (here, data = 0x33) into TWDR Register
      out     TWDR, r16
      ldi     r16, (1<<TWINT) | (1<<TWEN)
      out     TWCR, r16 ; Clear TWINT bit in TWCR to start transmission of data

wait3:in      r16, TWCR ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT; data has been transmitted, and ACK/NACK has
      rjmp    wait3     ; been received

      in      r16, TWSR ; Check value of TWI Status Register. If status
      cpi     r16, MT_DATA_ACK ; different from MT_DATA_ACK, go to ERROR
      brne    ERROR

      ldi     r16, 0x44 ; Load data (here, data = 0x44) into TWDR Register
      out     TWDR, r16
      ldi     r16, (1<<TWINT) | (1<<TWEN)
      out     TWCR, r16 ; Clear TWINT bit in TWCR to start transmission of data

  ;<send more data bytes if needed>

wait4:in      r16, TWCR ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT; data has been transmitted, and ACK/NACK has
      rjmp    wait4     ; been received

      in      r16, TWSR ; Check value of TWI Status Register. If status
      cpi     r16, MT_DATA_ACK; different from MT_DATA_ACK, go to ERROR
      brne    ERROR

      ldi     r16, (1<<TWINT) | (1<<TWSTO) | (1<<TWEN)
      out     TWCR, r16 ; Transmit STOP condition
```

Table 33. Status Codes for Master Receiver Mode

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+R	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+R or	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	X	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode.
$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR actio	1	0	1	X	A START condition will be transmitted when the bus becomes free
$40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
$48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
$50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
$58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be Reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset

Figure 53. Formats and States in the Master Receiver Mode



Assembly Code Example – Master Receiver Mode

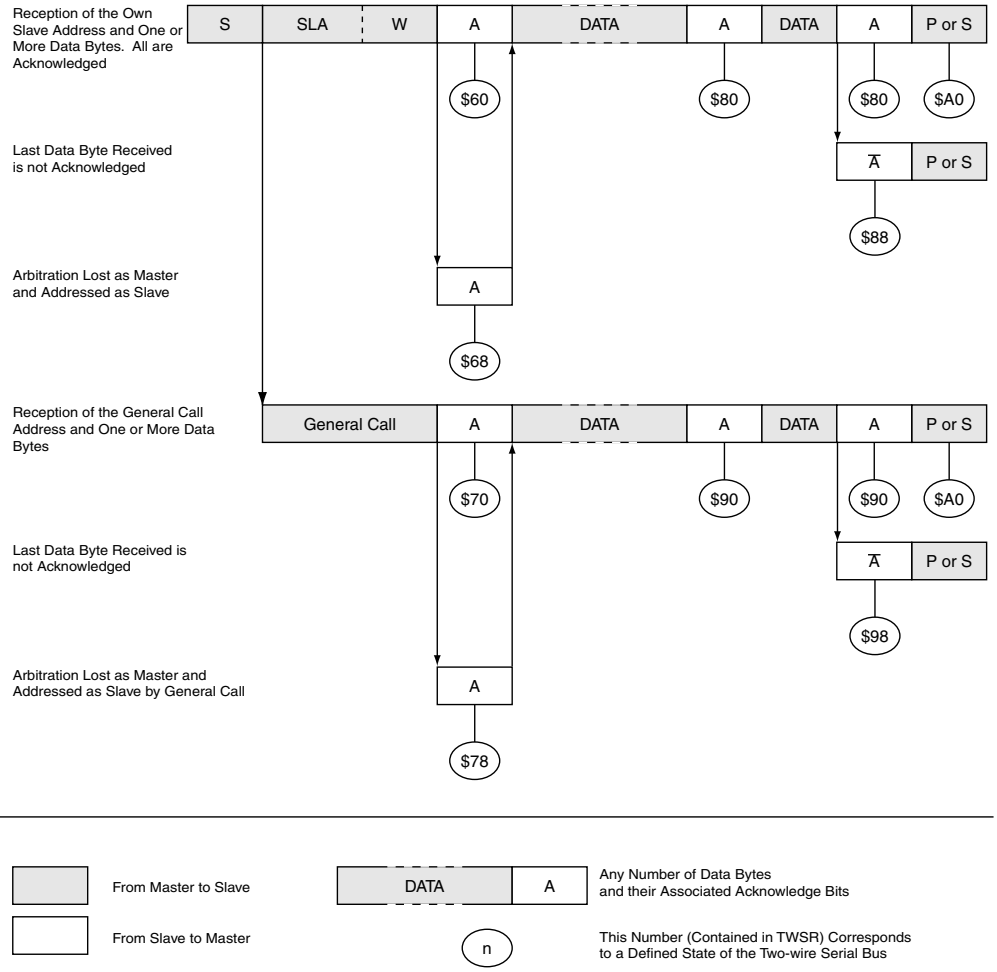
```
;Part specific include file and TWI include file must be included.
; <Initialize registers TWAR and TWBR>

      ldi     r16, (1<<TWINT) | (1<<TWSTA) | (1<<TWEN)
      out     TWCR, r16     ;Send START condition

wait5:in      r16,TWCR      ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT    ; the START condition has been transmitted
      rjmp    wait5

      in      r16, TWSR     ; Check value of TWI Status Register. If status
      cpi     r16, START    ; different from START, go to ERROR
      brne    ERROR

      ldi     r16, 0xc9     ; Load SLA+R into TWDR Register
      out     TWDR, r16
      ldi     r16, (1<<TWINT) | (1<<TWEN)
      out     TWCR, r16     ; Clear TWINT bit in TWCR to start transmission of
                            ; SLA+R

wait6:in    r16,TWCR        ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT    ; SLA+R has been transmitted, and ACK/NACK has
      rjmp    wait6         ; been received

      in      r16, TWSR       ; Check value of TWI Status Register. If status
      cpi     r16, MR_SLA_ACK; different from MR_SLA_ACK, go to ERROR
      brne    ERROR

      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16       ; Clear TWINT bit in TWCR to start reception of
                              ; data.
                              ; Setting TWEA causes ACK to be returned after
                              ; reception of data byte
wait7:in      r16,TWCR        ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT      ; data has been received and ACK returned
      rjmp    wait7

      in      r16, TWSR       ; Check value of TWI Status Register. If status
      cpi     r16, MR_DATA_ACK ; different from MR_DATA_ACK, go to ERROR
      brne    ERROR

      in      r16, TWDR       ; Input received data from TWDR.
      nop                     ;<do something with received data>
      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16       ; Clear TWINT bit in TWCR to start reception of
                              ; data. Setting TWEA causes ACK to be returned
                              ; after reception of data byte
  ;<Receive more data bytes if needed>

  ;receive next to last data byte.
wait8:in      r16,TWCR        ; Wait for TWINT flag set. This indicates that
```

```
        sbrs    r16, TWINT      ; data has been received and ACK returned
        rjmp    wait8

        in      r16, TWSR       ; Check value of TWI Status Register. If status
        cpi     r16, MR_DATA_ACK ; different from MR_DATA_ACK, go to ERROR
        brne    ERROR

        in      r16, TWDR       ; Input received data from TWDR.
      nop                       ;<do something with received data>
        ldi     r16, (1<<TWINT) | (1<<TWEN)
        out     TWCR, r16       ; Clear TWINT bit in TWCR to start reception of
                                ; data. Not setting TWEA causes NACK to be
                                ; returned after reception of next data byte
                                ; receive last data byte. Signal this to slave by
                                ; returning NACK
wait9:in        r16,TWCR        ; Wait for TWINT flag set. This indicates that
        sbrs    r16, TWINT      ; data has been received and NACK returned
        rjmp    wait9

        in      r16, TWSR       ; Check value of TWI Status Register. If status
        cpi     r16, MR_DATA_NACK ; different from MR_DATA_NACK, go to ERROR
        brne    ERROR

        in      r16, TWDR       ; Input received data from TWDR.
        nop                     ;<do something with received data>

        ldi     r16, (1<<TWINT) | (1<<TWSTO) | (1<<TWEN)
        out     TWCR, r16       ; Send STOP signal
```

Table 34. Status Codes for Slave Receiver Mode

Status code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardtware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$68	Arbitration lost in SLA+R/W as master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$78	Arbitration lost in SLA+R/W as master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
$90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
$A0	A STOP condition or repeated START condition has been received while still addressed as slave	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 54. Formats and States in the Slave Receiver Mode



Assembly Code Example – Slave Receiver Mode

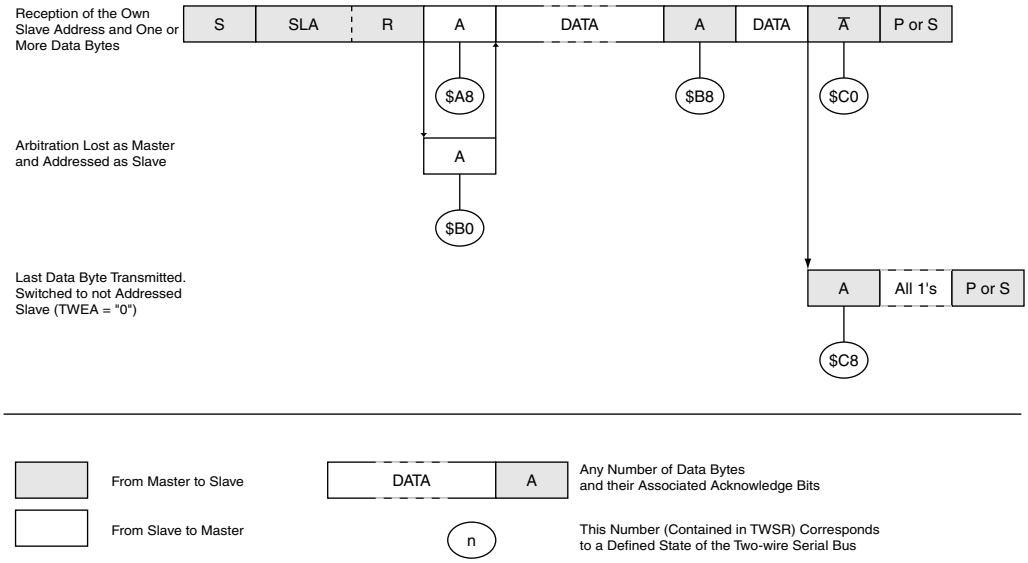
```
;Part specific include file and TWI include file must be included.
; <Initialize registers TWAR and TWBR>

      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16       ; Enable TWI in Slave Receiver Mode

; <Receive START condition and SLA+W>

wait10:in       r16,TWCR      ; Wait for TWINT flag set. This indicates that
      sbrs      r16, TWINT    ; START followed by SLA+W has been received
      rjmp      wait10

        in      r16, TWSR     ; Check value of TWI Status Register. If status
        cpi     r16, SR_SLA_ACK ; different from SR_SLA_ACK, go to ERROR
        brne    ERROR

        ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
        out     TWCR, r16     ; Clear TWINT bit in TWCR to start reception of
                              ; first data byte. Setting TWEA indicates that
```

```
                                ; ACK should be returned after receiving first
                                ; data byte
wait11:in     r16,TWCR          ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT        ; data has been received and ACK returned
      rjmp    wait11

      in      r16, TWSR         ; Check value of TWI Status Register. If status
      cpi     r16, SR_SLA_ACK; different from SR_SLA_ACK, go to ERROR
      brne    ERROR

      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16         ; Clear TWINT bit in TWCR to start reception of
                                ; data. Setting TWEA causes ACK to be returned
                                ; after reception of next data byte
wait12:in       r16,TWCR        ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT        ; data has been received and ACK returned
      rjmp    wait12

      in      r16, TWSR         ; Check value of TWI Status Register. If status
      cpi     r16, SR_DATA_ACK ; different from SR_DATA_ACK, go to ERROR
      brne    ERROR

      in      r16, TWDR         ; Input received data from TWDR.
      nop                       ;<do something with received data>
      ldi     r16, (1<<TWINT) | (1<<TWEN)
      out     TWCR, r16         ; Clear TWINT bit in TWCR to start reception of
                                ; data. Not setting TWEA causes NACK to be
                                ; returned after reception of next data byte
wait13:in     r16,TWCR          ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT        ; data has been received and NACK returned
      rjmp    wait13

      in      r16, TWSR         ; Check value of TWI Status Register. If status
      cpi     r16, SR_DATA_NACK ; different from SR_DATA_NACK, go to ERROR
      brne    ERROR

      in      r16, TWDR           ; Input received data from TWDR.
      nop                         ;<do something with received data>
      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16           ; Clear TWINT bit in TWCR to start reception of
                                  ; data. Setting TWEA causes TWI unit to enter
                                  ; not addressed slave mode with reckognition of
                                  ; own SLA
  ;<Wait for next data transmission or do something else>
```

Table 35. Status Codes for Slave Transmitter Mode

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$A8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B0	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$C0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
$C8	Last data byte in TWDR has been transmitted (TWEA = “0”); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 55. Formats and States in the Slave Transmitter Mode



Assembly Code Example – Slave Transmitter Mode

```
; Part specific include file and TWI include file must be included.
; <Initialize registers, including TWAR, TWBR and TWCR>

      ldi   r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out   TWCR, r16                          ; Enable TWI in Slave
Transmitter Mode

; <Receive START condition and SLA+R>

wait14:in   r16,TWCR       ; Wait for TWINT flag set. This indicates that
      sbrs  r16, TWINT     ; SLA+R has been received, and ACK/NACK has
      rjmp  wait14         ; been returned

      in    r16, TWSR      ; Check value of TWI Status Register. If status
      cpi   r16, ST_SLA_ACK; different from ST_SLA_ACK, go to ERROR
      brne  ERROR

      ldi   r16, 0x33      ; Load data (here, data = 0x33) into TWDR Register
      out   TWDR, r16
      ldi   r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out   TWCR, r16      ; Clear TWINT bit in TWCR to start transmission of
                           ; data. Setting TWEA indicates that ACK should be
                           ; received when transfer finished
; <Send more data bytes if needed>
wait15: in       r16,TWCR  ; Wait for TWINT flag set. This indicates that
      sbrs     r16, TWINT  ; data has been transmitted, and ACK/NACK has
      rjmp     wait15      ; been received

      in       r16, TWSR   ; Check value of TWI Status Register. If status
      cpi      r16, ST_DATA_ACK ; different from ST_DATA_ACK, go to ERROR
      brne     ERROR

      ldi      r16, 0x44   ; Load data (here, data = 0x44) into TWDR Register
      out      TWDR, r16
      ldi      r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out      TWCR, r16   ; Clear TWINT bit in TWCR to start transmission of
                           ; data. Setting TWEA indicates that ACK should be
                           ; received when transfer finished
wait16:in      r16,TWCR    ; Wait for TWINT flag set. This indicates that
      sbrs     r16, TWINT  ; data has been transmitted, and ACK/NACK has
      rjmp     wait16      ; been received

      in       r16, TWSR   ; Check value of TWI Status Register. If status
      cpi      r16, ST_DATA_ACK ; different from ST_DATA_ACK, go to ERROR
      brne     ERROR

      ldi      r16, 0x55   ; Load data (here, data = 0x55) into TWDR Register
      out      TWDR, r16
      ldi      r16, (1<<TWINT) | (1<<TWEN)
      out      TWCR, r16   ; Clear TWINT bit in TWCR to start transmission of
                           ; data. Not setting TWEA indicates that NACK should
```

```
                              ; be received after data byte Master signalling end
                              ; of transmission)
wait17:in       r16,TWCR      ; Wait for TWINT flag set. This indicates that
       sbrs     r16, TWINT    ; data has been transmitted, and ACK/NACK has
       rjmp     wait17        ; been received

       in       r16, TWSR     ; Check value of TWI Status Register. If status
       cpi      r16, ST_LAST_DATA ; different from ST_LAST_DATA, go to ERROR
       brne     ERROR

       ldi      r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
       out      TWCR, r16     ; Continue address reckognition in Slave
Transmitter mode
```

Table 36. Status Codes for Miscellaneous States

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$F8	No relevant state information available; TWINT = “0”	No TWDR action	No TWCR action				Wait or proceed current transfer
$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

TWI Include File

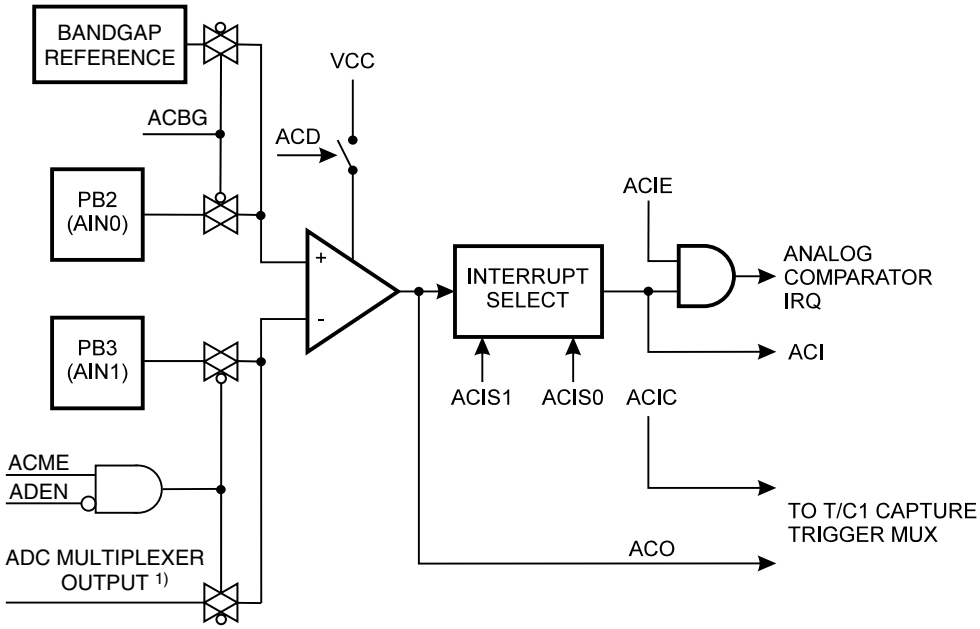
```
;***** General Master staus codes *****
.equ    START               =$08                ;START has been
transmitted
.equ    REP_START           =$10          ;Repeated START has been
transmitted

;***** Master Transmitter staus codes *****
.equ    MT_SLA_ACK      =$18    ;SLA+W has been tramsmitted and ACK received
.equ    MT_SLA_NACK     =$20    ;SLA+W has been tramsmitted and NACK received
.equ    MT_DATA_ACK     =$28    ;Data byte has been tramsmitted and ACK
                                ;received
.equ    MT_DATA_NACK    =$30    ;Data byte has been tramsmitted and NACK
received
.equ    MT_ARB_LOST     =$38    ;Arbitration lost in SLA+W or data bytes

;***** Master Receiver staus codes *****
.equ    MR_ARB_LOST     =$38    ;Arbitration lost in SLA+R or NACK bit
.equ    MR_SLA_ACK      =$40    ;SLA+R has been tramsmitted and ACK received
.equ    MR_SLA_NACK     =$48    ;SLA+R has been tramsmitted and NACK received
.equ    MR_DATA_ACK     =$50    ;Data byte has been received and ACK returned
.equ    MR_DATA_NACK    =$58    ;Data byte has been received and NACK
                                ; tramsmitted

;***** Slave Transmitter staus codes *****
.equ    ST_SLA_ACK      =$A8    ;Own SLA+R has been received and ACK returned
.equ    ST_ARB_LOST_SLA_ACK=$B0;Arbitration lost in SLA+R/W as Master. Own
                                ; SLA+W has been received and ACK returned
.equ    ST_DATA_ACK     =$B8    ;Data byte has been tramsmitted and ACK
                                ;received
```

```
.equ    ST_DATA_NACK    =$C0    ;Data byte has been tramsmitted and NACK
                                ;received
.equ    ST_LAST_DATA    =$C8    ;Last byte in I2DR has been transmitted (TWEA =
                                ;'0'), ACK has been received

;***** Slave Receiver staus codes *****
.equ    SR_SLA_ACK      =$60    ;SLA+R has been received and ACK returned
.equ    SR_ARB_LOST_SLA_ACK=$68;Arbitration lost in SLA+R/W as Master. Own
                                ;SLA+R has been received and ACK returned
.equ    SR_GCALL_ACK    =$70    ;Generall call has been received and ACK
                                ;returned
.equ    SR_ARB_LOST_GCALL_ACK=$78;Arbitration lost in SLA+R/W as Master.
                                ;General Call has been received and ACK
                                ;returned
.equ    SR_DATA_ACK     =$80    ;Previously addressed with own SLA+W. Data byte
                                ;has been received and ACK returned
.equ    SR_DATA_NACK    =$88    ;Previously addressed with own SLA+W. Data byte
                                ;has been received and NACK returned
.equ    SR_GCALL_DATA_ACK=$90;Previously addressed with General Call.Data
                                ;byte has been received and ACK returned
.equ    SR_GCALL_DATA_NACK=$98;Previously addressed with General Call. Data
                                ;byte has been received and NACK returned
.equ    SR_STOP         =$A0    ;A STOP condition or repeated START condition
                                ;has been received while still addressed as a
                                ;slave
;***** Miscellanous States *****
.equ    NO_INFO         =$F8    ;No relevant state information; TWINT = '0'
.equ    BUS_ERROR       =$00    ;Bus error due to illegal START or STOP
                                ;condition
```

The Analog Comparator

The Analog Comparator compares the input values on the positive pin PB2 (AIN0) and negative pin PB3 (AIN1). When the voltage on the positive pin PB2 (AIN0) is higher than the voltage on the negative pin PB3 (AIN1), the Analog Comparator Output, ACO, is set (one). 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 56.

Figure 56. Analog Comparator Block Diagram



Note: 1. See Figure 57 on page 106.

The Analog Comparator Control And Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
$08 ($28)	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 set(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 and the BOD is enabled (BODEN Fuse is programmed), a fixed bandgap voltage of nominally 1.22V replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator.

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

ACO is directly connected to the comparator output.

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

This bit is set (one) 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 (one) and the I-bit in SREG is set (one). 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 set (one) and the I-bit in the Status Register is set (one), the Analog Comparator interrupt is activated. When cleared (zero), the interrupt is disabled.

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

When set (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 cleared (zero), no connection between the Analog Comparator and the Input Capture function is given. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set (one).

- **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 37.

Table 37. 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.

Analog Comparator Multiplexed Input

It is possible to select any of the PA7..0 (ADC7..0) pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in SFIOR) is set (one) and the ADC is switched off (ADEN in ADCSR is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 38. If ACME is cleared (zero) or ADEN is set (one), PB3 (AIN1) is applied to the negative input to the Analog Comparator.

Table 38. Analog Comparator Multiplexed Input

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Analog to Digital Converter

Feature List

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **65 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **Up to 76 kSPS at 8-bit Resolution**
- **Eight Multiplexed Single Ended Input Channels**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - V_{CC} ADC Input Voltage Range**
- **Selectable 2.56V ADC Reference Voltage**
- **Free Run or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

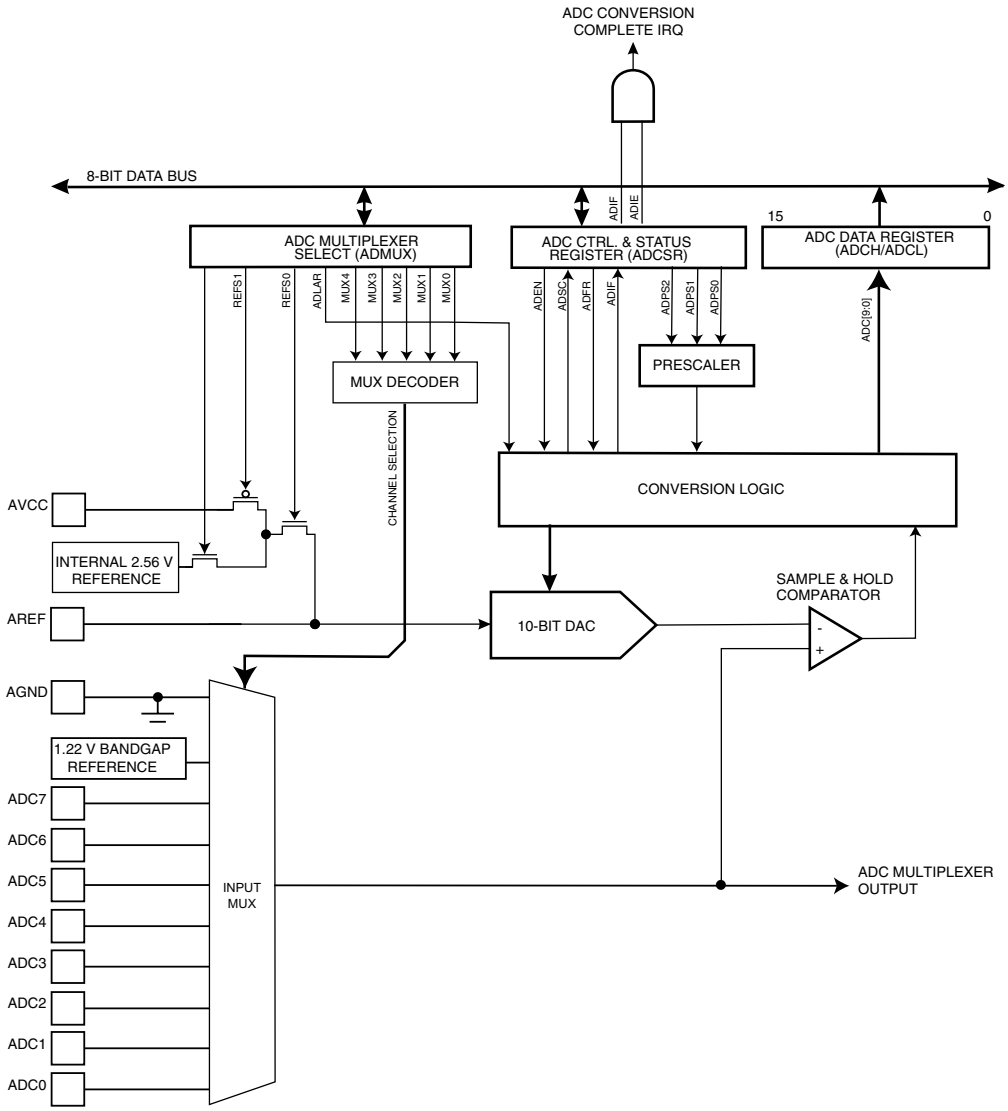
The ATmega163 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows each pin of Port A to be used as input for the ADC.

The ADC contains a Sample and Hold Amplifier which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 57.

The ADC has two separate analog supply voltage pins, AVCC and AGND. AGND must be connected to GND, and the voltage on AVCC must not differ more than ±0.3V from V_{CC}. See the paragraph ADC Noise Canceling Techniques on how to connect these pins.

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The 2.56V reference may be externally decoupled at the AREF pin by a capacitor for better noise perfomance. See “Internal Voltage Reference” on page 29 for a description of the internal voltage reference.

Figure 57. Analog to Digital Converter Block Schematic



Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents AGND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the eight ADC input pins ADC7..0, as well as AGND and a fixed bandgap voltage reference of nominally 1.22V (V_{BG}), can be selected as single ended inputs to the ADC.

The ADC can operate in two modes – Single Conversion and Free Running mode. In Single Conversion mode, each conversion will have to be initiated by the user. In Free Running mode, the ADC is constantly sampling and updating the ADC Data Register. The ADFR bit in ADCSR selects between the two available modes.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSR. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not

consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

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

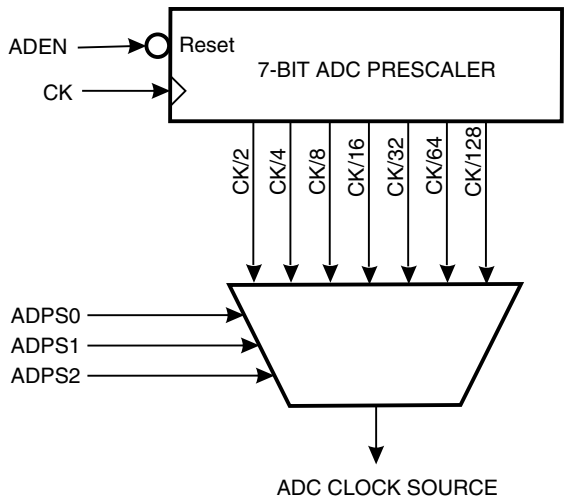
The ADC generates a 10-bit result, which are presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

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

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

Prescaling and Conversion Timing

Figure 58. ADC Prescaler



The successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to achieve maximum resolution. If a lower resolution than 10 bits is required, the input clock frequency to the ADC can be higher than 200 kHz to achieve a higher sampling rate. See “ADC Characteristics” on page 114 for more details. The ADC module contains a prescaler, which divides the system clock to an acceptable ADC clock frequency.

The ADPS bits in ADCSR are used to generate a proper ADC clock input frequency from any XTAL frequency above 100 kHz. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSR. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

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

A normal conversion takes 13 ADC clock cycles. In certain situations, the ADC needs more clock cycles to initalization and minimize offset errors. Extended conversions take 25 ADC clock cycles and occur as the first conversion after the ADC is switched on (ADEN in ADCSR is set). Additionally, when changing voltage reference, the user may improve accuracy by disregarding the first conversion result after the reference or MUX setting was changed.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an extended conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initated on the first rising ADC clock edge. In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. Using Free Running mode and an ADC clock frequency of 200 kHz gives the lowest conversion time with a maximum resolution, 65 µs, equivalent to 15 kSPS. For a summary of conversion times, see Table 39.

Figure 59. ADC Timing Diagram, Extended Conversion (Single Conversion Mode)

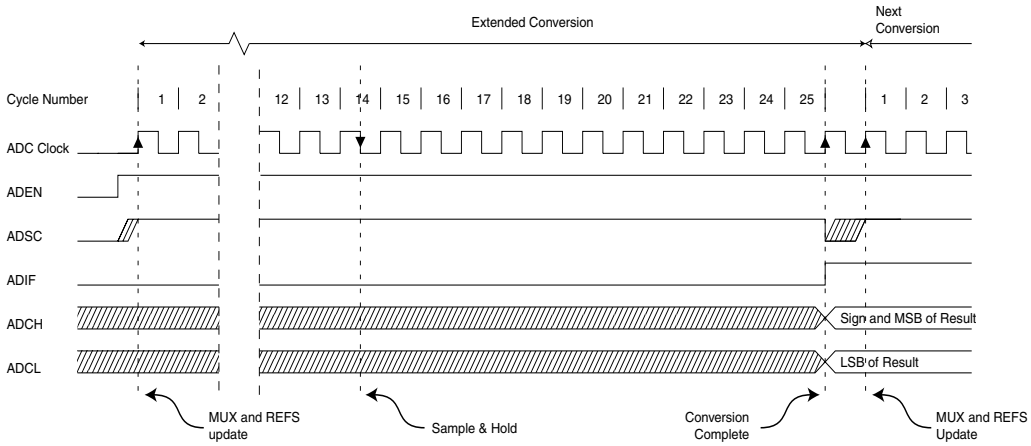


Figure 60. ADC Timing Diagram, Single Conversion

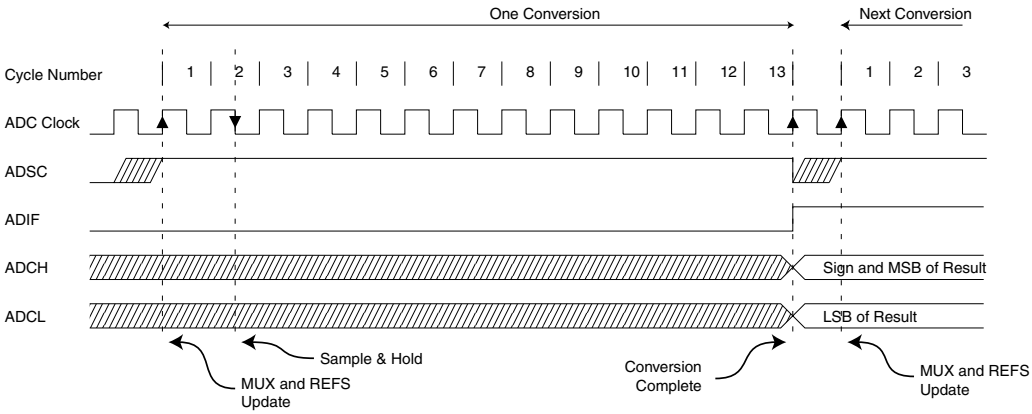


Figure 61. ADC Timing Diagram, Free Run Conversion

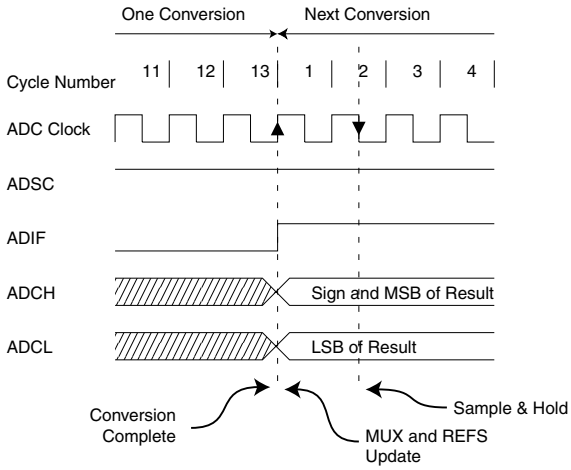


Table 39. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)	Conversion Time (µs)
Extended Conversion	13.5	25	125 - 500
Normal Conversions	1.5	13	65 - 260

ADC Noise Canceler Function

The ADC features a Noise Canceler that enables conversion during ADC Noise Reduction mode (see “Sleep Modes” on page 35) to reduce noise induced from the CPU core and other I/O peripherals. If other I/O peripherals must be active during conversion, this mode works equivalently for Idle mode. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting. Single Conversion Mode must be selected and the ADC conversion complete interrupt must be enabled.

   ADEN = 1

   ADSC = 0

   ADFR = 0

   ADIE = 1
2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine.

The ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
$07 ($27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – REFS1..0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 17. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set). The user should disregard the first conversion result after changing these bits to obtain maximum accuracy. The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 40. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. If ADLAR is cleared, the result is right adjusted. If ADLAR is set, the result is left adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see “The ADC Data Register – ADCL and ADCH” on page 112.

- **Bits 4..0 – MUX4..MUX0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 41 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set).

Table 41. Input Channel Selections

MUX4..0	Single-ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

Table 41. Input Channel Selections (Continued)

MUX4..0	Single-ended Input
01000..11101	Reserved
11110	1.22V (V_{BG})
11111	0V (AGND)

The ADC Control and Status Register – ADCSR

Bit	7	6	5	4	3	2	1	0	
$06 ($26)	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing a logical “1” to this bit enables the ADC. By clearing this bit to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

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

In Single Conversion mode, a logical “1” must be written to this bit to start each conversion. In Free Running mode, a logical “1” must be written to this bit to start the first conversion. The first time ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, an extended conversion will precede the initiated conversion. This extended conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. When a extended conversion precedes a real conversion, ADSC will stay high until the real conversion completes. Writing a 0 to this bit has no effect.

- **Bit 5 – ADFR: ADC Free Running Select**

When this bit is set (one) the ADC operates in Free Running mode. In this mode, the ADC samples and updates the Data Registers continuously. Clearing this bit (zero) will terminate Free Running mode.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set (one) when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set (one). ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSR, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

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

When this bit is set (one) and the I-bit in SREG is set (one), the ADC Conversion Complete Interrupt is activated.

- **Bits 2..0 – ADPS2..0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 42. ADC Prescaler Selections

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

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
$05 ($25)	SIGN	–	–	–	–	–	ADC9	ADC8	ADCH
$04 ($24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
$05 ($25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
$04 ($24)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

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

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

The ADLAR bit in ADMUX affects the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9..0: ADC Conversion result**

These bits represent the result from the conversion. $000 represents analog ground, and $3FF represents the selected reference voltage minus one LSB.

Scanning Multiple Channels

Since change of analog channel always is delayed until a conversion is finished, the Free Running mode can be used to scan multiple channels without interrupting the converter. Typically, the ADC Conversion Complete interrupt will be used to perform the channel shift. However, the user should take the following fact into consideration:

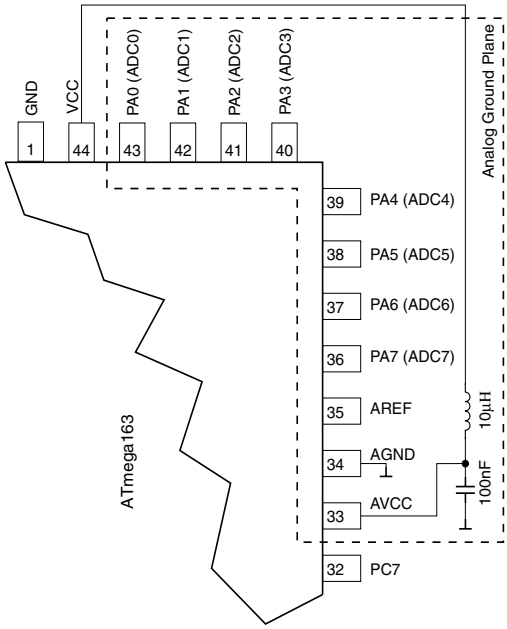
The interrupt triggers once the result is ready to be read. In Free Running mode, the next conversion will start immediately when the interrupt triggers. If ADMUX is changed after the interrupt triggers, the next conversion has already started, and the old setting is used.

ADC Noise Canceling Techniques

Digital circuitry inside and outside the ATmega163 generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. The analog part of the ATmega163 and all analog components in the application should have a separate analog ground plane on the PCB. This ground plane is connected to the digital ground plane via a single point on the PCB.
2. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
3. The AVCC pin on the ATmega163 should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 62.
4. Use the ADC noise canceler function to reduce induced noise from the CPU.
5. If some Port A pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Figure 62. ADC Power Connections



ADC Characteristics

Table 43. ADC Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution	Single-ended Conversion		10		Bits
	Absolute accuracy	V_{REF} = 4V ADC clock = 200 kHz		1	2	LSB
	Absolute accuracy	V_{REF} = 4V ADC clock = 1 MHz		4		LSB
	Absolute accuracy	V_{REF} = 4V ADC clock = 2 MHz		16		LSB
	Integral Non-linearity	V_{REF} > 2V		0.5		LSB
	Differential Non-linearity	V_{REF} > 2V		0.5		LSB
	Zero Error (Offset)	V_{REF} > 2V		1		LSB
	Conversion Time	Free Running Conversion	65		260	µs
	Clock Frequency		50		200	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3(1)		V_{CC} + 0.3(2)	V
V_{REF}	Reference Voltage		2 V		AV_{CC}	V
VINT	Internal Voltage Reference		2.35	2.56	2.77	V
V_{BG}	Bandgap Voltage Reference		1.12	1.22	1.32	V
R_{REF}	Reference Input Resistance		6	10	13	kΩ
V_{IN}	Input Voltage		AGND		AREF	V
R_{AIN}	Analog Input Resistance			100		MΩ

Notes: 1. Minimum for AVCC is 2.7V.
2. Maximum for AVCC is 5.5V.

I/O Ports

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 for changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input).

Port A

Port A is an 8-bit bi-directional I/O port with internal pull-ups.

Three I/O memory address locations are allocated for Port A, one each for the Data Register – PORTA, $1B($3B), Data Direction Register – DDRA, $1A($3A) and the Port A Input Pins – PINA, $19($39). The Port A Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The PORT A output buffers can sink 20 mA and thus drive LED displays directly. 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.

Port A has an alternate function as analog inputs for the ADC. If some Port A pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion.

During Power-down mode, the schmitt trigger of the digital input is disconnected. This allows analog signals that are close to $V_{CC}/2$ to be present during powerdown without causing excessive power consumption.

The Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
$1B ($3B)	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	

The Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
$1A ($3A)	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	

The Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
$19 ($39)	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	

The Port A Input Pins Address – PINA – is not a register, and this address enables access to the physical value on each Port A pin. When reading PORTA the PORTA Data Latch is read, and when reading PINA, the logical values present on the pins are read.

PORT A as General Digital I/O

All 8 bits in PORT A are equal when used as digital I/O pins.

PAn, General I/O pin: The DDAn bit in the DDRA Register selects the direction of this pin, if DDAn is set (one), PAn is configured as an output pin. If DDAn is cleared (zero), PAn is configured as an input pin. If PORTAn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTAn has to be cleared (zero), the pin has to be configured as an output pin, or the

PUD bit has to be set. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 44. DDAn Effects on PORTA Pins[1]

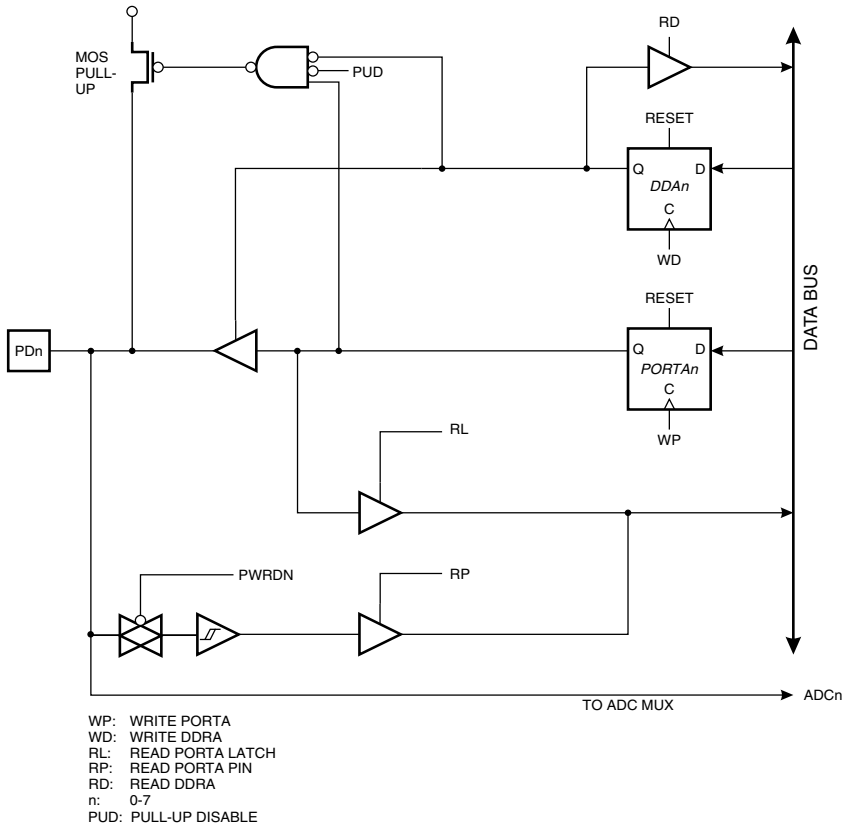
DDAn	PORTAn	PUD	I/O	Pull Up	Comment
0	0	x	Input	No	Tri-state (Hi-Z)
0	1	1	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PAn will source current if ext. pulled low.
1	0	x	Output	No	Push-pull Zero Output
1	1	x	Output	No	Push-pull One Output

Note: 1. n: 7,6…0, pin number.

PORT A Schematics

Note that all port pins are synchronized. The synchronization latches are not shown in the figure.

Figure 63. PORTA Schematic Diagrams (Pins PA0 - PA7)



Port B

Port B is an 8-bit bi-directional I/O port with internal pull-ups.

Three I/O memory address locations are allocated for Port B, one each for the Data Register – PORTB, $18($38), Data Direction Register – DDRB, $17($37) and the Port B Input Pins – PINB, $16($36). The Port B Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port B output buffers can sink 20 mA and thus drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

The Port B pins with alternate functions are shown in Table 45.

Table 45. Port B Pins Alternate Functions

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

When the pins are used for the alternate function, the DDRB and PORTB Registers have to be set according to the alternate function description.

The Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	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	

The Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	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	

The Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	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	

The Port B Input Pins Address – PINB – is not a register, and this address enables access to the physical value on each Port B pin. When reading PORTB, the PORTB Data Latch is read, and when reading PINB, the logical values present on the pins are read.

Port B As General Digital I/O

All eight bits in Port B are equal when used as digital I/O pins. PBn, General I/O pin: The DDBn bit in the DDRB Register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTBn has to be cleared (zero), the pin has to be configured as an output pin, or the PUD bit has to be set. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 46. DDBn Effects on Port B Pins[1]

DDBn	PORTBn	PUD	I/O	Pull Up	Comment
0	0	x	Input	No	Tri-state (Hi-Z)
0	1	1	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PBn will source current if ext. pulled low.
1	0	x	Output	No	Push-pull Zero Output
1	1	x	Output	No	Push-pull One Output

Note: 1. n: 7,6…0, pin number.

Alternate Functions Of PORTB

The alternate pin configuration is as follows:

- **SCK – PORTB, 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 to be an input, the pull-up can still be controlled by the PORTB7 bit. See the description of the SPI port for further details.

- **MISO – PORTB, 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 to be an input, the pull-up can still be controlled by the PORTB6 bit. See the description of the SPI port for further details.

- **MOSI – PORTB, 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 to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further details.

- **$\overline{SS}$ – PORTB, Bit 4**

$\overline{SS}$: Slave Port 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 to be an input, the pull-up can still be controlled by the PORTB4 bit. See the description of the SPI port for further details.

- **AIN1 – PORTB, Bit 3**

AIN1, Analog Comparator Negative input. When configured as an input (DDB3 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB3 is cleared (zero)), this pin also serves as the negative input of the On-chip Analog Comparator. During Power-down mode, the schmitt trigger of the digital input is disconnected. This allows analog signals which are close to $V_{CC}/2$ to be present during Power-down without causing excessive power consumption.

- **AIN0 – PORTB, Bit 2**

AIN0, Analog Comparator Positive input. When configured as an input (DDB2 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB2 is cleared (zero)), this pin also serves as the positive input of the On-chip Analog Comparator. During Power-down mode, the schmitt trigger of the digital input is disconnected. This allows analog signals which are close to $V_{CC}/2$ to be present during Power-down without causing excessive power consumption.

- **T1 – PORTB, Bit 1**

T1, Timer/Counter1 Counter Source. See the Timer description for further details.

- **T0 – PORTB, Bit 0**

T0: Timer/Counter0 Counter Source. See the Timer description for further details.

Port B Schematics

Note that all port pins are synchronized. The synchronization latches are not shown in the figures.

Figure 64. PORTB Schematic Diagram (Pins PB0 and PB1)

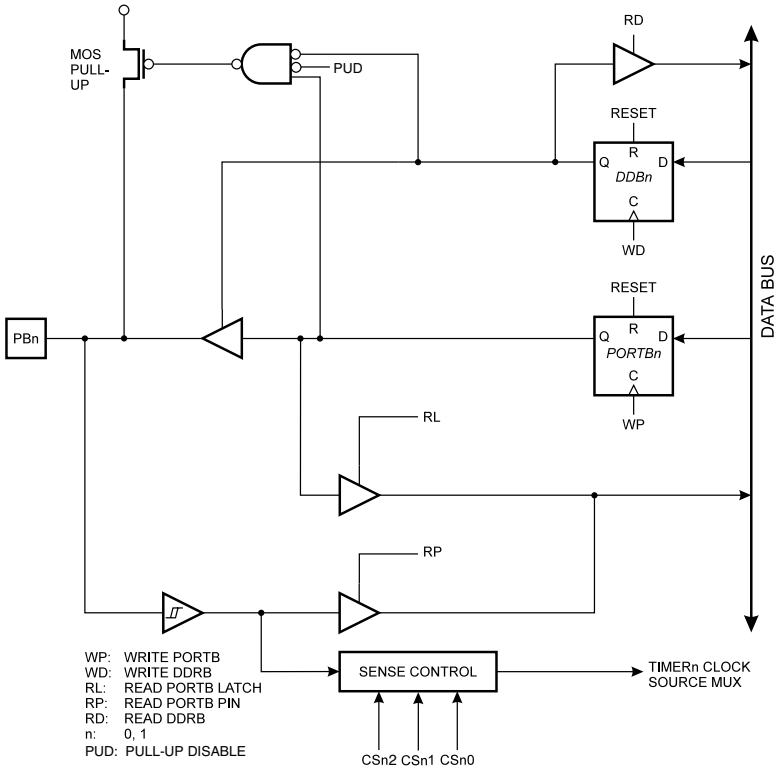


Figure 65. PORTB Schematic Diagram (Pins PB2 and PB3)

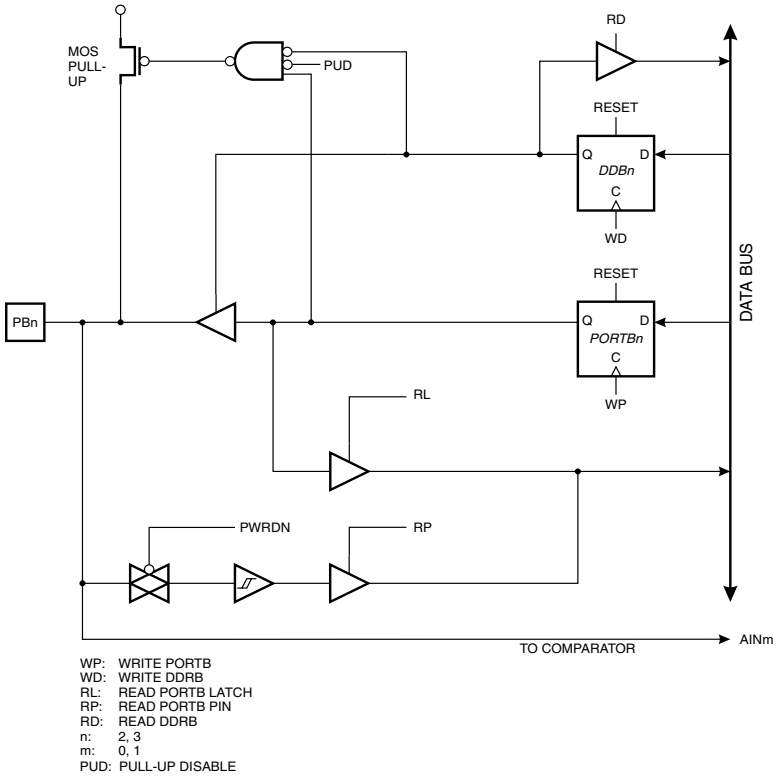


Figure 66. PORTB Schematic Diagram (Pin PB4)

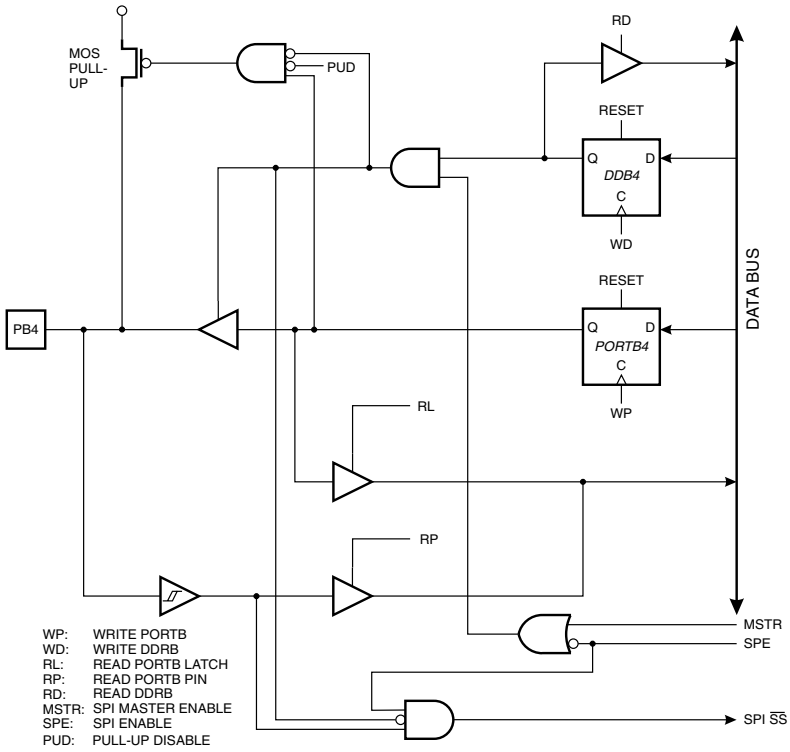


Figure 67. PORTB Schematic Diagram (Pin PB5)

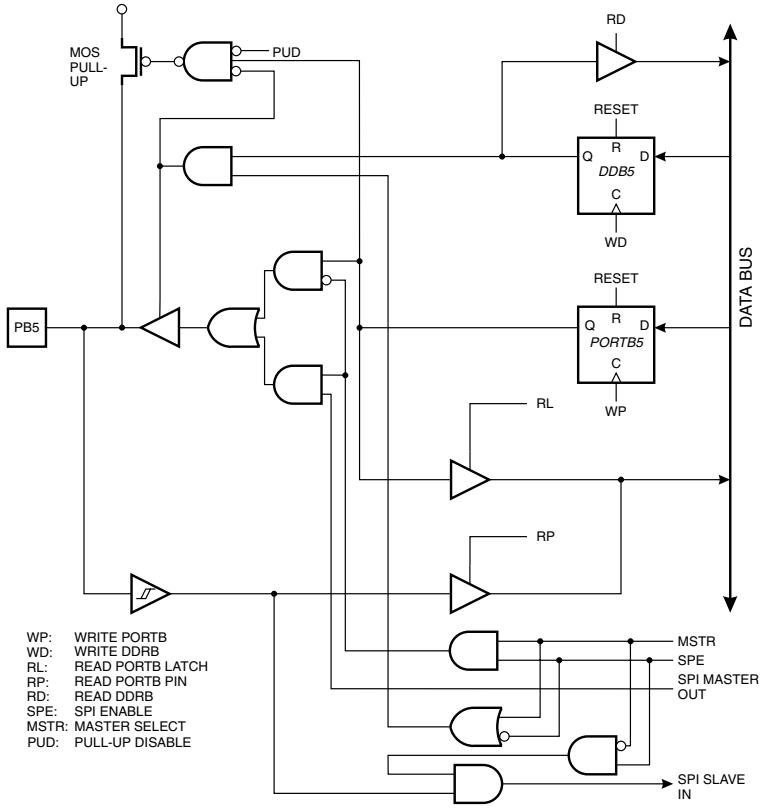


Figure 68. PORTB Schematic Diagram (Pin PB6)

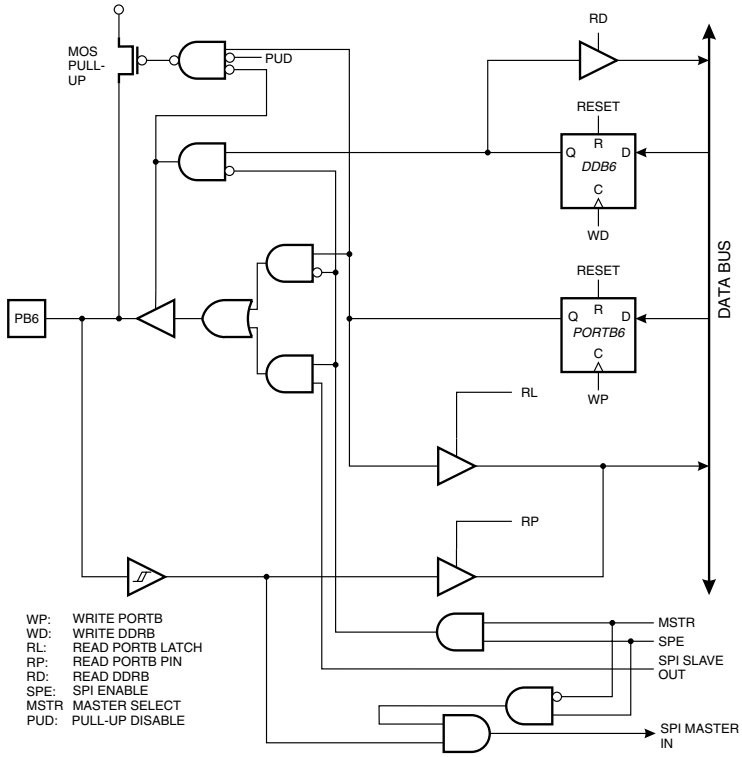
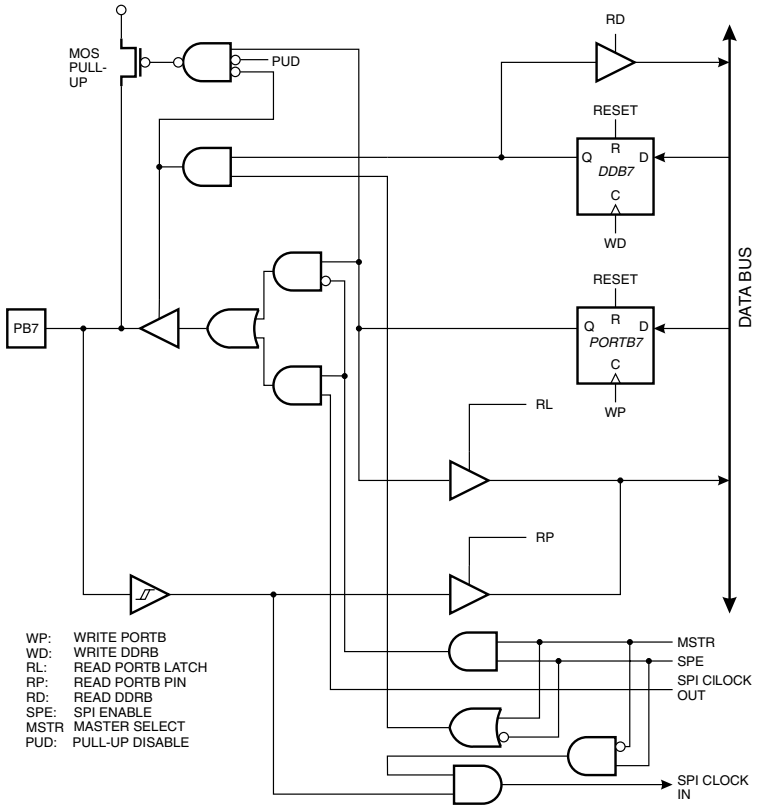


Figure 69. PORTB Schematic Diagram (Pin PB7)



Port C

Port C is an 8-bit bi-directional I/O port with internal pull-ups.

Three I/O memory address locations are allocated for the Port C, one each for the Data Register – PORTC, $15($35), Data Direction Register – DDRC, $14($34) and the Port C Input Pins – PINC, $13($33). The Port C Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The PORT C output buffers can sink 20 mA and thus drive LED displays directly. When pins PC0 to PC7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

Table 47. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC0	SCL (Two-wire Serial Bus Clock Line)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC7	TOSC2 (Timer Oscillator Pin 2)

The Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
$15 ($35)	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	

The Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
$14 ($34)	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	

The Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
$13 ($33)	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	

The Port C Input Pins Address – PINC – is not a register, and this address enables access to the physical value on each Port C pin. When reading PORTC, the PORTC Data Latch is read, and when reading PINC, the logical values present on the pins are read.

Port C as General Digital I/O

All eight bits in PORT C are equal when used as digital I/O pins.

PCn, General I/O pin: The DDCn bit in the DDRC Register selects the direction of this pin, if DDCn is set (one), PCn is configured as an output pin. If DDCn is cleared (zero), PCn is configured as an input pin. If PORTCn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, PORTCn has to be cleared (zero), the pin has to be configured as an output pin, or the PUD bit has to be set. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 48. DDCn Effects on PORT C Pins[1]

DDCn	PORTCn	PUD	I/O	Pull Up	Comment
0	0	x	Input	No	Tri-state (Hi-Z)
0	1	1	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PCn will source current if ext. pulled low.
1	0	x	Output	No	Push-pull Zero Output
1	1	x	Output	No	Push-pull One Output

Note: 1. n: 7…0, pin number

Alternate Functions of PORTC

- **TOSC2 – PORTC, Bit 7**

TOSC2, Timer Oscillator pin 2: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC7 is disconnected from the port, and becomes the inverting output of the Oscillator amplifier. In this mode, a crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- **TOSC1 – PORTC, Bit 6**

TOSC1, Timer Oscillator pin 1: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter1, pin PC6 is disconnected from the port, and becomes the input of the inverting Oscillator amplifier. In this mode, a crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- **SDA – PORTC, Bit 1**

SDA, Two-wire Serial Bus Data: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to capture spikes shorter than 50 ns on the input signal, and the pin is driven by an open collector driver with slew rate limitation.

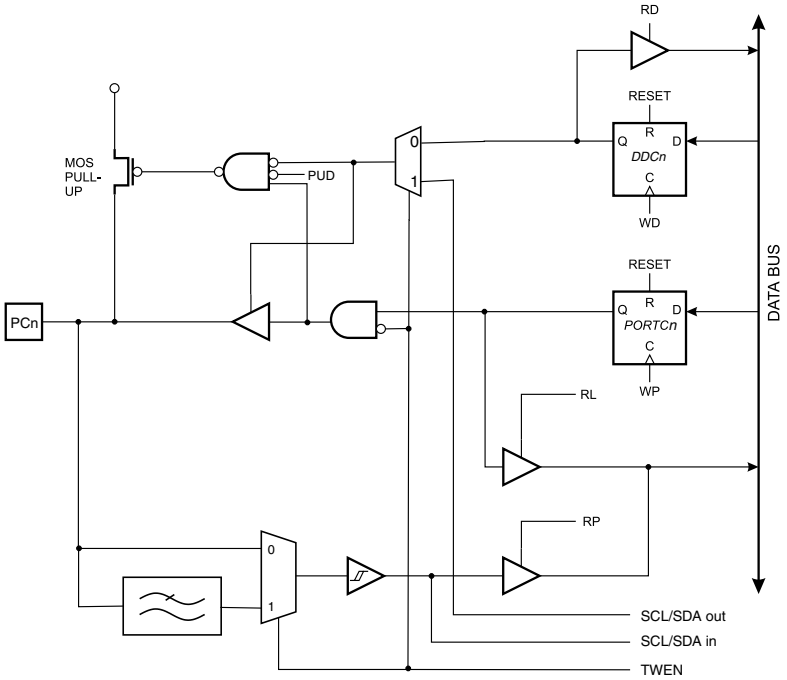
- **SCL – PORTC, Bit 0**

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Clock I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to capture spikes shorter than 50 ns on the input signal.

Port C Schematics

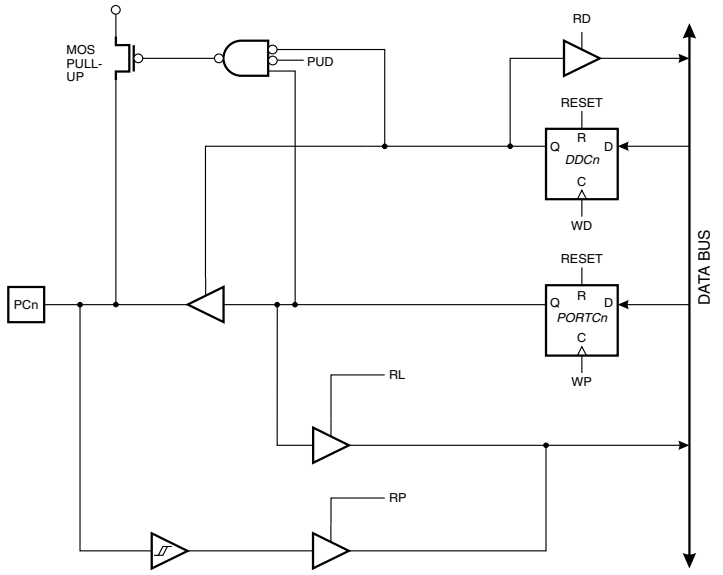
Note that all port pins are synchronized. The synchronization latches are not shown in the figure.

Figure 70. PORTC Schematic Diagram (Pins PC0 - PC1)



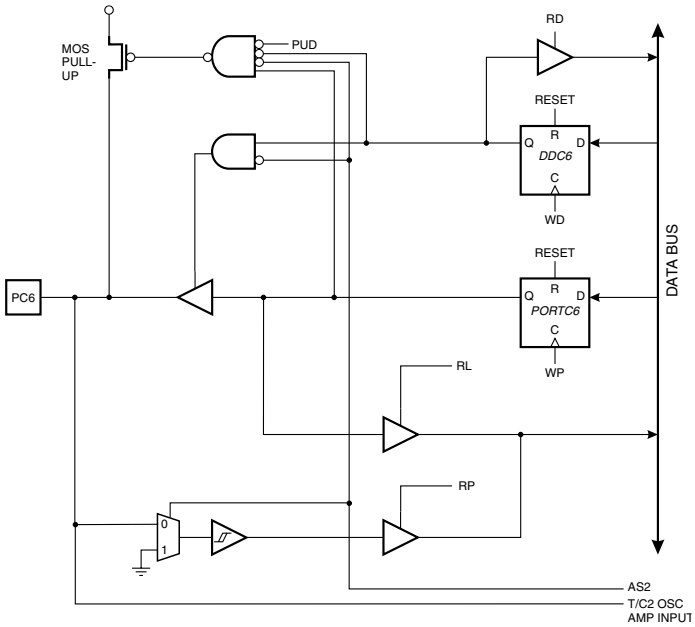
WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
PUD: PULL-UP DISABLE
n = 0, 1

Figure 71. PORTC Schematic Diagram (Pins PC2 - PC5)



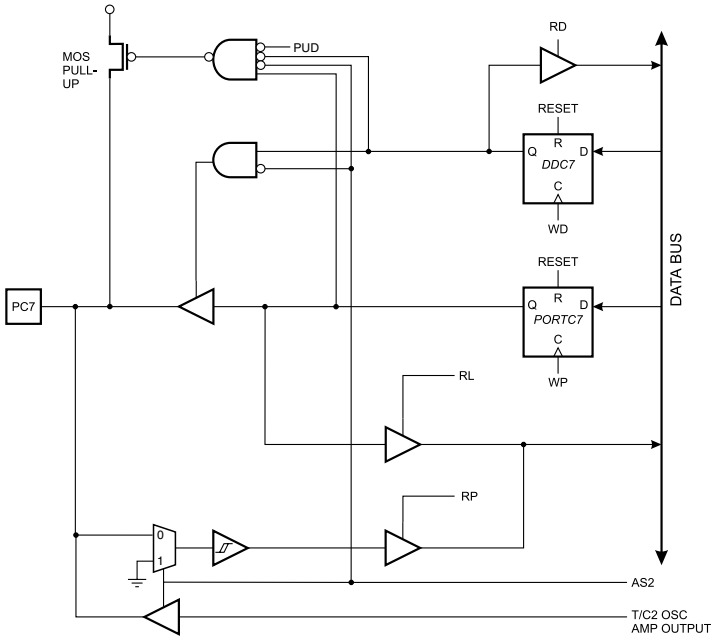
WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
PUD: PULL-UP DISABLE
n: 2..5

Figure 72. PORTC Schematic Diagram (Pins PC6)



WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
AS2: ASYNCH SELECT T/C2
PUD: PULL-UP DISABLE

Figure 73. PORTC Schematic Diagram (Pins PC7)



WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
AS2: ASYNCH SELECT T/C2
PUD: PULL-UP DISABLE

Port D

Port D is an 8 bit bi-directional I/O port with internal pull-up resistors.

Three I/O memory address locations are allocated for Port D, one each for the Data Register – PORTD, $12($32), Data Direction Register – DDRD, $11($31) and the Port D Input Pins – PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. Some Port D pins have alternate functions as shown in Table 49.

Table 49. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RXD (UART Input Pin)
PD1	TXD (UART Output Pin)
PD2	INT0 (External Interrupt 0 Input)
PD3	INT1 (External Interrupt 1 Input)
PD4	OC1B (Timer/Counter1 Output CompareB Match Output)
PD5	OC1A (Timer/Counter1 Output CompareA Match Output)
PD6	ICP (Timer/Counter1 Input Capture Pin)
PD7	OC2 (Timer/Counter2 Output Compare Match Output)

The Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	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	

The Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	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	

The Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	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	

The Port D Input Pins Address – PIND – is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

Port D as General Digital I/O

PDn, General I/O pin: The DDDn bit in the DDRD Register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin the MOS pull up resistor is activated. To switch the pull up resistor off the PDn has to be cleared (zero), the pin has to be configured as an output pin, or the PUD bit has to be set. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 50. DDDn Bits on Port D Pins(1)

DDDn	PORTDn	PUD	I/O	Pull Up	Comment
0	0	x	Input	No	Tri-state (Hi-Z)
0	1	1	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PDn will source current if ext. pulled low.
1	0	x	Output	No	Push-pull Zero Output
1	1	x	Output	No	Push-pull One Output

Note: 1. n: 7,6…0, pin number.

Alternate Functions of PORTD

- **OC2 – PORTD, Bit 7**

OC2, Timer/Counter2 Output Compare Match output: The PD7 pin can serve as an external output for the Timer/Counter2 Output Compare. The pin has to be configured as an output (DDD7 set (one)) to serve this function. See the Timer description on how to enable this function. The OC2 pin is also the output pin for the PWM mode timer function.

- **ICP – PORTD, Bit 6**

ICP – Input Capture Pin: The PD6 pin can act as an Input Capture pin for Timer/Counter1. The pin has to be configured as an input (DDD6 cleared(zero)) to serve this function. See the timer description on how to enable this function.

- **OC1A – PORTD, 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. See the timer description on how to enable this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **OC1B – PORTD, Bit 4**

OC1B, Output Compare Match B output: The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set (one)) to serve this function. See the timer description on how to enable this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **INT1 – PORTD, Bit 3**

INT1, External Interrupt Source 1: The PD3 pin can serve as an External Interrupt Source to the MCU. See the interrupt description for further details, and how to enable the source.

- **INT0 – PORTD, Bit 2**

INT0, External Interrupt Source 0: The PD2 pin can serve as an External Interrupt Source to the MCU. See the interrupt description for further details, and how to enable the source.

- **TXD – Port D, Bit 1**

TXD, Transmit Data (Data output pin for the UART). When the UART Transmitter is enabled, this pin is configured as an output regardless of the value of DDRD1.

- **RXD – Port D, Bit 0**

RXD, Receive Data (Data input pin for the UART). When the UART Receiver is enabled this pin is configured as an input regardless of the value of DDRD0. When the UART forces this pin to be an input, a logical one in PORTD0 will turn on the internal pull-up.

Port D Schematics

Note that all port pins are synchronized. The synchronization latches are not shown in the figures.

Figure 74. PORTD Schematic Diagram (Pin PD0)

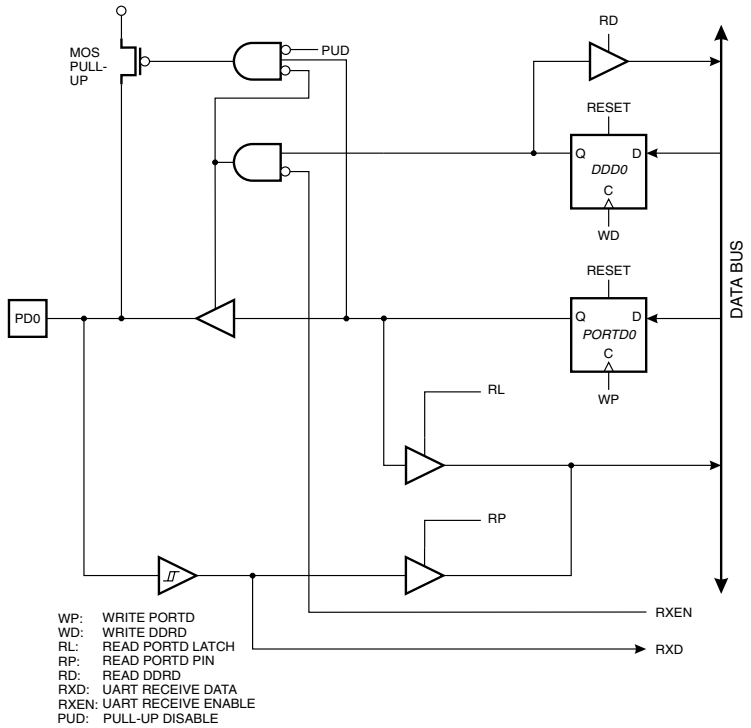


Figure 75. PORTD Schematic Diagram (Pin PD1)

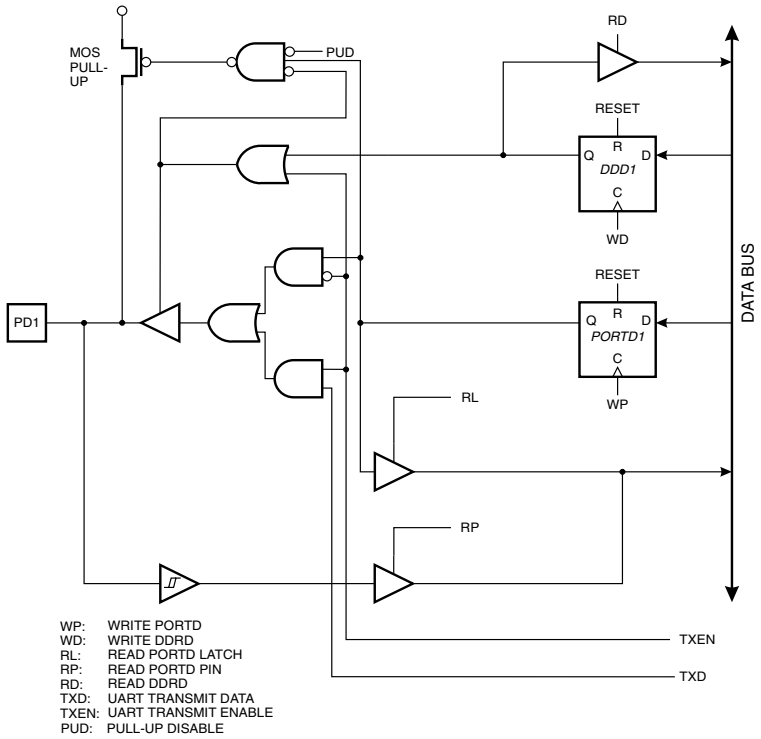


Figure 76. PORTD Schematic Diagram (Pins PD2 and PD3)

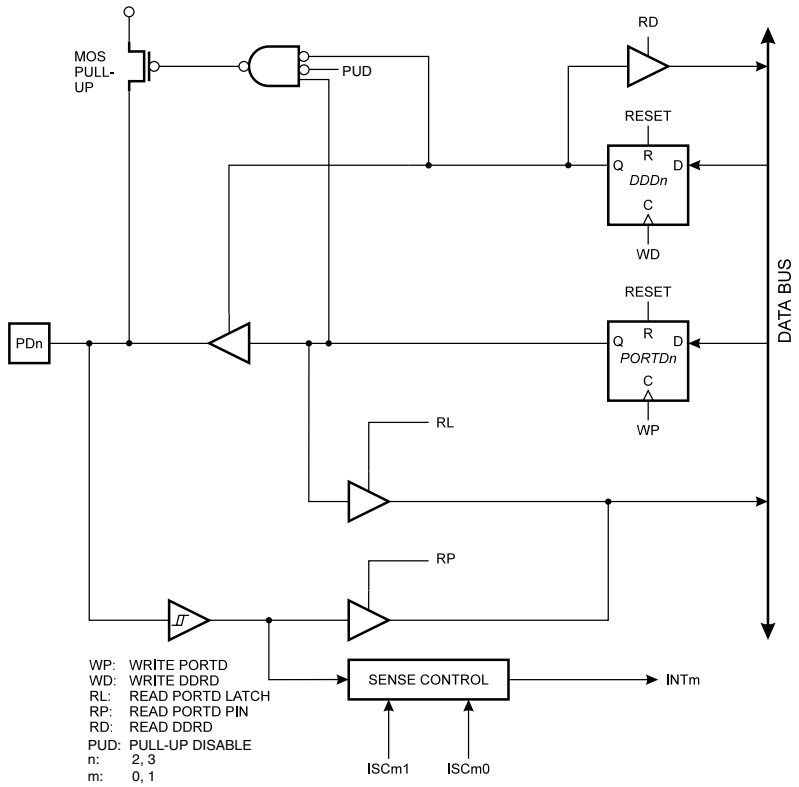


Figure 77. PORTD Schematic Diagram (Pins PD4 and PD5)

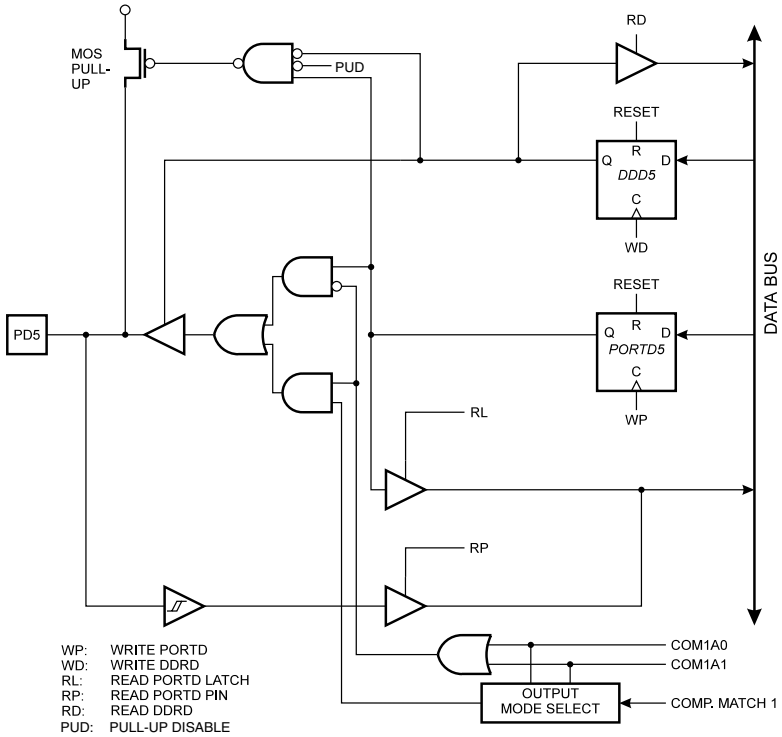


Figure 78. PORTD Schematic Diagram (Pin PD6)

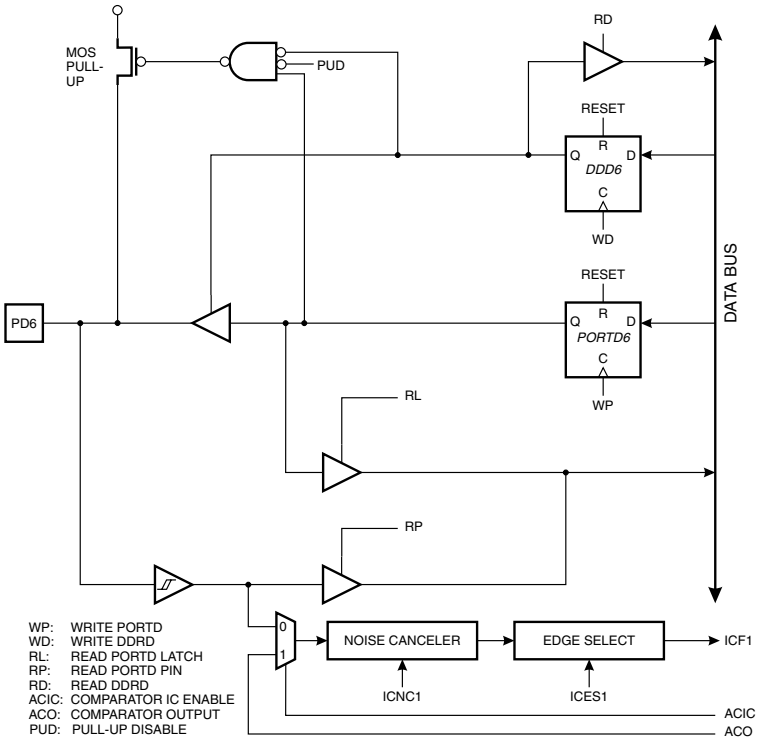
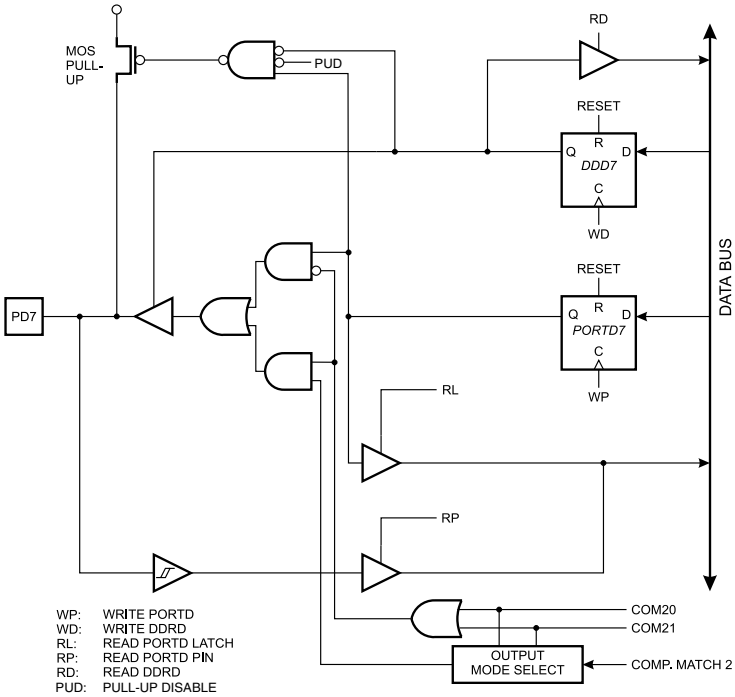


Figure 79. PORTD Schematic Diagram (Pin PD7)



Memory Programming

Boot Loader Support

The ATmega163 provides a mechanism for Programming and Re-programming code by the MCU itself. This feature allows flexible application software updates, controlled by the MCU using a Flash-resident Boot Loader program. This makes it possible to program the AVR in a target system without access to its SPI pins. The Boot Loader program can use any available data interface and associated protocol, such as UART serial bus interface, to input or output program code, and write (program) that code into the Flash memory, or read the code from the Flash memory.

The ATmega163 Flash memory is organized in two main sections:

- The Application Flash section
- The Boot Loader Flash section

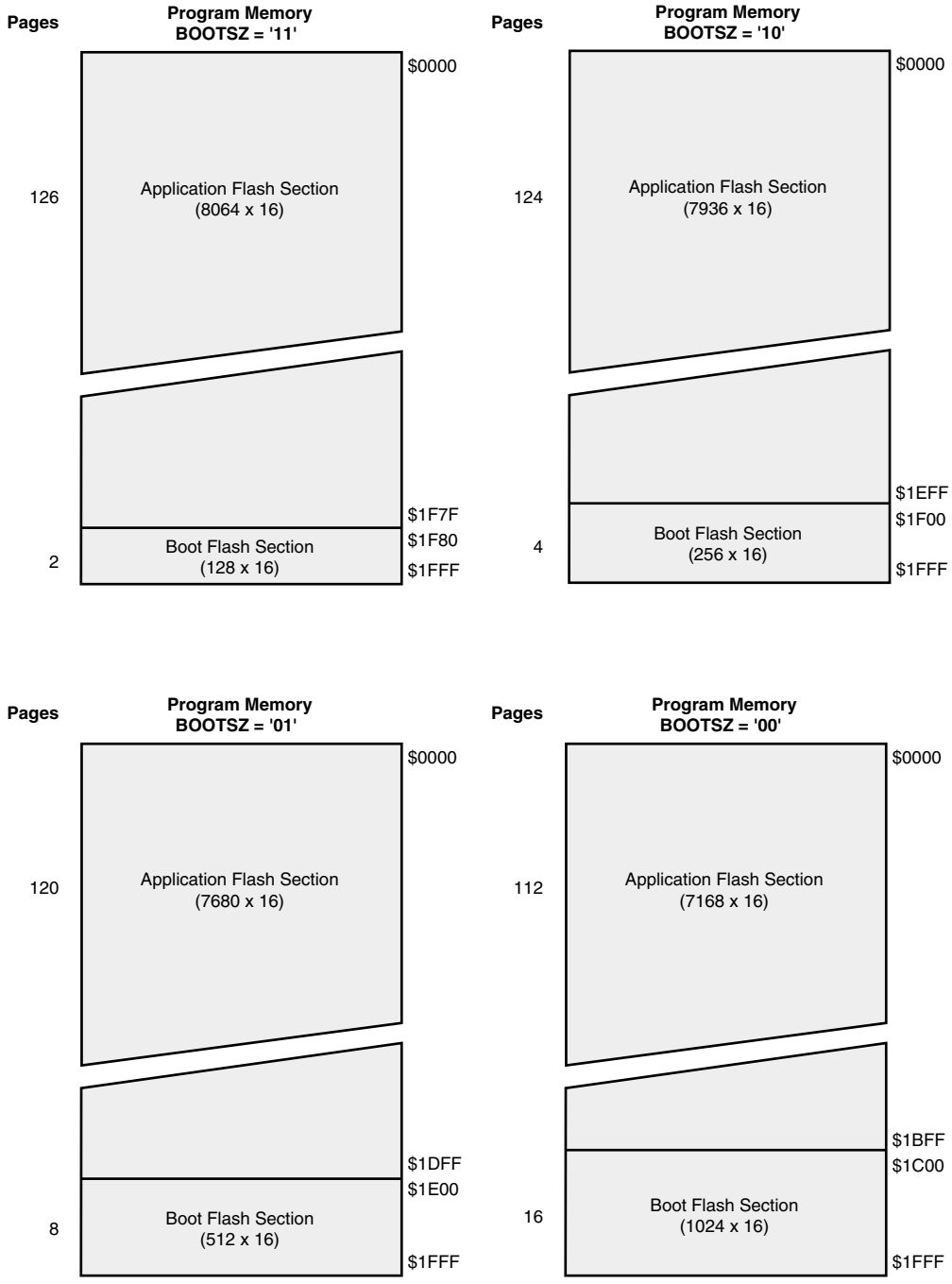
The Application Flash section and the Boot Loader Flash section have seperate Boot Lock bits. Thus the user can select different levels of protection for the two sections. The Store Program Memory (SPM) instruction can only be executed from the Boot Loader Flash section.

The Program Flash memory in ATmega163 is divided into 128 pages of 64 words each. The Boot Loader Flash section is located at the high address space of the Flash, and can be configured through the BOOTSZ Fuses as shown in Table 51.

Table 51. Boot Size Configuration

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses
1	1	128 Words	2	$0000 - $1F7F	$1F80 - $1FFF
1	0	256 Words	4	$0000 - $1EFF	$1F00 - $1FFF
0	1	512 Words	8	$0000 - $1DFF	$1E00 - $1FFF
0	0	1024 Words	16	$0000 - $1BFF	$1C00 - $1FFF

Figure 80. Memory Sections



Entering the Boot Loader Program

The SPM instruction can access the entire Flash, but can only be executed from the Boot Loader Flash section. If no Boot Loader capability is needed, the entire Flash is available for application code. Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by some trigger such as a command received via UART or SPI interface, for example. 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 52. Boot Reset Fuse

BOOTRST	Reset Address
0	Reset Vector = Application Reset (address $0000)
1	Reset Vector = Boot Loader Reset (see Table 51)

Capabilities of the Boot Loader

The program code within the Boot Loader section has the capability to read from and write into the entire Flash, including the Boot Loader memory. This allows the user to update both the Application code and the Boot Loader code that handles the software update. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore.

Self-Programming the Flash

Programming of the Flash is executed one page at a time. The Flash page must be erased first for correct programming. The general Write Lock (Lock bit 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock bit 1) does not control reading nor writing by LPM/SPM, if it is attempted.

The Program memory can only be updated page by page, not word by word. One page is 128 bytes (64 words). The Program memory will be modified by first performing Page Erase, then filling the temporary page buffer one word at a time using SPM, and then executing Page Write. If only part of the page needs to be changed, the other parts must be stored (for example in internal SRAM) before the erase, and then be rewritten. 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 “Assembly code example for a Boot Loader” on page 141 for an assembly code example.

See Table 60 on page 156 for typical programming times when using Self-Programming.

Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “00011” to the five LSB in 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 Z13:Z7. Other bits in the Z-pointer will be ignored during this operation. It is recommended that the interrupts are disabled during the page erase operation.

Fill the Temporary Buffer (Page Load)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00001” to the five LSB in SPMCR and execute SPM within four clock cycles after writing SPMCR. The content of Z6:Z1 is used to address the data in the temporary buffer. Z13:Z7 must point to the page that is supposed to be written.

Perform a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00101” to the five LSB in 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 Z13:Z7. During this operation, Z6:Z0 must be zero to ensure that the page is written correctly. It is recommended that the interrupts are disabled during the page write operation.

Consideration while Updating the Boot Loader Section

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit 11 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 Bit 11 to protect the Boot Loader software from any internal software changes.

Wait for SPM Instruction to Complete

Though the CPU is halted during Page Write, Page Erase or Lock bit write, for future compatibility, the user software must poll for SPM complete by reading the SPMCR Register and loop until the SPMEN bit is cleared after a programming operation. See “Assembly code example for a Boot Loader” on page 141 for a code example.

Instruction Word Read after Page Erase, Page Write, and Lock Bit Write

To ensure proper instruction pipelining after programming action (Page Erase, Page Write, or Lock bit write), the SPM instruction must be followed with the sequence (.dw $FFFF - NOP) as shown below:

```
spm
.dw $FFFF
nop
```

If not, the instruction following SPM might fail. It is not necessary to add this sequence when the SPM instruction only loads the temporary buffer.

Avoid Reading the Application Section During Self-Programming

During Self-Programming (either Page Erase or Page Write), the user software should not read the application section. The user software itself must prevent addressing this section during the Self-Programming operations. This implies that interrupts must be disabled. Before addressing the application section after the programming is completed, for future compatibility, the user software must write “10001” to the five LSB in SPMCR and execute SPM within four clock cycles. Then the user software should verify that the ASB bit is cleared. See “Assembly code example for a Boot Loader” on page 141 for an example. Though the ASB and ASRE bits have no special function in this device, it is important for future code compatibility that they are treated as described above.

Boot Loader Lock Bits

ATmega163 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 only protect the Boot Loader Flash section from a software update by the MCU
- To only protect application Flash section from a software update by the MCU
- Allowing software update in the entire Flash

See Table and Table for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can only be cleared by a chip erase command.

Table 53. Boot Lock Bit0 Protection Modes (Application Section)[1]

BLB0 mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM, 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
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section

Note: 1. "1" means unprogrammed, "0" means programmed

Table 54. Boot Lock Bit1 Protection Modes (Boot Loader Section)[1]

BLB1 mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM, 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 code is executed from Boot section, the interrupts are disabled when BLB12 is programmed.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If code is executed from Boot section, the interrupts are disabled when BLB12 is programmed.

Note: 1. "1" means unprogrammed, "0" means programmed

Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock bits, write the desired data to R0, write "00001001" 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**

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.

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 five 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 SPM, or LPM, instruction is executed within four, respectively five, CPU cycles. When BLBSET and SPMEN are cleared, LPM will work as described in "Constant Addressing Using The LPM and SPM Instructions" on page 15 and 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 five cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse Low bits will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	BODLEVEL	BODEN	SPIEN	–	CKSEL3	CKSEL2	CKSEL1	CKSEL0

Similarly, when reading the Fuse High bits, load $0003 in the Z-pointer. When an LPM instruction is executed within five cycles after the BLBSET and SPMEN bits are set in the SPMCR, the value of the Fuse High bits will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	BOOTSZ1	BOOTSZ0	BOOTRST

Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

In all cases, the read value of unused bit positions are undefined.

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. If EEPROM writing is performed inside an interrupt routine, the user software should disable that interrupt before checking the EEWE status bit.

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

Z15:Z14 always ignored

Z13:Z7 page select, for page erase and page write

Z6:Z1 word select, for filling temp buffer (must be zero during page write operation)

Z0 should be zero for all SPM commands, byte select for the LPM instruction.

The only 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.

Note that the Page Erase and Page Write operation is 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.

The LPM instruction also uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used. See page 15 for a detailed description.

Store Program Memory Control Register – SPMCR

The Store Program Memory Control Register contains the control bits needed to control the programming of the Flash from internal code execution.

Bit	7	6	5	4	3	2	1	0	
$37 ($57)	–	ASB	–	ASRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	x	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and always reads as zero. This bit should be written to zero when writing SPMCR.

- **Bit 6 – ASB: Application Section Busy**

Before entering the Application section after a Boot Loader operation (Page Erase or Page Write) the user software must verify that this bit is cleared. In future devices, this bit will be set to “1” by Page Erase and Page Write. In ATmega163, this bit always reads as zero.

- **Bit 5 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and always reads as zero. This bit should be written to zero when writing SPMCR.

- **Bit 4 – ASRE: Application Section Read Enable**

Before re-entering the Application section, the user software must set this bit together with the SPMEN bit and execute SPM within four clock cycles.

- **Bit 3 – BLBSET: Boot Lock Bit Set**

If this bit is set at the same time as SPMEN, the next SPM instruction within four clock cycles will set Boot Lock bits. Alternatively, an LPM instruction within five cycles will read either the Lock bBits or the Fuse bits. The BLBSET bit will auto-clear upon completion of the SPM or LPM instruction, or if no SPM, or LPM, instruction is executed within four, respectively five, clock cycles.

- **Bit 2 – PGWRT: Page Write**

If this bit is set at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 1 – PGERS: Page Erase**

If this bit is set 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 Erase operation.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If set together with either ASRE, BLBSET, PGWRT, or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is set, 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”, or “00001” in the lower five bits will have no effect.

Preventing Flash Corruption

During periods of low V_{CC}, the Flash 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 corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done be 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. The total Reset Time must be longer thatn the Flash write time. This can be achieved by holding the External Reset, or by selecting a long Reset Time-out.
2. Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the Flash from unintentional writes.

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 (lowest
address)
  ;  the first data location in Flash is pointed to by the Z-pointer (lowest
address)
  ;- error handling is not included
  ;- the routine must be placed inside the boot space
  ;  Only code inside boot loader
  ;  section should be read during Self-Programming.
  ;- registers used: r0, r1, temp1, temp2, looplo, loophi, spmcrval
  ;  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 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)
      call    Do_spm
```

```
  ; re-enable the Application Section
      ldi     spmcrval, (1<<ASRE) + (1<<SPMEN)
      call    Do_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)
      call    Do_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)
      call    Do_spm

  ; re-enable the Application Section
      ldi     spmcrval, (1<<ASRE) + (1<<SPMEN)
      call    Do_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
      jmp     Error
      sbiw    loophi:looplo, 2        ;use subi for PAGESIZEB<=256
      brne    Rdloop

  ; return to Application Section
  ; verify that Application Section is safe to read
Return:
      in      temp1, SPMCR
      sbrs    temp1, ASB              ; If ASB is set, the AS is not ready yet
      ret
  ; re-enable the Applicaiton Section
      ldi     spmcrval, (1<<ASRE) + (1<<SPMEN)
      call    Do_spm
      rjmp    Return

Do_spm:
  ; input: spmcrval determines SPM action
  ; check that no EEPROM write access is running
Wait_ee:
      sbic    EECR, EEWE
      rjmp    Wait_ee
  ; SPM timed sequence
      out     SPMCR, spmcrval
      spm
      .dw $FFFF                       ; ensure proper pipelining
      nop                             ; of next instruction
  ; check for SPM complete
Wait_spm:
      in      temp1, SPMCR
```

```
sbrc    temp1, SPMEN
rjmp    Wait_spm
ret
```

Program and Data Memory Lock Bits

The ATmega163 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 55. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 55. Lock Bit Protection Modes

Memory Lock Bits			
LB mode	**LB1**	**LB2**	**Protection Type**
1	1	1	No memory lock features enabled for Parallel and Serial Programming.
2	0	1	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	**BLB01**	**BLB02**	
1	1	1	No restrictions for SPM, LPM accessing the Application section.
2	0	1	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.
4	1	0	LPM executing from the Boot Loader section is not allowed to read from the Application section.
BLB1 mode	**BLB11**	**BLB12**	
1	1	1	No restrictions for SPM, LPM accessing the Boot Loader section.
2	0	1	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 code executed from the Boot Section, the interrupts are disabled when BLB12 is programmed.
4	1	0	LPM executing from the Application section is not allowed to read from the Boot Loader section. If code executed from the Boot Section, the interrupts are disabled when BLB12 is programmed.

Note: 1. Program the Fuse bits before programming the Lock bits.

Fuse Bits

The ATmega163 has ten Fuse bits, divided in two groups. The Fuse High bits are BOOTSZ1..0 and BOOTRST, and the Fuse Low bits are BODLEVEL, BODEN, SPIEN, and CKSEL3..0.

- BOOTSZ1..0 select the size and start address of the Boot Flash section according to Table 51 on page 134. Default value is “11” (both unprogrammed).
- When BOOTRST is programmed (“0”), the Reset Vector is set to the start address of the Boot Flash section, as selected by the BOOTSZ fuses according to Table 51 on page 134. If the BOOTRST is unprogrammed (“1”), the Reset Vector is set to address $0000. Default value is unprogrammed (“1”).
- The BODLEVEL Fuse selects the Brown-out Detection Level and changes the Start-up times, according to Table 4 on page 24 and Table 5 on page 25, respectively. Default value is unprogrammed (“1”).
- When the BODEN Fuse is programmed (“0”), the Brown-out Detector is enabled. See “Reset and Interrupt Handling” on page 21. Default value is unprogrammed (“1”).
- When the SPIEN Fuse is programmed (“0”), Serial Program and Data Downloading are enabled. Default value is programmed (“0”). The SPIEN Fuse is not accessible in serial programming mode.
- CKSEL3..0 select the clock source and the start-up delay after reset, according to Table 1 on page 5 and Table 5 on page 25. Default value is “0010” (Internal RC Oscillator).

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.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode. The three bytes reside in a separate address space.

The ATmega163 the signature bytes are:

1. $000: $1E (indicates manufactured by Atmel).
2. $001: $94 (indicates 16KB Flash memory).
3. $002: $02 (indicates ATmega163 device when $001 is $94).

Calibration Byte

The ATmega163 has a one byte calibration value for the internal RC Oscillator. This byte resides in the high byte of address $000 in the signature address space. During Memory Programming, the external programmer must read this location and program it into a selected location in the normal Flash Program memory. At start-up, the user software must read this Flash location and write the value to the OSCCAL Register.

Parallel Programming

This section describes how to Parallel Program and verify Flash Program memory, EEPROM Data memory + Program And Data Memory Lock bits and Fuse bits in the ATmega163. Pulses are assumed to be at least 500ns unless otherwise noted.

Signal Names

In this section, some pins of the ATmega163 are referenced by signal names describing their functionality during parallel programming, see Figure 81 and Table 56. 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 are shown in Table 57.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The Command is a byte where the different bits are assigned functions as shown in Table 58.

The BS2 pin should be low unless otherwise noted.

Figure 81. Parallel Programming

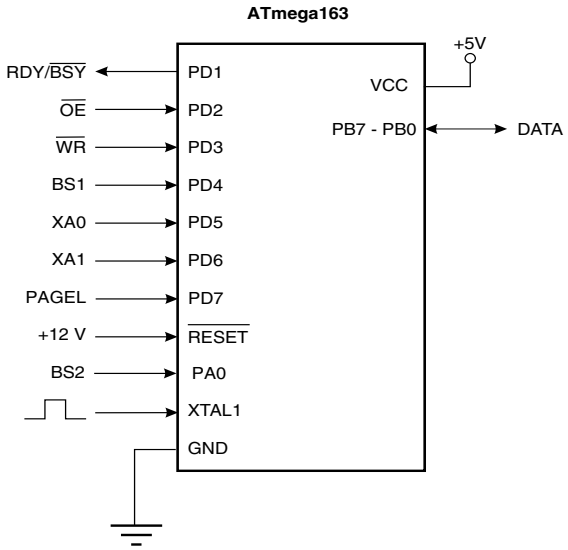


Table 56. Pin Name Mapping

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

Table 56. Pin Name Mapping (Continued)

Signal Name in Programming Mode	Pin Name	I/O	Function
PAGEL	PD7	I	Program Memory Page Load
BS2	PA0	I	Byte Select 2 (“0” selects low byte, “1” selects 2’nd high byte)
DATA	PB7 - 0	I/O	Bidirectional Databus (Output when $\overline{OE}$ is low)

Table 57. 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 58. 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
0000 0100	Read Fuse and Lock Bits
0000 0010	Read Flash
0000 0011	Read EEPROM

Enter Programming Mode

The following algorithm puts the device in Parallel Programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND.
2. Set $\overline{RESET}$ and BS pins to “0” and wait at least 100 ns.
3. Apply 11.5 - 12.5V to $\overline{RESET}$. Any activity on BS1 within 100 ns after +12V has been applied to $\overline{RESET}$, will cause the device to fail entering Programming mode.

Chip Erase

The Chip Erase command will erase the Flash and EEPROM memories and the 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 is re-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 $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
5. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

Programming the Flash

The Flash is organized as 128 pages of 128 bytes each. 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. Latch Data Low Byte

1. Set BS1 to “0”. This selects Low Data Byte.
2. Give PAGEL a positive pulse. This latches the data Low Byte. (See Figure 82 for signal waveforms)

E. 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.

F. Latch Data High Byte

1. Set BS1 to “1”. This selects High Data Byte.
2. Give PAGEL a positive pulse. This latches the data High Byte.

G. Repeat B through F 64 times to fill the page buffer.

To address a page in the Flash, seven bits are needed (128 pages). The five most significant bits are read from address high byte as described in section “H” below. The two least significant page address bits however, are the two most significant bits (bit7 and bit6) of the latest loaded address low byte as described in section “B”.

H. Load Address High byte

1. 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 - $1F).
4. Give XTAL1 a positive pulse. This loads the address High Byte.

I. Program Page

1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$goes low.
2. Wait until RDY/$\overline{BSY}$ goes high.
   (See Figure 83 for signal waveforms)

J. End Page Programming

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.

K. Repeat A through J 128 times or until all data has been programmed.

Figure 82. Programming the Flash Waveforms

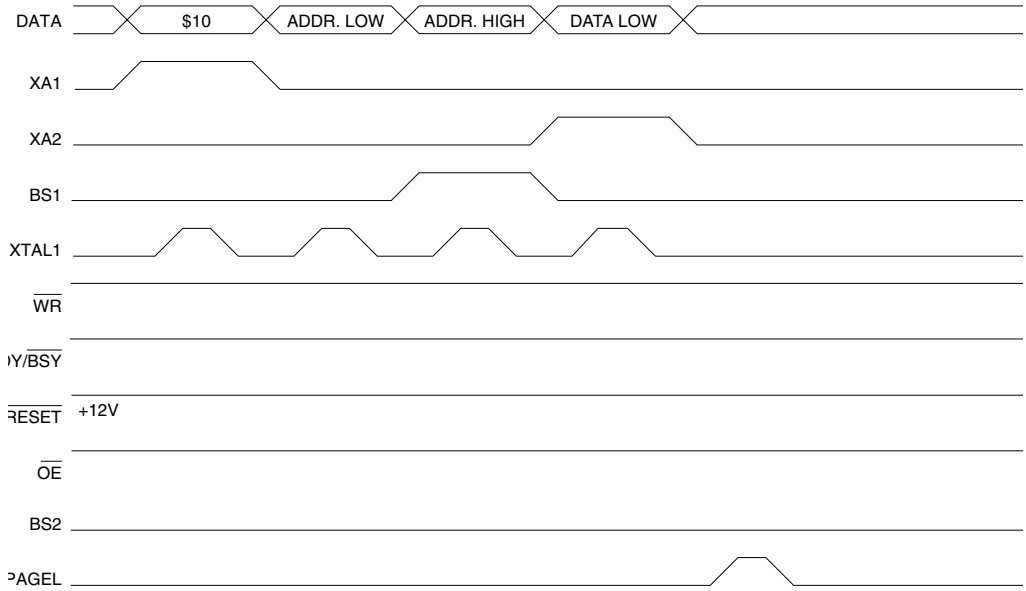
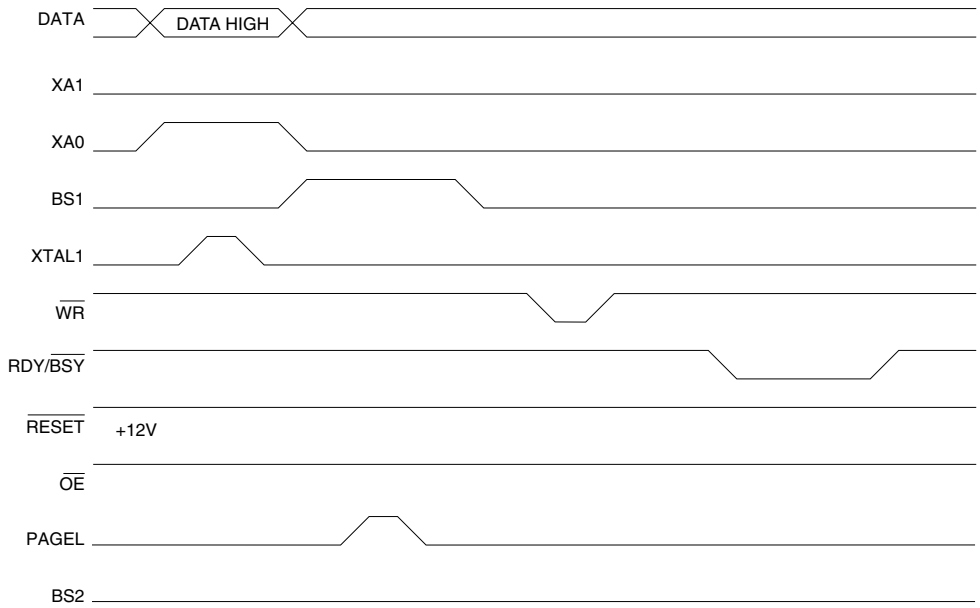


Figure 83. Programming the Flash Waveforms (continued)



Programming the EEPROM

The programming algorithm for the EEPROM Data Memory is as follows (refer to "Programming the Flash" on page 147 for details on Command, Address and Data loading):

1. A: Load Command "0001 0001".
2. H: Load Address High Byte ($00 - $01)
3. B: Load Address Low Byte ($00 - $FF)
4. E: Load Data Low Byte ($00 - $FF)

L: Write Data Low Byte

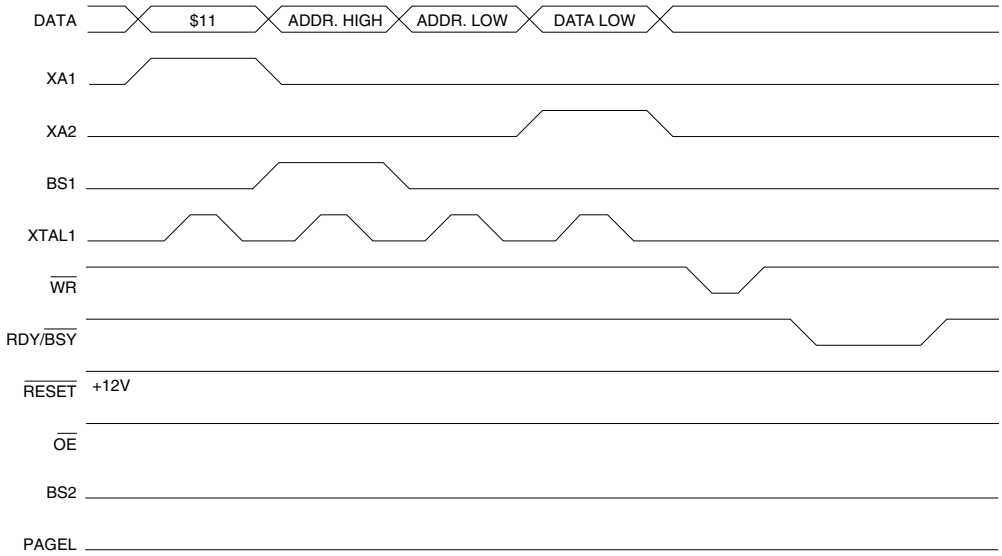
1. Set BS to "0". This selects low data.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next byte. (See Figure 84 for signal waveforms)

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.
- Address high byte needs only be loaded before programming a new 256 word page in the EEPROM.
- Skip writing the data value $FF, that is the contents of the entire EEPROM after a Chip Erase.

These considerations also applies to Flash, EEPROM and Signature bytes reading.

Figure 84. Programming the EEPROM Waveforms



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 147 for details on Command and Address loading):

1. A: Load Command "0000 0010".
2. H: Load Address High Byte ($00 - $1F).
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 BS 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 147 for details on Command and Address loading):

1. A: Load Command "0000 0011".
2. H: Load Address High Byte ($00 - $01).
3. B: Load Address ($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 147 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.
   Bit 7 = BODLEVEL Fuse bit
   Bit 6 = BODEN Fuse bit
   Bit 5 = SPIEN Fuse bit
   Bit 3..0 = CKSEL3..0 Fuse bits
   Bit 4 = "1". This bit is reserved and should be left unprogrammed ("1").
3. 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 147 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.
   Bit 2..1 = BOOTSZ1..0 Fuse bits
   Bit 0 = BOOTRST Fuse bit
   Bit 7..3 = “1”. These bits are reserved and should be left unprogrammed (“1”).
3. Set BS1 to “1”. 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.

Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 147 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. D: Load Data Low Byte. Bit n = “0” programs the Lock bit.
   Bit 5 = Boot Lock bit12
   Bit 4 = Boot Lock bit11
   Bit 3 = Boot Lock bit02
   Bit 2 = Boot Lock bit01
   Bit 1 = Lock bit2
   Bit 0 = Lock bit1
   Bit 7..6 = “1”. These bits are reserved and should be left unprogrammed (“1”).
3. L: Write Data Low Byte.

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 147 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).
   Bit 7 = BODLEVEL Fuse bit
   Bit 6 = BODEN Fuse bit
   Bit 5 = SPIEN Fuse bit
   Bit 3..0 = CKSEL3..0 Fuse bits
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).
   Bit 2..1 = BOOTSZ1..0 Fuse bits
   Bit 0 = BOOTRST Fuse bit
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).
   Bit 5 = Boot Lock bit12
   Bit 4 = Boot Lock bit11
   Bit 3 = Boot Lock bit02
   Bit 2 = Boot Lock bit01
   Bit 1 = Lock bit2
   Bit 0 = Lock bit1
5. Set $\overline{OE}$ to “1”.

Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to Programming the Flash for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. C: Load Address Low Byte ($00 - $02).
3. Set $\overline{OE}$ to “0”, and BS to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to Programming the Flash for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. C: Load Address Low Byte, $00.
   Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibaration byte can now be read at DATA.
3. Set $\overline{OE}$ to “1”.

Parallel Programming Characteristics

Figure 85. Parallel Programming Timing

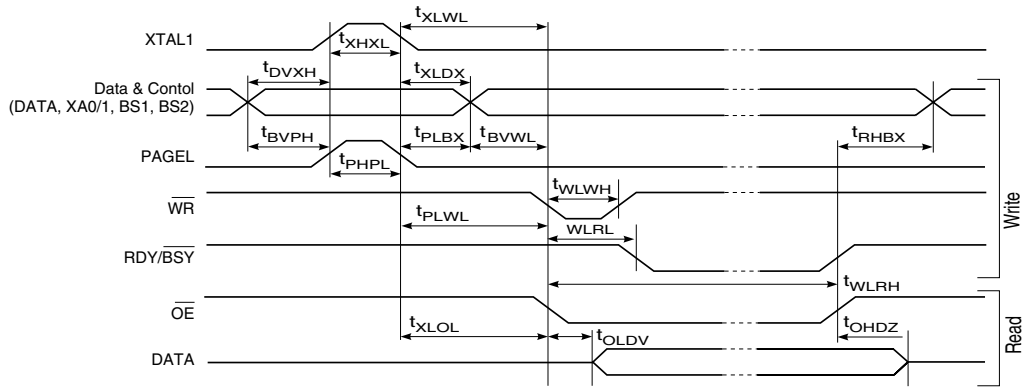


Table 59. Parallel Programming Characteristics, T_A = 25°C ± 10%, V_{CC} = 5 V ± 10%

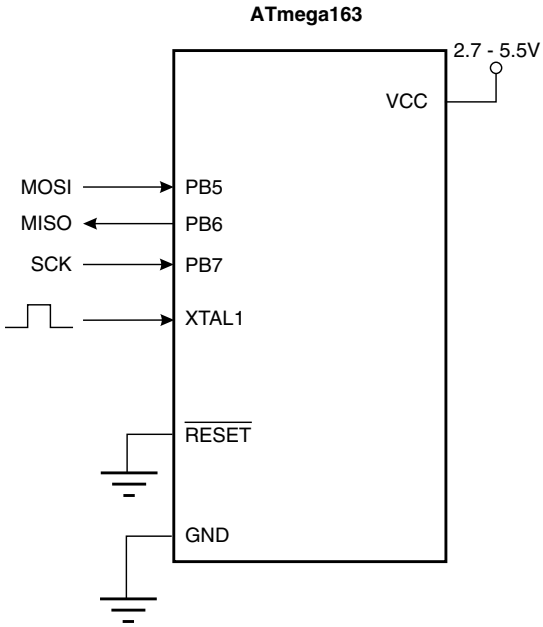
Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XHXL}	XTAL1 Pulse Width High	67			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	67			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	67			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{RHBX}	BS1 Hold after RDY/$\overline{BSY}$ High	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	67			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		2.5	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High(1)	1	1.5	1.9	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase(2)	16	23	30	ms
t_{WLRH_FLASH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Write Flash(3)	8	12	15	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	67			ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid		20		ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			20	ns

Notes: 1. t_{WLRH} is valid for the Write EEPROM, Write Fuse Bits and Write Lock Bits commands.
2. t_{WLRH_CE} is valid for the Chip Erase command.
3. t_{WLRH_FLASH} is valid for the Write Flash 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.

Figure 86. Serial Programming and Verify



Note: If the device is clocked by the internal Oscillator, connecting a clock source to XTAL1 is not required.

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.

The Program and EEPROM memory arrays have separate address spaces:

$0000 to $1FFF for Program memory and $0000 to $01FF for EEPROM memory.

The device can be clocked by any clock option during Serial Programming. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 MCU clock cycles

High: > 2 MCU clock cycles

Serial Programming Algorithm

When writing serial data to the ATmega163, data is clocked on the rising edge of SCK. When reading data from the ATmega163, data is clocked on the falling edge of SCK. See Figure 87, Figure 88 and Table 62 for timing details.

To program and verify the ATmega163 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 61):

1. Power-up sequence:

   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In accordance with the setting of CKSEL Fuses, apply a crystal/resonator, external clock, or RC network, or let the device run on the internal RC Oscillator. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, wait for 100 ms after SCK has been set to “0”. $\overline{\text{RESET}}$ must be then given a positive pulse of at least two XTAL1 cycles duration and then set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/PB5.
3. The Serial Programming instructions will not work if the communication is out of synchronization. When in sync. 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 SCK a positive pulse and issue a new Programming Enable command. If the $53 is not seen within 32 attempts, there is no functional device connected.
4. If a Chip Erase is performed (must be done to erase the Flash), wait $2 \bullet t_{WD\_FLASH}$ after the instruction, give $\overline{\text{RESET}}$ a positive pulse, and start over from Step 2. See Table 60 for the t_{WD_FLASH} figure.
5. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load Program Memory Page instruction. 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. (Please refer to Table 60). Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
6. 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. (Please refer to Table 60). In a chip erased device, no $FFs in the data file(s) need to be programmed.
7. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/PB6.
8. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
9. Power-off sequence (if needed):
   Set XTAL1 to “0” (if external clock is used).
   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 60 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 re-programmed 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 60 for t_{WD_EEPROM} value.

Programming Times for Non-volatile Memory

The internal RC Oscillator is used to control programming time when programming or erasing Flash, EEPORM, Fuses, and Lock bits. During Parallel or Serial Programming, the device is in reset, and this Oscillator runs at its initial, uncalibrated frequency, which may vary from 0.5 MHz to 1.0 MHz. In software it is possible to calibrate this Oscillator to 1.0 MHz (see “Calibrated Internal RC Oscillator” on page 37). Consequently, programming times will be shorter and more accurate when Programming or erasing non-volatile memory from software, using SPM or the EEPROM interface. See Table 60 for a summary of programming times.

Table 60. Maximum Programming Times for Non-volatile Memory

Operation	Symbol	Number of RC Oscillator Cycles	Parallel/Serial Programming		Self-Programming[1]
			2.7V	5.0V	
Chip Erase	t_{WD_CE}	16K	32 ms	30 ms	17 ms
Flash Write[3]	t_{WD_FLASH}	8K	16 ms	15 ms	8.5 ms
EEPROM Write[2]	t_{WD_EEPROM}	2K	4 ms	3.8 ms	2.2 ms
Fuse/lock bit write	t_{WD_FUSE}	1K	2 ms	1.9 ms	1.1 ms

Notes: 1. Includes variation over voltage and temperature after RC Oscillator has been calibrated to 1.0 MHz
2. Parallel EEPROM Programming takes 1K cycles
3. Per page

Figure 87. Serial Programming Waveforms

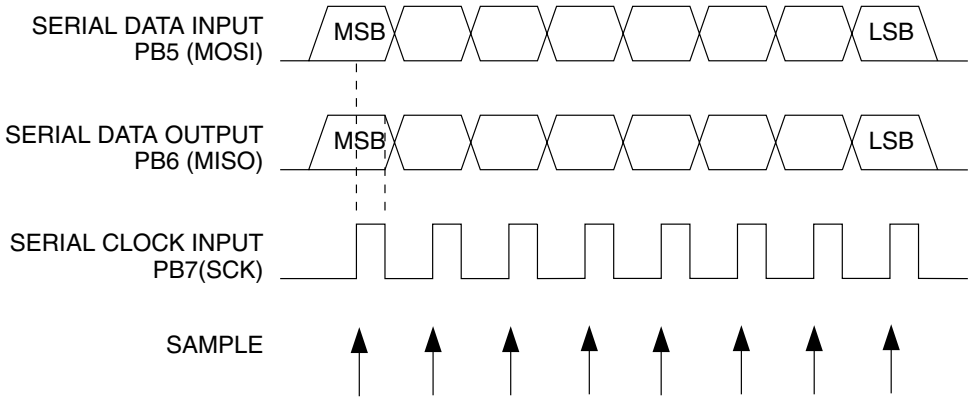


Table 61. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after $\overline{\text{RESET}}$ goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program Memory	0010 **H**000	xxxa **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	xxxx xxxx	xx**bb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program Memory page at word address **b**.
Write Program Memory Page	0100 1100	xxxa **aaaa**	**bb**xx xxxx	xxxx xxxx	Write Program Memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	xxxx xxx**a**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	xxxx xxx**a**	**bbbb bbbb**	iiii iiii	Write data **i** to EEPROM memory at address **a**:**b**.
Read Lock Bits	0101 1000	0000 0000	xxxx 0xxx	xx***65 4321***	Read Lock bits. “0” = programmed, “1” = unprogrammed.
Write Lock Bits	1010 1100	111x xxxx	xxxx xxxx	11***65 4321***	Write Lock bits. Set bits ***6*** - ***1*** = “0” to program Lock bits.
Read Signature Byte	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse Bits	1010 1100	1010 0000	xxxx xxxx	**CB**11 **A987**	Set bits **C** - **A**, **9** - **7** = “0” to program, “1” to unprogram
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	1111 1**FED**	Set bits **F** - **D** = “0” to program, “1” to unprogram
Read Fuse Bits	0101 0000	0000 0000	xxxx xxxx	**CB**xx **A987**	Read Fuse bits. “0” = programmed, “1” = unprogrammed
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	xxxx 1**FED**	Read Fuse high bits. “0” = programmed, “1” = unprogrammed
Read Calibration Byte	0011 1000	xxxx xxxx	0000 0000	**oooo oooo**	Read Signature Byte **o** at address **b**.

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
1 = lock bit 1, ***2*** = lock bit 2, ***3*** = Boot Lock Bit01, ***4*** = Boot Lock Bit02, ***5*** = Boot Lock Bit11, ***6*** = Boot Lock Bit12, **7** = CKSEL0 Fuse, **8** = CKSEL1 Fuse, **9** = CKSEL2 Fuse, **A** = CKSEL3 Fuse, **B**= BODEN Fuse, **C**= BODLEVEL Fuse, **D**= BOOTRST Fuse, **E**= BOOTSZ0 Fuse, **F**= BOOTSZ1 Fuse

Serial Programming Characteristics

Figure 88. Serial Programming Timing

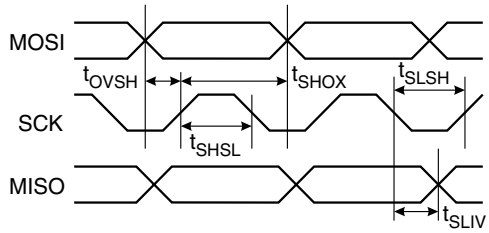


Table 62. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 2.7V - 5.5V (Unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency (V_{CC} = 2.7 - 5.5 V)	0		4	MHz
t_{CLCL}	Oscillator Period (V_{CC} = 2.7 - 5.5 V)	250			ns
$1/t_{CLCL}$	Oscillator Frequency (V_{CC} = 4.0 - 5.5 V)	0		8	MHz
t_{CLCL}	Oscillator Period (V_{CC} = 4.0 - 5.5 V)	125			ns
t_{SHSL}	SCK Pulse Width High	2 t_{CLCL}			ns
t_{SLSH}	SCK Pulse Width Low	2 t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	10	16	32	ns

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	-1.0V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +13.0V
Maximum Operating Voltage	6.6V
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)	-0.5		0.3 V_{CC}[1]	V
V_{IL1}	Input Low-voltage	(XTAL1), CKSEL3 fuse programmed	-0.5		0.3 V_{CC}[1]	V
		(XTAL1), CKSEL3 fuse unprogrammed	-0.5		0.2 V_{CC}[1]	V
V_{IH}	Input High-voltage	(Except XTAL1, $\overline{\text{RESET}}$)	0.6 V_{CC}[2]		V_{CC} + 0.5	V
V_{IH1}	Input High-voltage	(XTAL1), CKSEL3 fuse programmed	0.6 V_{CC}[2]		V_{CC} + 0.5	V
		(XTAL1), CKSEL3 fuse unprogrammed	0.8 V_{CC}[2]		V_{CC} + 0.5	V
V_{IH2}	Input High-voltage	($\overline{\text{RESET}}$)	0.9 V_{CC}[2]		V_{CC} + 0.5	V
V_{OL}	Output Low-voltage[3] (Ports A,B,C,D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.6 0.5	V V
V_{OH}	Output High-voltage[4] (Ports A,B,C,D)	I_{OH} = -3 mA, V_{CC} = 5V I_{OH} = -1.5 mA, V_{CC} = 3V	4.2 2.3			V V
I_{IL}	Input Leakage Current I/O pin	Vcc = 5.5V, pin low (absolute value)			8.0	µA
I_{IH}	Input Leakage Current I/O pin	Vcc = 5.5V, pin high (absolute value)			980	nA
RRST	Reset Pull-up Resistor		100		500	kΩ
$R_{I/O}$	I/O Pin Pull-up Resistor		35		120	kΩ

DC Characteristics (Continued)

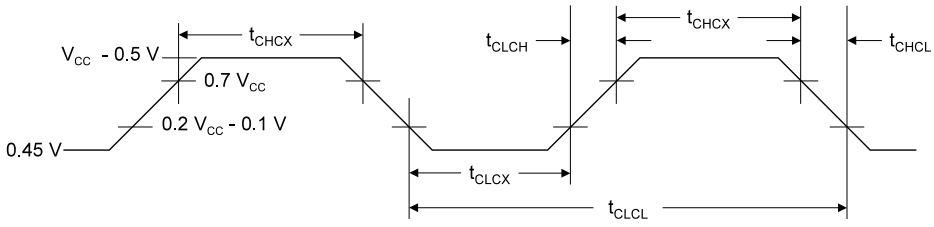
T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
I_{CC}	Power Supply Current	Active 4 MHz, V_{CC} = 3V (ATmega163L)			5.0	mA
		Active 8 MHz, V_{CC} = 5V (ATmega163)			15.0	mA
		Idle 4 MHz, V_{CC} = 3V (ATmega163L)			2.5	mA
		Idle 8 MHz, V_{CC} = 5V (ATmega163)			8	mA
	Power-down mode(5)	WDT enabled, V_{CC} = 3V		9	15.0	µA
		WDT disabled, V_{CC} = 3V		<1	4.0	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_{ACID}	Analog Comparator Initialization 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 Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all I_{OL}, for all ports, should not exceed 200 mA.
   2] The sum of all I_{OL}, for ports B0 - B7, D0 - D7 and XTAL2, should not exceed 100 mA.
   3] The sum of all I_{OL}, for ports A0 - A7 and C0 - C7 should not exceed 100 mA.
   If IOL exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (3 mA at Vcc = 5V, 1.5 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all I_{OH}, for all ports, should not exceed 200 mA.
   2] The sum of all I_{OH}, for ports B0 - B7, D0 - D7 and XTAL2, should not exceed 100 mA.
   3] The sum of all I_{OH}, for ports A0 - A7 and C0 - C7 should not exceed 100 mA.
   If IOH exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum V_{CC} for Power-down is 2.5V.

External Clock Drive Waveforms

Figure 89. External Clock Drive Waveforms



External Clock Drive

Table 63. External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 5.5V		V_{CC} = 4.0V to 5.5V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	8	MHz
t_{CLCL}	Clock Period	250		125		ns
t_{CHCX}	High Time	100		50		ns
t_{CLCX}	Low Time	100		50		ns
t_{CLCH}	Rise Time		1.6		0.5	µs
t_{CHCL}	Fall Time		1.6		0.5	µs

Table 64. External RC Oscillator, typical frequencies

R [kΩ]	C [pF]	f
100	70	100 kHz
31.5	20	1.0 MHz
6.5	20	4.0 MHz

Note: R should be in the range 3kΩ - 100kΩ, and C should be at least 20pF. The C values given in the table includes pin capacitance. This will vary with package type.

Two-wire Serial Interface Characteristics

Table 65 describes the requirements for devices connected to the Two-wire Serial Bus. The ATmega163 Two-wire Serial Interface meets or exceeds these requirements under the noted conditions.

Timing symbols refer to Figure 90.

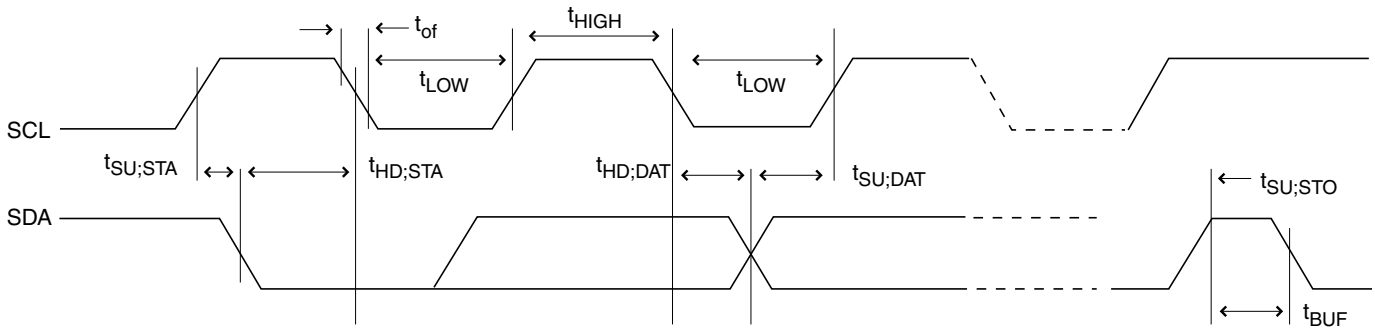
Table 65. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.3 V_{CC}	V
V_{IH}	Input High-voltage		0.7 V_{CC}	V_{CC} + 0.5	V
V_{hys}(1)	Hysteresis of Schmitt Trigger Inputs		0.05 V_{CC}(2)	–	V
V_{OL}(1)	Output Low-voltage	3 mA sink current	0	0.4	V
t_{of}(1)	Output Fall Time from V_{IHmin} to V_{ILmax}	10 pF < C_b < 400 pF(3)	20 + 0.1C_b(3)(2)	250	ns
t_{SP}(1)	Spikes Suppressed by Input Filter		0	50(2)	ns
I_i	Input Current each I/O Pin	0.1V_{CC} < V_i < 0.9V_{CC}	-10	10	µA
C_i(1)	Capacitance for each I/O Pin		–	10	pF
f_{SCL}	SCL Clock Frequency	f_{CK}(4) > max(16f_{SCL}, 250kHz)(5)	0	400	kHz
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{SCL} ≤ 100 kHz	4.0	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
t_{LOW}	Low Period of the SCL Clock	f_{SCL} ≤ 100 kHz(6)	4.7	–	µs
		f_{SCL} > 100 kHz(7)	1.3	–	µs
t_{HIGH}	High period of the SCL clock	f_{SCL} ≤ 100 kHz	4.0	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
$t_{SU;STA}$	Set–up time for a repeated START condition	f_{SCL} ≤ 100 kHz	4.7	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
$t_{HD;DAT}$	Data hold time	f_{SCL} ≤ 100 kHz	0	3.45	µs
		f_{SCL} > 100 kHz	0	0.9	µs
$t_{SU;DAT}$	Data setup time	f_{SCL} ≤ 100 kHz	250	–	ns
		f_{SCL} > 100 kHz	100	–	ns
$t_{SU;STO}$	Setup time for STOP condition	f_{SCL} ≤ 100 kHz	4.0	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
t_{BUF}	Bus free time between a STOP and START condition	f_{SCL} ≤ 100 kHz	4.7	–	µs
		f_{SCL} > 100 kHz	1.3	–	µs

Notes:
1. In ATmegan163, this parameter is characterized and not 100% tested.
2. Required only for f_{SCL} > 100 kHz.
3. C_b = capacitance of one bus line in pF.
4. f_{CK} = CPU clock frequency
5. This requirement applies to all ATmega163 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general f_{SCL} requirement.
6. The actual low period generated by the ATmega163 Two-wire Serial Interface is (1/f_{SCL} - 2/f_{CK}), thus f_{CK} must be greater than 6 MHz for the low time requirement to be strictly met at f_{SCL} = 100 kHz.
7. The actual low period generated by the ATmega163 Two-wire Serial Interface is (1/f_{SCL} - 2/f_{CK}), thus the low time requirement will not be strictly met for f_{SCL} > 308 kHz when f_{CK} = 8 MHz. Still, ATmega163 devices connected to the bus may

communicate at full speed (400 kHz) with other ATmega163 devices, as well as any other device with a proper t_{LOW} acceptance margin.

Figure 90. Two-wire Serial Bus Timing



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. All pins on Port F are pulled high externally. 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 $C_L \bullet V_{CC} \bullet f$, 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.

Figure 91. Analog Comparator Offset Voltage vs, Common Mode Voltage (V_{CC} = 5V)

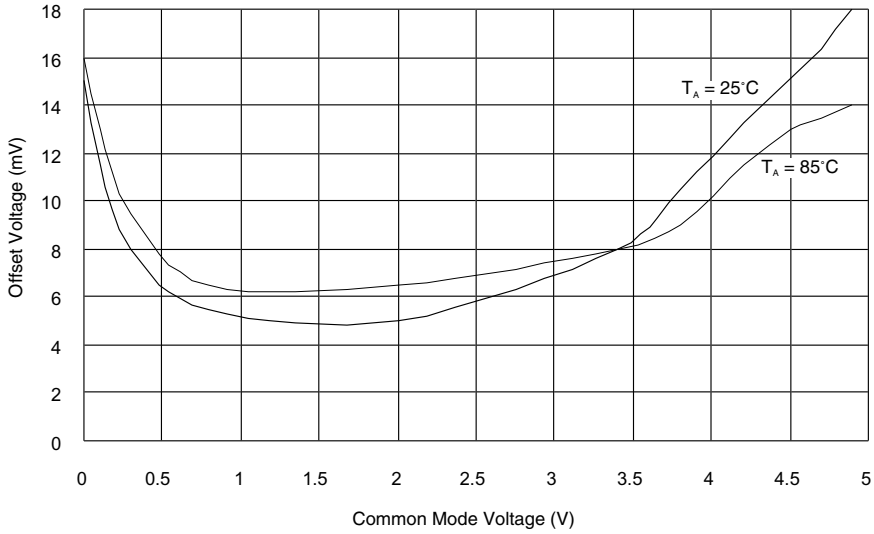


Figure 92. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC} = 2.7V)

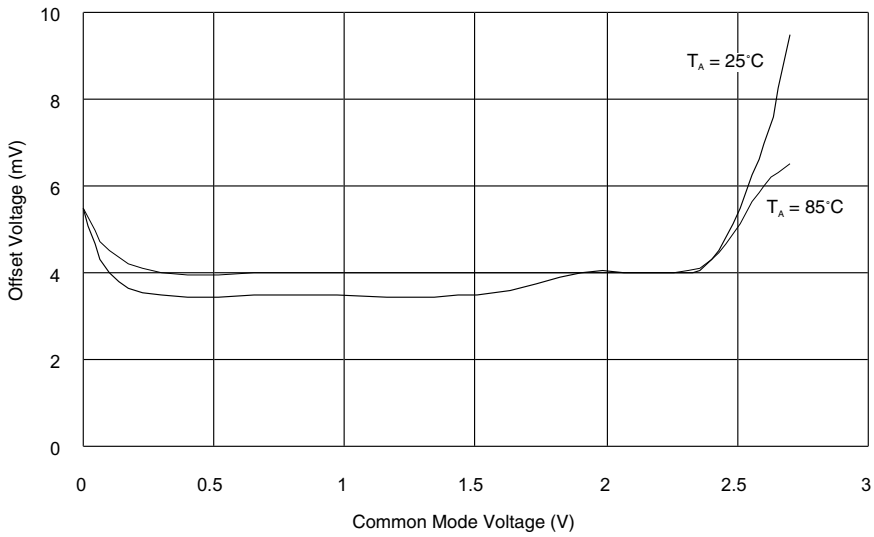


Figure 93. Analog Comparator Input Leakage Current (V_{CC} = 6V; TA = 25°C)

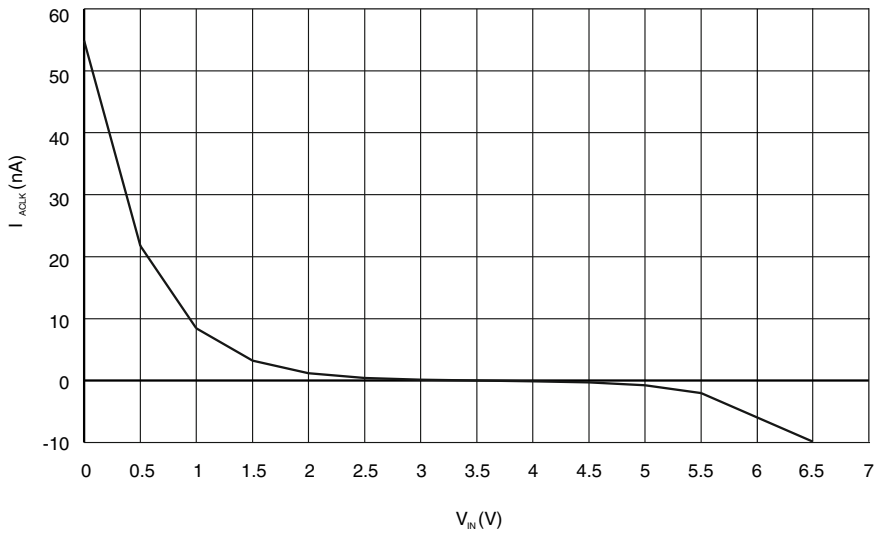
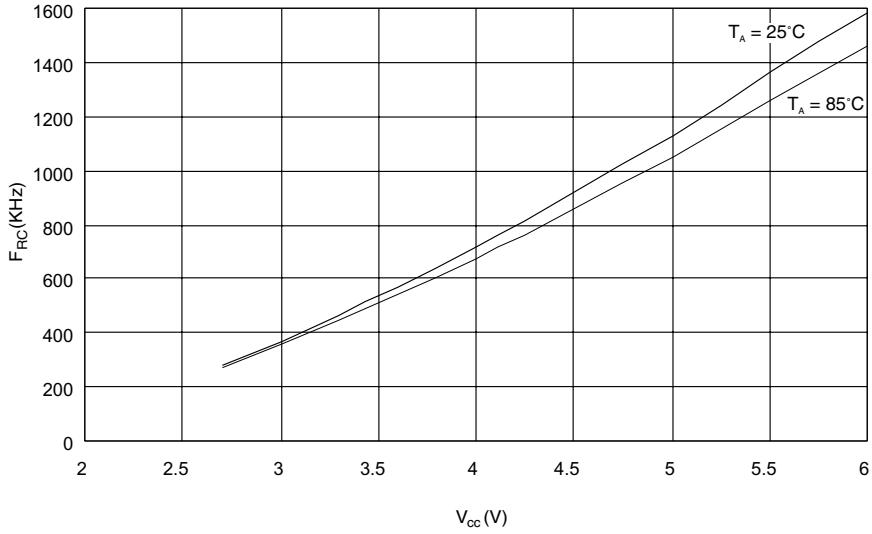


Figure 94. Watchdog Oscillator Frequency vs. V_{CC}



Sink and source capabilities of I/O ports are measured on one pin at a time.

Figure 95. Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

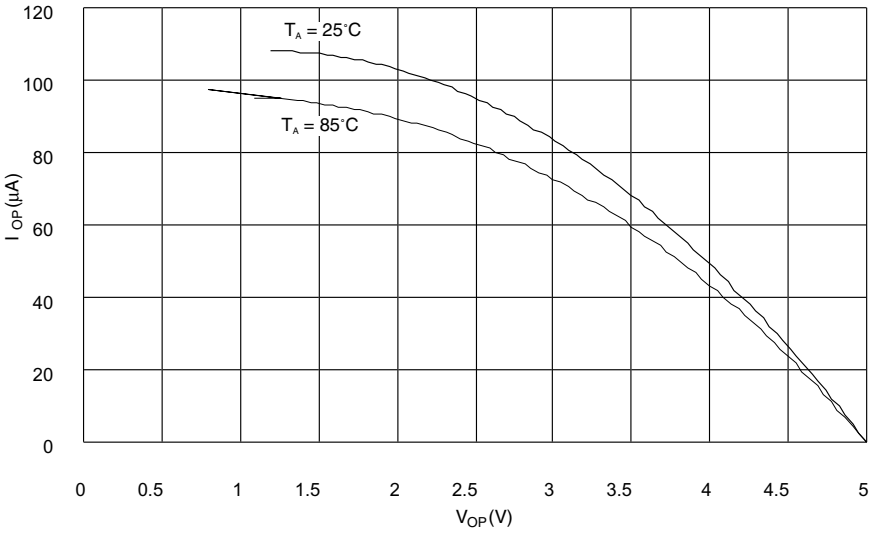


Figure 96. Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

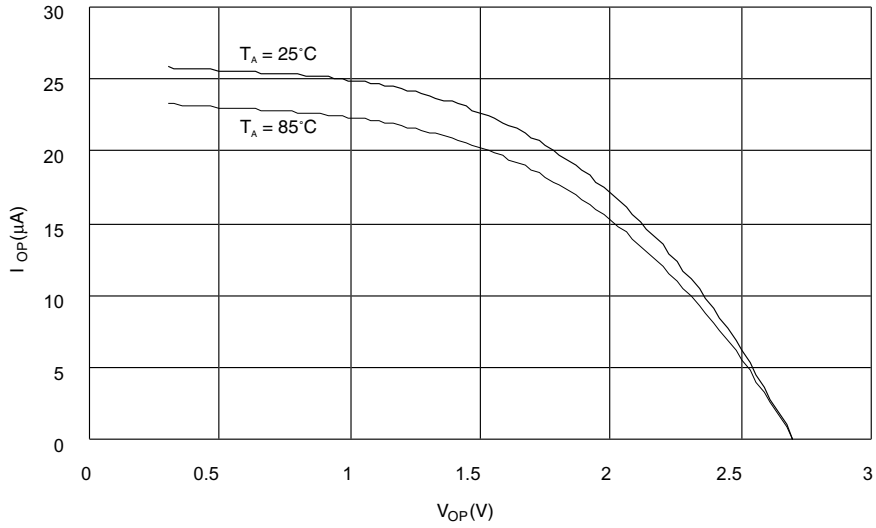


Figure 97. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

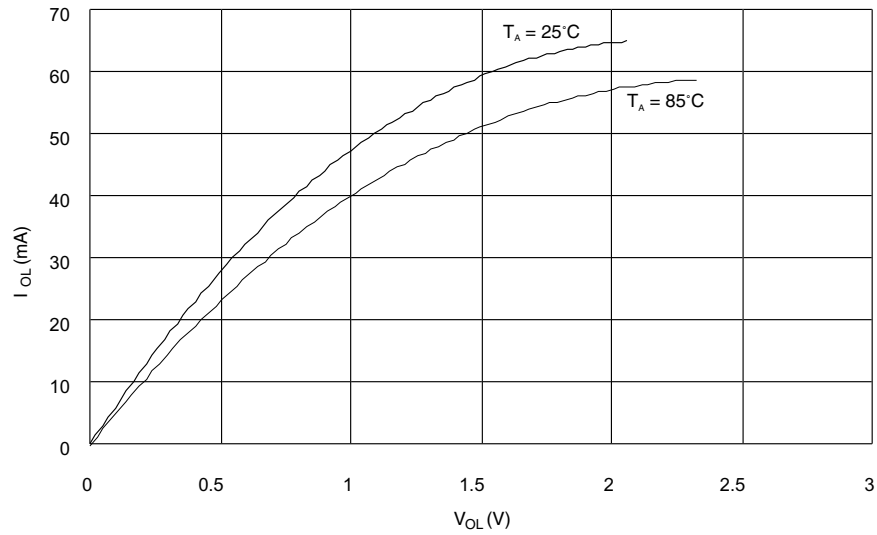


Figure 98. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

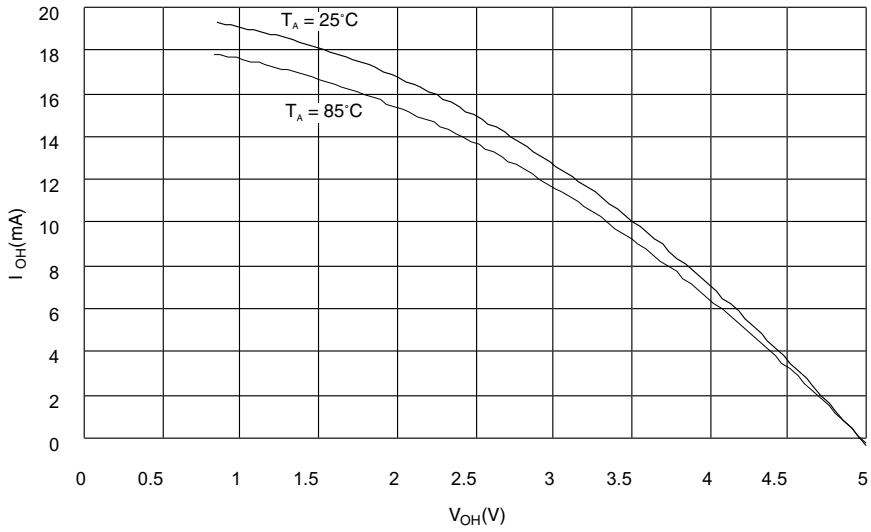


Figure 99. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)

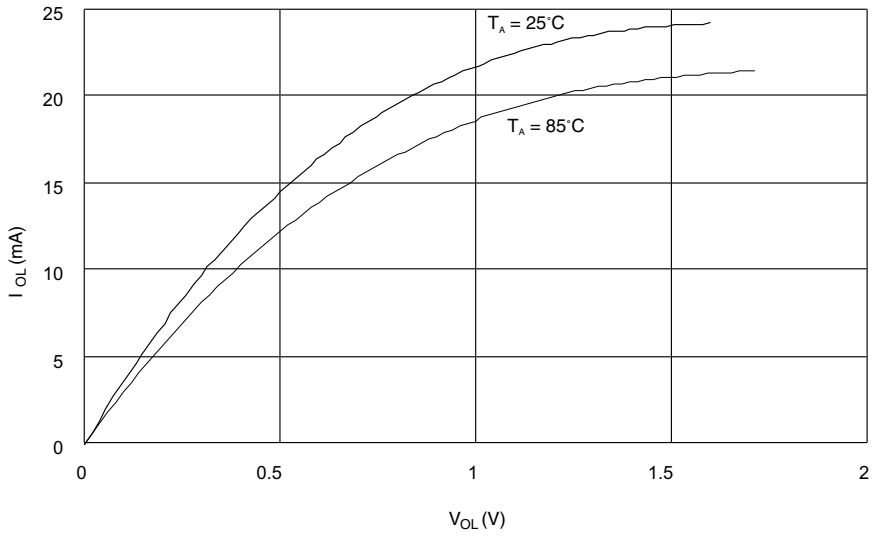


Figure 100. I/O Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

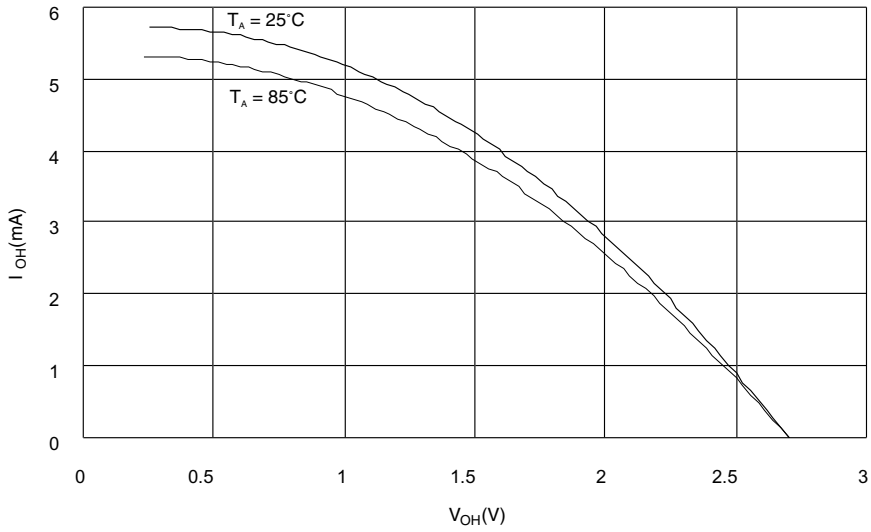


Figure 101. I/O Pin Input Threshold vs. V_{CC} (T_A = 25°C)

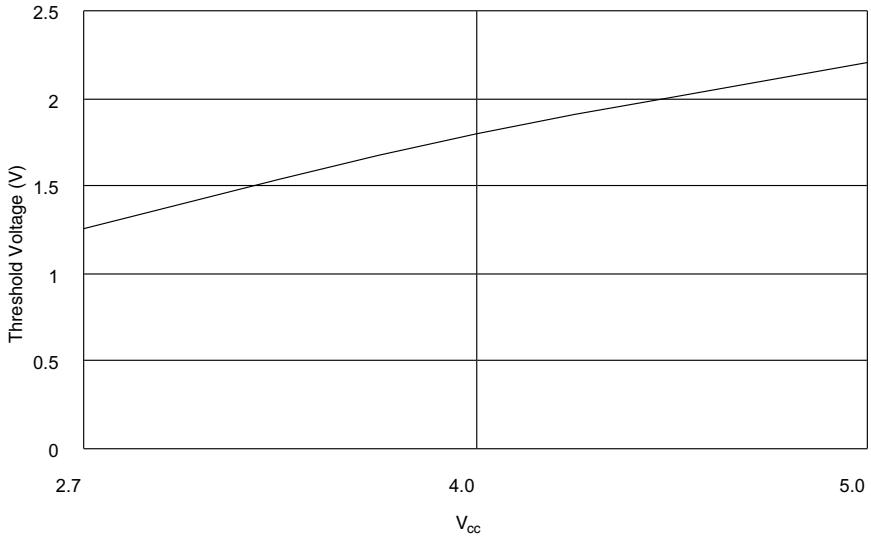
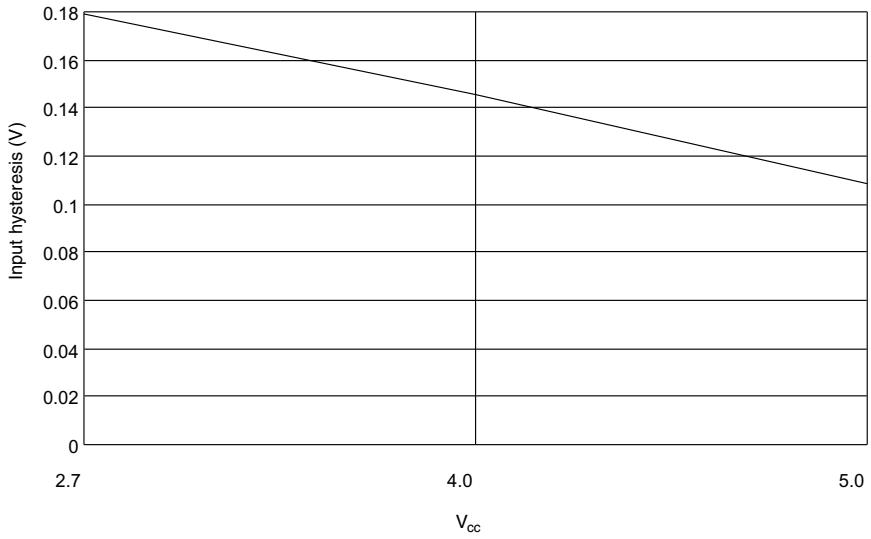


Figure 102. I/O Pin Input Hysteresis vs. V_{CC} (T_A = 25°C)



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	20
$3E ($5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	21
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	21
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	INT1	INT0	–	–	–	–	–	–	30
$3A ($5A)	GIFR	INTF1	INTF0	–	–	–	–	–	–	31
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	32
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	32
$37 ($57)	SPMCR	–	ASB	–	ASRE	BLBSET	PGWRT	PGERS	SPMEN	140
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	82
$35 ($55)	MCUCR	–	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	34
$34 ($54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	28
$33 ($53)	TCCR0	–	–	–	–	–	CS02	CS01	CS00	41
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								42
$31 ($51)	OSCCAL	Oscillator Calibration Register								37
$30 ($50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	40
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM11	PWM10	44
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	–	CTC1	CS12	CS11	CS10	45
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								46
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								46
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								47
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								47
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								47
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								47
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								48
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								48
$25 ($45)	TCCR2	FOC2	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	52
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								53
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								54
$22 ($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	57
$21 ($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	60
$20 ($40)	UBRRHI	–	–	–	–	UBRR[11:8]				78
$1F ($3F)	EEARH	–	–	–	–	–	–	–	EEAR8	62
$1E ($3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	62
$1D ($3D)	EEDR	EEPROM Data Register								62
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	63
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	115
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	115
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	115
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	117
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	117
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	117
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	123
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	123
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	123
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	128
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	128
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	128
$0F ($2F)	SPDR	SPI Data Register								69
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	68
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	67
$0C ($2C)	UDR	UART I/O Data Register								74
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	OR	–	U2X	MPCM	74
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	76
$09 ($29)	UBRR	UART Baud Rate Register								78
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	102
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	110
$06 ($26)	ADCSR	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	111
$05 ($25)	ADCH	ADC Data Register High Byte								112
$04 ($24)	ADCL	ADC Data Register Low Byte								112
$03 ($23)	TWDR	Two-wire Serial Interface Data Register								84
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	85
$01 ($21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	–	–	84

Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$00 ($20)	TWBR	Two-wire Serial Interface Bit Rate Register								82

Note: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

2. 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
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
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

Instruction Set Summary (Continued)

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
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

Instruction Set Summary (Continued)

CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
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

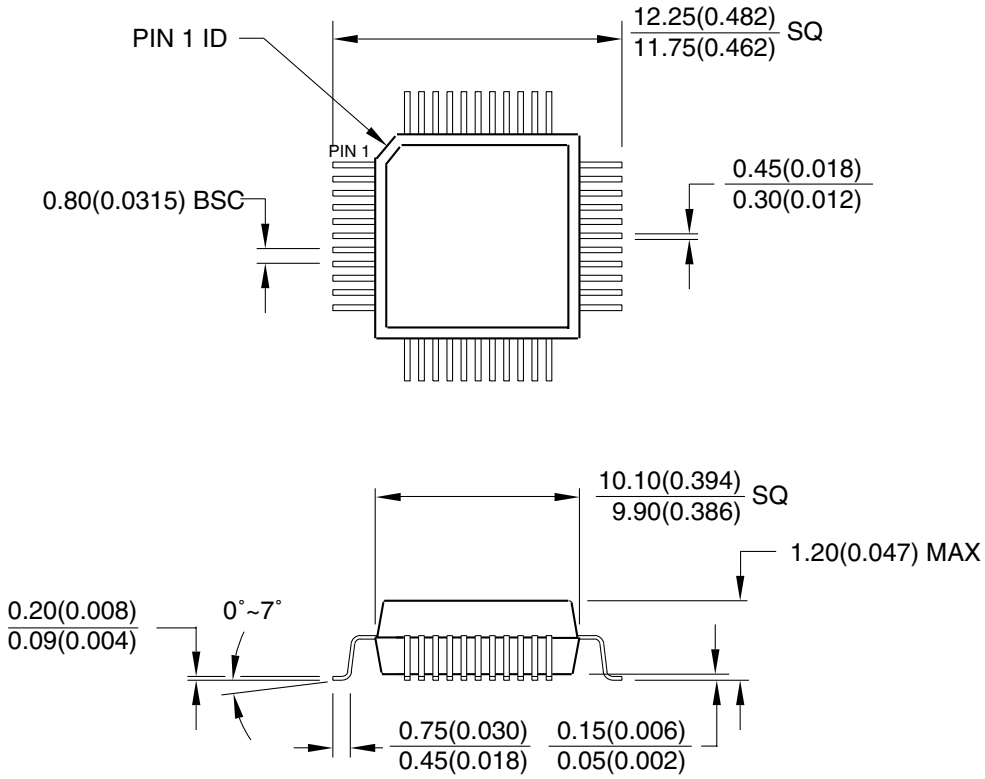
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
4	2.7 - 5.5V	ATmega163L-4AC ATmega163L-4PC	44A 40P6	Commercial (0°C to 70°C)
		ATmega163L-4AI ATmega163L-4PI	44A 40P6	Industrial (-40°C to 85°C)
8	4.0 - 5.5V	ATmega163-8AC ATmega163-8PC	44A 40P6	Commercial (0°C to 70°C)
		ATmega163-8AI ATmega163-8PI	44A 40P6	Industrial (-40°C to 85°C)

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)

Packaging Information

44A

44-lead, Thin (1.0mm) Plastic Quad Flat Package
(TQFP), 10x10mm body, 2.0mm footprint, 0.8mm pitch.
Dimension in Millimeters and (Inches)*
JEDEC STANDARD MS-026 ACB

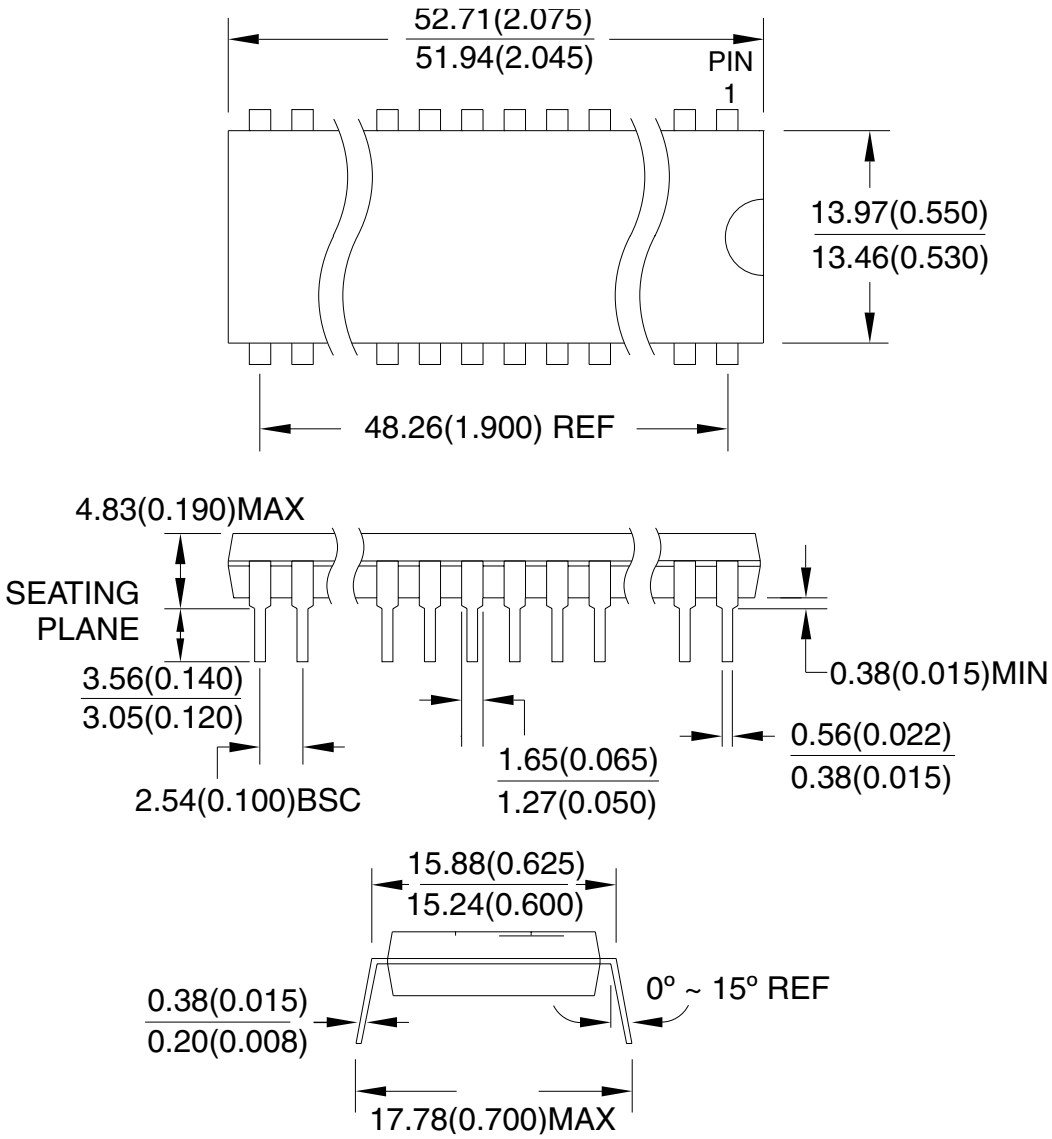


*Controlling dimension: millimetter

REV. A 04/11/2001

40P6

40-lead, Plastic Dual Inline
Parkage (PDIP), 0.600" wide
Demension in Millimeters and (Inches)*
JEDEC STANDARD MS-011 AC



*Controlling dimension: Inches

REV. A 04/11/2001

Change Log

This section containes a log on the changes made to the data sheet for ATmega163. All refereces to pages in Change Log, are referred to this document.

Changes from Rev. 1142C-09/01 to Ref. 1142D-09/02

1 Added “Not Recommend for New Designs. Use ATmega16.”.

Table of Contents

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

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

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

1142D–AVR–09/02 0M