Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 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
  - 32K Bytes of In-System Self-programmable Flash
    Endurance: 1,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
  - 1K Byte EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 2K Bytes Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE Std. 1149.1 Compliant) Interface
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
  - Boundary-scan Capabilities According to the JTAG Standard
- 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 Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
  - Byte-oriented 2-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP and 44-lead TQFP
- Operating Voltages
  - 2.7 - 5.5V (ATmega323L)
  - 4.0 - 5.5V (ATmega323)
- Speed Grades
  - 0 - 4 MHz (ATmega323L)
  - 0 - 8 MHz (ATmega323)

8-bit AVR® Microcontroller with 32K Bytes of In-System Programmable Flash

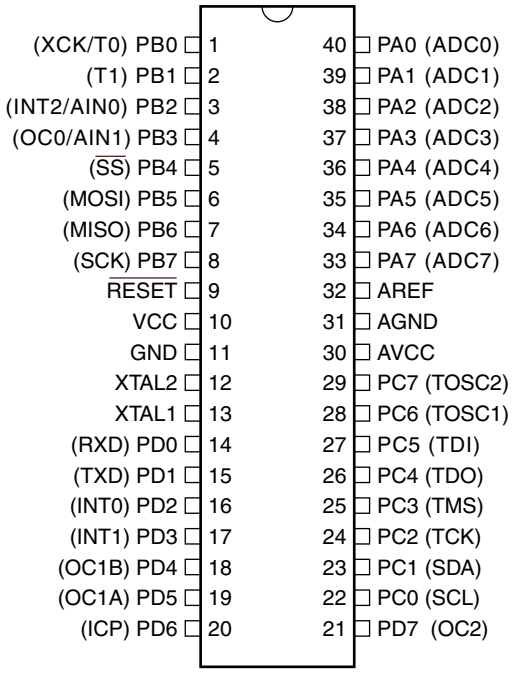
ATmega323
ATmega323L

Preliminary

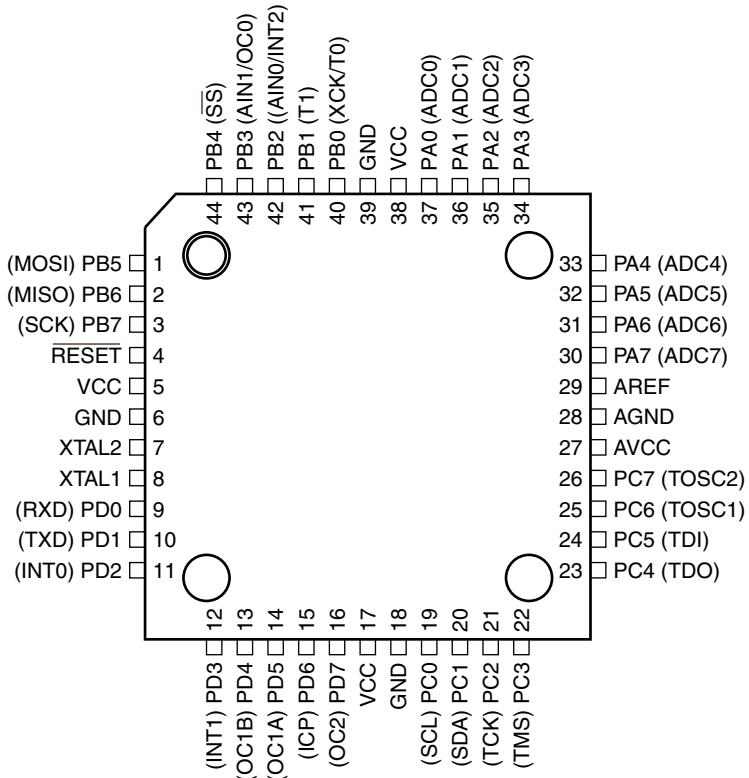
Rev. 1457E–11/01

Pin Configurations

PDIP



TQFP

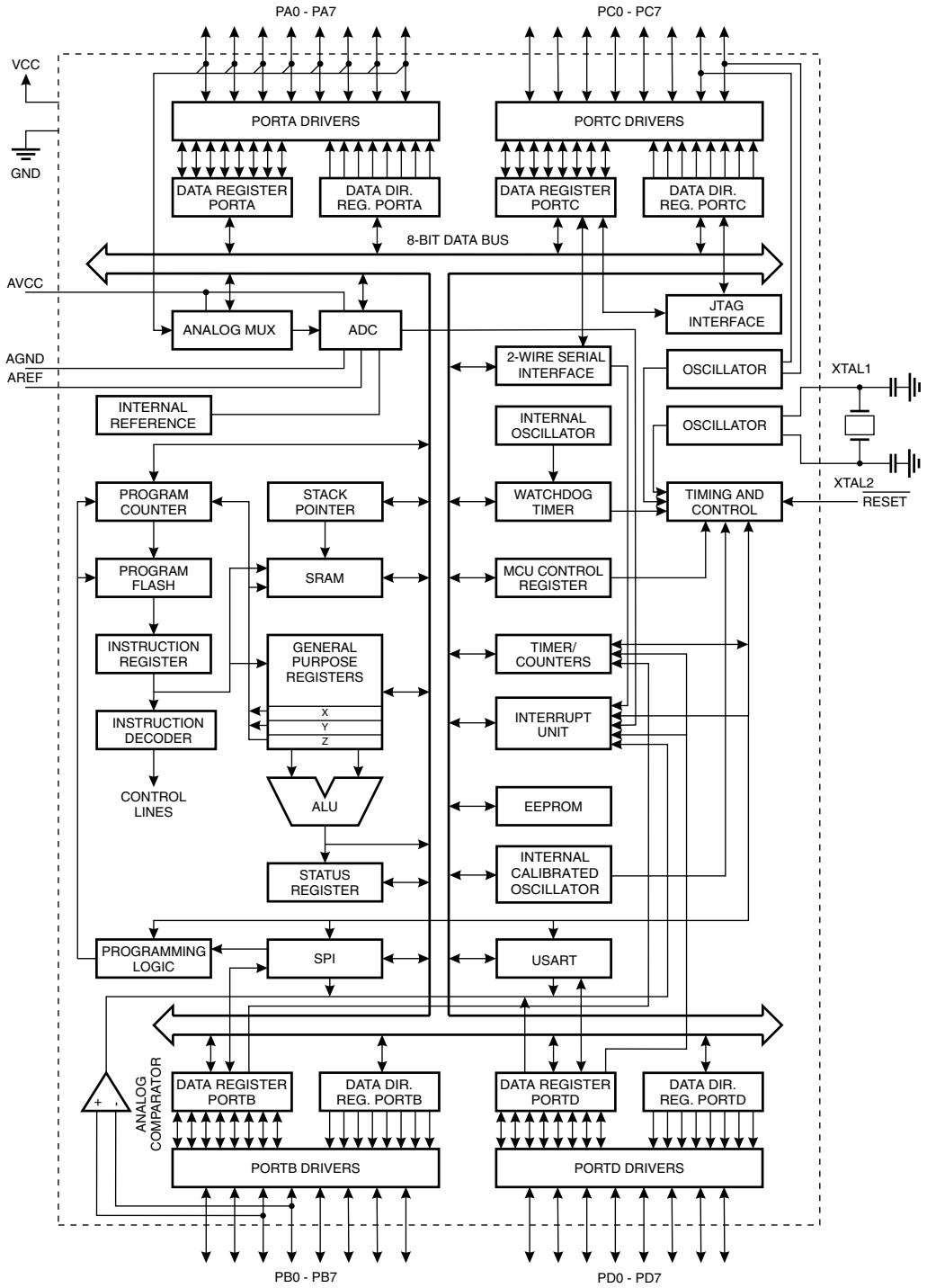


Overview

The ATmega323 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega323 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 ATmega323 provides the following features: 32K bytes of In-System Programmable Flash, 1K bytes EEPROM, 2K bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-Scan, On-Chip Debugging support and programming, three flexible timer/counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented 2-wire Serial Interface, an 8-channel, 10-bit ADC, a programmable Watchdog Timer with internal oscillator, an SPI serial port, and six 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 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. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. By combining an 8-bit RISC CPU with In-System Programmable Flash on a monolithic chip, the Atmel ATmega323 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega323 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 20 mA 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 tri-stated 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. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega323 as listed on page 135.

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 tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of the JTAG interface and other special features of the ATmega323 as listed on page 142. If the JTAG interface is enabled, the pull-up resistors on pins PC5 (TDI), PC3 (TMS) and PC2 (TCK) will be activated even if a reset occurs.

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. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega323 as listed on page 147.

$\overline{\text{RESET}}$ 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 AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. See page 124 for details on operation of the ADC.

AREF AREF is the analog reference pin for the A/D Converter. For ADC operations, a voltage in the range 2.56V 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

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" means unprogrammed, "0" means programmed.

The various choices for each clocking option give different start-up times as shown in Table 6 on page 27.

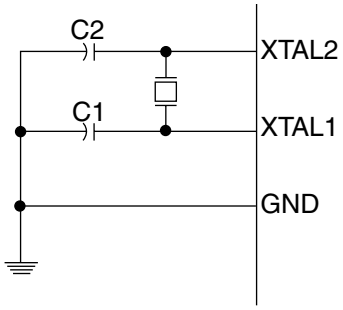
Internal RC Oscillator

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. See "Calibrated Internal RC Oscillator" on page 40 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.

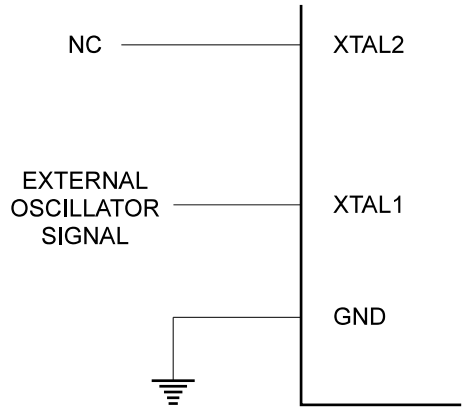
Figure 2. Oscillator Connections



External Clock

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

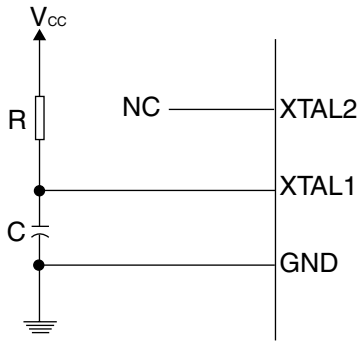
Figure 3. External Clock Drive Configuration



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 73 on page 209.

Figure 4. External RC Configuration



Timer Oscillator

For the Timer Oscillator pins, PC6(TOSC1) and PC7(TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock source to PC6(TOSC1) 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.

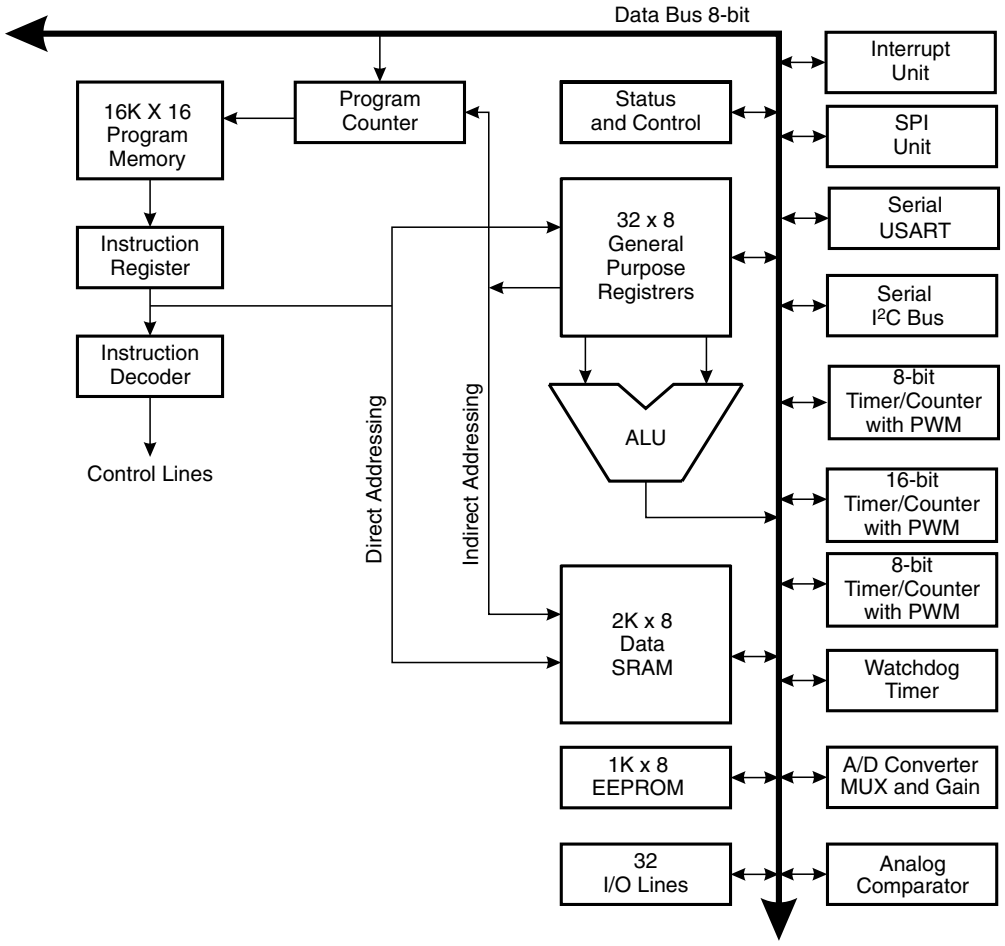
Six of the 32 registers can be used as three 16-bits indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the three address pointers is also used as the address pointer for look up tables in Flash program memory. These added function registers 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 ATmega323 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 ATmega323 AVR Enhanced 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 single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

With the jump and call instructions, the whole 16K 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 (512 to 4K bytes, see page 172) 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.

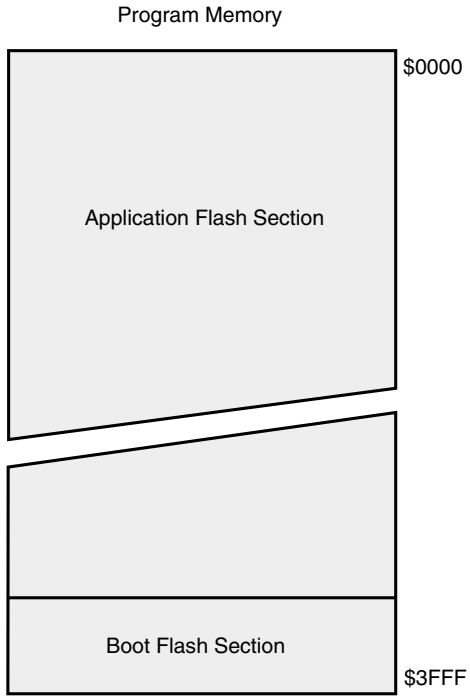
During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 12-bit stack pointer SP is read/write accessible in the I/O space.

The 2K 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. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

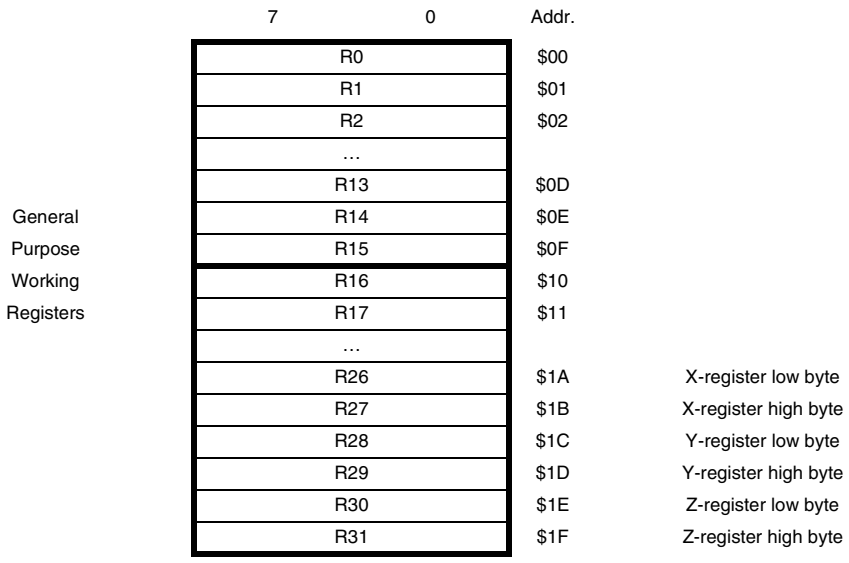
Figure 6. Memory Maps



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



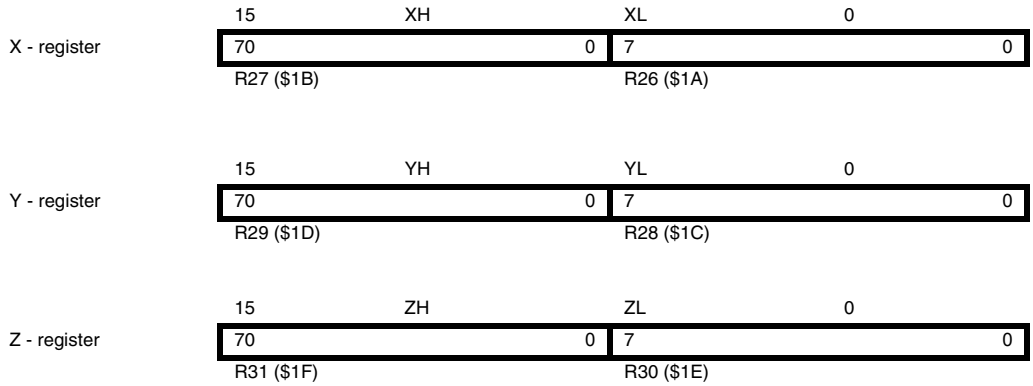
Most register operating instructions in the instruction set have direct access to all registers, and most of them are single cycle instructions.

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. ATmega323 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 Reprogrammable Flash Program Memory

The ATmega323 contains 32K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 16K 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 1000 write/erase cycles. The ATmega323 Program Counter (PC) is 14 bits wide, thus addressing the 16K program memory locations. The operation of Boot Program section and associated Boot Lock Bits for software protection are described in detail on page 172. See page 191 for a detailed description on Flash data serial downloading using the SPI pins. See page 196 for details on serial downloading using the JTAG interface.

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

See also page 13 for the different program memory addressing modes.

The SRAM Data Memory

Figure 9 shows how the ATmega323 SRAM Memory is organized.

The lower 2144 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 2048 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.

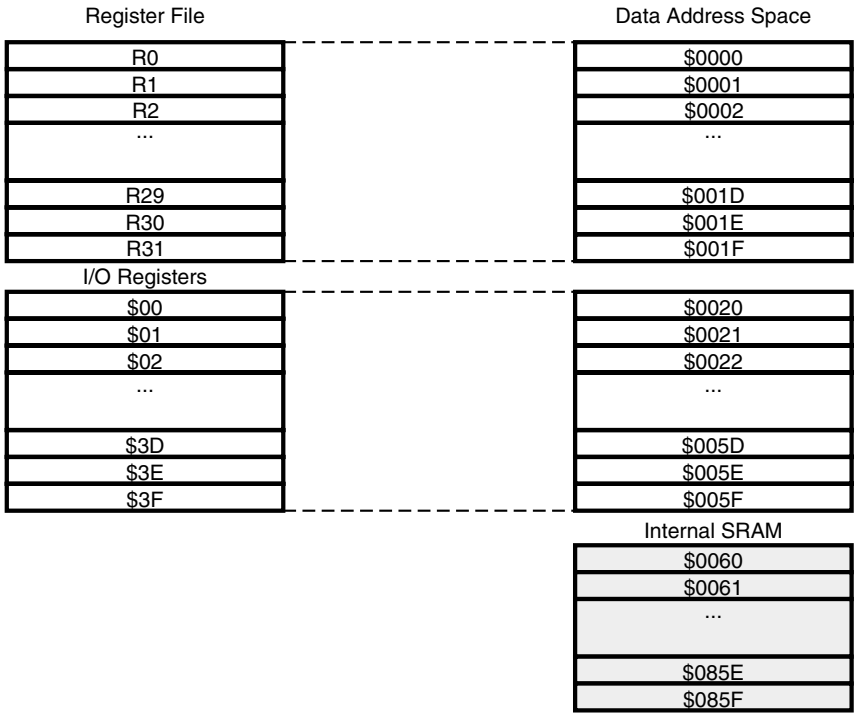
The direct addressing reaches the entire data space.

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 2048 bytes of internal data SRAM in the ATmega323 are all accessible through all these addressing modes.

Figure 9. SRAM Organization

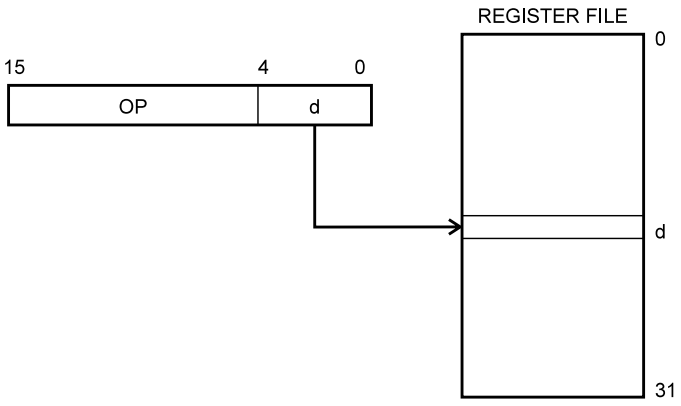


The Program and Data Addressing Modes

The ATmega323 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

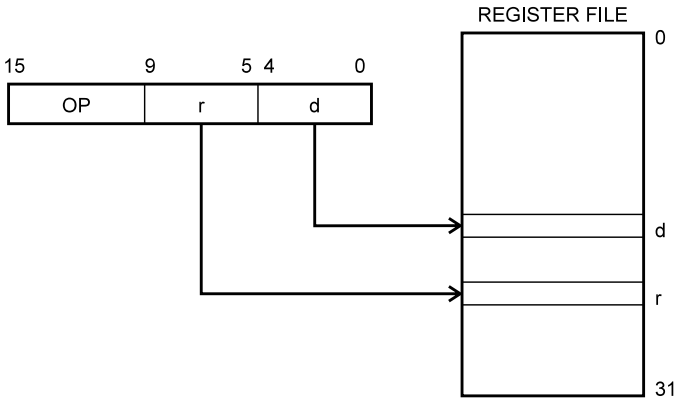
Figure 10. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

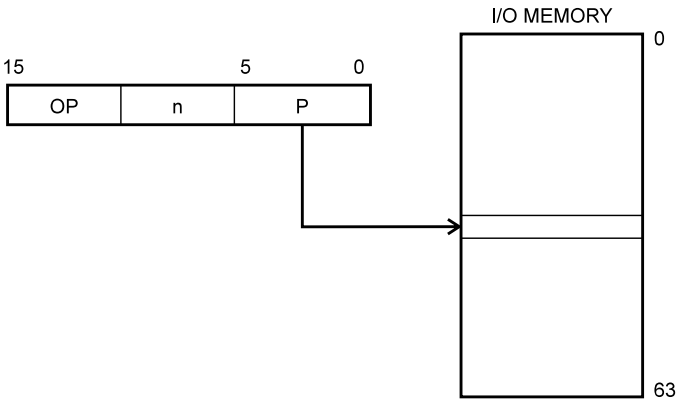
Figure 11. Direct Register Addressing, Two Registers



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

I/O Direct

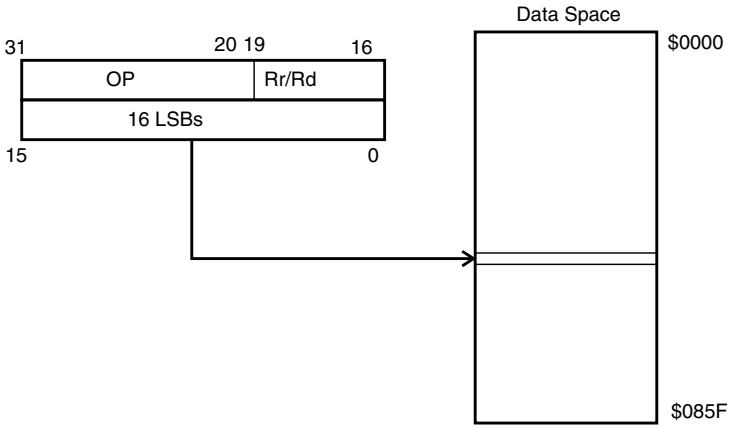
Figure 12. I/O Direct Addressing



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

Data Direct

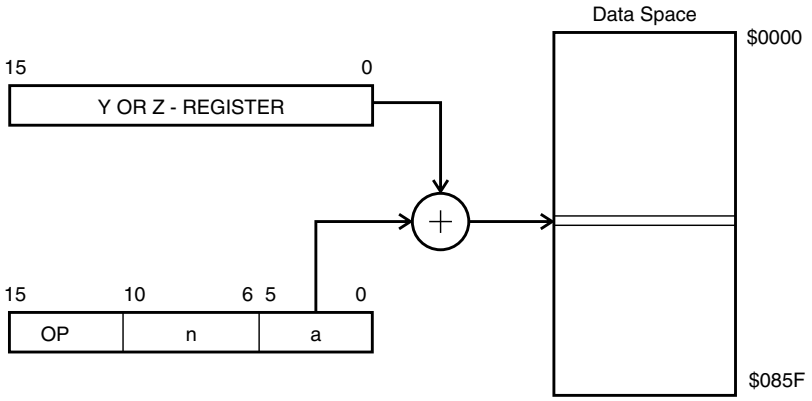
Figure 13. Direct Data Addressing



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

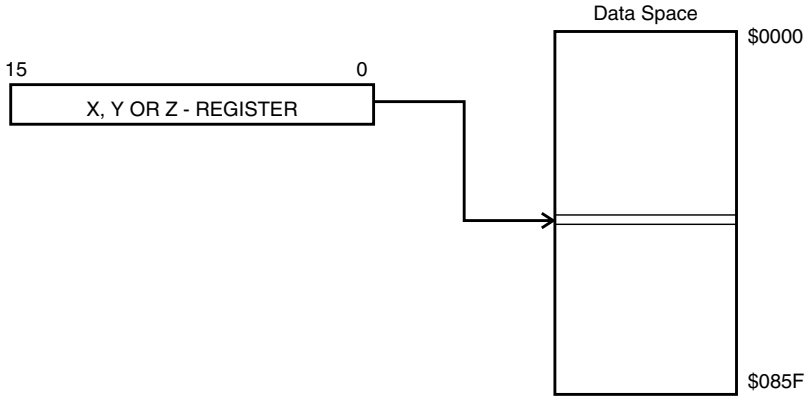
Figure 14. Data Indirect with Displacement



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

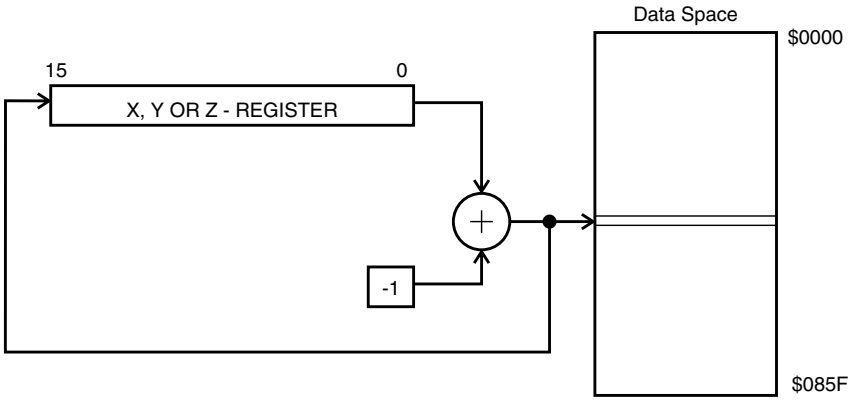
Figure 15. Data Indirect Addressing



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

Data Indirect with Pre--decrement

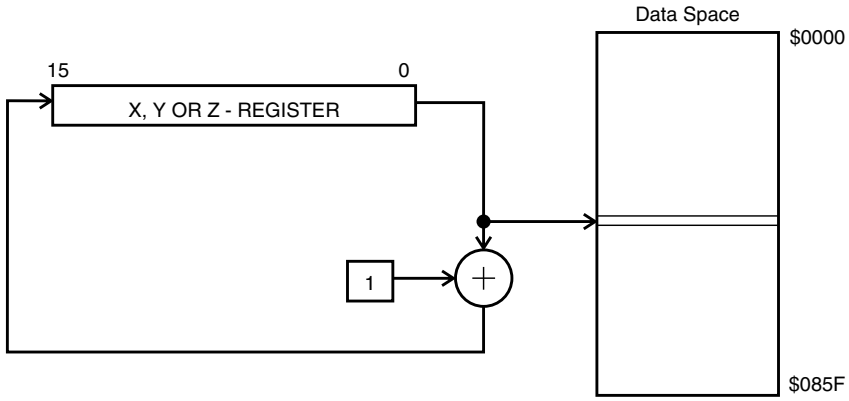
Figure 16. Data Indirect Addressing With Pre-decrement



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

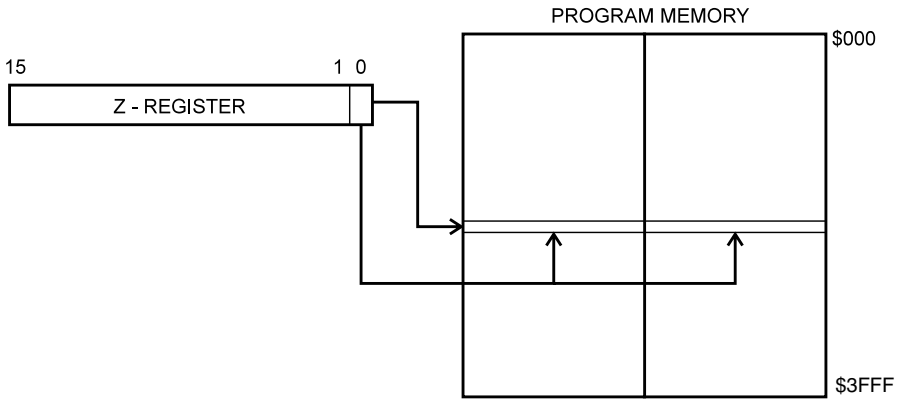
Figure 17. Data Indirect Addressing With Post-increment



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

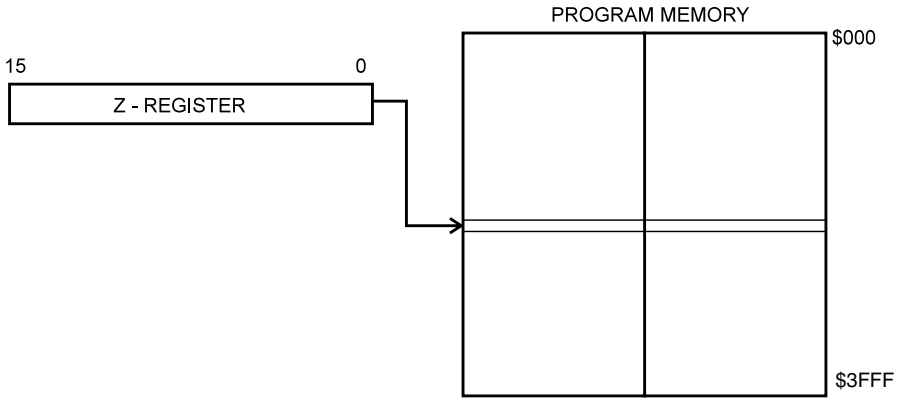
Figure 18. Code Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 16K). 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

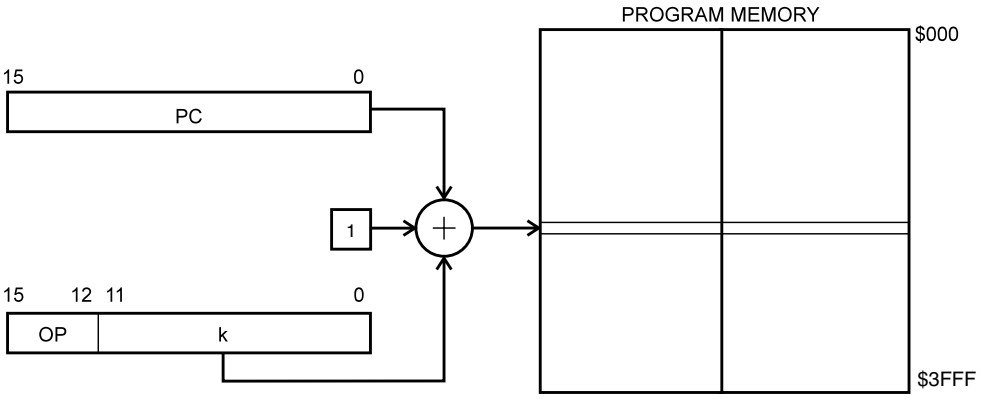
Figure 19. Indirect Program Memory Addressing



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 –2048 to 2047.

The EEPROM Data Memory

The ATmega323 contains 1K 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 65 specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For SPI data downloading of the EEPROM, see page 191 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 selected clock source 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

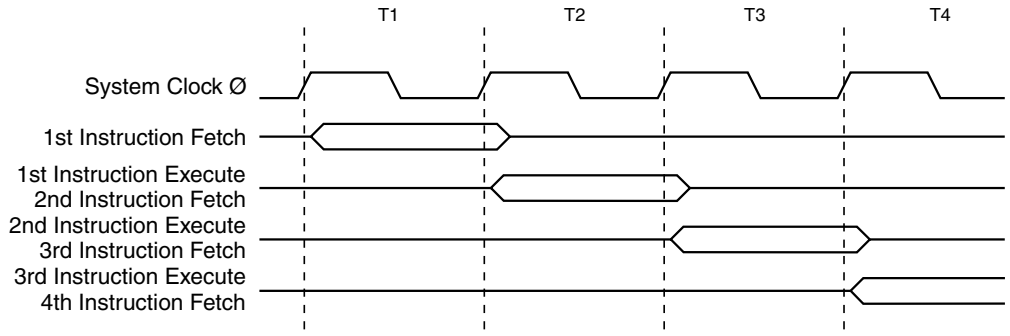
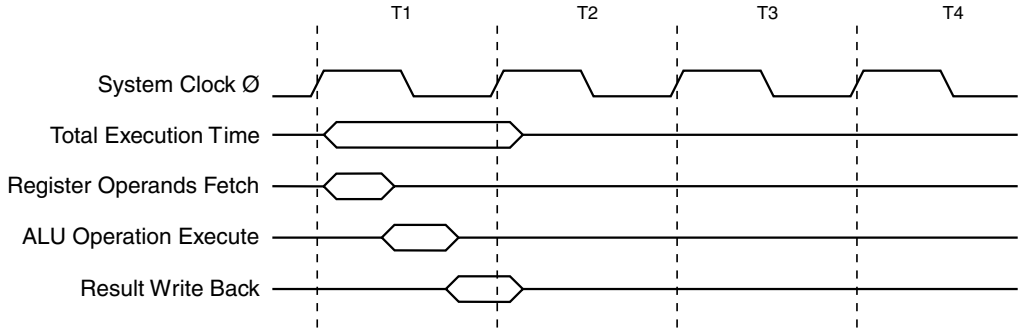


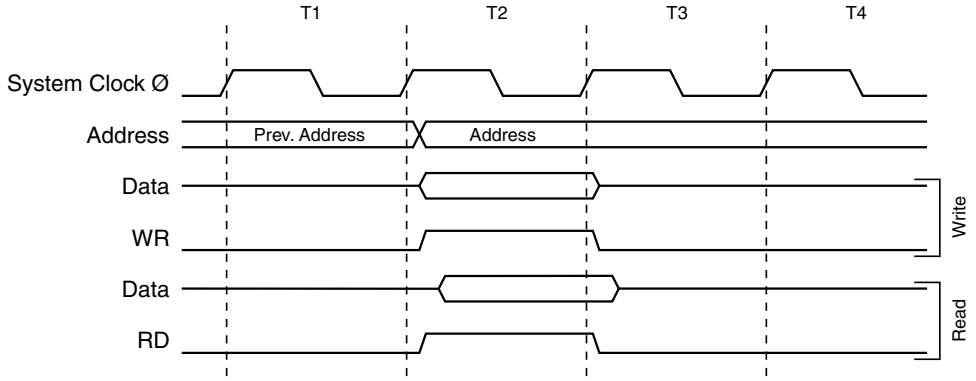
Figure 22 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

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 ATmega323 is shown in Table 2.

Table 2. ATmega323 I/O Space

I/O Address (SRAM Address)	Name	Function
$3F ($5F)	SREG	Status Register
$3E ($5E)	SPH	Stack Pointer High
$3D ($5D)	SPL	Stack Pointer Low
$3C ($3C)	OCR0	Timer/Counter0 Output Compare Register
$3B ($5B)	GICR	General Interrupt Control 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

Table 2. ATmega323 I/O Space (Continued)

I/O Address (SRAM Address)	Name	Function
$36 ($56)	TWCR	2-wire Serial Interface Control Register
$35 ($55)	MCUCR	MCU general Control Register
$34 ($54)	MCUCSR	MCU general Control and Status Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$31 ($51)[1]	OSCCAL	Oscillator Calibration Register
	OCDR	On-chip Debug 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)[2]	UBRRH	USART Baud Rate Register High Byte
	UCSRC	USART Control and Status Register C
$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

Table 2. ATmega323 I/O Space (Continued)

I/O Address (SRAM Address)	Name	Function
$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
$0F ($2F)	SPDR	SPI I/O Data Register
$0E ($2E)	SPSR	SPI Status Register
$0D ($2D)	SPCR	SPI Control Register
$0C ($2C)	UDR	USART I/O Data Register
$0B ($2B)	UCSRA	USART Control and Status Register A
$0A ($2A)	UCSRB	USART Control and Status Register B
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte
$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	2-wire Serial Interface Data Register
$02 ($22)	TWAR	2-wire Serial Interface (Slave) Address Register
$01 ($21)	TWSR	2-wire Serial Interface Status Register
$00 ($20)	TWBR	2-wire Serial Interface Bit Rate Register

Notes: 1. When the OCDEN fuse is unprogrammed, the OSCCAL register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR register.
2. Refer to the USART description for details on how to access UBRRH and UCSRC.

All ATmega323 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 separate control registers. If the global interrupt enable register is cleared (zero), none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts.

- **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 ATmega323 Stack Pointer is implemented as two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the ATmega323 data memory has $860 locations, 12 bits are used.

Bit	15	14	13	12	11	10	9	8	
$3E ($5E)	-	-	-	-	SP11	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/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 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 ATmega323 provides nineteen 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. Depending on the program counter value, interrupts may be disabled when Boot Lock bits BLB02 or BLB12 are set. See the section “Boot Loader Support” on page 172 for details

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. The interrupt vectors can be moved to the start of the boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). See the GICR description on page 32 for details..

Table 3. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	INT2	External Interrupt Request 2
5	$008	TIMER2 COMP	Timer/Counter2 Compare Match
6	$00A	TIMER2 OVF	Timer/Counter2 Overflow
7	$00C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	$00E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	$010	TIMER1 COMPB	Timer/Counter1 Compare Match B

Table 3. Reset and Interrupt Vectors (Continued)

Vector No.	Program Address(2)	Source	Interrupt Definition
10	$012	TIMER1 OVF	Timer/Counter1 Overflow
11	$014	TIMER0 COMP	Timer/Counter0 Compare Match
12	$016	TIMER0 OVF	Timer/Counter0 Overflow
13	$018	SPI, STC	Serial Transfer Complete
14	$01A	USART, RXC	USART, Rx Complete
15	$01C	USART, UDRE	USART Data Register Empty
16	$01E	USART, TXC	USART, Tx Complete
17	$020	ADC	ADC Conversion Complete
18	$022	EE_RDY	EEPROM Ready
19	$024	ANA_COMP	Analog Comparator
20	$026	TWSI	2-wire Serial Interface

Notes: 1. When the BOOTRST fuse is programmed, the device will jump to the Boot Loader address at reset, see “Boot Loader Support” on page 172.
2. When the IVSEL bit in GICR is set, interrupt vectors will be moved to the start of the boot Flash section. The address of each interrupt vector will then be address in this table plus the start address of the boot Flash section.

Table 4 shows reset and interrupt vectors placement for the various combinations of BOOTRST and IVSEL settings.

Table 4. Reset and Interrupt Vectors Placement

BOOTRST	IVSEL	Reset address	Interrupt Vectors Start Address
0	0	$0000	$0002
0	1	$0000	Boot Reset Address + $0002
1	0	Boot Reset Address	$0002
1	1	Boot Reset Address	Boot Reset Address + $0002

Note: The Boot Reset Address is shown in Table 59 on page 172

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega323 is:

```
Address   Labels  Code            Comments
$000              jmp   RESET      ; Reset Handler
$002              jmp   EXT_INT0   ; IRQ0 Handler
$004              jmp   EXT_INT1   ; IRQ1 Handler
$006              jmp   EXT_INT2   ; IRQ2 Handler
$008              jmp   TIM2_COMP  ; Timer2 Compare Handler
$00a              jmp   TIM2_OVF   ; Timer2 Overflow Handler
$00c              jmp   TIM1_CAPT  ; Timer1 Capture Handler
$00e              jmp   TIM1_COMPA ; Timer1 CompareA Handler
$010              jmp   TIM1_COMPB ; Timer1 CompareB Handler
$012              jmp   TIM1_OVF   ; Timer1 Overflow Handler
$014              jmp   TIM0_COMP  ; Timer0 Compare Handler
$016              jmp   TIM0_OVF   ; Timer0 Overflow Handler
$018              jmp   SPI_STC;   ; SPI Transfer Complete Handler
```

```
$01a            jmp   USART_RXC  ; USART RX Complete Handler
$01c            jmp   USART_UDRE ; UDR Empty Handler
$01e            jmp   USART_TXC  ; USART TX Complete Handler
$020            jmp   ADC        ; ADC Conversion Complete Interrupt Handler
$022            jmp   EE_RDY     ; EEPROM Ready Handler
$024            jmp   ANA_COMP   ; Analog Comparator Handler
$026            jmp   TWSI       ; 2-wire Serial Interface Interrupt Handler
;
$028       MAIN: ldi   r16,high(RAMEND); Main program start
$029             out   SPH,r16    ; Set stack pointer to top of RAM
$02a             ldi   r16,low(RAMEND)
$02b             out   SPL,r16
$02c             <instr>  xxx
...         ...   ...
```

When the BOOTRST fuse is unprogrammed, the boot section size set to 4K bytes and the IVSEL bit in the GICR register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses are:

```
Address   Labels  Code            Comments
$000            jmp   RESET      ; Reset handler
;
$002     MAIN:  ldi   r16,high(RAMEND); Main program start
$003            out   SPH,r16    ; Set stack pointer to top of RAM
$004            ldi   r16,low(RAMEND)
$005            out   SPL,r16
$006            <instr>  xxx
;
.org $3802
$3802           jmp   EXT_INT0   ; IRQ0 Handler
$3804           jmp   EXT_INT1   ; IRQ1 Handler
...             ...   ...        ;
$3826           jmp   TWSI       ; 2-wire Serial Interface Interrupt Handler
```

When the BOOTRST fuse is programmed and the boot section size set to 4K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses are:

```
Address   Labels  Code           Comments
.org $002
$002            jmp   EXT_INT0   ; IRQ0 Handler
$004            jmp   EXT_INT1   ; IRQ1 Handler
...             ...   ...        ;
$026            jmp   TWSI       ; 2-wire Serial Interface Interrupt Handler
;
$028     MAIN:  ldi   r16,high(RAMEND); Main program start
$029            out   SPH,r16    ; Set stack pointer to top of RAM
$02a            ldi   r16,low(RAMEND)
$02b            out   SPL,r16
$02c            <instr>  xxx
;
.org $3800
$3800           jmp   RESET      ; Reset handler
…        …      …      …
```

When the BOOTRST fuse is programmed, the boot section size set to 4K bytes and the IVSEL bit in the GICR register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses are:

```
Address Labels  Code            Comments
$000    MAIN:   ldi   r16,high(RAMEND); Main program start
$001            out   SPH,r16    ; Set stack pointer to top of RAM
$002            ldi   r16,low(RAMEND)
$003            out   SPL,r16
$004            <instr>  xxx
;
.org $3800
$3800           jmp   RESET    ; Reset handler
$3802           jmp   EXT_INT0 ; IRQ0 Handler
$3804           jmp   EXT_INT1 ; IRQ1 Handler
...             ...   ...      ;
$3826           jmp   TWSI     ; 2-wire Serial Interface Interrupt Handler
```

Reset Sources

The ATmega323 has five 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}).
- JTAG AVR Reset. The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system.

During reset, all I/O registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a 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. This is also the case if the Reset Vector is in the Application section while the interrupt vectors are in the boot section or vice versa. The circuit diagram in Figure 24 shows the reset logic. Table 5 and Table 6 define the timing and electrical parameters of the reset circuitry.

Figure 24. Reset Logic

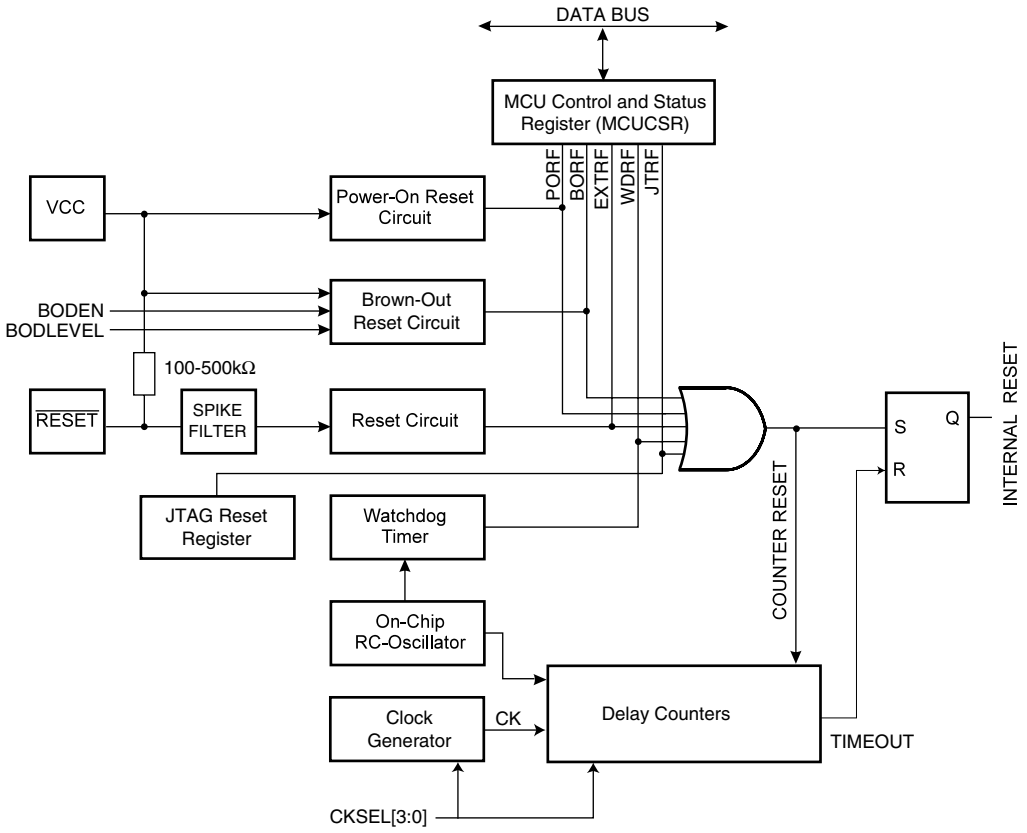


Table 5. Reset Characteristics[1]

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)[2]		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. Values are guidelines only. Actual values are TBD.
2. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

Table 6. 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	67 ms + 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 6.
4. When BODEN is programmed, add 100 µs.
5. When BODEN is programmed, add 25 µs.
6. Default value.

Table 6 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 WDT oscillator cycles used for each time-out is shown in Table 7.

The frequency of the Watchdog oscillator is voltage dependent as shown in the Electrical Characteristics section. The device is shipped with CKSEL = “0010” (Internal RC Oscillator, slowly rising power).

Table 7. Number of Watchdog Oscillator Cycles

BODLEVEL[1]	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 5. 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 6. 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 VCC

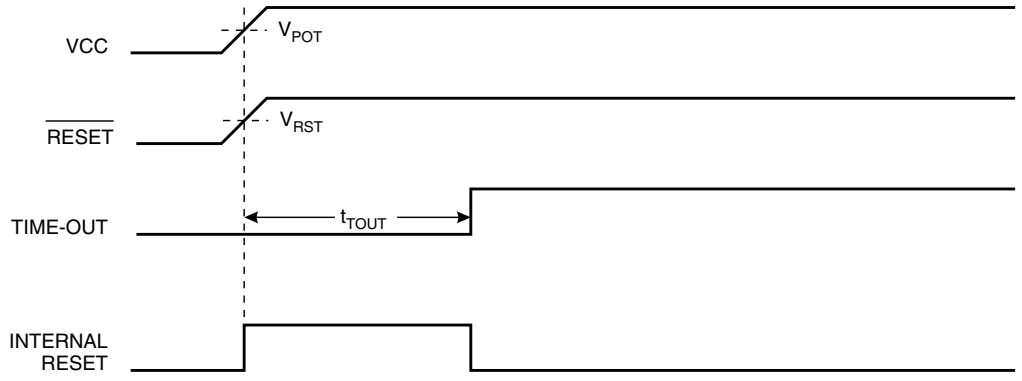
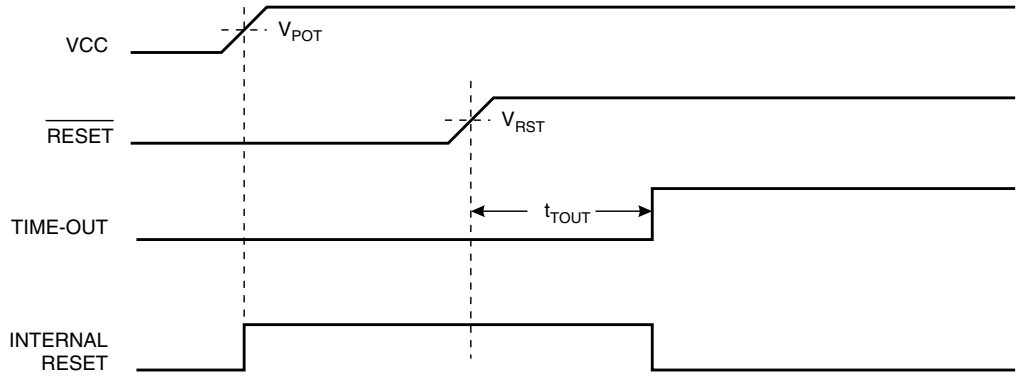


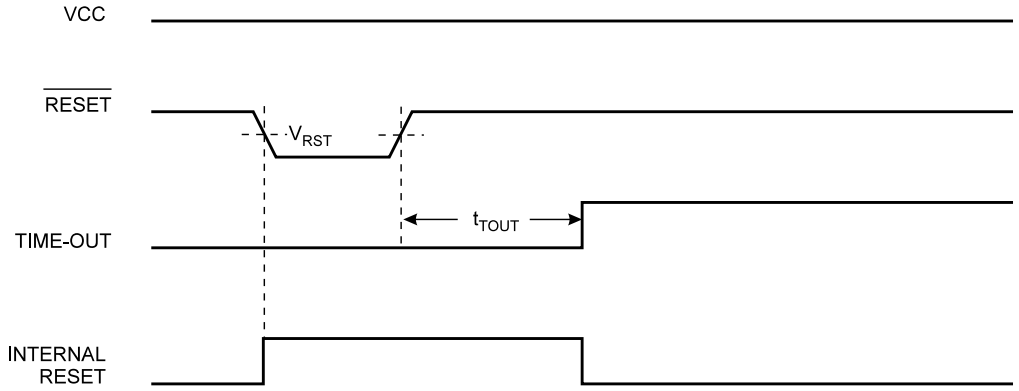
Figure 26. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



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

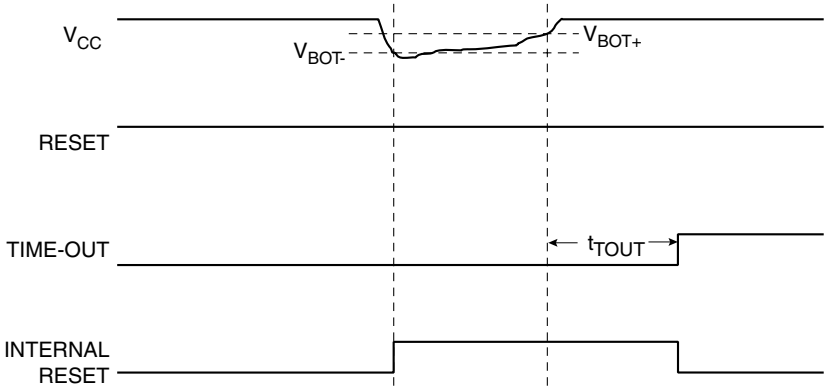


Brown-out Detection

ATmega323 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 6. 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

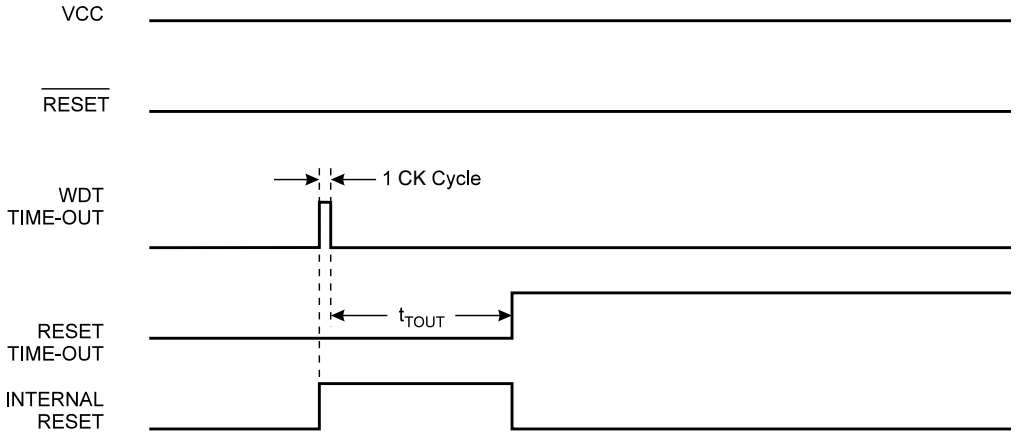


The hysteresis on V_{BOT}: $V_{BOT+} = V_{BOT} + 25$ mV, $V_{BOT-} = V_{BOT} - 25$ mV

Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 63 for details on operation of the Watchdog Timer.

Figure 29. Watchdog Reset During Operation



MCU Control and Status Register – MCUCSR

The MCU Control and Status Register contains control bits for general MCU functions, and provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
$34 ($54)	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- **Bits 7 - JTD: JTAG interface disable**

When this bit is cleared (zero), the JTAG interface is enabled if the JTAGEN fuse is programmed. If this bit is set (one), the JTAG interface is disabled. To avoid unintentional disabling of the JTAG interface, the user software must write this bit as one twice within four cycles to set the bit.

- **Bit 6 - ISC2: Interrupt Sense Control 2**

The asynchronous external interrupt 2 is activated by the external pin INT2 if the SREG I-flag and the corresponding interrupt mask in the GICR are set. If ISC2 is cleared

(zero), a falling edge on INT2 activates the interrupt. If ISC2 is set (one) a rising edge on INT2 activates the interrupt. Edges on INT2 are registered asynchronously. Pulses on INT2 wider than 50 ns will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. When changing the ISC2 bit, an interrupt can occur. Therefore, it is recommended to first disable INT2 by clearing its Interrupt Enable bit in the GICR register. Then, the ISC2 bit can be changed. Finally, the INT2 interrupt flag should be cleared by writing a logical one to its Interrupt Flag bit in the GIFR register before the interrupt is re-enabled.

- **Bit 5 - Res: Reserved Bit**

This bit is a reserved bit in the ATmega323 and always reads as zero.

- **Bit 4 - JTRF: JTAG Reset Flag**

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

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

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

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

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

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

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

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

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

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

Internal Voltage Reference

ATmega323 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 or the ADC. The 2.56V reference to the ADC is generated from the internal bandgap reference.

Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The maximum start-up time is TBD. 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 ATmega323 has two 8-bit Interrupt Mask control registers: GICR – General Interrupt Control 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 can 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 while 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 while 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 4 clock cycles minimum. After 4 clock cycles the program vector address for the actual interrupt handling routine is executed. During this 4 clock cycle period, the Program Counter (14 bits) is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes 3 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 4 clock cycles.

A return from an interrupt handling routine takes 4 clock cycles. During these 4 clock cycles, the Program Counter (2 bytes) is popped back from the Stack, the Stack Pointer is incremented by 2, 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 Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	**INT1**	**INT0**	**INT2**	**-**	**-**	**-**	**IVSEL**	**IVCE**	**GICR**
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is 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 the INT1 interrupt vector. See also “External Interrupts” on page 36.

- **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 the INT0 interrupt vector. See also “External Interrupts” on page 36.

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

When the INT2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control2 bit (ISC02) in the MCU Control and Status Register (MCUCSR) defines whether the external interrupt is activated on rising or falling edge of the INT2 pin. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 interrupt vector. See also “External Interrupts” on page 36.

- **Bits 4..2 - Res: Reserved bits**

These bits are reserved bits in the ATmega323 and always read as zero.

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

When the IVSEL bit is cleared (zero), the interrupt vectors are placed at the start of the Flash memory. When this bit is set (one), the interrupt vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address to the start of the boot Flash section is determined by the BOOTSZ fuses. Refer to the section “Boot Loader Support” on page 172 for details. To avoid unintentional changes of interrupt vector tables, a special write procedure must be followed to change the IVSEL bit:

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

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

Note: If Boot Lock bits BLB02 or BLB12 are set, changing the interrupt vector table will change from what part of the program memory interrupts are allowed. Refer to the section “Boot Loader Support” on page 172 for details on Boot Lock bits.

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

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

The General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	INTF1	INTF0	INTF2	-	-	-	-	-	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

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

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

- **Bit 6 - INTF0: External Interrupt Flag0**

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

- **Bit 5 - INTF2: External Interrupt Flag2**

When an event on the INT2 pin triggers an interrupt request, INTF2 becomes set (one). If the I-bit in SREG and the INT2 bit in GICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bits 4..0 - Res: Reserved Bits**

These bits are reserved bits in the ATmega323 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	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - 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 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 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 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 CompareA Match interrupt is enabled. The corresponding interrupt is executed if a CompareA 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 CompareB Match interrupt is enabled. The corresponding interrupt is executed if a CompareB 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 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 - OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable**

When the OCIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match interrupt is enabled. The corresponding interrupt is executed if a Compare0 match in Timer/Counter0 occurs, i.e., when the OCF0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **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 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	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - OCF2: Output Compare Flag 2**

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 PWM mode, this bit is set when Timer/Counter2 changes counting direction at $00.

- **Bit 5 - ICF1: Input Capture Flag 1**

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. When the I-bit in SREG, and TICIE1 (Timer/Counter1 Input Capture Interrupt Enable), and the ICF1 are set (one), the Timer/Counter1 Capture Interrupt is executed.

- **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 InterruptA Enable), and the OCF1A are set (one), the Timer/Counter1 Compare match A 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 InterruptB Enable), and the OCF1B are set (one), the Timer/Counter1 Compare match B 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 PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

- **Bit 1- OCF0: Output Compare Flag 0**

The OCF0 bit is set (one) when a compare match occurs between the Timer/Counter0 and the data in OCR0 – Output Compare Register 0. OCF0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE0 (Timer/Counter0 Compare match Interrupt Enable), and the OCF0 are set (one), the Timer/Counter0 Compare match Interrupt is executed.

- **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. In PWM mode, this bit is set when Timer/Counter0 changes counting direction at $00.

External Interrupts

The external interrupts are triggered by the INT0, INT1 and INT2 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..2 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level (INT2 is only an edge triggered interrupt). This is set up as indicated in the specification for the MCU Control Register – MCUCR and MCU Control and Status Register – MCUCSR. When the external interrupt

is enabled and is configured as level triggered (only INT0/INT1), the interrupt will trigger as long as the pin is held low.

MCU Control Register – MCUCR

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

Bit	7	6	5	4	3	2	1	0	
$37 ($57)	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7- 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 6..4 - SM2..0: Sleep Mode Select Bits 2, 1 and 0**

These bits select between the six available sleep modes as shown in Table 8.

Table 8. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Note: 1. Standby Mode and Extended Standby Mode are only available with external crystals or resonators.

- **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 GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 9. 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 9. 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 10. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 10. 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 six sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. The SM2, SM1 and SM0 bits in the MCUCR register select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby or Extended Standby) will be activated by the SLEEP instruction. See Table 8 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 SM2..0 bits are set to 000, the SLEEP instruction makes the MCU enter Idle Mode, stopping the CPU but allowing SPI, USART, Analog Comparator, ADC, 2-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD-bit in the Analog Comparator Control and Status register – ACSR. This will reduce power consumption in Idle Mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

ADC Noise Reduction Mode

When the SM2..0 bits are set to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction Mode, stopping the MCU but allowing the ADC, the external interrupts, the 2-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 form the ADC Conversion Complete interrupt, only an external reset, a Watchdog Reset, a Brown-out Reset, a 2-wire Serial Interface address match interrupt, or an external level interrupt on INT0 or INT1, or an external edge interrupt on INT2, 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 SM2..0 bits are 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the external oscillator is stopped, while the external interrupts, the 2-wire Serial Interface address watch, and the Watchdog continue operating (if enabled). Only an external reset, a Watchdog Reset, an 2-wire Serial Interface address match interrupt, an external level interrupt on INT0 or INT1, or an external edge interrupt on INT2 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 6 on page 27.

Power-save Mode

When the SM2..0 bits are 011, 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.

Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction forces the MCU into the Standby Mode. This mode is identical to Power-down with the exception that the oscillator is kept running. From Standby Mode, the device wakes up in only 6 clock cycles.

Extended Standby Mode

When the SM2..0 bits are 111 and an external crystal/resonator clock option is selected, the SLEEP instruction forces the MCU into the Extended Standby Mode. This mode is identical to Power-save mode with the exception that the oscillator is kept running. From Extended Standby Mode, the device wakes up in only 6 clock cycles.

Calibrated Internal RC Oscillator

The calibrated internal oscillator provides a fixed 1.0 MHz (nominal) clock at 5V and 25°C. This clock may be used as the system clock. See the section “Clock Options” on page 6 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. For details on how to use the pre-programmed calibration value, see the section “Calibration Byte” on page 183.

Oscillator Calibration Register – OSCCAL

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

Table 11. 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 IO 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 ATmega323 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 123.

- **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 134

- **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 52 for a detailed description of asynchronous operation.

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

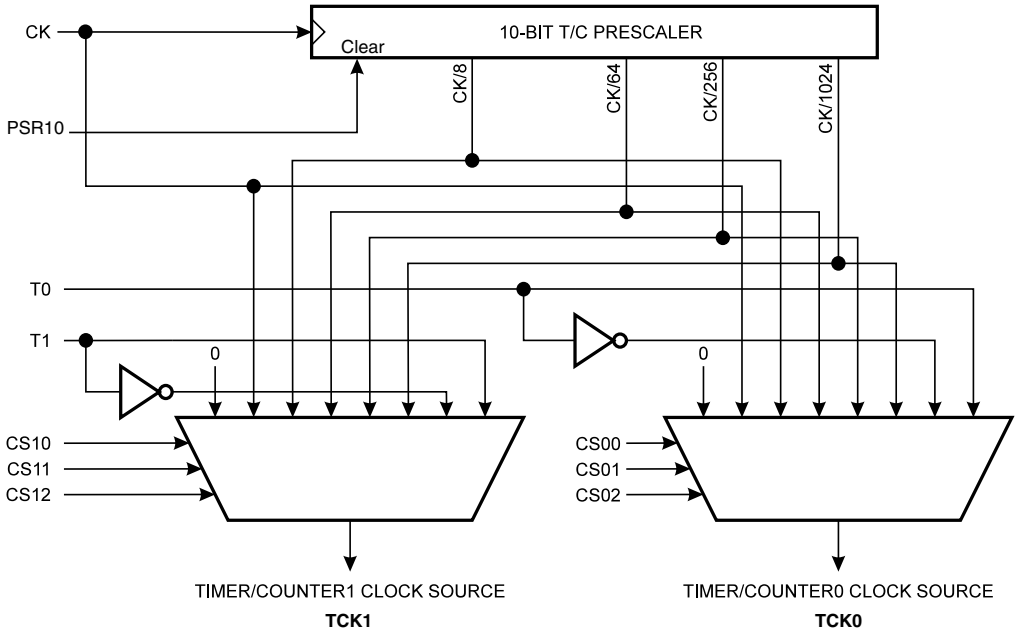
When this bit is set (one) the Timer/Counter1 and Timer/Counter0 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Writing a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler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

Timer/Counters

The ATmega323 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 Counter (RTC). Timer/Counters 0 and 1 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 IO 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.

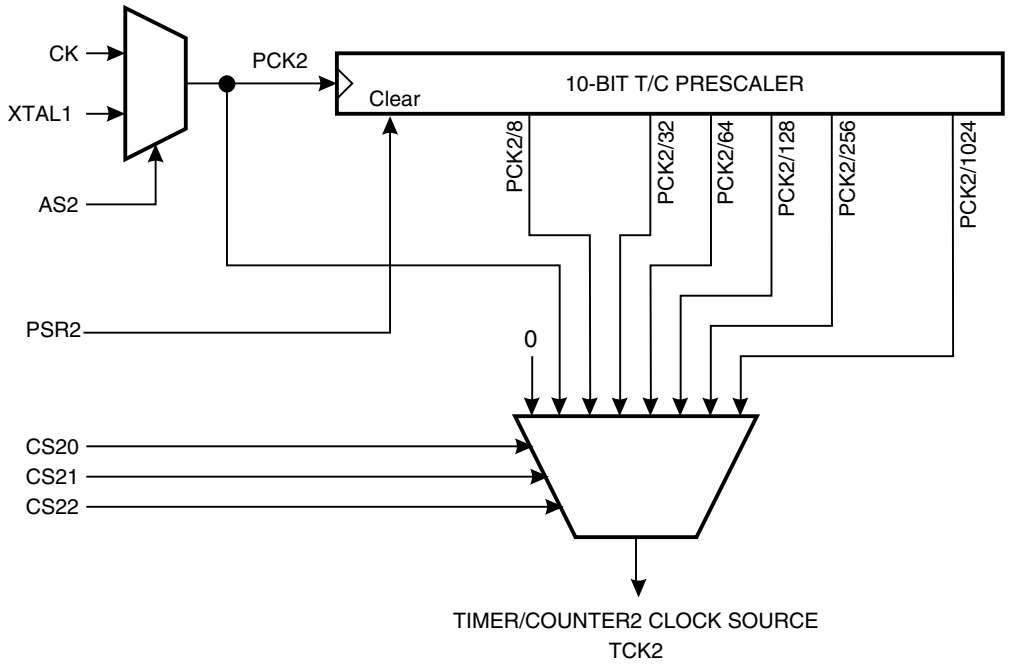
Timer/Counter Prescalers

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



For Timer/Counters 0 and 1, 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/Counters 0 and 1, 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/Counter 0 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 Counter (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 Timers/Counters T/C0 and T/C2

Figure 32 shows the block diagram for Timer/Counter0. Figure 33 shows the block diagram for Timer/Counter2.

Figure 32. Timer/Counter0 Block Diagram

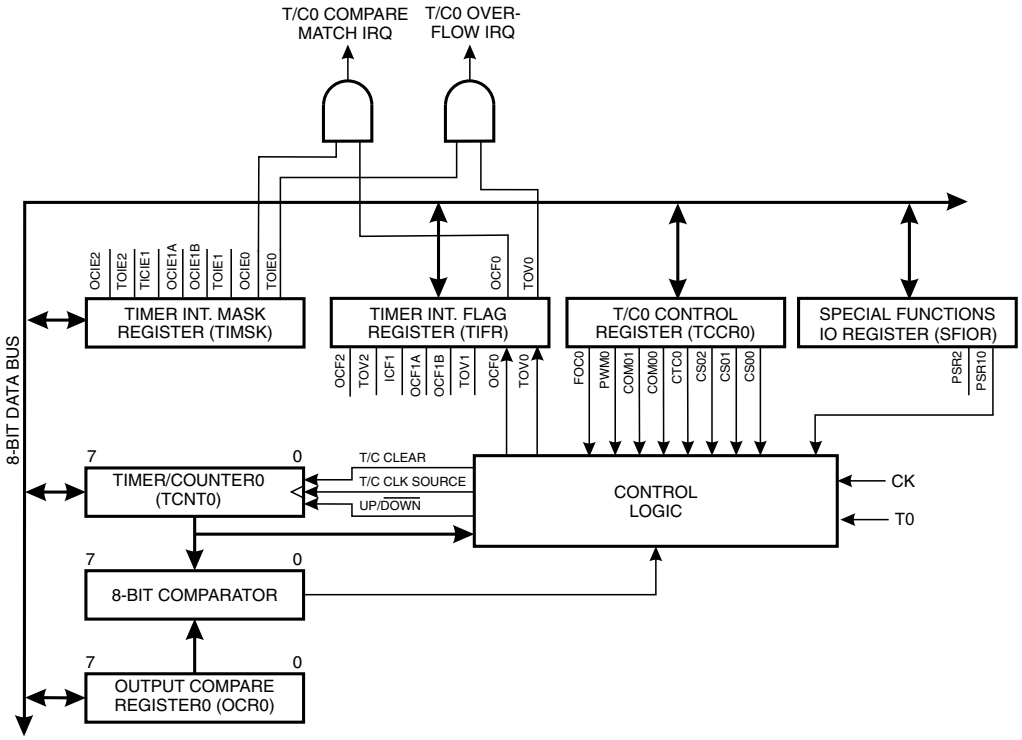
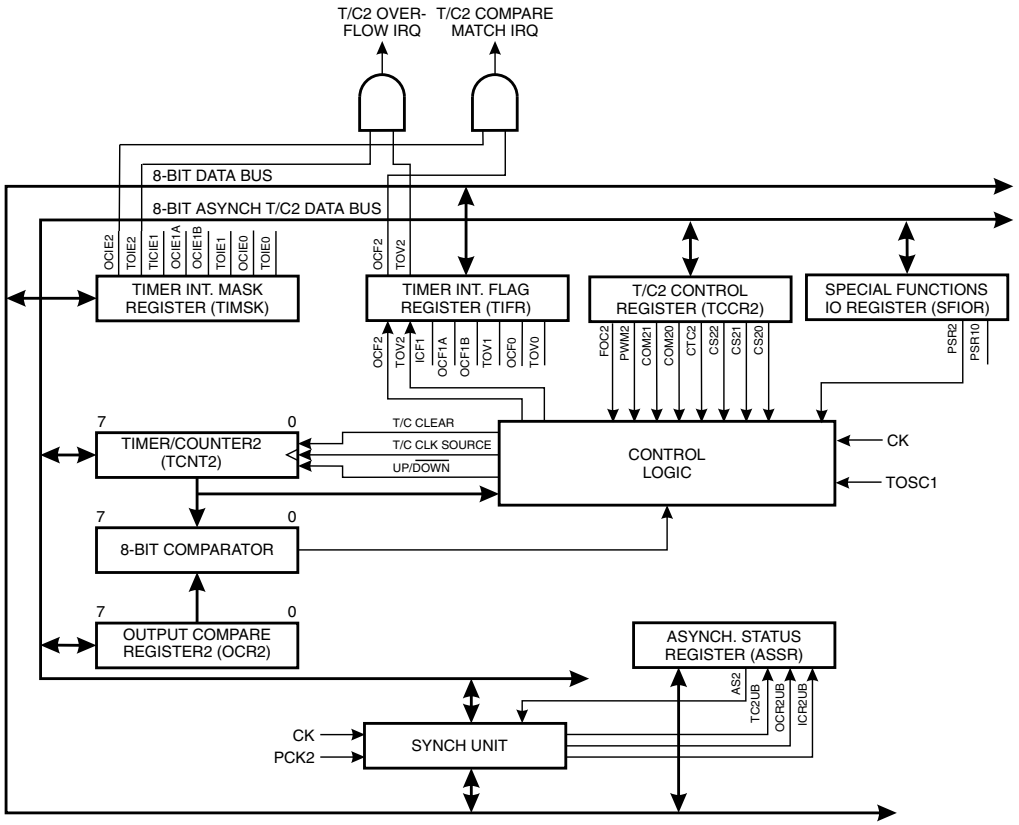


Figure 33. Timer/Counter2 Block Diagram



The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin.

The 8-bit Timer/Counter2 can select clock source from CK, prescaled CK, external TOSC1 or prescaled external TOSC1.

Both Timers/Counters can be stopped as described in section “Timer/Counter0 Control Register – TCCR0” on page 46 and “Bit 7 - FOC0/FOC2: Force Output Compare” on page 46.

The various status flags (overflow and compare match) are found in the Timer/Counter Interrupt Flag Register – TIFR (see page 35). Control signals are found in the Timer/Counter Control Register – TCCR0 and TCCR2. The interrupt enable/disable settings are found in the Timer/Counter Interrupt Mask Register – TIMSK (see page 34).

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/Counters feature 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.

Timer/Counter0 and 2 can also be used as 8-bit Pulse Width Modulators. In this mode, the Timer/Counter and the output compare register serve as a glitch-free, stand-alone PWM with centered pulses. Refer to page 48 for a detailed description on this function.

Timer/Counter0 Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	FOC0	PWM0	COM01	COM00	CTC0	CS02	CS01	CS00	TCCR0
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	

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 - FOC0/FOC2: Force Output Compare**

Writing a logical one to this bit, forces a change in the compare match output pin PB3 (Timer/Counter0) and PD7 (Timer/Counter2) according to the values already set in COMn1 and COMn0. If the COMn1 and COMn0 bits are written in the same cycle as FOC0/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 COMn1 and COMn0 happens as if a Compare Match had occurred, but no interrupt is generated and the Timer/Counters will not be cleared even if CTC0/CTC2 is set. The corresponding I/O pin must be set as an output pin for the FOC0/FOC2 bit to have effect on the pin. The FOC0/FOC2 bits will always be read as zero. Setting the FOC0/FOC2 bits has no effect in PWM mode.

- **Bit 6 - PWM0/PWM2: Pulse Width Modulator Enable**

When set (one) this bit enables PWM mode for Timer/Counter0 or Timer/Counter2. This mode is described on page 48.

- **Bits 5,4 - COM01, COM00/COM21, COM20: Compare Output Mode, Bits 1 and 0**

The COMn1 and COMn0 control bits determine any output pin action following a compare match in Timer/Counter0 or Timer/Counter2. Output pin actions affect pins PB3(OC0) or 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 12.

Table 12. Compare Mode Select[1]

COMn1[2]	COMn0	Description
0	0	Timer/Counter Disconnected from Output Pin OCn
0	1	Toggle the OCn Output Line.
1	0	Clear the OCn Output Line (to Zero).
1	1	Set the OCn Output Line (to One).

Notes: 1. In PWM mode, these bits have a different function. Refer to Table 15 for a description.
2. n = 0 or 2

- **Bit 3 - CTC0/CTC2: Clear Timer/Counter on Compare Match**

When the CTC0 or CTC2 control bit is set (one), Timer/Counter0 or Timer/Counter2 is reset to $00 in the CPU clock cycle following a compare match. If the control bit is cleared, the Timer/Counter 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 CTC0/CTC2 is set:

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

When the prescaler is set to divide by 8, 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 CTC0 or CTC2 bit is cleared in PWM mode, the Timer/Counter acts as an up/down counter. If the CTC0 or CTC2 bit is set (one), the Timer/Counter wraps when it reaches $FF. Refer to page 48 for a detailed description.

- **Bits 2,1,0 - CS02, CS01, CS00/ CS22, CS21, CS20: Clock Select bits 2,1 and 0**

The Clock Select bits 2,1 and 0 define the prescaling source of Timer/Counter0 and Timer/Counter2.

Table 13. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, the 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 PB0(T0), Falling Edge
1	1	1	External Pin PB0(T0), Rising Edge

Table 14. Clock 2 Prescale Select

CS22	CS21	CS20	Description
0	0	0	Stop, the 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 CK oscillator clock for Timer/Counter0 and PCK2 for Timer/Counter2. 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 Counter0 – TCNT0

Bit	7	6	5	4	3	2	1	0	
$32 ($52)	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	

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	

These 8-bit registers contain the value of the Timer/Counters.

Both Timer/Counters is realized as up or up/down (in PWM mode) counters with read and write access. If the Timer/Counter is written to and a clock source is selected, it continues counting in the timer clock cycle following the write operation.

Timer/Counter0 Output Compare Register – OCR0

Bit	7	6	5	4	3	2	1	0	
$3C ($5C)	MSB							LSB	OCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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 registers are 8-bit read/write registers. The Timer/Counter Output Compare Registers contains the data to be continuously compared with the Timer/Counter. Actions on compare matches are specified in TCCR0 and TCCR2. 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.

Timer/Counter 0 and 2 in PWM Mode

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

If the up/down mode is selected, the Timer/Counter and the Output Compare Registers – OCR0 or OCR2 form an 8-bit, free-running, glitch-free and phase correct PWM with outputs on the PB3(OC0/PWM0) or PD7(OC2/PWM2) pin.

If the overflow mode is selected, the Timer/Counter and the Output Compare Registers – OCR0 or 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 CTC0 or CTC2 bit in the Timer/Counter Control Registers -TCCR0 or TCCR2 respectively.

If CTC0/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 PB3(OC0/PWM0) or PD7(OC2/PWM2) pin is set or cleared according to the settings of the COMn1/COMn0 bits in the Timer/Counter Control Registers TCCR0 or TCCR2.

If CTC0/CTC2 is set and PWM mode is selected, the Timer/Counters will wrap and start counting from $00 after reaching $FF. The PB3(OC0/PWM0) or PD7(OC2/PWM2) pin will be set or cleared according to the settings of COMn1/COMn0 on a Timer/Counter overflow or when the counter value matches the contents of the Output Compare Register. Refer to Table 15 for details.

Table 15. Compare Mode Select in PWM Mode

CTCn	COMn1	COMn0	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: n = 0 or 2

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 the OCR when the Timer/Counter reaches $FF. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR0 or OCR2 write. See Figure 34 and Figure 35 for examples.

Figure 34. Effects of Unsynchronized OCR Latching in Up/Down Mode

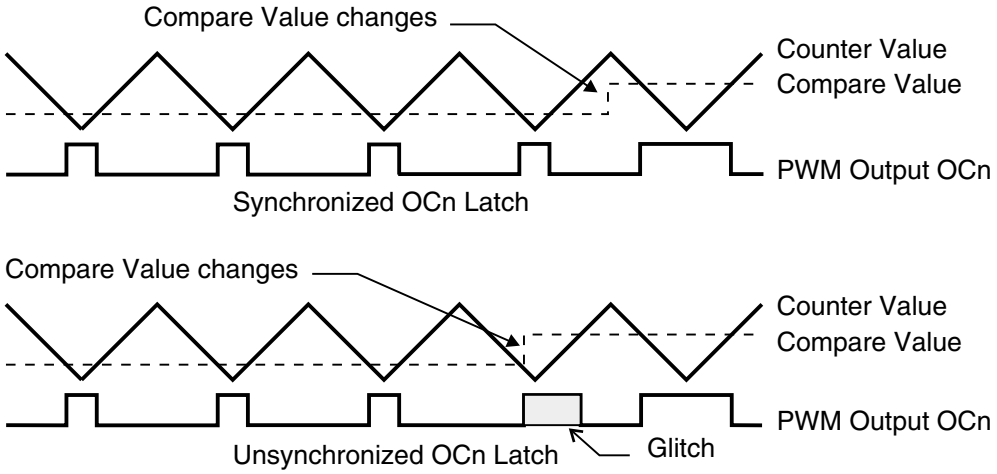
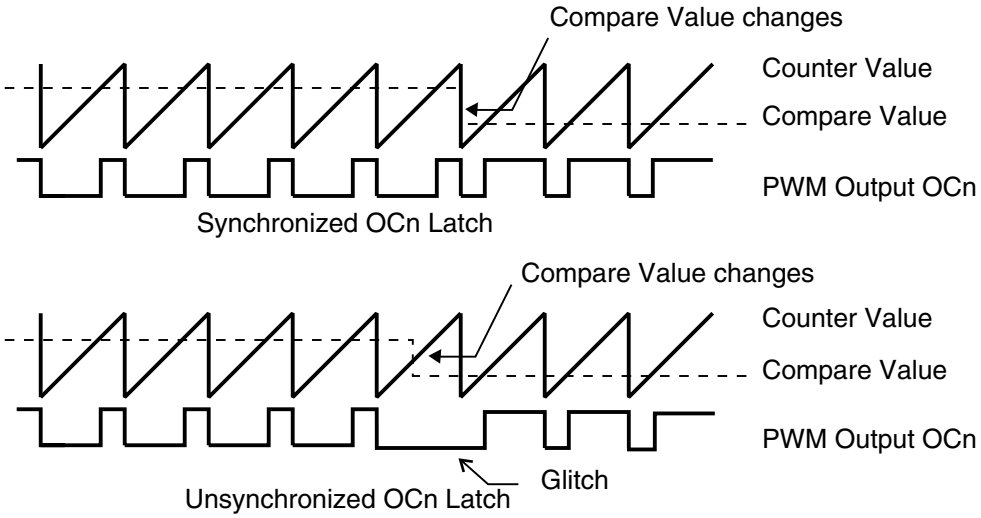


Figure 35. Effects of Unsynchronized OCR Latching in Overflow Mode



Note: n = 0 or 2 (Figure 34 and Figure 35)

During the time between the write and the latch operation, a read from the Output Compare Registers will read the contents of the temporary location. This means that the most recently written value always will read out of OCR0 and OCR2.

When the Output Compare Register contains $00 or $FF, and the up/down PWM mode is selected, the output PB3(OC0/PWM0)/PD7(OC2/PWM2) is updated to low or high on the next compare match according to the settings of COMn1/COMn0. This is shown in Table 16. In overflow PWM mode, the output PB3(OC0/PWM0)/PD7(OC2/PWM2) is held low or high only when the Output Compare Register contains $FF.

Table 16. PWM Outputs OCRn = $00 or $FF

COMn1	COMn0	OCRn	Output PWMn
1	0	$00	L
1	0	$FF	H
1	1	$00	H
1	1	$FF	L

Note: n = 0 or 2
In overflow PWM mode, the table above is only valid for OCRn = $FF.

In up/down PWM mode, the Timer Overflow Flag, TOV0 or TOV2, is set when the counter advances from $00. In overflow PWM mode, the Timer Overflow Flag is set as in normal Timer/Counter mode. Timer Overflow Interrupt0 and 2 operate exactly as in normal Timer/Counter mode, i.e. they are executed when TOV0 or TOV2 are set provided that Timer Overflow Interrupt and global interrupts are enabled. This does also apply to the Timer Output Compare flag and interrupt.

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 ATmega323 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 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 or Extended Standby 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 or Extended Standby mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep 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 or Extended Standby mode 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 or Extended Standby mode.

When the asynchronous operation is selected, the 32.768 kHZ oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a

power-up reset or wake-up from Power-down or Standby mode, 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 or Standby mode. The contents of all Timer/Counter2 registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the oscillator is in use or a clock signal is applied to the TOSC1 pin.

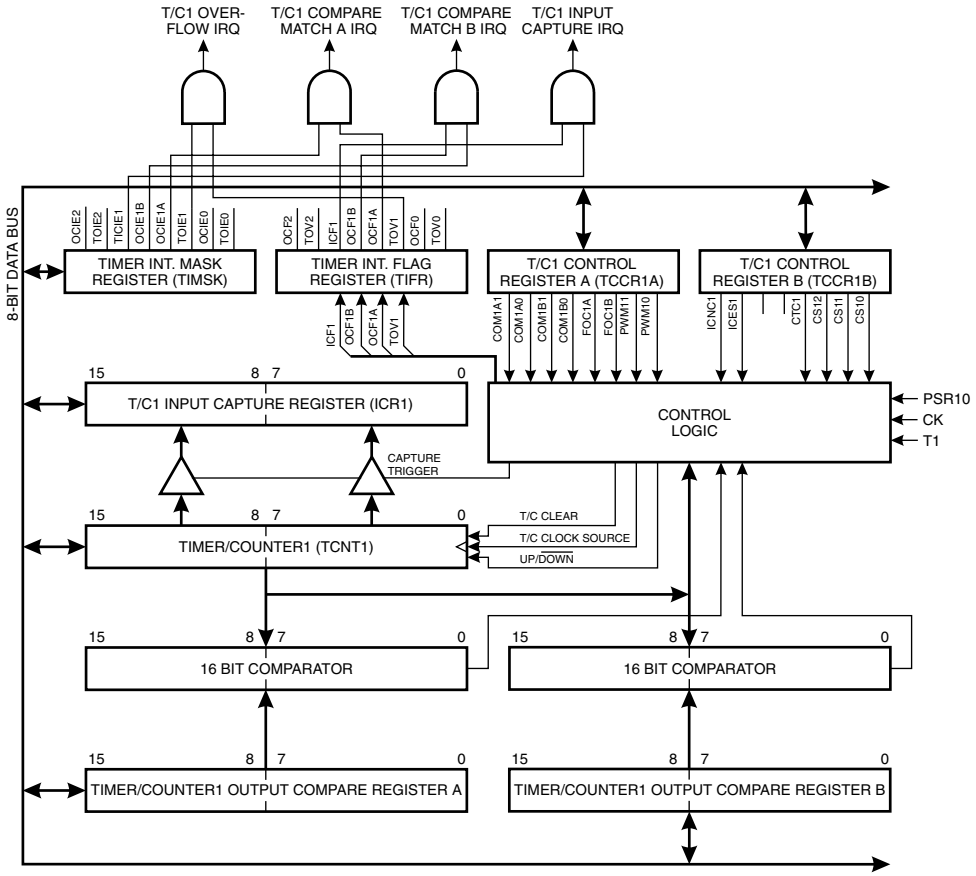
Description of wake up from Power-save or Extended Standby mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four 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 3 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.

16-bit Timer/Counter1

Figure 36 shows the block diagram for Timer/Counter1.

Figure 36. 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 56. 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.

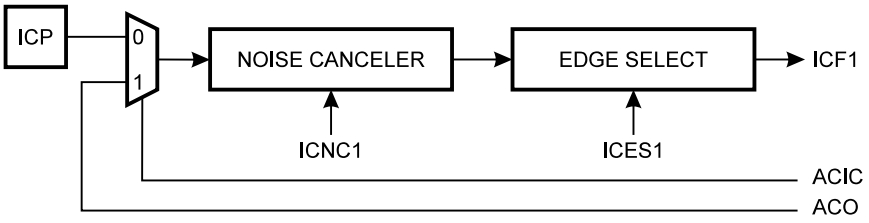
The 16-bit Timer/Counter1 features both a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high-prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

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 include optional clearing of the counter on compareA 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. In this mode the counter and the OCR1A/OCR1B registers serve as a dual glitch-free standalone 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 59 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", for details on this. The ICP pin logic is shown in Figure 37.

Figure 37. ICP Pin Schematic Diagram



If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over 4 samples, and all 4 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 CompareA. 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 CompareB. 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 17. Compare 1 Mode Select

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: X = A or B.

In PWM mode, these bits have a different function. Refer to Table 22 for a 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 59.

Table 18. 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 ATmega323 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 compareA 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 compareA 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 8, 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 59 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 19. 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 21), 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 17 on page 55 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 20. 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 20, 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. I.e. 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 21. 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)	$f_{TC1}/2046$

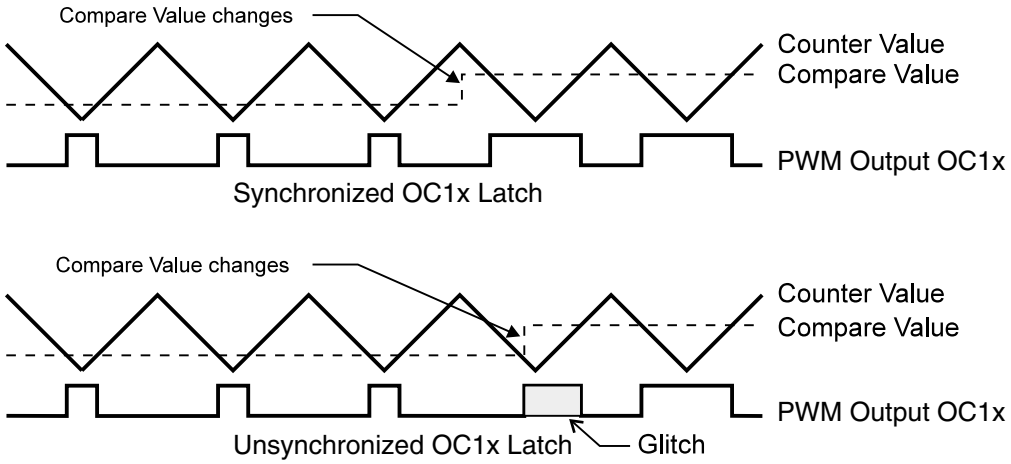
Table 22. Compare1 Mode Select in PWM Mode

CTC1	COM1X1	COM1X0	Effect on OCX1
0	0	0	Not connected
0	0	1	Reserved
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	Reserved
1	1	0	Cleared on compare match, set on overflow.
1	1	1	Set on compare match, cleared on overflow.

Note: X = A or B

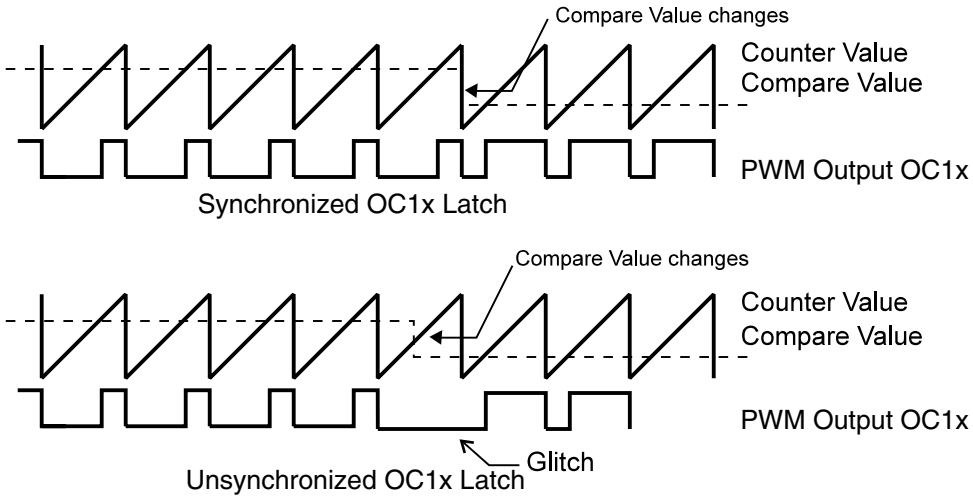
Note that in the PWM mode, the 8, 9, or 10 least significant OCR1A/OCR1B bits (depending on resolution), when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 38 and Figure 39 for an example in each mode.

Figure 38. Effects of Unsynchronized OCR1 Latching.



Note: x = A or B

Figure 39. 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 23. In overflow PWM mode, the output OC1A/OC1B is held low or high only when the Output Compare Register contains TOP.

Table 23. PWM Outputs OCR1X = $0000 or TOP

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

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

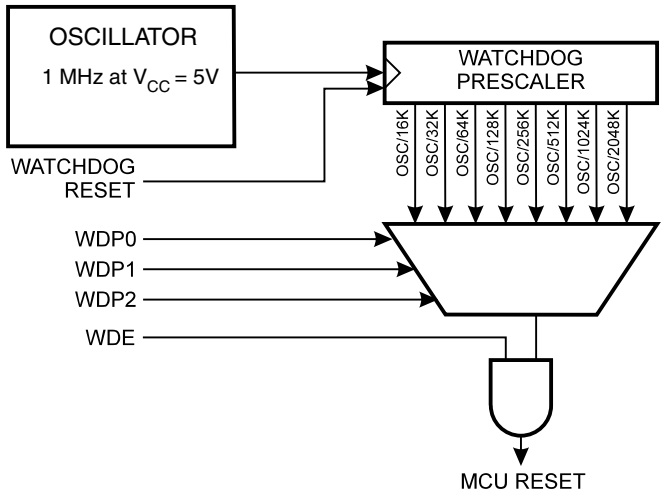
In PWM mode, the Timer Overflow Flag1, TOV1, is set when the counter advances from $0000. In overflow PWM mode, the Timer Overflow flag is set as in normal Timer/Counter mode. Timer Overflow Interrupt1 operates exactly as in normal Timer/Counter mode, i.e. it is executed when TOV1 is set provided that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

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 24 on page 64. 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 ATmega323 resets and executes from the reset vector. For timing details on the Watchdog Reset, refer to page 30.

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 ATmega323 and will always read as zero.

- **Bit 4 - WDTOE: Watch Dog 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: Watch Dog 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: Watch Dog 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 24.

Table 24. Watch Dog 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 frequency of the calibrated RC oscillator. See Table 25 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 possibly execute the EEPROM write code. To secure EEPROM integrity, the user is advised to use an external under-voltage reset circuit or the internal Brown-out Detector 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)	-	-	-	-	-	-	EEAR9	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/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	X	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..10 - Res: Reserved Bits**

These bits are reserved bits in the ATmega323 and will always read as zero.

- **Bits 9..0 - EEAR9..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 1K bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 1023. 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	X	0	

- **Bits 7..4 - Res: Reserved Bits**

These bits are reserved bits in the ATmega323 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 while writing a zero to EEWE 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 4 last steps to avoid these problems.

When the write access time 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 two cycles before the next instruction is executed.

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

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be 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 four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is not possible to set the EERE bit, nor to change the EEAR register.

The calibrated oscillator is used to time the EEPROM accesses. Table 25 lists the typical programming time for EEPROM access from the CPU.

Table 25. EEPROM Programming Time.

Symbol	Number of Calibrated RC Oscillator Cycles	Min Programming 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):

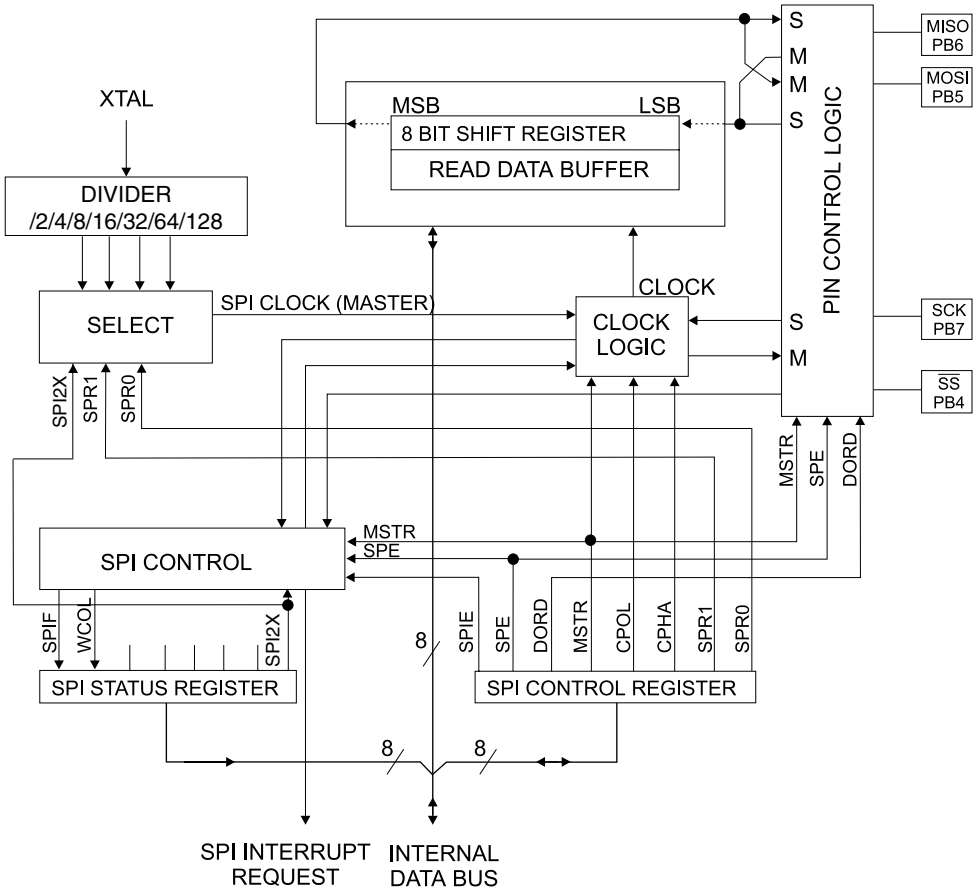
1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} Reset Protection circuit can be used. If a reset occurs while 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 172 for details.

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega323 and peripheral devices or between several AVR devices. The ATmega323 SPI includes the following features:

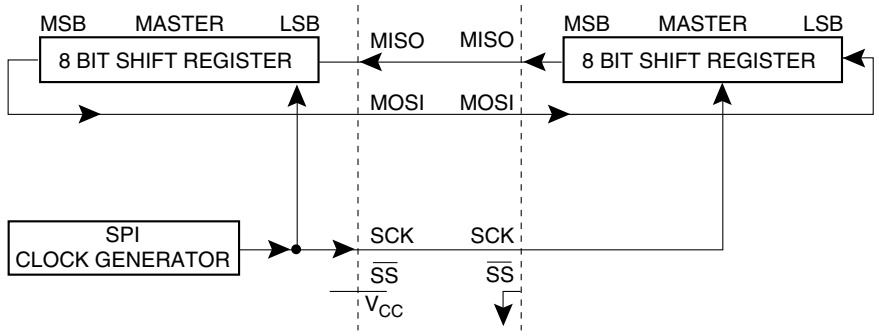
- **Full-duplex, 3-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master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

Figure 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 26.

Table 26. SPI Pin Overrides

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: See “Alternate Functions of Port B” on page 137 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.

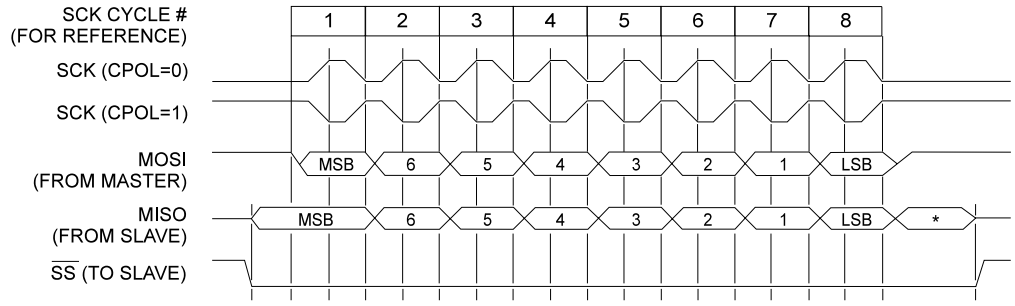
When the SPI is configured as a slave, the $\overline{SS}$ pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, 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

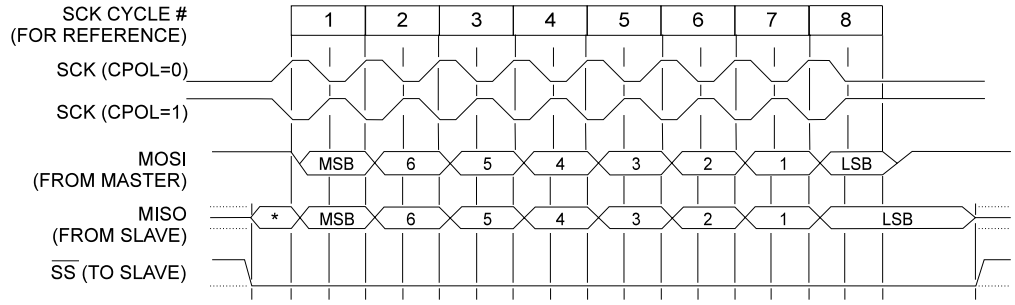
There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 43 and Figure 44.

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



Note: * Not defined but normally MSB of character just received.

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



Note: * 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 Table 27.

Table 27. 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 ATmega323 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 27). This means that the minimum SCK period will be 2 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 ATmega323 is also used for program memory and EEPROM downloading or uploading. See page 191 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.

USART

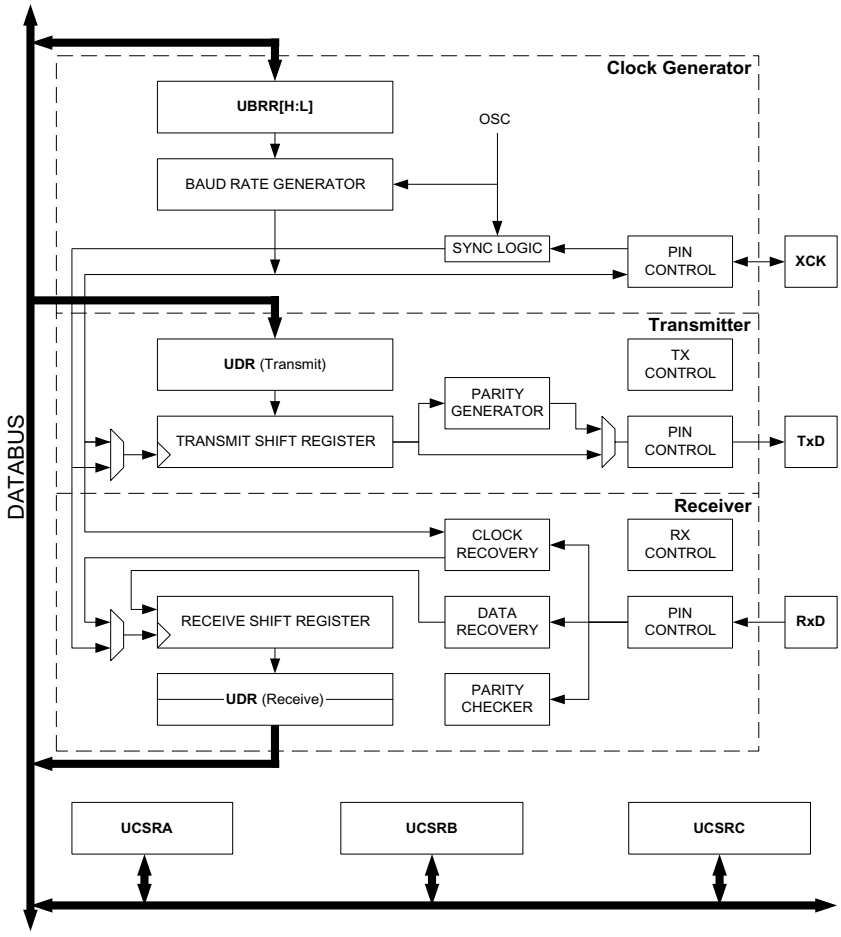
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **Asynchronous or Synchronous Operation**
- **Master or Slave Clocked Synchronous Operation**
- **High Resolution Baud Rate Generator**
- **Supports Serial Frames with 5, 6, 7, 8 or 9 Data Bits and 1 or 2 Stop Bits**
- **Odd or Even Parity Generation and Parity Check Supported by Hardware**
- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

Overview

A simplified block diagram of the USART transmitter is shown in Figure 45. CPU accessible I/O registers and I/O pins are shown in bold.

Figure 45. USART Block Diagram



The dashed boxes in the block diagram separates the three main parts of the USART (listed from the top): clock generation, transmitter and receiver. Control registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The

XCK (transfer clock) pin is only used by synchronous transfer mode. The transmitter consists of a single write buffer, a serial shift register, parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a shift register and a two level receive buffer (UDR). The receiver supports the same frame formats as the transmitter, and can detect frame error, data overrun and parity errors.

ATmega323 USART Pin Specification

Table 28 shows the ATmega323 specific USART pin placement.

Table 28. ATmega323 Specific USART Pin Placement

USART Pin Name	Corresponding ATmega323 Pin
RxD	PD0
TxD	PD1
XCK	PB0

As XCK is placed on PB0, DDR_XCK in the following refers to DDB0.

About Code Examples

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

AVR USART vs. AVR UART – Compatibility

The USART is fully compatible with the AVR UART regarding:

- Bit locations inside all USART registers
- Baud Rate Generation
- Transmitter Operation
- Transmit Buffer Functionality
- Receiver Operation

However, the receive buffering has two improvements that will affect the compatibility in some special cases:

- A second buffer register has been added. The two buffer registers operates as a circular FIFO buffer. Therefore the UDR must only be read once for each incoming data! More important is the fact that the error flags (FE and DOR) and the 9th data bit (RXB8) are buffered with the data in the receive buffer. Therefore the status bits must always be read before the UDR register is read. Otherwise the error status will be lost since the buffer state is lost.
- The receiver shift register can now act as a third buffer level. This is done by allowing the received data to remain in the serial shift register (see Figure 45) if the buffer registers are full, until a new start bit is detected. The USART is therefore more resistant to data overrun (DOR) error conditions.

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

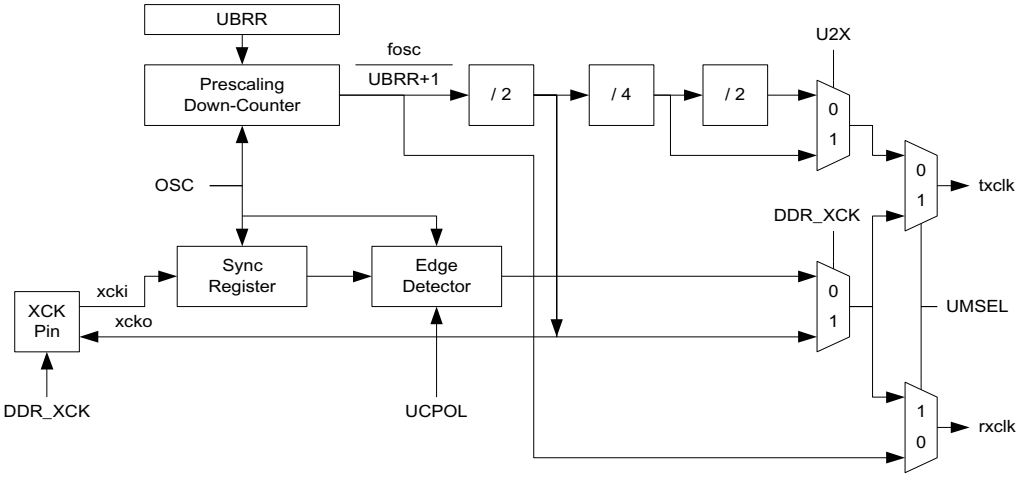
- CHR9 is changed to UCSZ2
- OR is changed to DOR

Clock Generation

The clock generation logic generates the base clock for the transmitter and receiver. The USART supports four modes of clock operation: normal asynchronous, double speed asynchronous, master synchronous and slave synchronous mode. The UMSEL bit in USART Control and Status Register C (UCSRC) selects between asynchronous and synchronous operation. Double speed (asynchronous mode only) is controlled by the U2X found in the UCSRA register. When using synchronous mode (UMSEL = 1), the Data Direction Register for the XCK pin (DDR_XCK) controls whether the clock source is internal (master mode) or external (slave mode). The XCK pin is only active when using synchronous mode.

Figure 46 shows a block diagram of the clock generation logic.

Figure 46. Clock Generation Logic, Block Diagram



Signal description:

- **txclk** Transmitter clock. (Internal Signal)
- **rxclk** Receiver base clock. (Internal Signal)
- **xcki** Input from XCK pin (internal Signal). Used for synchronous slave operation.
- **xcko** Clock output to XCK pin (Internal Signal). Used for synchronous master operation.
- **fosc** XTAL pin frequency (System Clock).

Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 46.

The USART Baud Rate Register (UBRR) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (fosc), is loaded with the UBRR value each time the counter has counted down to zero or when the UBRRL register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= fosc/(UBRR+1)). The transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSEL, U2X and DDR_XCK bits.

Table 29 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.

Table 29. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).
BAUD Baud rate (in bits per second, bps)
f_{OSC} System oscillator clock frequency
UBRR Contents of the UBRRH and UBRRL registers, (0-4095)
Some examples of UBRR values for some system clock frequency are found in Table 36 (see page 98).

Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the transmitter, there are no downsides.

External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 46 for details.

External clock input from the XCK pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK clock frequency is limited by the following equation:

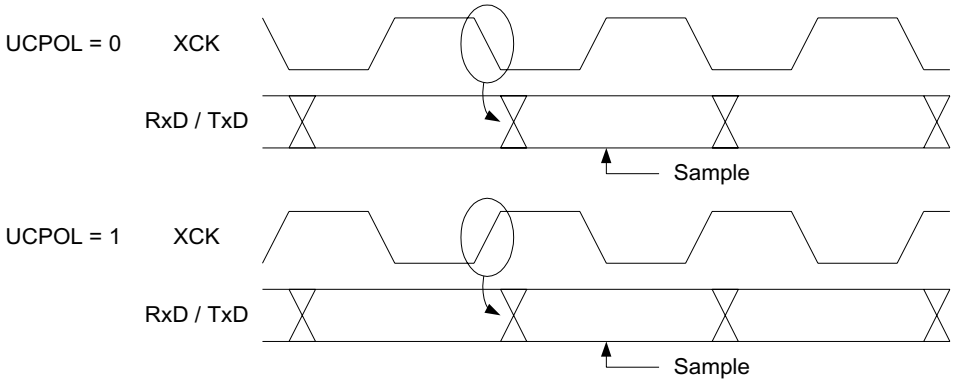
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCK pin will be used as either clock input (slave) or clock output (master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD) is sampled at the opposite XCK clock edge of the edge the data output (TxD) is changed.

Figure 47. Synchronous Mode XCK Timing



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 47 shows, when UCPOL is zero the data will be changed at falling XCK edge and sampled at rising XCK edge. If UCPOL is set, the data will be changed at rising XCK edge and sampled at falling XCK edge.

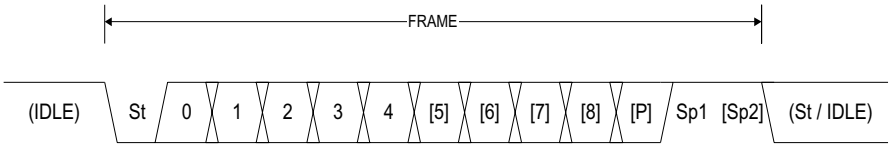
Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accept all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8 or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to a idle (high) state. Figure 48 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 48. Frame Formats



St Start bit, always low.

(n) Data bits (0 to 8).

P Parity bit. Can be odd or even.

Sp Stop bit, always high.

IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The receiver and transmitter uses the same setting. Note that

changing the setting of any of these bits will corrupt all ongoing communication for both the receiver and transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. *The* USART Parity Mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the *USART Stop Bit Select* (USBS) bit. The receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} Parity bit using even parity

P_{odd} Parity bit using odd parity

d_n Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the transmitter or the receiver depending on the usage. For interrupt driven USART operation, the global interrupt flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC flag can be used to check that the transmitter has completed all transfers, and the RXC flag can be used to check that there are no unread data in the receive buffer. Note that the TXC flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that is equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers. When the function writes to the UCSRC register, the URSEL bit (MSB) must be set due to the sharing of I/O location by UBRRH and UCSRC.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out UBRRH, r17
  out UBRRL, r16
  ; Enable receiver and transmitter
  ldi r16, (1<<RXEN)|(1<<TXEN)
  out UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
  out UCSRC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

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

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

Data Transmission – The USART Transmitter

The USART transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB register. When the transmitter is enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR I/O location. The buffered data in the transmit buffer will be moved to the shift register when the shift register is ready to send a new frame. The shift register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the shift register is loaded with new data, it will transfer one complete frame at the rate given by the baud register, U2X bit or by XCK depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the *Data Register Empty* (UDRE) flag. When using frames with less than eight bit, the most significant bits written to the UDR are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out   UDR,r16
  ret
```

C Code Example

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) ) {};
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

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

The function simply waits for the transmit buffer to be empty by checking the UDRE flag, before loading it with new data to be transmitted. If the data register empty interrupt is utilized, the interrupt routine writes the data into the buffer.

Sending Frames with 9 Data Bit

If 9 bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the low byte of the character written to UDR. The following code examples show a transmit function that handles 9 bit characters. For the assembly code, the data to be sent is assumed to be stored in Registers R17:R16.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; Copy 9th bit from r17 to TXB8
  cbi   UCSRB,TXB8
  sbrc  r17,0
  sbi   UCSRB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out   UDR,r16
  ret
```

C Code Example

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE))) ) {};
  /* Copy 9th bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. I.e. only the TXB8 bit of the UCSRB register is used after initialization.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. For compatibility with future devices, always set this bit to zero when writing the UCSRA register.

When the Data Register empty Interrupt Enable (UDRIE) bit in UCSRB is set, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the data register empty Interrupt routine must either write new data to UDR in order to clear UDRE or disable the data register empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC) flag bit is set one when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC flag is useful in half-duplex communication interfaces (like the RS485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRB is set, the USART Transmit Complete Interrupt will be executed when the TXC flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC flag, this is done automatically when the interrupt is executed.

Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

Disabling the Transmitter

The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e. when the transmit shift register and transmit buffer register does not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD pin.

Data Reception – The USART Receiver

The USART receiver is enabled by setting the Receive Enable (RXEN) bit in the UCSRB register. When the receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, i.e. a complete serial frame is present in the receive shift register, the contents of the shift register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get and return received data from buffer
  in    r16, UDR
  ret
```

C Code Example

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) ) {};
  /* Get and return received data from buffer */
  return UDR;
}
```

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

The function simply waits for data to be present in the receive buffer by checking the RXC flag, before reading the buffer and returning the value.

Receiving Frames with 9 Data Bits

If 9 bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR and PE status flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and PE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get status and 9th bit, then data from buffer
  in    r18, UCSRA
  in    r17, UCSRB
  in    r16, UDR
  ; If error, return -1
  andi  r18,(1<<FE)|(1<<DOR)|(1<<PE)
  breq  USART_ReceiveNoError
  ldi   r17, HIGH(-1)
  ldi   r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the 9th bit, then return
  lsr   r17
  andi  r17, 0x01
  ret
```

C Code Example

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) ) {};
  /* Get status and 9th bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<PE) )
    return -1;
  /* Filter the 9th bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

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

The receive function example reads all the I/O registers into the register file before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

Receive Compete Flag and Interrupt

The USART receiver has one flag that indicates the receiver state.

The Receive Complete (RXC) flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e. does not contain any unread data). If the receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete Interrupt will be executed as long as the RXC flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR in order to clear the RXC flag, otherwise a new interrupt will occur once the interrupt routine terminates.

Receiver Error Flags

The USART receiver has three error flags: Frame Error (FE), Data OverRun (DOR) and Parity Enable (PE). All can be accessed by reading UCSRA. Common for the error flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the error flags, the UCSRA must be read before the receive buffer (UDR), since reading the UDR I/O location changes the buffer read location. Another equality for the error flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRA is written for upward compatibility of future USART implementations. None of the error flags can generate interrupts.

The Frame Error (FE) flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE flag is zero when the stop bit was correctly read (as one), and the FE flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE flag is not affected by the setting of the USBS bit in UCSRC since the receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRA.

The Data OverRun (DOR) flag indicates data loss due to a receiver buffer full condition. A data overrun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive shift register, and a new start bit is detected. If the DOR flag is set there was one or more serial frame lost between the frame last read from UDR, and the next frame read from UDR. For compatibility with future devices, always set this bit to zero when writing to UCSRA. The DOR flag is cleared when the frame received was successfully moved from the shift register to the receive buffer.

The Parity Error (PE) flag indicates that the next frame in the receive buffer did have an parity error when received. If parity check is not enabled the PE bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRA. For more details see “Parity Bit Calculation” on page 78 and “Parity Checker” on page 85.

Parity Checker

The parity checker is active when the high USART Parity Mode (UPM1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPM0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error (PE) flag can then be read by software to check if the frame had a parity error.

The PE bit is set if the next character that can be read from the receive buffer had a parity error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read.

Disabling the Receiver

In contrast to the transmitter, disabling the of the receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e. the RXEN is set to zero) the receiver will no longer override the normal function of the RxD port pin. The receiver buffer FIFO will be flushed when the receiver is disabled. Remaining data in the buffer will be lost

Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the receiver is disabled, i.e. the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR I/O location until the RXC flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example[1]

```
USART_Flush:
  sbis  UCSRA, RXC
  ret
  in    r16, UDR
  rjmp  USART_Flush
```

C Code Example

```
void USART_Flush( void )
{
  unsigned char dummy;
  while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

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

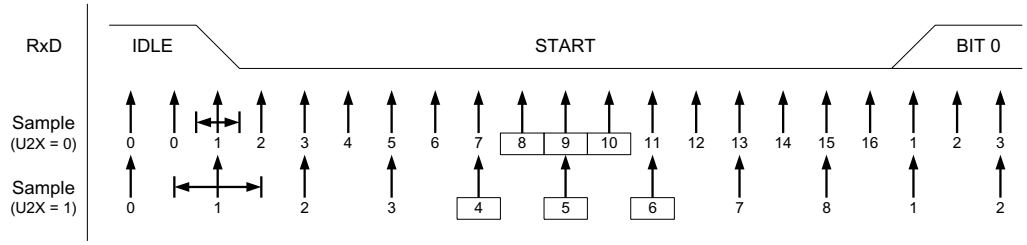
Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 49 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for normal mode, and 8 times the baud rate for double speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode (U2X = 1) of operation. Samples denoted zero are samples done when the RxD line is idle (i.e. no communication activity).

Figure 49. Start Bit Sampling

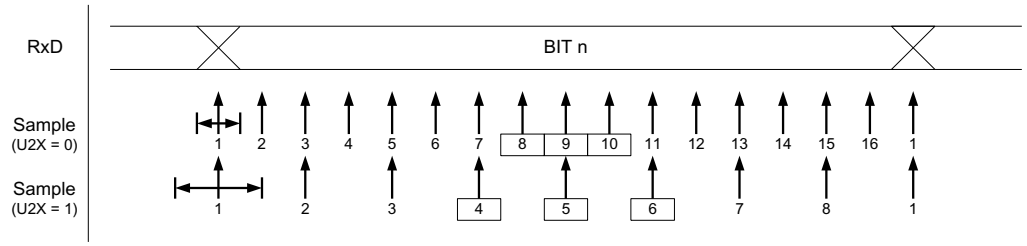


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9 and 10 for normal mode, and samples 4, 5 and 6 for double speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and 8 states for each bit in double speed mode. Figure 50 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

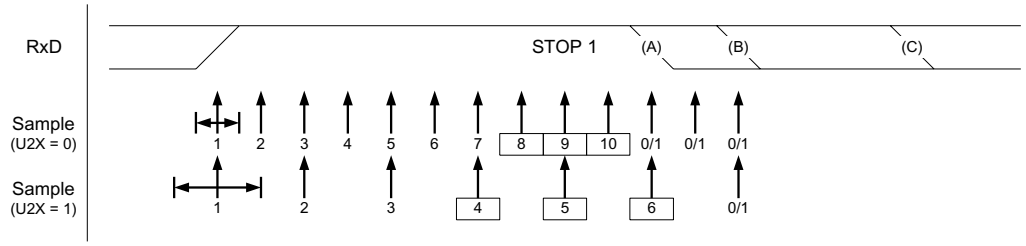
Figure 50. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process act as a low pass filter for the incoming signal on the RxD pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 51 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 51. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the frame error (FE) flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For normal speed mode, the first low level sample can be at point marked (A) in Figure 51. For double speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

Asynchronous Operational Range

The operational range of the receiver is dependent of the mismatch between the received bit rate and the internally generated baud rate. If the transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have exact base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D Sum of character size and parity size (D = 5 to 10 bit)

S Samples per bit. S = 16 for normal speed mode and S = 8 for double speed mode.

S_F First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for double speed mode.

S_M Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for double speed mode.

R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 30 and Table 31 list the maximum receiver baud rate error that can be tolerated. Note that normal speed mode has higher toleration of baud rate variations.

Table 30. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Table 31. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104.35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the receiver and transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

Multi-processor Communication Mode

Setting the Multi-Processor Communication Mode (MPCM) bit in UCSRA enables a filtering function of incoming frames received by the USART receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The transmitter is unaffected by the MPCM setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication Mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contain data or address information. If the receiver is set up for frames with 9 data bits, then the 9th bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the 9th bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor Communication Mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it will receive the following data frames as normal, while the other slave MCUs will ignore the received frames until another address frame is received.

Using MPCM

For an MCU to act as a master MCU, it can use a 9 bit character frame format (UCSZ = 7). The 9th bit (TXB8) must be set when an address frame (TXB8 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The slave MCUs must in this case be set to use a 9 bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication Mode:

1. All slave MCUs are in Multi-processor Communication Mode (MPCM in UCSRA is set).
2. The master MCU sends an address frame, and all slaves receive and read this frame. In the slave MCUs, the RXC flag in UCSRA will be set as normal.
3. Each slave MCU reads the UDR register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte and keeps the MPCM setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other slave MCUs, which still have the MPCM bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM bit and waits for a new address frame from master. The process then repeats from 2.

Using any of the 5 to 8 bit character frame formats is possible, but impractical since the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the transmitter and receiver uses the same character size setting. If 5 to 8 bit character frames are used, the transmitter must be set to use two stop bit (USBS = 1) since the first stop bit is used for indicating the frame type.

Do not use read-modify-write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC flag and this might accidentally be cleared when using SBI or CBI instructions.

Accessing UBRRH/UCSRC Registers

The UBRRH register shares the same I/O location as the UCSRC register. Therefore some special consideration must be taken when accessing this I/O location.

Write Access

When doing a write access of this I/O location, the high bit of the value written, the USART Register Select (URSEL) bit, controls which one of the two registers that will be written. If URSEL is zero during a write operation, the UBRRH value will be updated. If URSEL is one, the UCSRC setting will be updated.

The following code examples show how to access the two registers.

Assembly Code Examples[1]

```
...
; Set UBRRH to 2
ldi r16,0x02
out UBRRH,r16
...
; Set the USBS and the UCSZ1 bit to one, and
; the remaining bits to zero.
ldi r16,(1<<URSEL)|(1<<USBS)|(1<<UCSZ1)
out UCSRC,r16
...
```

C Code Examples

```
...
/* Set UBRRH to 2 */
UBRRH = 0x02;
...
/* Set the USBS and the UCSZ1 bit to one, and */
/* the remaining bits to zero. */
UCSRC = (1<<URSEL)|(1<<USBS)|(1<<UCSZ1)
...
```

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

As the code examples illustrate, write accesses of the two registers are relatively unaffected of the sharing of I/O location.

Read Access

Doing a read access to the UBRRH or the UCSRC register is a more complex operation. However, in most applications, it is rarely necessary to read any of these registers.

The read access is controlled by a timed sequence. Reading the I/O location once returns the UBRRH register contents. If the register location was read in previous system clock cycle, reading the register in the current clock cycle will return the UCSRC contents. Note that the timed sequence for reading the UCSRC is an atomic operation. Interrupts must therefore be disabled during the read operation.

The following code example shows how to read the UCSRC register contents.

Assembly Code Example[1]

```
USART_ReadUCSRC:
  ; Save global interrupt flag
  in    r17,SREG
  ; Disable interrupts
  cli
  ; Read UCSRC
  in  r16,UBRRH
  in  r16,UCSRC
  ; Restore global interrupt flag
  out SREG,r17
  ret
```

C Code Example

```
unsigned char USART_ReadUCSRC( void )
{
  unsigned char sreg, ucsrc;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Read UCSRC */
  ucsrc = UBRRH;
  ucsrc = UCSRC;
  /* Restore global interrupt flag */
  SREG = sreg;
  return ucsrc;
}
```

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

The assembly code example returns the UCSRC value in r16.

Reading the UBRRH contents is not an atomic operation and therefore it can be read as an ordinary register, as long as the previous instruction did not access the register location.

USART Register Description

USART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C) Read	RXB[7:0]								UDR (Read)
$0C ($2C) Write	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer register and USART Receive Data Buffer registers share the same I/O address referred to as USART Data Register or UDR. The transmit data buffer register (TXB) will be the destination for data written to the UDR register location. Reading the UDR register location will return the contents of the receive data buffer register (RXB).

For 5,6 or 7-bit characters the upper unused bits will be ignored by the transmitter and set to zero by the receiver.

The transmit buffer can only be written when the UDRE flag in the UCSRA register is set. Data written to UDR when the UDRE flag is not set, will be ignored by the USART transmitter. When data is written to the transmit buffer, and the transmitter is enabled, the transmitter will load the data into the transmit shift register when the shift register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read modify write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

USART Control and Status Register A – UCSRA

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

- **Bit 7 - RXC: USART Receive Complete**

This flag bit is one when there are unread data in the receive buffer and zero when the receive buffer is empty (i.e. does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 - TXC: USART Transmit Complete**

This flag bit is set one when the entire frame in the Transmit Shift register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 - UDRE: USART Data Register Empty**

The UDRE flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one the buffer is empty and therefore ready be written. The UDRE flag can generate a Data Register Empty interrupt (see description of the UDRIE bit).

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

- **Bit 4 - FE: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e. when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

- **Bit 3 - DOR: Data OverRun**

This bit is set if a data overrun condition is detected. A data overrun occur when the receive buffer is full (two characters), it is a new character waiting in the receive shift register, and a new start bit is detected. Always set this bit to zero when writing to UCSRA.

- **Bit 2 - PE: Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 1 - U2X: Double the USART Transmission Speed**

Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

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

Setting this bit enables the Multi-processor Communication Mode. When the MPCM bit is set, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 89.

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	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	0	0	

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

Setting this bit to one enables interrupt on the RXC flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is set, the global interrupt flag in SREG is set and the RXC bit in UCSRA is set.

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

Setting this bit to one enables interrupt on the TXC flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is set, the global interrupt flag in SREG is set and the TXC bit in UCSRA is set.

- **Bit 5 - UDRIE: USART Data Register Empty Interrupt Enable**

Setting this bit to one enables interrupt on the UDRE flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is set, the global interrupt flag in SREG is set and the UDRE bit in UCSRA is set.

- **Bit 4 - RXEN: Receiver Enable**

Setting this bit to one enables the USART receiver. The receiver will override normal port operation for the RxD pin when enabled. Disabling the receiver will flush the receive buffer invalidating the FE, DOR and PE flags.

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

Setting this bit to one enables the USART transmitter. The transmitter will override normal port operation for the TxD pin when enabled. The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e. when the transmit shift register and transmit buffer register does not contain data to be transmitted. When disabled the transmitter will no longer override the TxD port.

- **Bit 2 - UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (character size) in a frame the receiver and transmitter use.

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

RXB8 is the 9th data bit of the received character when operating with serial frames with 9 data bits. Must be read before reading the low bits from UDR.

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

TXB8 is the 9th data bit in the character to be transmitted when operating with serial frames with 9 data bits. Must be written before writing the low bits to UDR.

USART Control and Status Register C – UCSRC

Bit	7	6	5	4	3	2	1	0	
$20 ($40)	**URSEL**	**UMSEL**	**UPM1**	**UPM0**	**USBS**	**UCSZ1**	**UCSZ0**	**UCPOL**	**UCSRC**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

The UCSRC register shares the same I/O location as the UBRRH register. See the "Accessing UBRRH/UCSRC Registers" on page 91 section which describes how to access this register.

- **Bit 7 - URSEL: Register Select**

This bit selects between accessing the UCSRC or the UBRRH register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

- **Bit 6 - UMSEL: USART Mode Select**

This bit selects between asynchronous and synchronous mode of operation.

Table 32. USART Mode

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

- **Bit 5:4 - UPM1:0: Parity Mode**

This bit enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE flag in UCSRA will be set.

Table 33. Parity Mode

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	(Reserved)
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 - USBS: Stop Bit Select**

This bit selects number of stop bits to be inserted by the transmitter. The receiver ignores this setting.

Table 34. Stop Bit Select

USBS	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:1 - UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (character size) in a frame the receiver and transmitter uses.

Table 35. Character Size

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 - UCPOL: Clock Polarity**

This bit is used for synchronous mode only. Set this bit to zero when asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Falling XCK Edge	Rising XCK Edge
1	Rising XCK Edge	Falling XCK Edge

USART Baud Rate Registers – UBRRL and UBRRHs

Bit	15	14	13	12	11	10	9	8	
$20 ($40)	URSEL	-	-	-	UBRR[11:8]				UBRRH
$09 ($29)	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The UBRRH register shares the same I/O location as the UCSRC register. See the "Accessing UBRRH/UCSRC Registers" on page 91 section which describes how to access this register.

- **Bit 15 - URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 - Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be set to zero when UBRRH is written.

- **Bit 11:0 - UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the 4 most significant bits, and the UBRRL contains the 8 least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 36. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 88). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 36. Examples of UBRR Settings for Commonly Used Oscillator Frequencies – **UBRR = 0, Error = 0.0%**

Baud Rate (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
14.4K	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
19.2K	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
28.8K	1	8.5%	3	8.5%	3	0.0%	7	0.0%
38.4K	1	-18.6%	2	8.5%	2	0.0%	5	0.0%
57.6K	0	8.5%	1	8.5%	1	0.0%	3	0.0%
76.8K	–	–	1	-18.6%	1	-25.0%	2	0.0%
115.2K	–	–	0	8.5%	0	0.0%	1	0.0%
230.4K	–	–	–	–	–	–	0	0.0%
250K	–	–	–	–	–	–	–	–
Max	**62.5 Kbps**		**125 Kbps**		**115.2 Kbps**		**230.4 Kbps**	

Baud Rate (bps)	f_{osc} = 2.0000 MHz				f_{osc} = 3.6864 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	51	0.2%	103	0.2%	95	0.0%	191	0.0%
4800	25	0.2%	51	0.2%	47	0.0%	95	0.0%
9600	12	0.2%	25	0.2%	23	0.0%	47	0.0%
14.4K	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
19.2K	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
28.8K	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
38.4K	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
57.6K	1	8.5%	3	8.5%	3	0.0%	7	0.0%
76.8K	1	-18.6%	2	8.5%	2	0.0%	5	0.0%
115.2K	0	8.5%	1	8.5%	1	0.0%	3	0.0%
230.4K	–	–	–	–	0	0.0%	1	0.0%
250K	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M					–	–	0	-7.8%
Max	**125 Kbps**		**250 Kbps**		**230.4 Kbps**		**460.8 Kbps**	

Table 36. Examples of UBRR Settings for Commonly Used Oscillator Frequencies – **UBRR = 0, Error = 0.0%** (Continued)

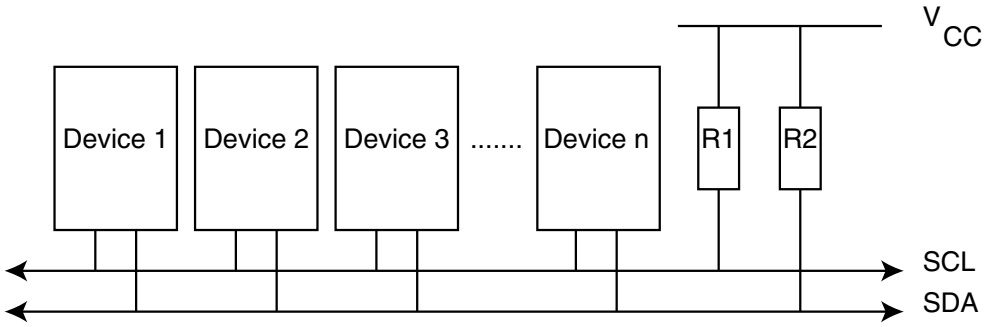
Baud Rate (bps)	f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4K	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2K	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8K	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4K	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6K	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8K	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2K	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4K	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250K	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	0	-7.8%
Max	**250 Kbps**		**0.5 Mbps**		**460.8 Kpbs**		**921.6 Kbps**	

Baud Rate (bps)	f_{osc} = 8.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%
4800	103	0.2%	207	0.2%
9600	51	0.2%	103	0.2%
14.4K	34	-0.8%	68	0.6%
19.2K	25	0.2%	51	0.2%
28.8K	16	2.1%	34	-0.8%
38.4K	12	0.2%	25	0.2%
57.6K	8	-3.5%	16	2.1%
76.8K	6	-7.0%	12	0.2%
115.2K	3	8.5%	8	-3.5%
230.4K	1	8.5%	3	8.5%
250K	1	0.0%	3	0.0%
0.5M	0	0.0%	1	0.0%
1M	-	-	0	0.0%
Max	**0.5 Mbps**		**1 Mbps**	

2-wire Serial Interface (Byte Oriented)

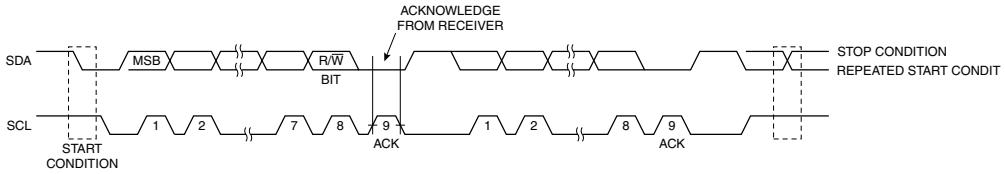
The 2-wire Serial Interface supports bi-directional serial communication. It is designed primarily for simple but efficient integrated circuit (IC) control. The system is comprised of two lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them. Various communication configurations can be designed using this bus. Figure 52 shows a typical 2-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 52. 2-wire Serial Bus Configuration



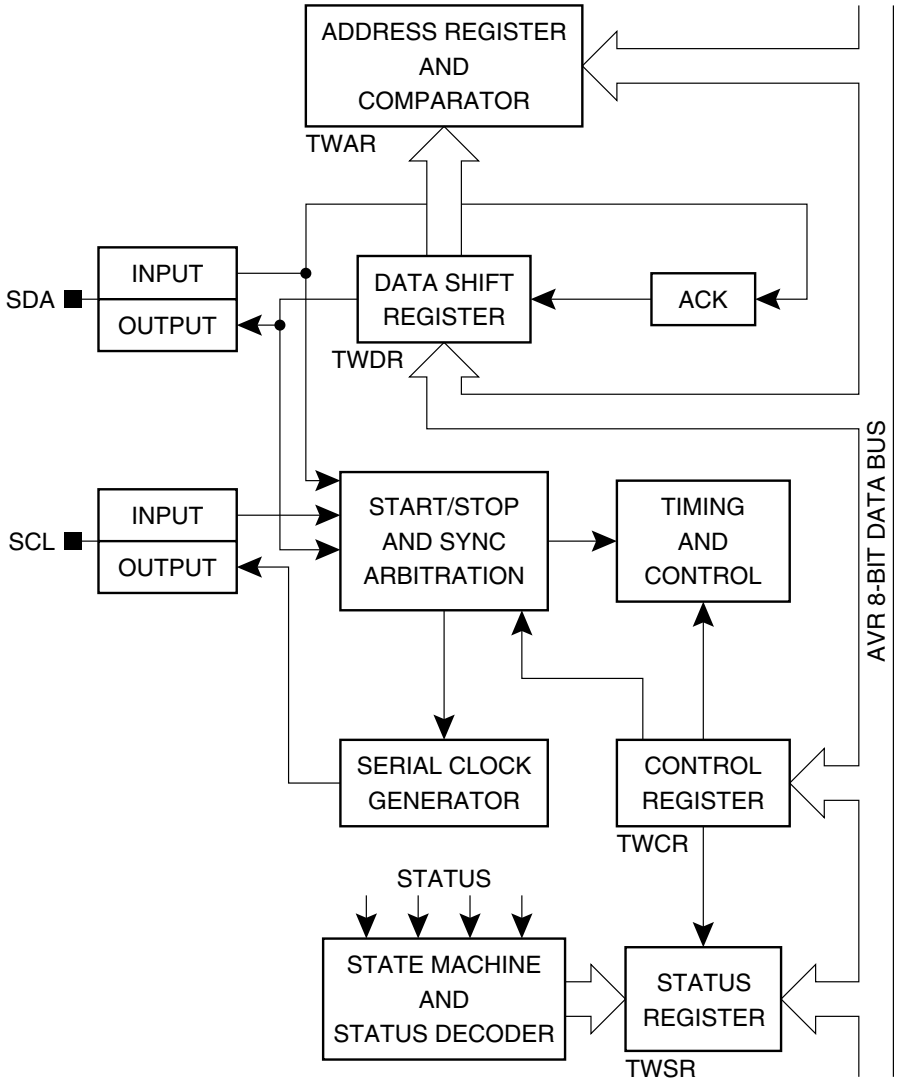
The 2-wire Serial Interface supports Master/Slave and Transmitter/Receiver operation at up to 400 kHz bus clock rate. The 2-wire Serial Interface has hardware support for 7 bit addressing. When the 2-wire Serial Interface is enabled (TWEN in TWCR is set), a glitch filter is enabled for the input signals from the pins PC0 (SCL) and PC1 (SDA), and the output from these pins is slew-rate controlled. The 2-wire Serial Interface is byte oriented. The operation of the 2-wire Serial Bus is shown as a pulse diagram in Figure 53, including the START and STOP conditions and generation of ACK signal by the bus receiver.

Figure 53. 2-wire Serial Bus Timing Diagram



The block diagram of the 2-wire Serial Interface is shown in Figure 54.

Figure 54. Block diagram of the 2-wire Serial Interface



The CPU interfaces with the 2-wire Serial Interface via the following five I/O registers: the 2-wire Serial Interface Bit Rate Register (TWBR), the 2-wire Serial Interface Control Register (TWCR), the 2-wire Serial Interface Status Register (TWSR), the 2-wire Serial Interface Data Register (TWDR), and the 2-wire Serial Interface Address Register (TWAR, used in slave mode).

The 2-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 - 2-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 2-wire Serial Interface Bit Rate Register
- t_A = Bus alignment adjustment

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 7 to ensure correct 2-wire Serial Bus functionality. The bus alignment adjustment is automatically inserted by the 2-wire Serial Interface, and ensures the validity of setup and hold times on the bus for any TWBR value higher than 7. 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 2-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: 2-wire Serial Interface Interrupt Flag**

This bit is set by hardware when the 2-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 $026. 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 automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the 2-wire Serial Interface, so all accesses to the 2-wire Serial Interface Address Register – TWAR, 2-wire Serial Interface Status Register – TWSR, and 2-wire Serial Interface Data Register – TWDR must be complete before clearing this flag.

- **Bit 6 - TWEA: 2-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 2-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 2-wire Serial Bus temporarily. Address recognition can then be resumed by setting the TWEA bit again.

- **Bit 5 - TWSTA: 2-wire Serial Bus START Condition Flag**

The TWSTA flag is set by the application when it desires to become a master on the 2-wire Serial Bus. The 2-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

2-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: 2-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 2-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 2-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: 2-wire Serial Bus Write Collision Flag**

The TWWC bit is set when attempting to write to the 2-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: 2-wire Serial Interface Enable Bit**

The TWEN bit enables 2-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 ATmega323 and will always read as zero.

- **Bit 0 - TWIE: 2-wire Serial Interface Interrupt Enable**

When this bit is enabled, and the I-bit in SREG is set, the 2-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 2-wire Serial Interface. It is used to enable the 2-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 2-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: 2-wire Serial Interface Status**

These 5 bits reflect the status of the 2-wire Serial Interface logic and the 2-wire Serial Bus.

- **Bits 2..0 - Res: Reserved bits**

These bits are reserved in ATmega323 and will always read as zero

The TWSR is read only. It contains a status code which reflects the status of the 2-wire Serial Interface logic and the 2-wire Serial Bus. There are 26 possible status codes. When TWSR contains $F8, no relevant state information is available and no 2-wire Serial Interface interrupt is requested. A valid status code is available in TWSR one CPU clock cycle after the 2-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 37 to Table 44 give the status information for the various modes.

The 2-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: 2-wire Serial Interface Data Register**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-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 writable while the 2-wire Serial Interface is not in the process of shifting a byte. This occurs when the 2-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 2-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 2-wire Serial Interface logic, the CPU cannot access the ACK bit directly.

The 2-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: 2-wire Serial Interface (Slave) Address Register**

These seven bits constitute the slave address of the 2-wire Serial Bus unit.

- **Bit 0 - TWGCE: 2-wire Serial Interface General Call Recognition Enable bit**

This bit enables, if set, the recognition of the General Call given over the 2-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 2-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 general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

2-wire Serial Interface Modes

The 2-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 55 to Figure 58. 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 55 to Figure 58, circles are used to indicate that the 2-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 2-wire Serial Bus transfer. The 2-wire Serial Bus transfer is suspended until the 2-wire Serial Interface interrupt flag is cleared by software.

The 2-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 2-wire transfer. Also note that the 2-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 2-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 37 to Table 44.

Master Transmitter Mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 55). Before master transmitter mode can be entered, the TWCR must be initialized as follows:

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

TWEN must be set to enable the 2-wire Serial Interface, TWSTA and TWSTO must be cleared.

The master transmitter mode may now be entered by setting the TWSTA bit. The 2-wire Serial Interface logic will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the 2-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 37. 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 2-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 56). 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 . 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 2-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 57). To initiate the slave receiver mode, TWAR and TWCR must be initialized as follows:

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

The upper 7 bits are the address to which the 2-wire Serial Interface will respond when addressed by a master. If the LSB is set, the 2-wire Serial Interface will respond to the general call address ($00), otherwise it will ignore the general call address.

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

TWEN must be set to enable the 2-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 2-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 2-wire Serial Interface to operate in the slave receiver mode. After its own slave address and the write bit have been received, the 2-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 42. The slave receiver mode may also be entered if arbitration is lost while the 2-wire Serial Interface is in the master mode (see states $68 and $78).

If the TWEA bit is reset during a transfer, the 2-wire Serial Interface will return a "Not Acknowledge" ("1") to SDA after the next received data byte. While TWEA is reset, the 2-wire Serial Interface does not respond to its own slave address. However, the 2-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 2-wire Serial Interface from the 2-wire Serial Bus.

In ADC Noise Reduction Mode, Power-down mode, and Power-save mode, the clock system to the 2-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 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the 2-wire Serial Interface will hold the SCL clock will low during the wake up and until the TWINT flag is cleared.

Note that the 2-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 58). The transfer is initialized as in the slave receiver mode. When TWAR and TWCR have been initialized, the 2-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 2-wire Serial Interface to operate in the slave transmitter mode. After its own slave address and the read bit have been received, the 2-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 43. The slave transmitter mode may also be entered if arbitration is lost while the 2-wire Serial Interface is in the master mode (see state $B0).

If the TWEA bit is reset during a transfer, the 2-wire Serial Interface will transmit the last byte of the transfer and enter state $C0 or state $C8. the 2-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 2-wire Serial Interface does not respond to its own slave address. However, the 2-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 2-wire Serial Interface from the 2-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 2-wire Serial Interface state, see Table 37.

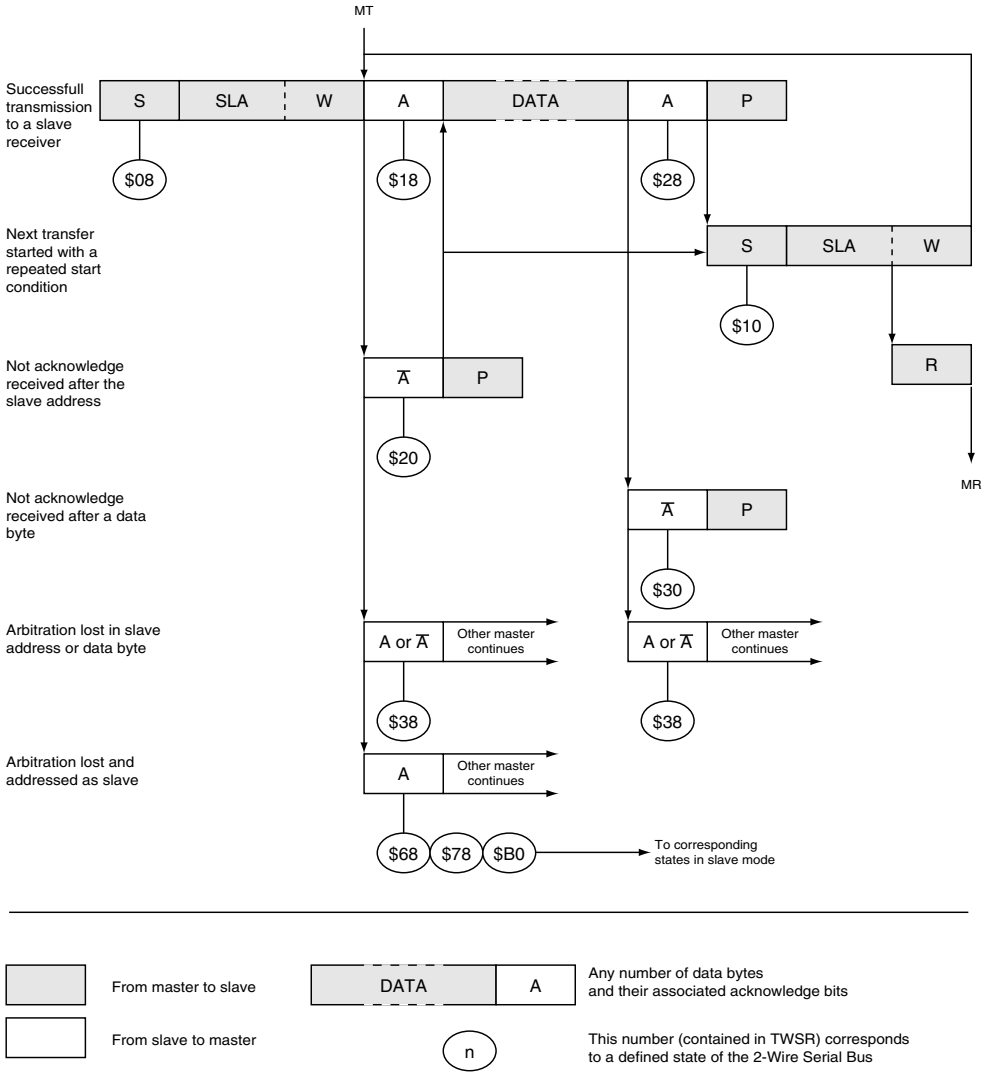
Status $F8 indicates that no relevant information is available because the 2-wire Serial Interface interrupt flag (TWINT) is not set yet. This occurs between other states, and when the 2-wire Serial Interface is not involved in a serial transfer.

Status $00 indicates that a bus error has occurred during a 2-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 2-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 37. Miscellaneous States

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by 2-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	2-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 55. Formats and States in the Master Transmitter Mode



Assembly Code Example – Master Transmitter Mode

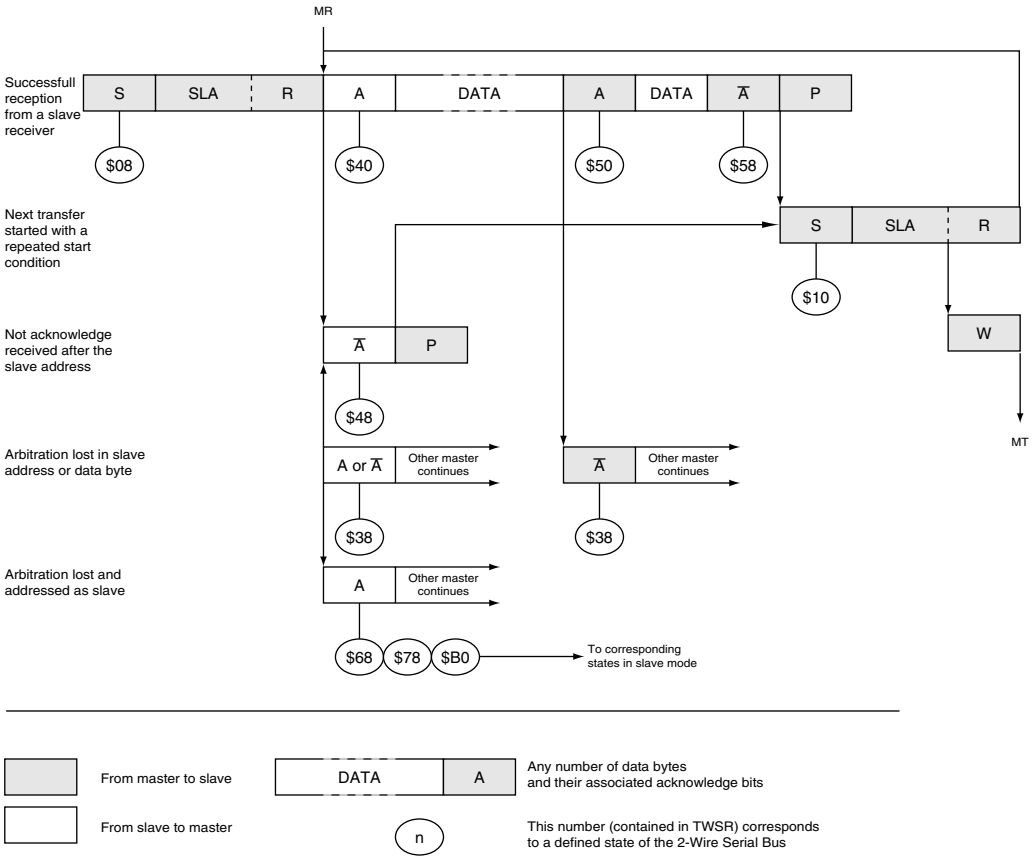
```
;The slave being addressed has address 0x64. The code examples also assumes
some sort of error handling routine named ERROR.
;Part specific include file and TWI include file must be included.

; <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 38. Miscellaneous States

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Interface hardware	Application Software Response					Next Action Taken by 2-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	2-wire Serial Bus will be released and not addressed slave mode will be entered
		No TWDR action	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 56. 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
```

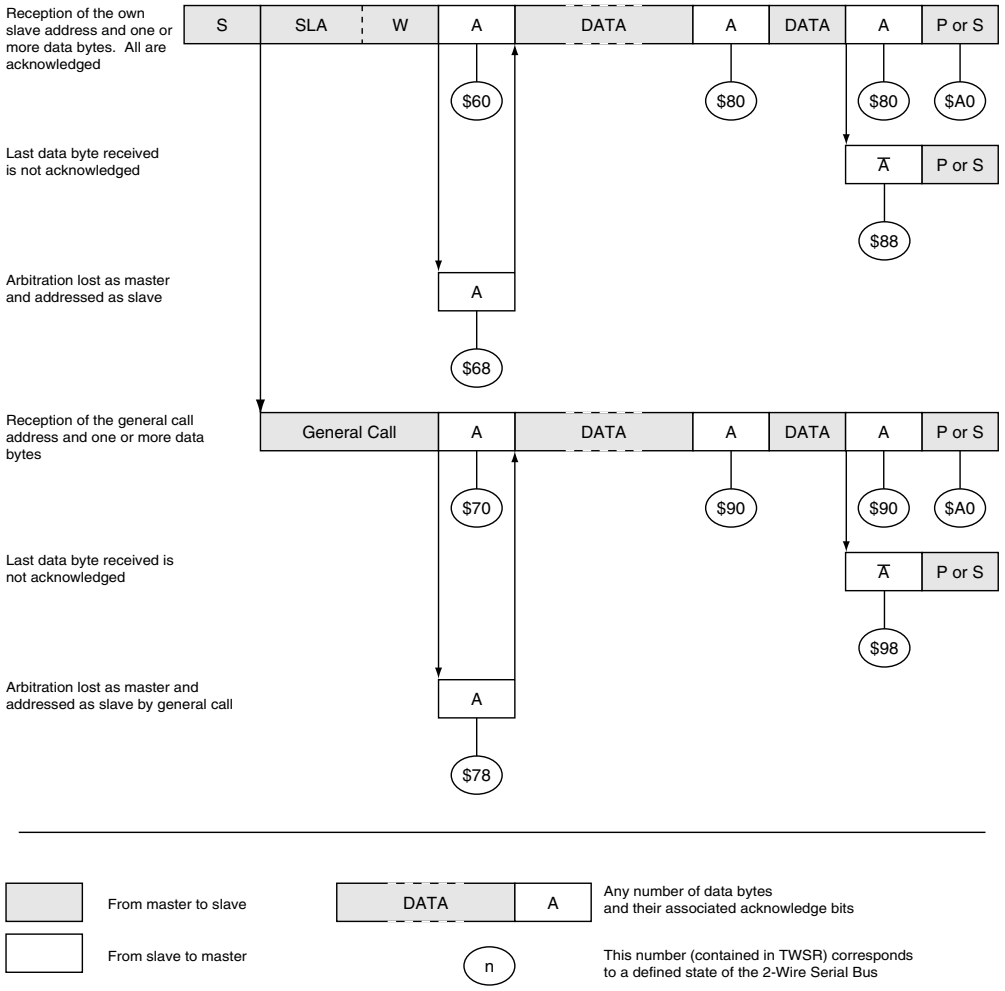
```
        in    r16, TWSR          ; Check value of TWI Status Register. If status
        cpi   r16, MR_SLA_ACK    ; different from MR_SLA_ACK, go to ERROR
        brne  ERROR

        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 39. Miscellaneous States

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by 2-wire Serial Interface Hardware
		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 57. Formats and States in the Slave Receiver Mode



Assembly Code Example – Slave Receiver Mode

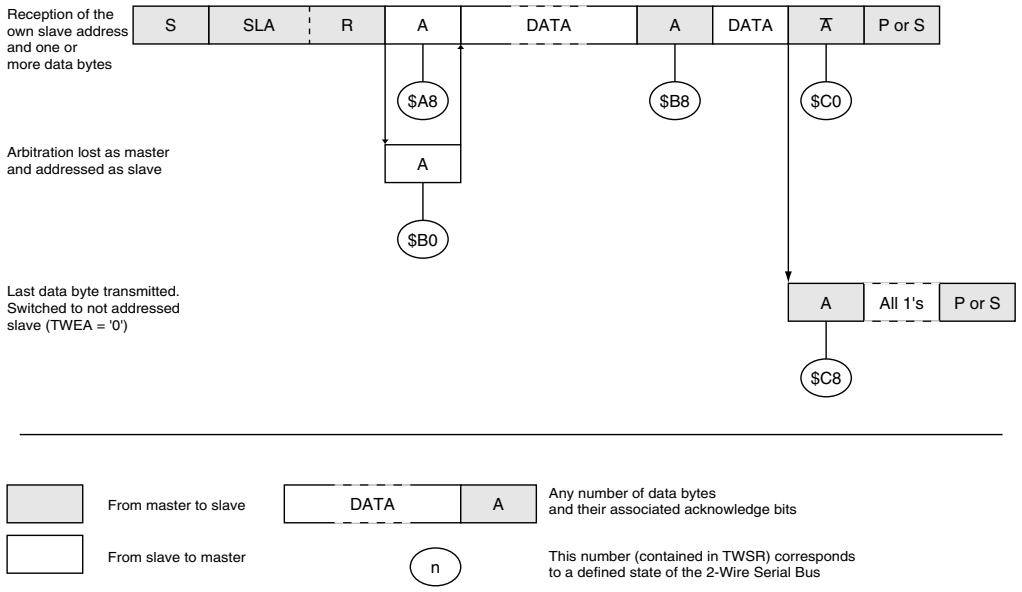
```
;Part specific include file and TWI include file must be included.
; <Initialize registers TWAR and TWBR>

      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
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 recognition of
                                ; own SLA
   ;<Wait for next data transmission or do something else>
```

Table 40. Miscellaneous States

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by 2-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 58. 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 recognition in Slave
                                 ; Transmitter mode
```

Table 41. Miscellaneous States

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by 2-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

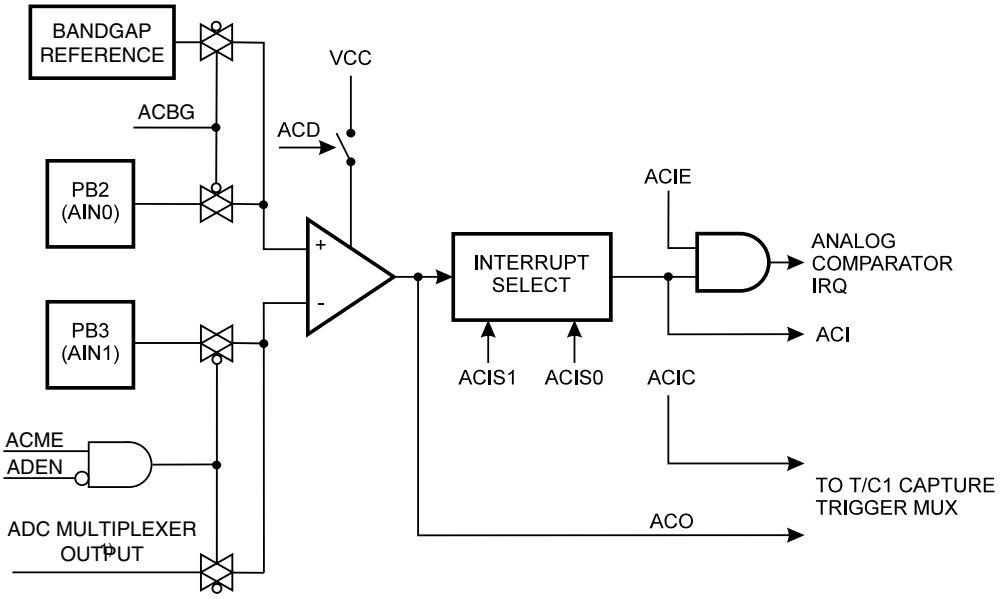
```
;***** General Master status codes *****
.equ    START           =$08    ;START has been transmitted
.equ    REP_START       =$10    ;Repeated START has been transmitted

;***** Master Transmitter status codes *****
.equ    MT_SLA_ACK      =$18    ;SLA+W has been transmitted and ACK received
.equ    MT_SLA_NACK     =$20    ;SLA+W has been transmitted and NACK received
.equ    MT_DATA_ACK     =$28    ;Data byte has been transmitted and ACK
                                ;received
```

```
.equ    MT_DATA_NACK    =$30    ;Data byte has been transmitted and NACK
                                ;received
.equ    MT_ARB_LOST     =$38    ;Arbitration lost in SLA+W or data bytes

;***** Master Receiver status codes *****
.equ    MR_ARB_LOST     =$38    ;Arbitration lost in SLA+R or NACK bit
.equ    MR_SLA_ACK      =$40    ;SLA+R has been transmitted and ACK received
.equ    MR_SLA_NACK     =$48    ;SLA+R has been transmitted 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
                                ;transmitted

;***** Slave Transmitter status 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     =$B8    ;Data byte has been transmitted and ACK
                                ;received
.equ    ST_DATA_NACK    =$C0    ;Data byte has been transmitted and NACK
                                ;received
.equ    ST_LAST_DATA    =$C8    ;Last byte in I2DR has been transmitted
                                ;(TWEA = "0"), ACK has been received

;***** Slave Receiver status 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    ;General 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
;***** Miscellaneous 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 59.

Figure 59. Analog Comparator Block Diagram



Note: See Figure 60 on page 125.

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, a fixed bandgap voltage of nominally 1.22 ± 0.10 V replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. See "Internal Voltage Reference" on page 31.

- **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 42.

Table 42. 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 potor, as shown in Table . If ACME is cleared (zero) or ADEN is set (one), PB3 (AIN1) is applied to the negative input to the Analog Comparator. t

Table 43. 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

Features

- **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 - VCC 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**

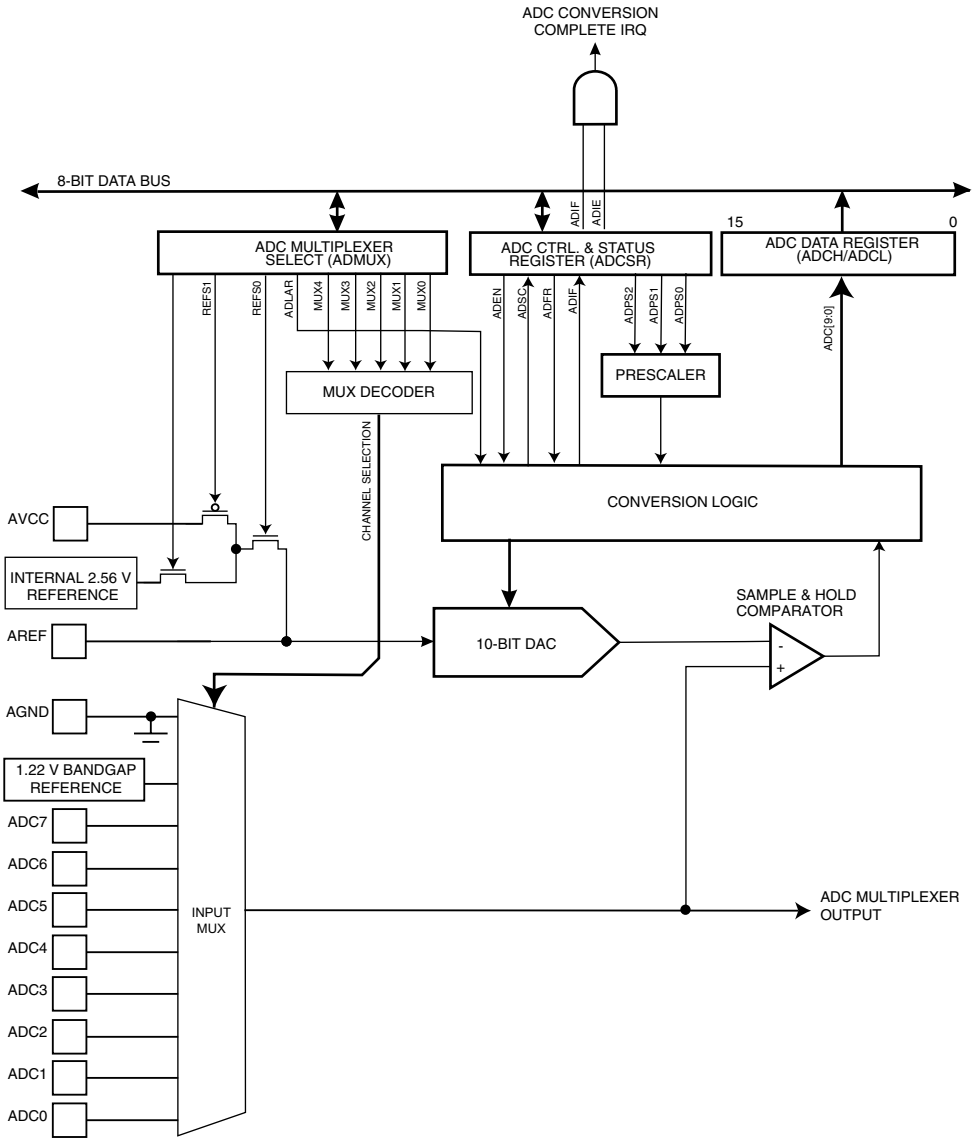
The ATmega323 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows each pin of Port A to be used as input for the ADC.

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 60.

The ADC has two separate analog supply voltage pins, AVCC and AGND. AGND must be connected to GND, and the voltage on AV_{CC} 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 AV_{CC} are provided On-chip. The 2.56V reference may be externally decoupled at the AREF pin by a capacitor for better noise performance. See “Internal Voltage Reference” on page 31 for a description of the internal voltage reference.

Figure 60. 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 1.22 V (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.

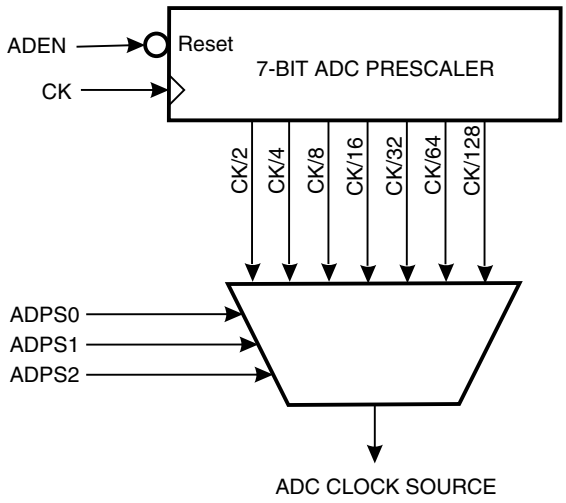
The ADC generates a 10-bit result, which are presented in the ADC data registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

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 61. 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 – Preliminary Data(1)” on page 133 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 initialization 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 initiated 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 43.

Figure 62. ADC Timing Diagram, Extended Conversion (Single Conversion Mode)

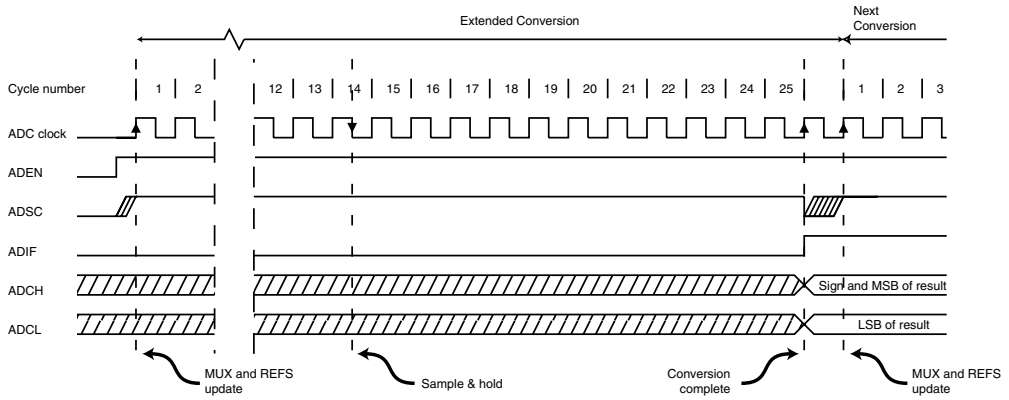


Figure 63. ADC Timing Diagram, Single Conversion

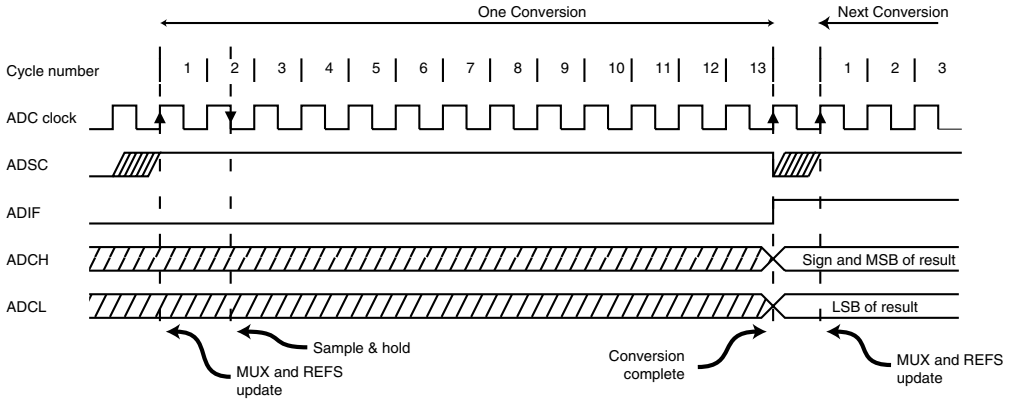


Figure 64. ADC Timing Diagram, Free Run Conversion

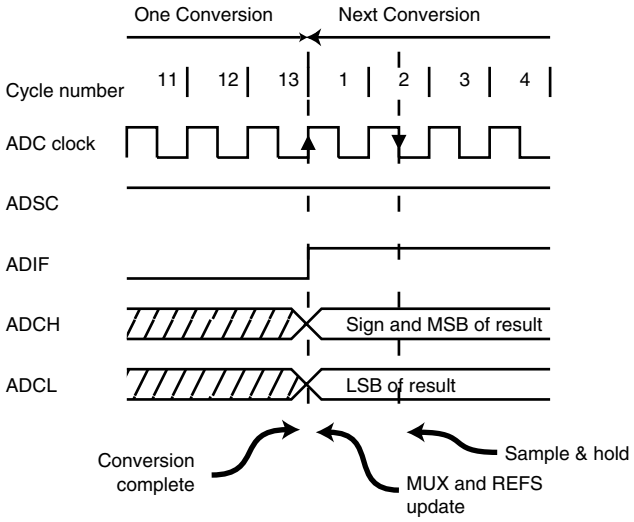


Table 44. ADC Conversion Time

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 38) 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 22. 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 45. 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 131.

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

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 46 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 46. Input Channel Selections

MUX4..0	Single-ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7
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 47. 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

Table 47. ADC Prescaler Selections (Continued)

ADPS2	ADPS1	ADPS0	Division Factor
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:

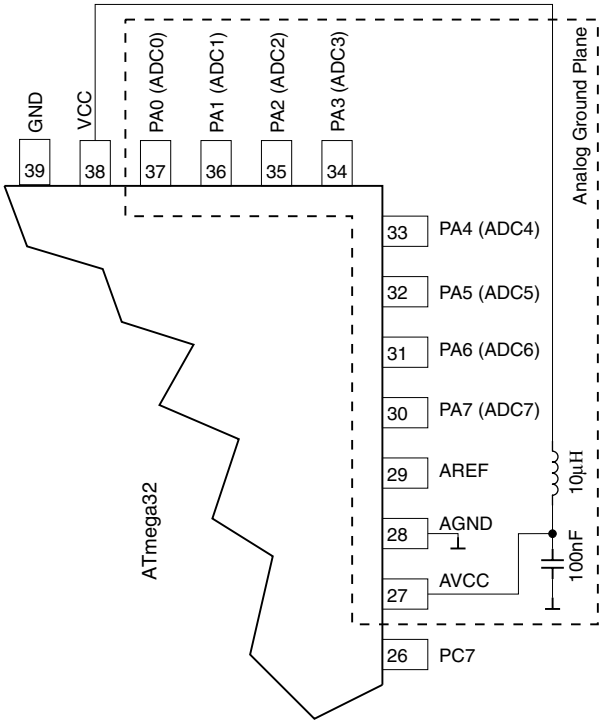
The interrupt triggers once the result is ready to be read. In Free Running Mode, the next conversion will start immediately when the interrupt triggers. If ADMUX is changed after the interrupt triggers, the next conversion has already started, and the old setting is used.

ADC Noise Canceling Techniques

Digital circuitry inside and outside the ATmega323 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 ATmega323 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 AV_{CC} pin on the ATmega323 should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 65.
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 65. ADC Power Connections



ADC Characteristics – Preliminary Data[1]

Symbol	Parameter	Condition	Min[2]	Typ	Max[3]	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[2]		V_{CC} + 0.3[3]	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. Values are guidelines only. Actual values are TBD.
2. Minimum for AV_{CC} is 2.7V.
3. Maximum for AV_{CC} 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 optional 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 power-down 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 Port A 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 48. DDAn Effects on Port A Pins

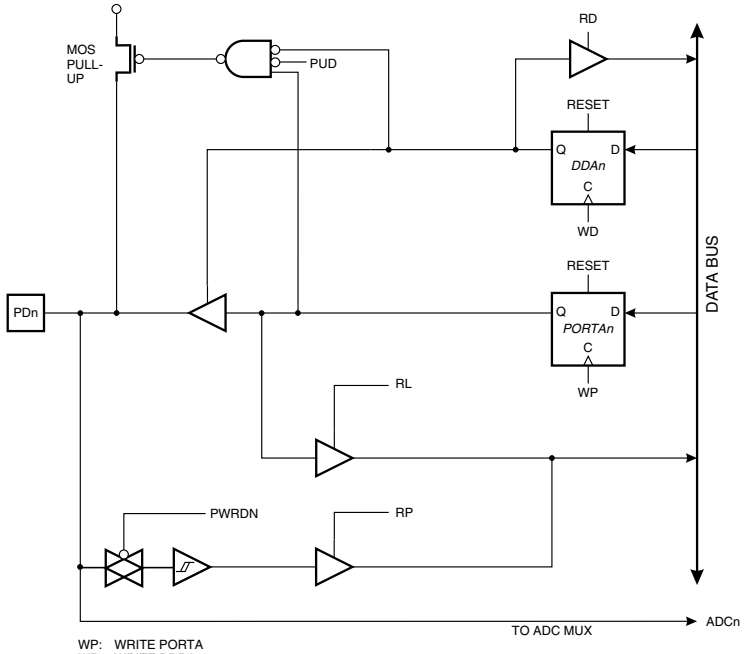
DDAn	PORTAn	PUD (in SFIOR)	I/O	Pull Up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PAn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Push-pull Zero Output
1	1	X	Output	No	Push-pull One Output

Note: 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 66. Port A Schematic Diagrams (Pins PA0 - PA7)



Port B

Port B is an 8-bit bi-directional I/O port with optional 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 49.:

Table 49. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB0	T0 (Timer/Counter 0 External Counter Input) XCK (USART External Clock Input/Output)
PB1	T1 (Timer/Counter 1 External Counter Input)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
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 Port B 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 8 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 50. DDBn Effects on Port B Pins

DDBn	PORTBn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PBn will Source Current if Ext. Pulled Low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Push-Pull Zero Output
1	1	X	Output	No	Push-Pull One Output

Note: n: 7,6…0, pin number.

Alternate Functions of Port B

The alternate pin configuration is as follows:

- **SCK - Port B, Bit 7**

SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced 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 - Port B, Bit 6**

MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced 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 - Port B, Bit 5**

MOSI: SPI Master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced 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.

- **$\overline{SS}$ - Port B, 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/OC0 - Port B, 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.

OC0, Output compare match output: The PB3 pin can serve as an external output for the Timer/Counter0 compare match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. See “8-bit Timers/Counters T/C0 and T/C2” on page 44 for further details, and how to enable the output. The OC0 pin is also the output pin for the PWM mode timer function.

- **AIN0/INT2 - Port B, 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 if INT2 is not enabled. This allows analog signals which are close to $V_{CC}/2$ to be present during power-down without causing excessive power consumption.

INT2, External Interrupt source 2: The PB2 pin can serve as an external interrupt source to the MCU. See “MCU Control and Status Register – MCUCSR” on page 30 for further details.

- **T1 - Port B, Bit 1**

T1, Timer/Counter1 counter source. See the timer description for further details.

- **T0/XCK - Port B, Bit 0**

T0, Timer/Counter0 counter source. See the timer description for further details.

XCK, USART external clock. See the USART description for further details.

Port B Schematics

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

Figure 67. Port B Schematic Diagram (Pin PB0)

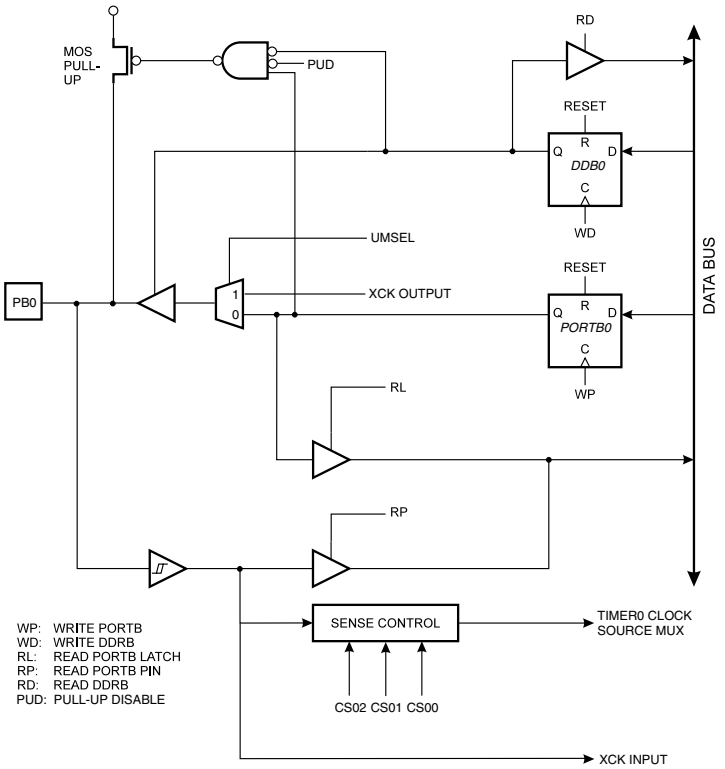


Figure 68. Port B Schematic Diagram (Pin PB1)

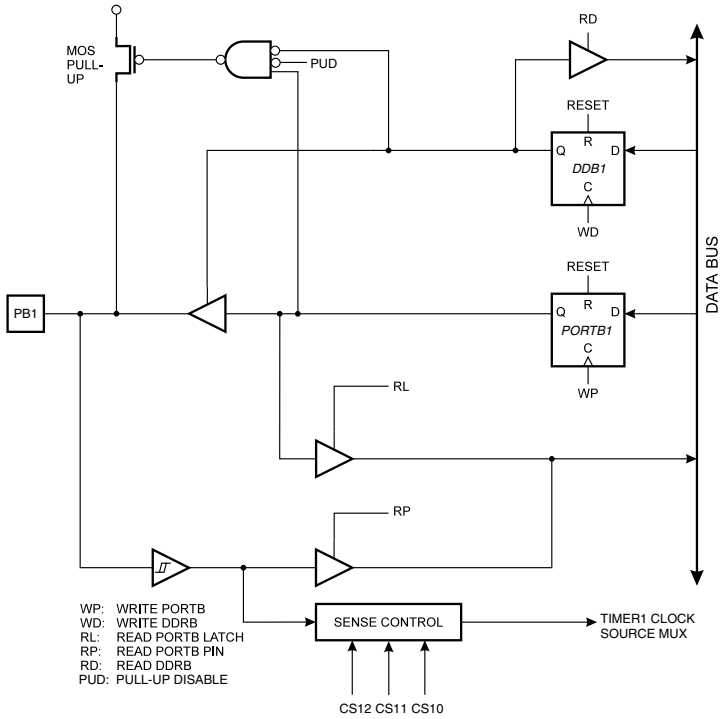


Figure 69. Port B Schematic Diagram (Pin PB2)

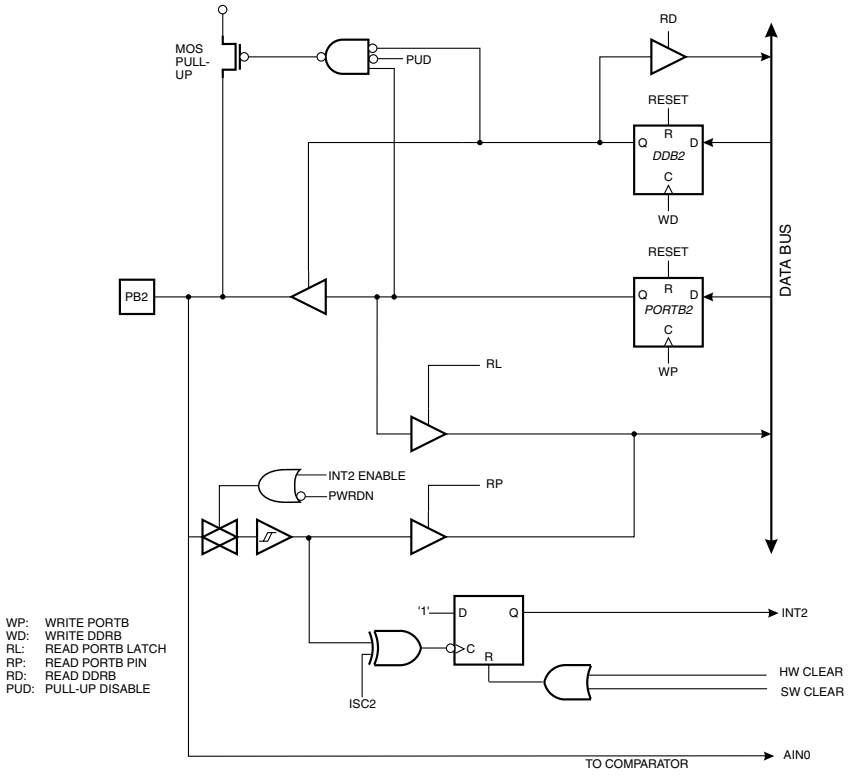


Figure 70. Port B Schematic Diagram (Pin PB3)

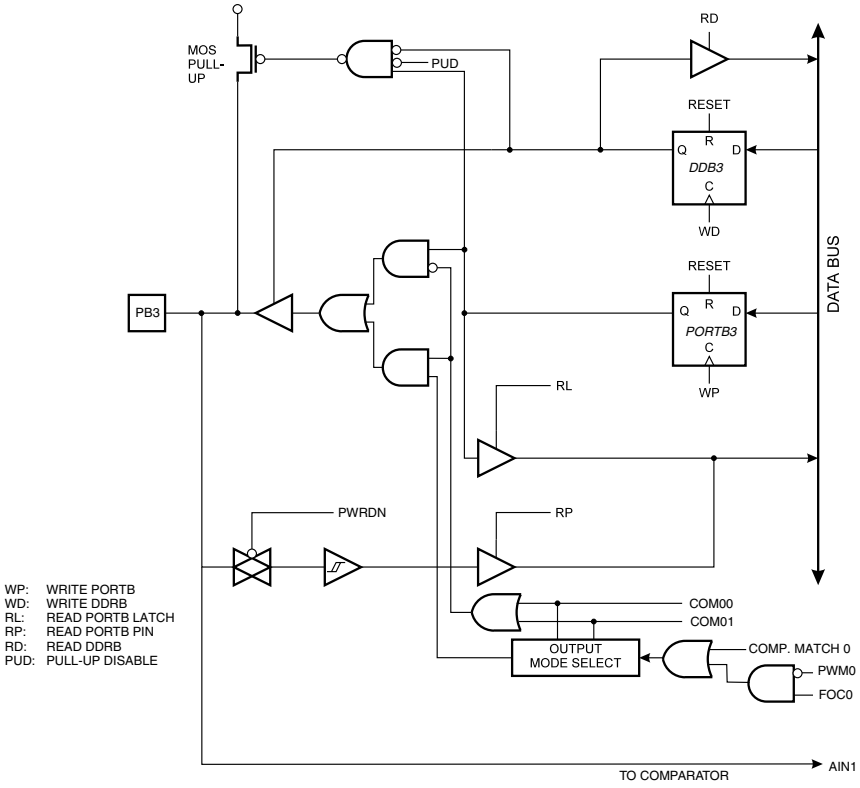


Figure 71. Port B Schematic Diagram (Pin PB4)

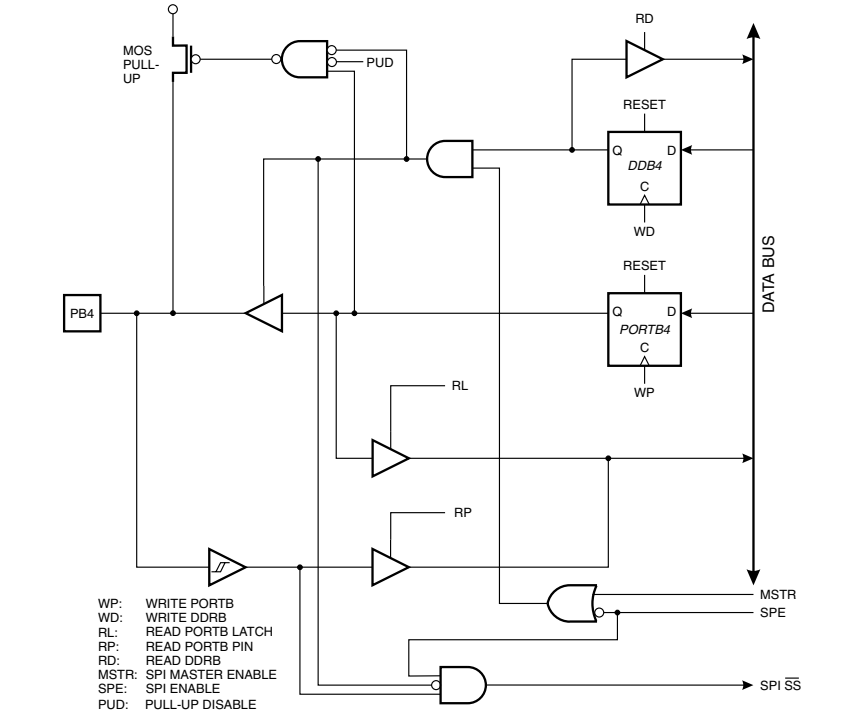


Figure 72. Port B Schematic Diagram (Pin PB5)

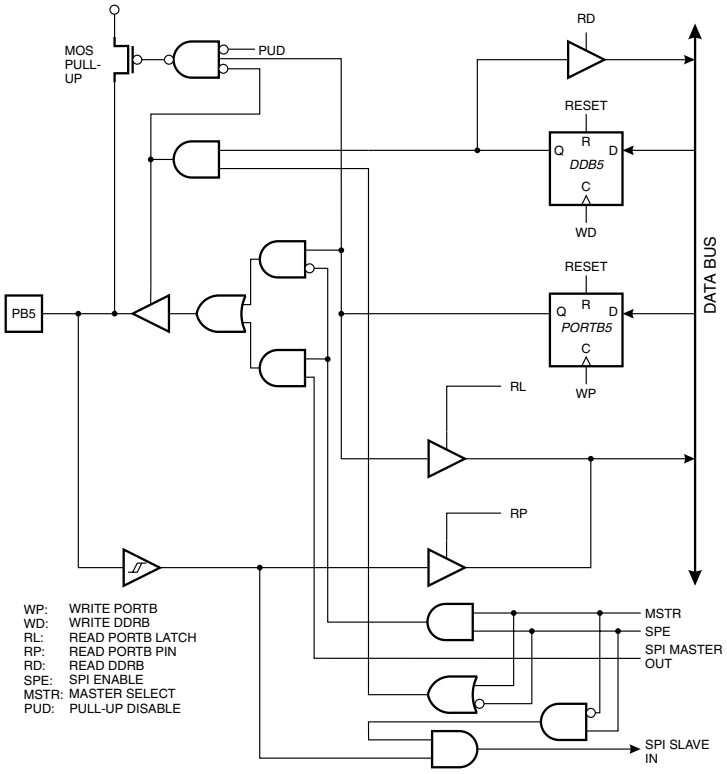


Figure 73. Port B Schematic Diagram (Pin PB6)

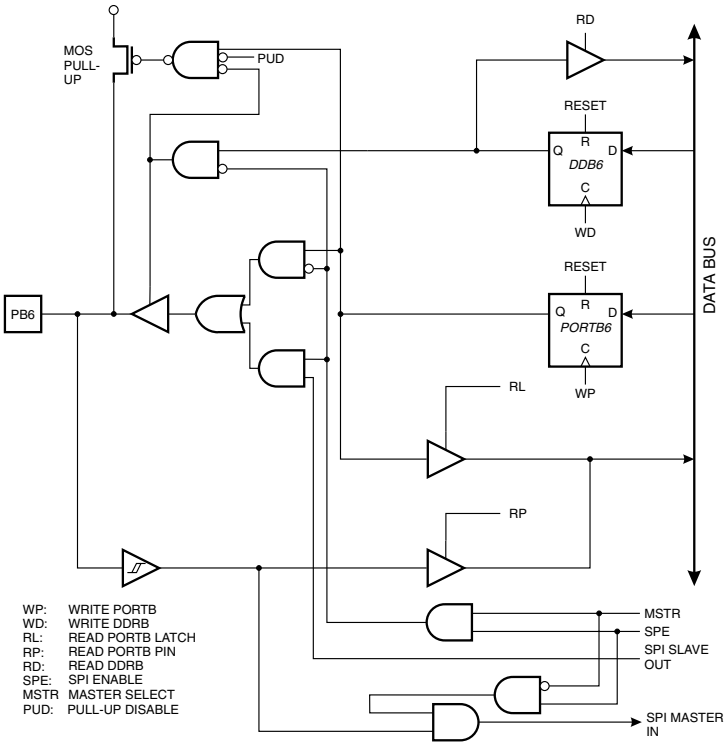
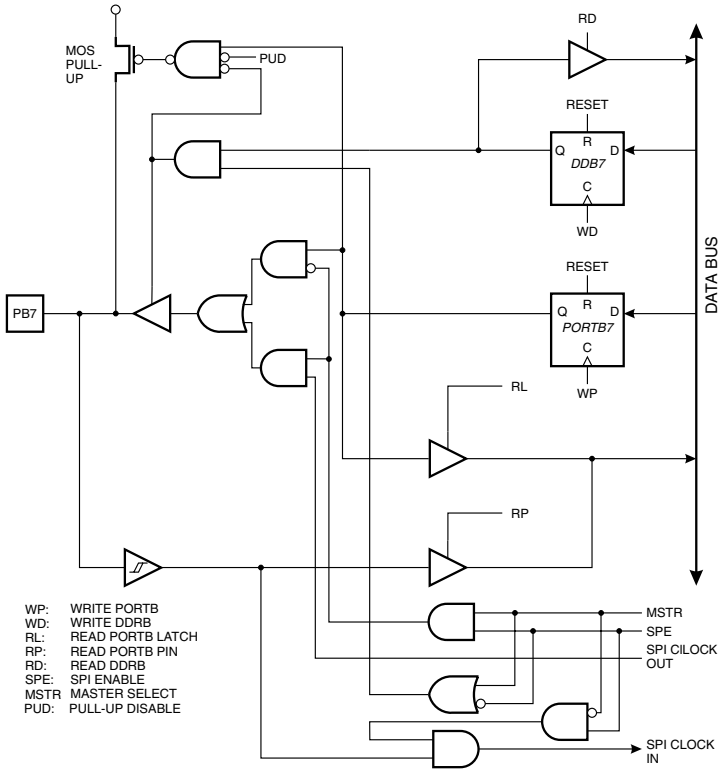


Figure 74. Port B Schematic Diagram (Pin PB7)



Port C

Port C is an 8-bit bi-directional I/O port with optional 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 20mA 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 51. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC0	SCL (2-wire Serial Bus Clock Line)
PC1	SDA (2-wire Serial Bus Data Input/Output Line)
PC2	TCK (JTAG Test Clock)
PC3	TMS (JTAG Test Mode Select)
PC4	TDO (JTAG Test Data Out)
PC5	TDI (JTAG Test Data In)
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 Port C 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 8 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 52. DDCn Effects on Port C Pins

DDCn	PORTCn	PUD (in SFIOR)	I/O	Pull-up	Comments
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PCn will Source Current if Ext. Pulled Low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Push-pull Zero Output
1	1	X	Output	No	Push-pull One Output

Note: n: 7…0, pin number

If the JTAG interface is enabled, the pull-up resistors on pins PC5 (TDI), PC3 (TMS) and PC2 (TCK) will be activated even if a reset occurs.

Alternate Functions of Port C

- **TOSC2 - Port C, 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 - Port C, 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.

- **TDI - Port C, Bit 5**

TDI, JTAG Test Data In: Serial input data to be shifted in to the Instruction Register or Data Register (scan chains). When the JTAG interface is enabled, this pin can not be used as an I/O pin. Refer to the section “JTAG Interface and the On-chip Debug System” on page 153 for details on operation of the JTAG interface.

- **TDO - Port C, Bit 4**

TDO, JTAG Test Data Out: Serial output data from Instruction register or Data Register. When the JTAG interface is enabled, this pin can not be used as an I/O pin. Refer to the section “JTAG Interface and the On-chip Debug System” on page 153 for details on operation of the JTAG interface.

- **TMS - Port C, Bit 3**

TMS, JTAG Test Mode Select: This pin is used for navigating through the TAP-controller state machine. When the JTAG interface is enabled, this pin can not be used as an I/O pin. Refer to the section “JTAG Interface and the On-chip Debug System” on page 153 for details on operation of the JTAG interface.

- **TCK - Port C, Bit 2**

TCK, JTAG Test Clock: JTAG operation is synchronous to TCK. When the JTAG interface is enabled, this pin can not be used as an I/O pin. Refer to the section “JTAG Interface and the On-chip Debug System” on page 153 for details on operation of the JTAG interface.

- **SDA - Port C, Bit 1**

SDA, 2-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the 2-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.

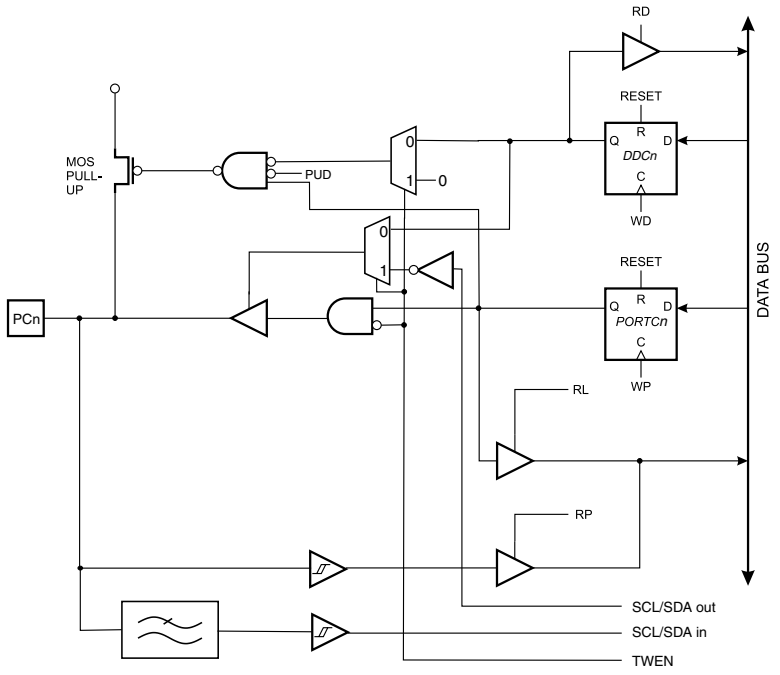
- **SCL - Port C, Bit 0**

SCL, 2-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Clock I/O pin for the 2-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

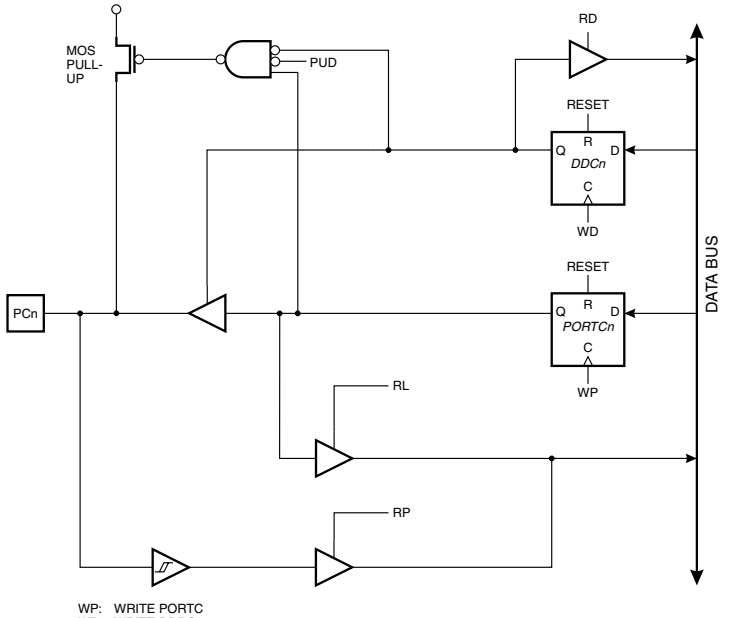
Note that all port pins are synchronized. The synchronization latches are not shown in the figure.

Figure 75. Port C Schematic Diagram (Pins PC0 - PC1)



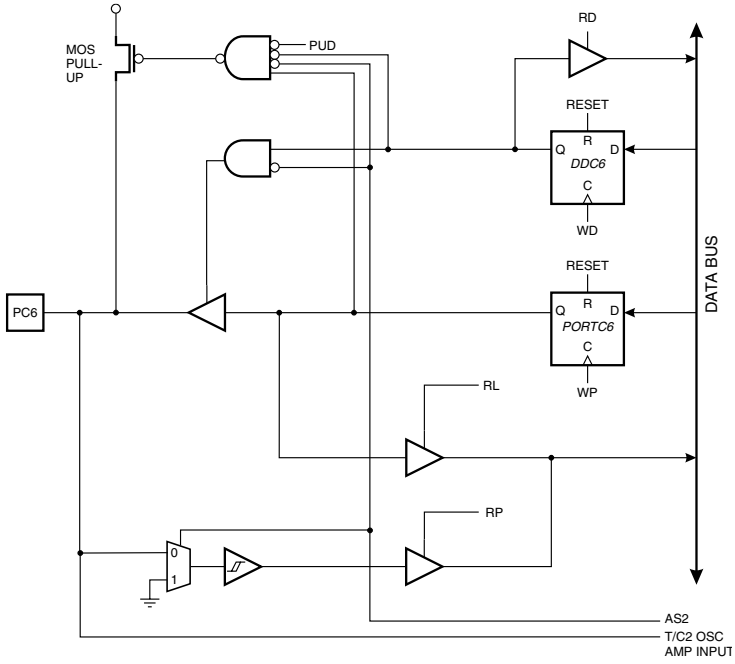
WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
PUD: PULL-UP DISABLE
n = 0, 1

Figure 76. Port C Schematic Diagram (Pins PC2 - PC5). The JTAG interface on these pins is not shown in the figure.



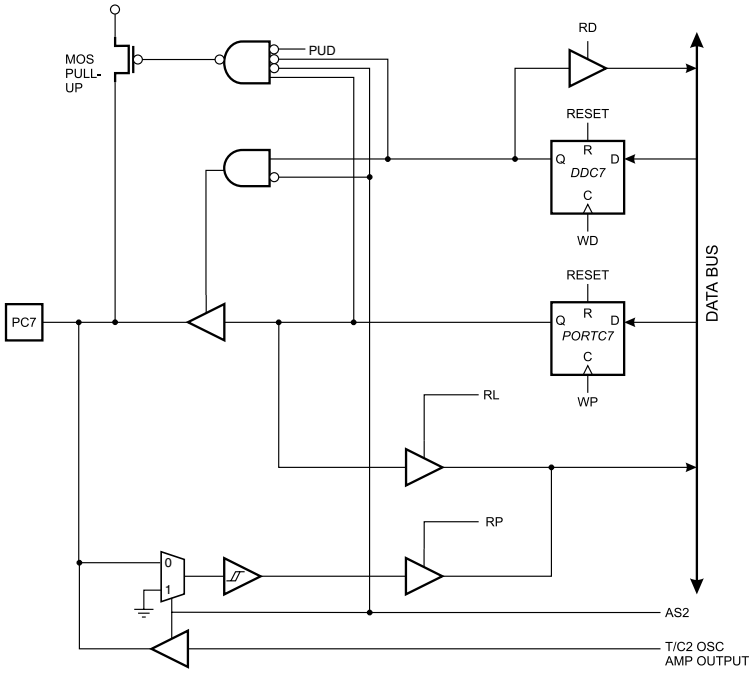
WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
PUD: PULL-UP DISABLE
n: 2..5

Figure 77. Port C 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 78. Port C 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 optional 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 53.

Table 53. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RXD (USART Input Pin)
PD1	TXD (USART 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 Port D 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 54. DDDn Effects on Port D Pins

DDDn	PORTDn	PUD (in SFIOR)	I/O	Pull Up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PDn will Source Current if Ext. Pulled Low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Push-pull Zero Output
1	1	X	Output	No	Push-pull One Output

Note: n: 7,6…0, pin number.

Alternate Functions of Port D

- **OC2 - Port D, 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 - Port D, 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 - Port D, Bit 5**

OC1A, Output compare matchA output: The PD5 pin can serve as an external output for the Timer/Counter1 output compareA. 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 - Port D, Bit 4**

OC1B, Output compare matchB output: The PD4 pin can serve as an external output for the Timer/Counter1 output compareB. 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 - Port D, 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 - Port D, 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 USART). When the USART transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

- **RXD - Port D, Bit 0**

RXD, Receive Data (Data input pin for the USART). When the USART receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART 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 79. Port D Schematic Diagram (Pin PD0)

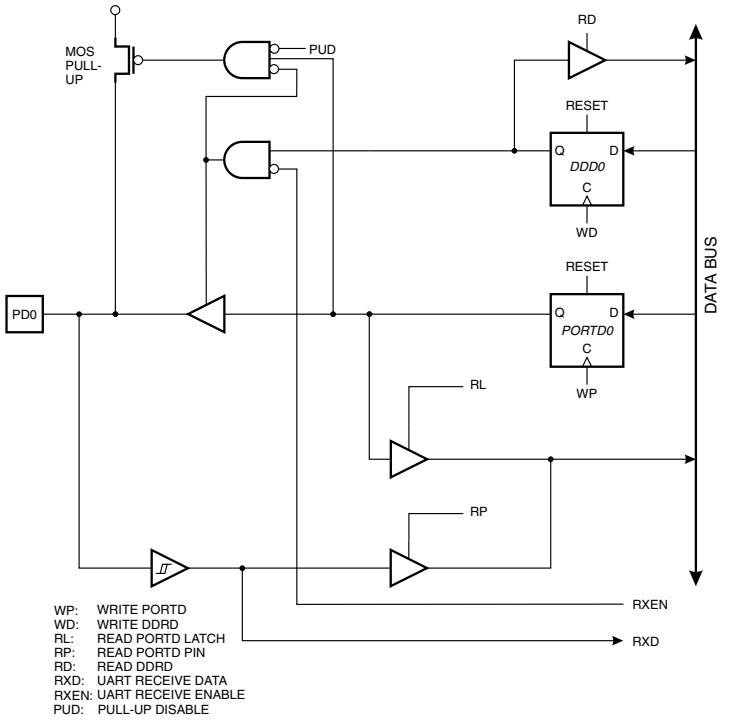


Figure 80. Port D Schematic Diagram (Pin PD1)

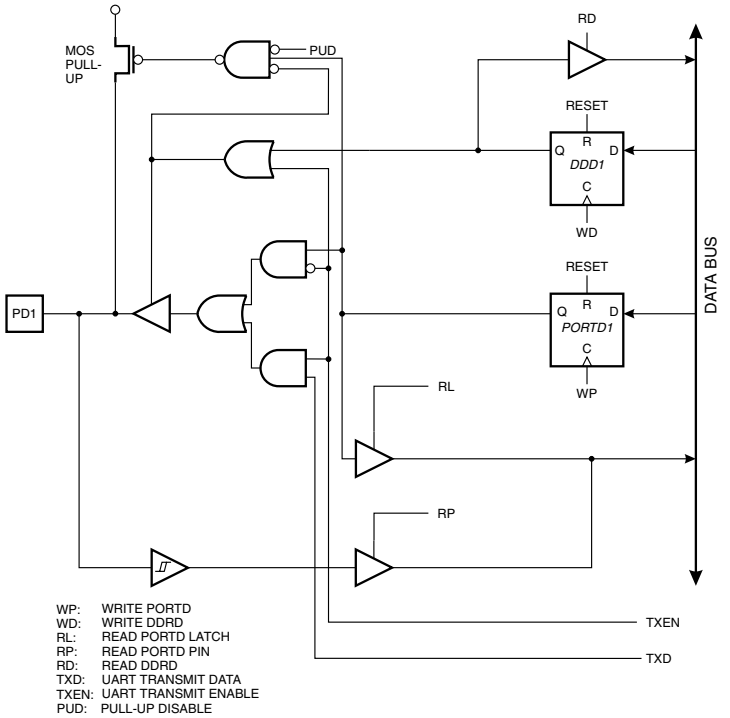


Figure 81. Port D Schematic Diagram (Pins PD2 and PD3)

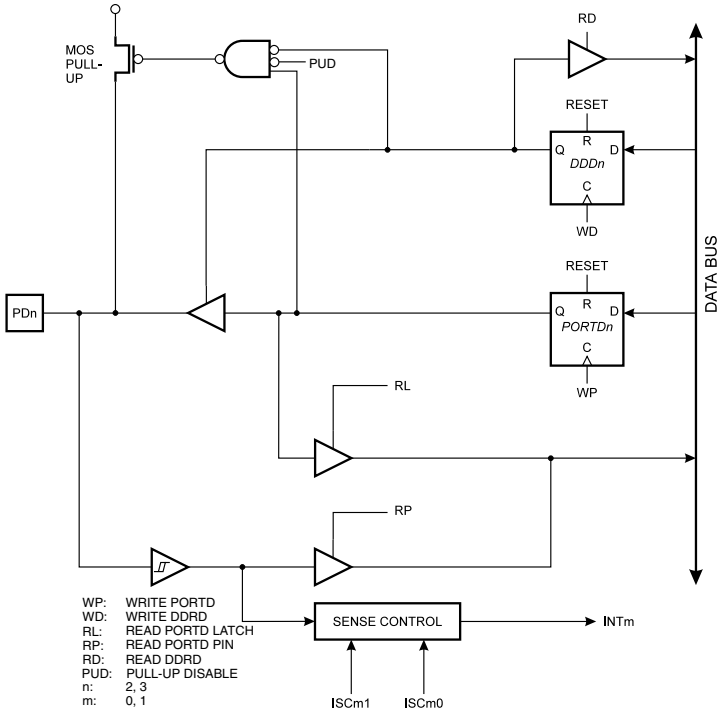


Figure 82. Port D Schematic Diagram (Pins PD4 and PD5)

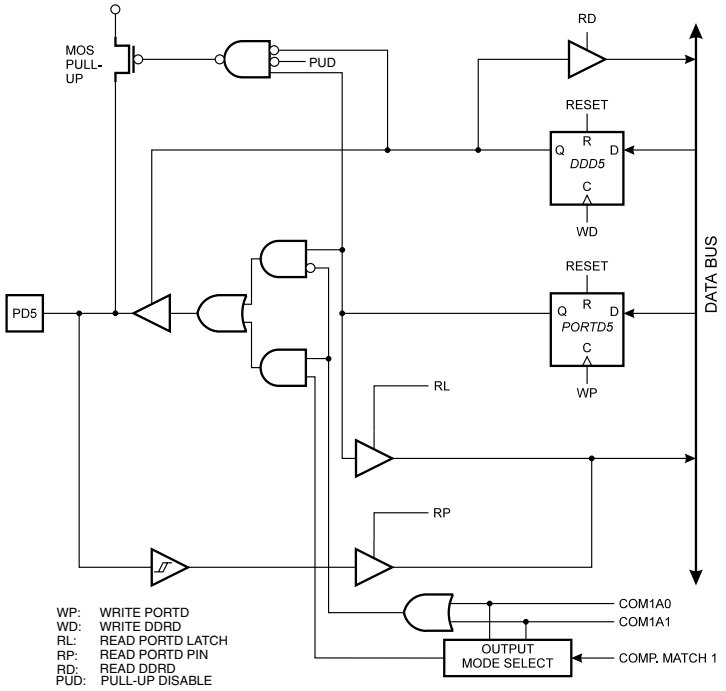


Figure 83. Port D Schematic Diagram (Pin PD6)

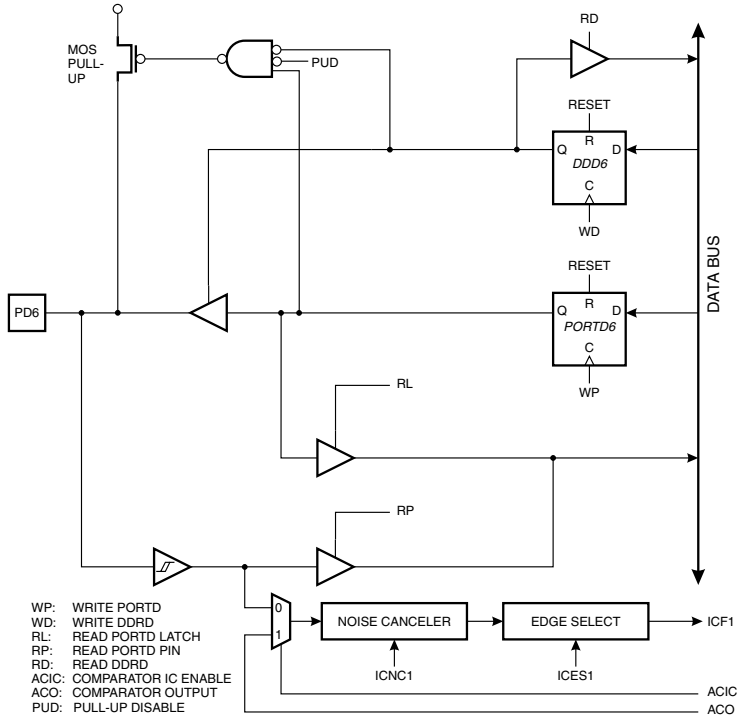
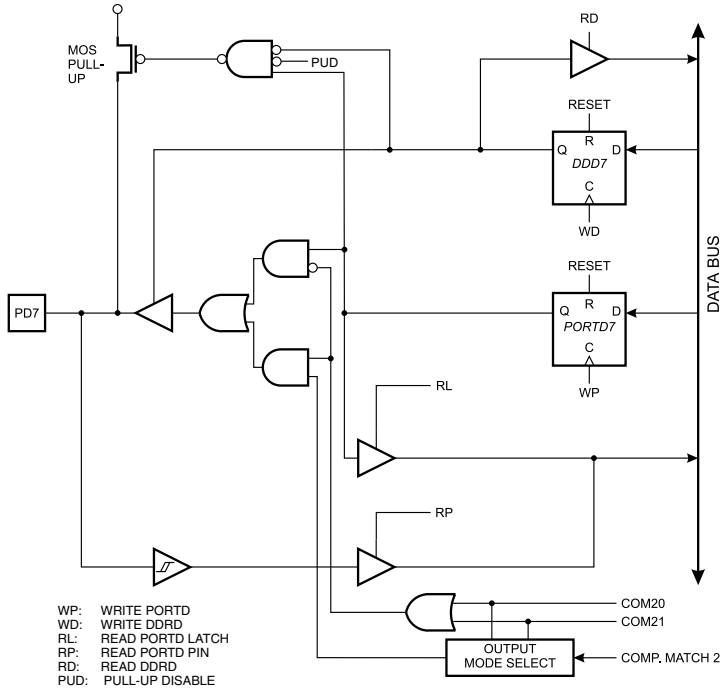


Figure 84. Port D Schematic Diagram (Pin PD7)



JTAG Interface and the On-chip Debug System

Features:

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **Boundary-scan Capabilities According to the JTAG Standard**
- **Debugger Access to:**
  - **All Internal Peripheral Units**
  - **Internal and External RAM**
  - **The Internal Register File**
  - **Program Counter**
  - **EEPROM and Flash Memories**
- **Extensive On-chip Debug Support for Break Conditions, Including**
  - **Break on Change of Program Memory Flow**
  - **Single Step Break**
  - **Program Memory Breakpoints on Single Address or Address Range**
  - **Data Memory Breakpoints on Single Address or Address Range**
- **Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface**
- **On-chip Debugging Supported by AVR Studio**

Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

- Testing PCBs by using the JTAG Boundary-scan Capability
- Programming the Non-volatile Memories, Fuses and Lock-bits
- On-chip Debugging

A brief description is given in the following sections. Detailed descriptions for Programming via the JTAG interface, and using the Boundary-Scan Chain can be found in the sections “Programming via the JTAG interface” on page 196 and “IEEE 1149.1 (JTAG) Boundary-scan” on page 159**,** respectively. The On-Chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected 3rd party vendors only.

Figure 85 shows a block diagram of the JTAG interface and the On-Chip Debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (shift register) between the TDI - input and TDO - output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

Of the Data Registers, the ID-Register, Bypass Register, and the Boundary-Scan Chain are used for board-level testing. The JTAG Programming Interface (actually consisting of several physical and virtual Data Registers) is used for serial programming via the JTAG interface. The Internal Scan Chain and Break-Point Scan Chain are used for On-Chip debugging only.

The Test Access Port – TAP

The JTAG interface is accessed through four of the AVR’s pins. In JTAG terminology, these pins constitute the Test Access Port - TAP. These pins are

- TMS: Test mode select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test clock. JTAG operation is synchronous to TCK
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains)
- TDO: Test Data Out. Serial output data from Instruction register or Data Register

The IEEE std. 1149.1 also specifies an optional TAP signal; TRST - Test ReSeT - which is not provided.

When the JTAGEN fuse is unprogrammed, these four TAP pins are normal port pins, and the TAP controller is in reset. When programmed, the input TAP signals are internally pulled high and the JTAG is enabled for Boundary-Scan and programming. The device is shipped with this fuse programmed.

For the On-Chip Debug system, in addition the $\overline{\text{RESET}}$ pin is monitored by the debugger to be able to detect external reset sources. The debugger can also pull the $\overline{\text{RESET}}$ pin low to reset the whole system, assuming only open collectors on reset line are used in the application.

Figure 85. Block Diagram

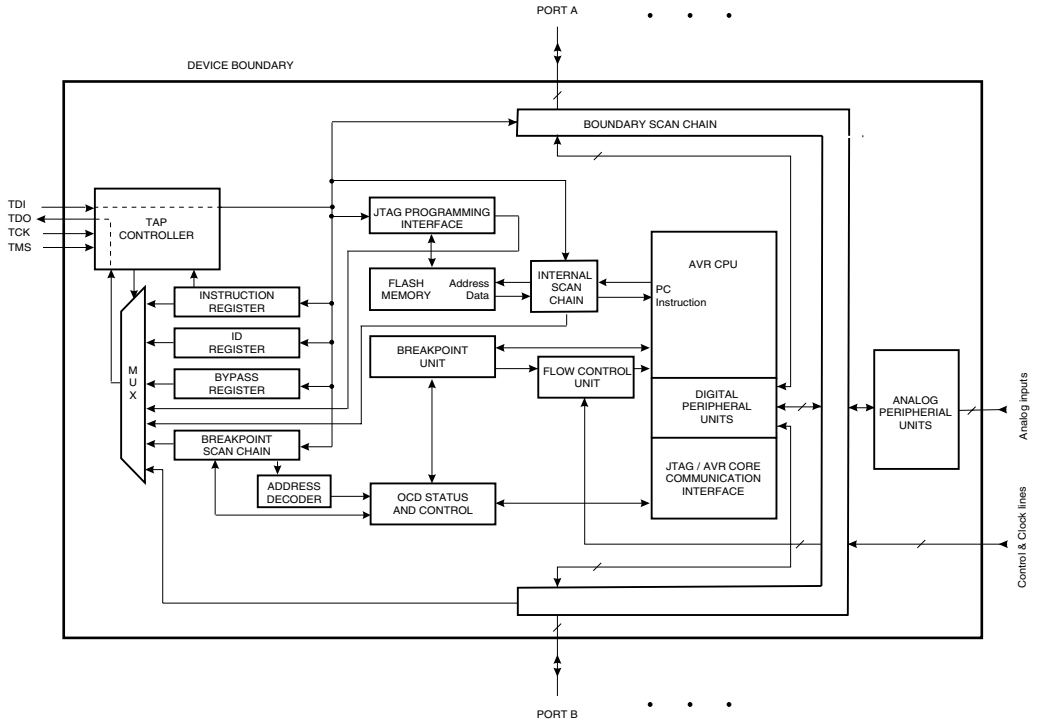
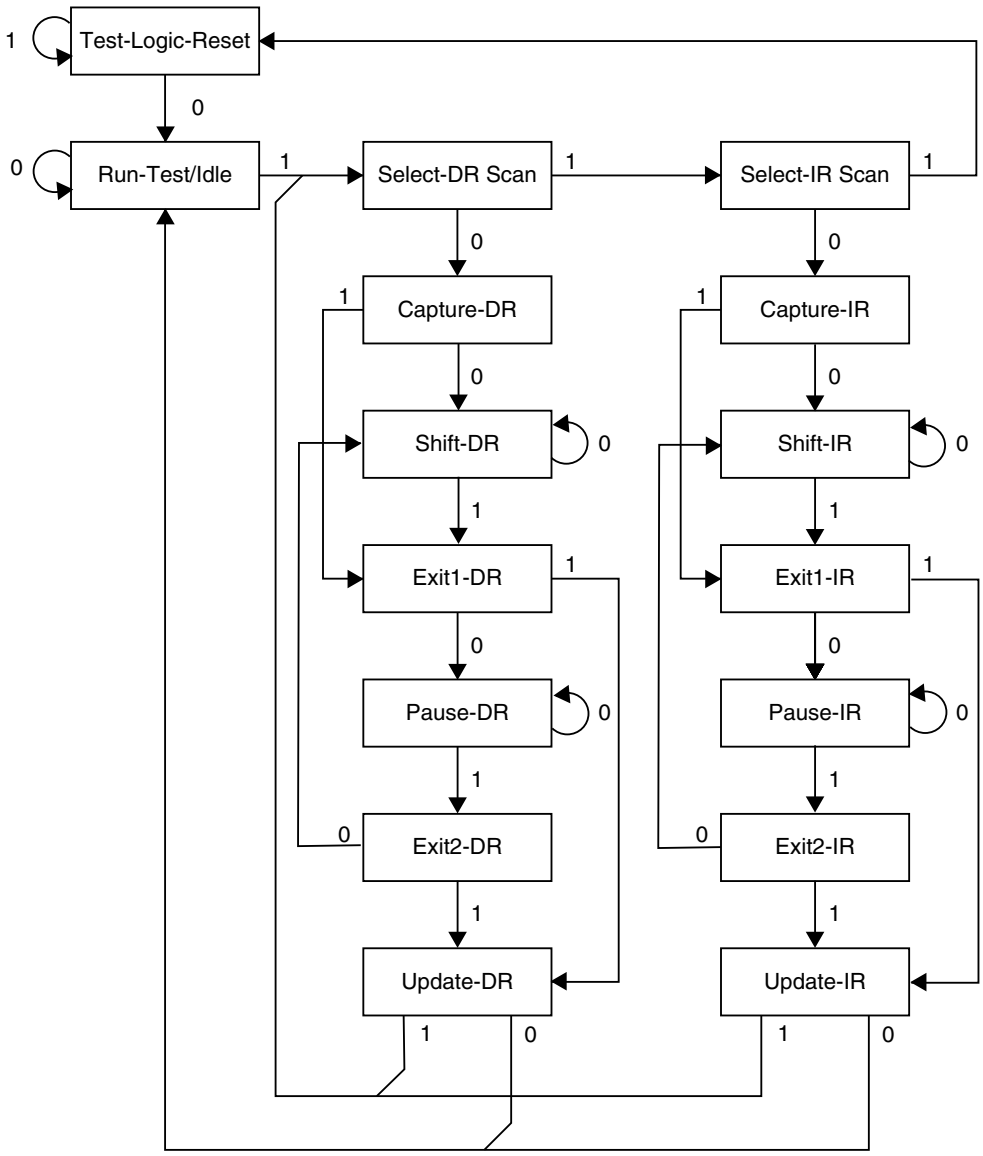


Figure 86. TAP Controller State Diagram



TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-Scan circuitry, JTAG programming circuitry, or On-Chip Debug system. The state transitions depicted in Figure 86 depends on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-On Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all shift registers.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register - Shift-IR state. While TMS is low, shift the 4 bit JTAG instructions into the JTAG instruction register from the TDI input at the rising edge of TCK, while the captured IR-state 0x01 is shifts out on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the shift register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register - Shift-DR state. While TMS is low, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. At the same time, the parallel inputs to the Data Register captured in the Capture-DR state shifts out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for 5 TCK clock periods.

For detailed information on the JTAG specification, refer to the literature listed in “Bibliography” on page 158.

Using the Boundary-scan Chain

A complete description of the Boundary-Scan capabilities are given in the section “IEEE 1149.1 (JTAG) Boundary-scan” on page 159.

Using the On-chip Debug System

As shown in Figure 85, the hardware support for On-Chip Debugging consists mainly of

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- Breakpoint unit
- Communication interface between the CPU and JTAG system

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Breakpoint Unit implements Break on Change of Program Flow, Single Step Break, 2 Program Memory Breakpoints, and 2 combined break points. Together, the 4 break-points can be configured as either:

- 4 single Program Memory break-points
- 3 Single Program Memory break point + 1 single Data Memory break point
- 2 single Program Memory break-points + 2 single Data Memory break points
- 2 single Program Memory break-points + 1 Program Memory break point with mask (‘range break point’)
- 2 single Program Memory break-points + 1 Data Memory break point with mask (‘range break point’)

A list of the On-Chip Debug specific JTAG instructions is given in “On-chip Debug Specific JTAG Instructions” on page 157. Note that Atmel supports the On-Chip Debug system with the AVR Studio front-end software for PCs. The details on hardware imple-

mentation and JTAG instructions are therefore irrelevant for the user of the On-Chip Debug system.

The JTAGEN fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN fuse must be programmed and no lock bits must be set for the On-Chip debug system to work. The disabling of the On-Chip debug system when any lock bits are set is a security feature. Otherwise, the On-Chip debug system would have provided a back-door into a secured device.

The AVR Studio enables the user to fully control execution of programs on an AVR device with On-Chip Debug capability, AVR In-Circuit Emulator, or the built-in AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly programs assembled with Atmel Corporation's AVR Assembler and C programs compiled with 3rd party vendors' compilers.

AVR Studio runs under Microsoft Windows 95/98/2000 and Microsoft Windows NT.

For a full description of the AVR Studio, please refer to the **AVR Studio User Guide**. Only highlights are presented in this document.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have up to 2 data memory breakpoints, alternatively combined as a mask (range) break-point.

On-chip Debug Specific JTAG Instructions

The On-Chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected 3rd party vendors only. Instruction opcode listed for reference.

PRIVATE0; $8 Private JTAG instruction for accessing On-Chip debug system.

PRIVATE1; $9 Private JTAG instruction for accessing On-Chip debug system.

PRIVATE2; $A Private JTAG instruction for accessing On-Chip debug system.

PRIVATE3; $B Private JTAG instruction for accessing On-Chip debug system.

Using the JTAG Programming Capabilities

Programming of AVR parts via JTAG is performed via the four-pin JTAG port, TCK, TMS, TDI and TDO. These are the only pins that need to be controlled/observed to perform JTAG programming (in addition to power pins). It is not required to apply 12V externally. The JTAGEN fuse must be programmed and the JTD bit in the MCUSR register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- Flash programming and verifying
- EEPROM programming and verifying
- Fuse programming and verifying
- Lock bit programming and verifying

The lock bit security is exactly as in parallel programming mode. If the lock-bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

A description of the programming specific JTAG instructions is given in “Programming specific JTAG instructions” on page 196. The details on programming through the JTAG interface is given in the section “Programming via the JTAG interface” on page 196

Bibliography

For more information about general Boundary-Scan, the following literature can be consulted:

- IEEE: IEEE Std 1149.1-1990. IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE, 1993
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992

IEEE 1149.1 (JTAG) Boundary-scan

Features

- **JTAG (IEEE std. 1149.1 compliant) Interface**
- **Boundary-scan Capabilities According to the JTAG Standard**
- **Full Scan of All Port Functions**
- **Supports the Optional IDCODE Instruction**
- **Additional Public AVR_RESET Instruction to Reset the AVR**

System Overview

The Boundary-Scan chain has the capability of driving and observing the logic levels on the digital I/O pins. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long shift register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-Scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the 4 TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the data register path will show the ID-code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the test mode. Entering reset, the outputs of any Port Pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external $\overline{\text{RESET}}$ pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

The JTAGEN fuse must be programmed and the JTD bit in the I/O register MCUSR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-Scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

Data Registers

The Data Registers are selected by the JTAG instruction registers described in section “Boundary-scan Specific JTAG Instructions” on page 161. The data registers relevant for Boundary-Scan operations are:

- Bypass Register
- Device Identification Register
- Reset Register
- Boundary-Scan Chain

Bypass Register

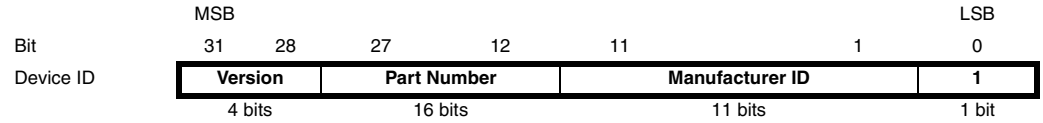
The Bypass register consists of a single shift-register stage. When the Bypass register is selected as path between TDI and TDO, the register is reset to 0 when leaving the

Capture-DR controller state. The Bypass register can be used to shorten the scan chain on a system when the other devices are to be tested.

Device Identification Register

Figure 87 shows the structure of the Device Identification register.

Figure 87. The format of the Device Identification register



Version

Version is a 4 bit number identifying the revision of the component. The relevant version numbers are shown in Table 55.

Table 55. JTAG Version Numbers

Version	JTAG Version number (Binary digits)
ATmega323 revision B	0010

Part Number

The part number is a 16 bit code identifying the component. The JTAG Part Number for AVR devices are listed in Table 56.

Table 56. AVR JTAG Part Number

Part number	JTAG Part Number (Hex)
ATmega323	0x9501

Manufacturer ID

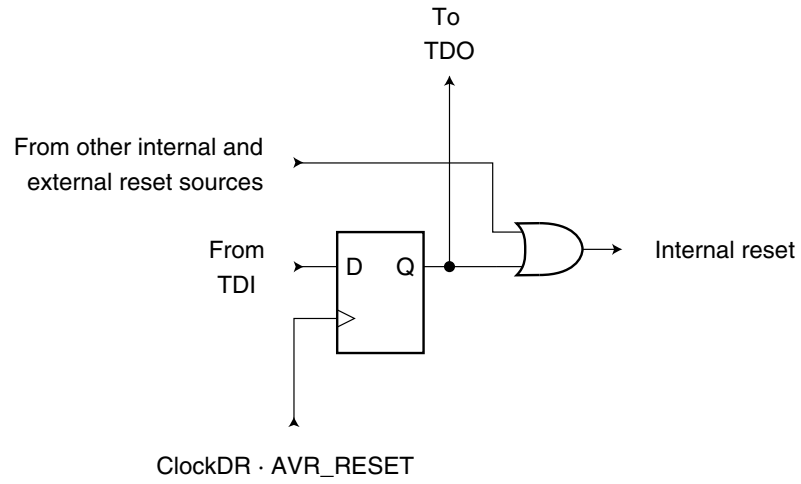
The manufacturer ID for ATMEL is 0x01F (11 bit)

Reset Register

The Reset Register is a Test Data Register used to reset the part. Since the AVR tri-states Port Pins when reset, the Reset Register can also replace the function of the unimplemented optional JTAG instruction HIGHZ.

A high value in the Reset Register corresponds to pulling the external Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-Out Period (refer to Table 6 on page 27) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 88.

Figure 88. Reset Register



Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins.

See “Boundary-scan Chain” on page 162 for a complete description.

Boundary-scan Specific JTAG Instructions

The instruction register is 4 bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-Scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedant state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this data sheet, the LSB is shifted in and out first for all shift registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which data register is selected as path between TDI and TDO for each instruction.

EXTEST; $0

Mandatory JTAG instruction for selecting the Boundary-Scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-Scan chain is driven out as soon as the JTAG IR-register is loaded by the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-Scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

IDCODE; $1

Optional JTAG instruction selecting the 32 bit ID register as Data Register. The ID register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE register is sampled into the Boundary-Scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

SAMPLE_PRELOAD; $2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-Scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-Scan Chain.
- Shift-DR: The Boundary-Scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-Scan chain is applied to the output latches. However, the output latches are not connected to the pins.

AVR_RESET; $C

The AVR specific public JTAG instruction for forcing the AVR device into the Reset Mode or releasing the JTAG reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic 'one' in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

BYPASS; $F

Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic ‘0’ into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

Boundary-scan Chain

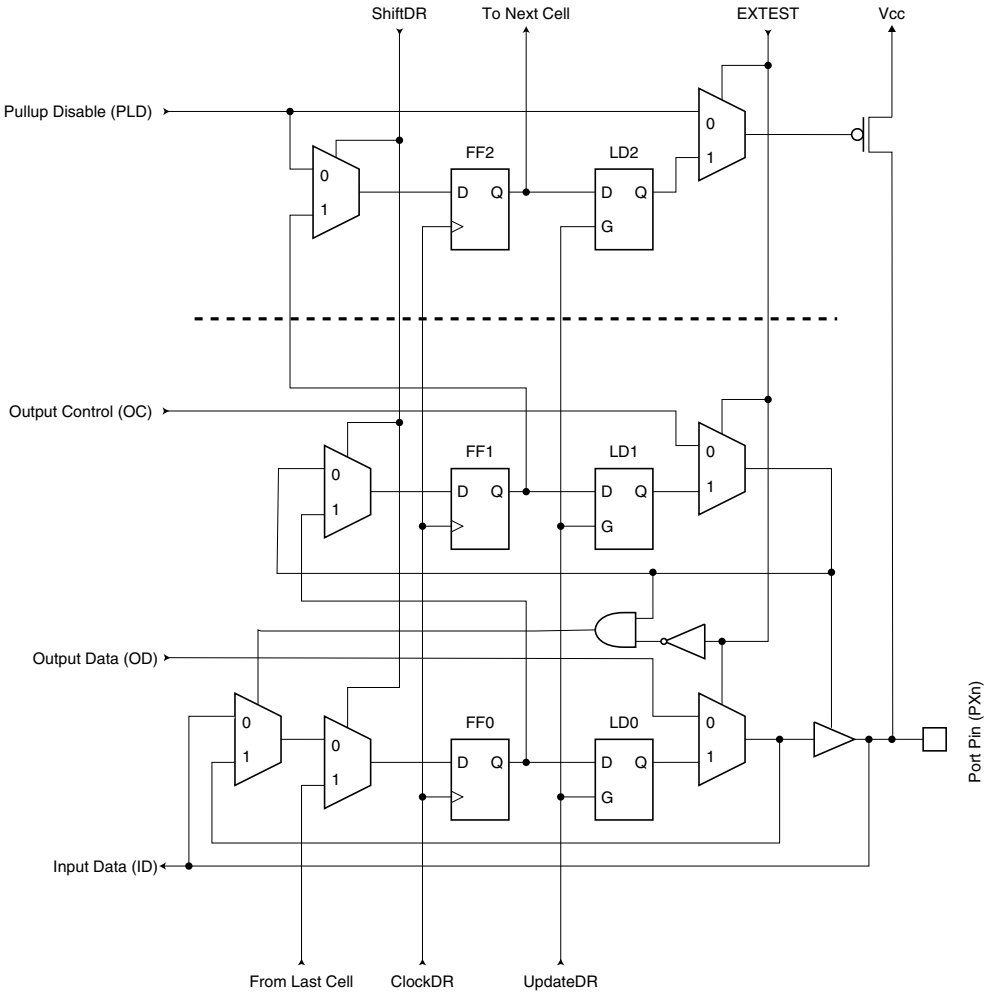
The Boundary-Scan chain has the capability of driving and observing the logic levels on the digital I/O pins.

Note: Compatibility issues regarding future devices: Future devices, included replacements for ATmega323 will have Pull-Up Enable signals instead of the Pull-Up Disable signals in the scan path (i.e. inverted logic). The scan cell for the reset signal will have the same logic level as the external pin (i.e. inverted logic). The length of the scan-chain is likely to change in future devices.

Scanning the Digital Port Pins

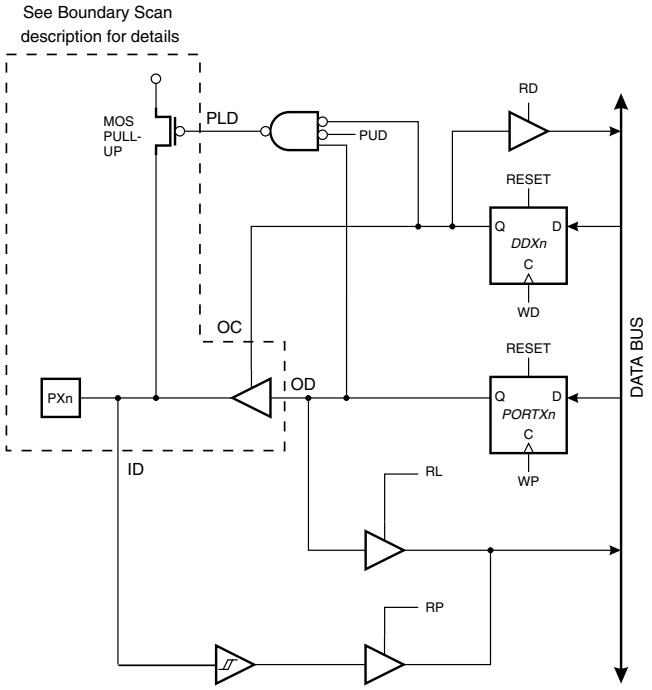
Figure 89 shows the Boundary-Scan Cell for Bidirectional Port Pins with Pull-up function. The cell consists of a standard Boundary-Scan cell for the Pull-up function, and a Bidirectional pin cell that combines the three signals Output Control - OC, Output Data - OD, and Input Data - ID, into only a two-stage shift register.

Figure 89. Boundary-scan Cell For bidirectional Port Pin with Pull-up Function.



The Boundary-Scan logic is not included in the figures in the Data Sheet. Figure 90 shows a simple digital Port Pin as described in the section “I/O Ports” on page 134. The Boundary-Scan details from Figure 89 replaces the dashed box in Figure 90.

Figure 90. General Port Pin Schematic diagram



WP: WRITE PORTX
WD: WRITE DDRX
RL: READ PORTX LATCH
RP: READ PORTX PIN
RD: READ DDRX
n: 0-7
PUD: PULL-UP DISABLE
PuD: JTAG PULL-UP DISABLE
OC: JTAG OUTPUT CONTROL
OD: JTAG OUTPUT DATA
ID: JTAG INPUT DATA

When no alternate port function is present, the Input Data - ID corresponds to the PINn register value, Output Data corresponds to the PORTn register, Output Control corresponds to the Data Direction - DDn register, and the PuLL-up Disable - PLD - corresponds to logic expression (DDn OR NOT(PORTBn)).

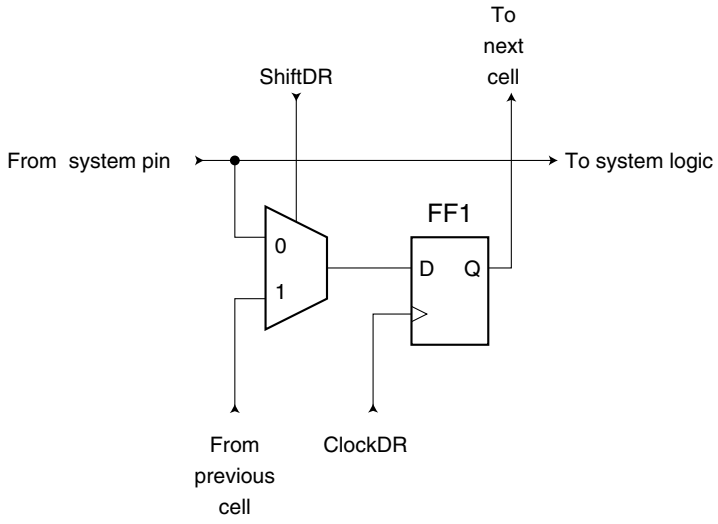
Digital alternate port functions are connected outside the dotted box in Figure 90 to make the scan chain read the actual pin value. For Analog function, there is a direct connection from the external pin to the analog circuit, and a scan chain is inserted on the interface between the digital logic and the analog circuit.

Scanning RESET

The RESET pin accepts 5V active low logic for standard reset operation. An observe-only cell as shown in Figure 91 is inserted at the output from the reset detector; RST.

Note: The scanned signal is active high, i.e., the RST signal is the inverse of the external RESET pin.

Figure 91. Observe-only Cell



Internal signals

ATmega323 contains a lot of scan chains for internal signals. The description of these signals are not public. However, the user must apply safe values to these cells before applying the Update-DR state of the TAP controller.

Note: Incorrect setting of the scan cells for internal signals may cause signal contention and can damage the part. Make sure the safe values are used.

Table 57. Boundary-scan signals for the ADC

Signal Name	Type of scan cell	Recommended input when not in use
SIG_PRIVATE0	General Scan Cell	1
SIG_PRIVATE1	General Scan Cell	0
SIG_PRIVATE2	General Scan Cell	0
SIG_PRIVATE3	General Scan Cell	0
SIG_PRIVATE4	General Scan Cell	0
SIG_PRIVATE5	General Scan Cell	0
SIG_PRIVATE6	General Scan Cell	0
SIG_PRIVATE7	General Scan Cell	0
SIG_PRIVATE8	General Scan Cell	0
SIG_PRIVATE9	General Scan Cell	0
SIG_PRIVATE10	General Scan Cell	0
SIG_PRIVATE11	General Scan Cell	1
SIG_PRIVATE12	General Scan Cell	0
SIG_PRIVATE13	General Scan Cell	0
SIG_PRIVATE14	General Scan Cell	0
SIG_PRIVATE15	General Scan Cell	0
SIG_PRIVATE16	General Scan Cell	0
SIG_PRIVATE17	General Scan Cell	0
SIG_PRIVATE18	General Scan Cell	0

Table 57. Boundary-scan signals for the ADC (Continued)

Signal Name	Type of scan cell	Recommended input when not in use
SIG_PRIVATE19	General Scan Cell	0
SIG_PRIVATE20	General Scan Cell	0
SIG_PRIVATE21	General Scan Cell	1
SIG_PRIVATE22	General Scan Cell	0
SIG_PRIVATE23	General Scan Cell	0
SIG_PRIVATE24	General Scan Cell	0
SIG_PRIVATE25	General Scan Cell	1
SIG_PRIVATE26	General Scan Cell	0
SIG_PRIVATE27	General Scan Cell	0
SIG_PRIVATE28	General Scan Cell	0
SIG_PRIVATE29	General Scan Cell	0
SIG_PRIVATE30	General Scan Cell	0
SIG_PRIVATE31	General Scan Cell	0
SIG_PRIVATE32	General Scan Cell	0
SIG_PRIVATE33	General Scan Cell	0
SIG_PRIVATE34	General Scan Cell	1
SIG_PRIVATE35	General Scan Cell	0
SIG_PRIVATE36	General Scan Cell	0
SIG_PRIVATE37	General Scan Cell	0
SIG_PRIVATE38	General Scan Cell	1
SIG_PRIVATE39	General Scan Cell	1
SIG_PRIVATE40	General Scan Cell	0
SIG_PRIVATE41	General Scan Cell	0
SIG_PRIVATE42	General Scan Cell	0
SIG_PRIVATE43	Observe Only	X
SIG_PRIVATE44	Observe Only	X
SIG_PRIVATE45	Observe Only	X
SIG_PRIVATE46	Observe Only	X

ATmega323 Boundary-scan Order

Table 64 shows the Scan order between TDI and TDO when the Boundary-Scan chain is selected as data path. Bit 0 is the LSB; the first bit scanned in, and the first bit scanned out. The scan order follows the pin-out order as far as possible. Therefore, the bits of Port A is scanned in the opposite bit order of the other ports. Exceptions from the rules are the Scan chains for the analog circuits, which constitute the most significant bits of the scan chain regardless of which physical pin they are connected to. In Figure 89, PXn.Data corresponds to FF0, PXn.Control corresponds to FF1, and PXn. Pullup_dissable corresponds to FF2. Bit 2, 3, 4, and 5 of Port C is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled

Table 58. ATmega323 Boundary-scan Order

Bit Number	Signal Name	Module
131	SIG_PRIVATE0	Privat Section 1
130	SIG_PRIVATE1	
129	SIG_PRIVATE2	
128	SIG_PRIVATE3	
127	SIG_PRIVATE4	Privat Section 2
126	SIG_PRIVATE5	
125	SIG_PRIVATE6	
124	SIG_PRIVATE7	
123	SIG_PRIVATE8	
122	SIG_PRIVATE9	
121	SIG_PRIVATE10	
120	SIG_PRIVATE11	
119	SIG_PRIVATE12	
118	SIG_PRIVATE13	
117	SIG_PRIVATE14	
116	SIG_PRIVATE15	
115	SIG_PRIVATE16	
114	SIG_PRIVATE17	
113	SIG_PRIVATE18	
112	SIG_PRIVATE19	
111	SIG_PRIVATE20	
110	SIG_PRIVATE21	
109	SIG_PRIVATE22	
108	SIG_PRIVATE23	
107	SIG_PRIVATE24	
106	SIG_PRIVATE25	
105	SIG_PRIVATE26	

Table 58. ATmega323 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
104	SIG_PRIVATE27	Private Section 2 (Cont.)
103	SIG_PRIVATE28	
102	SIG_PRIVATE29	
101	SIG_PRIVATE30	
100	SIG_PRIVATE31	
99	SIG_PRIVATE32	
98	SIG_PRIVATE33	
97	SIG_PRIVATE34	
96	SIG_PRIVATE35	
95	SIG_PRIVATE36	
94	SIG_PRIVATE37	
93	SIG_PRIVATE38	
92	SIG_PRIVATE39	
91	SIG_PRIVATE40	
90	SIG_PRIVATE41	
89	SIG_PRIVATE42	
88	PB0.Data	Port B
87	PB0.Control	
86	PB0.PuLLup_Disable	
85	PB1.Data	
84	PB1.Control	
83	PB1.PuLLup_Disable	
82	PB2.Data	
81	PB2.Control	
80	PB2.PuLLup_Disable	
79	PB3.Data	
78	PB3.Control	
77	PB3.PuLLup_Disable	
76	PB4.Data	
75	PB4.Control	
74	PB4.PuLLup_Disable	

Table 58. ATmega323 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
73	PB5.Data	Port B
72	PB5.Control	
71	PB5.PuLLup_Disable	
70	PB6.Data	
69	PB6.Control	
68	PB6.PuLLup_Disable	
67	PB7.Data	
66	PB7.Control	
65	PB7.PuLLup_Disable	
64	RSTT	Observe-Only cells
63	SIG_PRIVATE43	
62	SIG_PRIVATE44	
61	SIG_PRIVATE45	
60	SIG_PRIVATE46	
59	PD0.Data	Port D
58	PD0.Control	
57	PD0.PuLLup_Disable	
56	PD1.Data	
55	PD1.Control	
54	PD1.PuLLup_Disable	
53	PD2.Data	
52	PD2.Control	
51	PD2.PuLLup_Disable	
50	PD3.Data	
49	PD3.Control	
48	PD3.PuLLup_Disable	
47	PD4.Data	
46	PD4.Control	
45	PD4.PuLLup_Disable	

Table 58. ATmega323 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
44	PD5.Data	Port D
43	PD5.Control	
42	PD5.PuLLup_Disable	
41	PD6.Data	
40	PD6.Control	
39	PD6.PuLLup_Disable	
38	PD7.Data	
37	PD7.Control	
36	PD7.PuLLup_Disable	
35	PC0.Data	Port C
34	PC0.Control	
33	PC0.PuLLup_Disable	
32	PC1.Data	
31	PC1.Control	
30	PC1.PuLLup_Disable	
29	PC6.Data	
28	PC6.Control	
27	PC6.PuLLup_Disable	
26	PC7.Data	
25	PC7.Control	
24	PC7.PuLLup_Disable	
23	PA7.Data	Port A
22	PA7.Control	
21	PA7.PuLLup_Disable	
20	PA6.Data	
19	PA6.Control	
18	PA6.PuLLup_Disable	
17	PA5.Data	
16	PA5.Control	
15	PA5.PuLLup_Disable	
14	PA4.Data	
13	PA4.Control	
12	PA4.PuLLup_Disable	

Table 58. ATmega323 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
11	PA3.Data	Port A
10	PA3.Control	
9	PA3.PuLLup_Disable	
8	PA2.Data	
7	PA2.Control	
6	PA2.PuLLup_Disable	
5	PA1.Data	
4	PA1.Control	
3	PA1.PuLLup_Disable	
2	PA0.Data	
1	PA0.Control	
0	PA0.PuLLup_Disable	

Boundary-scan Description Language Files

Boundary-Scan Description Language (BSDL) files describe Boundary-Scan capable devices in a standard format used by automated test-generation software. The order and function of bits in the Boundary-Scan data register are included in this description. A BSDL file for ATmega323 is available.

Memory Programming

Boot Loader Support

The ATmega323 provides a mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates, controlled by the MCU using a Flash-resident Boot Loader program. 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 USART 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 ATmega323 Flash memory is organized in two main sections:

- The Application Flash section
- The Boot Loader Flash section

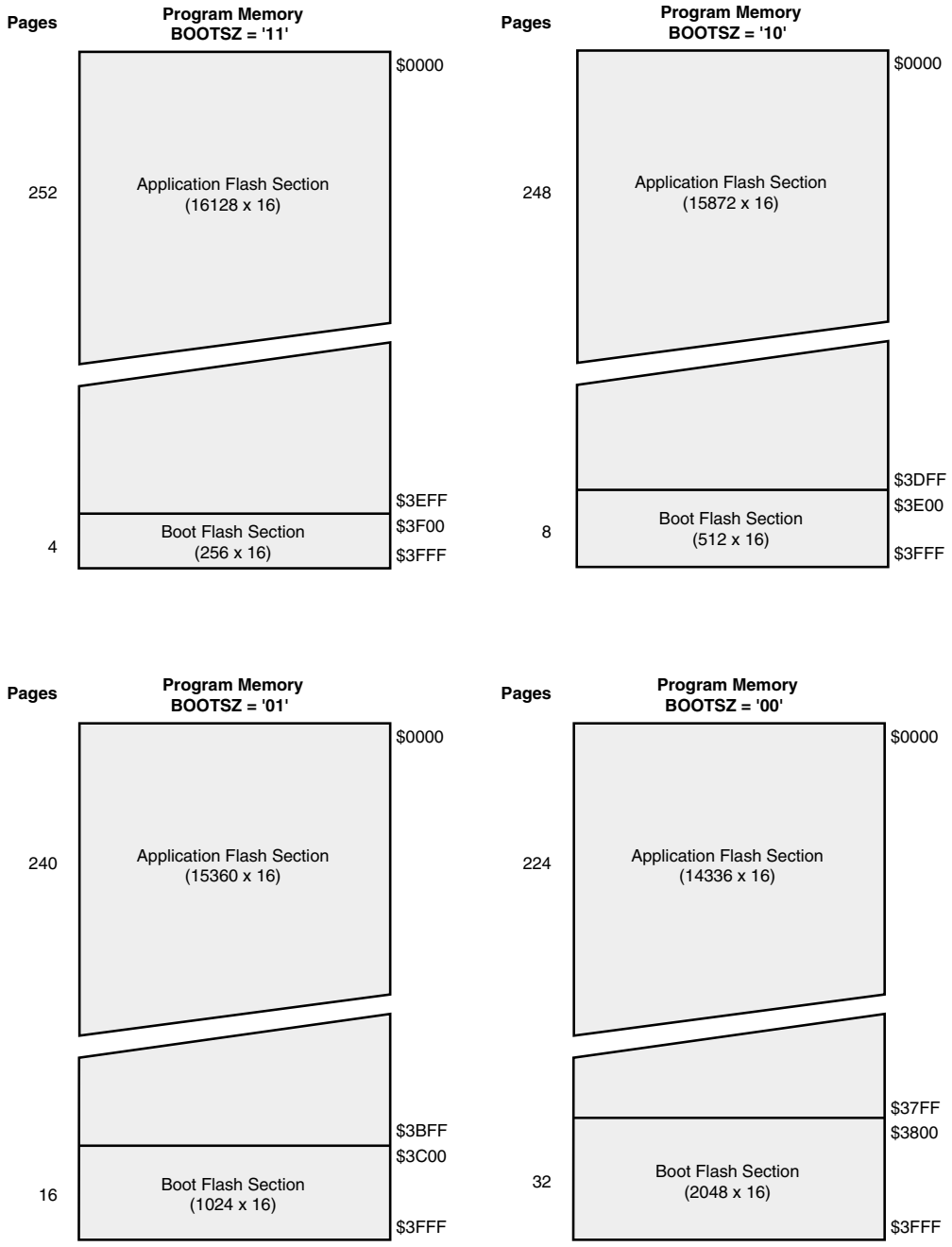
The Application Flash section and the Boot Loader Flash section have separate Boot Lock Bits. Thus the user can select different levels of protection for the two sections. The Store Program Memory (SPM) instruction can only be executed from the Boot Loader Flash section.

The program Flash memory in ATmega323 is divided into 256 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 59.

Table 59. Boot Size Configuration

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	256 words	4	$0000 - $3EFF	$3F00 - $3FFF	$3F00
1	0	512 words	8	$0000 - $3DFF	$3E00 - $3FFF	$3E00
0	1	1024 words	16	$0000 - $3BFF	$3C00 - $3FFF	$3C00
0	0	2048 words	32	$0000 - $37FF	$3800 - $3FFF	$3800

Figure 92. 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 USART 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 60. Boot Reset Fuse

BOOTRST	Reset Address
0	Reset Vector = Application reset (address $0000)
1	Reset Vector = Boot Loader reset (see Table 59)

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 179 for an assembly code example.

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 Z14: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. Z14: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 Z14: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 179 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 or moved to the Boot Loader section. 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 179 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

ATmega323 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 61 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 61. Boot Lock Bit0 Protection Modes (Application Section)

BLB0 Mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

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

Table 62. Boot Lock Bit1 Protection Modes (Boot Loader Section)

BLB1 mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: “1” 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 “Con-

stant Addressing Using the LPM and SPM Instructions" on page 16 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	-	-	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	OCDEN	JTAGEN	SPIEN	-	EESAVE	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 always ignored

Z14: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 16 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 ATmega323 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 ATmega323, this bit always reads as zero.

- **Bit 5 - Res: Reserved Bit**

This bit is a reserved bit in the ATmega323 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 4 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 Bits 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 than the Flash write time. This can be achieved by holding the external reset, or by selecting a long reset timeout.
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 ATmega323 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 63. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 63. Lock Bit Protection Modes

Memory Lock Bits			
LB mode	**LB2**	**LB1**	**Protection Type**
1	1	1	No memory lock features enabled for parallel, serial, and JTAG programming.
2	1	0	Further programming of the Flash and EEPROM is disabled in parallel, serial, and JTAG 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, serial, and JTAG programming mode. The Fuse bits are locked in both serial and parallel programming mode.(1)
BLB0 mode	**BLB02**	**BLB01**	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
BLB1 mode	**BLB12**	**BLB11**	
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: 1. Program the Fuse bits before programming the Lock bits.

Fuse Bits

The ATmega323 has 13 Fuse bits, divided in two groups. The Fuse High bits are OCDEN, JTAGEN, SPIEN, EESAVE, BOOTSZ1..0, and BOOTRST, and the Fuse Low bits are BODLEVEL, BODEN, and CKSEL3..0. All Fuses are accessible in parallel programming mode and when programming via the JTAG interface. In serial programming mode, all but the SPIEN Fuse is accessible.

- When the OCDEN fuse is programmed, the On-chip debug system is enabled if the JTAGEN fuse is programmed. If the JTAGEN fuse is unprogrammed, the OCDEN fuse has no visible effect.Never ship a product with the OCDEN fuse programmed. Regardless of the setting of lockbits and the JTAGEN Fuse, a programmed OCDEN fuse enables some parts of the clock system be running in all sleep modes. This may increase the power consumption. Default value is unprogrammed (“1”).
- When the JTAGEN fuse is programmed, the JTAG interface is enabled on port C pins PC5..2. Default value is programmed (“0”).
- 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 SPI serial programming mode.
- When EESAVE is programmed, the EEPROM memory is preserved through the Chip Erase cycle. Default value is unprogrammed (“1”). The EESAVE Fuse bit can not be programmed if any of the Lock bits are programmed.
- BOOTSZ1..0 select the size and start address of the Boot Flash section according to Table on page 172. 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 on page 172. 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 5 on page 26 and Table 6 on page 27, 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 22. Default value is unprogrammed (“1”).
- CKSEL3..0 select the clock source and the start-up delay after reset, according to Table 1 on page 6 and Table 6 on page 27. Default value is “0010” (Internal RC Oscillator, slowly rising power).

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.

For the ATmega323 the signature bytes are:

1. $000: $1E (indicates manufactured by Atmel)
2. $001: $95 (indicates 32KB Flash memory)
3. $002: $01 (indicates ATmega323 device when $001 is $95)

Calibration Byte

The ATmega323 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. To make use of this byte, it should be read from this location and written into the normal Flash program memory by the external programmer. 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 ATmega323. Pulses are assumed to be at least 500ns unless otherwise noted.

Signal Names

In this section, some pins of the ATmega323 are referenced by signal names describing their functionality during parallel programming, see Figure 93 and Table 64. 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 65.

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 66.

Figure 93. Parallel Programming

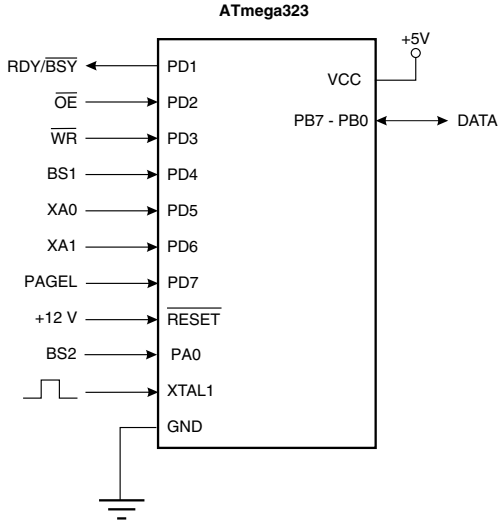


Table 64. Pin Name Mapping

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 64. Pin Name Mapping

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 Data Bus (Output When $\overline{OE}$ is Low)

Table 65. 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 66. 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 BS1 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 reprogrammed.

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 256 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. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the data byte.

E. Latch Data High and Low Byte

1. Set BS1 to “1”.
2. Give PAGEL a positive pulse. See Figure 94 for signal waveforms.

F. Repeat B through F 64 Times to Fill the Page Buffer.

To address a page in the Flash, 8 bits are needed (256 pages). The 6 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”.

G. 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 - $3F).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$goes low.
2. Wait until RDY/$\overline{BSY}$ goes high.
   (See Figure 95 for signal waveforms)

I. End Page Programming

1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

J. Repeat A through I 256 Times or Until All Data Has Been Programmed.

Figure 94. Programming the Flash Waveforms

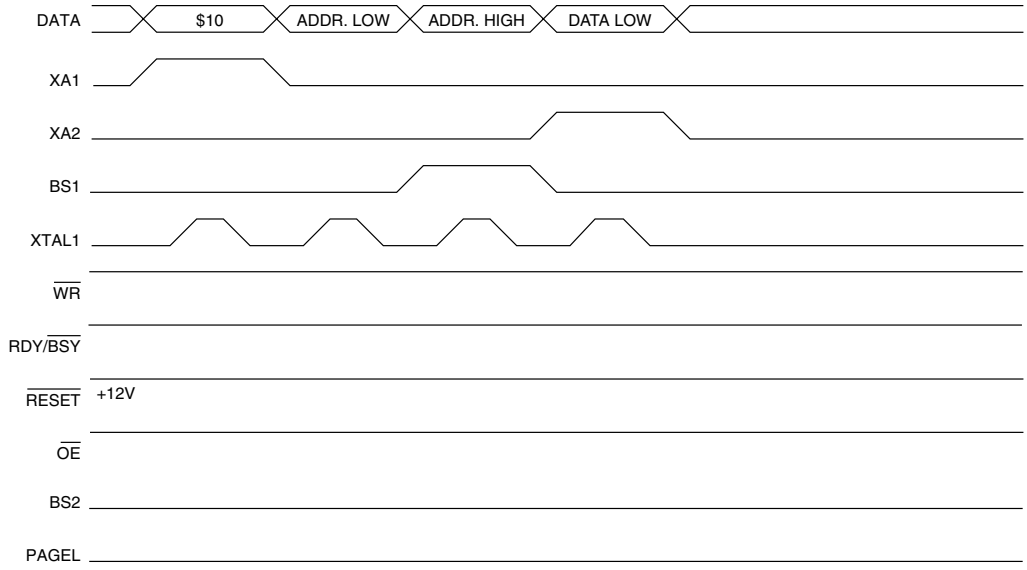
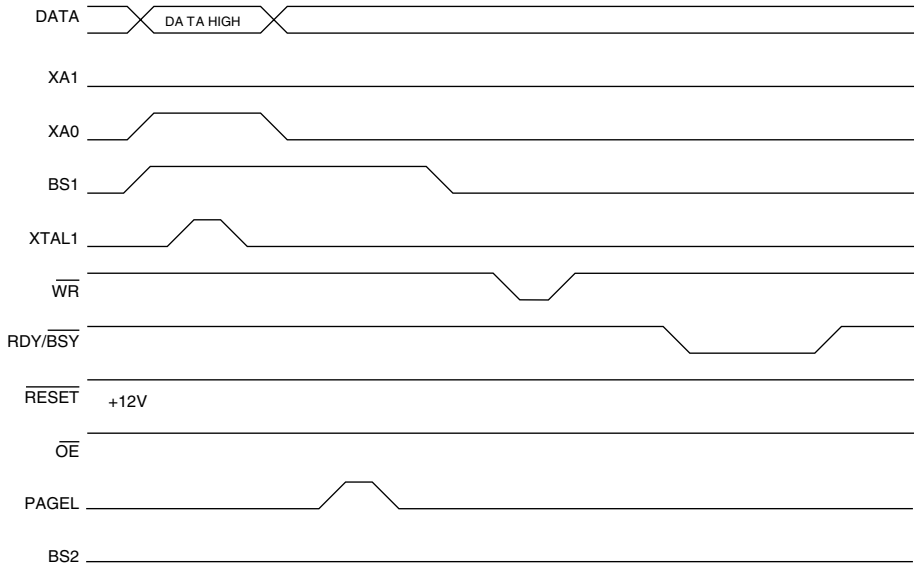


Figure 95. Programming the Flash Waveforms (continued)



Programming the EEPROM

The programming algorithm for the EEPROM data memory is as follows (refer to “Programming the Flash” on page 185 for details on Command, Address and Data loading):

1. A: Load Command “0001 0001”.
2. H: Load Address High Byte ($00 - $03)
3. B: Load Address Low Byte ($00 - $FF)
4. C: Load Data Low Byte ($00 - $FF)

K: Write Data Low Byte

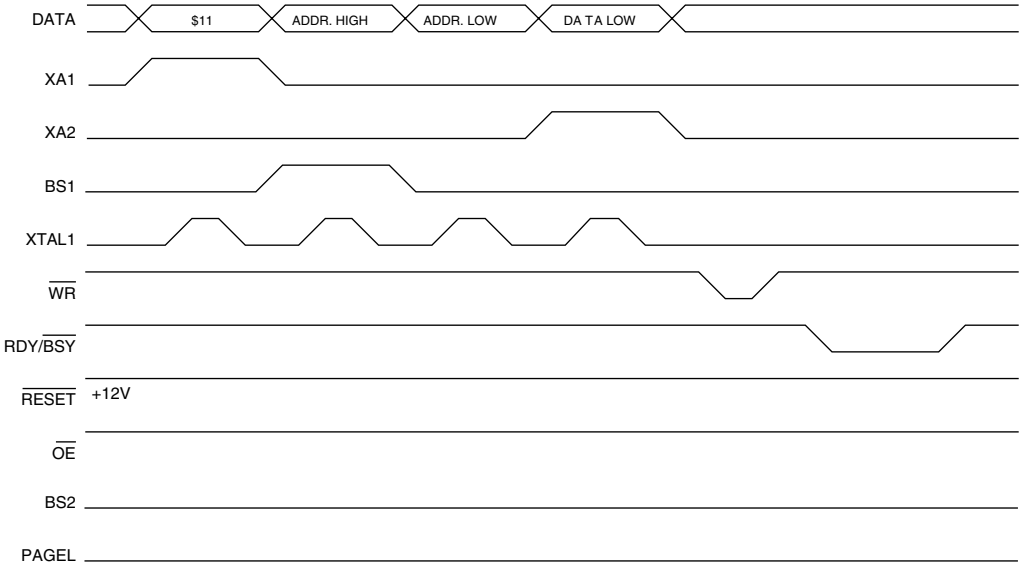
1. Set BS1 to “0”. This selects low data.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next byte. (See Figure 96 for signal waveforms)

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 byte window 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 96. Programming the EEPROM Waveforms



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 185 for details on Command and Address loading):

1. A: Load Command “0000 0010”.
2. H: Load Address High Byte ($00 - $3F)
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 185 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
2. H: Load Address High Byte ($00 - $03)
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 185 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
   Bit 7 = BODLEVEL Fuse bit
   Bit 6 = BODEN Fuse bit
   Bit 3..0 = CKSEL3..0 Fuse bits
   Bit 5,4 = “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 185 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
   Bit 7 = OCDEN Fuse bit
   Bit 6 = JTAGEN Fuse bit
   Bit 5 = SPIEN Fuse bit
   Bit 3 = EESAVE Fuse bit
   Bit 2..1 = BOOTSZ1..0 Fuse bits
   Bit 0 = BOOTRST Fuse bit
   Bit 7,6,4 = “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 185 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit.
   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 185 for details on Command loading):

1. A: Load Command "0000 0100".
2. Set $\overline{OE}$ to "0", BS2 to "0" and BS1 to "0". The status of the Fuse Low bits can now be read at DATA ("0" means programmed).
   Bit 7 = BODLEVEL Fuse bit
   Bit 6 = BODEN 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 7 = OCDEN Fuse bit
   Bit 6 = JTAGEN Fuse bit
   Bit 5 = SPIEN Fuse bit
   Bit 3 = EESAVE Fuse bit
   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 BS1 to "0". The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to Programming the Flash for details on Command and Address loading):

1. A: Load Command "0000 1000".
2. C: Load Address Low Byte, $00.
3. Set $\overline{OE}$ to "0", and BS1 to "1". The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

Parallel Programming Characteristics

Figure 97. Parallel Programming Timing

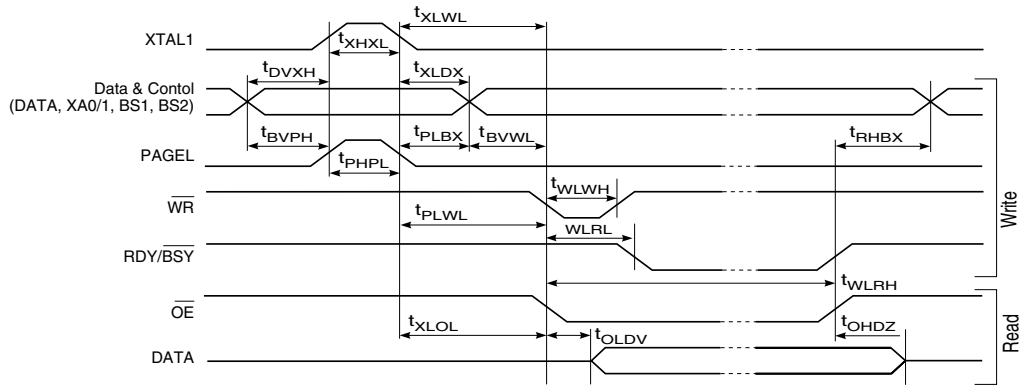


Table 67. Parallel Programming Characteristics, T_A = 25°C ± 10%, V_{CC} = 5 V ± 10%

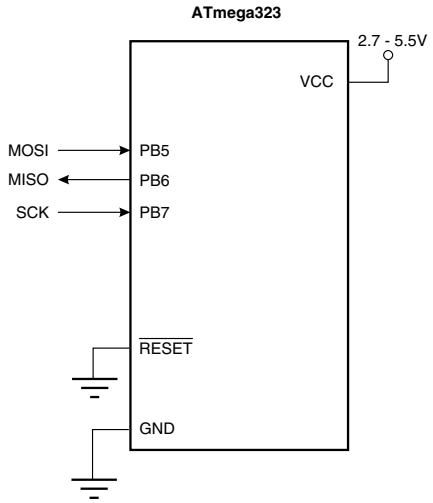
Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XHXL}	XTAL1 Pulse Width High	67			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ 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}[1]	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[1]	1	1.5	1.9	ms
t_{WLRH_CE}[2]	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	16	23	30	ms
t_{WLRH_FLASH}[3]	$\overline{WR}$ 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. tWLRH_CE is valid for the Chip Erase command.
3. tWLRH_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 98. Serial Programming and Verify



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 $3FFF for Program memory and $0000 to $03FF for EEPROM memory.

The device can be clocked by any clock option during Low Voltage Serial Programming. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles

High: > 2 CPU clock cycles

Serial Programming Algorithm

When writing serial data to the ATmega323, data is clocked on the rising edge of SCK.

When reading data from the ATmega323, data is clocked on the falling edge of SCK. See Figure 99, Figure 100 and Table 70 for timing details.

To program and verify the ATmega323 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 69):

1. Power-up sequence:

   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. The $\overline{\text{RESET}}$ 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, $\overline{\text{RESET}}$ must be given a positive pulse of at least two XTAL1 cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/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 4 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 68 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 8 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 68). 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 68). 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 a crystal is not 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 68 t_{WD_FLASH}.

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 68 for t_{WD_EEPROM}.

Programming Times for Nonvolatile 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 40). Consequently, programming times will be shorter and more accurate when programming or erasing nonvolatile memory from software, using SPM or the EEPROM interface. See Table 68 for a summary of programming times.

Table 68. Maximum Programming Times for Nonvolatile 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	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

Figure 99. Serial Programming Waveforms

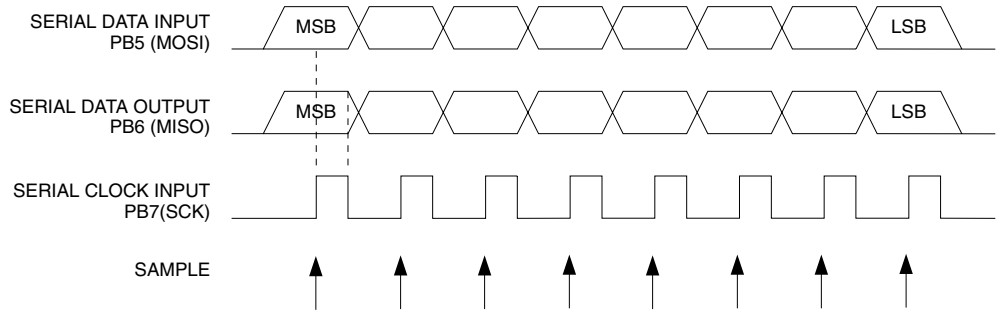


Table 69. Serial Programming Instruction Set

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	xx**aa 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	xx**aa aaaa**	**bb**xx xxxx	xxxx xxxx	Write Program Memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	xxxx xx**aa**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	xxxx xx**aa**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Read Lock Bits	0101 1000	0000 0000	xxxx xxxx	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	**IH**11 **GFED**	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	**IH**xx **GFED**	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; **G** = EESAVE Fuse; **H** = JTAGEN Fuse; and **I** = OCDEN Fuse

Serial Programming Characteristics

Figure 100. Serial Programming Timing

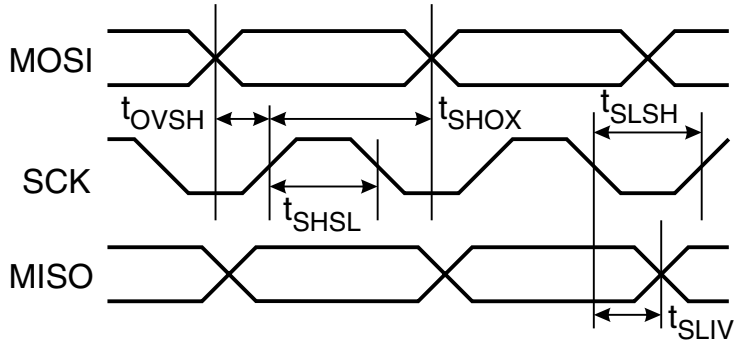


Table 70. Serial Programming Characteristics
T_A = -40°C to 85°C, V_{CC} = 2.7V - 5.5V (Unless otherwise noted)

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

Programming via the JTAG interface

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCSR must be cleared. Alternatively, if the JTD bit is set, the external reset can be forced low. Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in running mode while still allowing In-System Programming via the JTAG interface. Note that this technique can not be used when using the JTAG pins for Boundary-Scan or On-chip Debug. In these cases the JTAG pins must be dedicated for this purpose.

As a definition in this data sheet, the LSB is shifted in and out first of all shift registers.

Programming specific JTAG instructions

The instruction register is 4 bit wide, supporting up to 16 instructions. The JTAG instructions useful for Programming are listed below.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which data register is selected as path between TDI and TDO for each instruction.

The Run-Test/Idle state of the TAP controller is used to generate internal clocks. It can also be used as an idle state between JTAG sequences.

AVR_RESET ($C)

The AVR specific public JTAG instruction for setting the AVR device in the Reset Mode or taking the device out from the Reset Mode. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic 'one' in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

PROG_ENABLE ($4)

The AVR specific public JTAG instruction for enabling programming via the JTAG port. The 16 bit Programming Enable register is selected as data register. The active states are the following:

- Shift-DR: the programming enable signature is shifted into the data register.
- Update-DR: the programming enable signature is compared to the correct value, and programming mode is entered if the signature is valid.

PROG_COMMANDS ($5)

The AVR specific public JTAG instruction for entering programming commands via the JTAG port. The 15 bit Programming Command register is selected as data register. The active states are the following:

- Capture-DR: the result of the previous command is loaded into the data register.
- Shift-DR: the data register is shifted by the TCK input, shifting out the result of the previous command and shifting in the new command.
- Update-DR: the programming command is applied to the Flash inputs
- Run-Test/Idle: one clock cycle is generated, executing the applied command (not always required, see Table 71 below).

PROG_PAGELOAD ($6)

The AVR specific public JTAG instruction to directly load the Flash data page via the JTAG port. The 1024 bit Virtual Flash Page Load register is selected as data register. This is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the shift register is 8 bit. Unlike most JTAG instructions, the Update-DR state

is not used to transfer data from the shift register. The data are automatically transferred to the Flash page buffer byte by byte in the Shift-DR state by an internal state machine. This is the only active state:

- Shift-DR: Flash page data are shifted in from TDI by the TCK input, and automatically loaded into the Flash page one byte at a time.

PROG_PAGEREAD ($7)

The AVR specific public JTAG instruction to read one full Flash data page via the JTAG port. The 1032 bit Virtual Flash Page Read register is selected as data register. This is a virtual scan chain with length equal to the number of bits in one Flash page plus 8. Internally the shift register is 8 bit. Unlike most JTAG instructions, the Capture-DR state is not used to transfer data to the shift register. The data are automatically transferred from the Flash page buffer byte by byte in the Shift-DR state by an internal state machine. This is the only active state:

- Shift-DR: Flash data are automatically read one byte at a time and shifted out on TDO by the TCK input. The TDI input is ignored.

Data Registers

The data registers are selected by the JTAG instruction registers described in section “Programming specific JTAG instructions” on page 196. The data registers relevant for programming operations are:

- Reset Register
- Programming Enable Register
- Programming Command Register
- Virtual Flash Page Load Register
- Virtual Flash Page Read Register

Reset Register

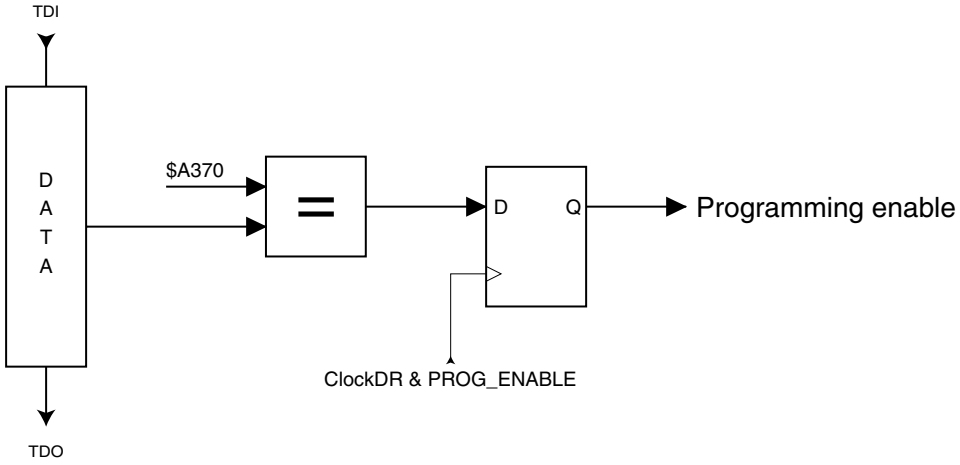
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering programming mode.

A high value in the Reset Register corresponds to pulling the external Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-Out Period (refer to Table 6 on page 27) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 93 on page 183.

Programming Enable Register

The Programming Enable register is a 16 bit register. The contents of this register is compared to the programming enable signature, binary code 1010_0011_0111_0000. When the contents of the register is equal to the programming enable signature, programming via the JTAG port is enabled. The register is reset to 0 on Power-on Reset, and should always be reset when leaving programming mode.

Figure 101. Programming Enable Register



Programming Command Register

The Programming Command register is a 15 bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG Programming Instruction Set is shown in Table 71. The state sequence when shifting in the programming commands is illustrated in Figure 102.

Figure 102. Programming Command Register

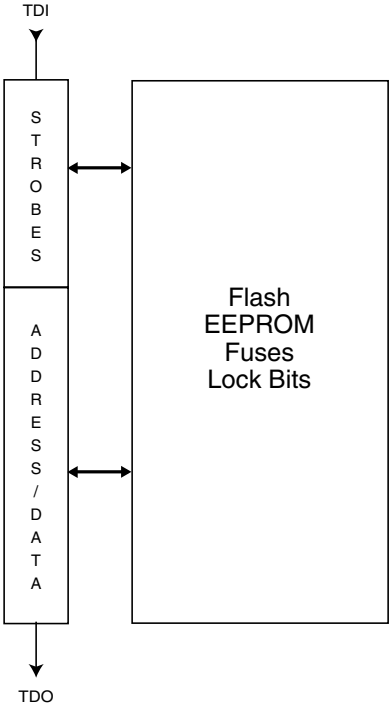


Table 71. JTAG Programming Instruction Set

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip erase	0100011_10000000 0110001_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for chip erase complete	0110011_10000000	xxxxxox_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address High Byte	0000111_00aaaaaa	xxxxxxx_xxxxxxxx	
2c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
2d. Load Data Low Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
2e. Load Data High Byte	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2g. Write Page	0110111_00000000 0110101_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Poll for Page Write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address High Byte	0000111_00aaaaaa	xxxxxxx_xxxxxxxx	
3c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
3d. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo	 low byte high byte
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_000000aa	xxxxxxx_xxxxxxxx	
4c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. Write EEPROM byte	0110011_00000000 0110001_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Poll for Byte Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_000000aa	xxxxxxx_xxxxxxxx	
5c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
5d. Read Data Byte	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data High Byte	0010011_IH11GFED	xxxxxxx_xxxxxxxx	(3)

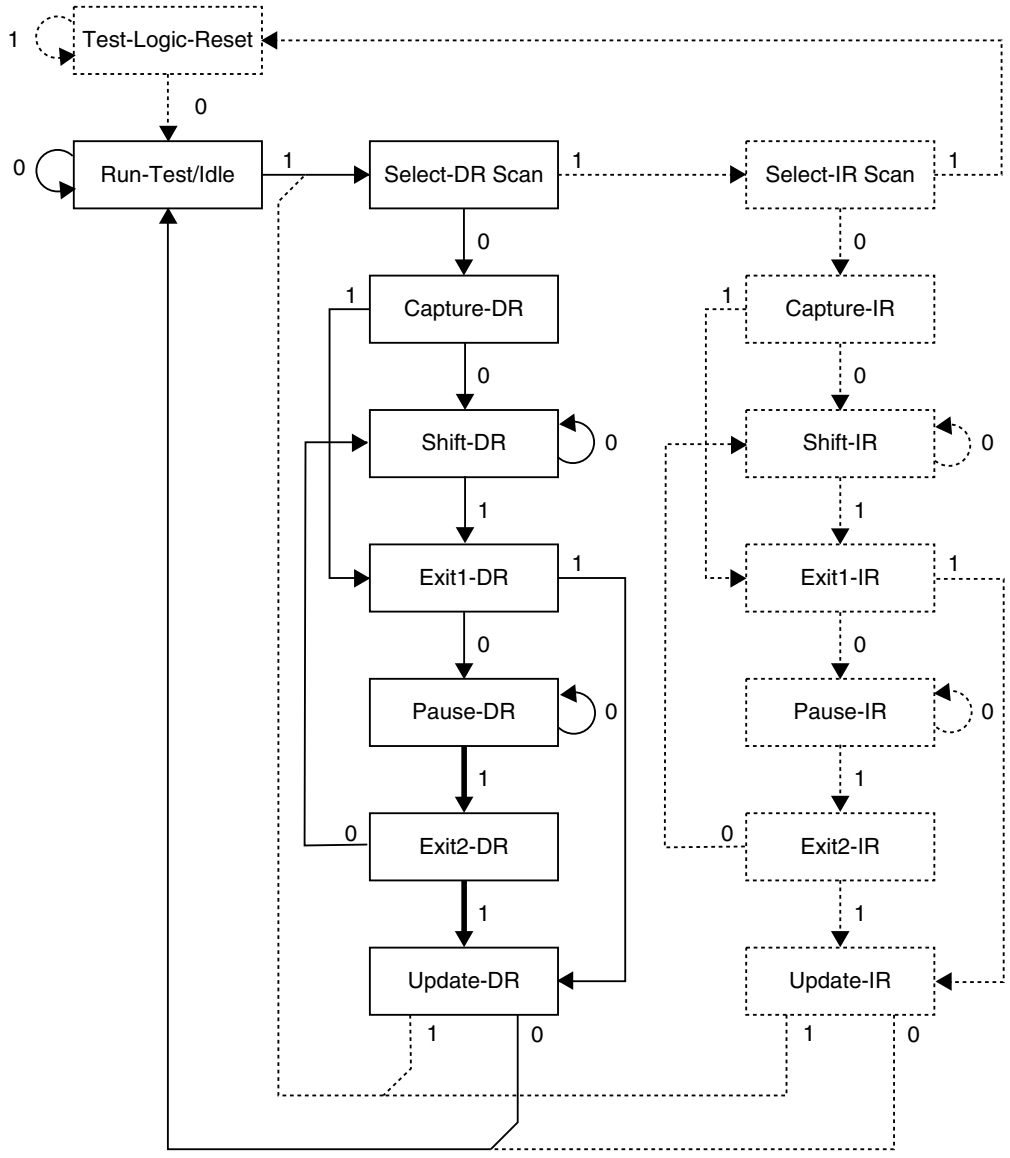
Table 71. JTAG Programming Instruction Set (Continued)

Instruction	TDI sequence	TDO sequence	Notes
6c. Write Fuse High byte	0110111_00000000 0110101_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6d. Poll for Fuse Write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
6e. Load Data Low Byte	0010011_CB11A987	xxxxxxx_xxxxxxxx	(3)
6f. Write Fuse Low byte	0110011_00000000 0110001_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6g. Poll for Fuse Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
7a. Enter Lock Bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte	0010011_11*654****321***	xxxxxxx_xxxxxxxx	(4)
7c. Write Lock Bits	0110011_00000000 0110001_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
7d. Poll for Lock Bit Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
8a. Enter Fuse/Lock Bit Read	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Read Fuse High Byte	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_IHxxGFED	
8c. Read Fuse Low Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_CBxxA987	
8d. Read Lock Bits	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xx*654****321***	(5)
8e. Read Fuses and Lock Bits	0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo	(5) fuse high byte fuse low byte lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Load Address Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Notes: 1. This command sequence is not required if the seven MSB are correctly set by the previous command sequence (which is normally the case).
2. Repeat until **o** = “1”
3. Set bits to “0” to program the corresponding fuse, “1” to unprogram the fuse.

4. Set bits to “0” to program the corresponding lock bit, “1” to leave the lock bit unchanged.
5. “0” = programmed, “1” = unprogrammed.
6. **a** = address high byte; **b** = address low byte; **i** = data in; **o** = data out; ***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; **G** = EESAVE Fuse; **H** = JTAGEN Fuse; **I** = OCDEN Fuse

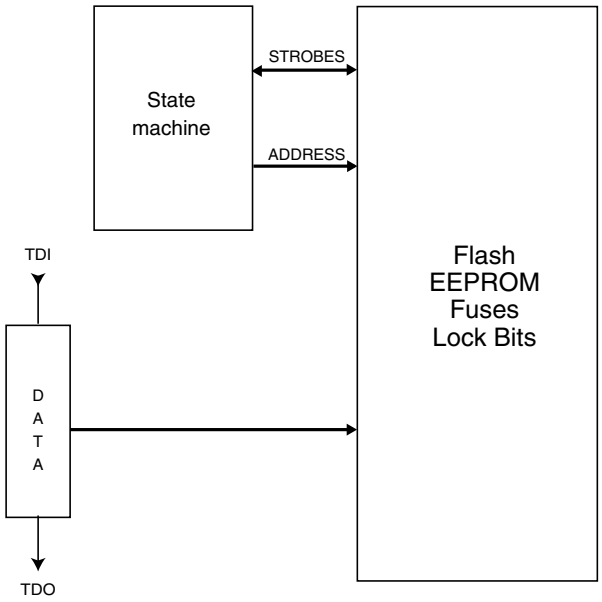
Figure 103. State Machine Sequence for Changing / Reading the Data Word



Virtual Flash Page Load Register

The Virtual Flash Page Load register is a virtual scan chain with length equal to the number of bits in one Flash page, 1024. Internally the shift register is 8 bit, and the data are automatically transferred to the Flash page buffer byte by byte. Shift in all instruction words in the page, starting with the LSB of the instruction with page address 0 and ending with the MSB of the instruction with page address 3F. This provides an efficient way to load the entire Flash page buffer before executing Page Write.

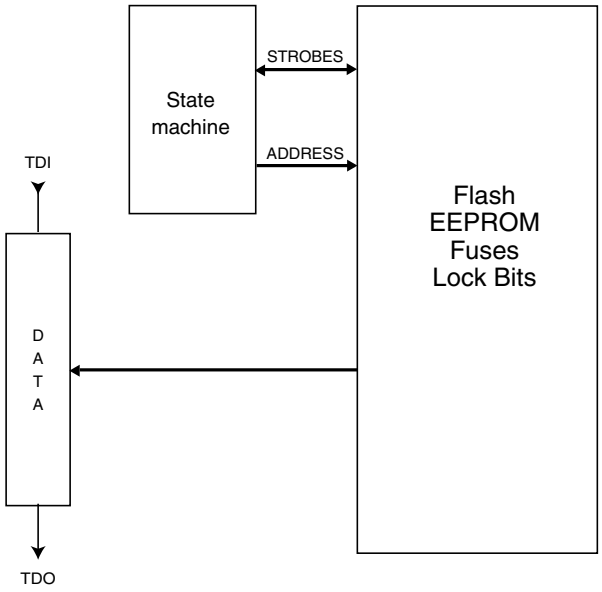
Figure 104. Virtual Flash Page Load Register



Virtual Flash Page Read Register

The Virtual Flash Page Read register is a virtual scan chain with length equal to the number of bits in one Flash page plus 8, 1032 in total. Internally the shift register is 8 bit, and the data are automatically transferred from the Flash data page byte by byte. The first 8 cycles are used to transfer the first byte to the internal shift register, and the bits that are shifted out during these 8 cycles should be ignored. Following this initialization, data are shifted out starting with the LSB of the instruction with page address 0 and ending with the MSB of the instruction with page address 3F. This provides an efficient way to read one full Flash page to verify programming.

Figure 105. Virtual Flash Page Read Register



Programming algorithm

All references below of type “1a”, “1b”, and so on, refer to Table 71.

Entering programming mode

1. Enter JTAG instruction AVR_RESET and shift 1 in the Reset register.
2. Enter instruction PROG_ENABLE and shift 1010_0011_0111_0000 in the Programming Enable register.

Leaving Programming Mode

1. Enter JTAG instruction PROG_COMMANDS.
2. Disable all programming instructions by usning no operation instruction 11a.
3. Enter instruction PROG_ENABLE and shift 0000_0000_0000_0000 in the programming Enable register.
4. Enter JTAG instruction AVR_RESET and shift 0 in the Reset register.

If PROG_ENABLE instruction is not followed by the AVR_RESET instruction, the following algorithm should be used:

1. Enter JTAG instruction PROG_COMMANDS.
2. Disable all programming instructions by using no operation instruction 11a.
3. Enter instruction PROG_ENABLE and shift 0000_0000_0000_0000 in the Programming Enable register.
4. Enter instruction PROG_ENABLE and shift 0000_0000_0000_0000 in the Programming Enable register.
5. Wait until the selected oscillator has started before applying more commands.

Performing Chip Erase

1. Enter JTAG instruction PROG_COMMANDS.
2. Start chip erase using programming instruction 1a.
3. Poll for chip erase complete using programming instruction 1b, or wait for t_{WLRH_CE} (refer to Table 67 on page 190).

Programming the Flash

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load address using programming instructions 2b and 2c.
4. Load data using programming instructions 2d, 2e and 2f.
5. Repeat step 3 and 4 for all 64 instruction words in the page.
6. Write the page using programming instruction 2g.
7. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH_FLASH} (refer to Table 67 on page 190).
8. Repeat steps 3 to 7 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load the page address using programming instructions 2b and 2c. The 6 LSB are used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGELOAD.
5. Load the entire page by shifting in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page.
6. Enter JTAG instruction PROG_COMMANDS.
7. Write the page using programming instruction 2g.
8. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH_FLASH} (refer to Table 67 on page 190).
9. Repeat steps 3 to 8 until all data have been programmed.

Reading the Flash

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load address using programming instructions 3b and 3c.
4. Read data using programming instruction 3d.
5. Repeat steps 3 and 4 until all data have been read.

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load the page address using programming instructions 3b and 3c. The 6 LSB are used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGEREAD.
5. Read the entire page by shifting out all instruction words in the page, starting with the LSB of the instruction with page address 0 and ending with the MSB of the instruction with page address 3F. Remember that the first 8 bits should be ignored.

6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

Programming the EEPROM

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM write using programming instruction 4a.
3. Load address using programming instructions 4b and 4c.
4. Load data using programming instructions 4d.
5. Write the data using programming instruction 4e.
6. Poll for EEPROM write complete using programming instruction 4f, or wait for t_{WLRH} (refer to Table 67 on page 190).
7. Repeat steps 3 to 6 until all data have been programmed.

Reading the EEPROM

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM read using programming instruction 5a.
3. Load address using programming instructions 5b and 5c.
4. Read data using programming instruction 5d.
5. Repeat steps 3 and 4 until all data have been read.

Programming the Fuses

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse write using programming instruction 6a.
3. Load data high byte using programming instructions 6b. A bit value of “0” will program the corresponding fuse, a “1” will unprogram the fuse.
4. Write Fuse high byte using programming instruction 6c.
5. Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to Table 67 on page 190).
6. Load data low byte using programming instructions 6e. A “0” will program the fuse, a “1” will unprogram the fuse.
7. Write Fuse low byte using programming instruction 6f.
8. Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to Table 67 on page 190).

Programming the Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Lock bit write using programming instruction 7a.
3. Load data using programming instructions 7b. A bit value of “0” will program the corresponding lock bit, a “1” will leave the lock bit unchanged.
4. Write Lock bits using programming instruction 7c.
5. Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 67 on page 190).

Reading the Fuses and Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse/Lock bit read using programming instruction 8a.
3. To read all Fuses and Lock bits, use programming instruction 8e.
   To only read Fuse high byte, use programming instruction 8b.
   To only read Fuse low byte, use programming instruction 8c.
   To only read Lock bits, use programming instruction 8d.

Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address $00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address $01 and address $02 to read the second and third signature bytes, respectively.

Reading the Calibration Byte

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Calibration byte read using programming instruction 10a.
3. Load address $00 using programming instruction 10b.
4. Read the calibration byte using programming instruction 10c.

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)

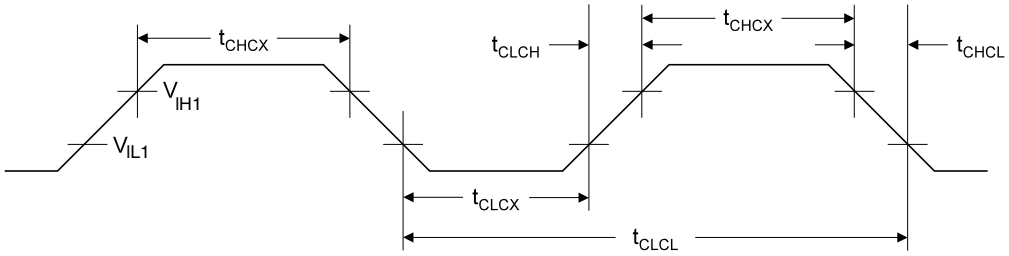
T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
I_{CC}	Power Supply Current	Active 4 MHz, V_{CC} = 3V (ATmega323L)			5	mA
		Active 8 MHz, V_{CC} = 5V (ATmega323)			15	mA
		Idle 4 MHz, V_{CC} = 3V (ATmega323L)			2.5	mA
		Idle 8 MHz, V_{CC} = 5V (ATmega323)			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

Note:
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 (20mA at Vcc = 5V, 10mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed – PDIP Package:
   1] The sum of all IOL, for all ports, should not exceed 200 mA.
   2] The sum of all IOL, for port A0-A7, should not exceed 100 mA.
   3] The sum of all IOL, for ports B0-B7,C0-C7, D0-D7 and XTAL2, should not exceed 100 mA – TQFP Package:
   1] The sum of all IOL, for all ports, should not exceed 400 mA.
   2] The sum of all IOL, for ports A0-A7, should not exceed 100 mA.
   3] The sum of all IOL, for ports B0-B3, should not exceed 100 mA.
   4] The sum of all IOL, for ports B4-B7, should not exceed 100 mA.
   5] The sum of all IOL, for ports C0-C3, should not exceed 100 mA.
   6] The sum of all IOL, for ports C4-C7, should not exceed 100 mA.
   7] The sum of all IOL, for ports D0-D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all IOL, for ports D4-D7, should not exceed 100 mA
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (3mA at Vcc = 5V, 1.5mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed – PDIP Package:
   1] The sum of all IOH, for all ports, should not exceed 200 mA.
   2] The sum of all IOH, for port A0-A7, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0-B7,C0-C7, D0-D7 and XTAL2, should not exceed 100 mA – TQFP Package:
   1] The sum of all IOH, for all ports, should not exceed 400 mA.
   2] The sum of all IOH, for ports A0-A7, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0-B3, should not exceed 100 mA.
   4] The sum of all IOH, for ports B4-B7, should not exceed 100 mA.
   5] The sum of all IOH, for ports C0-C3, should not exceed 100 mA.
   6] The sum of all IOH, for ports C4-C7, should not exceed 100 mA.
   7] The sum of all IOH, for ports D0-D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all IOH, for ports D4-D7, should not exceed 100 mA
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum V_{CC} for Power-down is 2.5V.

External Clock Drive Waveforms

Figure 106. External Clock Drive Waveforms



External Clock Drive

Table 72. 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 73. External RC Oscillator, Typical Frequencies

R [kΩ]	C [pF]	f
100	70	TBD
31.5	20	TBD
6.5	20	TBD

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.

2-wire Serial Interface Characteristics

Table 74 describes the requirements for devices connected to the 2-wire Serial Bus. The ATmega323 2-wire Serial Interface meets or exceeds these requirements under the noted conditions.

Timing symbols refer to Figure 107.

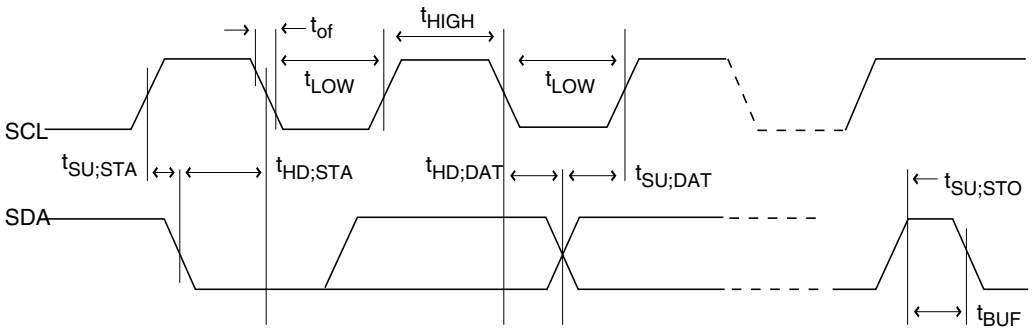
Table 74. 2-wire Serial Bus Requirements

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 ATmega323, 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 ATmega323 2-wire Serial Interface operation. Other devices connected to the 2-wire Serial Bus need only obey the general f_{SCL} requirement.
6. The actual low period generated by the ATmega323 2-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 ATmega323 2-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, ATmega323 devices connected to the bus may communi-

cate at full speed (400 kHz) with other ATmega323 devices, as well as any other device with a proper t_{LOW} acceptance margin.

Figure 107. 2-wire Serial Bus timing



ATmega323 Typical Characteristics – Preliminary Data

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L*V_{CC}*f where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog timer.

Figure 108. Active Supply Current vs. Frequency

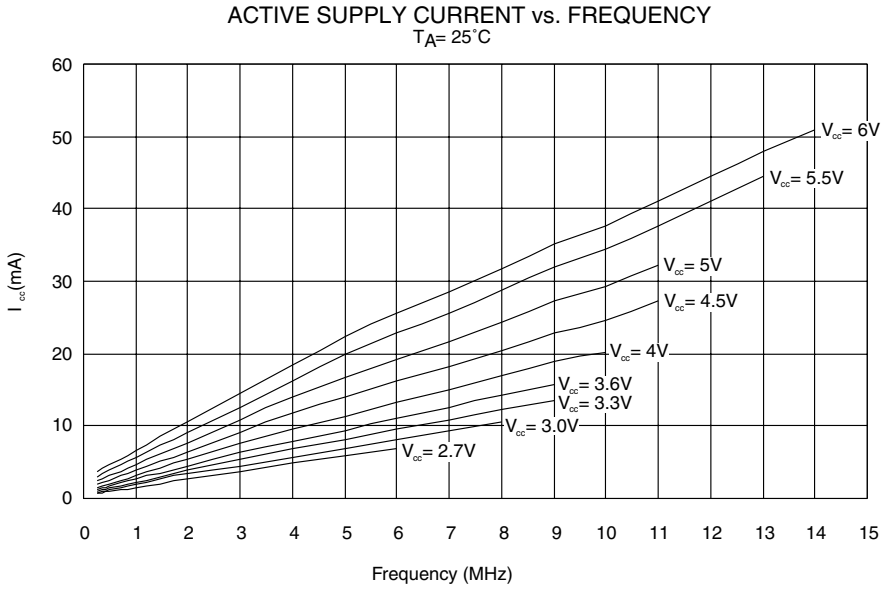


Figure 109. Active Supply Current vs. V_{CC}

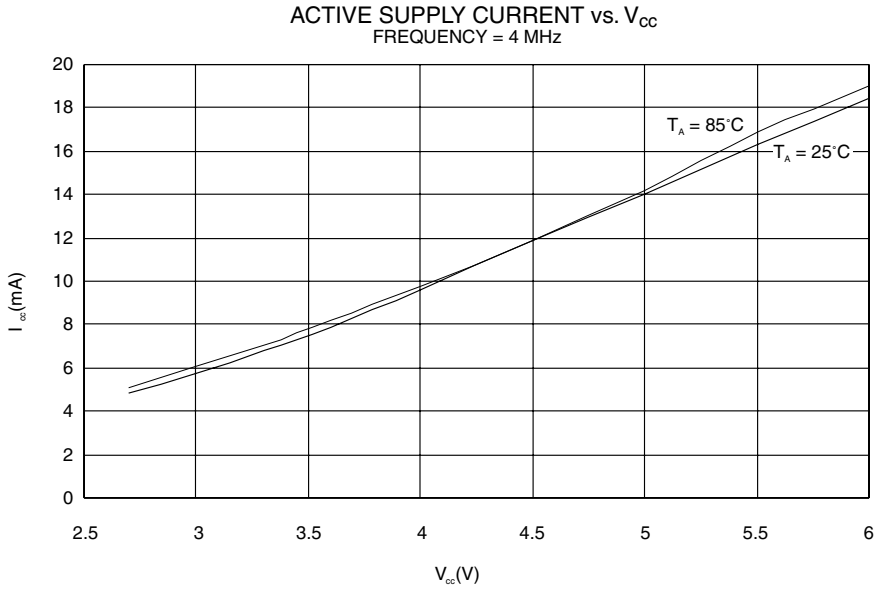


Figure 110. Active Supply Current vs. V_{CC}, Device Clocked by Internal Oscillator

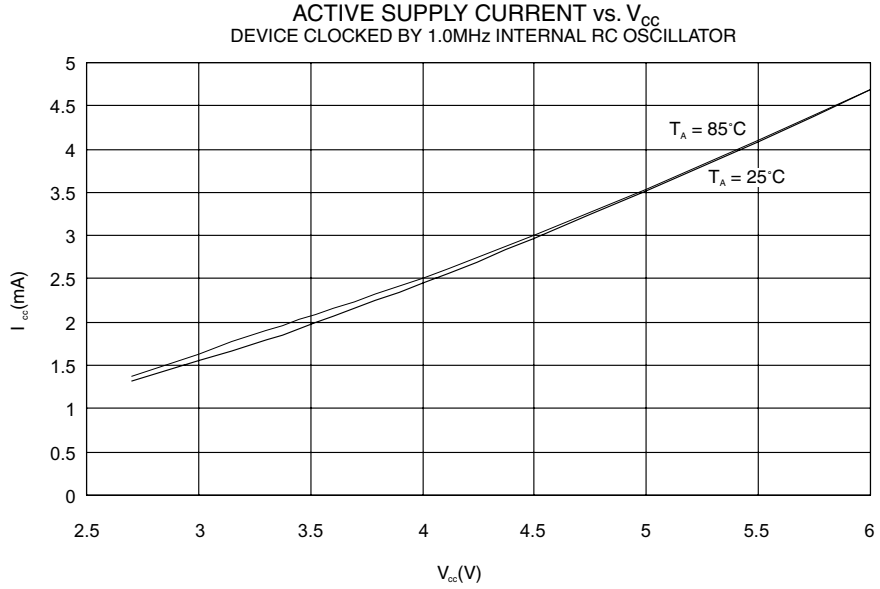


Figure 111. Active Supply Current vs. V_{CC}, Device Clocked by External 32kHz Crystal

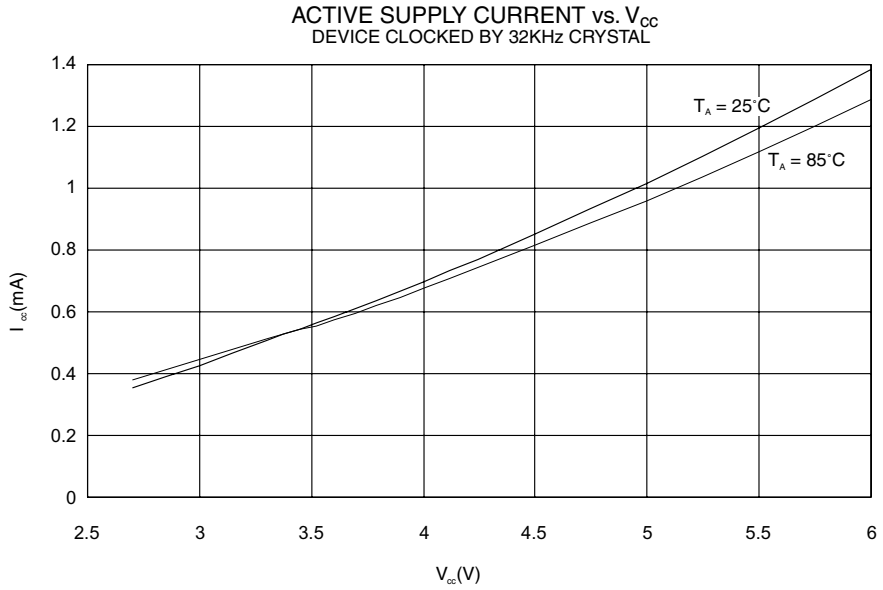


Figure 112. Idle Supply Current vs. Frequency

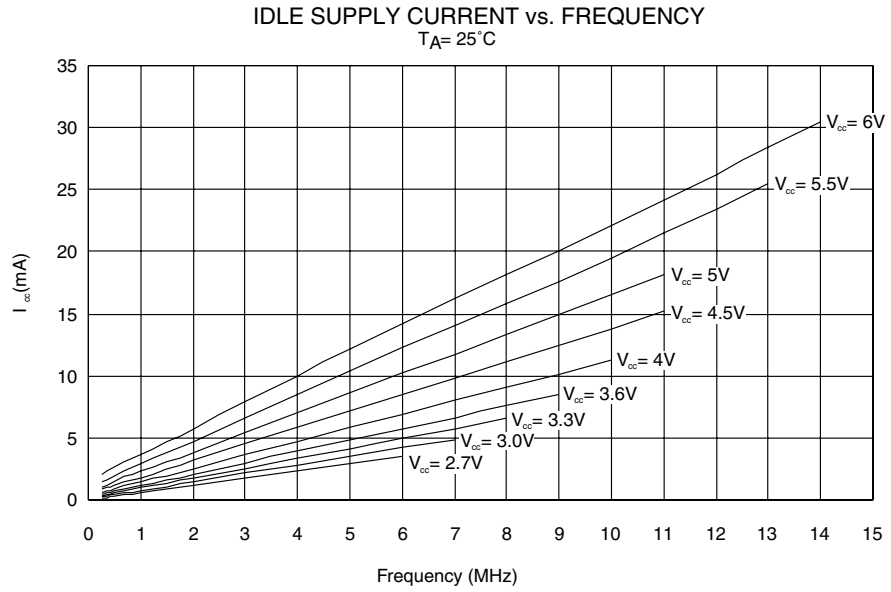


Figure 113. Idle Supply Current vs. V_{CC}

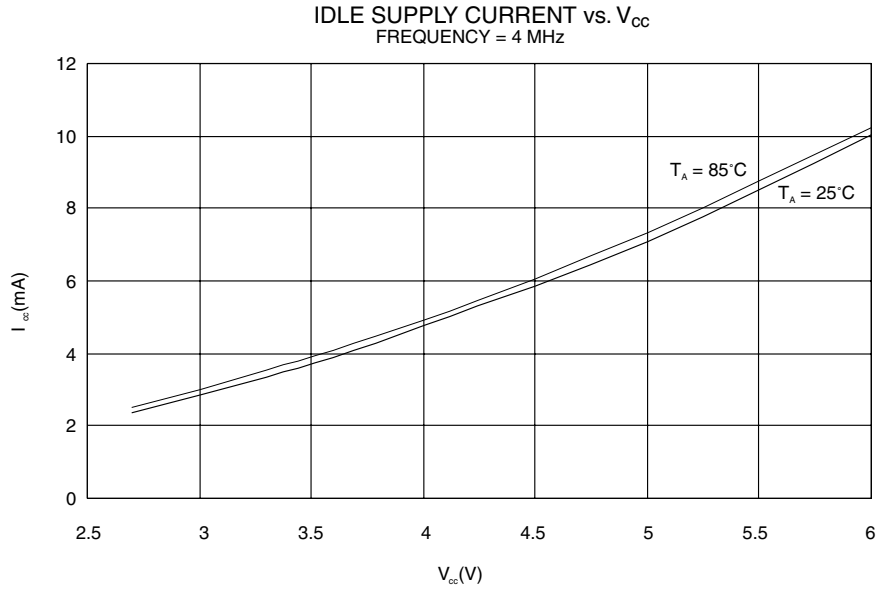


Figure 114. Idle Supply Current vs. V_{CC}, Device Clocked by Internal Oscillator

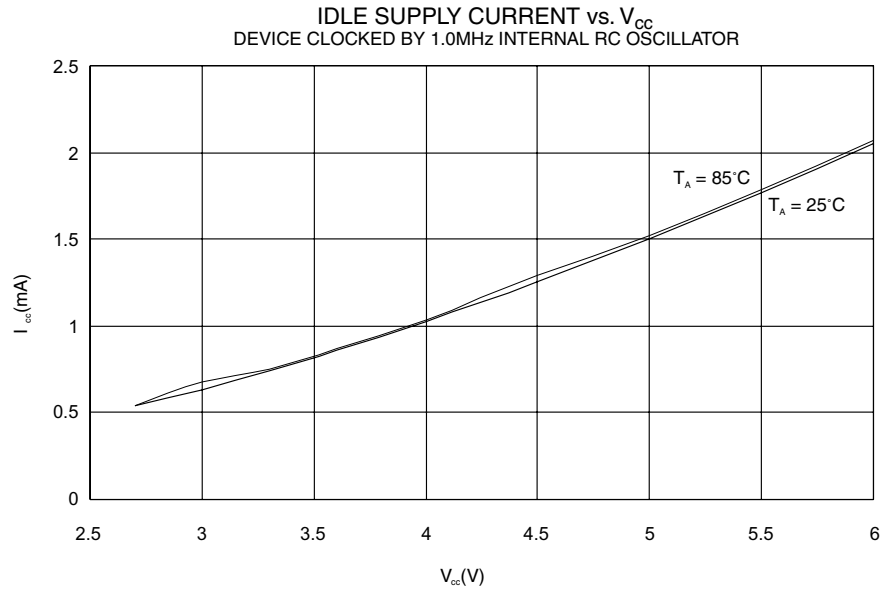


Figure 115. Idle Supply Current vs. V_{CC}, Device Clocked by External 32kHz Crystal

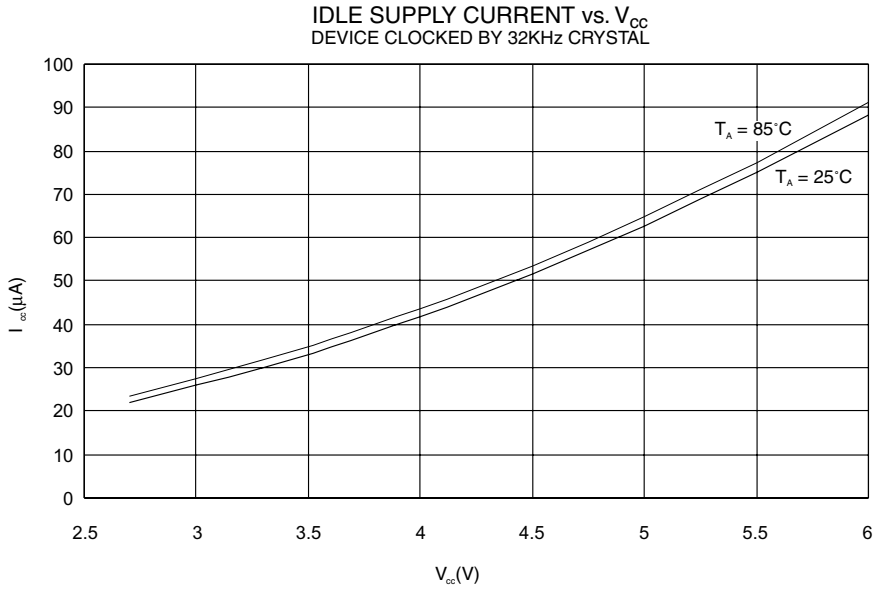


Figure 116. Power-down Supply Current vs. V_{CC}

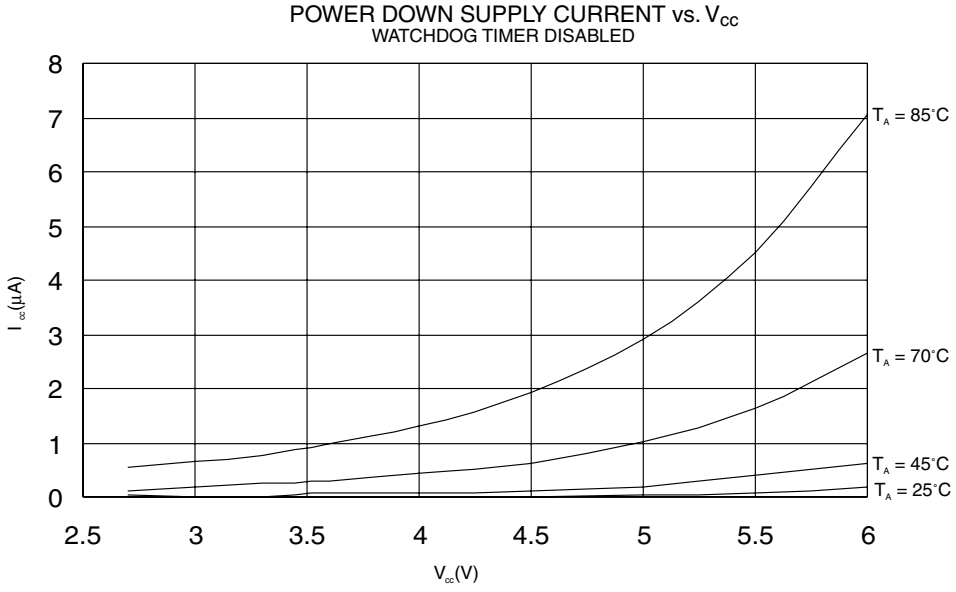


Figure 117. Power-down Supply Current vs. V_{CC}

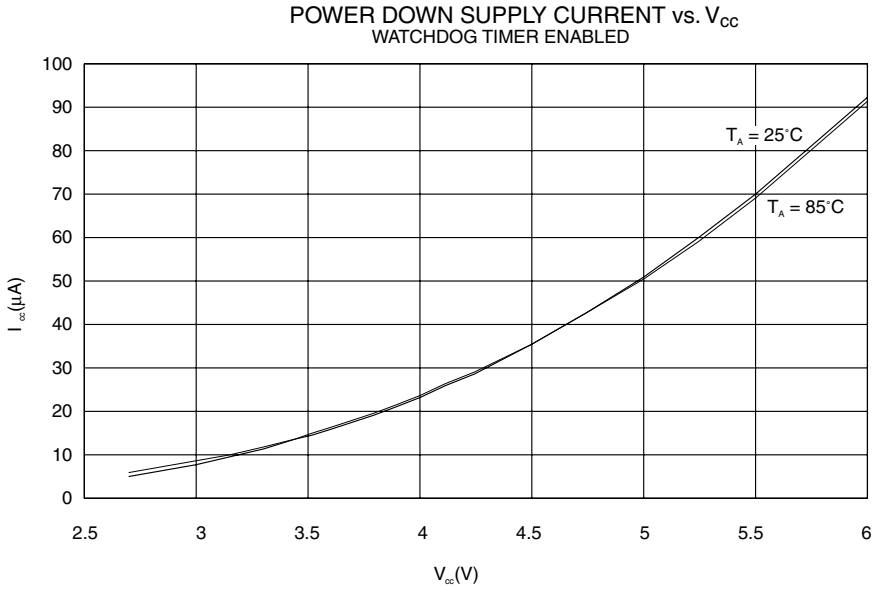


Figure 118. Power-save Supply Current vs. V_{CC}

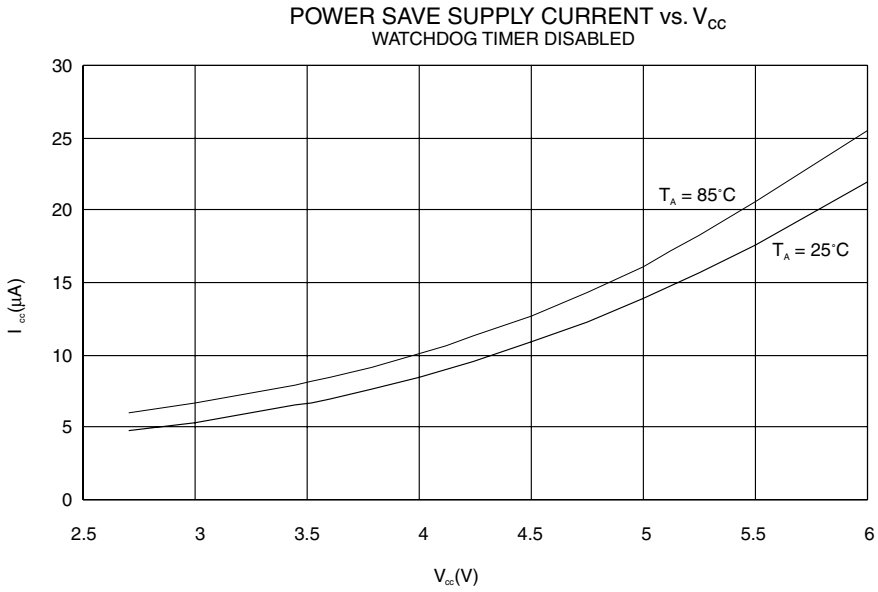
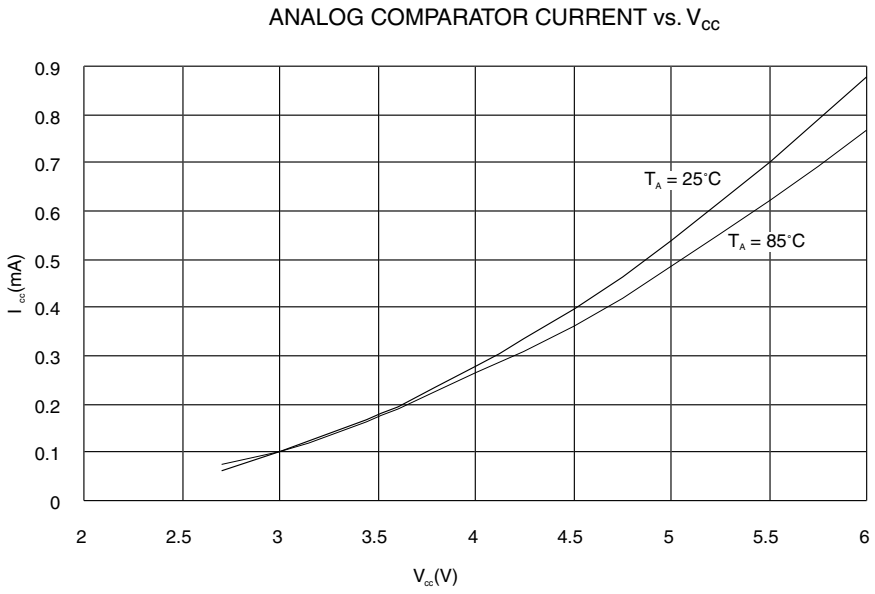


Figure 119. Analog Comparator Current vs. V_{CC}



Analog comparator offset voltage is measured as absolute offset

Figure 120. Analog Comparator Offset Voltage vs. Common Mode Voltage

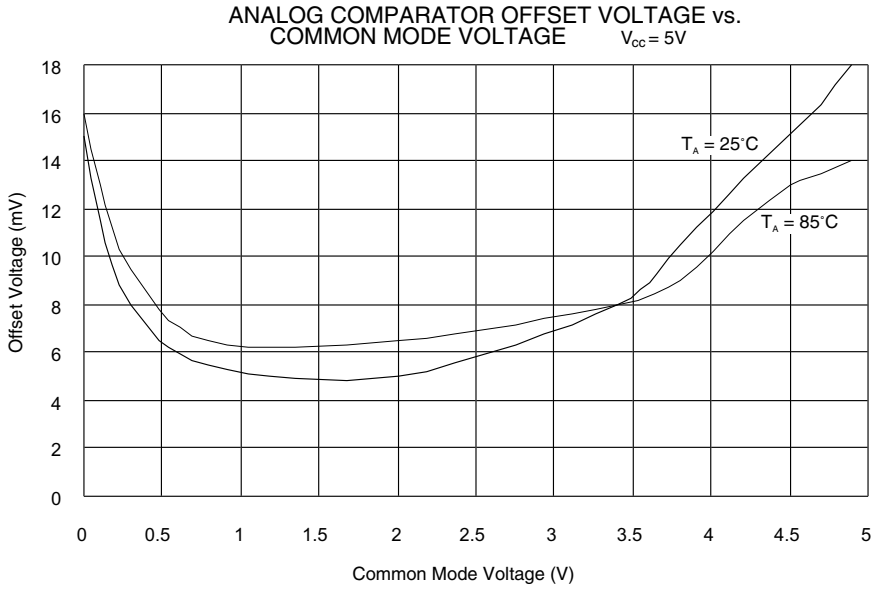


Figure 121. Analog Comparator Offset voltage vs. Common Mode Voltage

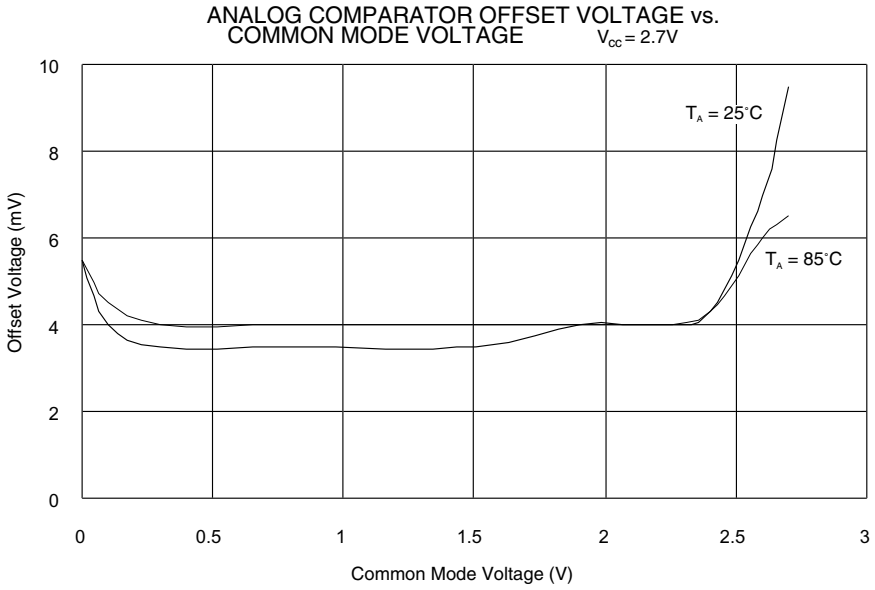


Figure 122. Analog Comparator Input Leakage Current

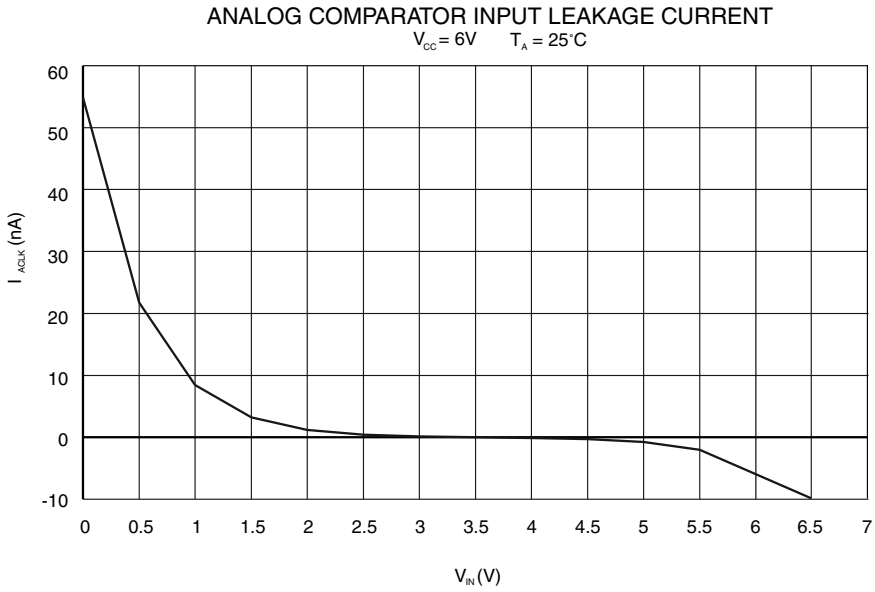


Figure 123. Calibrated RC Oscillator Frequency vs. V_{CC}

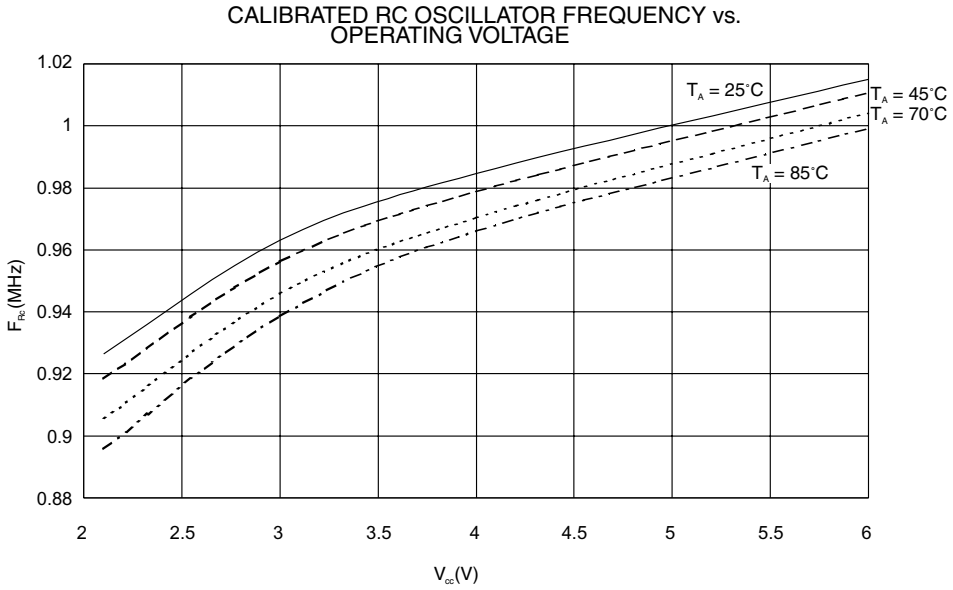
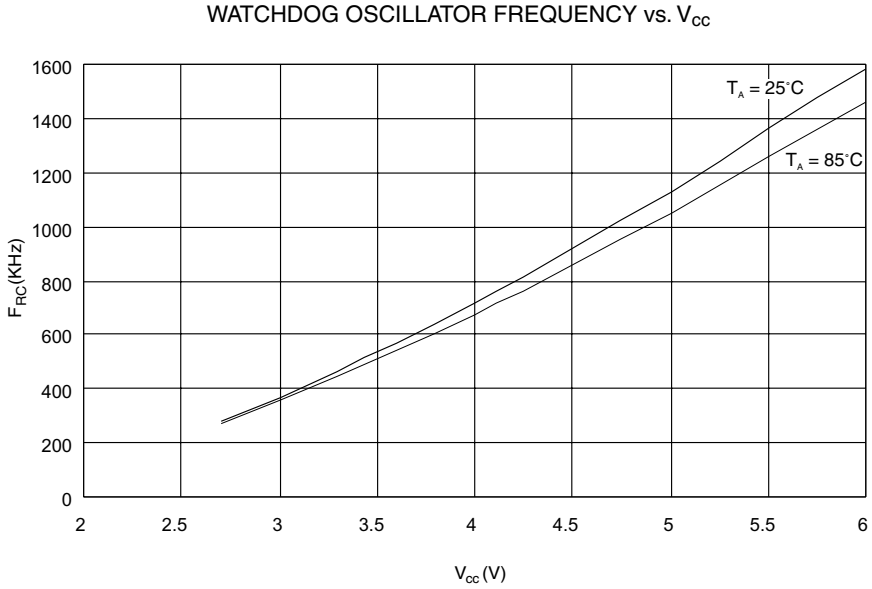


Figure 124. Watchdog Oscillator Frequency vs. V_{CC}



Sink and source capabilities of I/O ports are measured on one pin at a time.

Figure 125. Pull-Up Resistor Current vs. Input Voltage

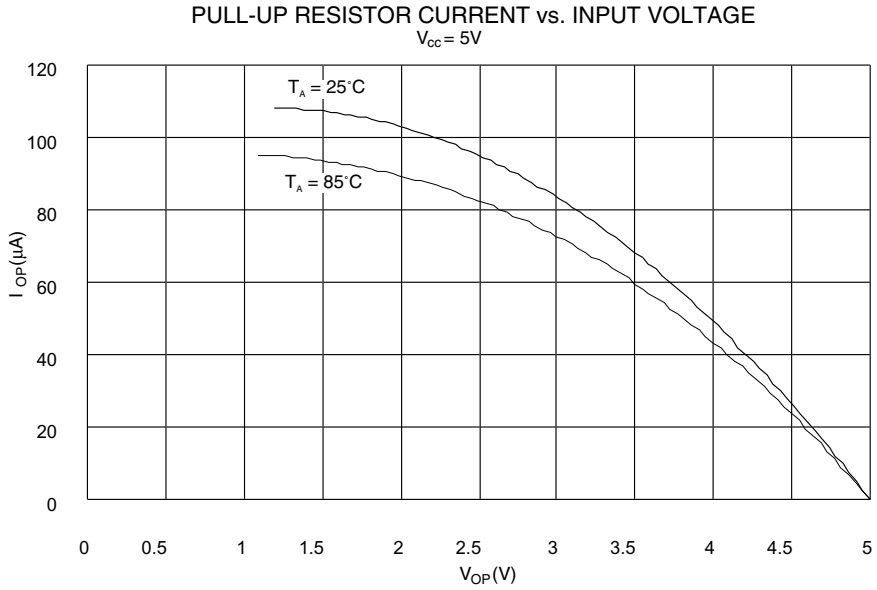


Figure 126. Pull-Up Resistor Current vs. Input Voltage

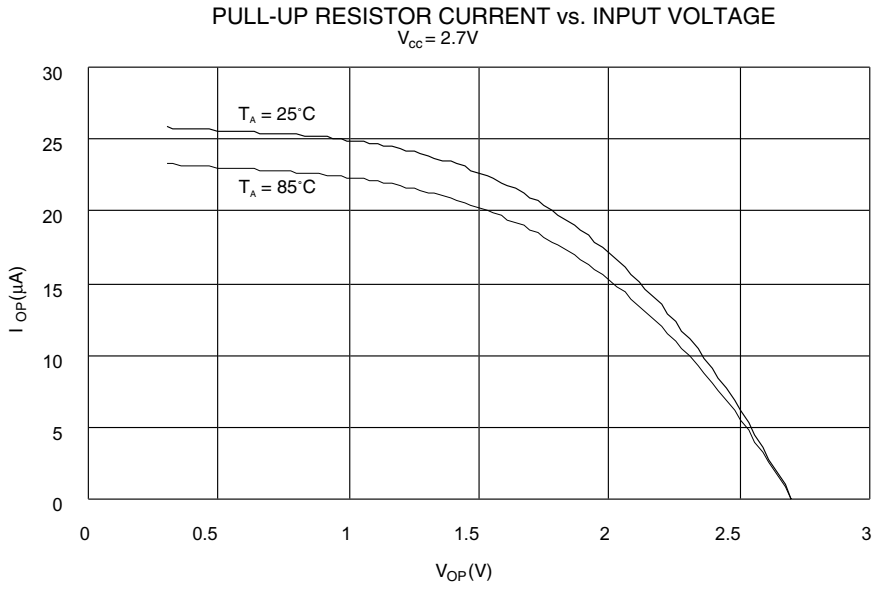


Figure 127. I/O Pin Sink Current vs. Output Voltage

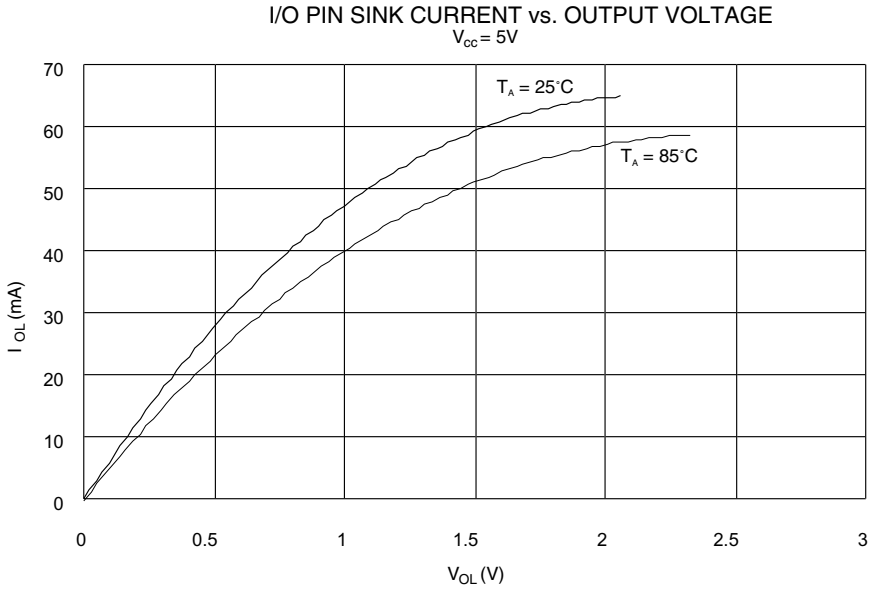


Figure 128. I/O Pin Source Current vs. Output Voltage

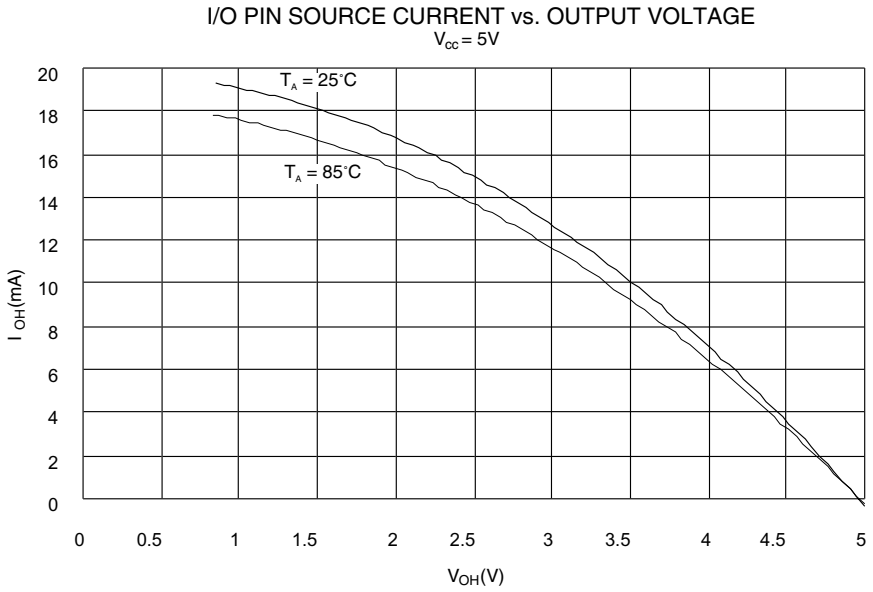


Figure 129. I/O Pin Sink Current vs. Output Voltage

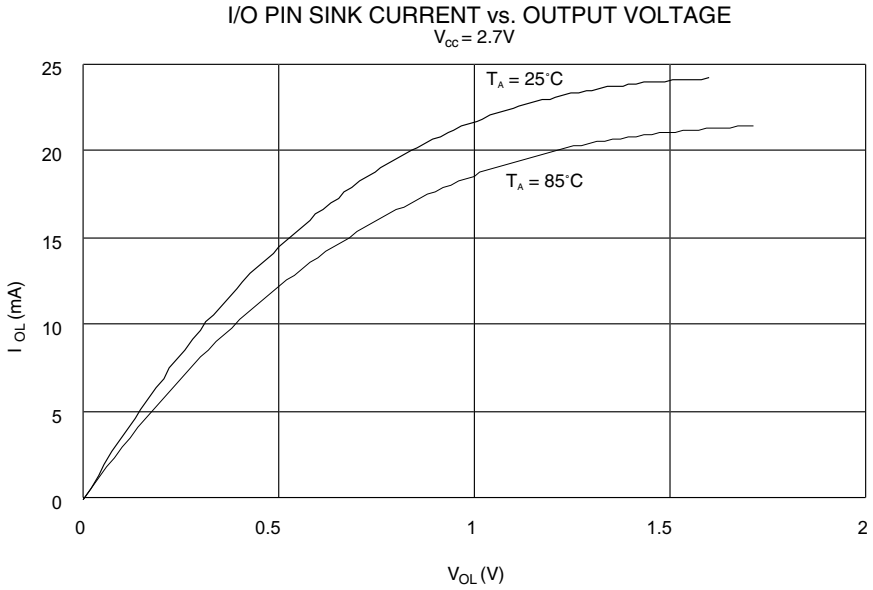


Figure 130. I/O Pin Source Current vs. Output Voltage

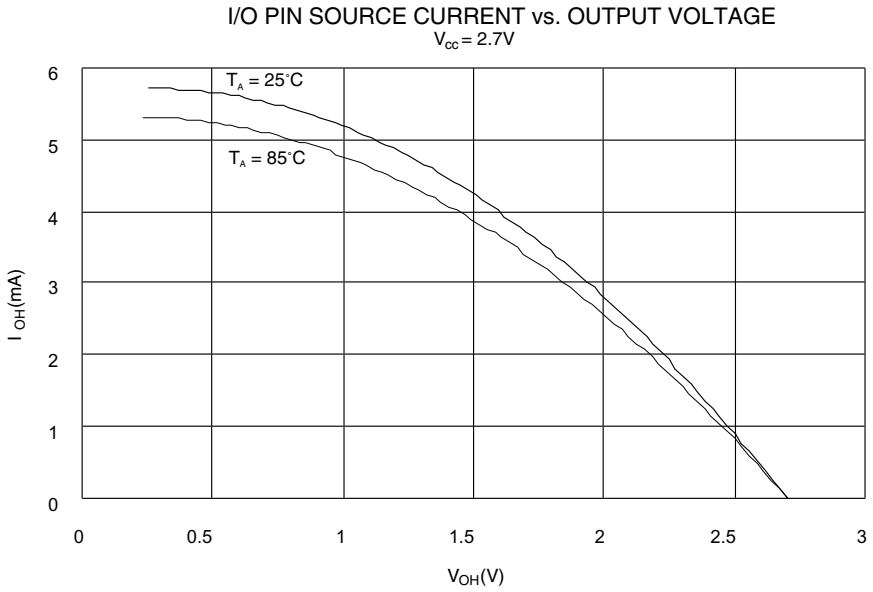


Figure 131. I/O Pin Input Threshold Voltage vs. V_{CC}

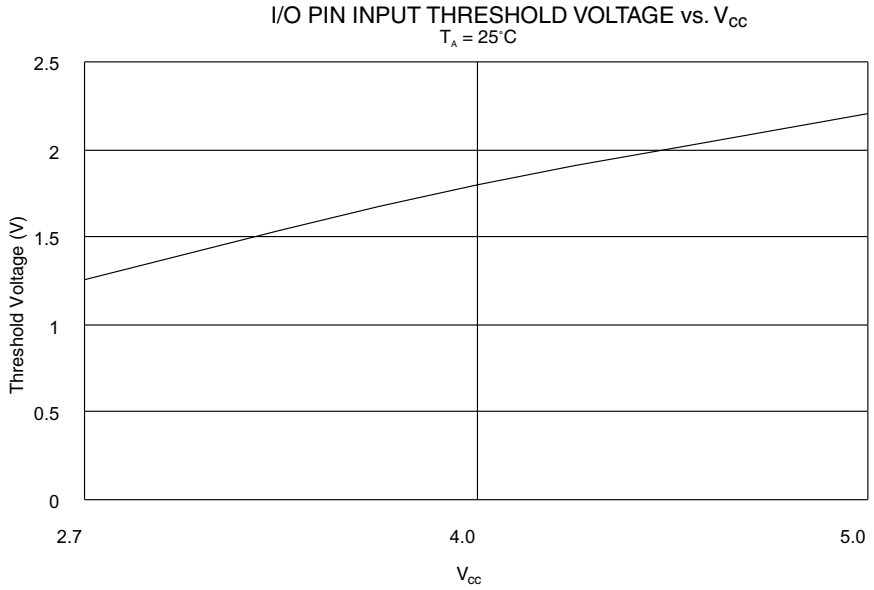
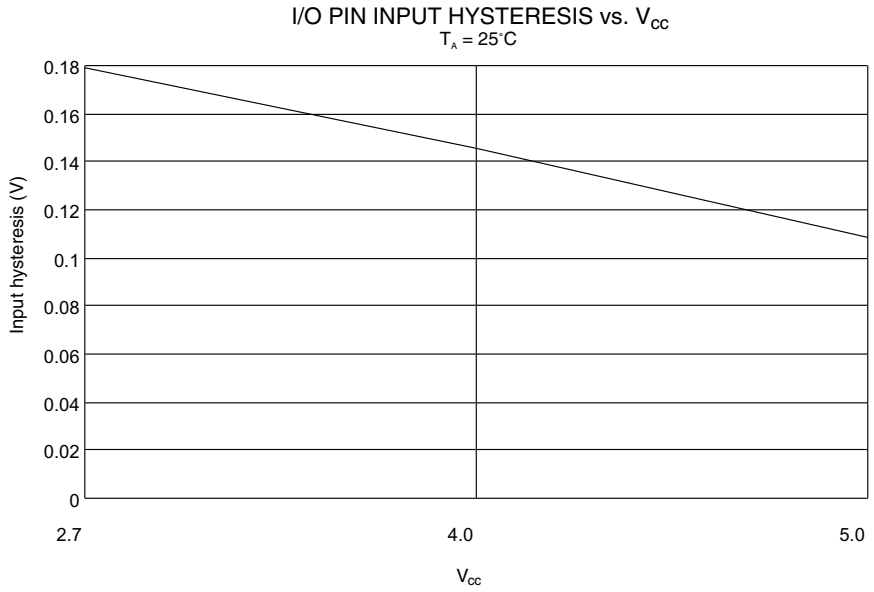


Figure 132. I/O Pin Input Hysteresis vs. V_{CC}



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	21
$3E ($5E)	SPH	-	-	-	-	SP11	SP10	SP9	SP8	22
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	22
$3C ($5C)	OCR0	Timer/Counter0 Output Compare Register								46
$3B ($5B)	GICR	INT1	INT0	INT2	-	-	-	IVSEL	IVCE	32
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	-	-	-	-	-	34
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	35
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	35
$37 ($57)	SPMCR	-	ASB	-	ASRE	BLBSET	PGWRT	PGERS	SPMEN	178
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	102
$35 ($55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	37
$34 ($54)	MCUCSR	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	30
$33 ($53)	TCCR0	FOC0	PWM0	COM01	COM00	CTC0	CS02	CS01	CS00	46
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								48
$31 ($51)	OSCCAL	Oscillator Calibration Register								40
	OCRD	On-chip Debug Register								156
$30 ($50)	SFIOR	-	-	-	-	ACME	PUD	PSR2	PSR10	44
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM11	PWM10	55
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	56
$2D ($4D)	TCNT1H	Timer/Counter1 - Counter Register High Byte								57
$2C ($4C)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								57
$2B ($4B)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								58
$2A ($4A)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								58
$29 ($49)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								58
$28 ($48)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								58
$27 ($47)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								59
$26 ($46)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								59
$25 ($45)	TCCR2	FOC2	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	46
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								48
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								48
$22 ($42)	ASSR	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB	51
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	63
$20 ($40)	UBRRH	URSEL	-	-	-	UBRR[11:8]				97
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	95
$1F ($3F)	EEARH	-	-	-	-	-	-	EEAR9	EEAR8	65
$1E ($3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	65
$1D ($3D)	EEDR	EEPROM Data Register								65
$1C ($3C)	EECR	-	-	-	-	EERIE	EEMWE	EEWE	EERE	66
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	134
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	134
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	134
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	136
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	136
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	136
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	143
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	143
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	143
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	148
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	148
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	148
$0F ($2F)	SPDR	SPI Data Register								72
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	71
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	70
$0C ($2C)	UDR	USART I/O Data Register								93
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	93
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	94
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								97
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	121
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	128
$06 ($26)	ADCSR	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	130
$05 ($25)	ADCH	ADC Data Register High Byte								131
$04 ($24)	ADCL	ADC Data Register Low Byte								131
$03 ($23)	TWDR	2-wire Serial Interface Data Register								104
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	104

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$01 ($21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	-	-	103
$00 ($20)	TWBR	2-wire Serial Interface Bit Rate Register								101

Notes:
1. When the OCDEN fuse is unprogrammed, the OSCCAL register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR register.
2. Refer to the USART description for details on how to access UBRRH and UCSRC.
3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
4. 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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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
BREAK		Break	For On-chip Debug Only	None	N/A

Ordering Information

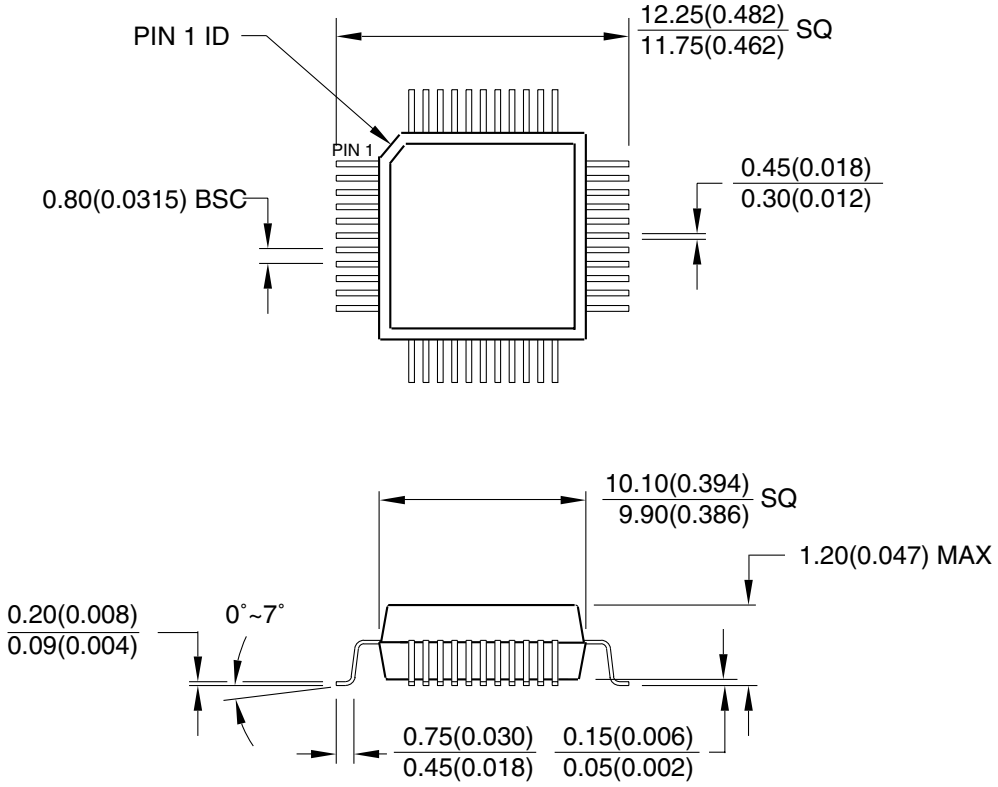
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
4	2.7-5.5V	ATmega323L-4AC ATmega323L-4PC	44A 40P6	Commercial (0°C to 70°C)
		ATmega323L-4AI ATmega323L-4PI	44A 40P6	Industrial (-40°C to 85°C)
8	4.0-5.5V	ATmega323-8AC ATmega323-8PC	44A 40P6	Commercial (0°C to 70°C)
		ATmega323-8AI ATmega323-8PI	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-pin, 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

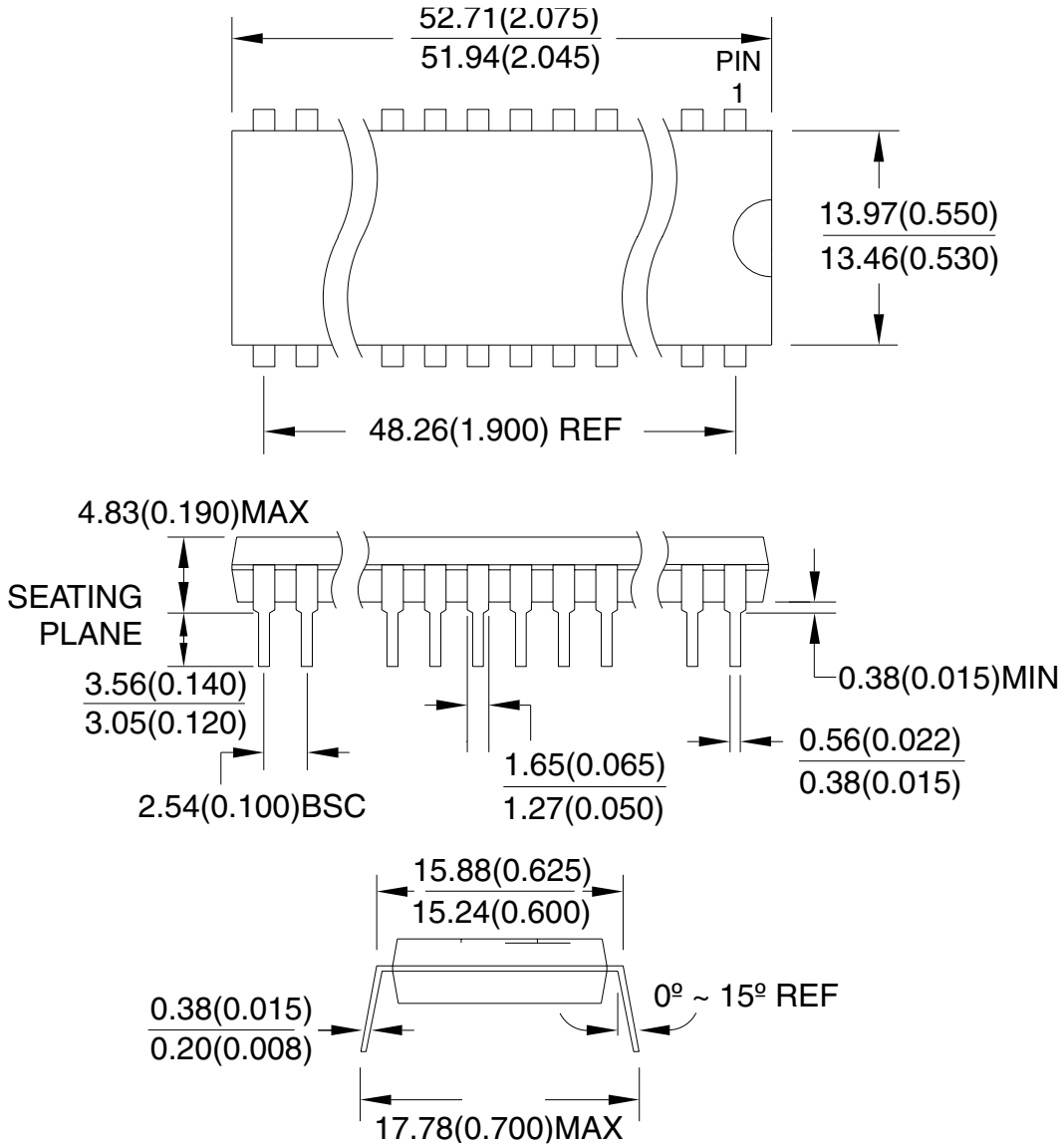


*Controlling dimension: millimetter

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

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
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 Product Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Atmel Heilbronn
Theresienstrasse 2
POB 3535
D-74025 Heilbronn, Germany
TEL (49) 71 31 67 25 94
FAX (49) 71 31 67 24 23

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 0 2 40 18 18 18
FAX (33) 0 2 40 18 19 60

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

© Atmel Corporation 2001.
Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

ATMEL® and AVR® are the registered trademarks of Atmel.

Other terms and product names may be the trademarks of others.

Printed on recycled paper.

1457E–11/01/xM