ATmega324PB

AVR Microcontroller with Core Independent Peripherals and PicoPower technology

Introduction

The picoPower® ATmega324PB 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 ATmega324PB achieves throughputs close to 1MIPS per MHz. This empowers system designers to optimize the device for power consumption versus processing speed.

Feature

High Performance, Low Power AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-Volatile Memory Segments
  - 32KBytes of In-System Self-Programmable Flash Program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 Years at 85°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-Scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits Through the JTAG Interface
- Peripheral Features
  - Peripheral Touch Controller (PTC)
    - Capacitive Touch Buttons, Sliders and Wheels
    - 32 Self-Sap Channels and 256 Mutual Cap Channels
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode

  - Three 16-bit Timer/Counters with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Ten PWM Channels
  - 8-Channel 10-Bit ADC
    - Differential Mode with Selectable Gain at 1×, 10× or 200×
  - Three Programmable Serial USARTs
  - Two Master/Slave SPI Serial Interfaces
  - Two Byte-oriented 2-wire Serial Interfaces (Philips I^2C Compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal 8 MHz Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
  - Clock Failure Detection Mechanism and Switch to Internal 8 MHz RC Oscillator in case of Failure
  - Individual Serial Number to Represent a Unique ID
- I/O and Packages
  - 39 Programmable I/O Lines
  - 44-Pin TQFP and 44-Pin QFN /MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 105°C
- Speed Grade:
  - 0 - 4MHz @ 1.8 - 5.5V
  - 0 - 10MHz @ 2.7 - 5.5.V
  - 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.24mA
  - Power-Down Mode: 0.2µA
  - Power-Save Mode: 1.3µA (Including 32kHz RTC)

Table of Contents

1. Description

The ATmega324PB 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 ATmega324PB achieves throughputs close to 1MIPS per MHz. This empowers system designers to optimize the device for power consumption versus processing speed.

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 a 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 ATmega324PB provides the following features: 32K bytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 39 general purpose I/O lines, 32 general purpose working registers, *five* flexible Timer/Counters with compare modes, internal and external interrupts, three serial programmable USART, two byte-oriented 2-wire Serial Interface (I2C), two SPI serial port, a 8-channel 10-bit ADC with optional differential input stage with programmable gain, a programmable Watchdog Timer with internal Oscillator, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming, Clock failure detection mechanism and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. PTC with enabling up to 32 self-cap and 256 mutual-cap sensors. 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. Also ability to run PTC in power-save mode/ wake-up on touch and Dynamic on/off of PTC analog and digital portion. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer, PTC, 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.

The device is manufactured using high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the ATmega324PB is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega324PB is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

2. Configuration Summary

Features	ATmega324PB
Pin count	44
Flash (KB)	32
SRAM (KB)	2
EEPROM (KB)	1
General Purpose I/O lines	39
SPI	2
TWI (I^2C)	2
USART	3
ADC	10-bit 15ksps
Differential ADC mode	Available
ADC channels	8
AC	1
8-bit Timer/Counters	2
16-bit Timer/Counters	3
PWM channels	10
PTC	Available
Peripheral Touch Controller (PTC) channels (X- x Y-Lines) for mutual capacitance	256 (16 x 16)
Peripheral Touch Controller (PTC) channels for self capacitance (Y-Lines only)	32
Clock Failure Detector (CFD)	Available
Output Compare Modulator (OCM1C2)	Available

3. Ordering Information

Speed [MHz]	Power Supply [V]	Ordering Code[2]	Package[1]	Operational Range
20	1.8 - 5.5	ATmega324PB-AU ATmega324PB-AUR[3] ATmega324PB-MU ATmega324PB-MUR[3]	44A 44A 44M1 44M1	Industrial (-40°C to 85°C)
		ATmega324PB-AN ATmega324PB-ANR[3] ATmega324PB-MN ATmega324PB-MNR[3]	44A 44A 44M1 44M1	Industrial (-40°C to 105°C)

Note: 1. This device can also be supplied in wafer form. Contact your local sales office for detailed ordering information and minimum quantities.

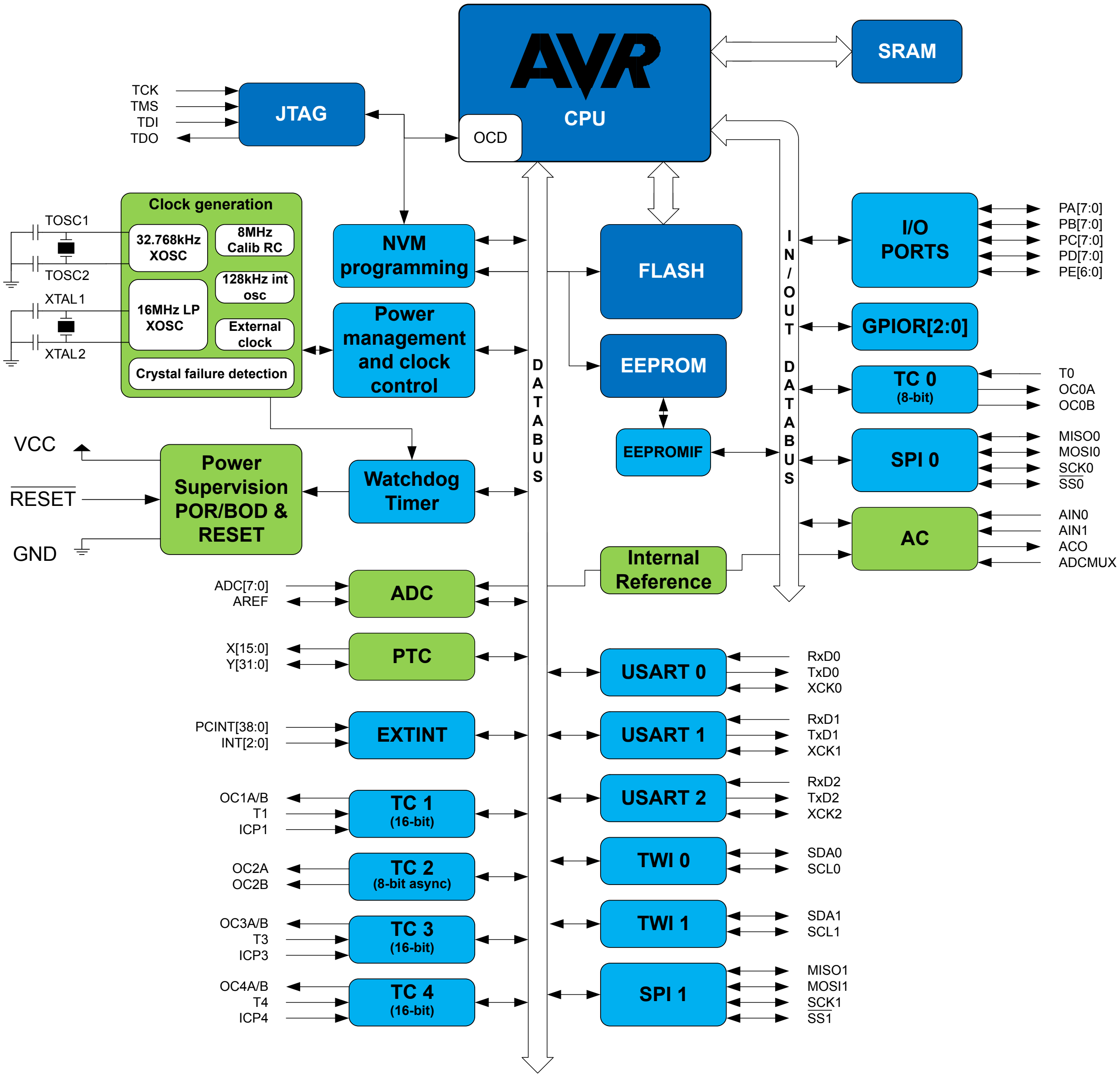
2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

3. Tape & Reel.

Package Type	
44A	44-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)
44M1	44-pad, 7 x 7 x 0.9mm body, Lead Pitch 0.50mm, Very-thin Fine pitch, Quad Flat No Lead Package/Quad Flat No-Lead/Micro Lead Frame Package (VQFN/QFN/MLF)

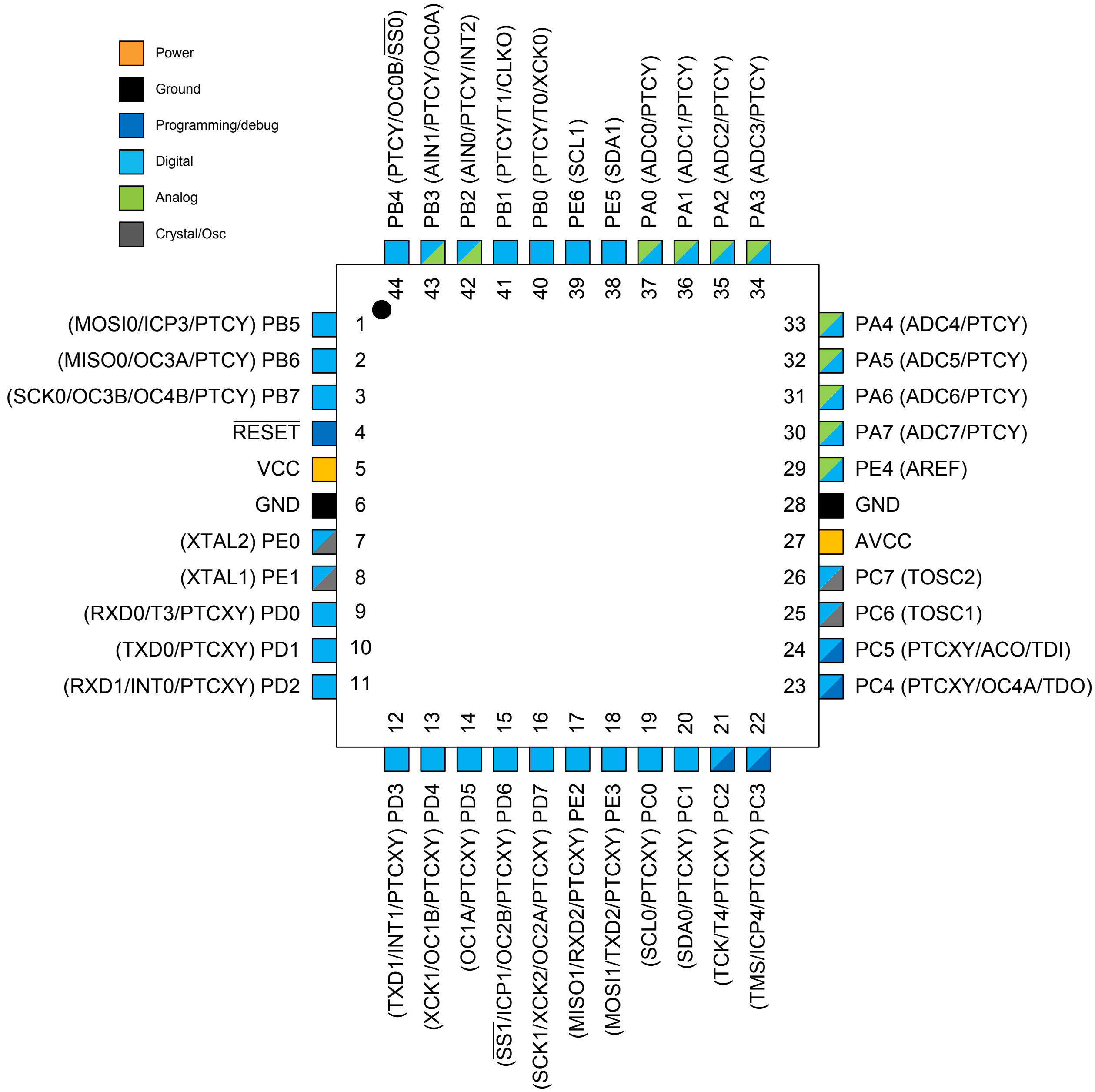
4. Block Diagram

Figure 4-1. Block Diagram



5. Pin Configurations

Figure 5-1. Pinout ATmega324PB



5.1 Pin Descriptions

5.1.1 VCC

Digital supply voltage.

5.1.2 GND

Ground.

5.1.3 Port A (PA[7:0])

This port serves as analog inputs to the Analog-to-digital Converter.

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

5.1.4 Port B (PB[7:0])

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

This port also serves the functions of various special features.

5.1.5 Port C (PC[7:0])

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

This port also serves the functions of the JTAG interface, along with special features.

5.1.6 Port D (PD[7:0])

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

This port also serves the functions of various special features.

5.1.7 Port E (PE6:0) XTAL1/XTAL2/AREF

This is a 7-bit bi-directional GPIO port with internal pull-up resistors (selected for each pin). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. Port pins are tri-stated during a reset condition, even if the clock is not running. PE0 and PE1 are multiplexed with XTAL1 and XTAL2 input. PE4 is multiplexed with AREF for the A/D Converter.

5.1.8 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

5.1.9 AVCC

AVCC is the supply voltage pin for Port A, PE4 (AREF) and the Analog-to-digital Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

6. I/O Multiplexing

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions.

The following table describes the peripheral signals multiplexed to the PORT I/O pins.

Table 6-1. PORT Function Multiplexing

No.	PAD	EXTINT	PCINT	ADC/AC	PTC X	PTC Y	OSC	T/C # 0	T/C # 1	USART	I2C	SPI	JTAG
1	PB[5]		PCINT13			Y29		ICP3				MOSI0	
2	PB[6]		PCINT14			Y30		OC3A				MISO0	
3	PB[7]		PCINT15			Y31		OC3B	OC4B			SCK0	
4	$\overline{\text{RESET}}$												
5	VCC												
6	GND												
7	PE[0]		PCINT32				XTAL2						
8	PE[1]		PCINT33				XTAL1						
9	PD[0]		PCINT24		X0	Y8		T3		RxD0			
10	PD[1]		PCINT25		X1	Y9				TxD0			
11	PD[2]	INT0	PCINT26		X2	Y10				RxD1			
12	PD[3]	INT1	PCINT27		X3	Y11				TXD1			
13	PD[4]		PCINT28		X4	Y12			OC1B	XCK1			
14	PD[5]		PCINT29		X5	Y13			OC1A				
15	PD[6]		PCINT30		X6	Y14		OC2B	ICP1			$\overline{\text{SS1}}$	
16	PD[7]		PCINT31		X7	Y15		OC2A		XCK2		SCK1	
17	PE[2]				X8	Y16				RxD2		MISO1	
18	PE[3]				X9	Y17				TxD2		MOSI1	
19	PC[0]		PCINT16		X10	Y18					SCL0		
20	PC[1]		PCINT17		X11	Y19					SDA0		
21	PC[2]		PCINT18		X12	Y20			T4				TCK
22	PC[3]		PCINT19		X13	Y21			ICP4				TMS
23	PC[4]		PCINT20		X14	Y22			OC4A				TDO
24	PC[5]		PCINT21	ACO	X15	Y23							TDI
25	PC[6]		PCINT22				TOSC1						
26	PC[7]		PCINT23				TOSC2						
27	AVCC												
28	GND												
29	PE[4]			AREF									
30	PA[7]		PCINT7	ADC7		Y7							
31	PA[6]		PCINT6	ADC6		Y6							
32	PA[5]		PCINT5	ADC5		Y5							
33	PA[4]		PCINT4	ADC4		Y4							
34	PA[3]		PCINT3	ADC3		Y3							

No.	PAD	EXTINT	PCINT	ADC/AC	PTC X	PTC Y	OSC	T/C # 0	T/C # 1	USART	I2C	SPI	JTAG
35	PA[2]		PCINT2	ADC2		Y2							
36	PA[1]		PCINT1	ADC1		Y1							
37	PA[0]		PCINT0	ADC0		Y0							
38	PE[5]										SDA1		
39	PE[6]										SCL1		
40	PB[0]		PCINT8			Y24		T0		XCK0			
41	PB[1]		PCINT9			Y25	CLKO		T1				
42	PB[2]	INT2	PCINT10	AIN0		Y26							
43	PB[3]		PCINT11	AIN1		Y27		OC0A					
44	PB[4]		PCINT12			Y28		OC0B				$\overline{\text{SS0}}$	

7. General Information

7.1 Resources

A comprehensive set of development tools, application notes, and datasheets are available for download on http://www.microchip.com/design-centers/8-bit/microchip-avr-mcus.

7.2 About Code Examples

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

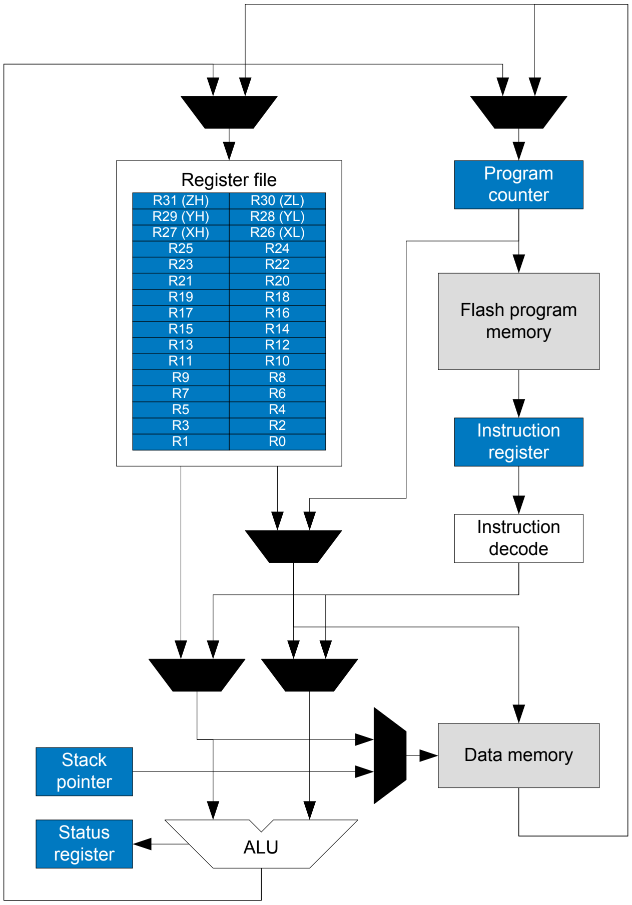
For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

8. AVR CPU Core

8.1 Overview

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

Figure 8-1. Block Diagram of the AVR Architecture



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

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

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

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

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

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

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

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

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

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, this device has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

8.2 ALU – Arithmetic Logic Unit

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

Related Links

Instruction Set Summary

8.3 Status Register

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

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

8.3.1 Status Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SREG
Offset: 0x5F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x3F

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – I: Global Interrupt Enable
The Global Interrupt Enable bit must be set for interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

Bit 6 – T: Copy Storage
The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

Bit 5 – H: Half Carry Flag
The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Flag is useful in BCD arithmetic. See the *Instruction Set Description* for detailed information.

Bit 4 – S: Sign Flag, 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 arithmetic. 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.

8.4 General Purpose Register File

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

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

Figure 8-2. AVR CPU General Purpose Working Registers

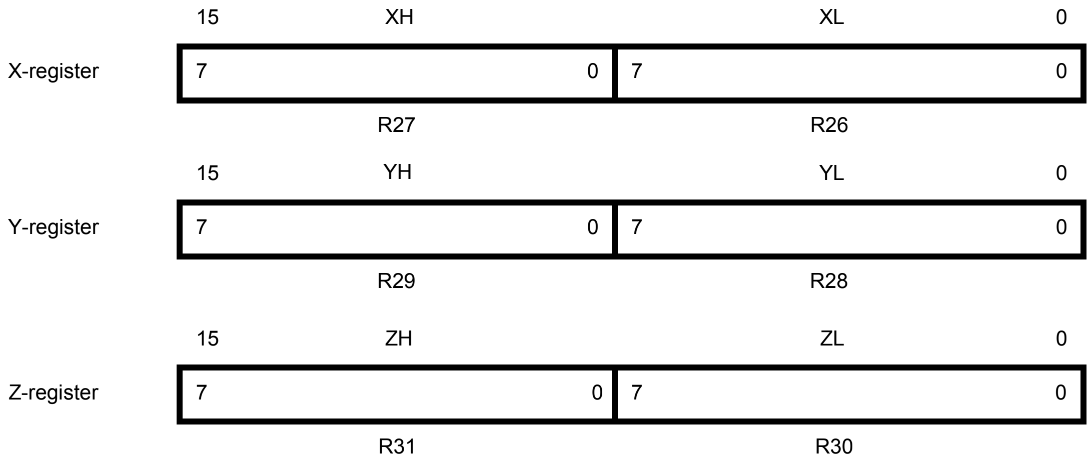
register Low Byte

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions. As shown in the figure, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer registers can be set to index any register in the file.

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

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

Figure 8-3. The X-, Y-, and Z-registers



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

Related Links

Instruction Set Summary

8.5 Stack Pointer

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

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer. The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM. See the table for Stack Pointer details.

Table 8-1. Stack Pointer Instructions

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

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

8.5.1 Stack Pointer Register Low and High byte

The SPL and SPH register pair represents the 16-bit value, SP.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these offset addresses.

Name: SPL and SPH
Offset: 0x5D
Reset: 0x8FF
Property: When addressing I/O Registers as data space the offset address is 0x3D

Bit	15	14	13	12	11	10	9	8
					SP[11:8]			
Access	R	R	R	R	RW	RW	RW	RW
Reset	0	0	0	0	1	0	0	0

Bit	7	6	5	4	3	2	1	0
	SP[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1

Bits 11:0 – SP[11:0]: Stack Pointer Register
SPL and SPH are combined into SP.

8.6 Accessing 16-bit Registers

The AVR data bus is 8 bits wide, and so accessing 16-bit registers requires atomic operations. These registers must be byte-accessed using two read or write operations. 16-bit registers are connected to the 8-bit bus and a temporary register using a 16-bit bus.

For a write operation, the low byte of the 16-bit register must be written before the high byte. The low byte is then written into the temporary register. When the high byte of the 16-bit register is written, the temporary register is copied into the low byte of the 16-bit register in the same clock cycle.

For a read operation, the low byte of the 16-bit register must be read before the high byte. When the low byte register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read. When the high byte is read, it is then read from the temporary register.

This ensures that the low and high bytes of 16-bit registers are always accessed simultaneously when reading or writing the register.

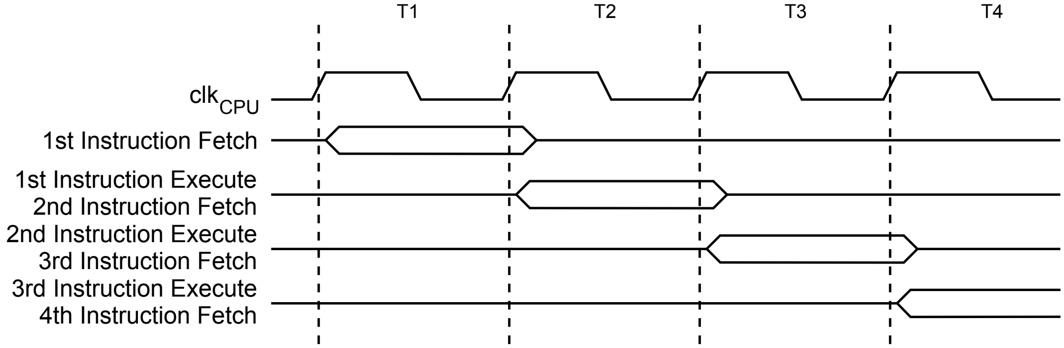
Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

The temporary registers can also be read and written directly from user software.

8.7 Instruction Execution Timing

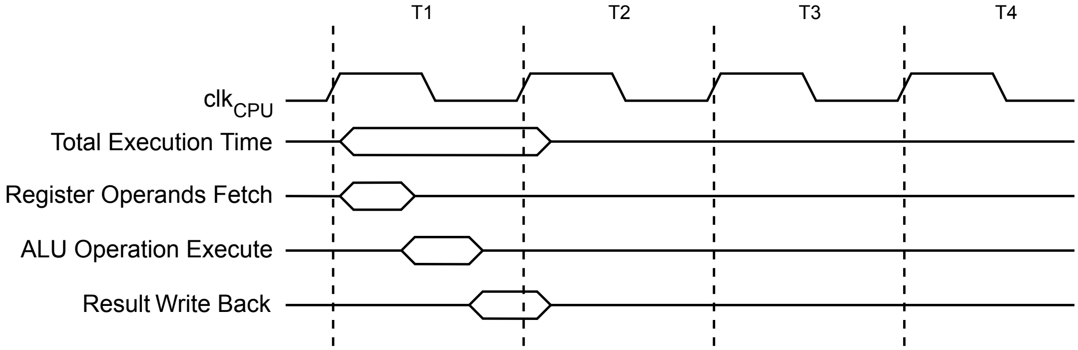
This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used. The Figure below shows the parallel instruction fetches and instruction 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 8-4. The Parallel Instruction Fetches and Instruction Executions



The following figure 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 8-5. Single Cycle ALU Operation



8.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. They have determined priority levels: The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse.

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

There are basically two types of interrupts:

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

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

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

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

Assembly Code Example(1)

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

C Code Example(1)

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

1. Refer to *About Code Examples*.

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

Assembly Code Example(1)

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

C Code Example(1)

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

1. Refer to *About Code Examples*.

Related Links

Memory Programming

Boot Loader Support – Read-While-Write Self-Programming

About Code Examples

8.8.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode. A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

9. AVR Memories

9.1 Overview

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

9.2 In-System Reprogrammable Flash Program Memory

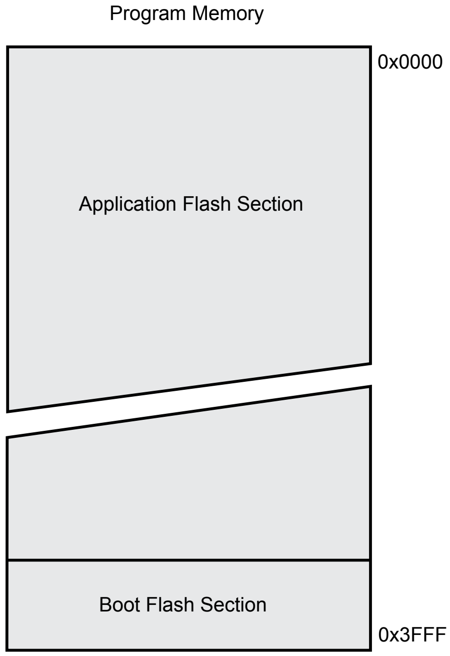
The ATmega324PB contains 32K bytes on-chip in-system reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 32K x 16. For software security, the Flash Program memory space is divided into two sections - Boot Loader Section and Application Program Section in the device .

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega324PB Program Counter (PC) is 15 bits wide, thus addressing the 32K program memory locations. The operation of the Boot Program section and associated Boot Lock bits for software protection are described in detail in *Boot Loader Support – Read-While-Write Self-Programming*. Refer to *Memory Programming* for the description on Flash data serial downloading using the SPI pins or the JTAG interface.

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

Timing diagrams for instruction fetch and execution are presented in *Instruction Execution Timing*.

Figure 9-1. Program Memory Map ATmega324PB



Related Links

BTLDR - Boot Loader Support – Read-While-Write Self-Programming

MEMPROG- Memory Programming

Instruction Execution Timing

9.3 SRAM Data Memory

The following figure shows how the device SRAM Memory is organized.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

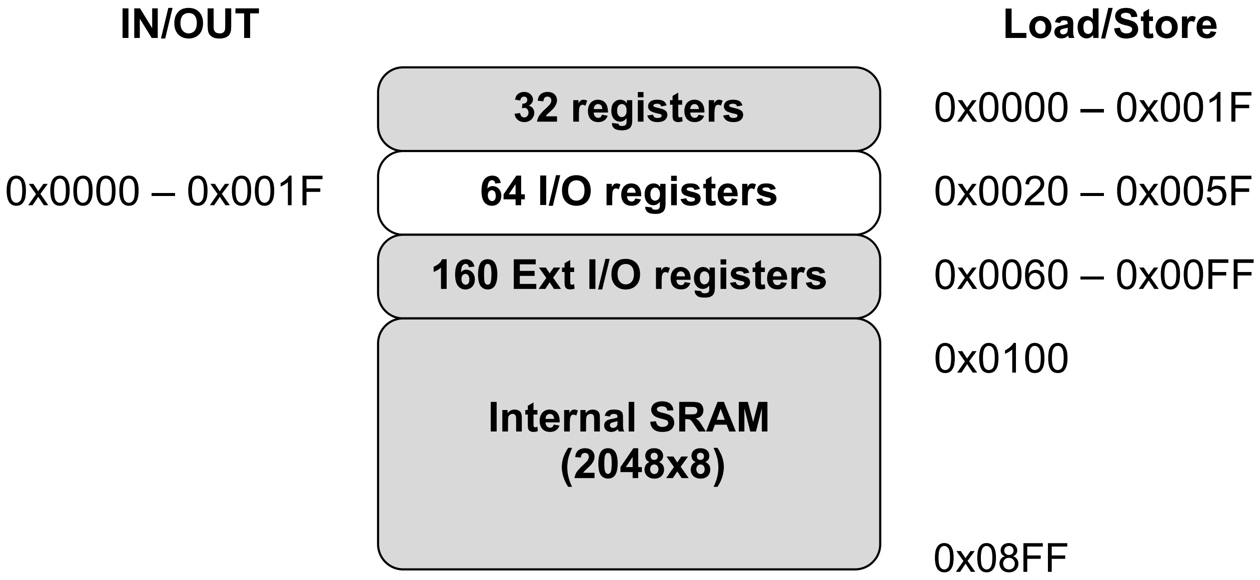
The lower 4352 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 4096 locations address the internal data SRAM.

The five different addressing modes for the data memory cover:

- Direct
  - The direct addressing reaches the entire data space.
- Indirect with Displacement
  - The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.
- Indirect
  - In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.
- Indirect with Pre-decrement
  - The address registers X, Y, and Z are decremented.
- Indirect with Post-increment
  - The address registers X, Y, and Z are incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 2K bytes of internal data SRAM in the device are all accessible through all these addressing modes.

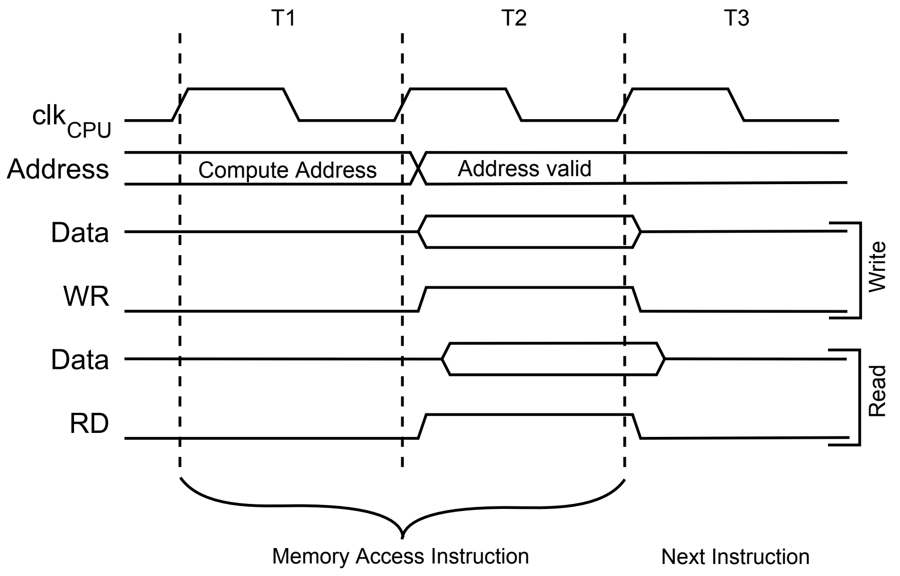
Figure 9-2. Data Memory Map with 2048 byte internal data SRAM



9.3.1 Data Memory Access Times

The internal data SRAM access is performed in two clk_{CPU} cycles as described in the following Figure.

Figure 9-3. On-chip Data SRAM Access Cycles



9.4 EEPROM Data Memory

The ATmega324PB 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 in the following, specifying the EEPROM Address registers, the EEPROM Data register, and the EEPROM Control register.

See the related links for a detailed description on EEPROM Programming in SPI or Parallel Programming mode.

Related Links

MEMPROG- Memory Programming

9.4.1 EEPROM Read/Write Access

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

The write access time for the EEPROM is given in Table 9-2. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. Please refer to Preventing EEPROM Corruption for details on how to avoid problems in these situations.

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

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

9.4.2 Preventing EEPROM Corruption

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

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

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

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

9.5 I/O Memory

The I/O space definition of the device is shown in the *Register Summary*.

All device I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00-0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.

When using the I/O specific commands IN and OUT, the I/O addresses 0x00-0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The device is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60..0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

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 '1' to them; this is described in the flag descriptions. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00-0x1F only.

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

Related Links

MEMPROG- Memory Programming

Instruction Set Summary

9.5.1 General Purpose I/O Registers

The device contains three General Purpose I/O Registers, General Purpose I/O Register 0/1/2 (GPIOR 0/1/2). These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

9.6 Register Description

9.6.1 Accessing 16-bit Registers

The AVR data bus is 8 bits wide, and so accessing 16-bit registers requires atomic operations. These registers must be byte-accessed using two read or write operations. 16-bit registers are connected to the 8-bit bus and a temporary register using a 16-bit bus.

For a write operation, the low byte of the 16-bit register must be written before the high byte. The low byte is then written into the temporary register. When the high byte of the 16-bit register is written, the temporary register is copied into the low byte of the 16-bit register in the same clock cycle.

For a read operation, the low byte of the 16-bit register must be read before the high byte. When the low byte register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read. When the high byte is read, it is then read from the temporary register.

This ensures that the low and high bytes of 16-bit registers are always accessed simultaneously when reading or writing the register.

Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

The temporary registers can also be read and written directly from user software.

9.6.2 EEPROM Address Register Low and High Byte

The EEARL and EEARH register pair represents the 16-bit value, EEAR. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EEARL and EEARH
Offset: 0x41 [ID-000004d0]
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x21

Bit	15	14	13	12	11	10	9	8
					EEAR[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					x	x	x	x

Bit	7	6	5	4	3	2	1	0
	EEAR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 11:0 – EEAR[11:0]: 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.

9.6.3 EEPROM Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EEDR
Offset: 0x40 [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x20

Bit	7	6	5	4	3	2	1	0
	EEDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – EEDR[7:0]: 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.

9.6.4 EEPROM Control Register

Name: EECR
Offset: 0x3F [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1F

Bit	7	6	5	4	3	2	1	0
			EEPM[1:0]		EERIE	EEMPE	EEPE	EERE
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			x	x	0	0	x	0

Bits 5:4 – EEPM[1:0]: EEPROM Programming Mode Bits

The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations into two different operations. The Programming times for the different modes are shown in the table below. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 9-1. EEPROM Mode Bits

EEPM[1:0]	Programming Time	Operation
00	3.4ms	Erase and Write in one operation (Atomic Operation)
01	1.8ms	Erase Only
10	1.8ms	Write Only
11	-	Reserved for future use

Bit 3 – EERIE: EEPROM Ready Interrupt Enable

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEPE is cleared. The interrupt will not be generated during EEPROM write or SPM.

Bit 2 – EEMPE: EEPROM Master Write Enable

The EEMPE bit determines whether writing EEPE to '1' causes the EEPROM to be written.
When EEMPE is '1', setting EEPE within four clock cycles will write data to the EEPROM at the selected address.

If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to '1' by software, hardware clears the bit to zero after four clock cycles. See the description of the EEPE bit for an EEPROM write procedure.

Bit 1 – EEPE: EEPROM Write Enable

The EEPROM Write Enable Signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to '1' to write the value into the EEPROM. The EEMPE bit must be written to '1' before EEPE is written to '1', otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

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

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted.

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

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

Bit 0 – EERE: EEPROM Read Enable
The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a '1' to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

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

The calibrated Oscillator is used to time the EEPROM accesses. See the following table for typical programming times for EEPROM access from the CPU.

Table 9-2. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ. Programming Time
EEPROM write (from CPU)	26,368	3.3ms

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

Assembly Code Example(1)

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

C Code Example(1)

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
```

```
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE))
    ;
    /* Set up address and Data Registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMPE */
    EECR |= (1<<EEMPE);
    /* Start eeprom write by setting EEPE */
    EECR |= (1<<EEPE);
}
```

Note: (1) Please refer to *About Code Examples*

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

Assembly Code Example(1)

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

C Code Example(1)

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

1. Refer to *About Code Examples*.

9.6.5 GPIOR2 – General Purpose I/O Register 2

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GPIOR2
Offset: 0x4B [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2B

Bit	7	6	5	4	3	2	1	0
	GPIOR2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – GPIOR2[7:0]: General Purpose I/O

9.6.6 GPIOR1 – General Purpose I/O Register 1

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GPIOR1
Offset: 0x4A [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2A

Bit	7	6	5	4	3	2	1	0
	GPIOR1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – GPIOR1[7:0]: General Purpose I/O

9.6.7 GPIOR0 – General Purpose I/O Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GPIOR0
Offset: 0x3E [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1E

Bit	7	6	5	4	3	2	1	0
	GPIOR0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – GPIOR0[7:0]: General Purpose I/O

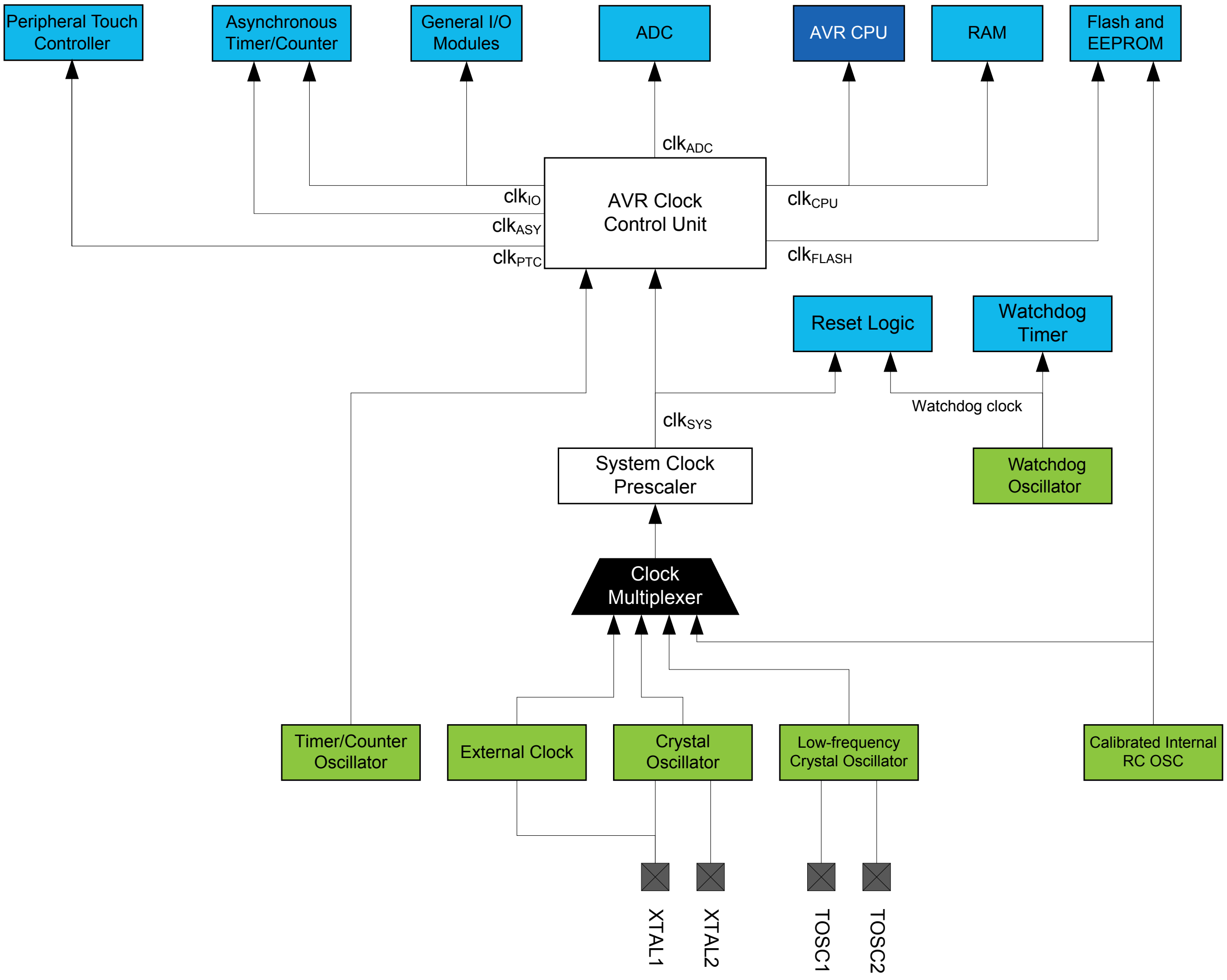
10. System Clock and Clock Options

10.1 Clock Systems and Their Distribution

The following figure illustrates the principal clock systems in the device and their distribution. All the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes. The clock systems are described in the following sections.

The system clock frequency refers to the frequency generated from the System Clock Prescaler. All clock outputs from the AVR Clock Control Unit runs in the same frequency.

Figure 10-1. Clock Distribution



10.1.1 CPU Clock – clk_{CPU}

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

10.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but the start condition detection in the USI module is carried out asynchronously when $clk_{I/O}$ is halted, TWI address recognition in all sleep modes.

Note: If a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses.

10.1.3 PTC Clock - clk_{PTC}

The PTC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise and power due to digital circuitry.

10.1.4 Flash Clock – clk_{FLASH}

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

10.1.5 Asynchronous Timer Clock – clk_{ASY}

The Asynchronous Timer clock allows Asynchronous Timer/Counters to be clocked directly from an external clock or an external 32kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

10.1.6 ADC Clock – clk_{ADC}

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

10.2 Clock Sources

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

Table 10-1. Device Clocking Options Select

Device Clocking Option	CKSEL[3:0]
Low Power Crystal Oscillator	1111 - 1000
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Note: For all fuses, '1' means unprogrammed while '0' means programmed.

10.2.1 Default Clock Source

The device is shipped with internal RC oscillator at 8.0MHz and with the fuse CKDIV8 programmed, resulting in 1.0MHz system clock. The start-up time is set to maximum, and the time-out period is enabled: CKSEL=0010, SUT=10, CKDIV8=0. This default setting ensures that all users can make their desired clock source setting using any available programming interface.

10.2.2 Clock Start-up Sequence

Any clock source needs a sufficient V_{CC} to start oscillating and a minimum number of oscillating cycles before it can be considered stable.

To ensure sufficient V_{CC}, the device issues an internal reset with a time-out delay (t_{TOUT}) after the device reset is released by all other reset sources. See the Related Links for a description of the start conditions for the internal reset. The delay (t_{TOUT}) is timed from the Watchdog Oscillator and the number of cycles in the delay is set by the SUTx and CKSELx fuse bits. The selectable delays are shown in the Table below. The frequency of the Watchdog Oscillator is voltage dependent.

Table 10-2. Number of Watchdog Oscillator Cycles

Typ. Time-out (V_{CC} = 5.0V)	Typ. Time-out (V_{CC} = 3.0V)
0ms	0ms
4ms	4.3ms
65ms	69ms

Main purpose of the delay is to keep the device in reset until it is supplied with minimum V_{CC}. The delay will not monitor the actual voltage, so it is required to select a delay longer than the V_{CC} rise time. If this is not possible, an internal or external Brown-Out Detection circuit should be used. A BOD circuit will ensure sufficient V_{CC} before it releases the reset, and the time-out delay can be disabled. Disabling the time-out delay without utilizing a Brown-Out Detection circuit is not recommended.

The oscillator is required to oscillate for a minimum number of cycles before the clock is considered stable. An internal ripple counter monitors the oscillator output clock, and keeps the internal reset active for a given number of clock cycles. The reset is then released and the device will start to execute. The recommended oscillator start-up time is dependent on the clock type, and varies from 6 cycles for an externally applied clock to 32K cycles for a low frequency crystal.

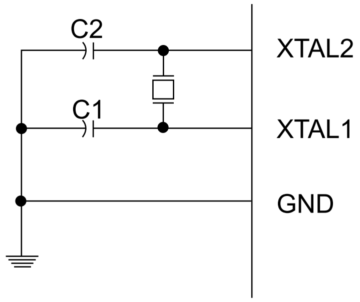
The start-up sequence for the clock includes both the time-out delay and the start-up time when the device starts up from reset. When starting up from Power-save or Power-down mode, V_{CC} is assumed to be at a sufficient level and only the start-up time is included.

10.2.3 Clock Source Connections

Pins 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 the Figure below. Either a quartz crystal or a ceramic resonator may be used.

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

Figure 10-2. Crystal Oscillator Connections



10.3 Low Frequency Crystal Oscillator

The Low-frequency Crystal Oscillator is optimized for use with a 32.768kHz watch crystal. When selecting crystals, load capacitance and crystal's Equivalent Series Resistance (ESR) must be taken into consideration. Both values are specified by the crystal vendor. The oscillator is optimized for very low power consumption, and thus when selecting crystals, consider the Maximum ESR Recommendations:

Table 10-3. Maximum ESR Recommendation for 32.768kHz Crystal

Crystal CL [pF]	Max. ESR [kΩ]
9.0	65
12.5	30

The Low-frequency Crystal Oscillator provides an internal load capacitance at each TOSC pin:

Table 10-4. Internal capacitance of Low-Frequency Oscillator

32kHz Osc. Type	Internal Pad Capacitance (XTAL1/TOSC1)	Internal Pad Capacitance (XTAL2/TOSC2)
C_i of system oscillator (XTAL pins)	6pF	6pF
C_i of timer oscillator (TOSC pins)	4pF	4pF

The capacitance (C_e+C_i) needed at each XTAL/TOSC pin can be calculated by using:

$$C_e + C_i = 2C_L - C_s$$

where:

- C_e - is optional external capacitors. (=C_1, C_2 as shown in Figure 10-2)
- C_i - is the pin capacitance in Table 10-4.
- C_L - is the load capacitance for a 32.768kHz crystal specified by the crystal vendor.
- C_S - is the total stray capacitance for one TOSC pin.

Crystals specifying a load capacitance (C_L) higher than 6pF require external capacitors applied according to above formula.

When this oscillator is selected, start-up times are determined by the SUT Fuses and CKSEL0 as shown in the following table.

Table 10-5. Start-up Times for the Low Frequency Crystal Oscillator Clock Selection

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL[0]	SUT[1:0]
BOD enabled	1K CK	14CK(1)	0	00
Fast rising power	1K CK	14CK + 4ms(1)	0	01
Slowly rising power	1K CK	14CK + 65ms(1)	0	10
Reserved			0	11
BOD enabled	32K CK	14CK	1	00
Fast rising power	32K CK	14CK + 4ms	1	01
Slowly rising power	32K CK	14CK + 65ms	1	10
Reserved			1	11

Note:

1. This option should only be used if frequency stability at start-up is not important for the application.

Related Links

Clock Source Connections

10.4 Low Power Crystal Oscillator

This Crystal Oscillator is a low power oscillator, with reduced voltage swing on the XTAL2 output. It gives the lowest power consumption, but is not capable of driving other clock inputs, and may be more susceptible to noise in noisy environments.

The crystal should be connected as described in Clock Source Connections. When selecting crystals, load capacitance must be taken into consideration. The capacitance (C_e+C_i) needed at each TOSC pin can be calculated by using:

$$C_e + C_i = 2C_L - C_s$$

where:

- C_e - is optional external capacitors. (= C_1, C_2 as shown in Figure 10-2)
- C_i - is the pin capacitance in Table 10-6.
- C_L - is the load capacitance specified by the crystal vendor.
- C_S - is the total stray capacitance for one XTAL pin.

Table 10-6. Internal capacitance of Low-Power Oscillator

32kHz Osc. Type	Internal Pad Capacitance (XTAL1)	Internal Pad Capacitance (XTAL2)
C_i of system oscillator (XTAL pins)	6pF	6pF

The Low Power Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL[3:1], as shown in the following table:

Table 10-7. Low Power Crystal Oscillator Operating Modes(1)

Frequency Range [MHz]	CKSEL[3:1](2)	Range for total capacitance of C1 and C2 [pF](4)
0.4 - 0.9	100(3)	–
0.9 - 3.0	101	12 - 22
3.0 - 8.0	110	12 - 22
8.0 - 16.0	111	12 - 22

Note:

1. If the crystal frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8. It must be ensured that the resulting divided clock meets the frequency specification of the device.
2. This is the recommended CKSEL settings for the difference frequency ranges.
3. This option should not be used with crystals, only with ceramic resonators.
4. The range of the
capacitance should not be larger than the value given in this table.

The CKSEL0 Fuse together with the SUT[1:0] Fuses select the start-up times, as shown in the following table:

Table 10-8. Start-up Times for the Low Power Crystal Oscillator Clock Selection

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT[1:0]
Ceramic resonator, fast rising power	258 CK	14CK + 4ms(1)	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65ms(1)	0	01
Ceramic resonator, BOD enabled	1K CK	14CK(2)	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4ms(2)	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65ms(2)	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65ms	1	11

Note:

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

Related Links

Clock Source Connections

10.5 Calibrated Internal RC Oscillator

By default, the internal RC oscillator provides an 8.0MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. The device is shipped with the CKDIV8 fuse programmed.

This clock may be selected as the system clock by programming the CKSEL fuses as shown in the following table. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL register and thereby automatically calibrates the RC Oscillator.

By changing the OSCCAL register from SW, it is possible to get a higher calibration accuracy than by using the factory calibration.

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.

Table 10-9. Internal Calibrated RC Oscillator Operating Modes

Frequency Range(1) [MHz]	CKSEL[3:0]
7.3 - 8.1	0010(2)

Note:

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

When this oscillator is selected, start-up times are determined by the SUT fuses:

Table 10-10. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection - SUT

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5V)	SUT[1:0]
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4ms	01
Slow rising power	6 CK	14CK + 65ms	10(1)
Reserved			11

Note:

1. The device is shipped with this option selected.

Related Links

Clock Characteristics

System Clock Prescaler

Calibration Byte

OSCCAL

10.6 128kHz Internal Oscillator

The 128kHz internal Oscillator is a low power Oscillator providing a clock of 128kHz. This clock may be select as the system clock by programming the CKSEL Fuses to '0011':

Table 10-11. 128kHz Internal Oscillator Operating Modes

Nominal Frequency[(1)]	CKSEL[3:0]
128kHz	0011

Note:

1. The 128kHz oscillator is a very low power clock source, and is not designed for high accuracy.

When this clock source is selected, start-up times are determined by the SUT Fuses:

Table 10-12. Start-Up Times for the 128kHz Internal Oscillator

Power Conditions	Start-Up Time from Power-down and Power-save	Additional Delay from Reset	SUT[1:0]
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4ms	01
Slowly rising power	6 CK	14CK + 65ms	10
Reserved			11

10.7 External Clock

To drive the device from an external clock source, EXTCLK should be driven as shown in the Figure below. To run the device on an external clock, the CKSEL Fuses must be programmed to '0000':

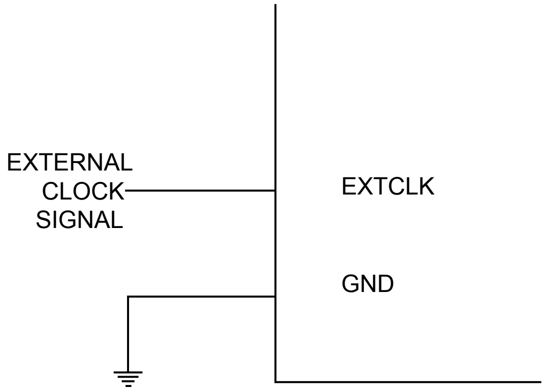
Table 10-13. External Clock Frequency

Frequency[(1)]	CKSEL[3:0]
0 - 20MHz	0000

Note:

1. If the cryatal frequency exceeds the specification of the device (depends on VCC), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8. It must be ensured that the resulting divided clock meets the frequency specification of the device.

Figure 10-3. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses:

Table 10-14. Start-Up Times for the External Clock Selection - SUT

Power Conditions	Start-Up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	SUT[1:0]
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4ms	01
Slowly rising power	6 CK	14CK + 65ms	10
Reserved			11

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

The System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation.

Related Links

System Clock Prescaler

10.8 Timer/Counter Oscillator

The device uses the same crystal oscillator for Low-frequency Oscillator and Timer/Counter Oscillator. See Low Frequency Crystal Oscillator for details on the oscillator and crystal requirements.

On this device, the Timer/Counter Oscillator Pins (TOSC1 and TOSC2) are shared with XTAL1 and XTAL2. When using the Timer/Counter Oscillator, the system clock needs to be four times the oscillator frequency. Due to this and the pin sharing, the Timer/Counter Oscillator can only be used when the Calibrated Internal RC Oscillator is selected as system clock source.

Applying an external clock source to TOSC1 can be done if the Enable External Clock Input bit in the Asynchronous Status Register (ASSR.EXCLK) is written to '1'. See the description of the Asynchronous Operation of Timer/Counter2 for further description on selecting external clock as input instead of a 32.768kHz watch crystal.

Related Links

Low Frequency Crystal Oscillator

OCR2B

ASSR

10.9 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT Fuse has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. The clock also will be output during reset, and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal RC Oscillator, can be selected when the clock is output on CLKO. If the System Clock Prescaler is used, it is the divided system clock that is output.

10.10 System Clock Prescaler

The device has a system clock prescaler, and the system clock can be divided by configuring the Clock Prescale Register (CLKPR). This feature can be used to decrease the system clock frequency and the power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in the CLKPR description.

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting. The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler - even if it were readable, the exact time it takes to switch from one clock division to the other cannot be exactly predicted. From the time the Clock Prescaler Selection bits (CLKPS[3:0]) values are written, it takes between T1 + T2 and T1 + 2 * T2 before the new clock frequency is active. In this interval, two active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

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

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to '1' and all other bits in CLKPR to zero: CLKPR=0x80.
2. Within four cycles, write the desired value to CLKPS[3:0] while writing a zero to CLKPCE: CLKPR=0x0N

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

Related Links

CLKPR

10.11 Register Description

10.11.1 Oscillator Calibration Register

Name: OSCCAL
Offset: 0x66
Reset: Device Specific Calibration Value
Property: -

Bit	7	6	5	4	3	2	1	0
	CAL [7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – CAL [7:0]: Oscillator Calibration Value
The Oscillator Calibration Register is used to trim the Calibrated Internal RC Oscillator to remove process variations from the oscillator frequency. A pre-programmed calibration value is automatically written to this register during chip reset, giving the Factory calibrated frequency as specified in the *Clock Characteristics* section of *Electrical Characteristics* chapter.. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in the *Clock Characteristics* section of *Electrical Characteristics* chapter.. Calibration outside that range is not guaranteed.

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

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL=0x7F gives a higher frequency than OSCCAL=0x80.

The CAL[6:0] bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range.

10.11.2 Clock Prescaler Register

Name: CLKPR
Offset: 0x61
Reset: Refer to the bit description
Property: -

Bit	7	6	5	4	3	2	1	0
	CLKPCE				CLKPS [3:0]			
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				x	x	x	x

Bit 7 – CLKPCE: Clock Prescaler Change Enable
The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

Bits 3:0 – CLKPS [3:0]: Clock Prescaler Select
These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in the table below.

The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of 8 at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The Application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 Fuse programmed.

Table 10-15. Clock Prescaler Select

CLKPS[3:0]	Clock Division Factor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

11. CFD - Clock Failure Detection mechanism

11.1 Overview

The Clock Failure Detection mechanism for the device is enabled by CFD fuse in the Extended Fuse Byte. CFD operates with 128kHz internal oscillator which will be enabled automatically when CFD is enabled.

11.2 Features

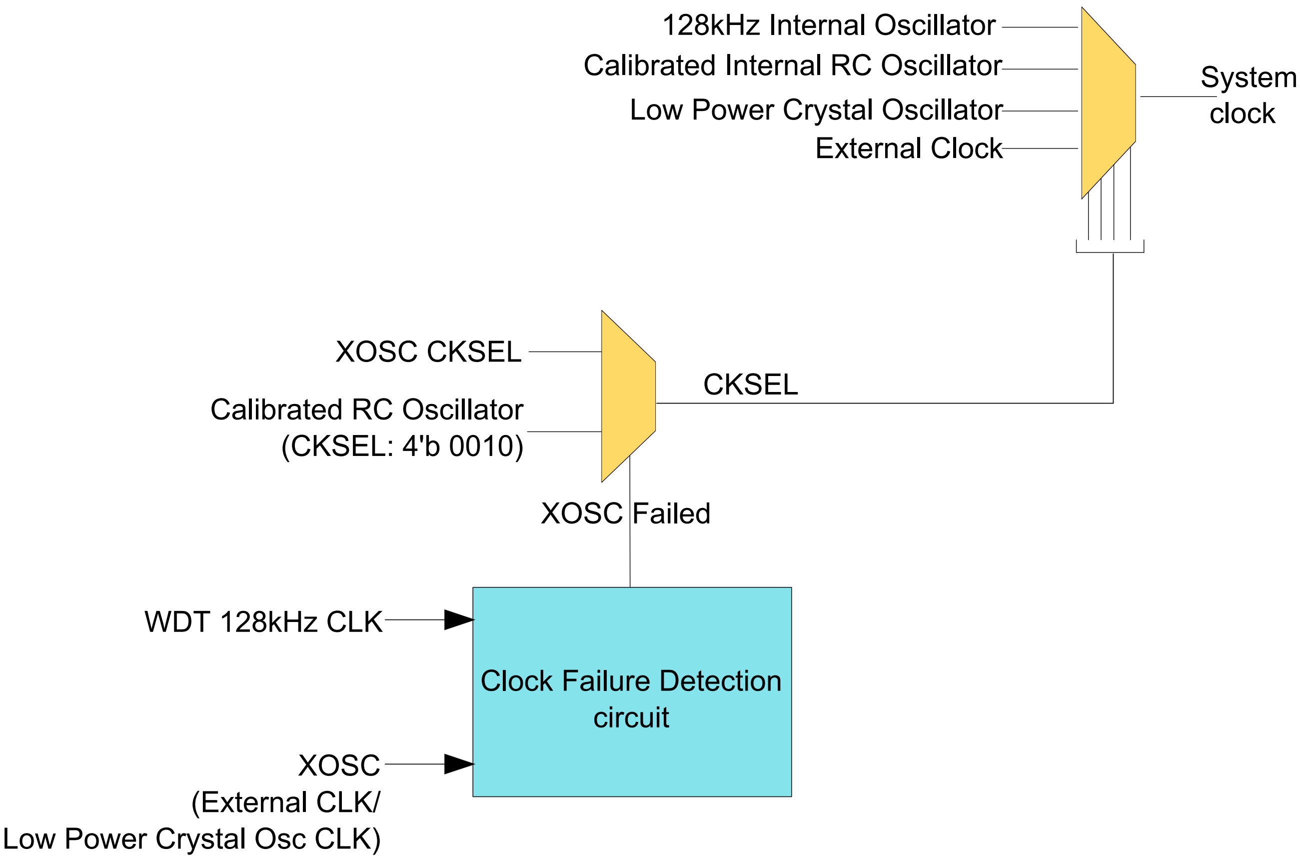
- Detection for the failure of the low power crystal oscillator and external clocks
- Operate with 128kHz internal oscillator
- Switch the clock to internal RC oscillator clock when clock failure happens
- Failure Detection Interrupt Flag (XFDIF) available for the status of CFD

11.3 Operations

The Clock Failure Detector (CFD) allows the user to monitor the low power crystal oscillator or external clock signal. CFD monitors XOSC clock and if it fails it will automatically switch to a safe clock. When operating on safe clock the device will switch back to XOSC clock after Power-On or External Reset, and continue monitoring XOSC clock for failures. The safe clock is derived from the 8MHz internal RC system clock. This allows to configure the safe clock in order to fulfill the operative conditions of the microcontroller.

Because the XOSC failure is monitored by the CFD circuit operating with the internal 128kHz oscillator, the current consumption of the 128kHz oscillator will be added into the total power consumption of the chip when CFD is enabled. CFD should be enabled only if the system clock (XOSC) frequency is above 256kHz.

Figure 11-1. System Clock Generation with CFD Mechanism



Clock Failure Detection

To start the CFD operation, the user must write a one to the CFD fuse bit in the Extended Fuse Byte (EFB.CFD). After the start or restart of the XOSC, the CFD does not detect failure until the start-up time is elapsed. Once the XOSC Start-Up Time is elapsed, the XOSC clock is constantly monitored.

If the external clock is not provided, the device will automatically switch to calibrated RC oscillator output.

When the failure is detected, the failure status is asserted, i.e Failure Detection Interrupt Flag bit in the XOSC Failure Detection Control And Status Register (XFDCSR.XFDIF) is set. The Failure Detection interrupt flag is generated, when the Interrupt Enable bit in the XOSC Failure Detection Control And Status Register (XFDCSR.XFDIE) is set. The XFDCSR.XFDIF reflects the current XOSC clock activity.

The detection will be automatically disabled when chip goes to power save/down sleep mode and enabled by itself when chip enters back to active mode.

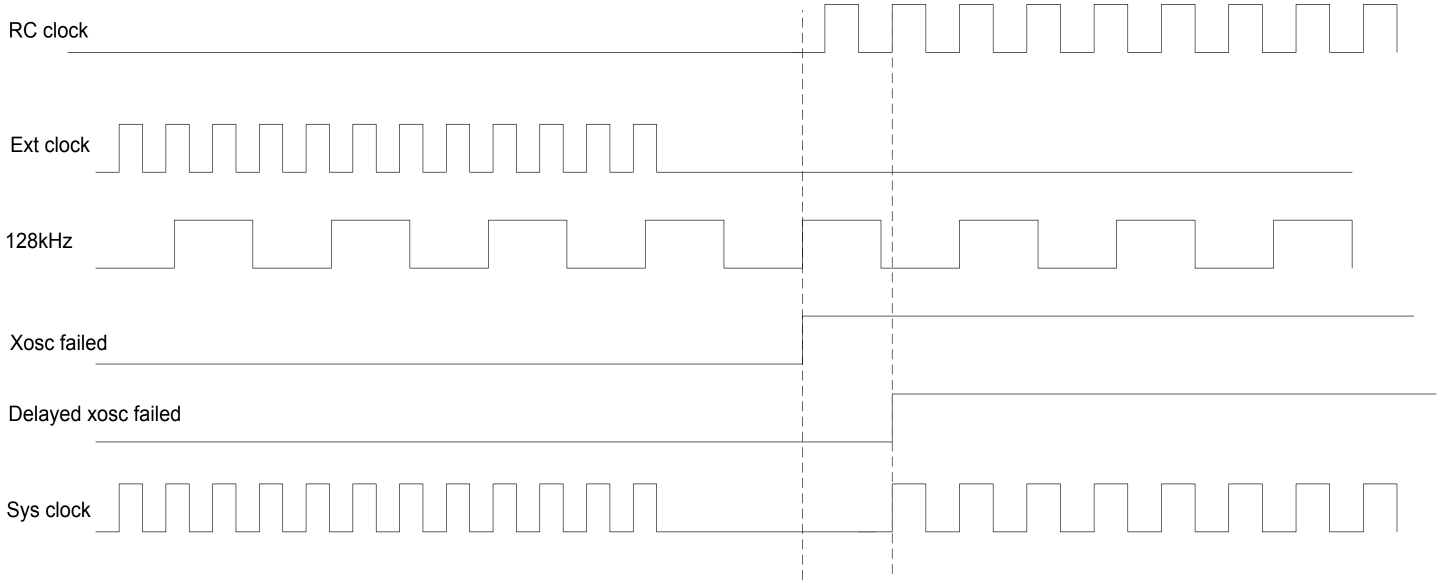
Clock Switch

When a clock failure is detected, the XOSC clock is replaced by the safe clock in order to maintain an active clock. The safe clock source is the calibrated RC oscillator clock (CKSEL: 4’b0010). The clock source can be downscaled with a configurable prescaler to ensure that the clock frequency does not exceed the operating conditions selected by the application after switching. To use to original clock source, user must provide a reset. When using CFD and clock failure has occurred the system operates using 1MHz internal fallback clock. The system will try to resume to original clock source either via Power-On-Reset (POR) or via external RESET.

11.4 Timing Diagram

The RC clock is enabled only after failure detection.

Figure 11-2. CFD mechanism timing diagram



11.5 Register Description

11.5.1 XOSC Failure Detection Control And Status Register

Name: XFDCSR
Offset: 0x62
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							XFDIF	XFDIE
Access							R	R/W
Reset							0	0

Bit 1 – XFDIF: Failure Detection Interrupt Flag
This bit is set when a failure is detected, and it can be cleared only by reset.

It serves as status bit for CFD.

Note: This bit is read only.

Bit 0 – XFDIE: Failure Detection Interrupt Enable
Setting this bit will enable the interrupt which will be issued when XFDIF is set. This bit is enable only. Once enabled, it is not possible for the user to disable.

12. PM - Power Management and Sleep Modes

12.1 Sleep Modes

The following Table shows the different sleep modes, BOD disable ability and their wake-up sources.

Table 12-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains						Oscillators		Wake-up Sources								Software BOD Disable
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	clk_{PTC}	Main Clock Source Enabled	Timer Oscillator Enabled	INT and PCINT	TWI Address Match	Timer2	SPM/ EEPROM Ready	ADC	WDT	USART(4)	Other I/O	
Idle			Yes	Yes	Yes	Yes	Yes	Yes(2)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
ADC Noise Reduction				Yes	Yes	Yes	Yes	Yes(2)	Yes(3)	Yes	Yes(2)	Yes	Yes	Yes	Yes		
Power-down									Yes(3)	Yes				Yes	Yes		Yes
Power-save					Yes	Yes	Yes(5)	Yes(2)	Yes(3)	Yes	Yes			Yes	Yes		Yes
Standby(1)							Yes		Yes(3)	Yes				Yes	Yes		Yes
Extended Standby					Yes(2)	Yes	Yes	Yes(2)	Yes(3)	Yes	Yes			Yes	Yes		Yes

Note:

1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT2, INT1 and INT0, only level interrupt.
4. Start frame detection, only.
5. Main clock is kept running if PTC is enabled.

To enter any of the six sleep modes, the Sleep Enable bit in the Sleep Mode Control Register (SMCR.SE) must be written to '1' and a SLEEP instruction must be executed. Sleep Mode Select bits (SMCR.SM[2:0]) select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby, or Extended Standby) will be activated by the SLEEP instruction.

Note: The block diagram in the section *System Clock and Clock Options* provides an overview over the different clock systems in the device, and their distribution. This figure is helpful in selecting an appropriate sleep mode.

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

Related Links

Clock Systems and Their Distribution

12.2 BOD Disable

When the Brown-out Detector (BOD) is enabled by BODLEVEL fuses (see also section *Fuse Bits*), the BOD is actively monitoring the power supply voltage during a sleep period. To save power, it is possible to disable the BOD by software for some of the sleep modes. The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses. If BOD is disabled in software, the BOD function is turned off immediately after entering the sleep mode. Upon wake-up from sleep, BOD is automatically enabled again. This ensures safe operation in case the V_{CC} level has dropped during the sleep period.

When the BOD has been disabled, the wake-up time from sleep mode will be approximately 60µs to ensure that the BOD is working correctly before the MCU continues executing code.

BOD disable is controlled by the BOD Sleep bit in the MCU Control Register (MCUCR.BODS). Writing this bit to '1' turns off the BOD in relevant sleep modes, while a zero in this bit keeps BOD active. The default setting, BODS=0, keeps BOD active.

Note: Writing to the BODS bit is controlled by a timed sequence and an enable bit.

Related Links

MCUCR

Fuse Bits

12.3 Idle Mode

When the SM[2:0] bits are written to '000', the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the SPI, USART, Analog Comparator, 2-wire Serial Interface, ADC, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

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

Related Links

ACSR

12.4 ADC Noise Reduction Mode

When the SM[2:0] bits are written to '001', the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the external interrupts, the 2-wire Serial Interface address watch, Timer/Counter(1), and the Watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC Conversion Complete interrupt, only these events can wake up the MCU from ADC Noise Reduction mode:

- External Reset
- Watchdog System Reset
- Watchdog Interrupt
- Brown-out Reset
- 2-wire Serial Interface address match
- Timer/Counter interrupt
- SPM/EEPROM ready interrupt
- External level interrupt on INT
- Pin change interrupt

Note: 1. Timer/Counter will only keep running in asynchronous mode.

Related Links

8-bit Timer/Counter2 with PWM and Asynchronous Operation

12.5 Power-Down Mode

When the SM[2:0] bits are written to '010', the SLEEP instruction makes the MCU enter Power-Down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the 2-wire Serial Interface address watch, and the Watchdog continue operating (if enabled).

Only one of these events can wake up the MCU:

- External Reset
- Watchdog System Reset
- Watchdog Interrupt
- Brown-out Reset
- 2-wire Serial Interface address match
- External level interrupt on INT
- Pin change interrupt

This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note: If a level triggered interrupt is used for wake-up from Power-Down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses.

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.

Related Links

Clock Sources

EXINT - External Interrupts

12.6 Power-Save Mode

When the SM[2:0] bits are written to 011, the SLEEP instruction makes the MCU enter Power-save mode. This mode is identical to Power-down, except:

If Timer/Counter2 is enabled, it will keep running 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 TIMSK2, and the Global Interrupt Enable bit in SREG is set.

If the PTC is enabled, the main clock is kept running.

If Timer/Counter2 is not running, Power-down mode is recommended instead of Power-save mode.

The Timer/Counter2 can be clocked both synchronously and asynchronously in Power-save mode. If Timer/Counter2 is not using the asynchronous clock, the Timer/Counter Oscillator is stopped during sleep. If Timer/Counter2 is not using the synchronous clock, the clock source is stopped during sleep. Even if the synchronous clock is running in Power-save, this clock is only available for Timer/Counter2.

12.7 Standby Mode

When the SM[2:0] bits are written to '110' and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-Down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

12.8 Extended Standby Mode

When the SM[2:0] bits are written to '111' and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter 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 six clock cycles.

12.9 Power Reduction Registers

The Power Reduction Registers (PRR2, PRR1 and PRR0) provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the corresponding bit in the PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

12.10 Minimizing Power Consumption

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

12.10.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion.

Related Links

Analog-to-Digital Converter

12.10.2 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode.

Related Links

Analog Comparator

12.10.3 Brown-Out Detector

If the Brown-Out Detector (BOD) is not needed by the application, this module should be turned off. If the BOD is enabled by the BODLEVEL Fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption.

Related Links

System Control and Reset

12.10.4 Internal Voltage Reference

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

Related Links

System Control and Reset

12.10.5 Watchdog Timer

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

Related Links

System Control and Reset

12.10.6 Port Pins

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

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the Digital Input Disable Registers (DIDR0 for ADC, DIDR1 for AC).

Related Links

Digital Input Enable and Sleep Modes

12.10.7 On-chip Debug System

If the On-chip debug system is enabled by the OCDEN Fuse and the chip enters sleep mode, the main clock source is enabled and hence always consumes power. In the deeper sleep modes, this will contribute significantly to the total current consumption.

There are three alternative ways to disable the OCD system:

- Disable the OCDEN Fuse
- Disable the JTAGEN Fuse

- Write one to the JTD bit in MCUCR

12.11 Register Description

12.11.1 Sleep Mode Control Register

The Sleep Mode Control Register contains control bits for power management.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SMCR
Offset: 0x53
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x33

Bit	7	6	5	4	3	2	1	0
					SM[2:0]			SE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:1 – SM[2:0]: Sleep Mode Select
The SM[2:0] bits select between the five available sleep modes.

Table 12-2. Sleep Mode Select

SM[2:0]	Sleep Mode
000	Idle
001	ADC Noise Reduction
010	Power-down
011	Power-save
100	Reserved
101	Reserved
110	Standby(1)
111	Extended Standby(1)

Note:

1. Standby mode is only recommended for use with external crystals or resonators.

Bit 0 – SE: Sleep Enable
The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer’s purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

12.11.2 MCU Control Register

The MCU Control Register controls the placement of the Interrupt Vector table in order to move interrupts between application and boot space.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: MCUCR
Offset: 0x55
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x35

Bit	7	6	5	4	3	2	1	0
	JTD	BODS	BODSE	PUD			IVSEL	IVCE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 7 – JTD
When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

Bit 6 – BODS: BOD Sleep
The BODS bit must be written to '1' in order to turn off BOD during sleep. Writing to the BODS bit is controlled by a timed sequence and the enable bit BODSE. To disable BOD in relevant sleep modes, both BODS and BODSE must first be written to '1'. Then, BODS must be written to '1' and BODSE must be written to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

Bit 5 – BODSE: BOD Sleep Enable
BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

Bit 4 – PUD: Pull-up Disable
When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

Bit 1 – IVSEL: Interrupt Vector Select
When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.

2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

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

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section.

Bit 0 – IVCE: Interrupt Vector Change Enable
The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example

```
Move_interrupts:
; Get MCUCR
in    r16, MCUCR
mov   r17, r16
; Enable change of Interrupt Vectors
ori   r16, (1<<IVCE)
out   MCUCR, r16
; Move interrupts to Boot Flash section
ori   r17, (1<<IVSEL)
out   MCUCR, r17
ret
```

C Code Example

```
void Move_interrupts(void)
{
uchar temp;
/* GET MCUCR*/
temp = MCUCR;
/* Enable change of Interrupt Vectors */
MCUCR = temp|(1<<IVCE);
/* Move interrupts to Boot Flash section */
MCUCR = temp|(1<<IVSEL);
}
```

12.11.3 Power Reduction Register 0

Name: PRR0
Offset: 0x64
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PRTWI0	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI0	PRUSART0	PRADC
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – PRTWI0: Power Reduction TWI0
Writing a logic one to this bit shuts down the TWI 0 by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

Bit 6 – PRTIM2: Power Reduction Timer/Counter2
Writing a logic one to this bit shuts down the Timer/Counter2 module in synchronous mode (AS2 is 0). When the Timer/Counter2 is enabled, operation will continue like before the shutdown.

Bit 5 – PRTIM0: Power Reduction Timer/Counter0
Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

Bit 4 – PRUSART1: Power Reduction USART1
Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re initialized to ensure proper operation.

Bit 3 – PRTIM1: Power Reduction Timer/Counter1
Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

Bit 2 – PRSPI0: Power Reduction Serial Peripheral Interface 0
If using debugWIRE On-chip Debug System, this bit should not be written to one. Writing a logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be re initialized to ensure proper operation.

Bit 1 – PRUSART0: Power Reduction USART0
Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re initialized to ensure proper operation.

Bit 0 – PRADC: Power Reduction ADC
Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

12.11.4 Power Reduction Register 1

Name: PRR1
Offset: 0x65
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							PRTIM4	PRTIM3
Access							R/W	R/W
Reset							0	0

Bit 1 – PRTIM4: Power Reduction Timer/Counter4
Writing a logic one to this bit shuts down the Timer/Counter4 module. When the Timer/Counter4 is enabled, operation will continue like before the shutdown.

Bit 0 – PRTIM3: Power Reduction Timer/Counter3
Writing a logic one to this bit shuts down the Timer/Counter3 module. When the Timer/Counter3 is enabled, operation will continue like before the shutdown.

12.11.5 Power Reduction Register 2

Name: PRR2
Offset: 0x63
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					PRPTC	PRUSART2	PRSPI1	PRTWI1
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – PRPTC: Power Reduction PTC
Writing a logic one to this bit shuts down the PTC module. When the PTC is enabled, operation will continue like before the shutdown.

Bit 2 – PRUSART2: Power Reduction USART2
Writing a logic one to this bit shuts down the USART2 by stopping the clock to the module. When waking up the USART2 again, the USART2 should be re initialized to ensure proper operation.

Bit 1 – PRSPI1: Power Reduction Serial Peripheral Interface
Writing a logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be re initialized to ensure proper operation.

Bit 0 – PRTWI1: Power Reduction TWI1
Writing a logic one to this bit shuts down the TWI1 by stopping the clock to the module. When waking up the TWI1 again, the TWI1 should be re initialized to ensure proper operation.

13. SCRST - System Control and Reset

13.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be an Absolute Jump instruction (JMP) to the reset handling routine for . 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 the next section shows the reset logic.

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

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL Fuses. The different selections for the delay period are presented in the *System Clock and Clock Options* chapter.

Related Links

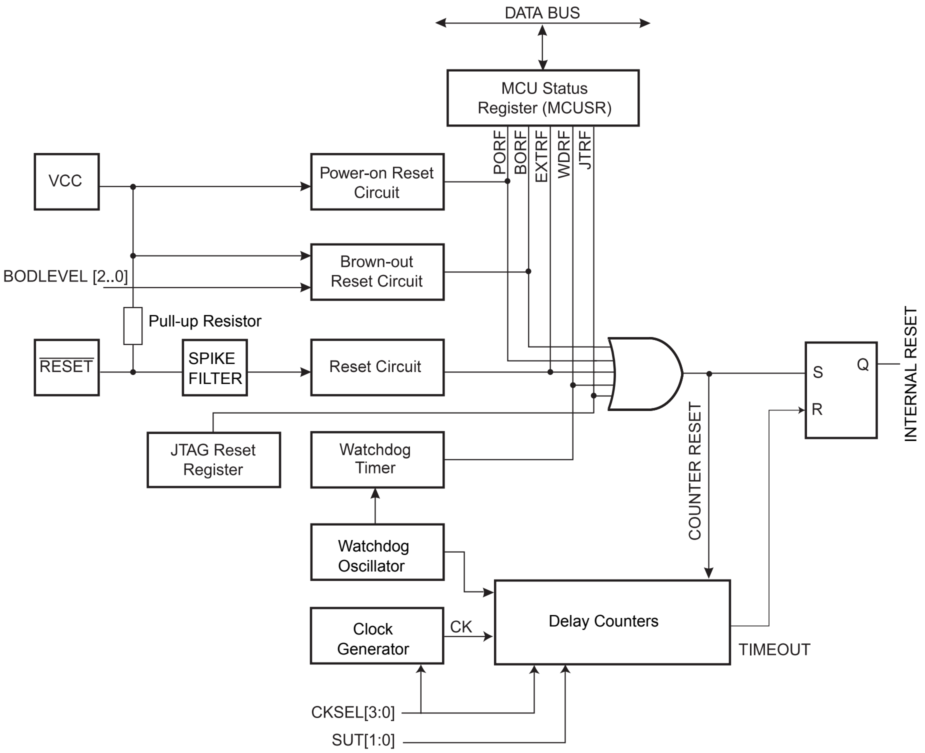
System Clock and Clock Options

13.2 Reset Sources

The device has the following sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is less than the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog System Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog System Reset mode is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is less than the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.
- 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. Refer to the section *IEEE 1149.1 (JTAG) Boundary-scan* for details.

Figure 13-1. Reset Logic



Related Links

IEEE 1149.1 (JTAG) Boundary-scan

13.3 Power-on Reset

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

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

Figure 13-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

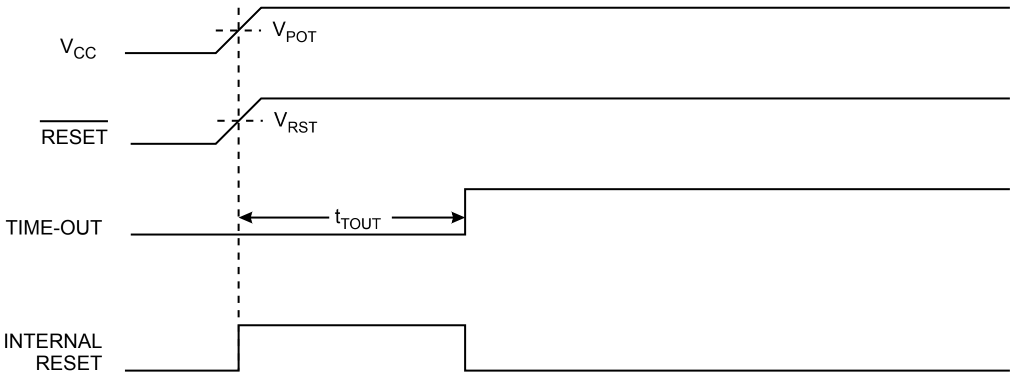
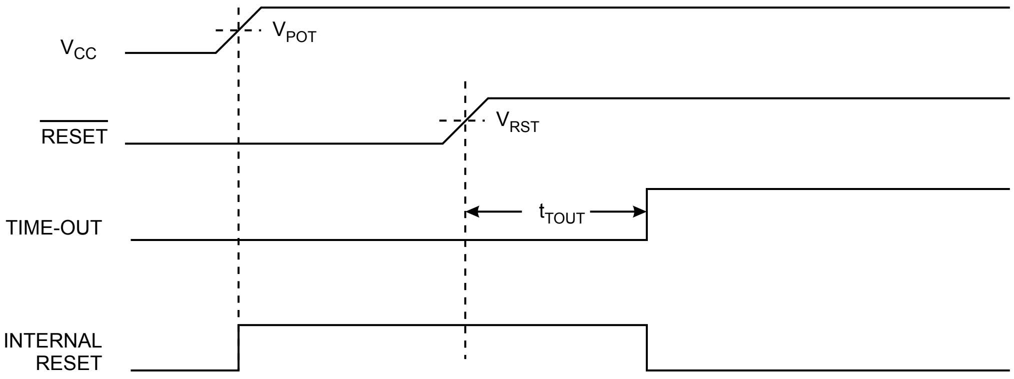


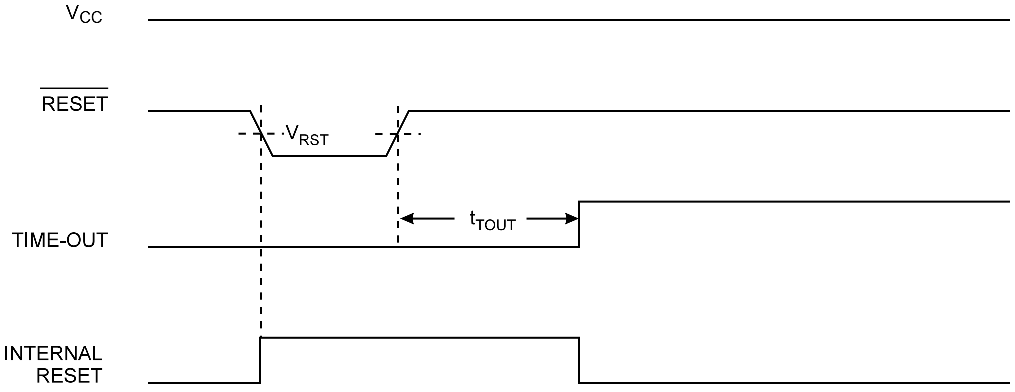
Figure 13-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



13.4 External Reset

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

Figure 13-4. External Reset During Operation

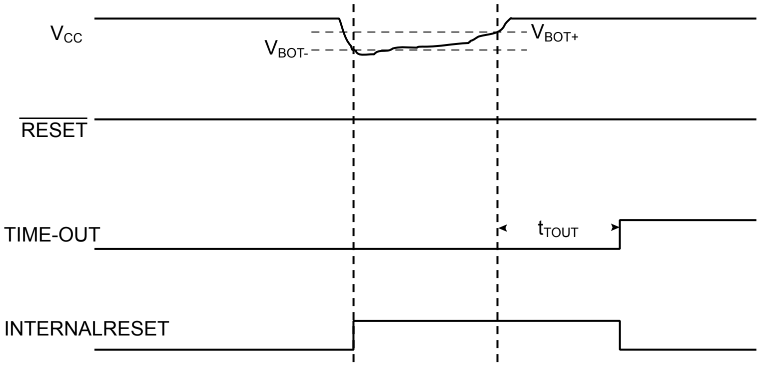


13.5 Brown-out Detection

The device has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$. When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in the following figure), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in the following figure), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

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

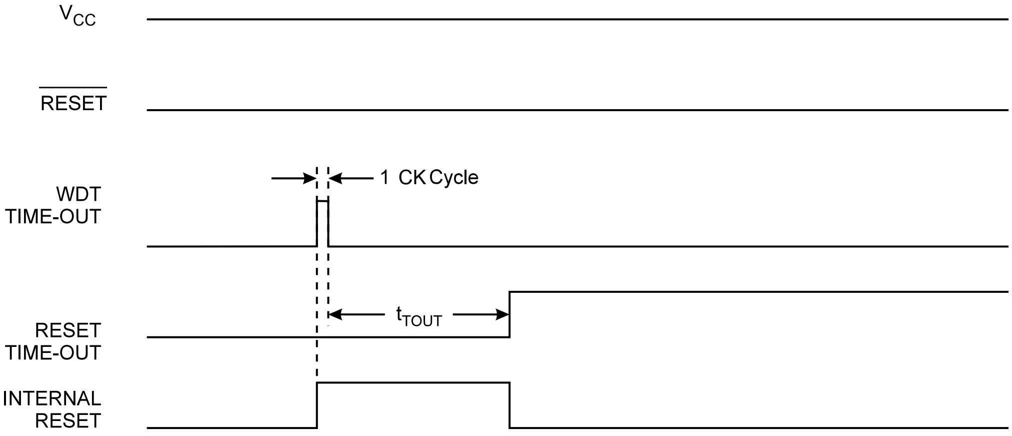
Figure 13-5. Brown-out Reset During Operation



13.6 Watchdog System Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}.

Figure 13-6. Watchdog System Reset During Operation



13.7 Internal Voltage Reference

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

13.7.1 Voltage Reference Enable Signals and Start-up Time

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

1. When the BOD is enabled (by programming the BODLEVEL [2:0] Fuses).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR (ACSR.ACBG)).
3. When the ADC is enabled.

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

13.8 Watchdog Timer

If the watchdog timer is not needed in the application, the module should be turned off. If the watchdog timer is enabled, it will be enabled in all sleep modes and hence always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption.

Refer to Watchdog System Reset for details on how to configure the watchdog timer.

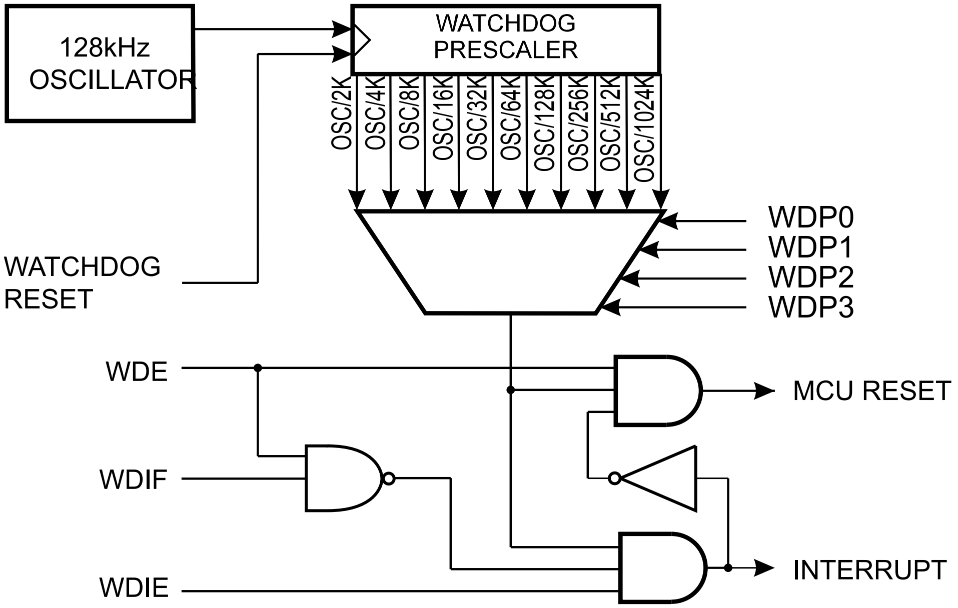
13.8.1 Features

- Clocked from separate On-chip Oscillator
- Three operating modes:
  - Interrupt
  - System Reset
  - Interrupt and System Reset
- Selectable Time-out period from 16ms to 8s
- Possible Hardware fuse Watchdog always on (WDTON) for fail-safe mode

13.8.2 Overview

The device has an Enhanced Watchdog Timer (WDT). The WDT is a timer counting cycles of a separate on-chip 128kHz oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the Watchdog Timer Reset (WDR) instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

Figure 13-7. Watchdog Timer



In Interrupt mode, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected. In System Reset mode, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code. The third mode, Interrupt and System Reset mode, combines the other two modes by first giving an interrupt and then switch to System Reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.

The Watchdog always on (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed the System Reset mode bit (WDE) and Interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

1. In the same operation, write a logic one to the Watchdog change enable bit (WDCE) and Watchdog System Reset Enable (WDE) in Watchdog Timer Control Register (WDTCSR.WDCE and WDTCSR.WDE). A logic one must be written to WDTCSR.WDE regardless of the previous value of the WDTCSR.WDE.
2. Within the next four clock cycles, write the WDTCSR.WDE and Watchdog prescaler bits group (WDTCSR.WDP) as desired, but with the WDTCSR.WDCE cleared. This must be done in one operation.

The following examples show a function for turning off the Watchdog Timer. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example

```
WDT_off:
   ; Turn off global interrupt
   cli
   ; Reset Watchdog Timer
   wdr
   ; Clear WDRF in MCUSR
   in      r16, MCUSR
   andi    r16, (0xff & (0<<WDRF))
   out     MCUSR, r16
   ; Write '1' to WDCE and WDE
   ; Keep old prescaler setting to prevent unintentional time-out
   lds     r16, WDTCSR
   ori     r16, (1<<WDCE) | (1<<WDE)
   sts     WDTCSR, r16
   ; Turn off WDT
   ldi     r16, (0<<WDE)
   sts     WDTCSR, r16
   ; Turn on global interrupt
   sei
   ret
```

C Code Example

```
void WDT_off(void)
{
    __disable_interrupt();
    __watchdog_reset();
    /* Clear WDRF in MCUSR */
    MCUSR &= ~(1<<WDRF);
    /* Write logical one to WDCE and WDE */
    /* Keep old prescaler setting to prevent unintentional time-out */
    WDTCSR |= (1<<WDCE) | (1<<WDE);
    /* Turn off WDT */
    WDTCSR = 0x00;
    __enable_interrupt();
}
```

Note: If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the Watchdog Timer will stay enabled. If the code is not set up to handle the Watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the Watchdog System Reset Flag (WDRF) and the WDE control bit in the initialization routine, even if the Watchdog is not in use.

The following code examples shows how to change the time-out value of the Watchdog Timer.

Assembly Code Example

```
WDT_Prescaler_Change:
   ; Turn off global interrupt
   cli
   ; Reset Watchdog Timer
   wdr
   ; Start timed sequence
   lds r16, WDTCSR
   ori r16, (1<<WDCE) | (1<<WDE)
   sts WDTCSR, r16
   ; -- Got four cycles to set the new values from here -
   ; Set new prescaler(time-out) value = 64K cycles (~0.5 s)
   ldi r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
   sts WDTCSR, r16
   ; -- Finished setting new values, used 2 cycles -
   ; Turn on global interrupt
   sei
   ret
```

C Code Example

```
void WDT_Prescaler_Change(void)
{
    __disable_interrupt();
    __watchdog_reset();
    /* Start timed sequence */
    WDTCSR |= (1<<WDCE) | (1<<WDE);
    /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
    WDTCSR = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
    __enable_interrupt();
}
```

Note: The Watchdog Timer should be reset before any change of the WDTCSR.WDP bits, since a change in the WDTCSR.WDP bits can result in a time-out when switching to a shorter time-out period.

13.9 Register Description

13.9.1 MCU Status Register

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.
When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: MCUSR
Offset: 0x54 [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x34

Bit	7	6	5	4	3	2	1	0
				JTRF	WDRF	BORF	EXTRF	PORF
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

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 System Reset Flag
This bit is set if a Watchdog System Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

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 '0' to it.

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 '0' to it.

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 '0' to it.

13.9.2 WDTCSR – Watchdog Timer Control Register

Name: WDTCSR
Offset: 0x60 [ID-000004d0]
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	WDIF	WDIE	WDP[3]	WDCE	WDE	WDP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – WDIF: Watchdog Interrupt Flag
This bit is set when a timeout occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a '1' to it. When the I-bit in SREG and WDIE are set, the Watchdog Timeout Interrupt is executed.

Bit 6 – WDIE: Watchdog Interrupt Enable
When this bit is written to '1' and the I-bit in the Status Register is set, the Watchdog Interrupt is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is executed if timeout in the Watchdog Timer occurs. If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first timeout in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode).

This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety function of the Watchdog System Reset mode. If the interrupt is not executed before the next timeout, a System Reset will be applied.

Table 13-1. Watchdog Timer Configuration

WDTON[1]	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	x	x	System Reset Mode	Reset

Note: 1. WDTON Fuse set to '0' means programmed and '1' means unprogrammed.

Bit 5 – WDP[3]: Watchdog Timer Prescaler 3

Bit 4 – WDCE: Watchdog Change Enable

This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set. Once written to '1', hardware will clear WDCE after four clock cycles. Refer to Overview for how to use WDCE.

Bit 3 – WDE: Watchdog System Reset Enable

WDE is overridden by WDRF in MCUSR. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared first. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

Bits 2:0 – WDP[2:0]: Watchdog Timer Prescaler 2, 1, and 0

The WDP[3:0] bits determine the Watchdog Timer prescaling when the Watchdog Timer is running. The different prescaling values and their corresponding timeout periods are shown in the following table.

Table 13-2. Watchdog Timer Prescale Select

WDP[3]	WDP[2]	WDP[1]	WDP[0]	Number of WDT Oscillator (Cycles)	Oscillator
0	0	0	0	2K (2048)	16ms
0	0	0	1	4K (4096)	32ms
0	0	1	0	8K (8192)	64ms
0	0	1	1	16K (16384)	0.125s
0	1	0	0	32K (32768)	0.25s
0	1	0	1	64K (65536)	0.5s
0	1	1	0	128K (131072)	1.0s
0	1	1	1	256K (262144)	2.0s
1	0	0	0	512K (524288)	4.0s
1	0	0	1	1024K (1048576)	8.0s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		

WDP[3]	WDP[2]	WDP[1]	WDP[0]	Number of WDT Oscillator (Cycles)	Oscillator
1	1	0	1		
1	1	1	0		
1	1	1	1		

14. INT- Interrupts

This section describes the specifics of the interrupt handling of the device. For a general explanation of the AVR interrupt handling, refer to the description of *Reset and Interrupt Handling*.

In general:

- Each Interrupt Vector occupies .
- The Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR.

Related Links

Reset and Interrupt Handling

14.1 Interrupt Vectors in ATmega324PB

Table 14-1. Reset and Interrupt Vectors in ATmega324PB

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	INT2	External Interrupt Request 2
5	0x0008	PCINT0	Pin Change Interrupt Request 0
6	0x000A	PCINT1	Pin Change Interrupt Request 1
7	0x000C	PCINT2	Pin Change Interrupt Request 2
8	0x000E	PCINT3	Pin Change Interrupt Request 3
9	0x0010	WDT	Watchdog Time-out Interrupt
10	0x0012	TIMER2_COMPA	Timer/Counter2 Compare Match A
11	0x0014	TIMER2_COMPB	Timer/Coutner2 Compare Match B
12	0x0016	TIMER2_OVF	Timer/Counter2 Overflow
13	0x0018	TIMER1_CAPT	Timer/Counter1 Capture Event
14	0x001A	TIMER1_COMPA	Timer/Counter1 Compare Match A
15	0x001C	TIMER1_COMPB	Timer/Coutner1 Compare Match B
16	0x001E	TIMER1_OVF	Timer/Counter1 Overflow
17	0x0020	TIMER0_COMPA	Timer/Counter0 Compare Match A
18	0x0022	TIMER0_COMPB	Timer/Coutner0 Compare Match B
19	0x0024	TIMER0_OVF	Timer/Counter0 Overflow
20	0x0026	SPI0_STC	SPI0 Serial Transfer Complete
21	0x0028	USART0_RX	USART0 Rx Complete
22	0x002A	USART0_UDRE	USART0, Data Register Empty

Vector No	Program Address[2]	Source	Interrupts definition
23	0x002C	USART0_TX	USART0, Tx Complete
24	0x002E	ANALOG_COMP	Analog Comparator
25	0x0030	ADC	ADC Conversion Complete
26	0x0032	EE_READY	EEPROM Ready
27	0x0034	TWI0	TWI0 Transfer complete
28	0x0036	SPM_READY	Store Program Memory Ready
29	0x0038	USART1_RX	USART1 Rx Complete
30	0x003A	USART1_UDRE	USART1, Data Register Empty
31	0x003C	USART1_TX	USART0, Tx Complete
32	0x003E	TIMER3_CAPT	Timer/Counter3 Capture Event
33	0x0040	TIMER3_COMPA	Timer/Counter3 Compare Match A
34	0x0042	TIMER3_COMPB	Timer/Coutner3 Compare Match B
35	0x0044	TIMER3_OVF	Timer/Counter3 Overflow
36	0x0046	USART0_START	USART0 Start Frame detection
37	0x0048	USART1_START	USART1 Start Frame detection
38	0x004A	PCINT4	Pin Change Interrupt 4
39	0x004C	CFD	Crystal Failure Detection
40	0x004E	PTC_EOC	PTC End of Conversion
41	0x0050	PTC_WCOMP	PTC Window comparator mode
42	0x0052	SPI1_STC	SPI1 Serial Transfer Complete
43	0x0054	TWI1	TWI1 Transfer complete
44	0x0056	TIMER4_CAPT	Timer/Counter4 Capture Event
45	0x0058	TIMER4_COMPA	Timer/Counter4 Compare Match A
46	0x005A	TIMER4_COMPB	Timer/Coutner4 Compare Match B
47	0x005C	TIMER4_OVF	Timer/Counter4 Overflow
48	0x005E	USART2_RX	USART2 Rx Complete
49	0x0060	USART2_UDRE	USART2, Data Register Empty
50	0x0062	USART2_TX	USART2, Tx Complete
51	0x0064	USART2_START	USART2 Start Frame detection

Note:

1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see *Memory programming*
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

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

Table 14-2. Reset and Interrupt Vectors placement

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x0000	0x0002
1	1	0x0000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x0002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

Note: The Boot Reset Address is shown in Table *Boot size configuration* in Boot Loader Parameters. For the BOOTRST Fuse “1” means unprogrammed while “0” means programmed.

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 8K bytes and the IVSEL bit in the MCUCR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address    Labels     Code                    Comments
0x00000    RESET:     ldi r16,high(RAMEND)    ; Main program start
0x00001               out SPH,r16             ; Set Stack Pointer to top of RAM
0x00002               ldi r16,low(RAMEND)
0x00003               out SPL,r16
0x00004               sei                     ; Enable interrupts
0x00005               <instr>            xxx
;
.org 0x1F002
0x1F002               jmp EXT_INT0            ; IRQ0 Handler
0x1F004               jmp EXT_INT1            ; IRQ1 Handler
...        ...         ...                    ;
0x1FO36               jmp SPM_RDY             ; SPM Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 8K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address    Labels     Code                     Comments
.org 0x0002
0x00002               jmp EXT_INT0            ; IRQ0 Handler
0x00004               jmp EXT_INT1            ; IRQ1 Handler
...          ...      ...                     ;
0x00036               jmp SPM_RDY             ; SPM Ready Handler
;
.org       0x1F000
0x1F000    RESET:     ldi r16,high(RAMEND)    ; Main program start
0x1F001               out SPH,r16             ; Set Stack Pointer to top of RAM
0x1F002               ldi r16,low(RAMEND)
0x1F003               out SPL,r16
0x1F004               sei                     ; Enable interrupts
0x1F005               <instr> xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 8K bytes and the IVSEL bit in the MCUCR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address    Labels      Code                     Comments
;
.org 0x1F000
0x1F000                jmp RESET               ; Reset handler
0x1F002                jmp EXT_INT0            ; IRQ0 Handler
```

```
0x1F004                jmp EXT_INT1            ; IRQ1 Handler
...        ...         ...                     ;
0x1F036                jmp SPM_RDY             ; SPM Ready Handler
;
0x1F03E    RESET:      ldi r16,high(RAMEND)    ; Main program start
0x1F03F                out SPH,r16             ; Set Stack Pointer to top of RAM
0x1F040                ldi r16,low(RAMEND)
0x1F041                out SPL,r16
0x1F042                sei                     ; Enable interrupts
0x1FO43                <instr> xxx
```

14.2 Register Description

14.2.1 Moving Interrupts Between Application and Boot Space

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

14.2.2 MCU Control Register

The MCU Control Register controls the placement of the Interrupt Vector table in order to move interrupts between application and boot space.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: MCUCR
Offset: 0x55
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x35

Bit	7	6	5	4	3	2	1	0
	JTD	BODS	BODSE	PUD			IVSEL	IVCE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 7 – JTD
When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

Bit 6 – BODS: BOD Sleep
The BODS bit must be written to '1' in order to turn off BOD during sleep. Writing to the BODS bit is controlled by a timed sequence and the enable bit BODSE. To disable BOD in relevant sleep modes, both BODS and BODSE must first be written to '1'. Then, BODS must be written to '1' and BODSE must be written to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

Bit 5 – BODSE: BOD Sleep Enable

BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

Bit 4 – PUD: Pull-up Disable

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

Bit 1 – IVSEL: Interrupt Vector Select

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

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

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

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section.

Bit 0 – IVCE: Interrupt Vector Change Enable

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

Assembly Code Example

```
Move_interrupts:
; Get MCUCR
in    r16, MCUCR
mov   r17, r16
; Enable change of Interrupt Vectors
ori   r16, (1<<IVCE)
out   MCUCR, r16
; Move interrupts to Boot Flash section
ori   r17, (1<<IVSEL)
out   MCUCR, r17
ret
```

C Code Example

```
void Move_interrupts(void)
{
uchar temp;
/* GET MCUCR*/
temp = MCUCR;
/* Enable change of Interrupt Vectors */
MCUCR = temp|(1<<IVCE);
/* Move interrupts to Boot Flash section */
MCUCR = temp|(1<<IVSEL);
}
```

15. External Interrupts

15.1 EXINT - External Interrupts

The External Interrupts are triggered by the INT pin or any of the PCINT pins. Observe that, if enabled, the interrupts will trigger even if the INT or PCINT pins are configured as outputs. This feature provides a way of generating a software interrupt.

The Pin Change Interrupt Request 4 (PCI4) will trigger if any enabled PCINT[38:32] pin toggles. The Pin Change Interrupt Request 3 (PCI3) will trigger if any enabled PCINT[31:24] pin toggles. The Pin Change Interrupt Request 2 (PCI2) will trigger if any enabled PCINT[23:16] pin toggles. The Pin Change Interrupt Request 1 (PCI1) will trigger if any enabled PCINT[15:8] pin toggles. The Pin Change Interrupt Request 0 (PCI0) will trigger if any enabled PCINT[7:0] pin toggles. The PCMSK4, PCMSK3, PCMSK2, PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A (EICRA). When the external interrupts are enabled and are configured as level triggered, the interrupts will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT requires the presence of an I/O clock. Low level interrupt on INT is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note: Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses.

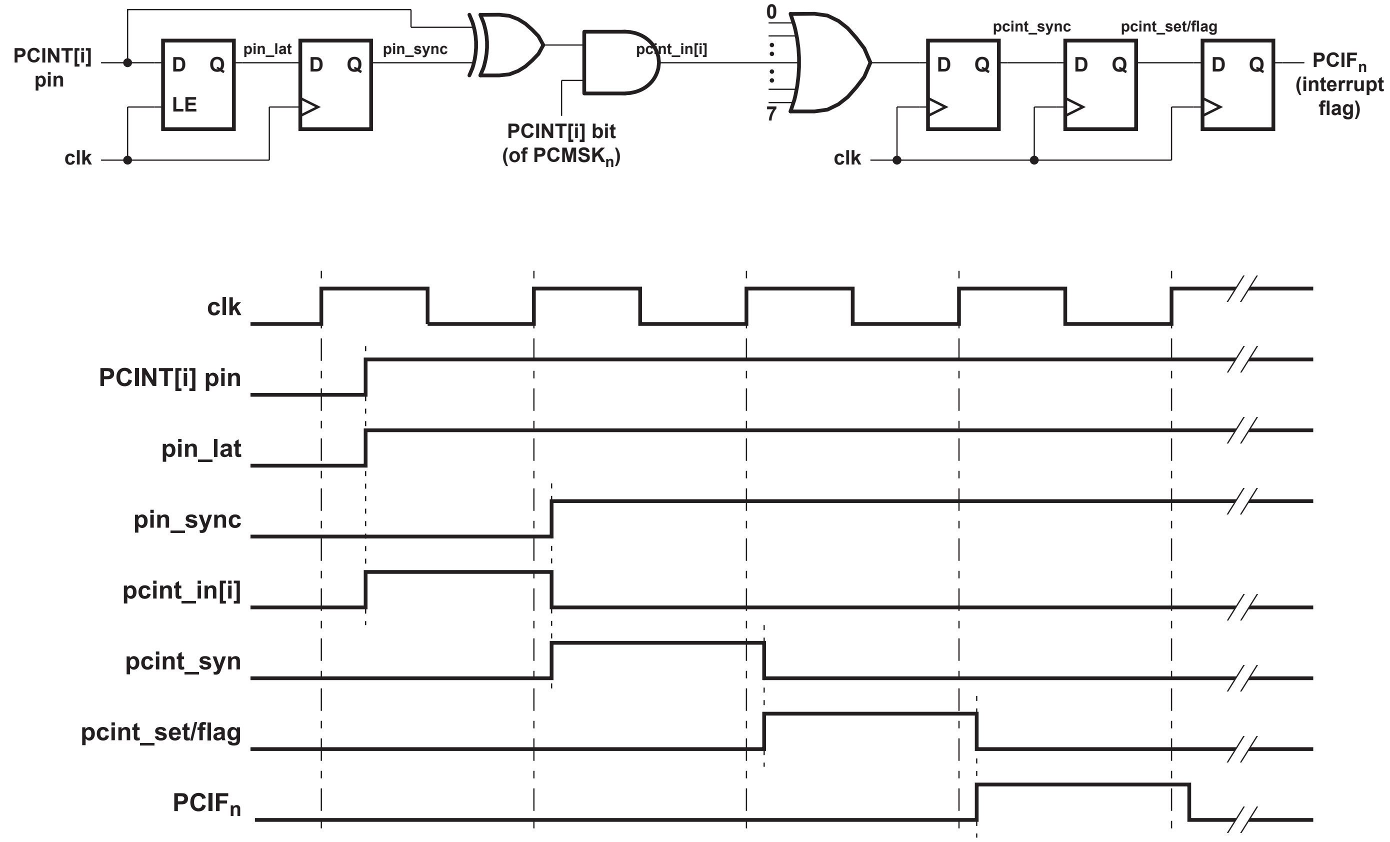
Related Links

System Clock and Clock Options

15.1.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in the following figure.

Figure 15-1. Timing of pin change interrupts



15.1.2 Register Description

15.1.2.1 External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Name: EICRA
Offset: 0x69
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			ISC2 [1:0]		ISC1 [1:0]		ISC0 [1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:4 – ISC2 [1:0]: Interrupt Sense Control 2
The External Interrupt 2 is activated by the external pin INT2 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT2 pin that activate the interrupt are defined in table below. The value on the INT2 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.

Value	Description
00	The low level of INT2 generates an interrupt request.
01	Any logical change on INT2 generates an interrupt request.
10	The falling edge of INT2 generates an interrupt request.
11	The rising edge of INT2 generates an interrupt request.

Bits 3:2 – ISC1 [1:0]: Interrupt Sense Control 1
The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in the table below. 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.

Value	Description
00	The low level of INT1 generates an interrupt request.
01	Any logical change on INT1 generates an interrupt request.
10	The falling edge of INT1 generates an interrupt request.
11	The rising edge of INT1 generates an interrupt request.

Bits 1:0 – ISC0 [1:0]: Interrupt Sense Control 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 below. 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.

Value	Description
00	The low level of INT0 generates an interrupt request.
01	Any logical change on INT0 generates an interrupt request.
10	The falling edge of INT0 generates an interrupt request.
11	The rising edge of INT0 generates an interrupt request.

15.1.2.2 External Interrupt Mask Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EIMSK
Offset: 0x3D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1D

Bit	7	6	5	4	3	2	1	0
						INT2	INT1	INT0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – INT2: External Interrupt Request 2 Enable
When the INT2 bit is set and the I-bit in the Status Register (SREG) is set, the external pin interrupt is enabled. The Interrupt Sense Control2 bits 1/0 (ISC21 and ISC20) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT2 pin or level sensed. 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.

Bit 1 – INT1: External Interrupt Request 1 Enable
When the INT1 bit is set and the I-bit in the Status Register (SREG) is set, the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

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

15.1.2.3 External Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EIFR
Offset: 0x3C
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1C

Bit	7	6	5	4	3	2	1	0
						INTF2	INTF1	INTF0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – INTF2: External Interrupt Flag 2
When an edge or logic change on the INT2 pin triggers an interrupt request, INTF2 will be set. If the I-bit in SREG and the INT2 bit in EIMSK are set, 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 '1' to it. This flag is always cleared when INT2 is configured as a level interrupt.

Bit 1 – INTF1: External Interrupt Flag 1
When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 will be set. If the I-bit in SREG and the INT1 bit in EIMSK are set, 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 '1' to it. This flag is always cleared when INT1 is configured as a level interrupt.

Bit 0 – INTF0: External Interrupt Flag 0
When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 will be set. If the I-bit in SREG and the INT0 bit in EIMSK are set, 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 '1' to it. This flag is always cleared when INT0 is configured as a level interrupt.

15.1.2.4 Pin Change Interrupt Control Register

Name: PCICR
Offset: 0x68
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				PCIE4	PCIE3	PCIE2	PCIE1	PCIE0
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – PCIE4: Pin Change Interrupt Enable 4
When the PCIE4 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 4 is enabled. Any change on any enabled PCINT[39:32] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI4 Interrupt Vector. PCINT[39:32] pins are enabled individually by the PCMSK4 Register.

Bit 3 – PCIE3: Pin Change Interrupt Enable 3
When the PCIE3 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 3 is enabled. Any change on any enabled PCINT[31:24] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI3 Interrupt Vector. PCINT[31:24] pins are enabled individually by the PCMSK3 Register.

Bit 2 – PCIE2: Pin Change Interrupt Enable 2
When the PCIE2 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 2 is enabled. Any change on any enabled PCINT[23:16] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI2 Interrupt Vector. PCINT[23:16] pins are enabled individually by the PCMSK2 Register.

Bit 1 – PCIE1: Pin Change Interrupt Enable 1
When the PCIE1 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 1 is enabled. Any change on any enabled PCINT[14:8] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT[14:8] pins are enabled individually by the PCMSK1 Register.

Bit 0 – PCIE0: Pin Change Interrupt Enable 0
When the PCIE0 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 0 is enabled. Any change on any enabled PCINT[7:0] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT[7:0] pins are enabled individually by the PCMSK0 Register.

15.1.2.5 Pin Change Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PCIFR
Offset: 0x3B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1B

Bit	7	6	5	4	3	2	1	0
				PCIF4	PCIF3	PCIF2	PCIF1	PCIF0
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – PCIF4: Pin Change Interrupt Flag 4
When a logic change on any PCINT[39:32] pin triggers an interrupt request, PCIF4 will be set. If the I-bit in SREG and the PCIE4 bit in PCICR are set, 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 '1' to it.

Bit 3 – PCIF3: Pin Change Interrupt Flag 3
When a logic change on any PCINT[31:24] pin triggers an interrupt request, PCIF3 will be set. If the I-bit in SREG and the PCIE3 bit in PCICR are set, 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 '1' to it.

Bit 2 – PCIF2: Pin Change Interrupt Flag 2
When a logic change on any PCINT[23:16] pin triggers an interrupt request, PCIF2 will be set. If the I-bit in SREG and the PCIE2 bit in PCICR are set, 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 '1' to it.

Bit 1 – PCIF1: Pin Change Interrupt Flag 1
When a logic change on any PCINT[15:8] pin triggers an interrupt request, PCIF1 will be set. If the I-bit in SREG and the PCIE1 bit in PCICR are set, 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 '1' to it.

Bit 0 – PCIF0: Pin Change Interrupt Flag 0
When a logic change on any PCINT[7:0] pin triggers an interrupt request, PCIF0 will be set. If the I-bit in SREG and the PCIE0 bit in PCICR are set, 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 '1' to it.

15.1.2.6 Pin Change Mask Register 0

Name: PCMSK0
Offset: 0x6B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT 7	PCINT 6	PCINT 5	PCINT 4	PCINT 3	PCINT 2	PCINT 1	PCINT 0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT : Pin Change Enable Mask
Each PCINT[7:0] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

15.1.2.7 Pin Change Mask Register 1

Name: PCMSK1
Offset: 0x6C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT8, PCINT9, PCINT10, PCINT11, PCINT12, PCINT13, PCINT14, PCINT15: Pin Change Enable Mask
Each PCINT[15:8]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[15:8] is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[15:8] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

15.1.2.8 Pin Change Mask Register 2

Name: PCMSK2
Offset: 0x6D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT16, PCINT17, PCINT18, PCINT19, PCINT20, PCINT21, PCINT22, PCINT23: Pin Change Enable Mask
Each PCINT[23:16]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[23:16] is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[23:16] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

15.1.2.9 Pin Change Mask Register 3

Name: PCMSK3
Offset: 0x73
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT24, PCINT25, PCINT26, PCINT27, PCINT28, PCINT29, PCINT30, PCINT31: Pin Change Enable Mask
Each PCINT[31:24]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[31:24] is set and the PCIE3 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[31:24] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

15.1.2.10 Pin Change Mask Register 4

Name: PCMSK4
Offset: 0x74
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		PCINT38	PCINT37	PCINT36	PCINT35	PCINT34	PCINT33	PCINT32
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

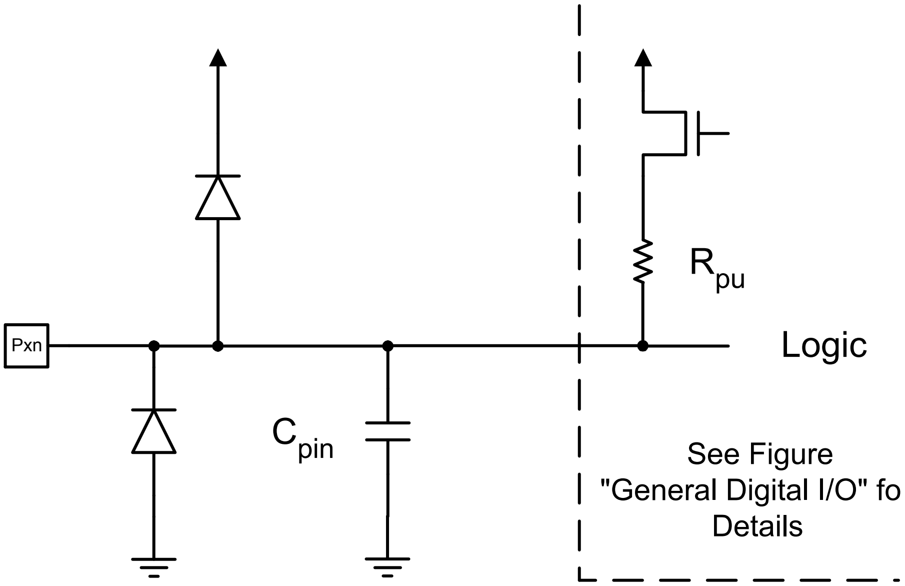
Bits 0, 1, 2, 3, 4, 5, 6 – PCINT32, PCINT33, PCINT34, PCINT35, PCINT36, PCINT37, PCINT38: Pin Change Enable Mask
Each PCINT[38:32]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[38:32] is set and the PCIE4 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[38:32] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

16. I/O-Ports

16.1 Overview

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

Figure 16-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn.

Four I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, Pull-up Enable Register – PUEx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing '1' to a bit in the PINx Register will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

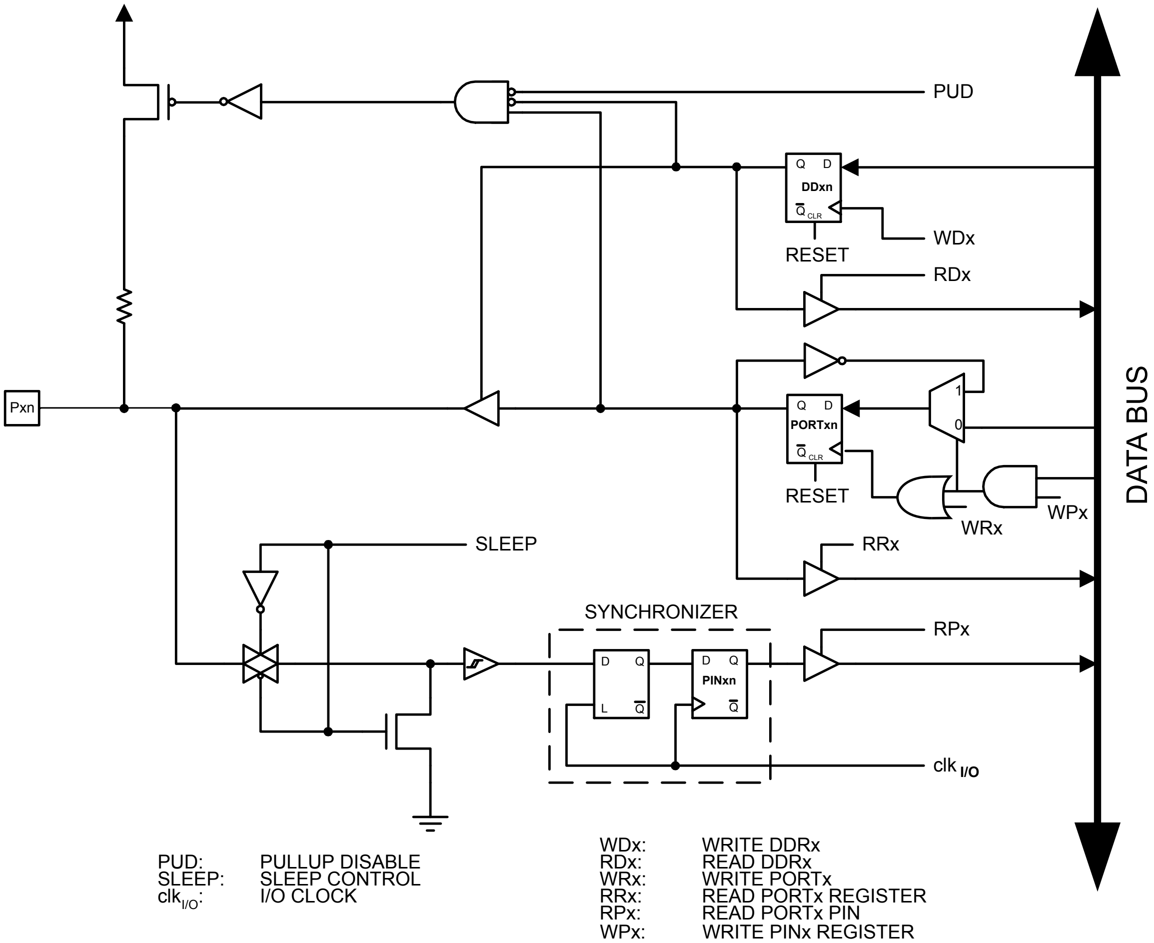
Using the I/O port as General Digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer to the individual module sections for a full description of the alternate functions.

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

16.2 Ports as General Digital I/O

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

Figure 16-2. General Digital I/O(1)



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

16.2.1 Configuring the Pin

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

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written to '1', Pxn is configured as an output pin. If DDxn is written to '0', Pxn is configured as an input pin.

If PORTxn is written to '1' when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written to '0' or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written to '1' when the pin is configured as an output pin, the port pin is driven high. If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low.

16.2.2 Toggling the Pin

Writing a '1' to PINxn toggles the value of PORTxn, independent on the value of DDRxn. The SBI instruction can be used to toggle one single bit in a port.

16.2.3 Switching Between Input and Output

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

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

The following table summarizes the control signals for the pin value.

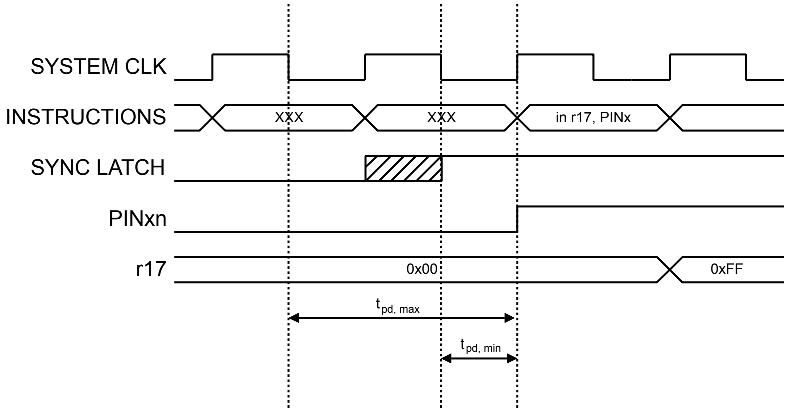
Table 16-1. Port Pin Configurations

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

16.2.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Ports as General Digital I/O, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. The following figure shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

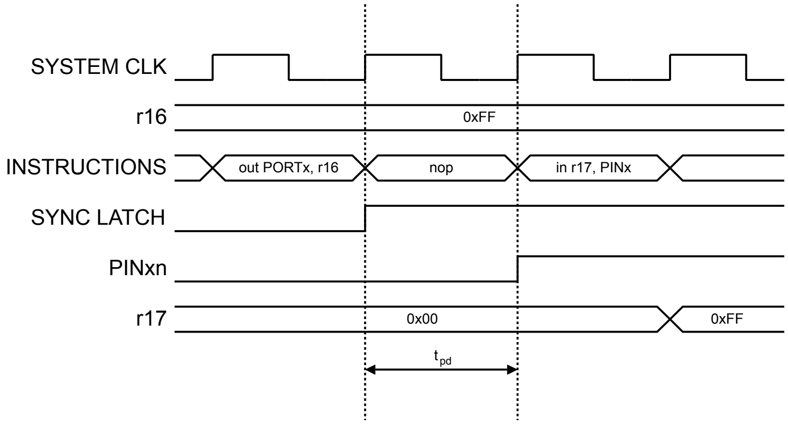
Figure 16-3. Synchronization when Reading an Externally Applied Pin value



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

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in the following figure. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 16-4. Synchronization when Reading a Software Assigned Pin Value



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

Assembly Code Example(1)

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

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

C Code Example

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

```
i = PINB;
...
```

16.2.5 Digital Input Enable and Sleep Modes

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

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in *Alternate Port Functions* section in this chapter.

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

16.2.6 Unconnected Pins

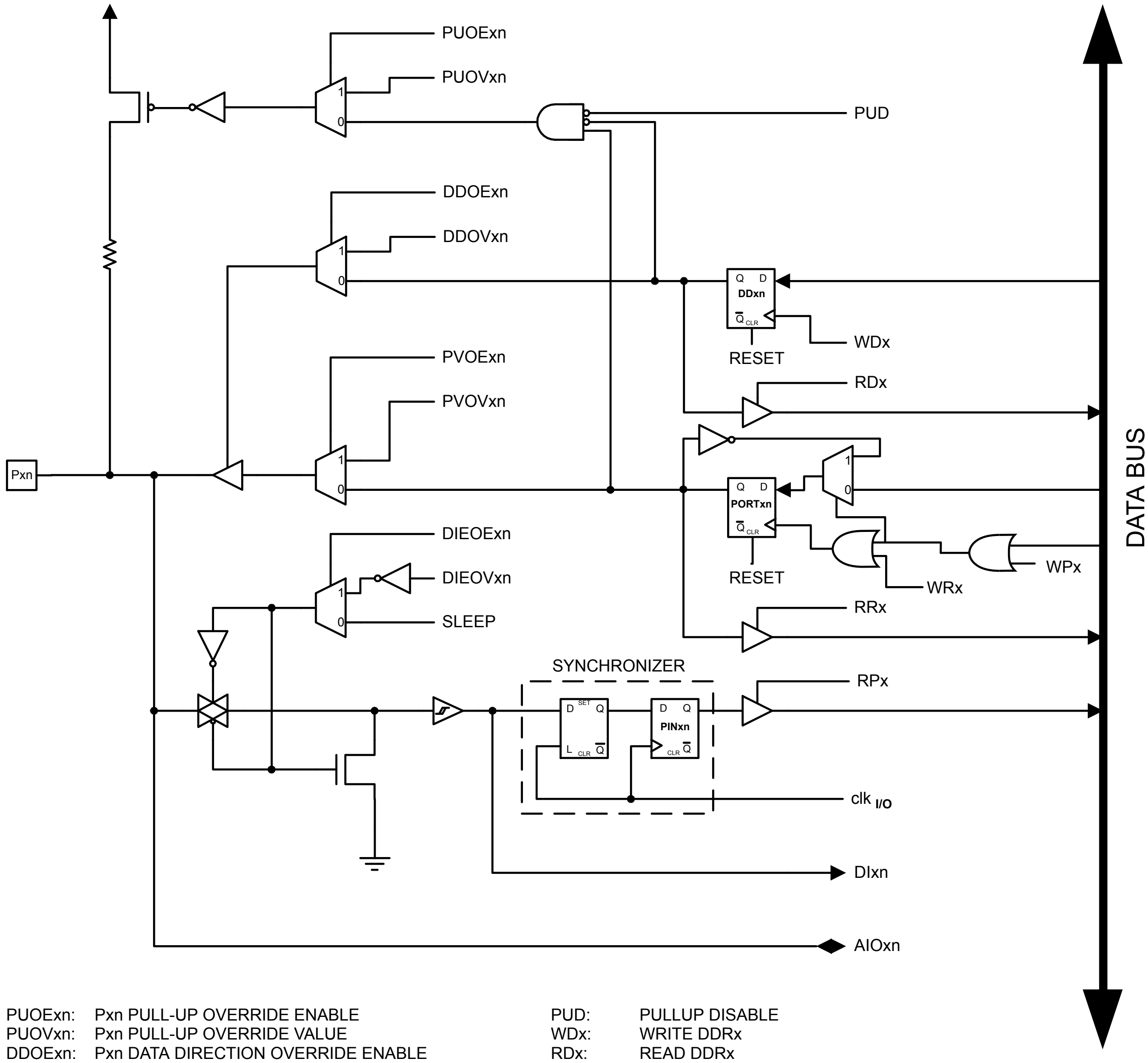
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

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

16.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. The following figure shows how the port pin control signals from the simplified Figure 16-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 16-5. Alternate Port Functions[1]



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

PUD:	PULLUP DISABLE
WDx:	WRITE DDRx
RDx:	READ DDRx
RRx:	READ PORTx REGISTER
WRx:	WRITE PORTx
RPx:	READ PORTx PIN
WPx:	WRITE PINx
$clk_{I/O}$:	I/O CLOCK
DIxn:	DIGITAL INPUT PIN n ON PORTx
AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

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

The following table summarizes the function of the overriding signals. The pin and port indexes from previous figure are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 16-2. Generic Description of Overriding Signals for Alternate Functions

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

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

16.3.1 Alternate Functions of Port A

The Port A pins with alternate functions are shown in the table below:

Table 16-3. Port A Pins Alternate Functions

Port Pin	Alternate Functions
PA7	ADC7 (ADC input channel 7) PCINT7 (Pin Change Interrupt 7)
PA6	ADC6 (ADC input channel 6) PCINT6 (Pin Change Interrupt 6)

Port Pin	Alternate Functions
PA5	ADC5 (ADC input channel 5) PCINT5 (Pin Change Interrupt 5)
PA4	ADC4 (ADC input channel 4) PCINT4 (Pin Change Interrupt 4)
PA3	ADC3 (ADC input channel 3) PCINT3 (Pin Change Interrupt 3)
PA2	ADC2 (ADC input channel 2) PCINT2 (Pin Change Interrupt 2)
PA1	ADC1 (ADC input channel 1) PCINT1 (Pin Change Interrupt 1)
PA0	ADC0 (ADC input channel 0) PCINT0 (Pin Change Interrupt 0)

The alternate pin configuration is as follows:

- ADC[7:0]/PCINT[7:0] – Port A, Bit [7:0]
  - ADC[7:0]: Analog to Digital Converter Channels [7:0].
  - PCINT[7:0]: Pin Change Interrupt source [7:0]. The PA[7:0] pins can serve as external interrupt sources.

Table 16-4. Overriding Signals for Alternate Functions in PA7...PA4

Signal Name	PA7/ADC7/ PCINT7	PA6/ADC6/ PCINT6	PA5/ADC5/ PCINT5	PA4/ADC4/ PCINT4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT7 • PCIE0 + ADC7D	PCINT6 • PCIE0 + ADC6D	PCINT5 • PCIE0 + ADC5D	PCINT4 • PCIE0 + ADC4D
DIEOV	PCINT7 • PCIE0	PCINT6 • PCIE0	PCINT5 • PCIE0	PCINT4 • PCIE0
DI	PCINT7 INPUT	PCINT6 INPUT	PCINT5 INPUT	PCINT4 INPUT
AIO	ADC7 INPUT	ADC6 INPUT	ADC5 INPUT	ADC4 INPUT

Table 16-5. Overriding Signals for Alternate Functions in PA3...PA0

Signal Name	PA3/ADC3/ PCINT3	PA2/ADC2/ PCINT2	PA1/ADC1/ PCINT1	PA0/ADC0/ PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT3 • PCIE0 + ADC3D	PCINT2 • PCIE0 + ADC2D	PCINT1 • PCIE0 + ADC1D	PCINT0 • PCIE0 + ADC0D
DIEOV	PCINT3 • PCIE0	PCINT2 • PCIE0	PCINT1 • PCIE0	PCINT0 • PCIE0
DI	PCINT3 INPUT	PCINT2 INPUT	PCINT1 INPUT	PCINT0 INPUT
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

16.3.2 Alternate Functions of Port B

The Port B pins with alternate functions are shown in the table below:

Table 16-6. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK0 (SPI0 Bus Master clock input) OC3B (Timer/Counter 3 Output Compare Match B Output) OC4B (Timer/Counter 4 Output Compare Match B Output) PCINT15 (Pin Change Interrupt 15)
PB6	MISO0 (SPI0 Bus Master Input/Slave Output) OC3A (Timer/Counter 3 Output Compare Match A Output) PCINT14 (Pin Change Interrupt 14)
PB5	MOSI0 (SPI0 Bus Master Output/Slave Input) ICP3 (Timer/Counter3 Input Capture Trigger) PCINT13 (Pin Change Interrupt 13)
PB4	$\overline{\text{SS0}}$ (SPI0 Slave Select input) OC0B (Timer/Counter 0 Output Compare Match B Output) PCINT12 (Pin Change Interrupt 12)
PB3	AIN1 (Analog Comparator Negative Input) OC0A (Timer/Counter 0 Output Compare Match A Output) PCINT11 (Pin Change Interrupt 11)

Port Pin	Alternate Functions
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input) PCINT10 (Pin Change Interrupt 10)
PB1	T1 (Timer/Counter 1 External Counter Input) CLKO (Divided System Clock Output) PCINT9 (Pin Change Interrupt 9)
PB0	T0 (Timer/Counter 0 External Counter Input) XCK0 (USART0 External Clock Input/Output) PCINT8 (Pin Change Interrupt 8)

The alternate pin configuration is as follows:

- SCK0/OC3B/OC4B/PCINT15 – Port B, Bit 7
  - SCK0: Master Clock output, Slave Clock input pin for SPI0 channel. When the SPI0 is enabled as a slave, this pin is configured as an input regardless of the setting of DDRB7. When the SPI0 is enabled as a master, the data direction of this pin is controlled by DDRB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit.
  - OC3B: Output Compare Match B output. The PB7 pin can serve as an external output for the Timer/Counter3 Output Compare. The pin has to be configured as an output (DDRB7 set “1”) to serve this function. The OC3B pin is also the output pin for the PWM mode timer function.
  - OC4B: Output Compare Match B output. The PB7 pin can serve as an external output for the Timer/Counter4 Output Compare. The pin has to be configured as an output (DDRB7 set “1”) to serve this function. The OC4B pin is also the output pin for the PWM mode timer function.
  - PCINT15: Pin Change Interrupt source 15. The PB7 pin can serve as an external interrupt source.
- MISO0/OC3A/PCINT14 – Port B, Bit 6
  - MISO0: Master Data input, Slave Data output pin for SPI0 channel. When the SPI0 is enabled as a master, this pin is configured as an input regardless of the setting of DDRB6. When the SPI0 is enabled as a slave, the data direction of this pin is controlled by DDRB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit.
  - OC3A: Output Compare Match A output. The PB6 pin can serve as an external output for the Timer/Counter3 Output Compare. The pin has to be configured as an output (DDRB6 set “1”) to serve this function. The OC3A pin is also the output pin for the PWM mode timer function.
  - PCINT14: Pin Change Interrupt source 14. The PB6 pin can serve as an external interrupt source.
- MOSI0/ICP3/PCINT13 – Port B, Bit 5
  - MOSI0: SPI0 Master Data output, Slave Data input for SPI0 channel. When the SPI0 is enabled as a slave, this pin is configured as an input regardless of the setting of DDRB5. When the SPI0 is enabled as a master, the data direction of this pin is controlled by DDRB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit.
  - ICP3: Input Capture Pin 3. The PB5 pin can act as an input capture pin for Timer/Counter3.
  - PCINT13: Pin Change Interrupt source 13. The PB5 pin can serve as an external interrupt source.

- $\overline{\text{SS0}}$/OC0B/PCINT12 – Port B, Bit 4
    - $\overline{\text{SS0}}$: Slave Port Select input. When the SPI0 is enabled as a slave, this pin is configured as an input regardless of the setting of DDRB4. As a slave, the SPI0 is activated when this pin is driven low. When the SPI0 is enabled as a master, the data direction of this pin is controlled by DDRB4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB4 bit.
    - OC0B: Output Compare Match B output. The PB4 pin can serve as an external output for the Timer/Counter0 Output Compare. The pin has to be configured as an output (DDRB4 set “1”) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
    - PCINT12: Pin Change Interrupt source 12. The PB4 pin can serve as an external interrupt source.
- AIN1/OC0A/PCINT11– Port B, Bit 3
    - AIN1: Analog Comparator Negative input. This pin is directly connected to the negative input of the Analog Comparator.
    - OC0A: Output Compare Match A output. The PB3 pin can serve as an external output for the Timer/Counter0 Output Compare. The pin has to be configured as an output (DDRB3 set “1”) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.
    - PCINT11: Pin Change Interrupt source 11. The PB3 pin can serve as an external interrupt source.
- AIN0/INT2/PCINT10 – Port B, Bit 2
    - AIN0: Analog Comparator Positive input. This pin is directly connected to the positive input of the Analog Comparator.
    - INT2: External Interrupt source 2. The PB2 pin can serve as an External Interrupt source to the MCU.
    - PCINT10: Pin Change Interrupt source 10. The PB2 pin can serve as an external interrupt source.
- T1/CLKO/PCINT9 – Port B, Bit 1
    - T1: Timer/Counter1 counter source.
    - CLKO: Divided System Clock: The divided system clock can be output on the PB1 pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTB1 and DDRB1 settings. It will also be output during reset.
    - PCINT9: Pin Change Interrupt source 9: The PB1 pin can serve as an external interrupt source.
- T0/XCK0/PCINT8 – Port B, Bit 0
    - T0: Timer/Counter0 counter source.
    - XCK0: USART0 External clock. The Data Direction Register (DDRB0) controls whether the clock is output (DDRB0 set “1”) or input (DDRB0 cleared). The XCK0 pin is active only when the USART0 operates in Synchronous mode.
    - PCINT8: Pin Change Interrupt source 8. The PB0 pin can serve as an external interrupt source.

Table 16-7 and Table 16-8 relate the alternate functions of Port B to the overriding signals shown in Figure 16-5. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 16-7. Overriding Signals for Alternate Functions in PB7...PB4

Signal Name	PB7/SCK0/OC3B/OC4B/ PCINT15	PB6/MISO0/OC3A/ PCINT14	PB5/MOSI0/ICP3/ PCINT13	PB4/$\overline{\text{SS0}}$/OC0B/ PCINT12
PUOE	SPE0 • $\overline{\text{MSTR}}$	SPE0 • MSTR	SPE0 • $\overline{\text{MSTR}}$	SPE0 • $\overline{\text{MSTR}}$
PUOV	PORTB7 • $\overline{\text{PUD}}$	PORTB6 • $\overline{\text{PUD}}$	PORTB5 • $\overline{\text{PUD}}$	PORTB4 • $\overline{\text{PUD}}$
DDOE	SPE0 • $\overline{\text{MSTR}}$	SPE0 • MSTR	SPE0 • $\overline{\text{MSTR}}$	SPE0 • $\overline{\text{MSTR}}$
DDOV	0	0	0	0
PVOE	SPE0 • MSTR OC3B/OC4B ENABLE	SPE0 • $\overline{\text{MSTR}}$ OC3A ENABLE	SPE0 • MSTR	OC0B ENABLE
PVOV	SCK0 OUTPUT + OC3B/ OC4B	SPI0 SLAVE OUTPUT + OC3A	SPI0 MSTR OUTPUT	OC0B
DIEOE	PCINT15 • PCIE1	PCINT14 • PCIE1	PCINT13 • PCIE1	PCINT12 • PCIE1
DIEOV	1	1	1	1
DI	SCK0 INPUT PCINT15 INPUT	SPI0 MSTR INPUT PCINT14 INPUT	SPI0 SLAVE INPUT ICP3 INPUT PCINT13 INPUT	SPI $\overline{\text{SS0}}$ PCINT12 INPUT
AIO	-	-	–	–

Table 16-8. Overriding Signals for Alternate Functions in PB3...PB0

Signal Name	PB3/AIN1/OC0A/PCINT11	PB2/AIN0/INT2/PCINT10	PB1/T1/CLKO/PCINT9	PB0/T0/XCK0/PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	CKOUT	0
DDOV	0	0	CKOUT	0
PVOE	OC0A ENABLE	0	CKOUT	UMSEL
PVOV	OC0A	0	CLK I/O	XCK0 OUTPUT
DIEOE	PCINT11 • PCIE1	INT2 ENABLE PCINT10 • PCIE1	PCINT9 • PCIE1	PCINT8 • PCIE1
DIEOV	1	1	1	1
DI	PCINT11 INPUT	INT2 INPUT PCINT10 INPUT	T1 INPUT PCINT9 INPUT	T0 INPUT XCK0 INPUT PCINT8 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

16.3.3 Alternate Functions of Port C

The Port C pins with alternate functions are shown in the table below:

Table 16-9. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator pin 2) PCINT23 (Pin Change Interrupt 23)
PC6	TOSC1 (Timer Oscillator pin 1) PCINT22 (Pin Change Interrupt 22)
PC5	ACO TDI (JTAG Test Data Input) PCINT21 (Pin Change Interrupt 21)
PC4	OC4A (Timer/Counter 4 Output Compare Match A Output) TDO (JTAG Test Data Output) PCINT20 (Pin Change Interrupt 20)
PC3	ICP4 (Timer/Counter4 Input Capture Trigger) TMS (JTAG Test Mode Select) PCINT19 (Pin Change Interrupt 19)
PC2	T4 (Timer/Counter 4 External Counter Input) TCK (JTAG Test Clock) PCINT18 (Pin Change Interrupt 18)
PC1	SDA0 (two-wire Serial Bus0 Data Input/Output Line) PCINT17 (Pin Change Interrupt 17)
PC0	SCL0 (two-wire Serial Bus0 Clock Line) PCINT16 (Pin Change Interrupt 16)

The alternate pin configuration is as follows:

- TOSC2/PCINT23 – Port C, Bit 7
  - TOSC2, Timer Oscillator pin 2. The PC7 pin can serve as an external interrupt source to the MCU.
  - PCINT23: Pin Change Interrupt source 23. The PC7 pin can serve as an external interrupt source.
- TOSC1/PCINT22 – Port C, Bit 6
  - TOSC1, Timer Oscillator pin 1. The PC6 pin can serve as an external interrupt source to the MCU.
  - PCINT22: Pin Change Interrupt source 22. The PC6 pin can serve as an external interrupt source.

- TDI/PCINT21 – Port C, Bit 5
  - TDI: JTAG Test Data Input
  - PCINT21: Pin Change Interrupt source 21. The PC5 pin can serve as an external interrupt source.
- OC4A/TDO/PCINT20 – Port C, Bit 4
  - OC4A: Timer/Counter 4 Output Compare Match A Output
  - TDO: JTAG Test Data Output
  - PCINT20: Pin Change Interrupt source 20. The PC4 pin can serve as an external interrupt source.
- TMS/ICP4/PCINT19 – Port C, Bit 3
  - TMS: JTAG Test Mode Select
  - ICP4: Timer/Counter4 Input Capture Trigger
  - PCINT19: Pin Change Interrupt source 19. The PC3 pin can serve as an external interrupt source.
- TCK/T4/PCINT18 – Port C, Bit 2
  - TCK: JTAG Test Clock
  - T4: Timer/Counter 4 External Counter Input
  - PCINT18: Pin Change Interrupt source 18. The PC2 pin can serve as an external interrupt source.
- SDA0/PCINT17 – Port C, Bit 1
  - SDA0: two-wire Serial Bus0 Data Input/Output Line
  - PCINT17: Pin Change Interrupt source 17. The PC1 pin can serve as an external interrupt source.
- SCL0/PCINT16 – Port C, Bit 0
  - SCL0: two-wire Serial Bus0 Clock Line
  - PCINT16: Pin Change Interrupt source 16. The PC0 pin can serve as an external interrupt source.

The tables below relate the alternate functions of Port C to the overriding signals shown in Figure 16-5.

Table 16-10. Overriding Signals for Alternate Functions in PC7...PC4

Signal Name	PC7/TOSC2/PCINT23	PC6/TOSC1/PCINT22	PC5/TDI/PCINT21	PC4/OC4A/TDO/PCINT20
PUOE	AS2 • $\overline{\text{EXCLK}}$	AS2	JTAGEN	JTAGEN
PUOV	0	0	1	1
DDOE	AS2 • $\overline{\text{EXCLK}}$	AS2	JTAGEN	JTAGEN OC4A ENABLE
DDOV	0	0	0	SHIFT_IR + SHIFT_DR + OC4A
PVOE	0	0	0	JTAGEN
PVOV	0	0	0	TDO

Signal Name	PC7/TOSC2/PCINT23	PC6/TOSC1/PCINT22	PC5/TDI/PCINT21	PC4/OC4A/TDO/PCINT20
DIEOE	AS2 • $\overline{\text{EXCLK}}$ + PCINT23 • PCIE2	AS2 + PCINT22 • PCIE2	JTAGEN + PCINT21 • PCIE2	JTAGEN + PCINT20 • PCIE2
DIEOV	$\overline{\text{AS2}}$	EXCLK + $\overline{\text{AS2}}$	$\overline{\text{JTAGEN}}$	$\overline{\text{JTAGEN}}$
DI	PCINT23 INPUT	PCINT22 INPUT	PCINT21 INPUT	PCINT20 INPUT
AIO	T/C2 OSC OUTPUT	T/C1 OSC INPUT	TDI INPUT	-

Table 16-11. Overriding Signals for Alternate Functions in PC3...PC0

Signal Name	PC3/TMS/ICP4/PCINT19	PC2/T4/PCINT18	PC1/SDA0/PCINT17	PC0/SCL0/PCINT16
PUOE	JTAGEN	JTAGEN	TWEN	TWEN
PUOV	1	1	PORTC1 • $\overline{\text{PUD}}$	PORTC0 • $\overline{\text{PUD}}$
DDOE	JTAGEN	JTAGEN	TWEN	TWEN
DDOV	0	0	0	0
PVOE	0	0	TWEN	TWEN
PVOV	0	0	SDA OUT	SCL OUT
DIEOE	JTAGEN + PCINT19 • PCIE2	JTAGEN + PCINT18 • PCIE2	PCINT17 • PCIE2	PCINT16 • PCIE2
DIEOV	$\overline{\text{JTAGEN}}$	$\overline{\text{JTAGEN}}$	1	1
DI	ICP4 INPUT PCINT19 INPUT	T4 INPUT PCINT18 INPUT	PCINT17 INPUT	PCINT16 INPUT
AIO	TMS INPUT	TCK INPUT	SDA0 INPUT	SCL0 INPUT

16.3.4 Alternate Functions of Port D

The Port D pins with alternate functions are shown in the table below:

Table 16-12. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2A (Timer/Counter2 Output Compare Match A Output) XCK2 (USART2 External Clock Input/Output) SCK1 (SPI1 Bus Master clock Input) PCINT31 (Pin Change Interrupt 31)
PD6	ICP1 (Timer/Counter1 Input Capture Trigger) OC2B (Timer/Counter2 Output Compare Match B Output) $\overline{\text{SS1}}$ (SPI1 Slave Select input) PCINT30 (Pin Change Interrupt 30)
PD5	OC1A (Timer/Counter1 Output Compare Match A Output)

Port Pin	Alternate Function
	PCINT29 (Pin Change Interrupt 29)
PD4	OC1B (Timer/Counter1 Output Compare Match B Output) XCK1 (USART1 External Clock Input/Output) PCINT28 (Pin Change Interrupt 28)
PD3	INT1 (External Interrupt1 Input) TXD1 (USART1 Transmit Pin) PCINT27 (Pin Change Interrupt 27)
PD2	INT0 (External Interrupt0 Input) RXD1 (USART1 Receive Pin) PCINT26 (Pin Change Interrupt 26)
PD1	TXD0 (USART0 Transmit Pin) PCINT25 (Pin Change Interrupt 25)
PD0	RXD0 (USART0 Receive Pin) T3 (Timer/Counter 3 External Counter Input) PCINT24 (Pin Change Interrupt 24)

The alternate pin configuration is as follows:

- OC2A/XCK2/SCK1/PCINT31 – Port D, Bit 7
  - OC2A: Output Compare Match output. The PD7 pin can serve as an external output for the Timer/Counter2 Compare Match A. The PD7 pin has to be configured as an output (DDRD7 set '1') to serve this function. The OC2A pin is also the output pin for the PWM mode timer function.
  - XCK2: USART2 External clock. The Data Direction Register (DDRD7) controls whether the clock is output (DDRD7 set “1”) or input (DDRD7 cleared). The XCK2 pin is active only when the USART2 operates in Synchronous mode.
  - SCK1: Master Clock output, Slave Clock input pin for SPI1 channel. When the SPI1 is enabled as a Slave, this pin is configured as an input regardless of the setting of DDRD7. When the SPI1 is enabled as a Master, the data direction of this pin is controlled by DDRD7.
  - PCINT31: Pin Change Interrupt source 31. The PD7 pin can serve as an external interrupt source.
- ICP1/OC2B/$\overline{\text{SS1}}$/PCINT30 – Port D, Bit 6
  - ICP1: Input Capture Pin 1. The PD6 pin can act as an input capture pin for Timer/Counter1.
  - OC2B: Output Compare Match B output. The PD6 pin can serve as an external output for the Timer/Counter2 Output Compare B. The pin has to be configured as an output (DDRD6 set '1') to serve this function. The OC2B pin is also the output pin for the PWM mode timer function.
  - $\overline{\text{SS1}}$: Slave Port Select input. When the SPI1 is enabled as a slave, this pin is configured as an input regardless of the setting of DDRD6. As a slave, the SPI1 is activated when this pin is driven low. When the SPI1 is enabled as a master, the data direction of this pin is controlled

  by DDRD6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD6 bit.
  - PCINT30: Pin Change Interrupt source 30. The PD6 pin can serve as an external interrupt source.
- OC1A/PCINT29 – Port D, Bit 5
  - OC1A: Output Compare Match output. The PD5 pin can serve as an external output for the Timer/Counter1 Compare Match A. The PD5 pin has to be configured as an output (DDRD5 set '1') to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.
  - PCINT29: Pin Change Interrupt source 29. The PD4 pin can serve as an external interrupt source.
- OC1B/XCK1/PCINT28 – Port D, Bit 4
  - OC1B: Output Compare Match B output. The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDRD4 set '1') to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.
  - XCK1: USART1 external clock.
  - PCINT28: Pin Change Interrupt source 28. The PD4 pin can serve as an external interrupt source.
- INT1/TXD1/PCINT27 – Port D, Bit 3
  - INT1: External Interrupt source 1. The PD3 pin can serve as an external interrupt source.
  - TXD1: 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 DDRD3.
  - PCINT27: Pin Change Interrupt source 27. The PD3 pin can serve as an external interrupt source.
- INT0/RXD1/PCINT26 – Port D, Bit 2
  - INT0: External Interrupt source 0. The PD2 pin can serve as an external interrupt source.
  - RXD1: Receive Data (Data input pin for the USART1). When the USART1 Receiver is enabled this pin is configured as an input regardless of the value of DDRD2. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.
  - PCINT26: Pin Change Interrupt source 26. The PD2 pin can serve as an external interrupt source.
- TXD0/PCINT25 – Port D, Bit 1
  - TXD0: Transmit Data (Data output pin for the USART0). When the USART0 Transmitter is enabled, this pin is configured as an output regardless of the value of DDRD1.
  - PCINT25: Pin Change Interrupt source 25. The PD1 pin can serve as an external interrupt source.
- RXD0/PCINT24 – Port D, Bit 0
  - RXD0: Receive Data (Data input pin for the USART0). When the USART0 Receiver is enabled this pin is configured as an input regardless of the value of DDRD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.
  - T3: Timer/Counter 3 External Counter Input
  - PCINT24: Pin Change Interrupt source 24. The PD0 pin can serve as an external interrupt source.

The tables below relate the alternate functions of Port D to the overriding signals shown in Figure 16-5.

Table 16-13. Overriding Signals for Alternate Functions PD7...PD4

Signal Name	PD7/OC2A/XCK2/SCK1/ PCINT31	PD6/ICP1/OC2B/$\overline{SS1}$/ PCINT30	PD5/OC1A/PCINT29	PD4/OC1B/XCK1/ PCINT28
PUOE	SPE1 • $\overline{MSTR}$	SPE1 • $\overline{MSTR}$	0	0
PUO	PORTD7 • $\overline{PUD}$	PORTD6 • $\overline{PUD}$	0	0
DDOE	SPE1 • $\overline{MSTR}$	SPE1 • $\overline{MSTR}$	0	0
DDOV	0	0	0	0
PVOE	SPE1 • $\overline{MSTR}$ OC2A ENABLE	OC2B ENABLE	OC1A ENABLE	OC1B ENABLE
PVOV	OC2A SCK1 OUTPUT	OC2B	OC1A	OC1B
DIEOE	PCINT31 • PCIE3	PCINT30 • PCIE3	PCINT29 • PCIE3	PCINT28 • PCIE3
DIEOV	1	1	1	1
DI	SCK1 INPUT PCINT31 INPUT	ICP1 INPUT SPI1 $\overline{SS}$ PCINT30 INPUT	PCINT29 INPUT T1 INPUT	PCINT28 INPUT
AIO	-	-	–	–

Table 16-14. Overriding Signals for Alternate Functions in PD3...PD0

Signal Name	PD3/INT1/TXD1/PCINT27	PD2/INT0/RXD1/PCINT26	PD1/TXD0/PCINT25	PD0/T3/RXD0/PCINT24
PUOE	TXEN1	RXEN1	TXEN0	RXEN0
PUO	0	PORTD2 • $\overline{PUD}$	0	PORTD0 • $\overline{PUD}$
DDOE	TXEN1	RXEN1	TXEN0	RXEN0
DDOV	1	0	1	0
PVOE	TXEN1	0	TXEN0	0
PVOV	TXD1	0	TXD0	0
DIEOE	INT1 ENABLE PCINT27 • PCIE3	INT2 ENABLE PCINT26 • PCIE3	PCINT25 • PCIE3	PCINT24 • PCIE3
DIEOV	1	1	1	1
DI	INT1 INPUT PCINT27 INPUT	INT0 INPUT	PCINT25 INPUT	RXD0 T4 INPUT PCINT24 INPUT

Signal Name	PD3/INT1/TXD1/PCINT27	PD2/INT0/RXD1/PCINT26	PD1/TXD0/PCINT25	PD0/T3/RXD0/PCINT24
		RXD1 PCINT26 INPUT		
AIO	–	–	–	–

16.3.5 Alternate Functions of Port E

The Port E pins with alternate functions are shown in this table:

Table 16-15. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE6	SCL1 (two-wire Serial Bus1 Clock Line)
PE5	SDA1 (two-wire Serial Bus1 Data Input/Output Line))
PE4	AREF (Analog Reference Pin)
PE3	TXD2 (USART2 Transmit Pin) MOSI1 (SPI Bus1 Master Output/Slave Input)
PE2	RXD2 (USART2 Receive Pin) MISO1 (SPI Bus1 Master Input/Slave Output)
PE1	XTAL1 (Chip Clock Oscillator pin 1)
PE0	XTAL2 (Chip Clock Oscillator pin 2)

The alternate pin configuration is as follows:

- SCL1 – Port E, Bit 6
  - SCL1: two-wire Serial Bus1 Clock Line.
- SDA1 – Port E, Bit 5
  - SDA1: two-wire Serial Bus1 Data Input/Output Line.
- AREF – Port E, Bit 4
  - AREF (Analog Reference Pin)
- TXD2/MOSI1– Port E, Bit 3
  - TXD2: Transmit Data (Data output pin for the USART2). When the USART2 Transmitter is enabled, this pin is configured as an output regardless of the value of DDRE3.
  - MOSI1: SPI Master Data output, Slave Data input for SPI1 channel. When the SPI1 is enabled as a slave, this pin is configured as an input regardless of the setting of DDRE3. When the SPI1 is enabled as a master, the data direction of this pin is controlled by DDRE3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTE3 bit.
- RXD2/MISO1 – Port E, Bit 2
  - RXD2: Receive Data (Data input pin for the USART2). When the USART2 receiver is enabled this pin is configured as an input regardless of the value of DDRE2. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTE2 bit.
  - MISO1: Master Data input, Slave Data output pin for SPI1 channel. When the SPI1 is enabled as a master, this pin is configured as an input regardless of the setting of DDRE2. When the

SPI1 is enabled as a slave, the data direction of this pin is controlled by DDRE2. When the pin is forced to be an input, the pull-up can still be controlled by the PORTE2 bit.

- XTAL1 – Port E, Bit 1
  - XTAL1: Chip clock Oscillator pin 1. Used for all chip clock sources except internal calibrated RC Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.
- XTAL2 – Port E, Bit 0
  - XTAL2: Chip clock Oscillator pin 1. Used for all chip clock sources except internal calibrated RC Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

Table 16-16 relate the alternate functions of Port E to the overriding signals shown in Figure 16-5.

Table 16-16. Overriding Signals for Alternate Functions in PE6...PE3

Signal Name	PE6/SCL1	PE5/SDA1	PE4/AREF	PE3/TXD2/MOSI1
PUOE	TWEN1	TWEN1	0	TXEN2+SPE1 • $\overline{\text{MSTR}}$
PUOV	PORTE6 • $\overline{\text{PUD}}$	PORTE5 • $\overline{\text{PUD}}$	0	PORTE3 • $\overline{\text{PUD}}$
DDOE	TWEN1	TWEN1	0	TXEN2+SPE1 • $\overline{\text{MSTR}}$
PVOE	0	0	0	1
PVOV	TWEN1	TWEN1	0	TXEN2+SPE1 • $\overline{\text{MSTR}}$
DIEOE	SCL1 OUT	SDA1 OUT	0	SPI1 MSTR OUTPUT
DIEOV	1	1	0	1
DI	SCL1 INPUT	SDA1 INPUT	0	SPI1 SLAVE INPUT
AIO	-	-	-	-

Table 16-17. Overriding Signals for Alternate Functions in PE2...PE0

Signal Name	PE2/RXD2/MISO1	PE1/XTAL1	PE0/XTAL2
PUOE	RXEN2+SPE1 • MSTR	0	0
PUOV	PORTE2 • $\overline{\text{PUD}}$	0	0
DDOE	RXEN2+SPE1 • MSTR	0	0
PVOE	0	0	0
PVOV	RXEN2+SPE1 • MSTR	0	0
DIEOE	SPI1 SLAVE OUTPUT	0	0
DIEOV	1	0	0
DI	RXD2	0	0

Signal Name	PE2/RXD2/MISO1	PE1/XTAL1	PE0/XTAL2
	SPI1 MSTR INPUT		
AIO	-	-	-

16.4 Register Description

16.4.1 MCU Control Register

The MCU Control Register controls the placement of the Interrupt Vector table in order to move interrupts between application and boot space.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: MCUCR
Offset: 0x55
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x35

Bit	7	6	5	4	3	2	1	0
	JTD	BODS	BODSE	PUD			IVSEL	IVCE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 7 – JTD
When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

Bit 6 – BODS: BOD Sleep
The BODS bit must be written to '1' in order to turn off BOD during sleep. Writing to the BODS bit is controlled by a timed sequence and the enable bit BODSE. To disable BOD in relevant sleep modes, both BODS and BODSE must first be written to '1'. Then, BODS must be written to '1' and BODSE must be written to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

Bit 5 – BODSE: BOD Sleep Enable
BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

Bit 4 – PUD: Pull-up Disable
When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

Bit 1 – IVSEL: Interrupt Vector Select
When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

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

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

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section.

Bit 0 – IVCE: Interrupt Vector Change Enable
The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example

```
Move_interrupts:
; Get MCUCR
in    r16, MCUCR
mov   r17, r16
; Enable change of Interrupt Vectors
ori   r16, (1<<IVCE)
out   MCUCR, r16
; Move interrupts to Boot Flash section
ori   r17, (1<<IVSEL)
out   MCUCR, r17
ret
```

C Code Example

```
void Move_interrupts(void)
{
uchar temp;
/* GET MCUCR*/
temp = MCUCR;
/* Enable change of Interrupt Vectors */
MCUCR = temp|(1<<IVCE);
/* Move interrupts to Boot Flash section */
MCUCR = temp|(1<<IVSEL);
}
```

16.4.2 Port A Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PORTA
Offset: 0x22
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x02

Bit	7	6	5	4	3	2	1	0
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PORTA: Port A Data

16.4.3 Port A Data Direction Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: DDRA
Offset: 0x21
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x01

Bit	7	6	5	4	3	2	1	0
	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – DDRA: Port A Data Direction
This bit field selects the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual DIR register for the Port.

16.4.4 Port A Input Pins Address

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PINA
Offset: 0x20
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x00

Bit	7	6	5	4	3	2	1	0
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PINA: Port A Input Pins Address
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

16.4.5 Port B Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PORTB
Offset: 0x25
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x05

Bit	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PORTB: Port B Data

16.4.6 Port B Data Direction Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: DDRB
Offset: 0x24
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x04

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – DDRB: Port B Data Direction
This bit field selects the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual DIR register for the Port.

16.4.7 Port B Input Pins Address

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PINB
Offset: 0x23
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x03

Bit	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PINB: Port B Input Pins Address
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

16.4.8 Port C Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PORTC
Offset: 0x28
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x08

Bit	7	6	5	4	3	2	1	0
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PORTC: Port C Data

16.4.9 Port C Data Direction Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: DDRC
Offset: 0x27
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x07

Bit	7	6	5	4	3	2	1	0
	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – DDRC: Port C Data Direction
This bit field selects the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual DIR register for the Port.

16.4.10 Port C Input Pins Address

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PINC
Offset: 0x26
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x06

Bit	7	6	5	4	3	2	1	0
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PINC: Port C Input Pins Address
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

16.4.11 Port D Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PORTD
Offset: 0x2B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0B

Bit	7	6	5	4	3	2	1	0
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PORTD: Port D Data

16.4.12 Port D Data Direction Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: DDRD
Offset: 0x2A
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0A

Bit	7	6	5	4	3	2	1	0
	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – DDRD: Port D Data Direction
This bit field selects the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual DIR register for the Port.

16.4.13 Port D Input Pins Address

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PIND
Offset: 0x29
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x09

Bit	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PIND: Port D Input Pins Address
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

16.4.14 Port E Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PORTE
Offset: 0x2E
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0E

Bit	7	6	5	4	3	2	1	0
		PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6 – PORTE: Port E Data [n = 6:0]

16.4.15 Port E Data Direction Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: DDRE
Offset: 0x2D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0D

Bit	7	6	5	4	3	2	1	0
		DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6 – DDRE: Port E Data Direction
This bit field selects the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual DIR register for the Port.

16.4.16 Port E Input Pins Address

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PINE
Offset: 0x2C
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x0C

Bit	7	6	5	4	3	2	1	0
		PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6 – PINE: Port E Input Pins Address [n = 6:0]
Writing to the pin register provides toggle functionality for I/O.

17. TC0 - 8-bit Timer/Counter0 with PWM

17.1 Features

- Two independent Output Compare Units
- Double Buffered Output Compare Registers
- Clear Timer on Compare Match (Auto Reload)
- Glitch free, phase correct Pulse Width Modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

17.2 Overview

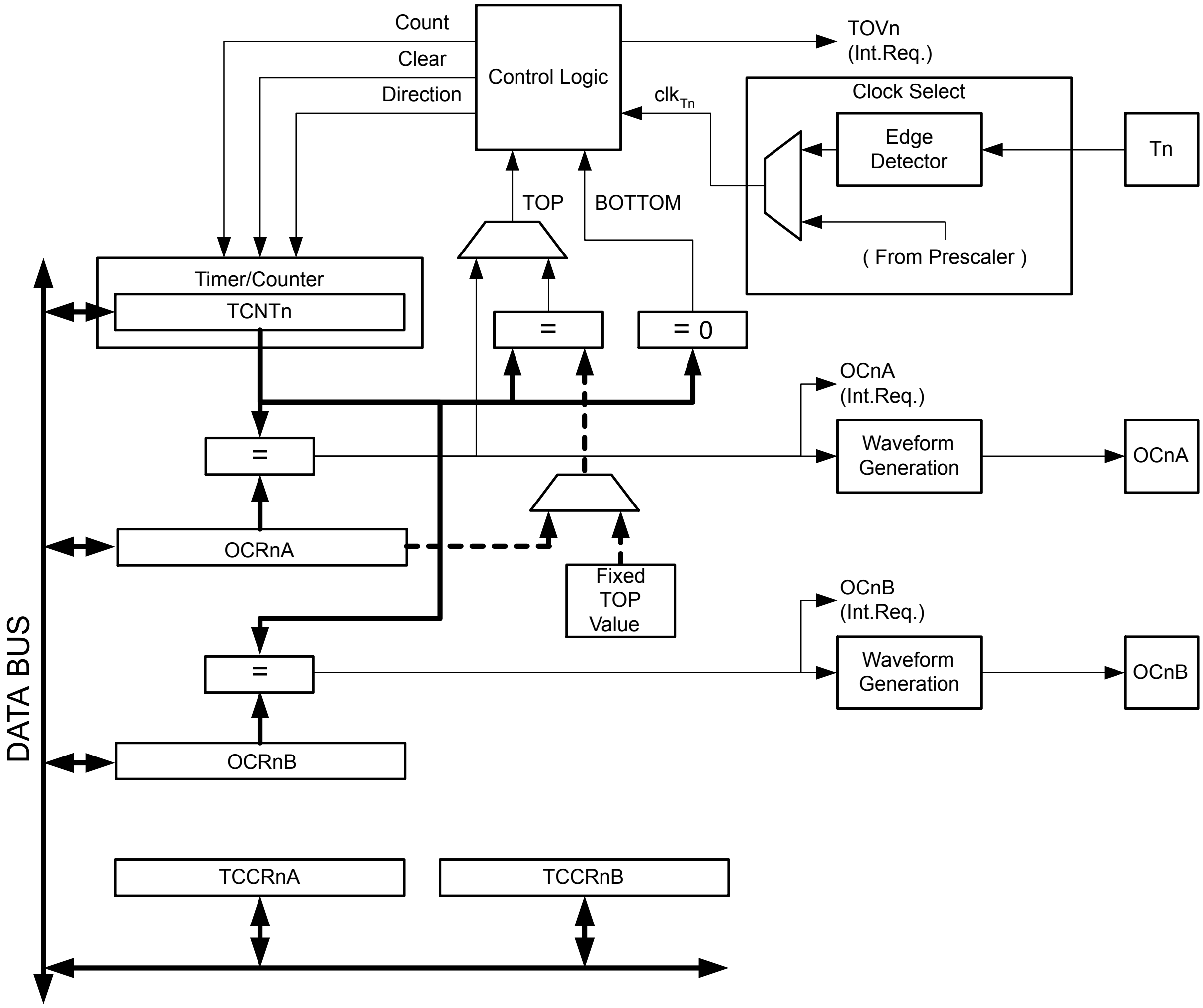
Timer/Counter0 (TC0) is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown below. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the Register Description. For the actual placement of I/O pins, refer to the pinout diagram.

The TC0 is enabled by writing the PRTIM0 bit in ”Minimizing Power Consumption” to '0'.

The TC0 is enabled when the PRTIM0 bit in the Power Reduction Register (PRR0.PRTIM0) is written to '1'.

Figure 17-1. 8-bit Timer/Counter Block Diagram



17.2.1 Definitions

Many register and bit references in this section are written in general form:

- n=0 represents the Timer/Counter number
- x=A,B represents the Output Compare Unit A or B

However, when using the register or bit definitions in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value.

The following definitions are used throughout the section:

Table 17-1. Definitions

Constant	Description
BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00 for 8-bit counters, or 0x0000 for 16-bit counters).
MAX	The counter reaches its Maximum when it becomes 0xFF (decimal 255, for 8-bit counters) or 0xFFFF (decimal 65535, for 16-bit counters).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value MAX or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

17.2.2 Registers

The Timer/Counter 0 register (TCNT0) and Output Compare TC0x registers (OCR0x) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the block diagram) signals are all visible in the Timer Interrupt Flag Register 0 (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register 0 (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

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

The double buffered Output Compare Registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC0A and OC0B). See Output Compare Unit for details. The compare match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

Related Links

Timer/Counter 0, 1, 3, 4 Prescalers

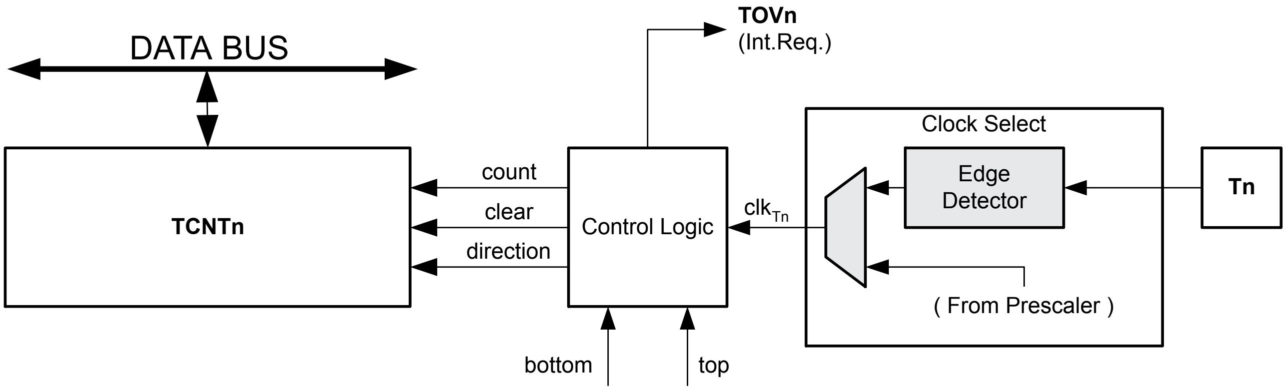
17.3 Timer/Counter Clock Sources

The TC can be clocked by an internal or an external clock source. The clock source is selected by writing to the Clock Select (CS0[2:0]) bits in the Timer/Counter Control Register (TCCR0B).

17.4 Counter Unit

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

Figure 17-2. Counter Unit Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

Table 17-2. Signal description (internal signals)

Signal Name	Description
count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS0[2:0]). When no clock source is selected (CS0=0x0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

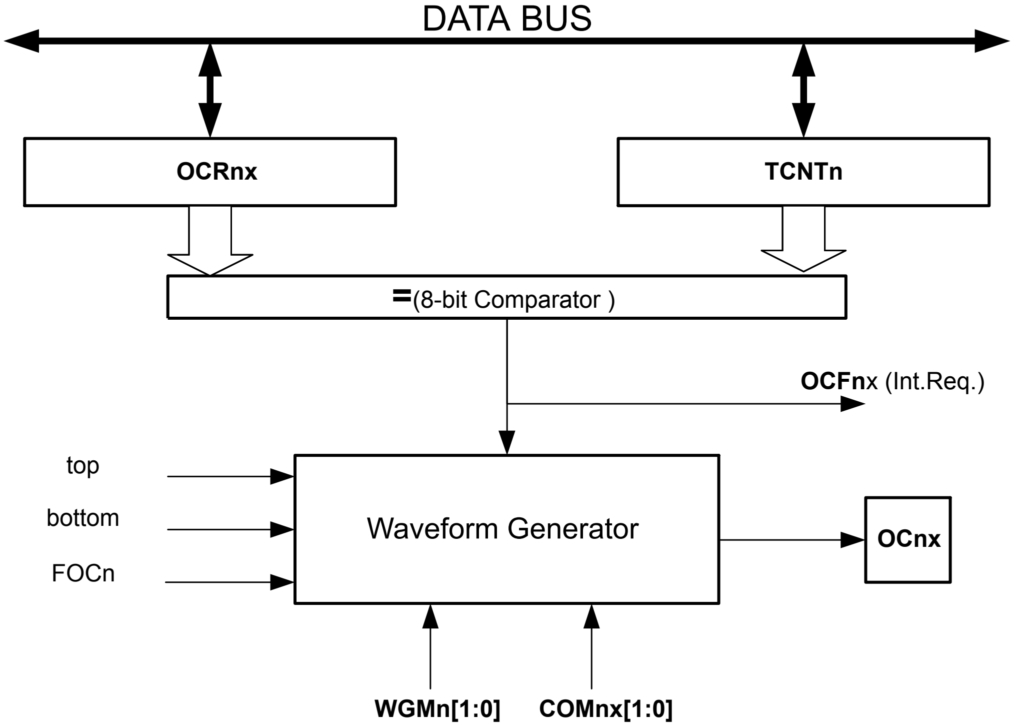
The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see Modes of Operation.

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM0[2:0] bits. TOV0 can be used for generating a CPU interrupt.

17.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a '1' to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM02, WGM01, and WGM00 bits and Compare Output mode (COM0x[1:0]) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation.

Figure 17-3. Output Compare Unit, Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The OCR0x Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. When double buffering is enabled, the CPU has access to the OCR0x Buffer Register. The double buffering synchronizes the update of the OCR0x Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The double buffering is disabled for the normal and Clear Timer on Compare (CTC) modes of operation, and the CPU will access the OCR0x directly.

17.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a '1' to the Force Output Compare (TCCR0C.FOCnx) bit. Forcing compare match will not set the OCFnx Flag or reload/clear the timer, but the OCnx pin will be updated as if a real compare match had occurred (the TCCRnA.COMnx[1:0] bits define whether the OCnx pin is set, cleared or toggled).

17.5.2 Compare Match Blocking by TCNTn Write

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

17.5.3 Using the Output Compare Unit

Since writing TCNTn in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNTn when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNTn equals the OCRnx value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNTn1 value equal to BOTTOM when the counter is down counting.

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

Be aware that the TCCRnA.COMnx[1:0] bits are not double buffered together with the compare value. Changing the TCCRnA.COMnx[1:0] bits will take effect immediately.

17.6 Compare Match Output Unit

The Compare Output mode bits in the Timer/Counter Control Register A (TCCR0A.COM0x) have two functions:

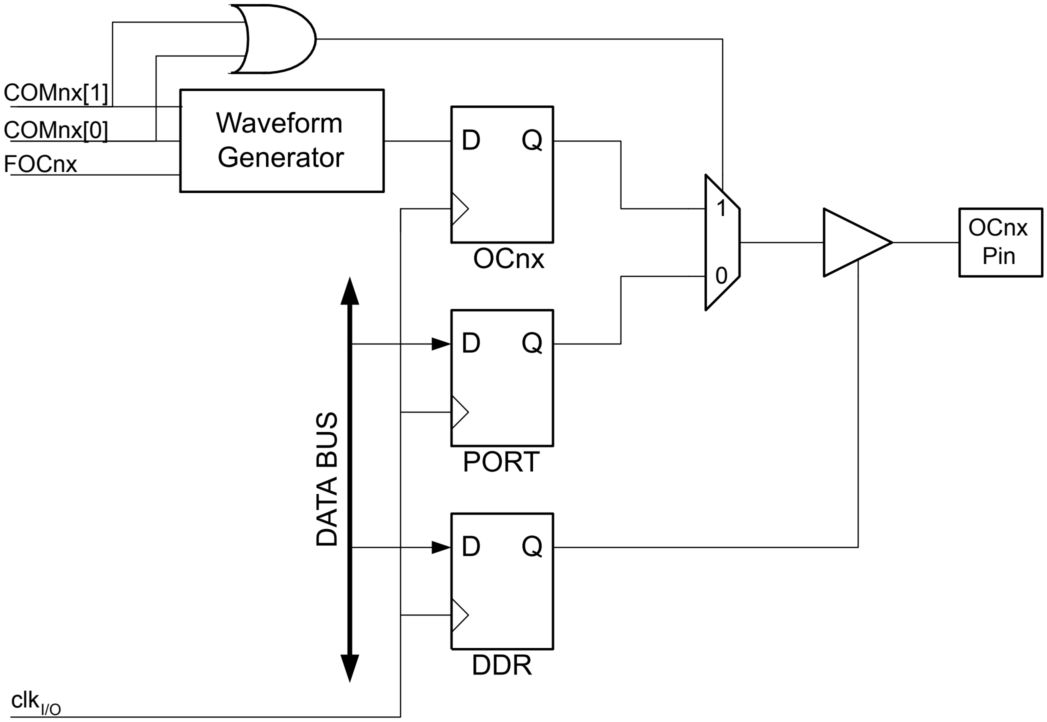
- The Waveform Generator uses the COM0x bits for defining the Output Compare (OC0x) register state at the next compare match.
- The COM0x bits control the OC0x pin output source

The figure below shows a simplified schematic of the logic affected by COM0x. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers that are affected by the COM0x bits are shown, namely PORT and DDR.

On system reset the OC0x Register is reset to 0x00.

Note: 'OC0x state' is always referring to internal OC0x *registers*, not the OC0x *pin*.

Figure 17-4. Compare Match Output Unit, Schematic



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OC0x) from the Waveform Generator if either of the COM0x[1:0] bits are set. However, the OC0x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. In the Data Direction Register, the bit for the OCnx pin (DDR.OC0x) must be set as output before the OC0x value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC0x register state before the output is enabled. Some TCCR0A.COM0x[1:0] bit settings are reserved for certain modes of operation.

The TCCR0A.COM0x[1:0] bits have no effect on the Input Capture unit.

Related Links

Register Description

17.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the TCCR0A.COM0x[1:0] bits differently in Normal, CTC, and PWM modes. For all modes, setting the TCCR0A.COM0x[1:0]=0x0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the TCCR0A.COM0x[1:0] bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the TCCR0C.FOC0x strobe bits.

17.7 Modes of Operation

The mode of operation determines the behavior of the Timer/Counter and the Output Compare pins. It is defined by the combination of the Waveform Generation mode bits and Compare Output mode (TCCR0A.WGM0[2:0]) bits in the Timer/Counter control Registers A and B (TCCR0A.COM0x[1:0]). The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x[1:0] bits control whether the output should be set, cleared, or toggled at a compare match (See previous section *Compare Match Output Unit*).

For detailed timing information refer to the following section *Timer/Counter Timing Diagrams*.

Related Links

Compare Match Output Unit

Timer/Counter Timing Diagrams

17.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM0[2:0] = 0x0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP=0xFF) and then restarts from the bottom (0x00). In Normal mode operation, the Timer/Counter Overflow Flag (TOV0) will be set in the same clock cycle in which the TCNT0 becomes zero. In this case, the TOV0 Flag behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

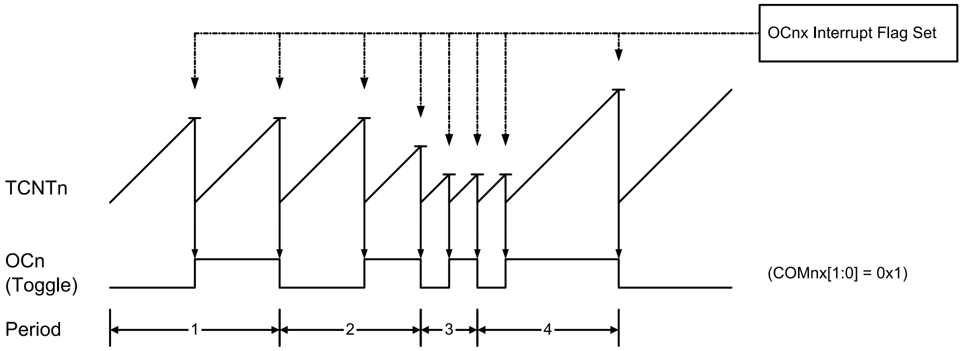
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

17.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM0[2:0]=0x2), the OCR0A Register is used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the counting of external events.

The timing diagram for the CTC mode is shown below. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 17-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by setting the OCF0A Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value.

Note: Changing TOP to a value close to BOTTOM while the counter is running must be done with care, since the CTC mode does not provide double buffering. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then count to its maximum value (0xFF for a 8-bit counter, 0xFFFF for a 16-bit counter) and wrap around starting at 0x00 before the compare match will occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by writing the two least significant Compare Output mode bits in the Timer/Counter Control Register A Control to toggle mode (TCCR0A.COM0A[1:0]=0x1). The OC0A value will only be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is written to 0x00. The waveform frequency is defined by the following equation:

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

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

As for the Normal mode of operation, the Timer/Counter Overflow Flag TOV0 is set in the same clock cycle that the counter wraps from MAX to 0x00.

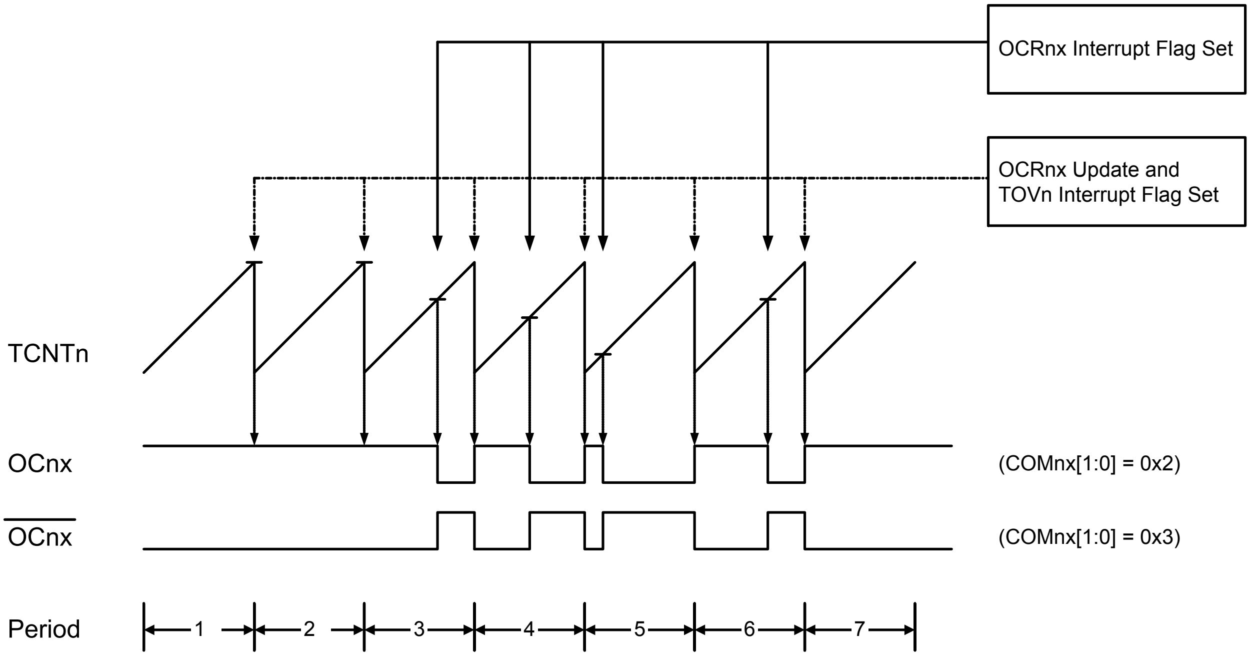
17.7.3 Fast PWM Mode

The Fast Pulse Width Modulation or Fast PWM modes (WGM0[2:0]=0x3 or WGM0[2:0]=0x7) provide a high frequency PWM waveform generation option. The Fast PWM modes differ from the other PWM options by their single-slope operation. The counter counts from BOTTOM to TOP, then restarts from BOTTOM. TOP is defined as 0xFF when WGM0[2:0]=0x3. TOP is defined as OCR0A when WGM0[2:0]=0x7.

In non-inverting Compare Output mode, the Output Compare register (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the Fast PWM mode can be twice as high as the phase correct PWM modes, which use dual-slope operation. This high frequency makes the Fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In Fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the Fast PWM mode is shown below. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT0 slopes mark compare matches between OCR0x and TCNT0.

Figure 17-6. Fast PWM Mode, Timing Diagram



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

In Fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Writing the TCCR0A.COM0x[1:0] bits to 0x2 will produce a non-inverted PWM; TCCR0A.COM0x[1:0]=0x3 will produce an inverted PWM output. Writing the TCCR0A.COM0A[1:0] bits to 0x1 allows the OC0A pin to toggle on Compare Matches if the TCCRnB.WGMn2 bit is set. This option is not available for the OC0B pin. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

N represents the prescale divider (1, 8, 64, 256, or 1024).

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

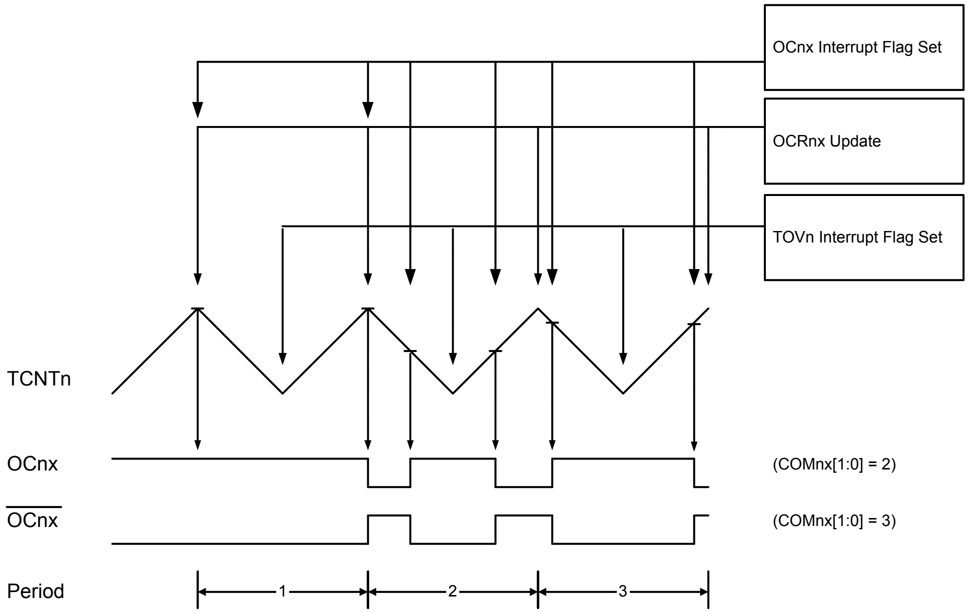
A frequency waveform output with 50% duty cycle can be achieved in Fast PWM mode by selecting OC0x to toggle its logical level on each compare match (COM0x[1:0]=0x1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A=0x00. This feature is similar to the OC0A toggle in CTC mode, except double buffering of the Output Compare unit is enabled in the Fast PWM mode.

17.7.4 Phase Correct PWM Mode

The Phase Correct PWM mode (WGM0[2:0]=0x1 or WGM0[2:0]=0x5) provides a high resolution, phase correct PWM waveform generation. The Phase Correct PWM mode is based on dual-slope operation: The counter counts repeatedly from BOTTOM to TOP, and then from TOP to BOTTOM. When WGM0[2:0]=0x1 TOP is defined as 0xFF. When WGM0[2:0]=0x5, TOP is defined as OCR0A. In non-inverting Compare Output mode, the Output Compare (OC0x) bit is cleared on compare match between TCNT0 and OCR0x while up-counting, and OC0x is set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has a lower maximum operation frequency than single slope operation. Due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In Phase Correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the Phase Correct PWM mode is shown below. The TCNT0 value is shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 17-7. Phase Correct PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

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

In Phase Correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pin. Writing the COM0x[1:0] bits to 0x2 will produce a non-inverted PWM. An inverted PWM output can be generated by writing COM0x[1:0]=0x3. Setting the Compare Match Output A Mode bit to '1' (TCCR0A.COM0A0) allows the OC0A pin to toggle on Compare Matches if the TCCR0B.WGM02 bit is set. This option is not available for the OC0B pin. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x Register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using Phase Correct PWM can be calculated by:

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

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

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the Phase Correct PWM mode: If the OCR0A register is written equal to BOTTOM, the output will be continuously low. If OCR0A is written to MAX, the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in the timing diagram above, OC0x has a transition from high to low even though there is no Compare Match. This transition serves to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

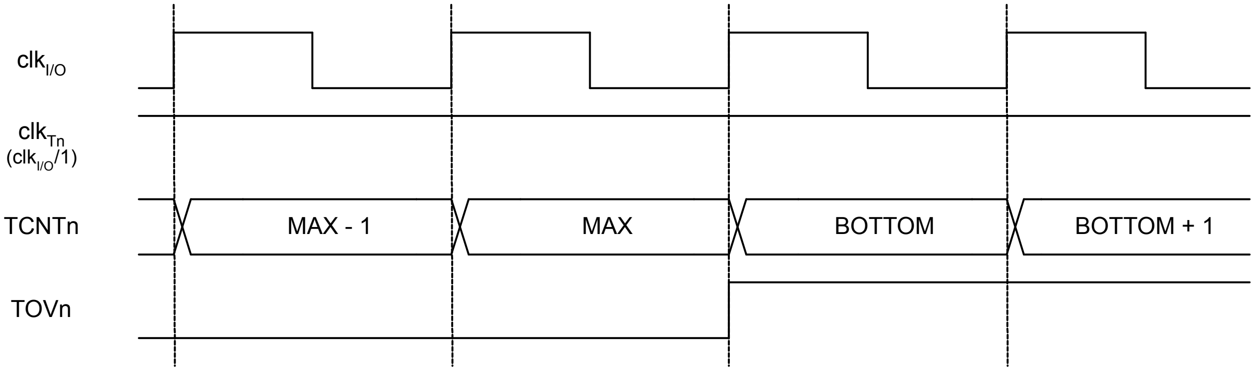
- OCR0x changes its value from MAX, as in the timing diagram. When the OCR0A value is MAX, the OC0 pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OC0x value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts up-counting from a value higher than the one in OCR0x, and for that reason misses the Compare Match and consequently, the OC0x does not undergo the change that would have happened on the way up.

17.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock ($\mathrm{clk_{T0}}$) is therefore shown as a clock enable signal in the following figures. If the given instance of the TC0 supports an asynchronous mode, $\mathrm{clk_{I/O}}$ should be replaced by the TC oscillator clock.

The figures include information on when interrupt flags are set. The first figure below illustrates timing data for basic Timer/Counter operation close to the MAX value in all modes other than Phase Correct PWM mode.

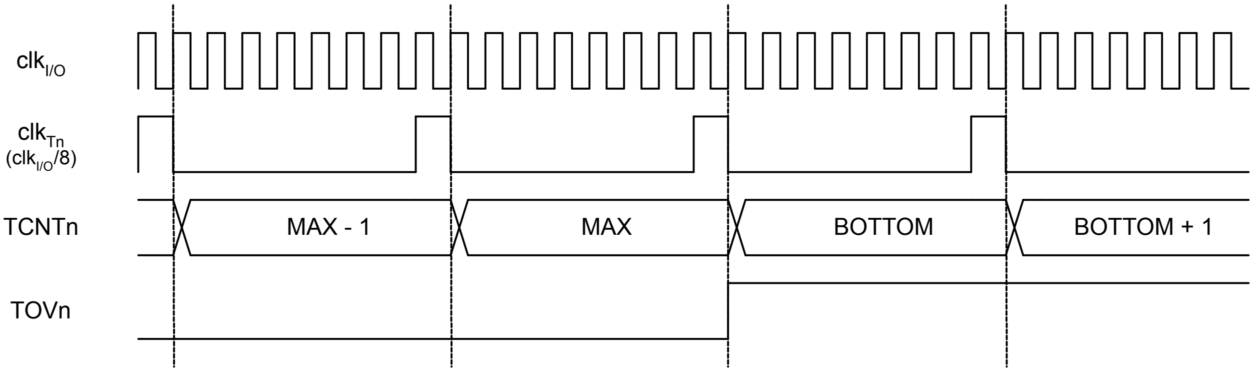
Figure 17-8. Timer/Counter Timing Diagram, no Prescaling



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The next figure shows the same timing data, but with the prescaler enabled.

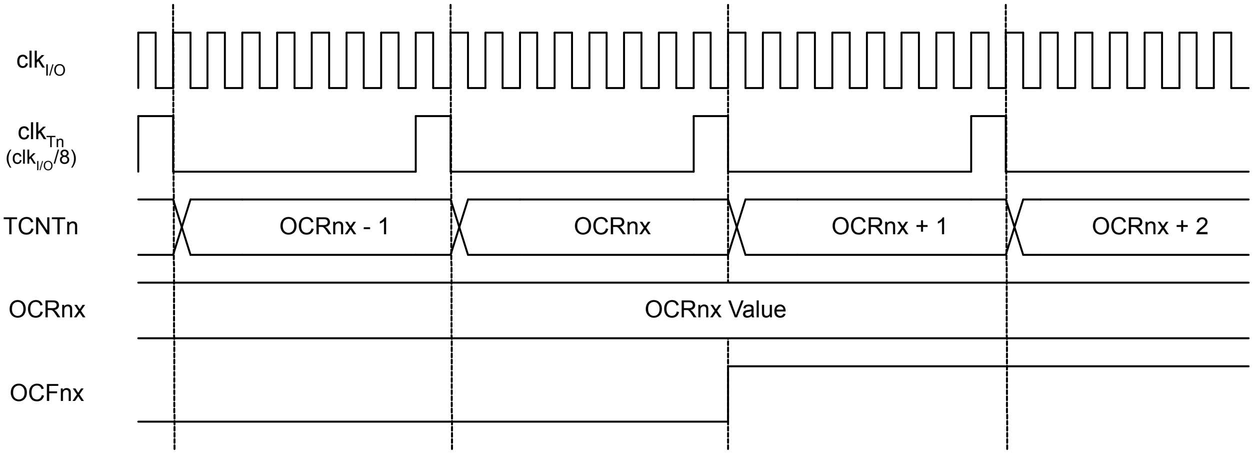
Figure 17-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



Note: The "n" in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the "x" indicates Output Compare unit (A/B).

The next figure shows the setting of OCF0B in all modes and OCF0A in all modes (except CTC mode and PWM mode where OCR0A is TOP).

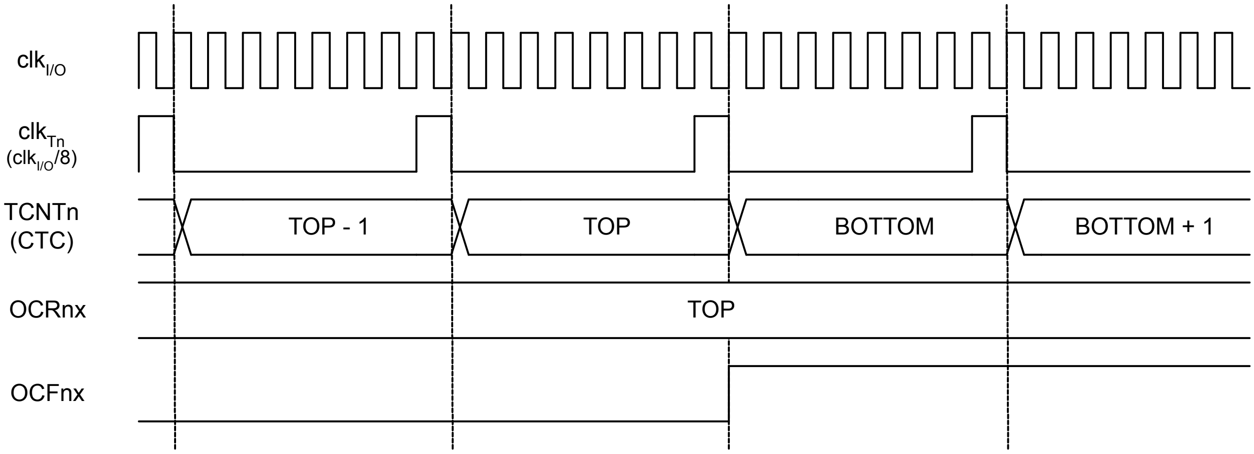
Figure 17-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)



Note: The "n" in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the "x" indicates Output Compare unit (A/B).

The next figure shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 17-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}$/8)



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

17.9 Register Description

17.9.1 TC0 Control Register A

Name: TCCR0A
Offset: 0x44
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x24

Bit	7	6	5	4	3	2	1	0
	COM0A[1:0]		COM0B [1:0]				WGM0[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 7:6 – COM0A[1:0]: Compare Output Mode for Channel A
These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A[1:0] bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

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

Table 17-3. Compare Output Mode, non-PWM

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match.

COM0A1	COM0A0	Description
1	0	Clear OC0A on Compare Match.
1	1	Set OC0A on Compare Match .

The table below shows the COM0A[1:0] bit functionality when the WGM0[1:0] bits are set to fast PWM mode.

Table 17-4. Compare Output Mode, Fast PWM[(1)]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

Note:

1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM0A[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

Table 17-5. Compare Output Mode, Phase Correct PWM Mode[(1)]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Note:

1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

Bits 5:4 – COM0B [1:0]: Compare Output Mode for Channel B

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

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

Table 17-6. Compare Output Mode, non-PWM

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on Compare Match.
1	0	Clear OC0B on Compare Match.
1	1	Set OC0B on Compare Match.

The table below shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to fast PWM mode.

Table 17-7. Compare Output Mode, Fast PWM(1)

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

Note:

1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Fast PWM Mode for details.

The table below shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

Table 17-8. Compare Output Mode, Phase Correct PWM Mode(1)

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

Note:

1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

Bits 1:0 – WGM0[1:0]: Waveform Generation Mode
Combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be

used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see Modes of Operation).

Table 17-9. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Note:

1. MAX = 0xFF
2. BOTTOM = 0x00

17.9.2 TC0 Control Register B

Name: TCCR0B
Offset: 0x45
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x25

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM0 [2]	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 7 – FOC0A: Force Output Compare A
The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

To ensure compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A[1:0] bits setting. The FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A[1:0] bits that determines the effect of the forced compare.

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

The FOC0A bit is always read as zero.

Bit 6 – FOC0B: Force Output Compare B
The FOC0B bit is only active when the WGM bits specify a non-PWM mode.

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

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

The FOC0B bit is always read as zero.

Bit 3 – WGM0 [2]: Waveform Generation Mode
Refer to TCCR0A.

Bits 2:0 – CS0[2:0]: Clock Select 0
The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 17-10. Clock Select Bit Description

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

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

17.9.3 TC0 Interrupt Mask Register

Name: TIMSK0
Offset: 0x6E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						OCIE0B	OCIE0A	TOIE0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCIE0B: Timer/Counter0, Output Compare B Match Interrupt Enable
When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in TIFR0.

Bit 1 – OCIE0A: Timer/Counter0, Output Compare A Match Interrupt Enable
When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in TIFR0.

Bit 0 – TOIE0: Timer/Counter0, Overflow Interrupt Enable
When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in TIFR0.

17.9.4 General Timer/Counter Control Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GTCCR
Offset: 0x43
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x23

Bit	7	6	5	4	3	2	1	0
	TSM						PSRASY	PSRSYNC
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – TSM: Timer/Counter Synchronization Mode
Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

Bit 1 – PSRASY: Prescaler Reset Timer/Counter2
When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

Bit 0 – PSRSYNC: Prescaler Reset
When this bit is one, Timer/Counter 0, 1, 3, 4 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter 0, 1, 3, 4 share the same prescaler and a reset of this prescaler will affect the mentioned timers.

17.9.5 TC0 Counter Value Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TCNT0
Offset: 0x46
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x26

Bit	7	6	5	4	3	2	1	0
	TCNT0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TCNT0[7:0]: TC0 Counter Value
The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

17.9.6 TC0 Output Compare Register A

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: OCR0A
Offset: 0x47
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x27

Bit	7	6	5	4	3	2	1	0
	OCR0A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR0A[7:0]: Output Compare 0 A
The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

17.9.7 TC0 Output Compare Register B

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: OCR0B
Offset: 0x48
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x28

Bit	7	6	5	4	3	2	1	0
	OCR0B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR0B[7:0]: Output Compare 0 B
The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0B pin.

17.9.8 TC0 Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TIFR0
Offset: 0x35
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x15

Bit	7	6	5	4	3	2	1	0
						OCF0B	OCF0A	TOV0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCF0B: Timer/Counter 0, Output Compare B Match Flag
The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

Bit 1 – OCF0A: Timer/Counter 0, Output Compare A Match Flag
The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

Bit 0 – TOV0 : Timer/Counter 0, Overflow Flag
The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter 0 Overflow interrupt is executed.

The setting of this flag is dependent of the WGM0[2:0] bit setting. Refer to Table 17-9.

18. TC1, 3, 4 - 16-bit Timer/Counter1, 3, 4 with PWM

18.1 Features

- Three 16-bit Timer/Counter instances TC1, TC3, TC4.
- True 16-bit Design (i.e., allows 16-bit PWM)
- Two independent Output Compare Units
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Independent interrupt Sources (TOV, OCFA, OCFB, and ICF)

18.2 Overview

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

A block diagram of the 16-bit Timer/Counter is shown below. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in Register Description. For the actual placement of I/O pins, refer to the *Pin Configurations* description.

Related Links

I/O-Ports

Pin Configurations

18.2.1 Definitions

Many register and bit references in this section are written in general form:

- n=1,3,4 represents the Timer/Counter number
- x=A,B represents the Output Compare Unit A or B

However, when using the register or bit definitions in a program, the precise form must be used, i.e., TCNT3 for accessing Timer/Counter 3 counter value.

The following definitions are used extensively throughout the section:

Table 18-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x0 for 8-bit counters, or 0x00 for 16-bit counters).
MAX	The counter reaches its MAXimum when it becomes 0xF (decimal 15, for 8-bit counters) or 0xFF (decimal 255, for 16-bit counters).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value MAX or the value stored in the OCRnA Register. The assignment is dependent on the mode of operation.

18.2.2 Registers

The Timer/Counter (TCNTn), Output Compare Registers (OCRA/B), and Input Capture Register (ICRn) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in section Accessing 16-bit Registers.

The Timer/Counter Control Registers (TCCRnA/B/C) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the block diagram) signals are all visible in the Timer Interrupt Flag Register (TIFRn). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSKn). TIFRn and TIMSKn are not shown in the figure.

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

The double buffered Output Compare Registers (OCRnA/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OCnA/B). See Output Compare Units. The compare match event will also set the Compare Match Flag (OCFnA/B) which can be used to generate an Output Compare interrupt request.

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

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

18.3 Accessing 16-bit Timer/Counter Registers

The TCNTn, OCRnA/B, and ICRn are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be accessed byte-wise, using two read or write operations. Each 16-bit timer has a single 8-bit TEMP register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer.

Accessing the low byte triggers the 16-bit read or write operation: When the low byte of a 16-bit register is written by the CPU, the high byte that is currently stored in TEMP and the low byte being written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the TEMP register in the same clock cycle as the low byte is read, and must be read subsequently.

Note: To perform a 16-bit write operation, the low byte must be written before the high byte. For a 16-bit read, the low byte must be read before the high byte.

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

16-bit Access

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

Assembly Code Example[1]

```
   ...
; Set TCNTn to 0x01FF
ldi    r17,0x01
ldi    r16,0xFF
out    TCNTnH,r17
out    TCNTnL,r16
; Read TCNTn into r17:r16
in    r16,TCNTnL
in    r17,TCNTnH
   ...
```

The assembly code example returns the TCNT**n** value in the r17:r16 register pair.

C Code Example[1]

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

Note:

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

Atomic Read

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

The following code examples show how to perform an atomic read of the TCNTn Register contents. The OCRnA/B or ICRn Registers can be ready by using the same principle.

Assembly Code Example[1]

```
TIM16_ReadTCNTn:
   ; Save global interrupt flag
   in    r18,SREG
```

```
    ; Disable interrupts
    cli
    ; Read TCNTn into r17:r16
    in     r16,TCNTnL
    in     r17,TCNTnH
    ; Restore global interrupt flag
    out    SREG,r18
    ret
```

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

C Code Example(1)

```
unsigned int TIM16_ReadTCNTn( void )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Read TCNTn into i */
    i = TCNTn;
    /* Restore global interrupt flag */
    SREG = sreg;
    return i;
}
```

Note:

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

Atomic Write

The following code examples show how to do an atomic write of the TCNTn Register contents. Writing any of the OCRnA/B or ICRn Registers can be done by using the same principle.

Assembly Code Example(1)

```
TIM16_WriteTCNTn:
    ; Save global interrupt flag
    in      r18,SREG
    ; Disable interrupts
    cli
    ; Set TCNTn to r17:r16
    out     TCNTnH,r17
    out     TCNTnL,r16
    ; Restore global interrupt flag
    out     SREG,r18
    ret
```

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

C Code Example(1)

```
void TIM16_WriteTCNTn( unsigned int i )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Set TCNTn to i */
    TCNTn = i;
```

```
    /* Restore global interrupt flag */
    SREG = sreg;
}
```

Note:

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

Related Links

About Code Examples

18.3.1 Reusing the Temporary High Byte Register

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

18.4 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select bits in the Timer/Counter control Register B (TCCRnB.CS[2:0]).

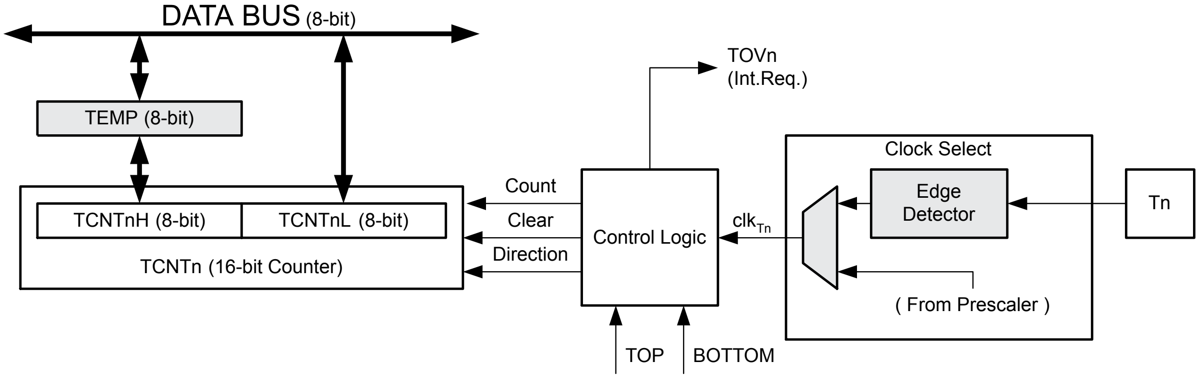
Related Links

Timer/Counter 0, 1, 3, 4 Prescalers

18.5 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit, as shown in the block diagram:

Figure 18-1. Counter Unit Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

Table 18-2. Signal description (internal signals)

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

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNTnH) containing the upper eight bits of the counter, and Counter Low (TCNTnL) containing the lower eight bits. The TCNTnH Register can only be accessed indirectly by the CPU. When the CPU does an access to the TCNTnH I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNTnH value when the TCNTnL is read, and TCNTnH is updated with the temporary register value when TCNTnL is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus.

Note: That there are special cases when writing to the TCNTn Register while the counter is counting will give unpredictable results. These special cases are described in the sections where they are of importance.

Depending on the selected mode of operation, the counter is cleared, incremented, or decremented at each timer clock (clk_{Tn}). The clock clk_{Tn} can be generated from an external or internal clock source, as selected by the Clock Select bits in the Timer/Countern Control Register B (TCCRnB.CS[2:0]). When no clock source is selected (CS[2:0]=0x0) the timer is stopped. However, the TCNTn value can be accessed by the CPU, independent of whether clk_{Tn} is present or not. A CPU write overrides (i.e., has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the Waveform Generation mode bits in the Timer/Counter Control Registers A and B (TCCRnB.WGMn[3:2] and TCCRnA.WGMn[1:0]). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0x. For more details about advanced counting sequences and waveform generation, see Modes of Operation.

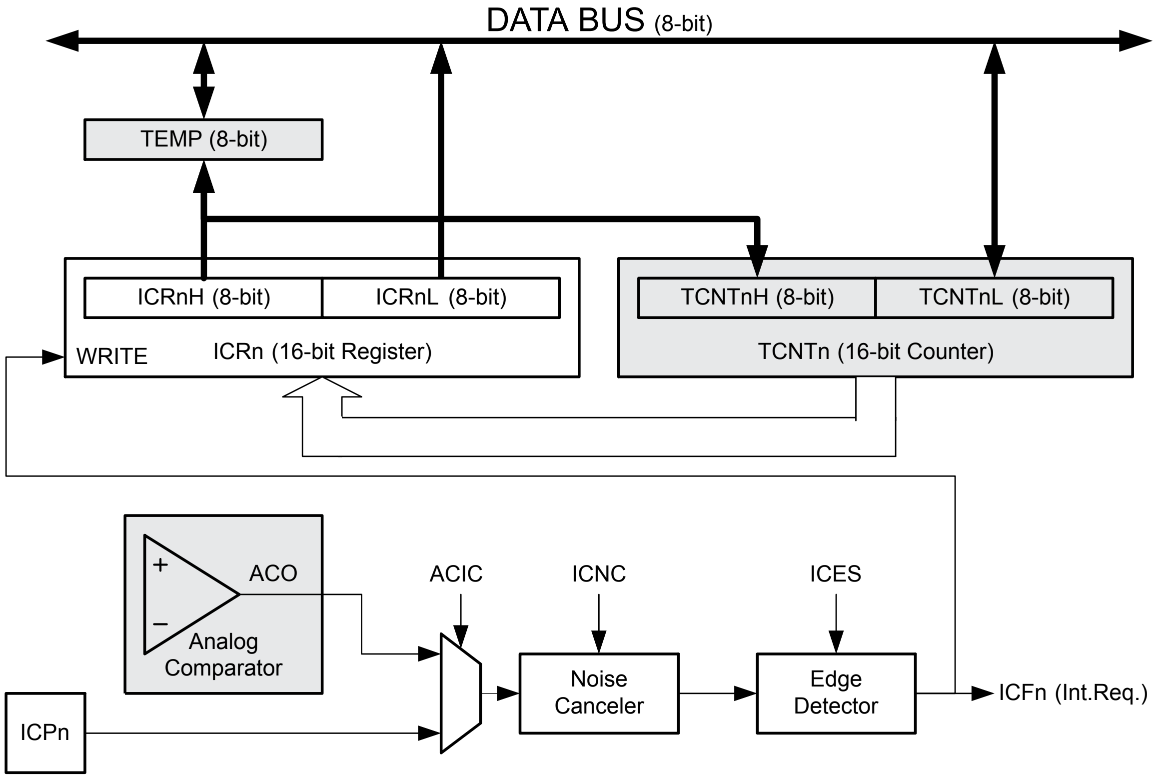
The Timer/Counter Overflow Flag in the TCn Interrupt Flag Register (TIFRn.TOV) is set according to the mode of operation selected by the WGMn[3:0] bits. TOV can be used for generating a CPU interrupt.

18.6 Input Capture Unit

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

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

Figure 18-2. Input Capture Unit Block Diagram for TCn



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

When a change of the logic level (an event) occurs on the Input Capture pin (ICPn), or alternatively on the Analog Comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered: the 16-bit value of the counter (TCNTn) is written to the Input Capture Register (ICRn). The Input Capture Flag (ICF) is set at the same system clock cycle as the TCNTn value is copied into the ICRn Register. If enabled (TIMSKn.ICIE=1), the Input Capture Flag generates an Input Capture interrupt. The ICFn Flag is automatically cleared when the interrupt is executed. Alternatively the ICF Flag can be cleared by software by writing '1' to its I/O bit location.

Reading the 16-bit value in the Input Capture Register (ICRn) is done by first reading the low byte (ICRnL) and then the high byte (ICRnH). When the low byte is read form ICRnL, the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICRnH I/O location it will access the TEMP Register.

The ICRn Register can only be written when using a Waveform Generation mode that utilizes the ICRn Register for defining the counter’s TOP value. In these cases the Waveform Generation mode bits (WGMn[3:0]) must be set before the TOP value can be written to the ICRn Register. When writing the ICRn Register, the high byte must be written to the ICRnH I/O location before the low byte is written to ICRnL.

See also Accessing 16-bit Timer/Counter Registers.

18.6.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the Input Capture pin (ICPn). Timer/Countern can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the Analog Comparator Input Capture (ACIC) bit in the Analog Comparator Control and Status Register (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

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

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

Related Links

Timer/Counter 0, 1, 3, 4 Prescalers

18.6.2 Noise Canceler

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

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

18.6.3 Using the Input Capture Unit

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

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

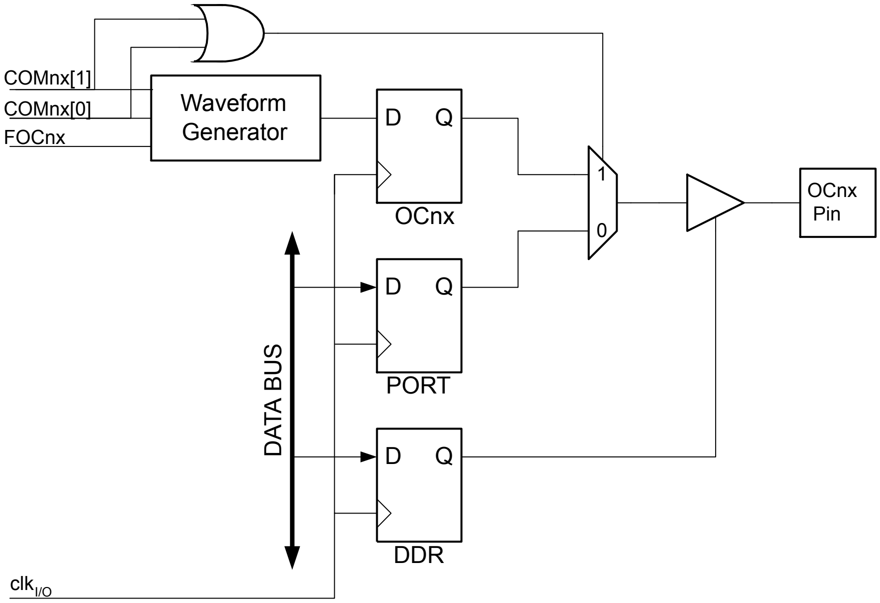
Using the Input Capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

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

18.7 Compare Match Output Unit

The Compare Output mode (TCCRnA.COMnx[1:0]) bits have two functions. The Waveform Generator uses the TCCRnA.COMnx[1:0] bits for defining the Output Compare (OCnx) state at the next compare match. Secondly the TCCRnA.COMnx[1:0] bits control the OCnx pin output source. The figure below shows a simplified schematic of the logic affected by the TCCRnA.COMnx[1:0] bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the TCCRnA.COMnx[1:0] bits are shown. When referring to the OCnx state, the reference is for the internal OCnx Register, not the OCnx pin. If a system reset occur, the OCnx Register is reset to “0”.

Figure 18-3. Compare Match Output Unit, Schematic



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OCnx) from the Waveform Generator if either of the TCCRnA.COMnx[1:0] bits are set. However, the OCnx pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OCnx pin (DDR_OCnx) must be set as output before the OCnx value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions.

The design of the Output Compare pin logic allows initialization of the OCnx state before the output is enabled. Note that some TCCRnA.COMnx[1:0] bit settings are reserved for certain modes of operation.

The TCCRnA.COMnx[1:0] bits have no effect on the Input Capture unit.

18.7.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the TCCRnA.COMnx[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the TCCRnA.COMnx[1:0] = 0 tells the Waveform Generator that no action on the OCnx Register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the TCCRnA.COMnx[1:0] bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the TCCRnC.FOCnx strobe bits.

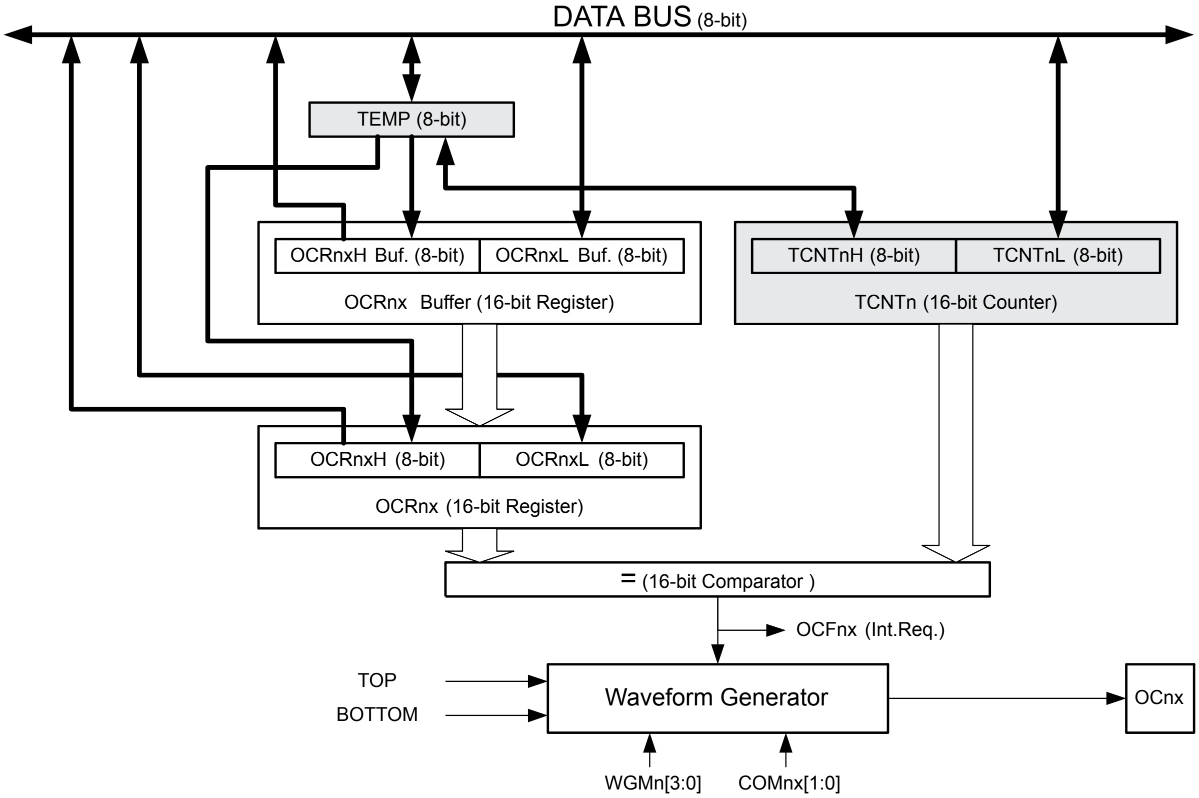
18.8 Output Compare Units

The 16-bit comparator continuously compares TCNTn with the Output Compare Register (OCRnx). If TCNT equals OCRnx the comparator signals a match. A match will set the Output Compare Flag (TIFRn.OCFx) at the next timer clock cycle. If enabled (TIMSKn.OCIEx = 1), the Output Compare Flag generates an Output Compare interrupt. The OCFx Flag is automatically cleared when the interrupt is executed. Alternatively the OCFx Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the Waveform Generation mode (WGMn[3:0]) bits and Compare Output mode (COMnx[1:0]) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation, see Modes of Operation.

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

Below is a block diagram of the Output Compare unit. The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

Figure 18-4. Output Compare Unit, Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

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

When double buffering is enabled, the CPU has access to the OCRnx Buffer Register. When double buffering is disabled, the CPU will access the OCRnx directly.

The content of the OCRnx (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNTn and ICRn Register). Therefore OCRnx is not read via the high byte temporary register (TEMP). However, it is good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCRnx Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCRnxH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the low byte (OCRnxL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCRnx buffer or OCRnx Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to Accessing 16-bit Timer/Counter Registers.

18.9 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGMn[3:0]) and Compare Output mode (TCCRnA.COMnx[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The TCCRnA.COMnx[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the TCCRnA.COMnx[1:0] bits control whether the output should be set, cleared, or toggle at a compare match.

Related Links

Timer/Counter Timing Diagrams

Compare Match Output Unit

18.9.1 Normal Mode

The simplest mode of operation is the Normal mode (TCCRnA.WGMn[3:0]=0x0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX=0xFFFF) and then restarts from BOTTOM=0x0000. In normal operation the Timer/Counter Overflow Flag (TIFRn.TOV) will be set in the same timer clock cycle as the TCNTn becomes zero. In this case, the TOV Flag in behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

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

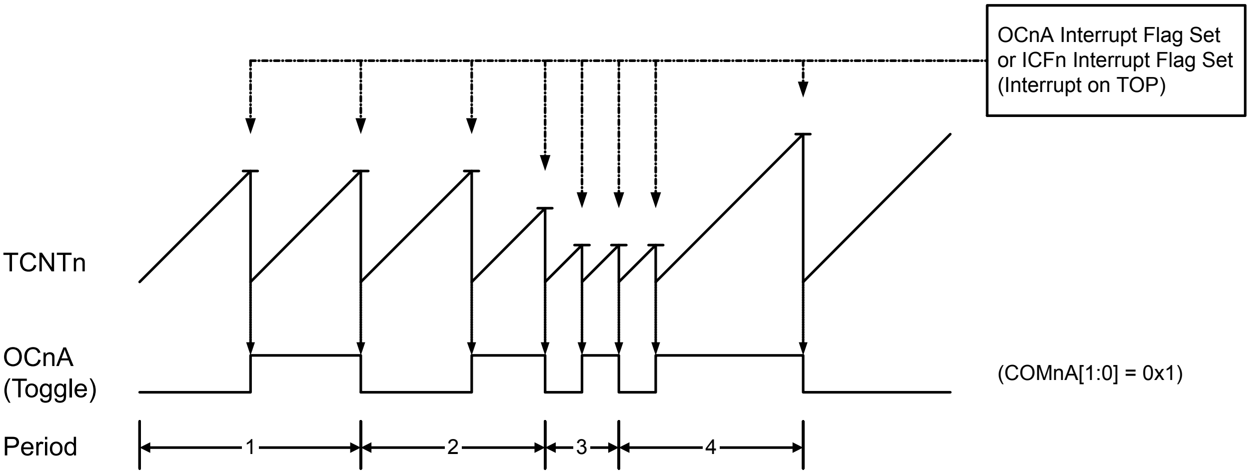
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

18.9.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC modes (mode 4 or 12, WGMn[3:0]=0x4 or 0xC), the OCRnA or ICRn registers are used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNTn) matches either the OCRnA (if WGMn[3:0]=0x4) or the ICRn (WGMn[3:0]=0xC). The OCRnA or ICRn define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown below. The counter value (TCNTn) increases until a compare match occurs with either OCRnA or ICRn, and then TCNTn is cleared.

Figure 18-5. CTC Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCFnA or ICFn Flag, depending on the actual CTC mode. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value.

Note: Changing TOP to a value close to BOTTOM while the counter is running must be done with care, since the CTC mode does not provide double buffering. If the new value written to OCRnA is lower than the current value of TCNTn, the counter will miss the compare match. The counter will then count to its maximum value (0xFF for a 8-bit counter, 0xFFFF for a 16-bit counter) and wrap around starting at 0x00 before the compare match will occur.
In many cases this feature is not desirable. An alternative will then be to use the Fast PWM mode using OCRnA for defining TOP (WGMn[3:0]=0xF), since the OCRnA then will be double buffered.

For generating a waveform output in CTC mode, the OCnA output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COMnA[1:0]=0x1). The OCnA value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OCnA=1). The waveform generated will have a maximum frequency of $f_{\mathrm{OCnA}} = f_{\mathrm{clk\_I/O}}/2$ when OCRnA is set to ZERO (0x0000). The waveform frequency is defined by the following equation:

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

Note:

- The “n” indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).
- *N* represents the prescaler factor (1, 8, 64, 256, or 1024).

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

18.9.3 Fast PWM Mode

The Fast Pulse Width Modulation or Fast PWM modes (modes 5, 6, 7, 14,and 15, WGMn[3:0]= 0x5, 0x6, 0x7, 0xE, 0xF) provide a high frequency PWM waveform generation option. The Fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM.

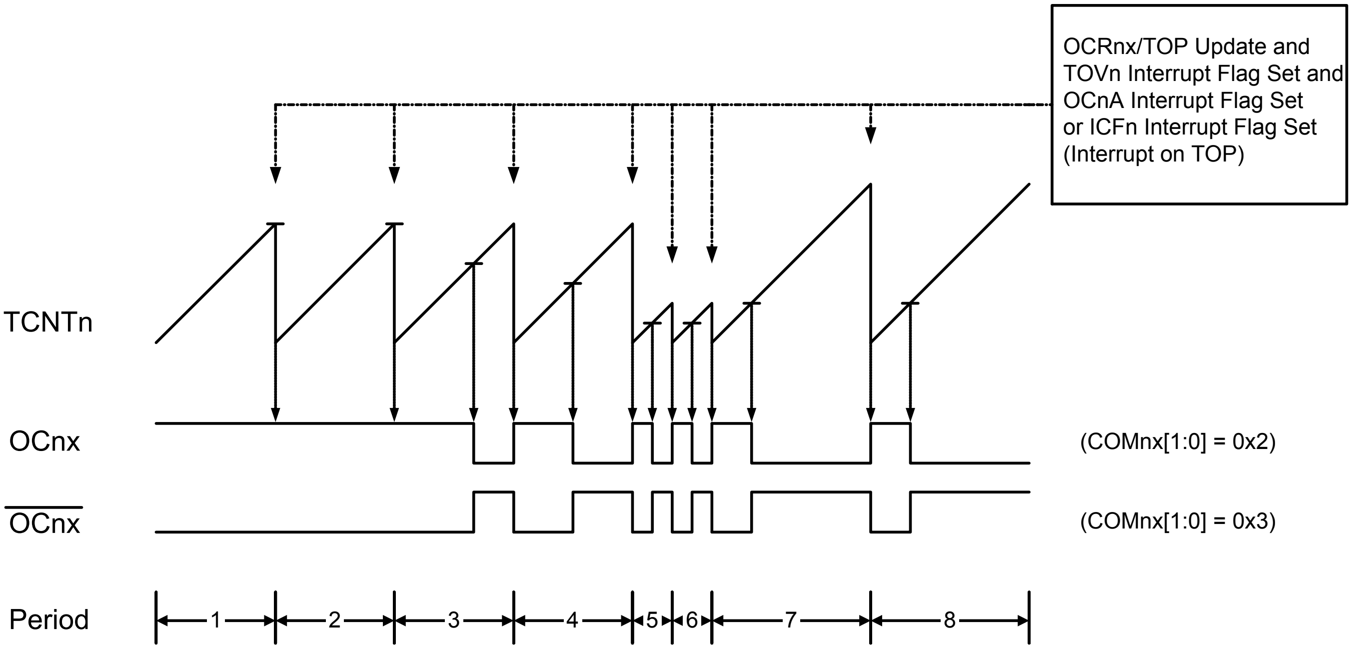
In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the compare match between TCNTn and OCRnx, and set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the Fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the Fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

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

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

In Fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGMn[3:0] = 0x5, 0x6, or 0x7), the value in ICRn (WGMn[3:0]=0xE), or the value in OCRnA (WGMn[3:0]=0xF). The counter is then cleared at the following timer clock cycle. The timing diagram for the Fast PWM mode using OCRnA or ICRn to define TOP is shown below. The TCNTn value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNTn slopes mark compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 18-6. Fast PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

The Timer/Counter Overflow Flag (TOVn) is set each time the counter reaches TOP. In addition, when either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag is set at the same timer clock cycle TOVn is set. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNTn and the OCRnx. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCRnx Registers are written.

The procedure for updating ICRn differs from updating OCRnA when used for defining the TOP value. The ICRn Register is not double buffered. This means that if ICRn is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICRn value written is lower than the current value of TCNTn. As result, the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCRnA Register however, is double buffered. This feature allows the OCRnA I/O location to be written anytime. When the OCRnA I/O location is written the value written will be put into the OCRnA Buffer Register. The OCRnA Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNTn matches TOP. The update is done at the same timer clock cycle as the TCNTn is cleared and the TOVn Flag is set.

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

In Fast PWM mode, the compare units allow generation of PWM waveforms on the OCnx pins. Writing the COMnx[1:0] bits to 0x2 will produce an inverted PWM and a non-inverted PWM output can be generated by writing the COMnx[1:0] to 0x3. The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn, and clearing (or setting) the OCnx Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

Note:

- The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).
- *N* represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCRnx registers represents special cases when generating a PWM waveform output in the Fast PWM mode. If the OCRnx is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCRnx equal to TOP will result in a constant high or low output (depending on the polarity of the output which is controlled by COMnx[1:0]).

A frequency waveform output with 50% duty cycle can be achieved in Fast PWM mode by selecting OCnA to toggle its logical level on each compare match (COMnA[1:0]=0x1). This applies only if OCRnA is used to define the TOP value (WGMn[3:0]=0xF). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk\_I/O}/2$ when OCRnA is set to zero (0x0000). This feature is similar to the OCnA toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the Fast PWM mode.

18.9.4 Phase Correct PWM Mode

The Phase Correct Pulse Width Modulation or Phase Correct PWM modes (WGMn[3:0]= 0x1, 0x2, 0x3, 0xA, and 0xB) provide a high resolution, phase correct PWM waveform generation option. The Phase

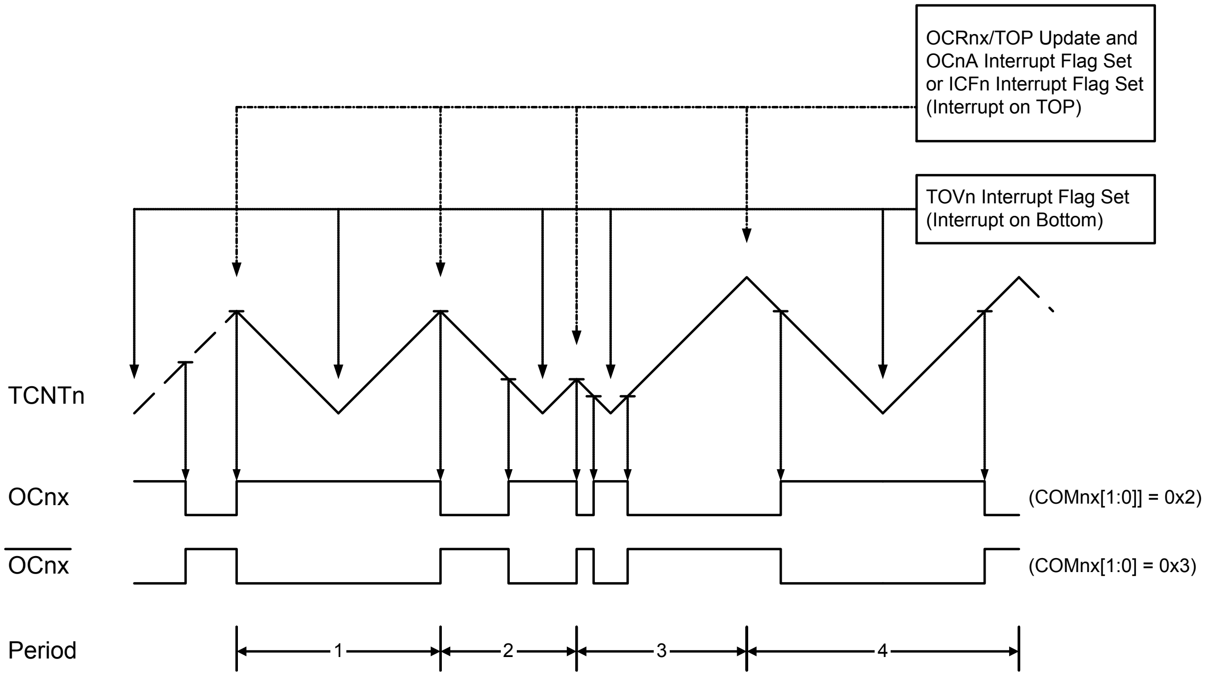
Correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the compare match between TCNTn and OCRnx while up-counting, and set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

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

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

In Phase Correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGMn[3:0]= 0x1, 0x2, or 0x3), the value in ICRn (WGMn[3:0]=0xA), or the value in OCRnA (WGMn[3:0]=0xB). The counter has then reached the TOP and changes the count direction. The TCNTn value will be equal to TOP for one timer clock cycle. The timing diagram for the Phase Correct PWM mode is shown below, using OCRnA or ICRn to define TOP. The TCNTn value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNTn slopes mark compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 18-7. Phase Correct PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

The Timer/Counter Overflow Flag (TOVn) is set each time the counter reaches BOTTOM. When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag is set accordingly at the same timer clock cycle as the OCRnx Registers are updated with the double buffer value (at TOP). The

Interrupt Flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNTn and the OCRnx. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCRnx registers is written. As illustrated by the third period in the timing diagram, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCRnx Register. Since the OCRnx update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

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

In Phase Correct PWM mode, the compare units allow generation of PWM waveforms on the OCnx pins. Writing COMnx[1:0] bits to 0x2 will produce a non-inverted PWM. An inverted PWM output can be generated by writing the COMnx[1:0] to 0x3. The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx Register at compare match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using Phase Correct PWM can be calculated by the following equation:

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

N represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCRnx Register represent special cases when generating a PWM waveform output in the Phase Correct PWM mode. If the OCRnx is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCRnA is used to define the TOP value (WGMn[3:0]=0xB) and COMnA[1:0]=0x1, the OCnA output will toggle with a 50% duty cycle.

18.9.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation, or phase and frequency correct PWM mode (WGMn[3:0] = 0x8 or 0x9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the compare match between TCNTn and OCRnx while up-counting, and set on the compare match while down-counting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

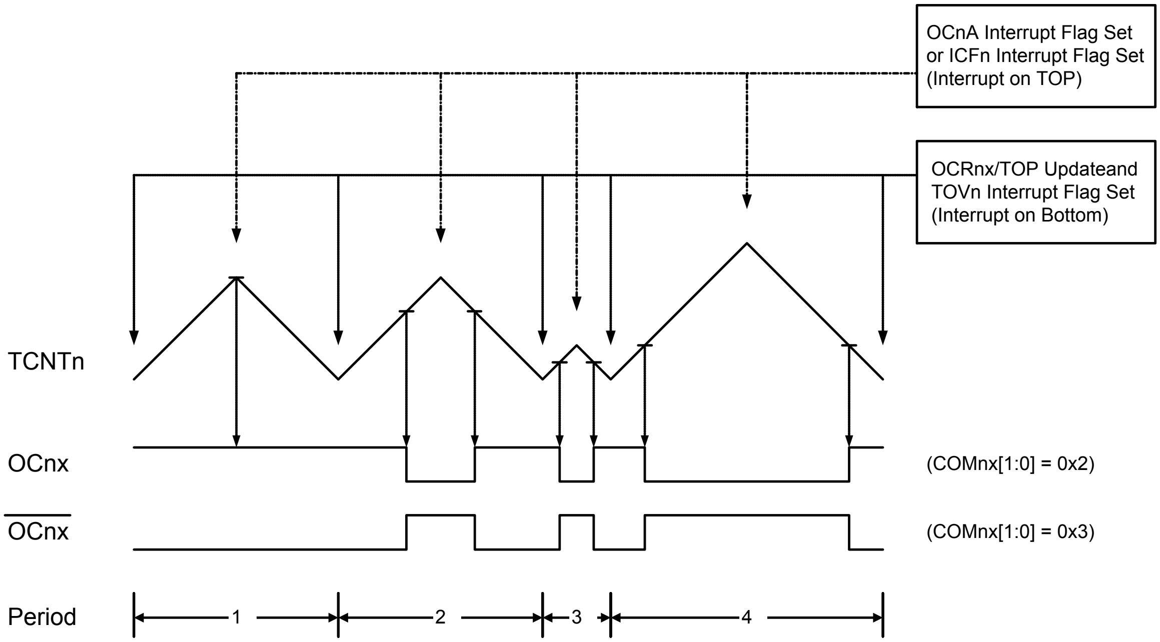
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCRnx Register is updated by the OCRnx Buffer Register, (see Figure 18-7 and the Timing Diagram below).

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

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

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICRn (WGMn[3:0]=0x8), or the value in OCRnA (WGMn[3:0]=0x9). The counter has then reached the TOP and changes the count direction. The TCNTn value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown below. The figure shows phase and frequency correct PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 18-8. Phase and Frequency Correct PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

The Timer/Counter Overflow Flag (TOVn) is set at the same timer clock cycle as the OCRnx Registers are updated with the double buffer value (at BOTTOM). When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag set when TCNTn has reached TOP. The Interrupt Flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

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

As shown in the timing diagram above, the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCRnx Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

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

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OCnx pins. Setting the COMnx[1:0] bits to 0x2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx[1:0] to 0x3 (See description of TCCRA.COMnx). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx Register at compare match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

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

Note:

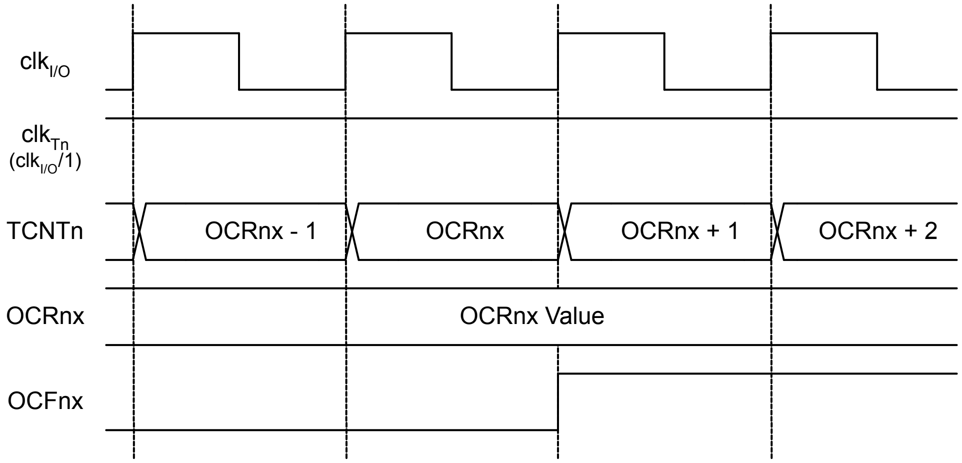
- The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).
- *N* represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCRnx Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRnx is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCRnA is used to define the TOP value (WGMn[3:0]=0x9) and COMnA[1:0]=0x1, the OCnA output will toggle with a 50% duty cycle.

18.10 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock ($\mathrm{clk}_{\mathrm{Tn}}$) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCRnx Register is updated with the OCRnx buffer value (only for modes utilizing double buffering). The first figure shows a timing diagram for the setting of OCFnx.

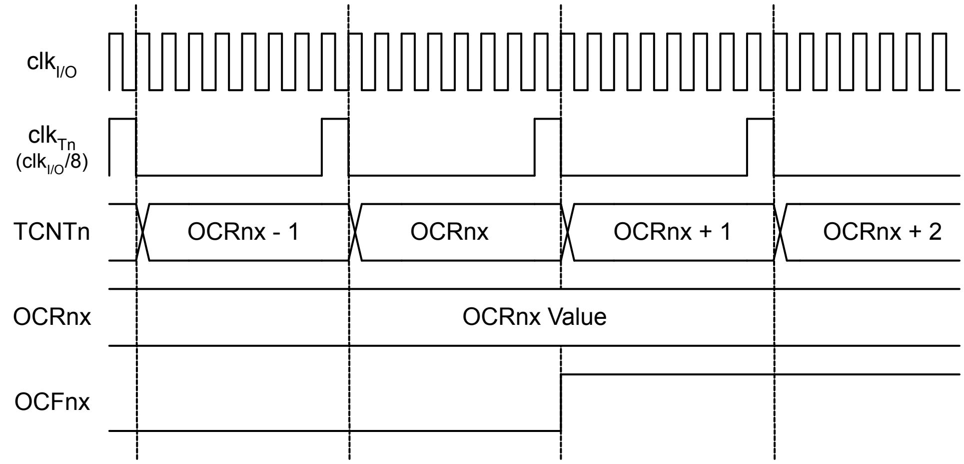
Figure 18-9. Timer/Counter Timing Diagram, Setting of OCFnx, no Prescaling



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

The next figure shows the same timing data, but with the prescaler enabled.

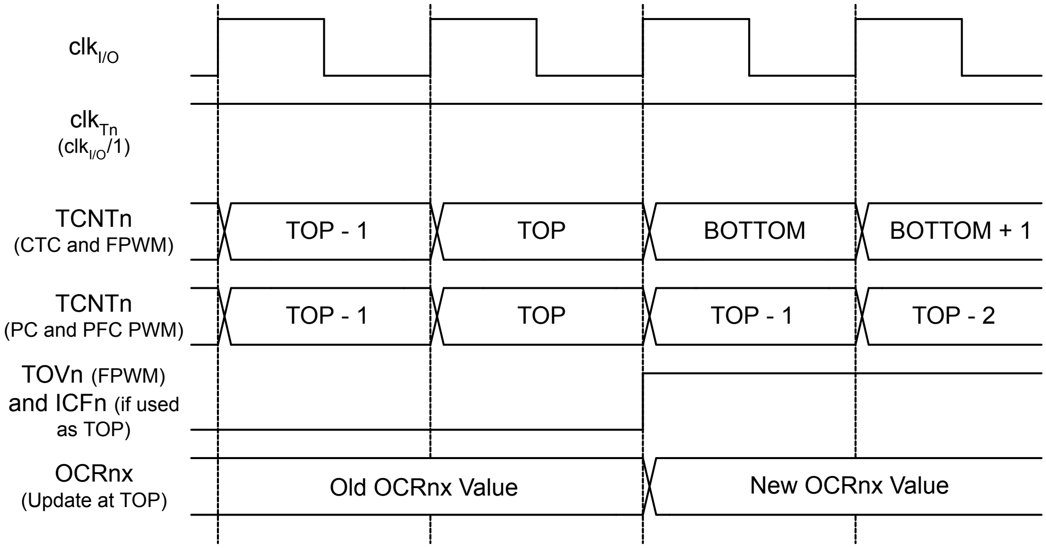
Figure 18-10. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk\_I/O}/8$)



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

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

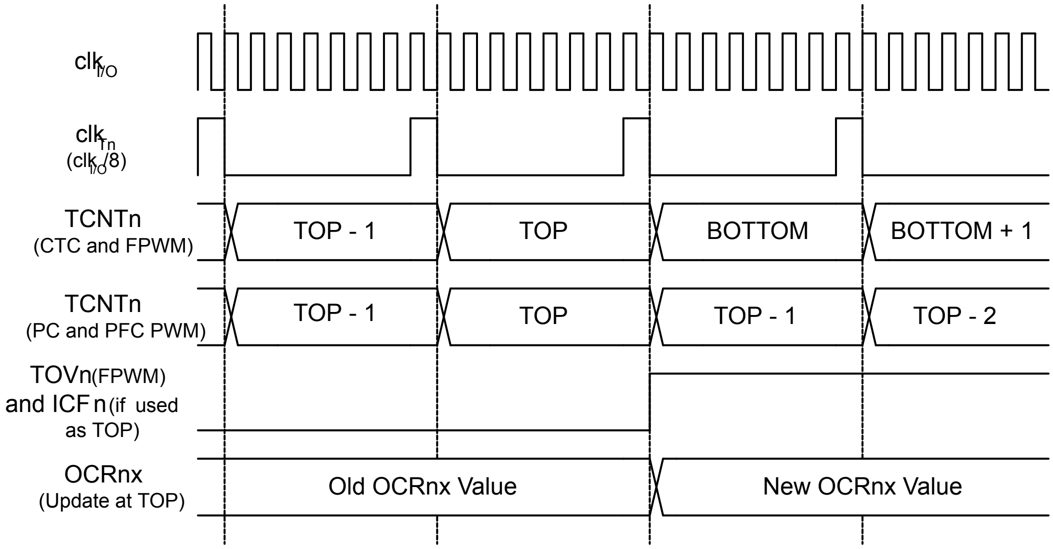
Figure 18-11. Timer/Counter Timing Diagram, no Prescaling.



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

The next figure shows the same timing data, but with the prescaler enabled.

Figure 18-12. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



Note: The “n” in the register and bit names indicates the device number (n = 1, 3, 4 for Timer/Counter 1, 3, 4), and the “x” indicates Output Compare unit (A/B).

18.11 Register Description

18.11.1 TC1 Control Register A

Name: TCCR1A
Offset: 0x80
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	COM1A[1:0]		COM1B[1:0]				WGM1[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 4:5, 6:7 – COM1B, COM1A: Compare Output Mode for Channel
The COM1A[1:0] and COM1B[1:0] control the Output Compare pins (OC1A and OC1B respectively) behavior. If one or both of the COM1A[1:0] bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B[1:0] bit are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1x[1:0] bits is dependent of the WGM1[3:0] bits setting. The table below shows the COM1x[1:0] bit functionality when the WGM1[3:0] bits are set to a Normal or a CTC mode (non-PWM).

Table 18-3. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match.
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
1	1	Set OC1A/OC1B on Compare Match (Set output to high level).

The table below shows the COM1x[1:0] bit functionality when the WGM1[3:0] bits are set to the fast PWM mode.

Table 18-4. Compare Output Mode, Fast PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

Note:

1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM1x[1:0] bit functionality when the WGM1[3:0] bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 18-5. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 9 or 11: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when down-counting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when down-counting.

Note:

1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. Refer to Phase Correct PWM Mode for details.

Bits 1:0 – WGM1[1:0]: Waveform Generation Mode
Combined with the WGM1[3:2] bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See Modes of Operation).

Table 18-6. Waveform Generation Mode Bit Description

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

Note:

1. The CTC1 and PWM1[1:0] bit definition names are obsolete. Use the WGM1[3:0] definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

18.11.2 TC1 Control Register B

Name: TCCR1B
Offset: 0x81
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM1[3]	WGM1[2]	CS1[2:0]		
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit 7 – ICNC1: Input Capture Noise Canceler

Writing this bit to '1' activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

Bit 6 – ICES1: Input Capture Edge Select

This bit selects which edge on the Input Capture pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to '1', a rising (positive) edge will trigger the capture.

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

When the ICR1 is used as TOP value (see description of the WGM1[3:0] bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

Bits 3, 4 – WGM1[2], WGM1[3]: Waveform Generation Mode

Refer to TCCR1A.

Bits 2:0 – CS1[2:0]: Clock Select 1

The three Clock Select bits select the clock source to be used by the Timer/Counter. Refer to Figure 18-9 and Figure 18-10.

Table 18-7. Clock Select Bit Description

CS1[2]	CS1[1]	CS1[0]	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

18.11.3 TC1 Control Register C

Name: TCCR1C
Offset: 0x82
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	FOC1A	FOC1B						
Access	R/W	R/W						
Reset	0	0						

Bits 6, 7 – FOC1B, FOC1A: Force Output Compare for Channel B and A
The FOC1A/FOC1B bits are only active when the WGM1[3:0] bits specifies a non-PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x[1:0] bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x[1:0] bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR1A as TOP. The FOC1A/FOC1B bits are always read as zero.

18.11.4 TC1 Counter Value Low and High byte

The TCNT1L and TCNT1H register pair represents the 16-bit value, TCNT1.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: TCNT1L and TCNT1H
Offset: 0x84
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	TCNT1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCNT1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – TCNT1[15:0]: Timer/Counter 1 Counter Value
The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

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

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

18.11.5 Input Capture Register 1 Low and High byte

The ICR1L and ICR1H register pair represents the 16-bit value, ICR1.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: ICR1L and ICR1H
Offset: 0x86
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	ICR1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ICR1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – ICR1[15:0]: Input Capture 1
The Input Capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the Analog Comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.6 Output Compare Register 1 A Low and High byte

The OCR1AL and OCR1AH register pair represents the 16-bit value, OCR1A.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: OCR1AL and OCR1AH
Offset: 0x88
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	OCR1A[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCR1A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – OCR1A[15:0]: Output Compare 1 A
The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.7 Output Compare Register 1 B Low and High byte

The OCR1BL and OCR1BH register pair represents the 16-bit value, OCR1B.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: OCR1BL and OCR1BH
Offset: 0x8A
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	OCR1B[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCR1B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – OCR1B[15:0]: Output Compare 1 B
The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1B pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.8 TC3 Control Register A

Name: TCCR3A
Offset: 0x90
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	COM3A[1:0]		COM3B[1:0]				WGM3[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 4:5, 6:7 – COM3B, COM3A: Compare Output Mode for Channel
The COM3A[1:0] and COM3B[1:0] control the Output Compare pins (OC3A and OC3B respectively) behavior. If one or both of the COM3A[1:0] bits are written to one, the OC3A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM3B[1:0] bit are written to one, the OC3B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC3A or OC3B pin must be set in order to enable the output driver.

When the OC3A or OC3B is connected to the pin, the function of the COM3n[1:0] bits is dependent of the WGM3[3:0] bits setting. The table below shows the COM3n[1:0] bit functionality when the WGM3[3:0] bits are set to a Normal or a CTC mode (non-PWM).

For OC3B or OC4B when not using the Output Compare Modulator, PORTD2 must also be set in order to enable the output.

Table 18-8. Compare Output Mode, non-PWM

COM3A1/COM3B1	COM3A0/COM3B0	Description
0	0	Normal port operation, OC3A/OC3B disconnected.
0	1	Toggle OC3A/OC3B on Compare Match.
1	0	Clear OC3A/OC3B on Compare Match (Set output to low level).
1	1	Set OC3A/OC3B on Compare Match (Set output to high level).

The table below shows the COM1x1:0 bit functionality when the WGM33:0 bits are set to the fast PWM mode.

Table 18-9. Compare Output Mode, Fast PWM

COM3A1/ COM3B1	COM3A0/ COM3B0	Description
0	0	Normal port operation, OC3A/OC3B disconnected.
0	1	WGM33:0 = 14 or 15: Toggle OC3A on Compare Match, OC3B disconnected (normal port operation). For all other WGM3 settings, normal port operation, OC3A/OC3B disconnected.

COM3A1/ COM3B1	COM3A0/ COM3B0	Description
1	0	Clear OC3A/OC3B on Compare Match, set OC3A/OC3B at BOTTOM (non-inverting mode)
1	1	Set OC3A/OC3B on Compare Match, clear OC3A/OC3B at BOTTOM (inverting mode)

Note:

1. A special case occurs when OCR3A/OCR3B equals TOP and COM3A1/COM3B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM3x1:0 bit functionality when the WGM33:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 18-10. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COM3A1/ COM3B1	COM3A0/ COM3B0	Description
0	0	Normal port operation, OC3A/OC3B disconnected.
0	1	WGM33:0 = 9 or 11: Toggle OC3A on Compare Match, OC3B disconnected (normal port operation). For all other WGM3 settings, normal port operation, OC3A/OC3B disconnected.
1	0	Clear OC3A/OC3B on Compare Match when up-counting. Set OC3A/OC3B on Compare Match when down-counting.
1	1	Set OC3A/OC3B on Compare Match when up-counting. Clear OC3A/OC3B on Compare Match when down-counting.

Note:

1. A special case occurs when OCR3A/OCR3B equals TOP and COM3A1/COM3B1 is set. Refer to Phase Correct PWM Mode for details.

Bits 1:0 – WGM3[1:0]: Waveform Generation Mode [n = 1:0]

Combined with the WGM3[3:2] bits found in the TCCR3B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See Modes of Operation).

Table 18-11. Waveform Generation Mode Bit Description

Mode	WGM33	WGM32 (CTC1)[1]	WGM31 (PWM11)[1]	WGM30 (PWM10)[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM

Mode	WGM33	WGM32 (CTC1)[1]	WGM31 (PWM11)[1]	WGM30 (PWM10)[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR3A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR3A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR3	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR3A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR3	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR3	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR3A	BOTTOM	TOP

Note:

1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM32:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

18.11.9 TC3 Control Register B

Name: TCCR3B
Offset: 0x91
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ICNC3	ICES3		WGM3[3]	WGM3[2]	CS3[2:0]		
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit 7 – ICNC3: Input Capture Noise Canceler
Writing this bit to '1' activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture pin (ICP3) is filtered. The filter function requires four successive equal valued samples of the ICP3 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

Bit 6 – ICES3: Input Capture Edge Select
This bit selects which edge on the Input Capture pin (ICP3) that is used to trigger a capture event. When the ICES3 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES3 bit is written to '1', a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES3 setting, the counter value is copied into the Input Capture Register (ICR3). The event will also set the Input Capture Flag (ICF3), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR3 is used as TOP value (see description of the WGM13:0 bits located in the TCCR3A and the TCCR3B Register), the ICP3 is disconnected and consequently the Input Capture function is disabled.

Bits 3, 4 – WGM3[2], WGM3[3]: Waveform Generation Mode
Refer to TCCR1A.

Bits 2:0 – CS3[2:0]: Clock Select 3
The three Clock Select bits select the clock source to be used by the Timer/Counter. Refer to Figure 18-9 and Figure 18-10.

Table 18-12. Clock Select Bit Description

CS32	CS31	CS30	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$\text{clk}_{I/O}$/1 (No prescaling)
0	1	0	$\text{clk}_{I/O}$/8 (From prescaler)
0	1	1	$\text{clk}_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$\text{clk}_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

18.11.10 TC3 Control Register C

Name: TCCR3C
Offset: 0x92
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	FOC3A	FOC3B						
Access	R/W	R/W						
Reset	0	0						

Bits 6, 7 – FOC3B, FOC3A: Force Output Compare for Channel B and A
The FOC3A/FOC3B bits are only active when the WGM3[3:0] bits specifies a non-PWM mode. When writing a logical one to the FOC3A/FOC3B bit, an immediate compare match is forced on the Waveform

Generation unit. The OC3A/OC3B output is changed according to its COM3x[1:0] bits setting. Note that the FOC3A/FOC3B bits are implemented as strobes. Therefore it is the value present in the COM3x[1:0] bits that determine the effect of the forced compare.

A FOC3A/FOC3B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR3A as TOP. The FOC3A/FOC3B bits are always read as zero.

18.11.11 TC3 Counter Value Low and High byte

The TCNT3L and TCNT3H register pair represents the 16-bit value, TCNT3.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: TCNT3L and TCNT3H
Offset: 0x94
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	TCNT3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCNT3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – TCNT3[15:0]: Timer/Counter 3 Counter Value
The two Timer/Counter I/O locations (TCNT3H and TCNT3L, combined TCNT3) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

Modifying the counter (TCNT3) while the counter is running introduces a risk of missing a compare match between TCNT3 and one of the OCR3x Registers.

Writing to the TCNT3 Register blocks (removes) the compare match on the following timer clock for all compare units.

18.11.12 Input Capture Register 3 Low and High byte

The ICR3L and ICR3H register pair represents the 16-bit value, ICR3.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: ICR3L and ICR3H
Offset: 0x96
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	ICR3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ICR3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – ICR3[15:0]: Input Capture 3
The Input Capture is updated with the counter (TCNT3) value each time an event occurs on the ICP3 pin (or optionally on the Analog Comparator output for Timer/Counter3). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.13 Output Compare Register 3 A Low and High byte

The OCR3AL and OCR3AH register pair represents the 16-bit value, OCR3A.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: OCR3AL and OCR3AH
Offset: 0x98
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	OCR3A[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCR3A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – OCR3A[15:0]: Output Compare 3 A
The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3A pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.14 Output Compare Register 3 B Low and High byte

The OCR3BL and OCR3BH register pair represents the 16-bit value, OCR3B.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: OCR3BL and OCR3BH
Offset: 0x9A
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	OCR3B[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCR3B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – OCR3B[15:0]: Output Compare 3 B
The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3B pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.15 TC4 Control Register A

Name: TCCR4A
Offset: 0xA0
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	COM4A[1:0]		COM4B[1:0]				WGM4[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 4:5, 6:7 – COM4B, COM4A: Compare Output Mode for Channel
The COM4A[1:0] and COM4B[1:0] control the Output Compare pins (OC4A and OC4B respectively) behavior. If one or both of the COM4A[1:0] bits are written to one, the OC4A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM4B[1:0] bit are written to one, the OC4B output overrides the normal port functionality of the I/O pin it is connected to. However, note

that the Data Direction Register (DDR) bit corresponding to the OC4A or OC4B pin must be set in order to enable the output driver.

When the OC4A or OC4B is connected to the pin, the function of the COM4x[1:0] bits is dependent of the WGM4[3:0] bits setting. The table below shows the COM4n[1:0] bit functionality when the WGM4[3:0] bits are set to a Normal or a CTC mode (non-PWM).

For OC3B or OC4B when not using the Output Compare Modulator, PORTD2 must also be set in order to enable the output.

Table 18-13. Compare Output Mode, non-PWM

COM4A1/COM4B1	COM4A0/COM4B0	Description
0	0	Normal port operation, OC4A/OC4B disconnected.
0	1	Toggle OC4A/OC4B on Compare Match.
1	0	Clear OC4A/OC4B on Compare Match (Set output to low level).
1	1	Set OC4A/OC4B on Compare Match (Set output to high level).

The table below shows the COM4x[1:0] bit functionality when the WGM4[3:0] bits are set to the fast PWM mode.

Table 18-14. Compare Output Mode, Fast PWM

COM4A1/ COM4B1	COM4A0/ COM4B0	Description
0	0	Normal port operation, OC4A/OC4B disconnected.
0	1	WGM4[3:0] = 14 or 15: Toggle OC4A on Compare Match, OC4B disconnected (normal port operation). For all other WGM settings, normal port operation, OC4A/OC4B disconnected.
1	0	Clear OC4A/OC4B on Compare Match, set OC4A/OC4B at BOTTOM (non-inverting mode)
1	1	Set OC4A/OC4B on Compare Match, clear OC4A/OC4B at BOTTOM (inverting mode)

Note:

1. A special case occurs when OCR4A/OCR4B equals TOP and COM4A1/COM4B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM4x[1:0] bit functionality when the WGM4[3:0] bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 18-15. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COM4A1/ COM4B1	COM4A0/ COM4B0	Description
0	0	Normal port operation, OC4A/OC4B disconnected.
0	1	WGM4[3:0] = 9 or 11: Toggle OC4A on Compare Match, OC4B disconnected (normal port operation). For all other WGM4 settings, normal port operation, OC4A/OC4B disconnected.
1	0	Clear OC4A/OC4B on Compare Match when up-counting. Set OC4A/OC4B on Compare Match when down-counting.
1	1	Set OC4A/OC4B on Compare Match when up-counting. Clear OC4A/OC4B on Compare Match when down-counting.

Note:

1. A special case occurs when OCR4A/OCR4B equals TOP and COM4A1/COM4B1 is set. Refer to Phase Correct PWM Mode for details.

Bits 1:0 – WGM4[1:0]: Waveform Generation Mode

Combined with the WGM4[3:2] bits found in the TCCR4B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See Modes of Operation).

Table 18-16. Waveform Generation Mode Bit Description

Mode	WGM43	WGM42 (CTC1)[1]	WGM41 (PWM11)[1]	WGM40 (PWM10)[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR4x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR4A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR4	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR4A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR4	TOP	BOTTOM

Mode	WGM43	WGM42 (CTC1)(1)	WGM41 (PWM11)(1)	WGM40 (PWM10)(1)	Timer/ Counter Mode of Operation	TOP	Update of OCR4x at	TOV1 Flag Set on
11	1	0	1	1	PWM, Phase Correct	OCR4A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR4	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR4	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR4A	BOTTOM	TOP

Note:

1. The CTC1 and PWM1[1:0] bit definition names are obsolete. Use the WGM4[2:0] definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

18.11.16 TC4 Control Register B

Name: TCCR4B
Offset: 0xA1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ICNC4	ICES4		WGM4[3]	WGM4[2]	CS4[2:0]		
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit 7 – ICNC4: Input Capture Noise Canceler
Writing this bit to '1' activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture pin (ICP4) is filtered. The filter function requires four successive equal valued samples of the ICP4 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

Bit 6 – ICES4: Input Capture Edge Select
This bit selects which edge on the Input Capture pin (ICP4) that is used to trigger a capture event. When the ICES4 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES4 bit is written to '1', a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES4 setting, the counter value is copied into the Input Capture Register (ICR4). The event will also set the Input Capture Flag (ICF4), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR4 is used as TOP value (see description of the WGM4[3:0] bits located in the TCCR4A and the TCCR4B Register), the ICP4 is disconnected and consequently the Input Capture function is disabled.

Bits 3, 4 – WGM4[2], WGM4[3]: Waveform Generation Mode
Refer to TCCR4A.

Bits 2:0 – CS4[2:0]: Clock Select
The three Clock Select bits select the clock source to be used by the Timer/Counter. Refer to Figure 18-9 and Figure 18-10.

Table 18-17. Clock Select Bit Description

CS4[2]	CS4[1]	CS4[0]	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge.
1	1	1	External clock source on Tn pin. Clock on rising edge.

18.11.17 TC4 Control Register C

Name: TCCR4C
Offset: 0xA2
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	FOC4A	FOC4B						
Access	R/W	R/W						
Reset	0	0						

Bits 6, 7 – FOC4B, FOC4A: Force Output Compare for Channel B and A
The FOCA/FOCB bits are only active when the WGM4[3:0] bits specifies a non-PWM mode. When writing a logical one to the FOC4A/FOC4B bit, an immediate compare match is forced on the Waveform Generation unit. The OC4A/OC4B output is changed according to its COM4x[1:0] bits setting. Note that the FOCA/FOCB bits are implemented as strobes. Therefore it is the value present in the COM4x[1:0] bits that determine the effect of the forced compare.

A FOC4A/FOC4B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR4A as TOP. The FOC4A/FOC4B bits are always read as zero.

18.11.18 TC4 Counter Value Low and High byte

The TCNT4L and TCNT4H register pair represents the 16-bit value, TCNT4.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: TCNT4L and TCNT4H
Offset: 0xA4
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	TCNT4[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCNT4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – TCNT4[15:0]: Timer/Counter 4 Counter Value
The two Timer/Counter I/O locations (TCNT4H and TCNT4L, combined TCNT4) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

Modifying the counter (TCNT4) while the counter is running introduces a risk of missing a compare match between TCNT4 and one of the OCR4x Registers.

Writing to the TCNT4 Register blocks (removes) the compare match on the following timer clock for all compare units.

18.11.19 Input Capture Register 4 Low and High byte

The ICR4L and ICR4H register pair represents the 16-bit value, ICR4.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: ICR4L and ICR4H
Offset: 0xA6
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	ICR4[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ICR4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – ICR4[15:0]: Input Capture 3
The Input Capture is updated with the counter (TCNT4) value each time an event occurs on the ICP4 pin (or optionally on the Analog Comparator output for Timer/Counter4). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.20 Output Compare Register 4 A Low and High byte

The OCR4AL and OCR4AH register pair represents the 16-bit value, OCR4A.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: OCR4AL and OCR4AH
Offset: 0xA8
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	OCR4A[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCR4A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – OCR4A[15:0]: Output Compare 4 A
The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC4A pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.21 Output Compare Register 4 B Low and High byte

The OCR4BL and OCR4BH register pair represents the 16-bit value, OCR4B.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: OCR4BL and OCR4BH
Offset: 0xAA
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	OCR4B[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCR4B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – OCR4B[15:0]: Output Compare 4 B
The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC4B pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. Refer to Accessing 16-bit Timer/Counter Registers for details.

18.11.22 Timer/Counter 1 Interrupt Mask Register

Name: TIMSK1
Offset: 0x6F
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			ICIE1			OCIE1B	OCIE1A	TOIE1
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 5 – ICIE1: Timer/Counter 1, Input Capture Interrupt Enable
When this bit is written to '1', and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 1 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF1 Flag, located in TIFR1, is set.

Bit 2 – OCIE1B: Timer/Counter 1, Output Compare B Match Interrupt Enable
When this bit is written to '1', and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 1 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF1B Flag, located in TIFR1, is set.

Bit 1 – OCIE1A: Timer/Counter 1, Output Compare A Match Interrupt Enable
When this bit is written to '1', and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF1A Flag, located in TIFR1, is set.

Bit 0 – TOIE1: Timer/Counter 1, Overflow Interrupt Enable
When this bit is written to '1', and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 1 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV Flag, located in TIFR1, is set.

18.11.23 Timer/Counter 3 Interrupt Mask Register

Name: TIMSK3
Offset: 0x71
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			ICIE3			OCIE3B	OCIE3A	TOIE3
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 5 – ICIE3: Timer/Counter 3, Input Capture Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF3 Flag, located in TIFR3, is set.

Bit 2 – OCIE3B: Timer/Counter3, Output Compare B Match Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 3 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3B Flag, located in TIFR3, is set.

Bit 1 – OCIE3A: Timer/Counter 3, Output Compare A Match Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 3 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3A Flag, located in TIFR3, is set.

Bit 0 – TOIE3: Timer/Counter 3, Overflow Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 3 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV3 Flag, located in TIFR3, is set.

18.11.24 Timer/Counter 4 Interrupt Mask Register

Name: TIMSK4
Offset: 0x72
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			ICIE4			OCIE4B	OCIE4A	TOIE4
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 5 – ICIE4: Timer/Counter 4, Input Capture Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 4 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF4 Flag, located in TIFR4, is set.

Bit 2 – OCIE4B: Timer/Counter 4, Output Compare B Match Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter n Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4B Flag, located in TIFR4, is set.

Bit 1 – OCIE4A: Timer/Counter 4, Output Compare A Match Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter n Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4A Flag, located in TIFR4, is set.

Bit 0 – TOIE4: Timer/Counter 4, Overflow Interrupt Enable
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 4 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV Flag, located in TIFR4, is set.

18.11.25 TC1 Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TIFR1
Offset: 0x36
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x16

Bit	7	6	5	4	3	2	1	0
			ICF1			OCF1B	OCF1A	TOV1
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 5 – ICF1: Timer/Counter 1, Input Capture Flag
This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM1[3:0] to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value.

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

Bit 2 – OCF1B: Timer/Counter 1, Output Compare B Match Flag
This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a Forced Output Compare (FOC1B) strobe will not set the OCF1B Flag.

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

Bit 1 – OCF1A: Timer/Counter 1, Output Compare A Match Flag
This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag.

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

Bit 0 – TOV1: Timer/Counter 1, Overflow Flag
The setting of this flag is dependent of the WGM1[3:0] bits setting. In Normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to the Waveform Generation Mode bit description for the TOV1 Flag behavior when using another WGM1[3:0] bit setting.

TOV1 is automatically cleared when the Timer/Counter 1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

18.11.26 TC3 Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TIFR3
Offset: 0x38
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x18

Bit	7	6	5	4	3	2	1	0
			ICF3			OCF3B	OCF3A	TOV3
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 5 – ICF3: Timer/Counter3, Input Capture Flag
This flag is set when a capture event occurs on the ICP3 pin. When the Input Capture Register (ICR3) is set by the WGM33:0 to be used as the TOP value, the ICF3 Flag is set when the counter reaches the TOP value.

ICF3 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF3 can be cleared by writing a logic one to its bit location.

Bit 2 – OCF3B: Timer/Counter3, Output Compare B Match Flag
This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register B (OCR3B).

Note that a Forced Output Compare (FOC3B) strobe will not set the OCF3B Flag.

OCF3B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF3B can be cleared by writing a logic one to its bit location.

Bit 1 – OCF3A: Timer/Counter3, Output Compare A Match Flag
This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register A (OCR3A).

Note that a Forced Output Compare (FOC3A) strobe will not set the OCF3A Flag.

OCF3A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF3A can be cleared by writing a logic one to its bit location.

Bit 0 – TOV3: Timer/Counter1, Overflow Flag
The setting of this flag is dependent of the WGM33:0 bits setting. In Normal and CTC modes, the TOV3 Flag is set when the timer overflows. Refer to Table 18-6 for the TOV3 Flag behavior when using another WGM33:0 bit setting.

TOV3 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV3 can be cleared by writing a logic one to its bit location.

18.11.27 TC4 Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TIFR4
Offset: 0x39
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x18, 0x19

Bit	7	6	5	4	3	2	1	0
			ICF4			OCF4B	OCF4A	TOV4
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 5 – ICF4: Timer/Counter 4, Input Capture Flag

This flag is set when a capture event occurs on the ICP4 pin. When the Input Capture 4 Register (ICR4) is set by the WGM[3:0] to be used as the TOP value, the ICF4 Flag is set when the counter reaches the TOP value.

ICF4 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF4 can be cleared by writing a logic one to its bit location.

Bit 2 – OCF4B: Timer/Counter 4, Output Compare B Match Flag

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register B (OCR4B).

Note that a Forced Output Compare (FOC4B) strobe will not set the OCF4B Flag.

OCF4B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF4B can be cleared by writing a logic one to its bit location.

Bit 1 – OCF4A: Timer/Counter 4, Output Compare A Match Flag

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare 4 Register A (OCR4A).

Note that a Forced Output Compare (FOC4A) strobe will not set the OCF4A Flag.

OCF4A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF4A can be cleared by writing a logic one to its bit location.

Bit 0 – TOV4: Timer/Counter 4, Overflow Flag

The setting of this flag is dependent of the WGM[3:0] bits setting. In Normal and CTC modes, the TOV4 Flag is set when the timer overflows. Refer to the Waveform Generation Mode bit description for the TOV4 Flag behavior when using another WGM[3:0] bit setting.

TOV is automatically cleared when the Timer/Counter 4 Overflow Interrupt Vector is executed. Alternatively, TOV4 can be cleared by writing a logic one to its bit location.

19. Timer/Counter 0, 1, 3, 4 Prescalers

The 8-bit Timer/Counter0 (TC0) , 16-bit Timer/Counters 1, 3, 4 (TC1, TC3, and TC4) share the same prescaler module, but the Timer/Counters can have different prescaler settings. The following description applies to: TC0 , TC1 ,TC3 , TC4.

Related Links

TC0 - 8-bit Timer/Counter0 with PWM

TC1, 3, 4 - 16-bit Timer/Counter1, 3, 4 with PWM

19.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn[2:0]=0x1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, or $f_{CLK\_I/O}/1024$.

19.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (0x6 > CSn[2:0] > 0x1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

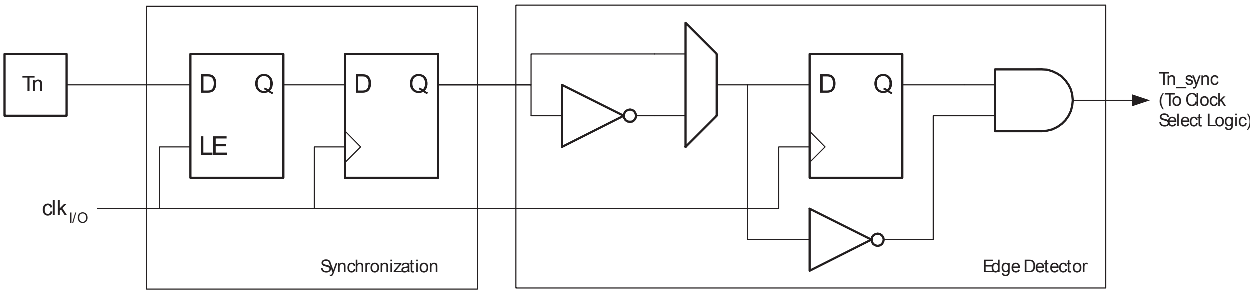
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

19.3 External Clock Source

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

The edge detector generates one clk_{T1}/clk_{T0} pulse for each positive (CSn[2:0]=0x7) or negative (CSn[2:0]=0x6) edge it detects.

Figure 19-1. T1/T0 Pin Sampling



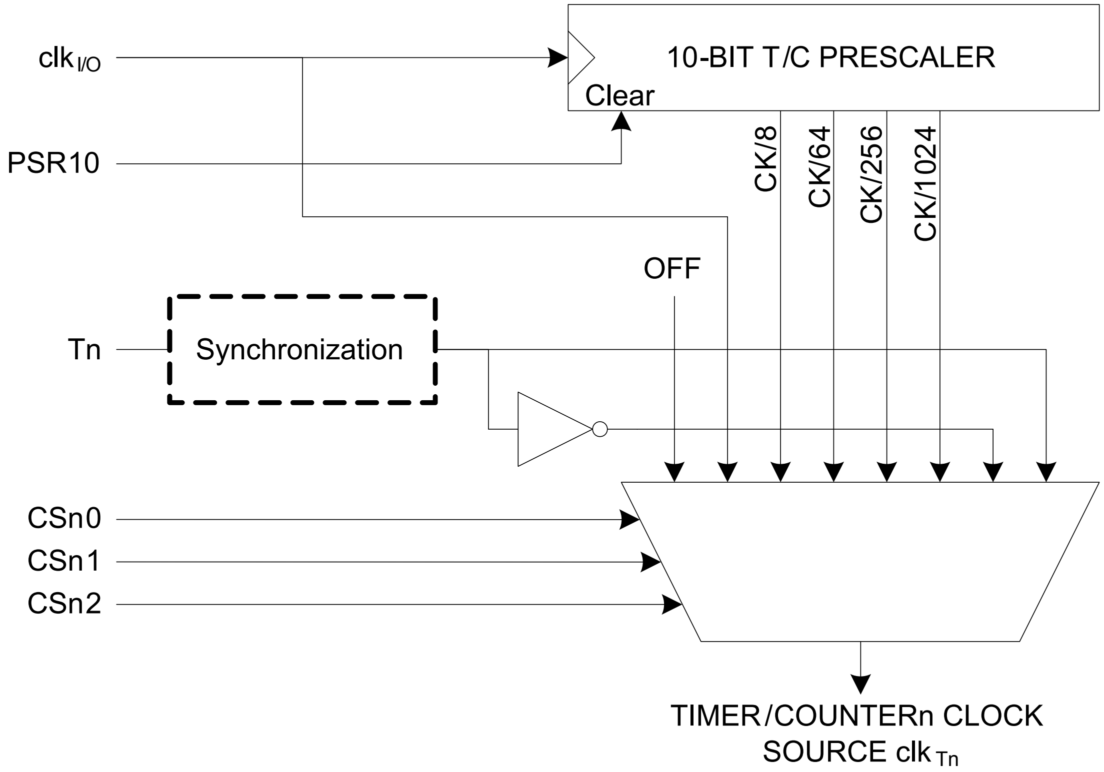
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

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

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{Tn} < f_{clk\_I/O}/2$) given a 50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by the tolerances of the oscillator source (crystal, resonator, and capacitors), it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

Figure 19-2. Prescaler for Timer/Counter0 and Timer/Counter1(1)



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in the block diagram above.

19.4 Register Description

19.4.1 General Timer/Counter Control Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GTCCR
Offset: 0x43
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x23

Bit	7	6	5	4	3	2	1	0
	TSM						PSRASY	PSRSYNC
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – TSM: Timer/Counter Synchronization Mode
Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

Bit 1 – PSRASY: Prescaler Reset Timer/Counter2
When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

Bit 0 – PSRSYNC: Prescaler Reset
When this bit is one, Timer/Counter 0, 1, 3, 4 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter 0, 1, 3, 4 share the same prescaler and a reset of this prescaler will affect the mentioned timers.

20. TC2 - 8-bit Timer/Counter2 with PWM and Asynchronous Operation

20.1 Features

- Dual Channel Counter
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV2, OCF2A, and OCF2B)
- Allows Clocking from External 32kHz Watch Crystal Independent of the I/O Clock

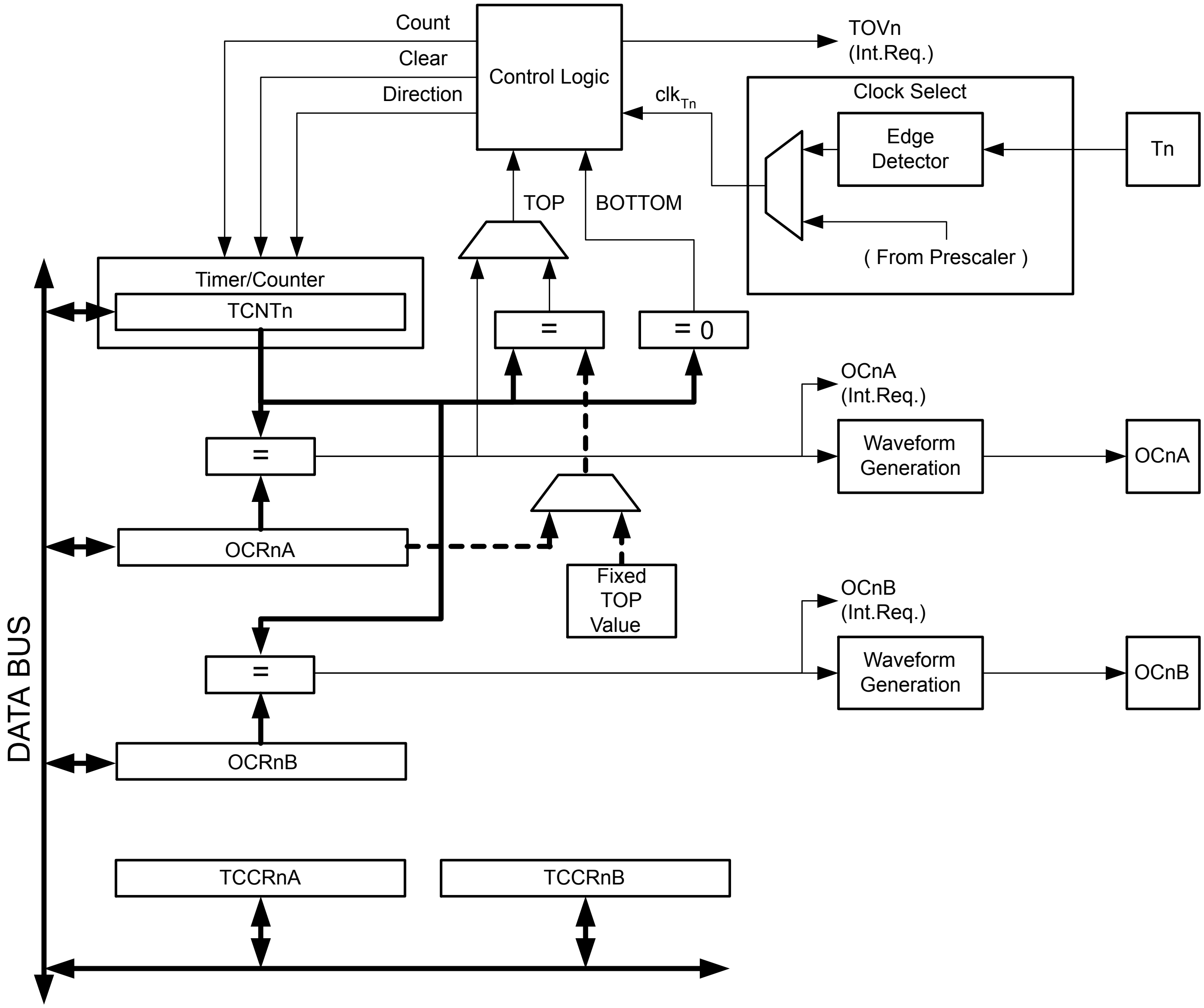
20.2 Overview

Timer/Counter2 (TC2) is a general purpose, dual channel, 8-bit Timer/Counter module.

A simplified block diagram of the 8-bit Timer/Counter is shown below. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the following Register Description. For the actual placement of I/O pins, refer to the pinout diagram.

The TC2 is enabled when the PRTIM2 bit in the Power Reduction Register (PRR0.PRTIM2) is written to '1'.

Figure 20-1. 8-bit Timer/Counter Block Diagram



Related Links

Pin Configurations

Pin Descriptions

20.2.1 Definitions

Many register and bit references in this section are written in general form:

- n=2 represents the Timer/Counter number
- x=A,B represents the Output Compare Unit A or B

However, when using the register or bit definitions in a program, the precise form must be used, i.e., TCNT2 for accessing Timer/Counter2 counter value.

The following definitions are used throughout the section:

Table 20-1. Definitions

Constant	Description
BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its maximum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

20.2.2 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source he Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).

The double buffered Output Compare Register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC2A and OC2B). See Output Compare Unit for details. The compare match event will also set the Compare Flag (OCF2A or OCF2B) which can be used to generate an Output Compare interrupt request.

20.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source:

The clock source clk_{T2} is by default equal/synchronous to the MCU clock, $clk_{I/O}$.

When the Asynchronous TC2 bit in the Asynchronous Status Register (ASSR.AS2) is written to '1', the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2.

For details on asynchronous operation, see the description of the ASSR. For details on clock sources and prescaler, see Timer/Counter Prescaler.

20.4 Counter Unit

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

Figure 20-2. Counter Unit Block Diagram

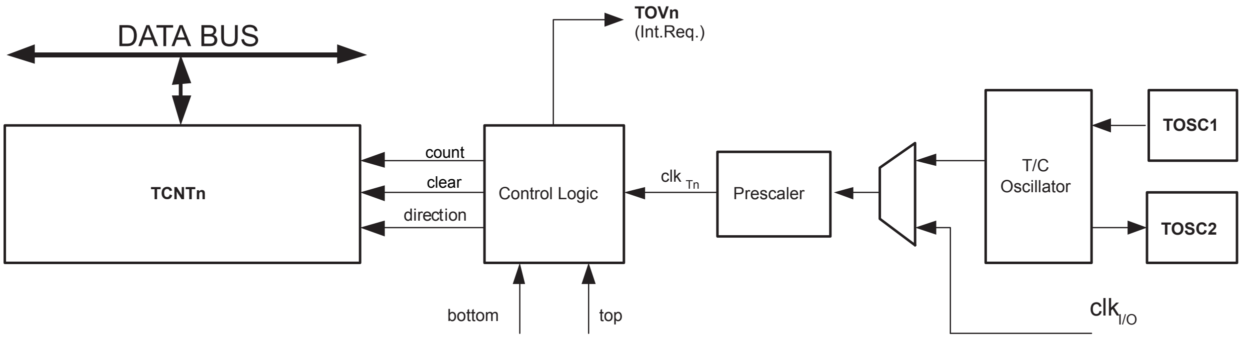


Table 20-2. Signal description (internal signals):

Signal name	Description
count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS2[2:0]). When no clock source is selected (CS2[2:0]=0x0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2A) and the WGM22 bit located in the Timer/Counter Control Register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see "Modes of Operation".

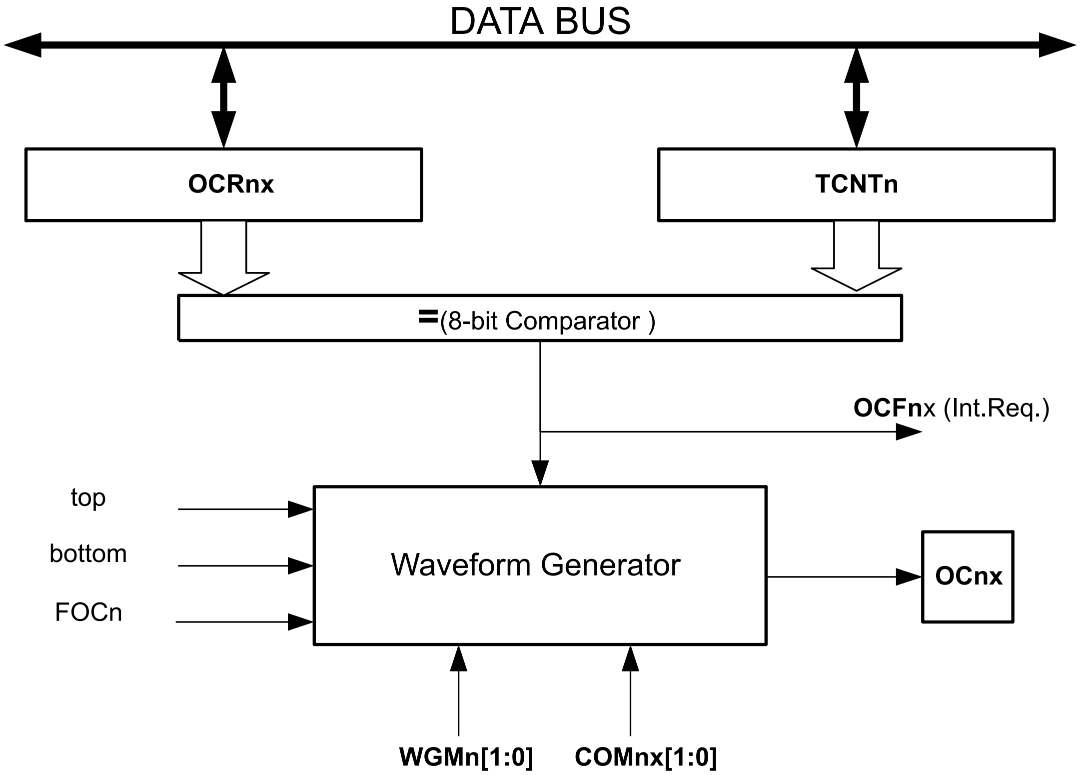
The Timer/Counter Overflow Flag (TOV2) is set according to the mode of operation selected by the TCC2B.WGM2[2:0] bits. TOV2 can be used for generating a CPU interrupt.

20.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the Output Compare Flag (OCF2A or OCF2B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the Output Compare Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM2[2:0] bits and Compare Output mode (COM2x[1:0]) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See Modes of Operation).

The following figure shows a block diagram of the Output Compare unit.

Figure 20-3. Output Compare Unit, Block Diagram



The OCR2x Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2x Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR2x Buffer Register, and if double buffering is disabled the CPU will access the OCR2x directly.

Related Links

Modes of Operation

20.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2x) bit. Forcing compare match will not set the OCF2x Flag or reload/clear the timer, but the OC2x pin will be updated as if a real compare match had occurred (the COM2x[1:0] bits settings define whether the OC2x pin is set, cleared or toggled).

20.5.2 Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2x to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

20.5.3 Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the Output Compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.

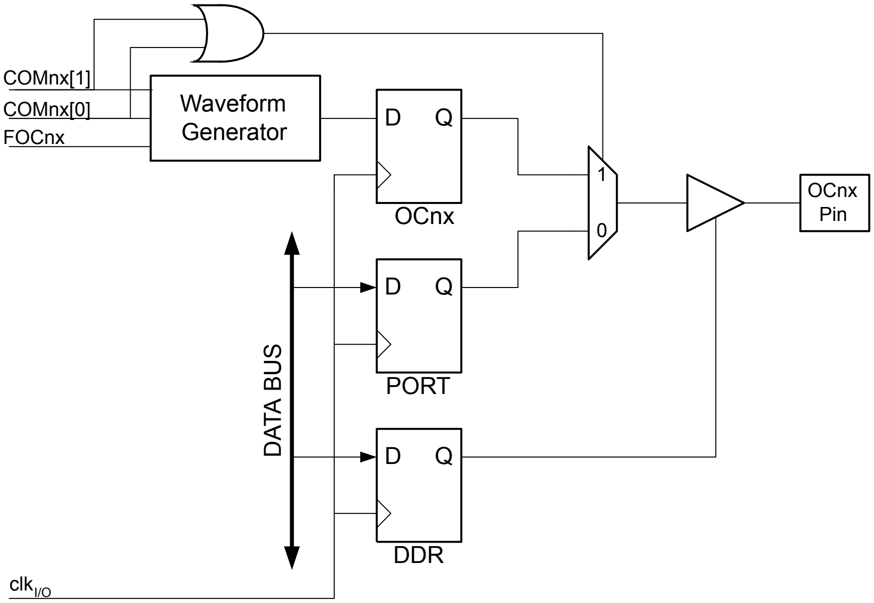
The setup of the OC2x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2x value is to use the Force Output Compare (FOC2x) strobe bit in Normal mode. The OC2x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM2x[1:0] bits are not double buffered together with the compare value. Changing the COM2x[1:0] bits will take effect immediately.

20.6 Compare Match Output Unit

The Compare Output mode (COM2x[1:0]) bits have two functions. The Waveform Generator uses the COM2x[1:0] bits for defining the Output Compare (OC2x) state at the next compare match. Also, the COM2x[1:0] bits control the OC2x pin output source. The following figure shows a simplified schematic of the logic affected by the COM2x[1:0] bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM2x[1:0] bits are shown. When referring to the OC2x state, the reference is for the internal OC2x Register, not the OC2x pin.

Figure 20-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC2x) from the Waveform Generator if either of the COM2x1:0 bits are set. However, the OC2x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC2x pin (DDR_OC2x) must be set as output before the OC2x value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC2x state before the output is enabled. Note that some COM2x[1:0] bit settings are reserved for certain modes of operation. See Register Description.

Related Links

Modes of Operation

20.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM2x[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the COM2x[1:0] = 0 tells the Waveform Generator that no action on the OC2x Register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the COM2x[1:0] bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC2x strobe bits.

20.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM2[2:0]) and Compare Output mode (COM2x[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM2x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM2x[1:0] bits control whether the output should be set, cleared, or toggled at a compare match (See Compare Match Output Unit).

For detailed timing information refer to Timer/Counter Timing Diagrams.

20.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM2[2:0] = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

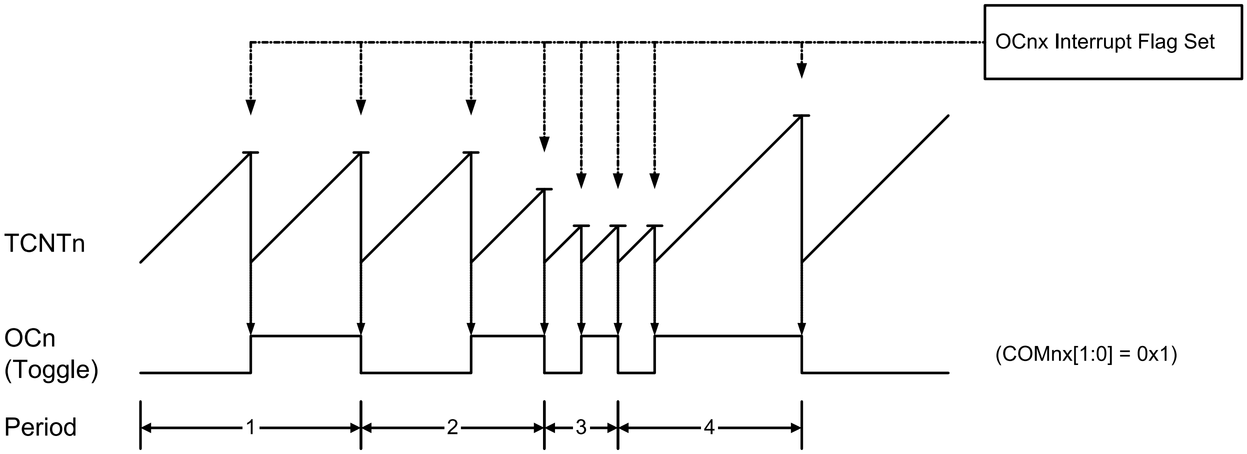
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

20.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM2[2:0] = 2), the OCR2A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is as follows. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and then counter (TCNT2) is cleared.

Figure 20-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2A is lower than the current value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC2A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM2A[1:0] = 1). The OC2A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC2A} = f_{clk\_I/O}/2$ when OCR2A is set to zero (0x00). The waveform frequency is defined by the following equation:

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

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

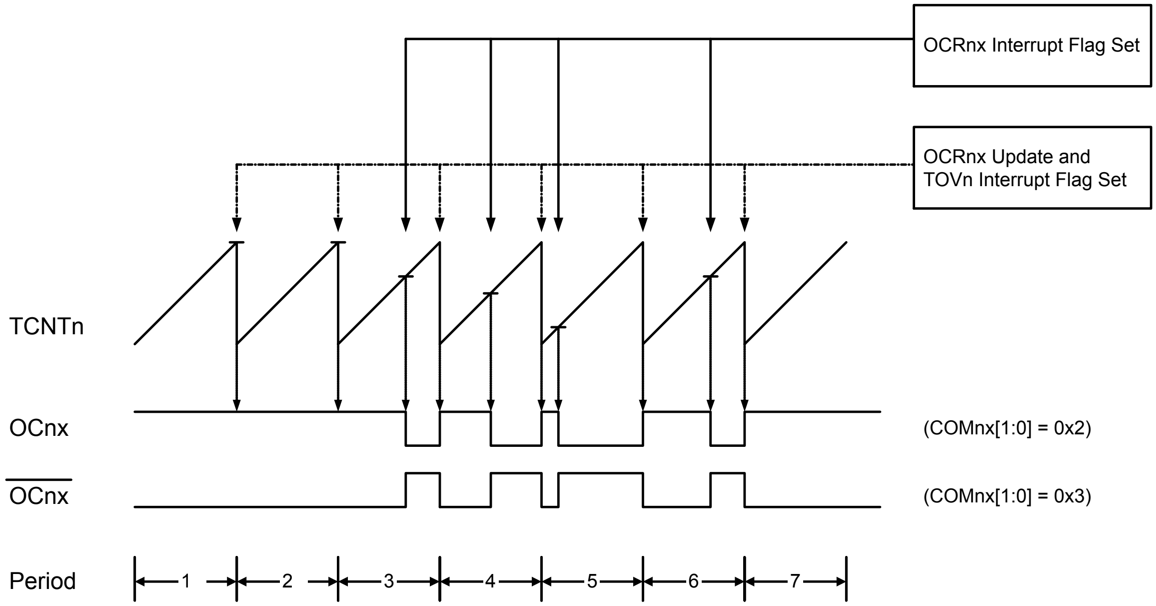
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

20.7.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM2[2:0] = 0x3 or 0x7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2[2:0] = 0x3, and OCR2A when WGM2[2:0] = 0x7. In non-inverting Compare Output mode, the Output Compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is depicted in the following figure. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2x and TCNT2.

Figure 20-6. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM2x[1:0] to three. TOP is defined as 0xFF when WGM2[2:0] = 0x3, and OCR2A when MGM2[2:0] = 0x7. The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2x Register at the compare match between OCR2x and TCNT2, and clearing (or setting) the OC2x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR2A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR2A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM2A[1:0] bits.)

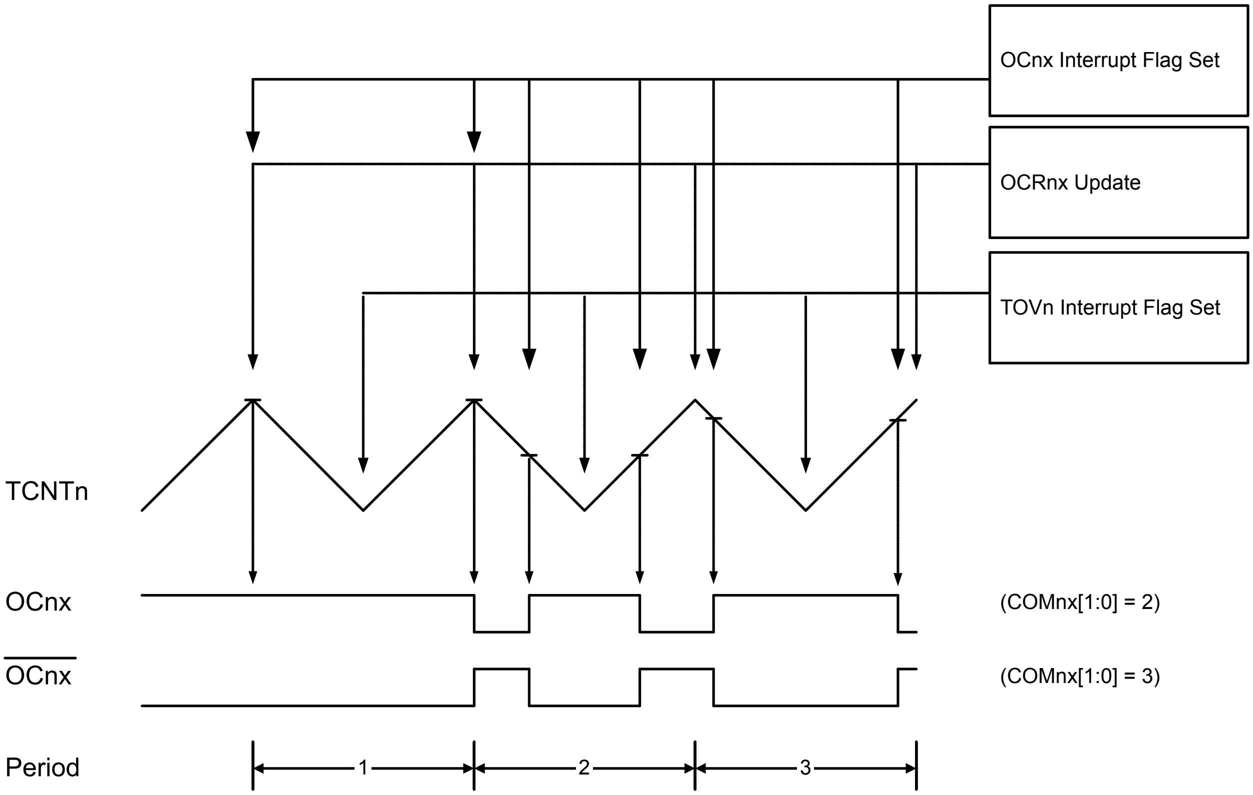
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2x to toggle its logical level on each compare match (COM2x[1:0] = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk\_I/O}/2$ when OCR2A is set to zero. This feature is similar to the OC2A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

20.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM2[2:0] = 0x1 or 0x5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2[2:0] = 0x3, and OCR2A when MGM2[2:0] = 7. In non-inverting Compare Output mode, the Output Compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT2 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 20-7. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2x and TCNT2.

Figure 20-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x[1:0] bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM2x[1:0] to three. TOP is defined as 0xFF when WGM2[2:0] = 0x3, and OCR2A when WGM2[2:0] = 7. The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2x Register at the compare match between OCR2x and TCNT2 when the counter increments, and setting (or clearing) the OC2x Register at compare match between OCR2x and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

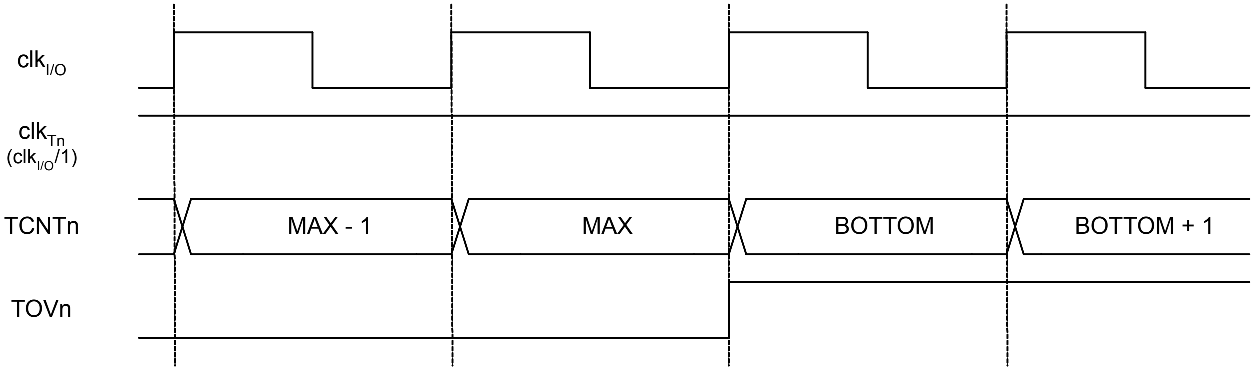
At the very start of period 2 in the above figure OC2x has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR2A changes its value from MAX, as shown in the preceeding figure. When the OCR2A value is MAX the OC2 pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OC2 value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OC2 change that would have happened on the way up.

20.8 Timer/Counter Timing Diagrams

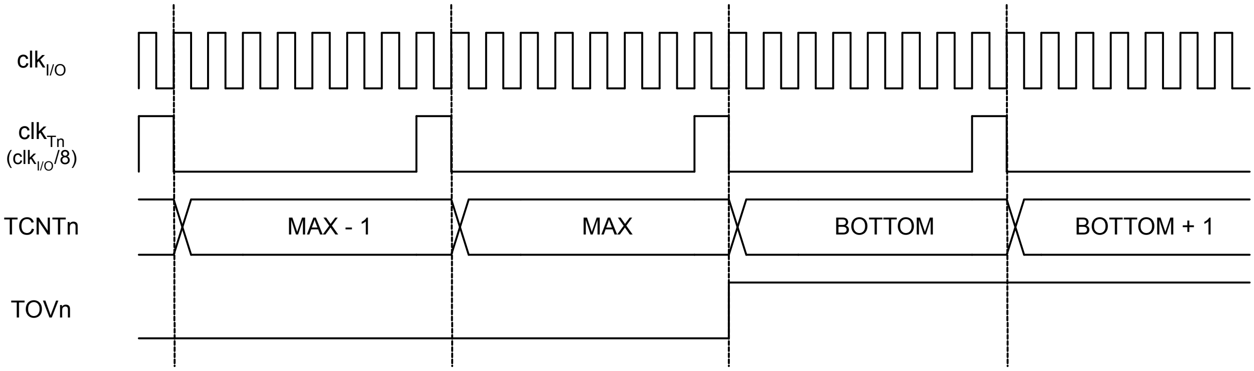
The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. The following figure contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 20-8. Timer/Counter Timing Diagram, no Prescaling



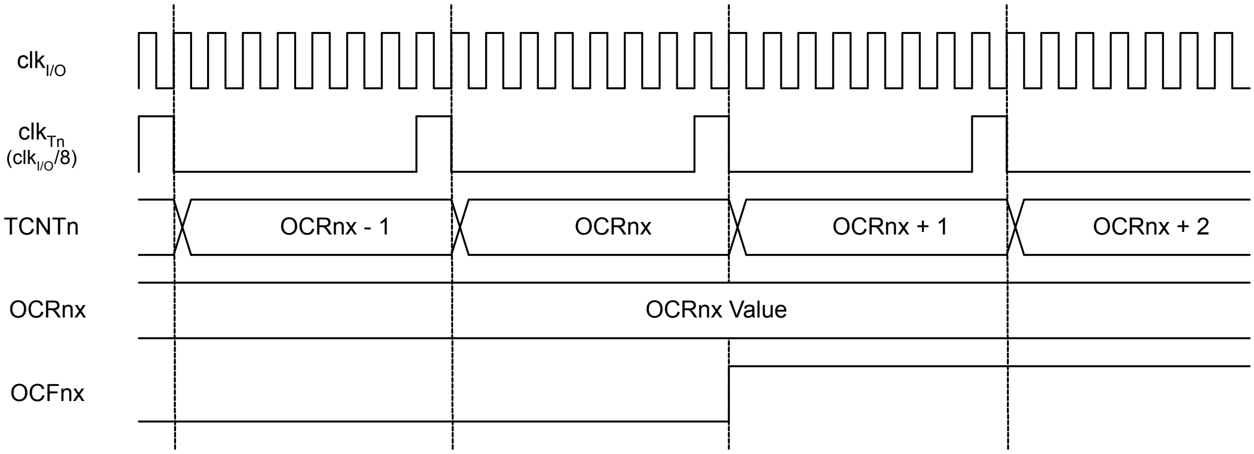
The following figure shows the same timing data, but with the prescaler enabled.

Figure 20-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}$/8)



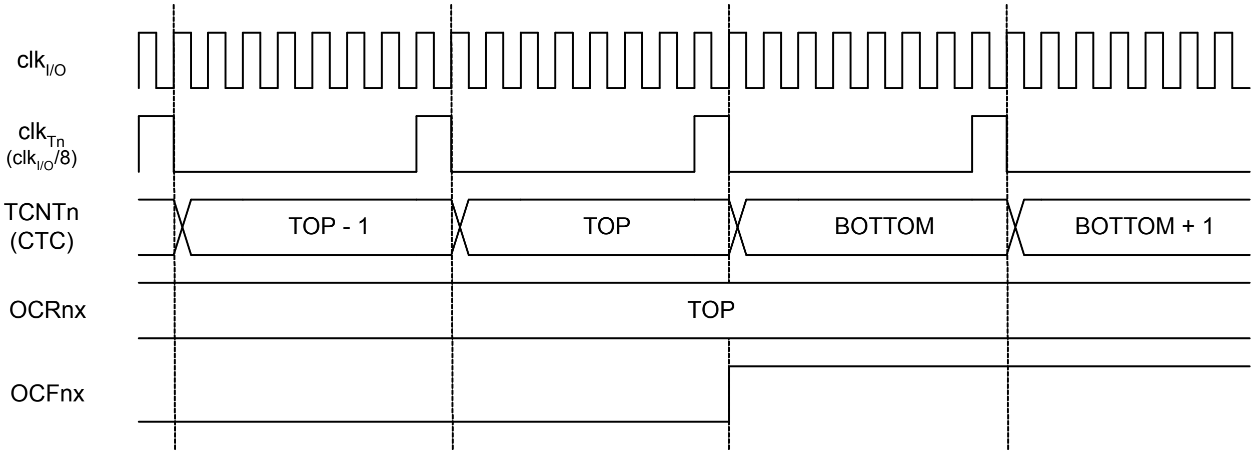
The following figure shows the setting of OCF2A in all modes except CTC mode.

Figure 20-10. Timer/Counter Timing Diagram, Setting of OCF2A, with Prescaler ($f_{clk\_I/O}$/8)



The following figure shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

Figure 20-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}$/8)



20.9 Asynchronous Operation of Timer/Counter2

When TC2 operates asynchronously, some considerations must be taken:

- When switching between asynchronous and synchronous clocking of TC2, the registers TCNT2, OCR2x, and TCCR2x might be corrupted. A safe procedure for switching clock source is:
    1. Disable the TC2 interrupts by clearing OCIE2x and TOIE2.
    2. Select clock source by setting AS2 as appropriate.
    3. Write new values to TCNT2, OCR2x, and TCCR2x.
    4. To switch to asynchronous operation: Wait for TCN2xUB, OCR2xUB, and TCR2xUB.
    5. Clear the TC2 Interrupt Flags.
    6. Enable interrupts, if needed.
- The CPU main clock frequency must be more than four times the oscillator frequency.
- When writing to one of the registers TCNT2, OCR2x, or TCCR2x, 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 five mentioned registers has its individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2x write in progress. The Asynchronous Status Register (ASSR) indicates that a transfer to the destination register has taken place.
- When entering Power-save or ADC Noise Reduction mode after having written to TCNT2, OCR2x, or TCCR2x, the user must wait until the written register has been updated if TC2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if any of the Output Compare2 interrupts is used to wake up the device, since the Output Compare function is disabled during writing to OCR2x or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the corresponding OCR2xUB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If TC2 is used to wake the device up from Power-save or ADC Noise Reduction mode, precautions must be taken if the user wants to re-enter one of these modes: If re-entering sleep mode within the TOSC1 cycle, the interrupt will immediately occur and the device wake up again. The result is multiple interrupts and wake-ups within one TOSC1 cycle from the first interrupt. If the user is in doubt whether the time before re-entering Power-save or ADC Noise Reduction mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
    1. Write a value to TCCR2x, TCNT2, or OCR2x.
    2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
    3. Enter Power-save or ADC Noise Reduction mode.
- When the asynchronous operation is selected, the 32.768kHz oscillator for TC2 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 TC2 after power-up or wake-up from Power-down or Standby mode. The contents of all TC2 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 ADC Noise Reduction 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.
- Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Power-save mode, and the I/O clock ($clk_{I/O}$) again

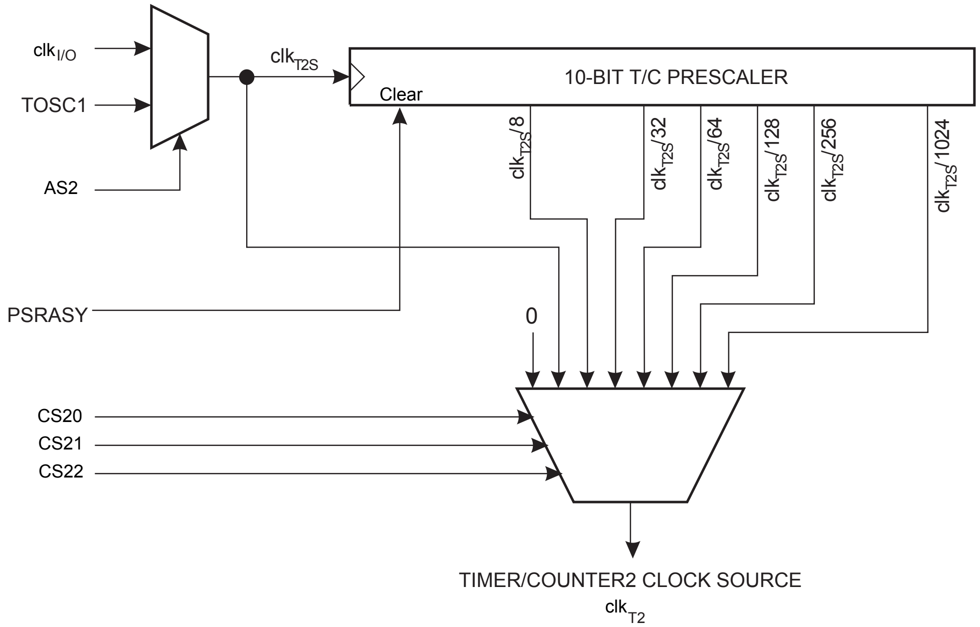
becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:

1. Wait for the corresponding Update Busy Flag to be cleared.
2. Read TCNT2.

- 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.

20.10 Timer/Counter Prescaler

Figure 20-12. Prescaler for TC2



The clock source for TC2 is named clk_{T2S}. It is by default connected to the main system I/O clock $clk_{I/O}$. By writing a '1' to the Asynchronous TC2 bit in the Asynchronous Status Register (ASSR.AS2), TC2 is asynchronously clocked from the TOSC1 pin. This enables use of TC2 as a Real Time Counter (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from Port B. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for TC2. The Oscillator is optimized for use with a 32.768kHz crystal.

For TC2, the possible prescaled selections are: clk_{T2S}/8, clk_{T2S}/32, clk_{T2S}/64, clk_{T2S}/128, clk_{T2S}/256, and clk_{T2S}/1024. Additionally, clk_{T2S} as well as 0 (stop) may be selected. The prescaler is reset by writing a '1' to the Prescaler Reset TC2 bit in the General TC2 Control Register (GTCCR.PSRASY). This allows the user to operate with a defined prescaler.

20.11 Register Description

20.11.1 TC2 Control Register A

Name: TCCR2A
Offset: 0xB0
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	COM2A [1:0]		COM2B [1:0]				WGM2 [1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 7:6 – COM2A [1:0]: Compare Output Mode for Channel A
These bits control the Output Compare pin (OC2A) behavior. If one or both of the COM2A[1:0] bits are set, the OC2A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2A pin must be set in order to enable the output driver.

When OC2A is connected to the pin, the function of the COM2A[1:0] bits depends on the WGM2[2:0] bit setting. The table below shows the COM2A[1:0] bit functionality when the WGM2[2:0] bits are set to a normal or CTC mode (non- PWM).

Table 20-3. Compare Output Mode, non-PWM

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Toggle OC2A on Compare Match.
1	0	Clear OC2A on Compare Match.
1	1	Set OC2A on Compare Match .

The table below shows the COM2A[1:0] bit functionality when the WGM2[1:0] bits are set to fast PWM mode.

Table 20-4. Compare Output Mode, Fast PWM(1)

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected WGM22 = 1: Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match, set OC2A at BOTTOM (non-inverting mode)
1	1	Set OC2A on Compare Match, clear OC2A at BOTTOM (inverting mode)

Note:

1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM2A[1:0] bit functionality when the WGM2[2:0] bits are set to phase correct PWM mode.

Table 20-5. Compare Output Mode, Phase Correct PWM Mode[(1)]

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match.
1	0	Clear OC2A on Compare Match when up-counting. Set OC2A on Compare Match when down-counting.
1	1	Set OC2A on Compare Match when up-counting. Clear OC2A on Compare Match when down-counting.

Note:

1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

Bits 5:4 – COM2B [1:0]: Compare Output Mode for Channel B
These bits control the Output Compare pin (OC2B) behavior. If one or both of the COM2B[1:0] bits are set, the OC2B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2B pin must be set in order to enable the output driver.

When OC2B is connected to the pin, the function of the COM2B[1:0] bits depends on the WGM2[2:0] bit setting. The table shows the COM2B[1:0] bit functionality when the WGM2[2:0] bits are set to a normal or CTC mode (non- PWM).

Table 20-6. Compare Output Mode, non-PWM

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Toggle OC2B on Compare Match.
1	0	Clear OC2B on Compare Match.
1	1	Set OC2B on Compare Match.

The table below shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to fast PWM mode.

Table 20-7. Compare Output Mode, Fast PWM[(1)]

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

Note:

1. A special case occurs when OCR2B equals TOP and COM2B[1] is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Fast PWM Mode for details.

The table below shows the COM2B[1:0] bit functionality when the WGM2[2:0] bits are set to phase correct PWM mode.

Table 20-8. Compare Output Mode, Phase Correct PWM Mode[1]

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on Compare Match when up-counting. Set OC2B on Compare Match when down-counting.
1	1	Set OC2B on Compare Match when up-counting. Clear OC2B on Compare Match when down-counting.

Note:

1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

Bits 1:0 – WGM2 [1:0]: Waveform Generation Mode
Combined with the WGM22 bit found in the TCCR2B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see Modes of Operation).

Table 20-9. Waveform Generation Mode Bit Description

Mode	WGM22	WGM21	WGM20	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on[1]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Note:

1. MAX = 0xFF
2. BOTTOM = 0x00

20.11.2 TC2 Control Register B

Name: TCCR2B
Offset: 0xB1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	FOC2A	FOC2B			WGM2 [2]	CS2[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 7 – FOC2A: Force Output Compare A

The FOC2A bit is only active when the WGM bits specify a non-PWM mode.

To ensure compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2A output is changed according to its COM2A[1:0] bits setting. Note that the FOC2A bit is implemented as a strobe. Therefore it is the value present in the COM2A[1:0] bits that determines the effect of the forced compare.

A FOC2A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2A as TOP.

The FOC2A bit is always read as zero.

Bit 6 – FOC2B : Force Output Compare B

The FOC2B bit is only active when the WGM bits specify a non-PWM mode.

To ensure compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2B output is changed according to its COM2B[1:0] bits setting. Note that the FOC2B bit is implemented as a strobe. Therefore it is the value present in the COM2B[1:0] bits that determines the effect of the forced compare.

A FOC2B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2B as TOP.

The FOC2B bit is always read as zero.

Bit 3 – WGM2 [2]: Waveform Generation Mode

Refer to TCCR2A.

Bits 2:0 – CS2[2:0]: Clock Select 2 [n = 0..2]

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

Table 20-10. Clock Select Bit Description

CA22	CA21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/32 (From prescaler)

CA22	CA21	CS20	Description
1	0	0	clkI/O/64 (From prescaler)
1	0	1	clk$_{I/O}$/128 (From prescaler)
1	1	0	clk$_{I/O}$/256 (From prescaler)
1	1	1	clk$_{I/O}$/1024 (From prescaler)

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

20.11.3 TC2 Counter Value Register

Name: TCNT2
Offset: 0xB2
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TCNT2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TCNT2[7:0]: Timer/Counter 2 Counter Value
The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT2 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a Compare Match between TCNT2 and the OCR2x Registers.

20.11.4 TC2 Output Compare Register A

Name: OCR2A
Offset: 0xB3
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OCR2A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR2A[7:0]: Output Compare 2 A
The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2A pin.

20.11.5 TC2 Output Compare Register B

Name: OCR2B
Offset: 0xB4
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OCR2B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR2B[7:0]: Output Compare 2 B
The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2B pin.

20.11.6 TC2 Interrupt Mask Register

Name: TIMSK2
Offset: 0x70
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						OCIE2B	OCIE2A	TOIE2
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCIE2B: Timer/Counter 2, Output Compare B Match Interrupt Enable
When the OCIE2B bit is written to '1' and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter 2 occurs, i.e., when the OCF2B bit is set in TIFR2.

Bit 1 – OCIE2A: Timer/Counter 2, Output Compare A Match Interrupt Enable
When the OCIE2A bit is written to '1' and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter 2 occurs, i.e., when the OCF2A bit is set in TIFR2.

Bit 0 – TOIE2: Timer/Counter 2, Overflow Interrupt Enable
When the TOIE2 bit is written to '1' 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/Counter 2 occurs, i.e., when the TOV2 bit is set in TIFR2.

20.11.7 TC2 Interrupt Flag Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TIFR2
Offset: 0x37
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x17

Bit	7	6	5	4	3	2	1	0
						OCF2B	OCF2A	TOV2
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCF2B: Timer/Counter 2, Output Compare B Match Flag
The OCF2B bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2B – Output Compare Register2. OCF2B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2B (Timer/Counter2 Compare match Interrupt Enable), and OCF2B are set (one), the Timer/Counter 2 Compare match Interrupt is executed.

Bit 1 – OCF2A: Timer/Counter 2, Output Compare A Match Flag
The OCF2A bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCRA – Output Compare Register2. OCF2A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2A (Timer/Counter2 Compare match Interrupt Enable), and OCF2A are set (one), the Timer/Counter 2 Compare match Interrupt is executed.

Bit 0 – TOV2: Timer/Counter 2, Overflow Flag
The TOV2 bit is set (one) when an overflow occurs in Timer/Counter 2. 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, TOIE2A (Timer/Counter 2 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter 2 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter 2 changes counting direction at 0x00.

20.11.8 Asynchronous Status Register

Name: ASSR
Offset: 0xB6
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
Access		R/W	R/W	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 6 – EXCLK: Enable External Clock Input
When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on Timer Oscillator 1 (TOSC1) pin instead of a 32kHz crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal Oscillator will only run when this bit is zero.

Bit 5 – AS2: Asynchronous Timer/Counter2
When AS2 is written to zero, Timer/Counter2 is clocked from the I/O clock, clkI/O. When AS2 is written to one, Timer/Counter2 is clocked from a crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B might be corrupted.

Bit 4 – TCN2UB: Timer/Counter2 Update Busy
When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

Bit 3 – OCR2AUB: Output Compare Register2A Update Busy
When Timer/Counter2 operates asynchronously and OCR2A is written, this bit becomes set. When OCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2A is ready to be updated with a new value.

Bit 2 – OCR2BUB: Output Compare Register2B Update Busy
When Timer/Counter2 operates asynchronously and OCR2B is written, this bit becomes set. When OCR2B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2B is ready to be updated with a new value.

Bit 1 – TCR2AUB: Timer/Counter Control Register2 Update Busy
When Timer/Counter2 operates asynchronously and TCCR2A is written, this bit becomes set. When TCCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2A is ready to be updated with a new value.

Bit 0 – TCR2BUB: Timer/Counter Control Register2 Update Busy
When Timer/Counter2 operates asynchronously and TCCR2B is written, this bit becomes set. When TCCR2B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2B is ready to be updated with a new value.

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

20.11.9 General Timer/Counter Control Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GTCCR
Offset: 0x43
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x23

Bit	7	6	5	4	3	2	1	0
	TSM						PSRASY	PSRSYNC
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – TSM: Timer/Counter Synchronization Mode
Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

Bit 1 – PSRASY: Prescaler Reset Timer/Counter2
When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

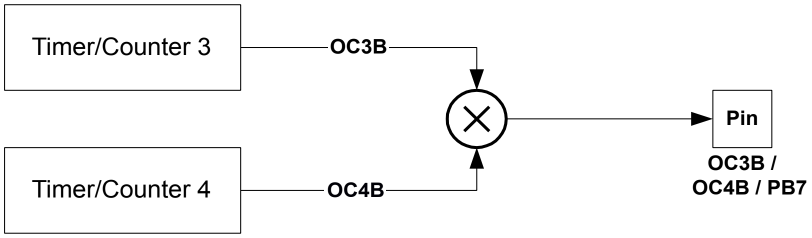
Bit 0 – PSRSYNC: Prescaler Reset
When this bit is one, Timer/Counter 0, 1, 3, 4 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter 0, 1, 3, 4 share the same prescaler and a reset of this prescaler will affect the mentioned timers.

21. OCM - Output Compare Modulator

21.1 Overview

The Output Compare Modulator (OCM) allows generation of waveforms modulated with a carrier frequency. The modulator uses the outputs from the Output Compare Unit B of the 16-bit Timer/Counter3 and the Output Compare Unit of the 16-bit Timer/Counter4. For more details about these Timer/Counters see *16-bit Timer/Counter*. When the modulator is enabled, the two output compare channels are modulated together as shown in the block diagram (as the following figure).

Figure 21-1. Output Compare Modulator, Block Diagram



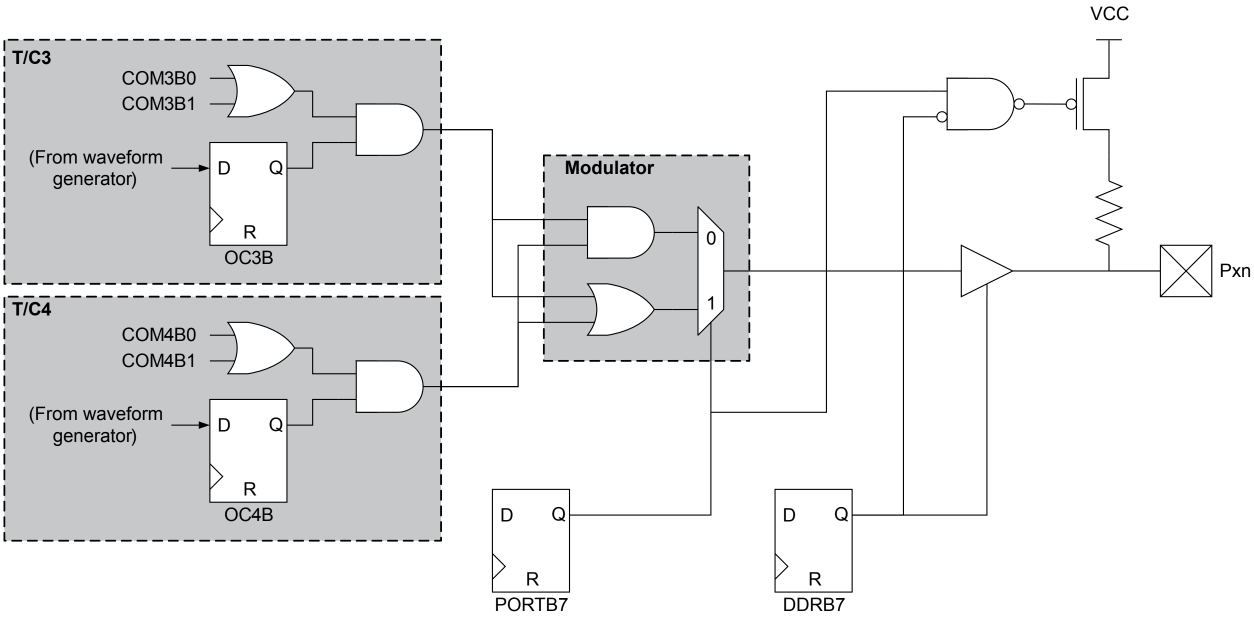
21.2 Description

The Output Compare unit 3B and Output Compare unit 4B shares the PB7 port pin for output. The outputs of the Output Compare units (OC3B and OC4B) overrides the normal PORTB7 Register when one of them is enabled (that is, when COMx[1:0] is not equal to zero). When both OC3B and OC4B are enabled at the same time, the modulator is automatically enabled.

The functional equivalent schematic of the modulator is shown on the figure below. The schematic includes part of the Timer/Counter units and the port B pin 7 output driver circuit. When the modulator is enabled the type of modulation (logical AND or OR) can be selected by setting the PORTB7 Register as '1'.

Note: The DDRB7 controls the direction of the port independent of the COMx[1:0] bit setting.

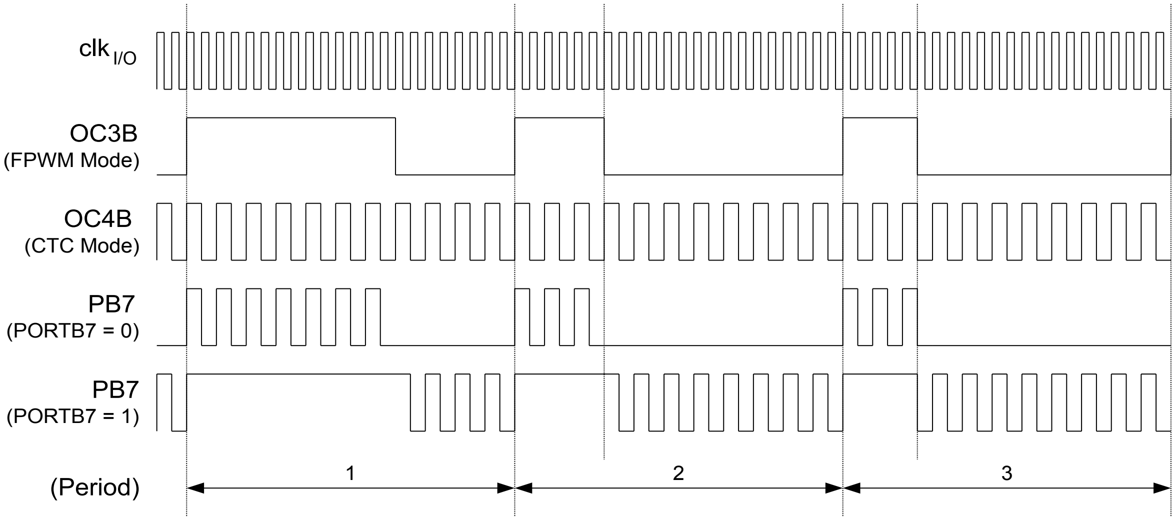
Figure 21-2. Output Compare Modulator, Schematic



21.2.1 Timing Example

The figure below illustrates the modulator in action. In this example the Timer/Counter3 is set to operate in fast PWM mode (non-inverted) and Timer/Counter4 uses CTC waveform mode with toggle Compare Output mode (COMx[1:0] = 0x1).

Figure 21-3. Output Compare Modulator, Timing Diagram



In this example, Timer/Counter4 provides the carrier, while the modulating signal is generated by the Output Compare unit B of the Timer/Counter3.

The resolution of the PWM signal (OC3B) is reduced by the modulation. The reduction factor is equal to the number of system clock cycles of one period of the carrier (OC4B). In this example the resolution is reduced by a factor of two. The reason for the reduction is illustrated in the above figure at the second and third period of the PB7 output when PORTB7 equals zero. The period 2 high time is one cycle longer than the period 3 high time, but the result on the PB7 output is equal in both periods.

22. SPI – Serial Peripheral Interface

22.1 Features

- Two SPIs are available - SPI0 and SPI1
- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

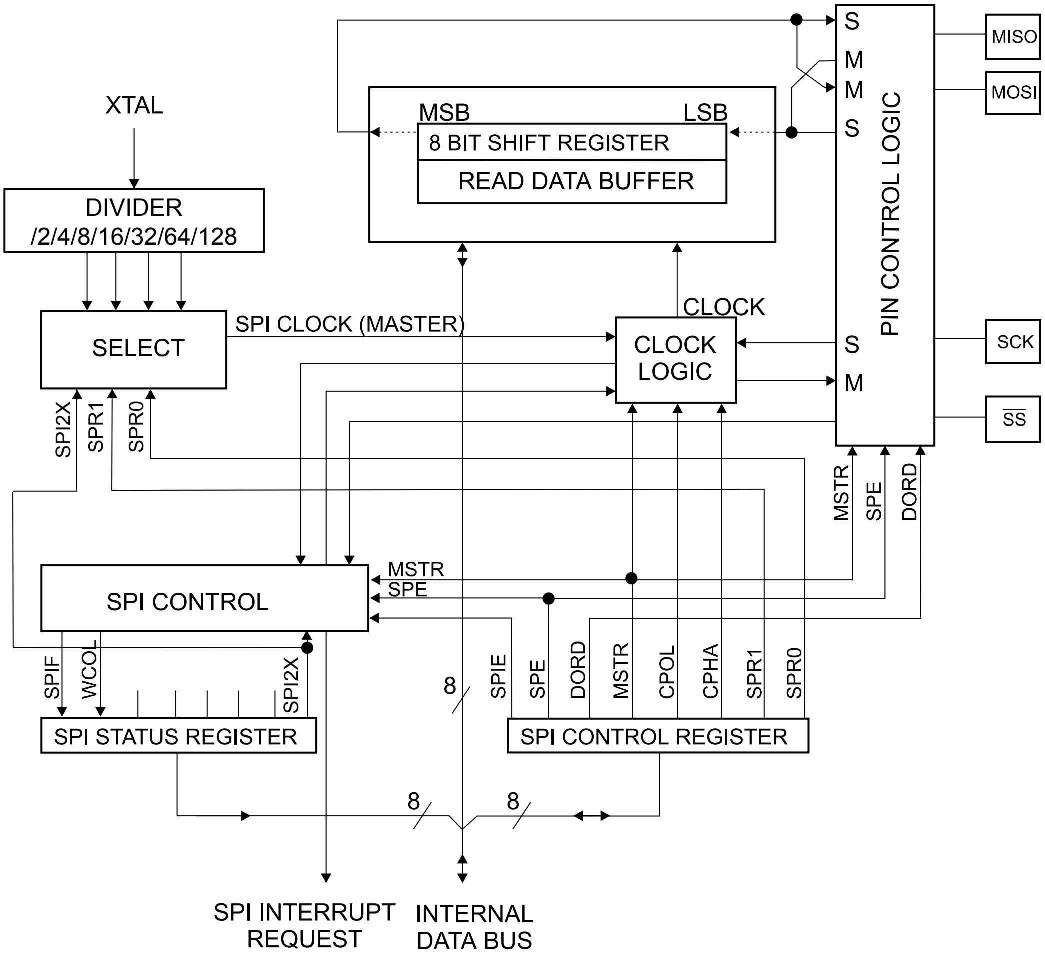
22.2 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the device and peripheral units, or between several AVR devices.

The USART can also be used in Master SPI mode, please refer to *USART in SPI Mode* chapter.

To enable the SPI module, Power Reduction Serial Peripheral Interface bit in the Power Reduction Register (PRR0.PRSPI0and PRR1.PRSPI1) must be written to '0'.

Figure 22-1. SPI Block Diagram



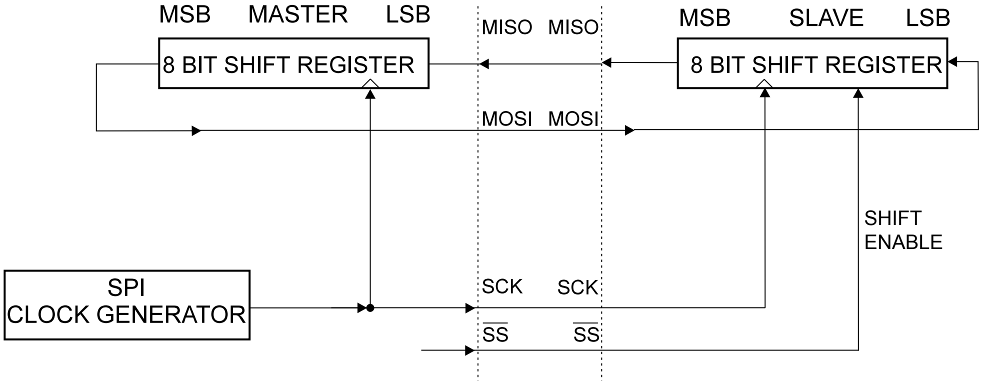
Note: Refer to the pin-out description and the IO Port description for SPI pin placement.

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

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

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

Figure 22-2. SPI Master-slave Interconnection



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

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

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{\text{SS}}$ pins is overridden according to the table below. For more details on automatic port overrides, refer to the IO Port description.

Table 22-1. SPI Pin Overrides

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

Note: 1. See the IO Port description for how to define the SPI pin directions.

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

Assembly Code Example

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

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

C Code Example

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

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

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

Assembly Code Example

```
SPI_SlaveInit:
   ; Set MISO output, all others input
   ldi     r17,(1<<DD_MISO)
   out     DDR_SPI,r17
   ; Enable SPI
   ldi     r17,(1<<SPE)
   out     SPCR,r17
   ret
SPI_SlaveReceive:
   ; Wait for reception complete
   in      r16, SPSR
   sbrs    r16, SPIF
   rjmp SPI_SlaveReceive
   ; Read received data and return
   in      r16,SPDR
   ret
```

C Code Example

```
void SPI_SlaveInit(void)
{
   /* Set MISO output, all others input */
   DDR_SPI = (1<<DD_MISO);
   /* Enable SPI */
   SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
   /* Wait for reception complete */
   while(!(SPSR & (1<<SPIF)))
      ;
   /* Return Data Register */
   return SPDR;
}
```

Related Links

Pin Descriptions
USARTSPI - USART in SPI Mode
PM - Power Management and Sleep Modes
I/O-Ports
About Code Examples

22.3 $\overline{\text{SS}}$ Pin Functionality

22.3.1 Slave Mode

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

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

22.3.2 Master Mode

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

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

If $\overline{\text{SS}}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{\text{SS}}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ 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.

22.4 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. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. The following table, summarizes SPCR.CPOL and SPCR.CPHA settings.

Table 22-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing Edge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)

SPI Mode	Conditions	Leading Edge	Trailing Edge
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

The SPI data transfer formats are shown in the following figure.

Figure 22-3. SPI Transfer Format with CPHA = 0

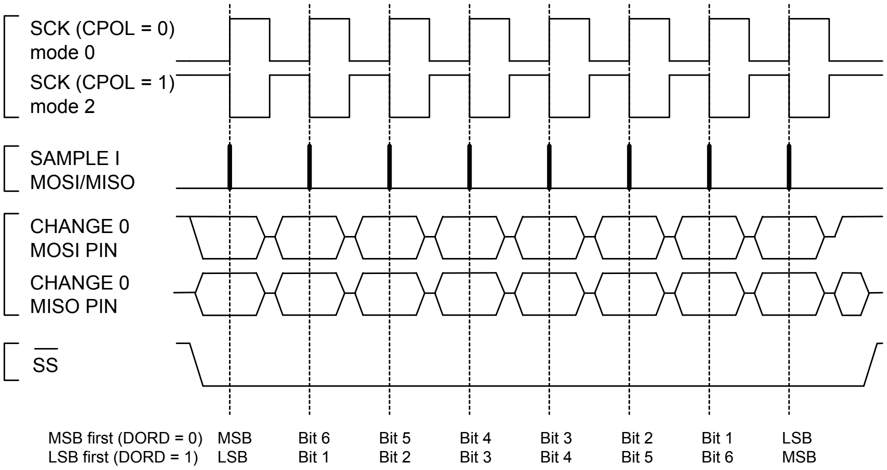
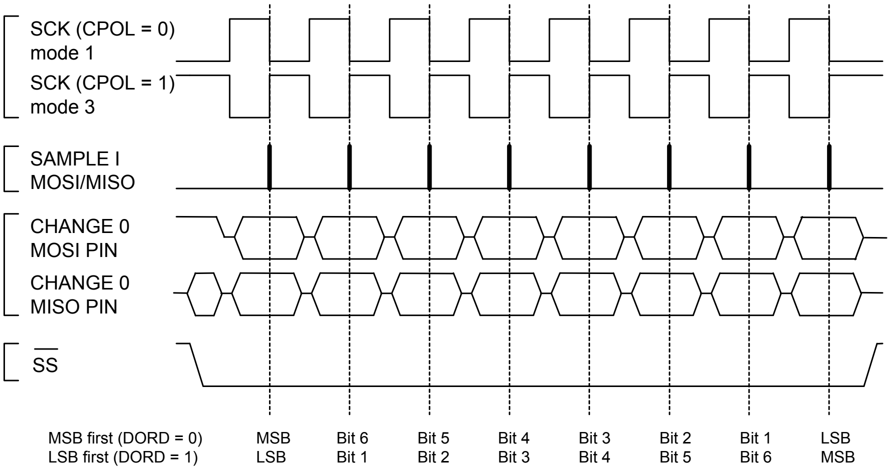


Figure 22-4. SPI Transfer Format with CPHA = 1



22.5 Register Description

22.5.1 SPI Control Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPCR0
Offset: 0x4C [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2C

Bit	7	6	5	4	3	2	1	0
	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR0 [1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPIE0: SPI0 Interrupt Enable
This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

Bit 6 – SPE0: SPI0 Enable
When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 – DORD0: Data0 Order
When the DORD bit is written to one, the LSB of the data word is transmitted first.

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

Bit 4 – MSTR0: Master/Slave0 Select
This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If 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 – CPOL0: Clock0 Polarity
When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 22-3 and Figure 22-4 for an example. The CPOL functionality is summarized below:

Table 22-3. CPOL0 Functionality

CPOL0	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Bit 2 – CPHA0: Clock0 Phase
The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 22-3 and Figure 22-4 for an example. The CPHA functionality is summarized below:

Table 22-4. CPHA0 Functionality

CPHA0	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Bits 1:0 – SPR0 [1:0]: SPI0 Clock Rate Select
These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the table below.

Table 22-5. Relationship between SCK and Oscillator Frequency

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

22.5.2 SPI Control Register 1

Name: SPCR1
Offset: 0xAC [ID-000004d0]
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	SPIE1	SPE1	DORD1	MSTR1	CPOL1	CPHA1	SPR1 [1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPIE1: SPI1 Interrupt Enable
This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

Bit 6 – SPE1: SPI1 Enable
When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 – DORD1: Data1 Order
When the DORD bit is written to one, the LSB of the data word is transmitted first.

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

Bit 4 – MSTR1: Master/Slave1 Select
This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If 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 – CPOL1: Clock1 Polarity
When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 22-3 and Figure 22-4 for an example. The CPOL functionality is summarized below:

Table 22-6. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Bit 2 – CPHA1: Clock1 Phase

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 22-3 and Figure 22-4 for an example. The CPHA functionality is summarized below:

Table 22-7. CPHA1 Functionality

CPHA1	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Bits 1:0 – SPR1 [1:0]: SPI1 Clock Rate Select

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

Table 22-8. Relationship between SCK and Oscillator Frequency

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

22.5.3 SPI Status Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPSR0
Offset: 0x4D [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2D

Bit	7	6	5	4	3	2	1	0
	SPIF0	WCOL0						SPI2X0
Access	R	R						R/W
Reset	0	0						0

Bit 7 – SPIF0: SPI Interrupt Flag
When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Bit 6 – WCOL0: Write Collision Flag
The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

Bit 0 – SPI2X0: Double SPI Speed Bit
When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (refer to Table 22-5). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower.

The SPI interface is also used for program memory and EEPROM downloading or uploading. See *Serial Downloading* for serial programming and verification.

Related Links

Serial Downloading

22.5.4 SPI Status Register 1

Name: SPSR1
Offset: 0xAD [ID-000004d0]
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	SPIF1	WCOL1						SPI2X1
Access	R	R						R/W
Reset	0	0						0

Bit 7 – SPIF1: SPI Interrupt Flag 1
When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Bit 6 – WCOL1: Write Collision Flag 1
The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

Bit 0 – SPI2X1: Double SPI Speed Bit 1
When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (refer to Table 22-5). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower.

The SPI interface is also used for program memory and EEPROM downloading or uploading. See *Serial Downloading* for serial programming and verification.

22.5.5 SPI Data Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPDR0
Offset: 0x4E [ID-000004d0]
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x2E

Bit	7	6	5	4	3	2	1	0
	SPID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SPID[7:0]: SPI Data
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.

22.5.6 SPI Data Register 1

Name: SPDR1
Offset: 0xAE [ID-000004d0]
Reset: 0xXX

Bit	7	6	5	4	3	2	1	0
	SPID1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SPID1[7:0]: SPI Data 1
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.

23. USART - Universal Synchronous Asynchronous Receiver Transceiver

23.1 Features

- Three USART instances USART0, USART1, USART2
- 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
- Start Frame Detection

23.2 Overview

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

The USART can also be used in Master SPI mode. The Power Reduction USART bit in the Power Reduction Register (PRR.PRUSARTn) must be written to '0' in order to enable USARTn. USART 0 and 1 are in PRR0, and USART 2 is in PRR2.

Related Links

USARTSPI - USART in SPI Mode

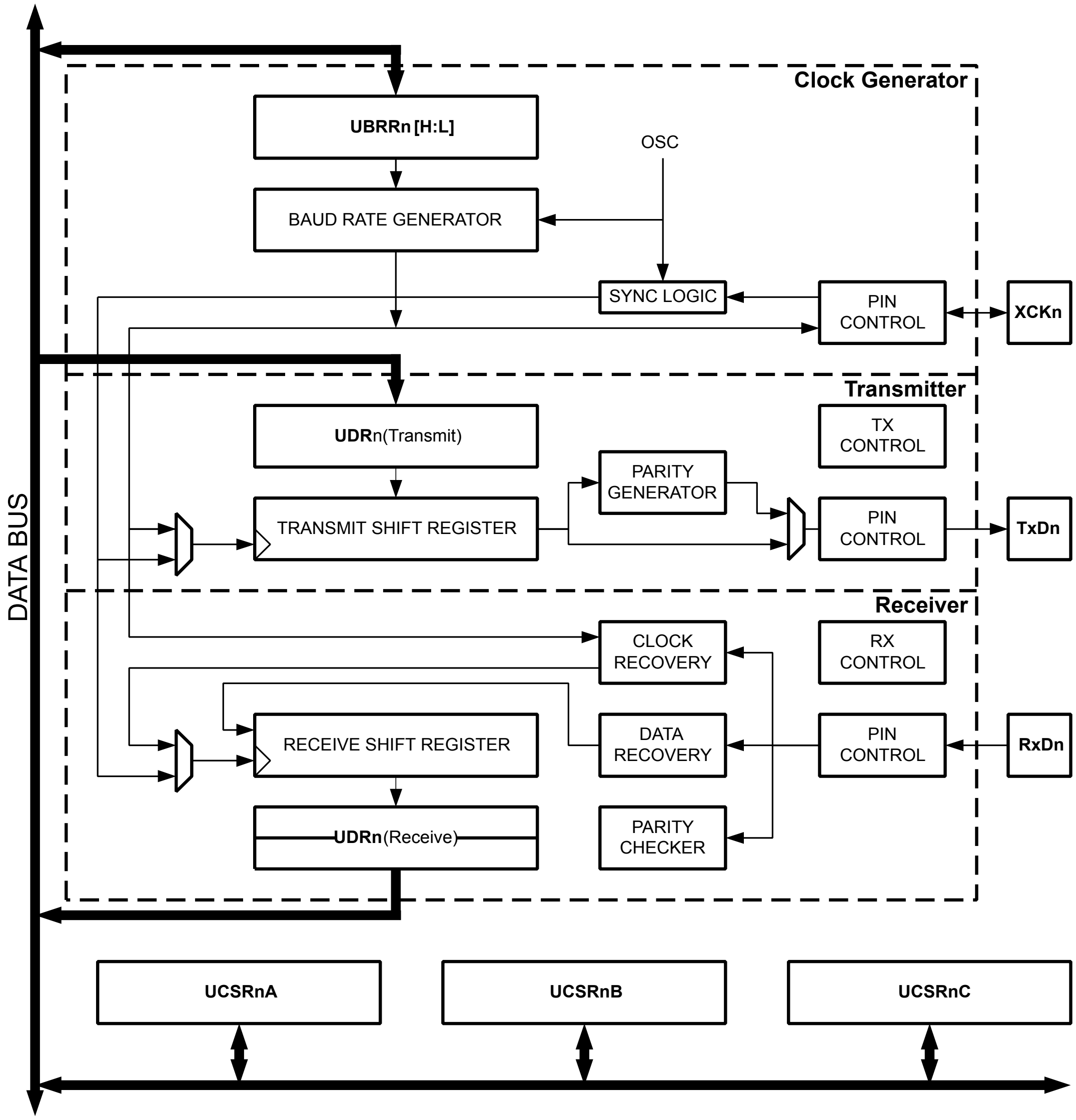
I/O-Ports

Pin Configurations

23.3 Block Diagram

In the USART Block Diagram, the CPU accessible I/O Registers and I/O pins are shown in bold. The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter, and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCKn (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 (UDRn). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun, and Parity Errors.

Figure 23-1. USART Block Diagram



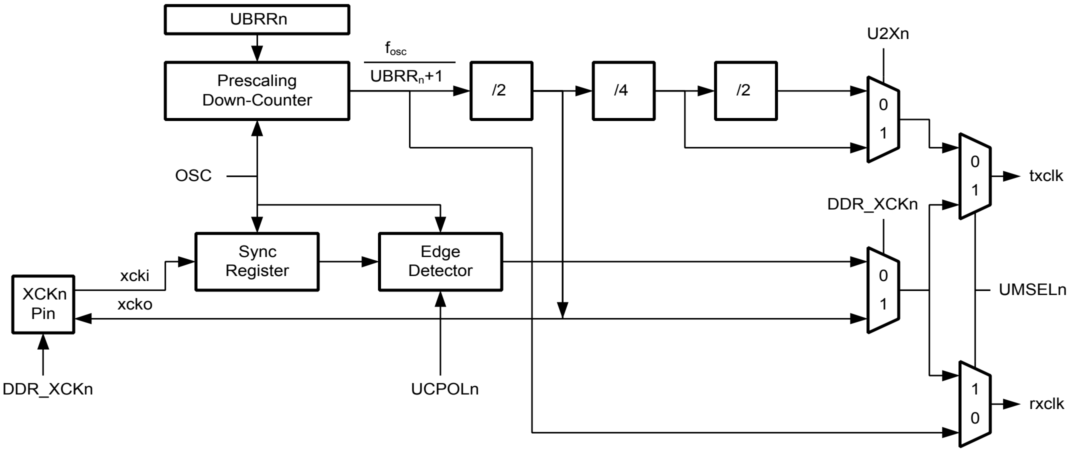
Note: Refer to the *Pin Configurations* and the *I/O-Ports* description for USART pin placement.

23.4 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 USART Mode Select bit 0 in the USART Control and Status Register n C (UCSRnC.UMSELn0) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2X found in the UCSRnA Register. When using synchronous mode (UMSELn0=1), the Data Direction Register for the XCKn pin (DDR_XCKn) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCKn pin is only active when using synchronous mode.

Below is a block diagram of the clock generation logic.

Figure 23-2. Clock Generation Logic, Block Diagram



Signal description:

- txclk: Transmitter clock (internal signal).
- rxclk: Receiver base clock (internal signal).
- xcki: Input from XCKn pin (internal signal). Used for synchronous slave operation.
- xcko: Clock output to XCKn pin (internal signal). Used for synchronous master operation.
- f_{OSC}: System clock frequency.

23.4.1 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 the Clock Generation Logic block diagram in the previous section..

The USART Baud Rate Register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{OSC}), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRnL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{OSC}/(UBRRn+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.

The table below contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 23-1. Equations for Calculating Baud Rate Register Setting

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

Note: 1. The baud rate is defined to be the transfer rate in bits per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{OSC}	System oscillator clock frequency
UBRRn	Contents of the UBRRnH and UBRRnL Registers, (0-4095). Some examples of UBRRn values for some system clock frequencies are found in Examples of Baud Rate Settings.

23.4.2 Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRnA. 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. However, in this case, the Receiver will 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.

23.4.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to the Clock Generation Logic block diagram in the previous section.

External clock input from the XCKn 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 XCKn clock frequency is limited by the following equation:

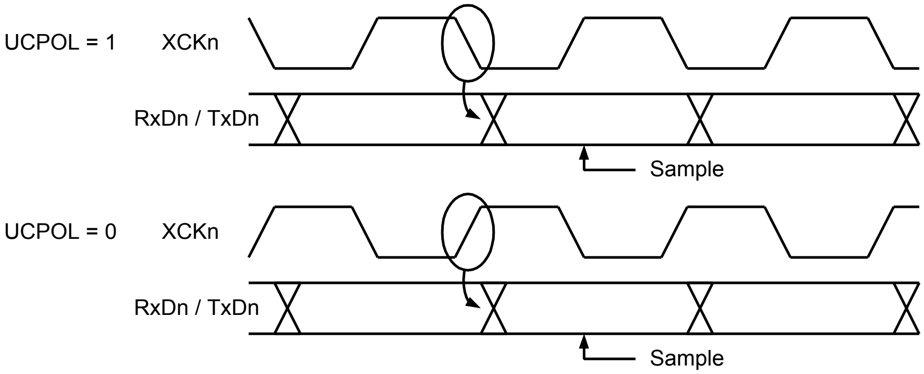
$$f_{\mathrm{XCKn}} < \frac{f_{\mathrm{OSC}}}{4}$$

The value of 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.

23.4.4 Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCKn 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 RxDn) is sampled at the opposite XCKn clock edge of the edge the data output (TxDn) is changed.

Figure 23-3. Synchronous Mode XCKn Timing



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

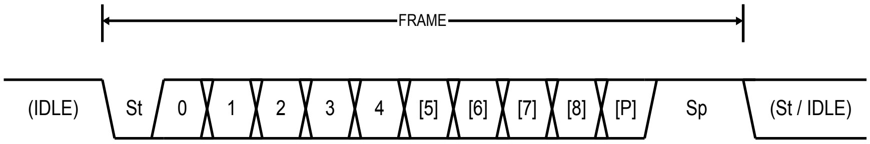
23.5 Frame Formats

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

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

A frame starts with the start bit, followed by the data bits (from five up to nine data bits in total): first the least significant data bit, then the next data bits ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the one or two stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. the figure below illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 23-4. 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 (RxDn or TxDn). An IDLE line must be high.

The frame format used by the USART is set by:

- Character Size bits (UCSRnC.UCSZn[2:0]) select the number of data bits in the frame.
- Parity Mode bits (UCSRnC.UPMn[1:0]) enable and set the type of parity bit.
- Stop Bit Select bit (UCSRnC.USBSn) select the number of stop bits. The Receiver ignores the second stop bit.

The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter. An FE (Frame Error) will only be detected in cases where the first stop bit is zero.

23.5.1 Parity Bit Calculation

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

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

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

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.

23.6 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and 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 (UCSRnA.TXC) 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. The UCSRnA.TXC must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17, r16 Registers.

Assembly Code Example

```
USART_Init:
   ; Set baud rate to UBRR0
   out    UBRR0H, r17
   out    UBRR0L, r16
   ; Enable receiver and transmitter
   ldi    r16, (1<<RXEN0)|(1<<TXEN0)
   out    UCSR0B,r16
   ; Set frame format: 8data, 2stop bit
   ldi    r16, (1<<USBS0)|(3<<UCSZ00)
   out    UCSR0C,r16
   ret
```

C Code Example

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
   ...
   USART_Init(MYUBRR)
   ...
}
void USART_Init( unsigned int ubrr)
{
   /*Set baud rate */
   UBRR0H = (unsigned char)(ubrr>>8);
   UBRR0L = (unsigned char)ubrr;
   Enable receiver and transmitter */
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   /* Set frame format: 8data, 2stop bit */
```

```
    UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

More advanced initialization routines can be written to 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.

Related Links

About Code Examples

23.7 Data Transmission – The USART Transmitter

The USART Transmitter is enabled by setting the Transmit Enable (TXEN) bit in the UCSRnB Register. When the Transmitter is enabled, the normal port operation of the TxDn 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 XCKn pin will be overridden and used as transmission clock.

23.7.1 Sending Frames with 5 to 8 Data Bits

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDRn 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 XCKn depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the Data Register Empty (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR0 are ignored. The USART 0 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 R17.

Assembly Code Example

```
USART_Transmit:
   ; Wait for empty transmit buffer
   in       r17, UCSR0A
   sbrs     r17, UDRE
   rjmp     USART_Transmit
   ; Put data (r16) into buffer, sends the data
   out     UDR0,r16
   ret
```

C Code Example

```
void USART_Transmit( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSR0A & (1<<UDRE)) )
      ;
   /* Put data into buffer, sends the data */
   UDR0 = data;
}
```

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.

Related Links

About Code Examples

23.7.2 Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZn = 7), the ninth bit must be written to the TXB8 bit in UCSRnB before the low byte of the character is written to UDRn.

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.

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

```
USART_Transmit:
   ; Wait for empty transmit buffer
   in      r18, UCSR0A
   sbrs    r18, UDRE
   rjmp    USART_Transmit
   ; Copy 9th bit from r17 to TXB8
   cbi     UCSR0B,TXB8
   sbrc    r17,0
   sbi     UCSR0B,TXB8
   ; Put LSB data (r16) into buffer, sends the data
   out    UDR0,r16
   ret
```

C Code Example

```
void USART_Transmit( unsigned int data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSR0A & (1<<UDRE))) )
      ;
   /* Copy 9th bit to TXB8 */
   UCSR0B &= ~(1<<TXB8);
   if ( data & 0x0100 )
      UCSR0B |= (1<<TXB8);
   /* Put data into buffer, sends the data */
   UDR0 = data;
}
```

Note: These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB Register is used after initialization.

Related Links

About Code Examples

23.7.3 Transmitter Flags and Interrupts

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

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

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRnB is written to '1', 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 UDRn. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDRn in order to clear 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 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 either automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a '1' to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRnB is written to '1', 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.

23.7.4 Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UCSRnC.UPM[1]=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.

23.7.5 Disabling the Transmitter

When writing the TX Enable bit in the USART Control and Status Register n B (UCSRnB.TXEN) to zero, the disabling of the Transmitter will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn pin.

23.8 Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRnB Register to '1'. When the Receiver is enabled, the normal pin operation of the RxDn 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 XCKn pin will be used as transfer clock.

23.8.1 Receiving Frames with 5 to 8 Data Bits

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCKn 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 UDRn 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 UDR0 will be masked to zero. The USART 0 has to be initialized before the function can be used. For the assembly code, the received data will be stored in R16 after the code completes.

Assembly Code Example

```
USART_Receive:
   ; Wait for data to be received
   in    r17, UCSR0A
   sbrs  r17, RXC
   rjmp  USART_Receive
   ; Get and return received data from buffer
   in    r16, UDR0
   ret
```

C Code Example

```
unsigned char USART_Receive( void )
{
   /* Wait for data to be received */
   while ( !(UCSR0A & (1<<RXC)) )
      ;
   /* Get and return received data from buffer */
   return UDR0;
}
```

For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The 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.

Related Links

About Code Examples

23.8.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZn=7) the ninth bit must be read from the RXB8 bit in UCSRnB before reading the low bits from the UDRn. This rule applies to the FE, DOR and UPE Status Flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and UPE bits, which all are stored in the FIFO, will change.

The following code example shows a simple receive function for USART0 that handles both nine bit characters and the status bits. For the assembly code, the received data will be stored in R17:R16 after the code completes.

Assembly Code Example

```
USART_Receive:
   ; Wait for data to be received
   in     r16, UCSR0A
   sbrs   r16, RXC
   rjmp   USART_Receive
   ; Get status and 9th bit, then data from buffer
   in     r18, UCSR0A
   in     r17, UCSR0B
   in     r16, UDR0
   ; If error, return -1
   andi   r18,(1<<FE)|(1<<DOR)|(1<<UPE)
   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 ( !(UCSR0A & (1<<RXC)) )
      ;
   /* Get status and 9th bit, then data */
   /* from buffer */
   status = UCSR0A;
   resh = UCSR0B;
   resl = UDR0;
   /* If error, return -1 */
   if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
      return -1;
   /* Filter the 9th bit, then return */
   resh = (resh >> 1) & 0x01;
   return ((resh << 8) | resl);
}
```

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.

Related Links

About Code Examples

23.8.3 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 RXCn bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRnB 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.

23.8.4 Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (UPE). All can be accessed by reading UCSRnA. 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 UCSRnA must be read before the receive buffer (UDRn), since reading the UDRn 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 UCSRnA 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 '1', 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 UCSRnC 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 UCSRnA.

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), a new character is waiting in the Receive Shift Register, and a new start bit is detected. If the DOR Flag is set, one or more serial frames were lost between the last frame read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRnA. The DOR Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPE) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPE bit will always read '0'. For compatibility with future devices, always set this bit to zero when writing to UCSRnA. For more details see Parity Bit Calculation and 'Parity Checker' below.

23.8.5 Parity Checker

The Parity Checker is active when the high USART Parity Mode bit 1 in the USART Control and Status Register n C (UCSRnC.UPM[1]) is written to '1'. The type of Parity Check to be performed (odd or even) is selected by the UCSRnC.UPM[0] 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 USART Parity Error Flag in the USART Control and Status Register n A (UCSRnA.UPE) can then be read by software to check if the frame had a Parity Error.

The UPEn 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 (UPM[1] = 1). This bit is valid until the receive buffer (UDRn) is read.

23.8.6 Disabling the Receiver

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

23.8.7 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 UDRn I/O location until the RXCn Flag is cleared.

The following code shows how to flush the receive buffer of USART0.

Assembly Code Example

```
USART_Flush:
   in       r16, UCSR0A
   sbrs     r16, RXC
   ret
   in       r16, UDR0
   rjmp     USART_Flush
```

C Code Example

```
void USART_Flush( void )
{
   unsigned char dummy;
```

```
    while ( UCSR0A & (1<<RXC) ) dummy = UDR0;
}
```

Related Links

About Code Examples

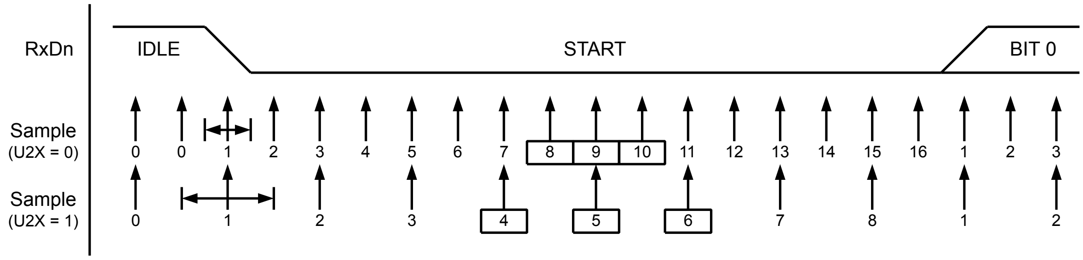
23.9 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxDn pin. The data recovery logic samples and low pass filters each incoming bit, 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.

23.9.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. The figure below 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 (UCSRnA.U2X=1) of operation. Samples denoted '0' are samples taken while the RxDn line is idle (i.e., no communication activity).

Figure 23-5. Start Bit Sampling

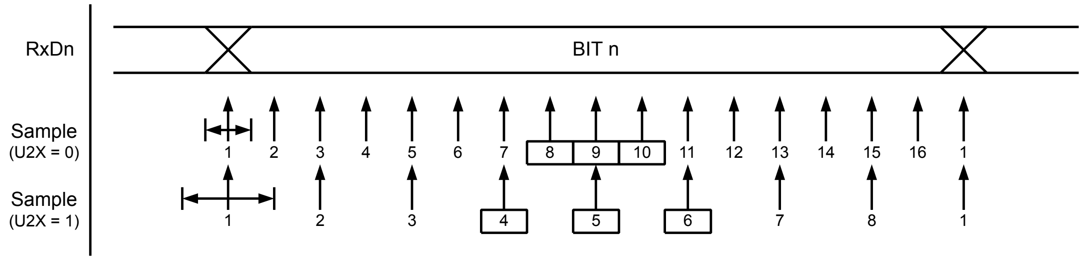


When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 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 on RxDn. 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.

23.9.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. The figure below 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.

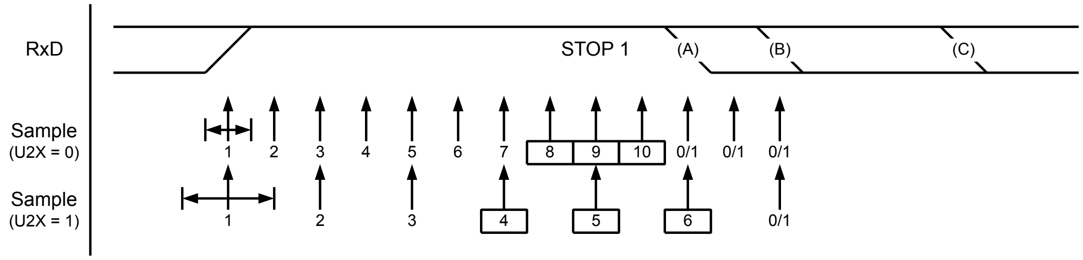
Figure 23-6. Sampling of Data and Parity Bit



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: If two or all three center samples (those marked by their sample number inside boxes) have high levels, the received bit is registered to be a logic '1'. If two or all three samples have low levels, the received bit is registered to be a logic '0'. This majority voting process acts as a low pass filter for the incoming signal on the RxDn pin. The recovery process is then repeated until a complete frame is received, including the first stop bit. The Receiver only uses the first stop bit of a frame.

The following figure shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 23-7. 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 (UCSRnA.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 taken at point marked (A) in the figure above. 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.

23.9.3 Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar base frequency (see recommendations below), 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.

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

- *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.

The following tables list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 23-2. 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 23-3. 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 (EXTCLK) 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 resonator's 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 UBRRn value that gives an acceptable low error can be used if possible.

23.9.4 Start Frame Detection

The USART start frame detector can wake up the MCU from Power-down and Standby sleep mode when it detects a start bit.

When a high-to-low transition is detected on RxDn, the internal 8MHz oscillator is powered up and the USART clock is enabled. After start-up the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the internal 8MHz oscillator start-up time. Start-up time of the internal 8MHz oscillator varies with supply voltage and temperature.

The USART start frame detection works both in asynchronous and synchronous modes. It is enabled by writing the Start Frame Detection Enable bit (SFDE). If the USART Start Interrupt Enable (RXSIE) bit is set, the USART Receive Start Interrupt is generated immediately when start is detected.

When using the feature without start interrupt, the start detection logic activates the internal 8MHz oscillator and the USART clock while the frame is being received, only. Other clocks remain stopped until the Receive Complete Interrupt wakes up the MCU.

The maximum baud rate in synchronous mode depends on the sleep mode the device is woken up from, as follows:

- Idle sleep mode: system clock frequency divided by four
- Standby or Power-down: 500kbps

The maximum baud rate in asynchronous mode depends on the sleep mode the device is woken up from, as follows:

- Idle sleep mode: the same as in active mode

Table 23-4. Maximum Total Baudrate Error in Normal Speed Mode

Baudrate	Frame Size					
	5	6	7	8	9	10
0 - 28.8kbps	+6.67 -5.88	+5.79 -5.08	+5.11 -4.48	+4.58 -4.00	+4.14 -3.61	+3.78 -3.30
38.4kbps	+6.63 -5.88	+5.75 -5.08	+5.08 -4.48	+4.55 -4.00	+4.12 -3.61	+3.76 -3.30
57.6kbps	+6.10 -5.88	+5.30 -5.08	+4.69 -4.48	+4.20 -4.00	+3.80 -3.61	+3.47 -3.30
76.8kbps	+5.59 -5.88	+4.85 -5.08	+4.29 -4.48	+3.85 -4.00	+3.48 -3.61	+3.18 -3.30
115.2kbps	+4.57 -5.88	+3.97 -5.08	+3.51 -4.48	+3.15 -4.00	+2.86 -3.61	+2.61 -3.30

Table 23-5. Maximum Total Baudrate Error in Double Speed Mode

Baudrate	Frame Size					
	5	6	7	8	9	10
0 - 57.6kbps	+5.66 -4.00	+4.92 -3.45	+4.35 -3.03	+3.90 -2.70	+3.53 -2.44	+3.23 -2.22
76.8kbps	+5.59 -4.00	+4.85 -3.45	+4.29 -3.03	+3.85 -2.70	+3.48 -2.44	+3.18 -2.22
115.2kbps	+4.57 -4.00	+3.97 -3.45	+3.51 -3.03	+3.15 -2.70	+2.86 -2.44	+2.61 -2.22

23.10 Multi-Processor Communication Mode

Setting the Multi-Processor Communication mode (MPCMn) bit in UCSRnA 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 MPCMn setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the Receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the Receiver is set up for frames with 9 data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is '1', the frame contains an address. When the frame type bit is '0', 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.

23.10.1 Using MPCMn

For an MCU to act as a master MCU, it can use a 9-bit character frame format (UCSZ1=7). The ninth bit (TXB8) must be set when an address frame (TXB8=1) or cleared when a data frame (TXB=0) is being transmitted. The slave MCUs must in this case be set to use a 9-bit character frame format.

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

1. All Slave MCUs are in Multi-Processor Communication mode (MPCM in UCSRnA 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 UCSRnA will be set as normal.
3. Each Slave MCU reads the UDRn Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRnA, 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 step 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.

23.11 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 UBRRn settings as listed in the table below.

UBRRn 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 also section *Asynchronous Operational Range*). The error values are calculated using the following equation:

$$Error\left[\%\right] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right)^2 \ 100\,\%$$

Table 23-6. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max.(1)	62.5kbps		125kbps		115.2kbps		230.4kbps		125kbps		250kbps	

Note: 1. UBRRn = 0, Error = 0.0%

Table 23-7. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 3.6864MHz				f_{osc} = 4.0000MHz				f_{osc} = 7.3728MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%

Baud Rate [bps]	f_{osc} = 3.6864MHz				f_{osc} = 4.0000MHz				f_{osc} = 7.3728MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max.(1)	230.4kbps		460.8kbps		250kbps		0.5Mbps		460.8kbps		921.6kbps	

(1) UBRRn = 0, Error = 0.0%

Table 23-8. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max.(1)	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

(1) UBRRn = 0, Error = 0.0%

Table 23-9. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

(1) UBRRn = 0, Error = 0.0%

Related Links

Asynchronous Operational Range

23.12 Register Description

23.12.1 USART I/O Data Register n

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 UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR1 Register location. Reading the UDRn 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 UCSRnA Register is set. Data written to UDRn when the UCSRnA.UDRE Flag is not set, will be ignored by the USART Transmitter n. 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 TxDn 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.

Name: UDR
Offset: 0xC6 + n*0x08 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TXB / RXB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TXB / RXB[7:0]: USART Transmit / Receive Data Buffer

23.12.2 USARTn Control and Status Register A

Name: UCSR0A, UCSR1A, UCSR2A
Offset: 0xC0 + n*0x08 [n=0..2]
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC: USART Receive Complete
This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

Bit 6 – TXC: USART Transmit Complete
This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDRn). 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 (UDRn) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register Empty interrupt (see description of the UDRIE bit). UDRE is set after a reset to indicate that the Transmitter is ready.

Bit 4 – FE: Frame Error
This bit is set if the next character in the receive buffer had a Frame Error when received. 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 (UDRn) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRnA.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 3 – DOR: Data OverRun
This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 2 – UPE: USART Parity Error
This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UCSRnC.UPM1 = 1). This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 1 – U2X: Double the USART Transmission Speed
This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

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

This bit is reserved in Master SPI Mode (MSPIM).

Bit 0 – MPCM: Multi-processor Communication Mode
This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver n that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. Refer to Multi-Processor Communication Mode for details.

This bit is reserved in Master SPI Mode (MSPIM).

23.12.3 USART Control and Status Register n B

Name: UCSR0B, UCSR1B, UCSR2B
Offset: 0xC1 + n*0x08 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RXCIE: RX Complete Interrupt Enable
Writing this bit to one enables interrupt on the UCSRnA.RXC Flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRnA is set.

Bit 6 – TXCIE: TX Complete Interrupt Enable
Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRnA is set.

Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable
Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRnA is set.

Bit 4 – RXEN: Receiver Enable
Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxDn pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and UPE Flags.

Bit 3 – TXEN: Transmitter Enable
Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxDn pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn port.

Bit 2 – UCSZ2: Character Size
The UCSZ2 bits combined with the UCSZ[1:0] bit in UCSRnC sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 1 – RXB8: Receive Data Bit 8
RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDRn.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 0 – TXB8: Transmit Data Bit 8
TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDRn.

This bit is reserved in Master SPI Mode (MSPIM).

23.12.4 USART Control and Status Register n C

Name: UCSR0C, UCSR1C, UCSR2C
Offset: 0xC2 + n*0x08 [n=0..2]
Reset: 0x06
Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bits 7:6 – UMSEL[1:0]: USART Mode Select
These bits select the mode of operation of the USARTn

Table 23-10. USART Mode Selection

UMSEL[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

Note:
1. The UDORD, UCPHA, and UCPOL can be set in the same write operation where the MSPIM is enabled.

Bits 5:4 – UPM[1:0]: USART Parity Mode
These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM setting. If a mismatch is detected, the UPE Flag in UCSRnA will be set.

Table 23-11. USART Mode Selection

UPM[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

These bits are reserved in Master SPI Mode (MSPIM).

Bit 3 – USBS: USART Stop Bit Select
This bit selects the number of stop bits to be inserted by the Transmitter n. The Receiver ignores this setting.

Table 23-12. Stop Bit Settings

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

This bit is reserved in Master SPI Mode (MSPIM).

Bit 2 – UCSZ1 / UDORD: USART Character Size / Data Order
UCSZ1[1:0]: USART Modes: The UCSZ1[1:0] bits combined with the UCSZ12 bit in UCSR1B sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 23-13. Character Size Settings

UCSZ1[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

UDORD0: Master SPI Mode: When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to the *USART in SPI Mode - Frame Formats* for details.

Bit 1 – UCSZ0 / UCPHA: USART Character Size / Clock Phase
UCSZ0: USART Modes: Refer to UCSZ1.

UCPHA: Master SPI Mode: The UCPHA bit setting determine if data is sampled on the leasing edge (first) or tailing (last) edge of XCK. Refer to the *SPI Data Modes and Timing* for details.

Bit 0 – UCPOL: Clock Polarity
USART n Modes: This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 23-14. USART Clock Polarity Settings

UCPOL	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn Edge	Falling XCKn Edge
1	Falling XCKn Edge	Rising XCKn Edge

Master SPI Mode: The UCPOL bit sets the polarity of the XCKn clock. The combination of the UCPOL and UCPHA bit settings determine the timing of the data transfer. Refer to the *SPI Data Modes and Timing* for details.

23.12.5 USART Control and Status Register n D

This register is not used in Master SPI Mode (`UCSRnC.UMSEL[1:0] = 11`)

Name: UCSR0D, UCSR1D, UCSR2D
Offset: 0xC3 + n*0x08 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXSIE	RXS	SFDE					
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit 7 – RXSIE: USART RX Start Interrupt Enable
Writing this bit to one enables the interrupt on the RXS flag. In sleep modes this bit enables start frame detector that can wake up the MCU when a start condition is detected on the RxD line. The USART RX Start Interrupt is generated only, if the RXSIE bit, the Global Interrupt flag, and RXS are set.

Bit 6 – RXS: USART RX Start
The RXS flag is set when a start condition is detected on the RxD line. If the RXSIE bit and the Global Interrupt Enable flag are set, an RX Start Interrupt will be generated when the flag is set. The flag can only be cleared by writing a logical one on the RXS bit location.

If the start frame detector is enabled (RXSIE = 1) and the Global Interrupt Enable flag is set, the RX Start Interrupt will wake up the MCU from all sleep modes.

Bit 5 – SFDE: Start Frame Detection Enable
Writing this bit to one enables the USART Start Frame Detection mode. The start frame detector is able to wake up the MCU from sleep mode when a start condition, i.e. a high (IDLE) to low (START) transition, is detected on the RxD line.

Table 23-15. USART Start Frame Detection Modes

SFDE	RXSIE	RXCIE	Description
0	X	X	Start frame detector disabled
1	0	0	Reserved
1	0	1	Start frame detector enabled. RXC flag wakes up MCU from all sleep modes
1	1	0	Start frame detector enabled. RXS flag wakes up MCU from all sleep modes
1	1	1	Start frame detector enabled. Both RXC and RXS wake up the MCU from all sleep modes

23.12.6 USART Baud Rate n Register Low and High byte

The UBRRnL and UBRRnH register pair represents the 16-bit value, UBRRn (n=0,1,2).The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: UBRR0L and UBRR0H, UBRR1L and UBRR1H, UBRR2L and UBRR2H
Offset: 0xC4 + n*0x08 [n=0..2]
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
					UBRR[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	UBRR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 11:0 – UBRR[11:0]: USART Baud Rate
This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits and the UBRRnL contains the eight least significant bits of the USART n baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRnL will trigger an immediate update of the baud rate prescaler.

24. USARTSPI - USART in SPI Mode

24.1 Features

- Full Duplex, Three-wire Synchronous Data Transfer
- Master Operation
- Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)
- LSB First or MSB First Data Transfer (Configurable Data Order)
- Queued Operation (Double Buffered)
- High Resolution Baud Rate Generator
- High Speed Operation ($f_{XCKmax} = f_{CK}/2$)
- Flexible Interrupt Generation

24.2 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation.

Setting both UMSELn[1:0] bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

24.3 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The table below contains the equations for calculating the baud rate or UBRRn setting for Synchronous Master Mode.

Table 24-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Synchronous Master mode	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$

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
UBRRn	Contents of the UBRRnH and UBRRnL Registers, (0-4095)

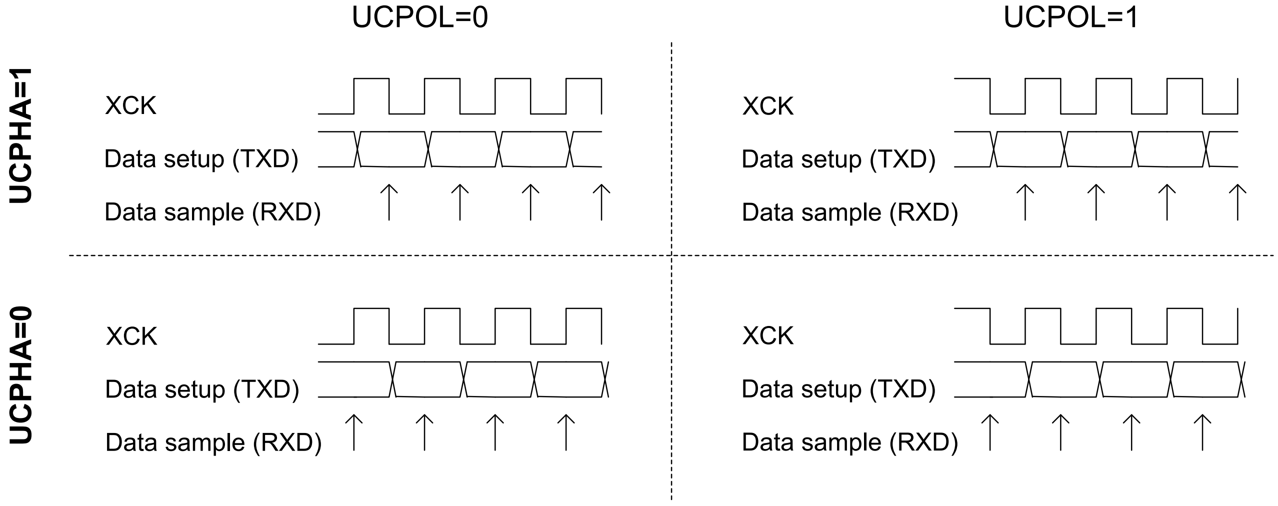
24.4 SPI Data Modes and Timing

There are four combinations of XCKn (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHAn and UCPOLn. The data transfer timing diagrams are shown in the following figure. Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAn functionality is summarized in the following table. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

Table 24-2. UCPOLn and UCPHAn Functionality

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
0	0	0	Sample (Rising)	Setup (Falling)
0	1	1	Setup (Rising)	Sample (Falling)
1	0	2	Sample (Falling)	Setup (Rising)
1	1	3	Setup (Falling)	Sample (Rising)

Figure 24-1. UCPHAn and UCPOLn data transfer timing diagrams.



24.5 Frame Formats

A serial frame for the MSPIM is defined to be one character of eight data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. Then the next data bits, up to a total of eight, are succeeding, ending with the most or least significant bit accordingly. When a complete frame is transmitted, a new frame can directly follow it, or the communication line can be set to an idle (high) state.

The UDORDn bit in UCSRnC sets the frame format used by the USART in MSPIM mode. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

24.5.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

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

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example

```
clr r18
out UBRRnH,r18
out UBRRnL,r18
; Setting the XCKn port pin as output, enables master mode.
sbi XCKn_DDR, XCKn
; Set MSPI mode of operation and SPI data mode 0.
ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
out UCSRnC,r18
; Enable receiver and transmitter.
ldi r18, (1<<RXENn)|(1<<TXENn)
out UCSRnB,r18
; Set baud rate.
; IMPORTANT: The Baud Rate must be set after the transmitter is enabled!
out UBRRnH, r17
out UBRRnL, r18
ret
```

C Code Example

```
{
 UBRRn = 0;
 /* Setting the XCKn port pin as output, enables master mode. */
 XCKn_DDR |= (1<<XCKn);
 /* Set MSPI mode of operation and SPI data mode 0. */
 UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
 /* Enable receiver and transmitter. */
 UCSRnB = (1<<RXENn)|(1<<TXENn);
 /* Set baud rate. */
```

```
  /* IMPORTANT: The Baud Rate must be set after the transmitter is enabled */
  UBRRn = baud;
}
```

Related Links

About Code Examples

24.6 Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, i.e. the TXENn bit in the UCSRnB register is set to one. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the Receiver's serial input. The XCKn will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDRn I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, the UDRn register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDREn) Flag and the Receive Complete (RXCn) Flag. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16 and the data received will be available in the same register (R16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDREn Flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

Assembly Code Example

```
USART_MSPIM_Transfer:
 ; Wait for empty transmit buffer
 in r16, UCSRnA
 sbrs r16, UDREn
 rjmp USART_MSPIM_Transfer
 ; Put data (r16) into buffer, sends the data
 out UDRn,r16
 ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
 in r16, UCSRnA
 sbrs r16, RXCn
 rjmp USART_MSPIM_Wait_RXCn
 ; Get and return received data from buffer
 in r16, UDRn
 ret
```

C Code Example

```
{
 /* Wait for empty transmit buffer */
 while ( !( UCSRnA & (1<<UDREn)) );
 /* Put data into buffer, sends the data */
 UDRn = data;
 /* Wait for data to be received */
 while ( !(UCSRnA & (1<<RXCn)) );
 /* Get and return received data from buffer */
 return UDRn;
}
```

Related Links

About Code Examples

24.6.1 Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREn flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and is always read as zero.

24.6.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

24.7 AVR USART MSPIM vs. AVR SPI

The USART in MSPIM mode is fully compatible with the AVR SPI regarding:

- Master mode timing diagram
- The UCPOLn bit functionality is identical to the SPI CPOL bit
- The UCPHAn bit functionality is identical to the SPI CPHA bit
- The UDORDn bit functionality is identical to the SPI DORD bit

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer
- The USART in MSPIM mode receiver includes an additional buffer level
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting UBRRn accordingly
- Interrupt timing is not compatible
- Pin control differs due to the master only operation of the USART in MSPIM mode

A comparison of the USART in MSPIM mode and the SPI pins is shown in the table below.

Table 24-3. Comparison of USART in MSPIM mode and SPI pins

USART_MSPIM	SPI	Comments
TxDn	MOSI	Master Out only
RxDn	MISO	Master In only

USART_MSPIM	SPI	Comments
XCKn	SCK	(Functionally identical)
(N/A)	$\overline{\text{SS}}$	Not supported by USART in MSPIM

24.8 Register Description

Refer to the USART Register Description.

Related Links

Register Description

25. TWI - 2-wire Serial Interface

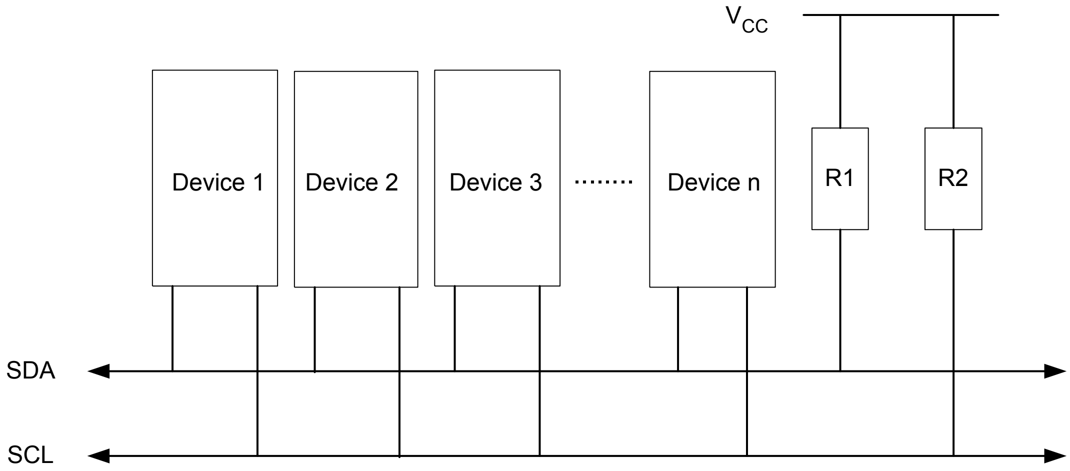
25.1 Features

- Two TWI instances TWI0 and TWI1
- Simple, yet Powerful and Flexible Communication Interface, only two Bus Lines Needed
- Both Master and Slave Operation Supported
- Device can Operate as Transmitter or Receiver
- 7-bit Address Space Allows up to 128 Different Slave Addresses
- Multi-master Arbitration Support
- Up to 400kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- Noise Suppression Circuitry Rejects Spikes on Bus Lines
- Fully Programmable Slave Address with General Call Support
- Address Recognition Causes Wake-up When AVR is in Sleep Mode
- Compatible with Philips' I^{2}C protocol

25.2 Two-Wire Serial Interface Bus Definition

The Two-Wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines: one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 25-1. TWI Bus Interconnection



25.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 25-1. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

The Power Reduction TWI bit in the Power Reduction Register (PRRn.PRTWI) must be written to '0' to enable the two-wire Serial Interface.

TWI0 is in PRR0, and TWI1 is in PRR2.

Related Links

Power Management and Sleep Modes

25.2.2 Electrical Interconnection

As depicted in the TWI Bus Definition, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices tri-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

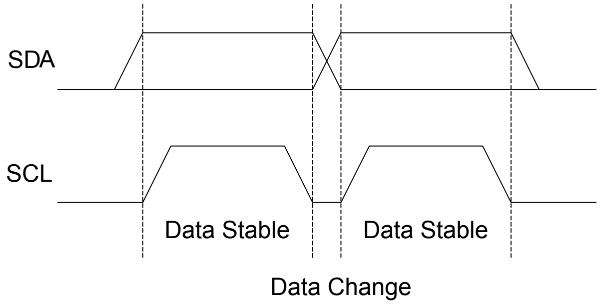
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400pF and the 7-bit slave address space. Two different sets of specifications are presented there, one relevant for bus speeds below 100kHz, and one valid for bus speeds up to 400kHz.

25.3 Data Transfer and Frame Format

25.3.1 Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

Figure 25-2. Data Validity

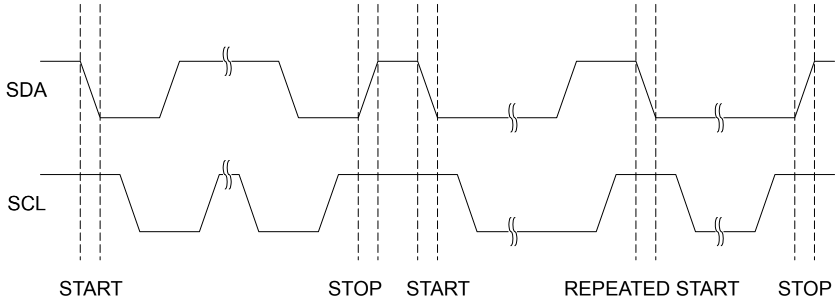


25.3.2 START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition.

Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without relinquishing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signaled by changing the level of the SDA line when the SCL line is high.

Figure 25-3. START, REPEATED START, and STOP conditions



25.3.3 Address Packet Format

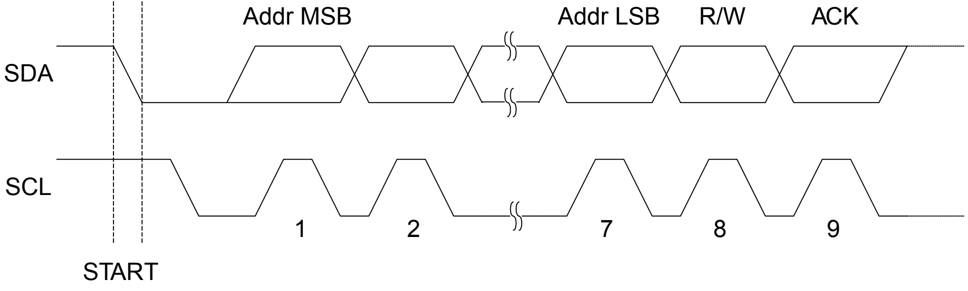
All address packets transmitted on the TWI bus are nine bits long, consisting of seven address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address '0000 000' is reserved for a general call.

When a general call is issued, all slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all slaves set up to acknowledge the general call will pull the SDA line low in the ACK cycle. The following data packets will then be received by all the slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several slaves started transmitting different data.

All addresses of the format '1111 xxx' should be reserved for future purposes.

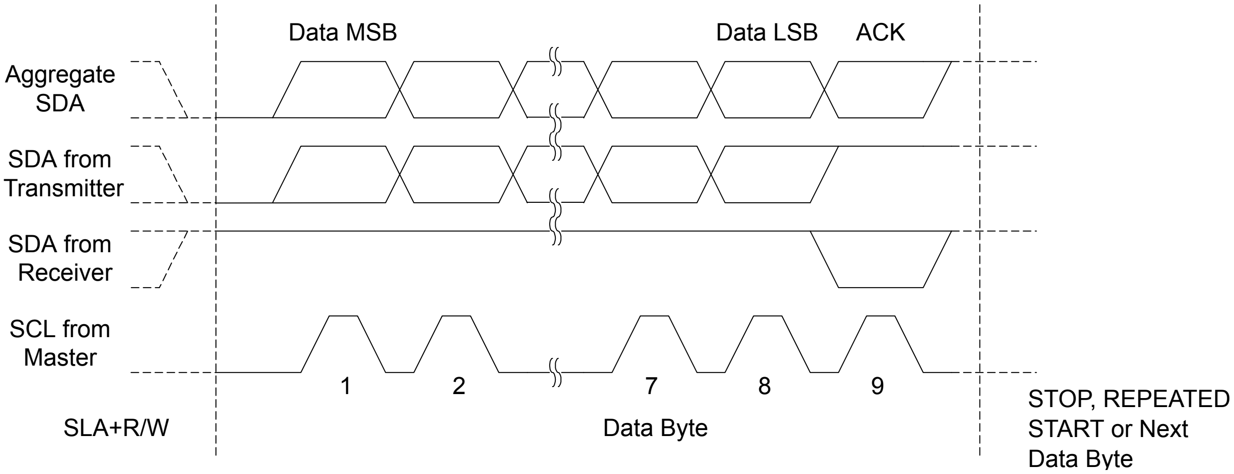
Figure 25-4. Address Packet Format



25.3.4 Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the Receiver pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signalled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 25-5. Data Packet Format

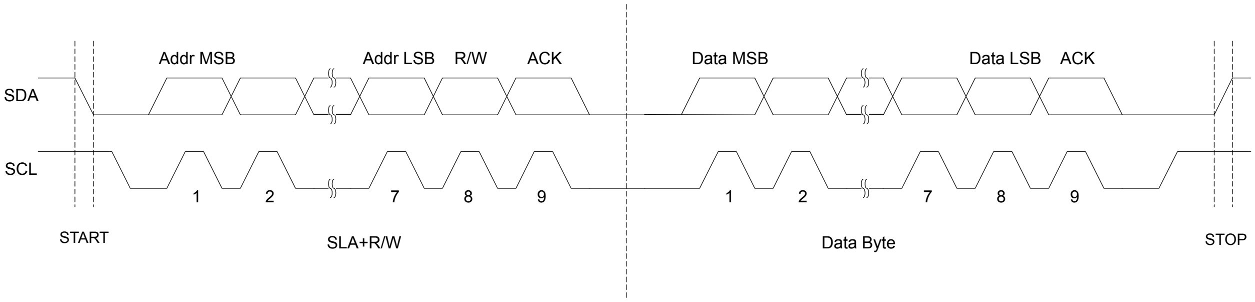


25.3.5 Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the "Wired-ANDing" of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

The following figure depicts a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 25-6. Typical Data Transmission



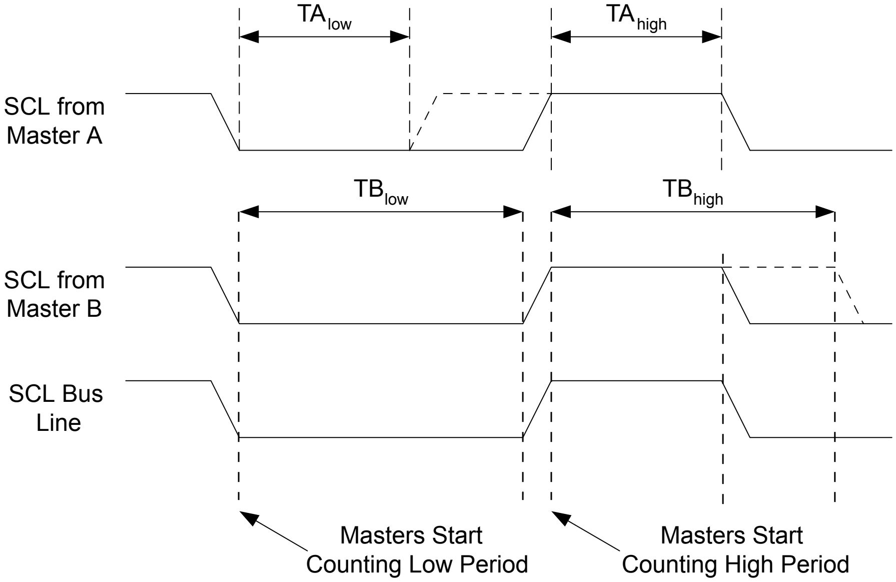
25.4 Multi-master Bus Systems, Arbitration, and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, i.e. the data being transferred on the bus must not be corrupted.
- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

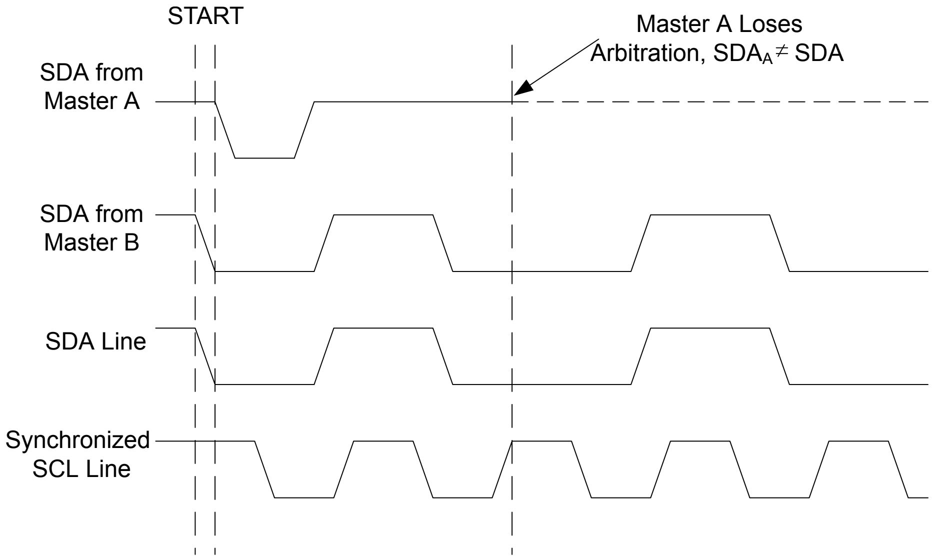
The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 25-7. SCL Synchronization Between Multiple Masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several masters are trying to address the same Slave, arbitration will continue into the data packet.

Figure 25-8. Arbitration Between Two Masters



Note that arbitration is not allowed between:

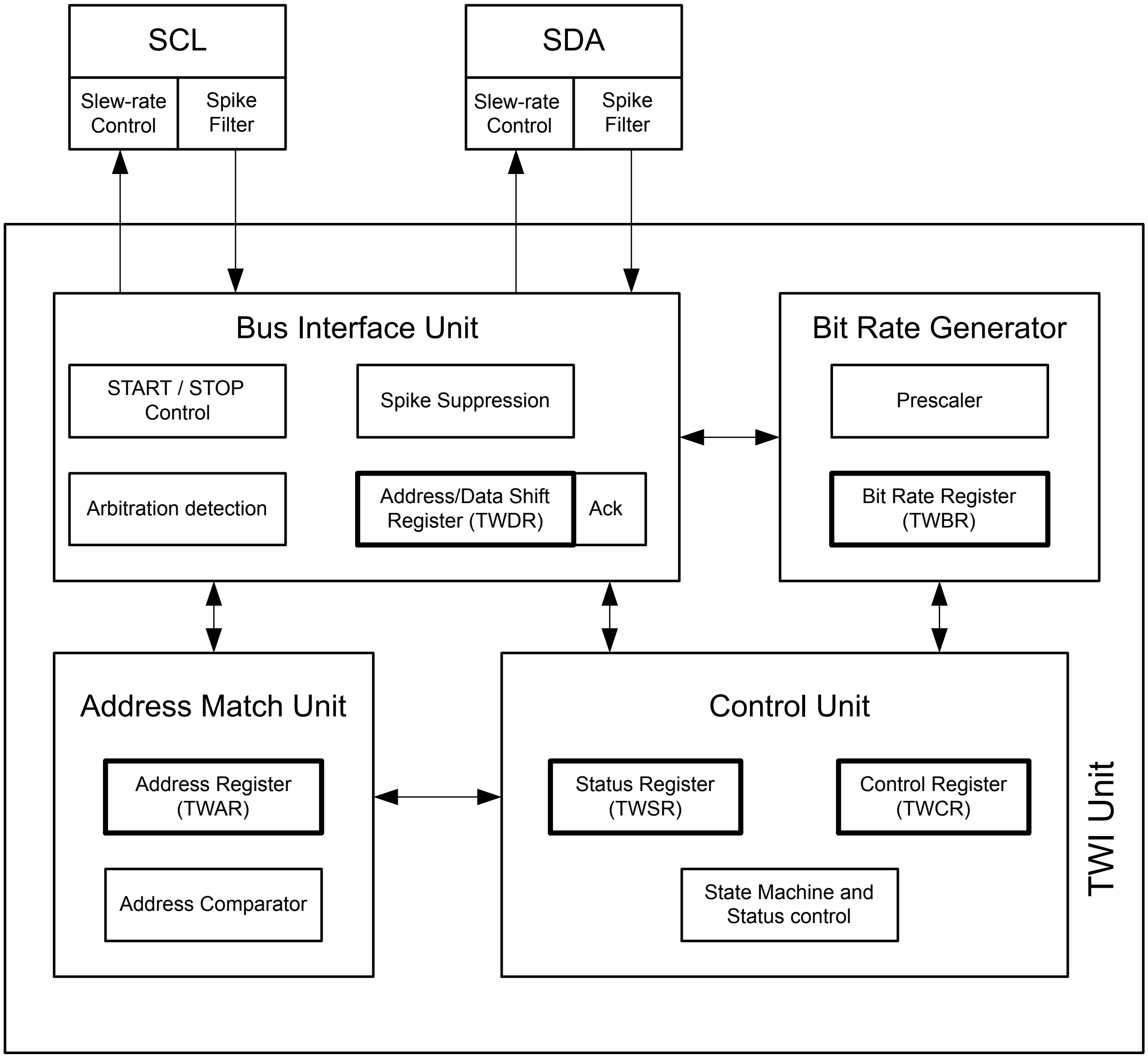
- A REPEATED START condition and a data bit
- A STOP condition and a data bit
- A REPEATED START and a STOP condition

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words; All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

25.5 Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in the following figure. The registers drawn in a thick line are accessible through the AVR data bus.

Figure 25-9. Overview of the TWI Module



25.5.1 SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

25.5.2 Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBRn) and the Prescaler bits in the TWI Status Register (TWSRn). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period.

The SCL frequency is generated according to the following equation:

$$\mathrm{SCL\ frequency} = \frac{\mathrm{CPU\ Clock\ frequency}}{16 + 2(\mathrm{TWBR}) \cdot (\mathrm{PrescalerValue})}$$

- TWBR = Value of the TWI Bit Rate Register TWBRn
- PrescalerValue = Value of the prescaler, see description of the TWI Prescaler bits in the TWSR Status Register description (TWSRn.TWPS[1:0])

Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See the *Two-Wire Serial Interface Characteristics* for a suitable value of the pull-up resistor.

Related Links

Two-wire Serial Interface Characteristics

25.5.3 Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDRn), a START/STOP Controller and Arbitration detection hardware. The TWDRn contains the address or data bytes to be transmitted, or the address or data bytes received. In addition to the 8-bit TWDRn, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCRn). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSRn.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

25.5.4 Address Match Unit

The Address Match unit checks if received address bytes match the seven-bit address in the TWI Address Register (TWARn). If the TWI General Call Recognition Enable bit (TWARn.TWGCE) is written to '1', all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWI Control Register (TWCRn). The Address Match unit is able to compare addresses even when the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master.

25.5.5 Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCRn). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSRn) is updated with a status code identifying the event. The TWSRn only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSRn contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition
- After the TWI has transmitted SLA+R/W
- After the TWI has transmitted an address byte
- After the TWI has lost arbitration
- After the TWI has been addressed by own slave address or general call

- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a Slave
- When a bus error has occurred due to an illegal START or STOP condition

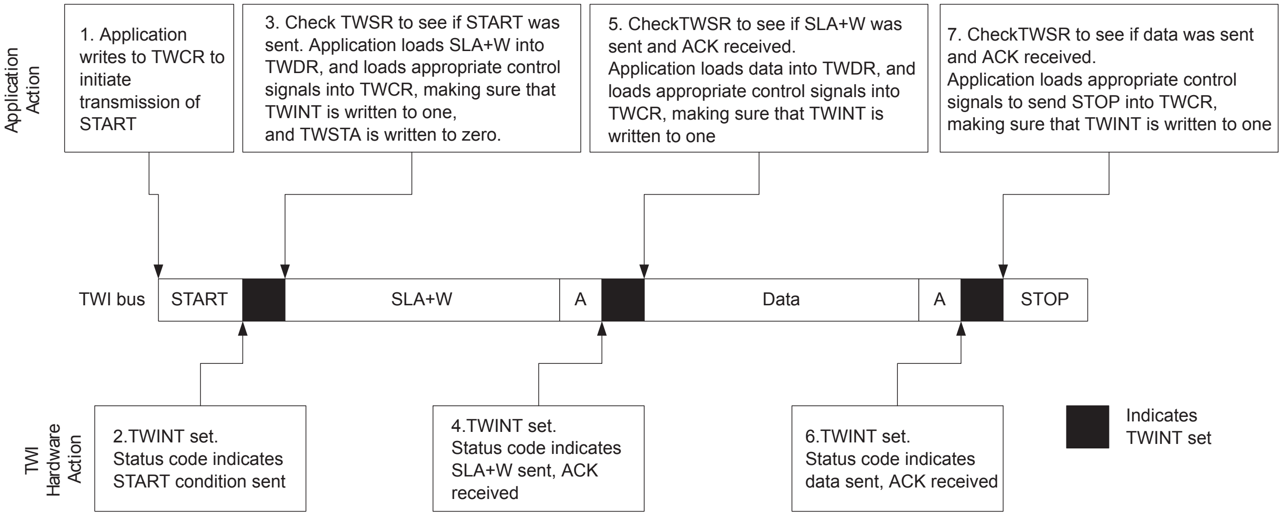
25.6 Using the TWI

The AVR TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCRn together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSRn) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCRn and TWDRn Registers.

The following figure illustrates a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. A more detailed explanation follows later in this section. Simple code examples are presented in the table below.

Figure 25-10. Interfacing the Application to the TWI in a Typical Transmission



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCRn, instructing the TWI n hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI n will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI n will initiate transmission of the START condition.
2. When the START condition has been transmitted, the TWINT Flag in TWCRn is set, and TWSRn is updated with a status code indicating that the START condition has successfully been sent.
3. The application software should now examine the value of TWSRn, to make sure that the START condition was successfully transmitted. If TWSRn indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDRn is used both for

address and data. After TWDRn has been loaded with the desired SLA+W, a specific value must be written to TWCRn, instructing the TWI n hardware to transmit the SLA+W present in TWDRn. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.

4. When the address packet has been transmitted, the TWINT Flag in TWCRn is set, and TWSRn is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
5. The application software should now examine the value of TWSRn, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSRn indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDRn. Subsequently, a specific value must be written to TWCRn, instructing the TWI n hardware to transmit the data packet present in TWDRn. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI n will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.
6. When the data packet has been transmitted, the TWINT Flag in TWCRn is set, and TWSRn is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
7. The application software should now examine the value of TWSRn, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCRn, instructing the TWI n hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI n will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is *not* set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI n Registers with the value relevant for the next TWI n bus cycle. As an example, TWDRn must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI n Register updates and other pending application software tasks have been completed, TWCRn is written. When writing TWCRn, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI n will then commence executing whatever operation was specified by the TWCRn setting.

The following table lists assembly and C implementation examples for TWI0. Note that the code below assumes that several definitions have been made, e.g. by using include-files.

Table 25-2. Assembly and C Code Example

	Assembly Code Example	C Example	Comments

1	`ldi r16, (1<<TWINT)|`
	`(1<<TWSTA)|(1<<TWEN)`
	`out TWCR0, r16`
		`TWCR0 = (1<<TWINT)|`
		`(1<<TWSTA)|(1<<TWEN)`
			Send START condition

2	`wait1:`
	`in r16,TWCR0`
	`sbrs r16,TWINT`
	`rjmp wait1`
		`while (!(TWCR0 &`
		`(1<<TWINT)));`
			Wait for TWINT Flag set. This indicates that the START condition has been transmitted.

3	`in r16,TWSR0`
	`andi r16, 0xF8`
	`cpi r16, START`
	`brne ERROR`
		`if ((TWSR0 & 0xF8) !=`
		`START)`
		`  ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR.

	`ldi r16, SLA_W`
	`out TWDR0, r16`
	`ldi r16, (1<<TWINT) |`
	`(1<<TWEN)`
	`out TWCR0, r16`
		`TWDR0 = SLA_W;`
		`TWCR0 = (1<<TWINT) |`
		`(1<<TWEN);`
			Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address.

4	`wait2:`
	`in r16,TWCR0`
	`sbrs r16,TWINT`
	`rjmp wait2`
		`while (!(TWCR0 &`
		`(1<<TWINT)));`
			Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.

5	`in r16,TWSR0`
	`andi r16, 0xF8`
	`cpi r16, MT_SLA_ACK`
	`brne ERROR`
		`if ((TWSR0 & 0xF8) !=`
		`MT_SLA_ACK) ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR.

	`ldi r16, DATA`
	`out TWDR0, r16`
	`ldi r16, (1<<TWINT) |`
	`(1<<TWEN)`
	`out TWCR, r16`
		`TWDR0 = DATA;`
		`TWCR0 = (1<<TWINT) |`
		`(1<<TWEN);`
			Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data.

6	`wait3:`
	`in r16,TWCR0`
	`sbrs r16,TWINT`
	`rjmp wait3`
		`while (!(TWCR0 &`
		`(1<<TWINT)));`
			Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.

7	`in r16,TWSR0`
	`andi r16, 0xF8`
	`cpi r16, MT_DATA_ACK`
	`brne ERROR`
		`if ((TWSR0 & 0xF8) !=`
		`MT_DATA_ACK) ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR.

	`ldi r16, (1<<TWINT)|`
	`(1<<TWEN)| (1<<TWSTO)`
	`out TWCR0, r16`
		`TWCR0 = (1<<TWINT)|`
		`(1<<TWEN)|(1<<TWSTO);`
			Transmit STOP condition.

25.7 Transmission Modes

The TWI can operate in one of four major modes:

- Master Transmitter (MT)
- Master Receiver (MR)
- Slave Transmitter (ST)
- Slave Receiver (SR)

Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures use the following abbreviations:

S	START condition
Rs	REPEATED 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

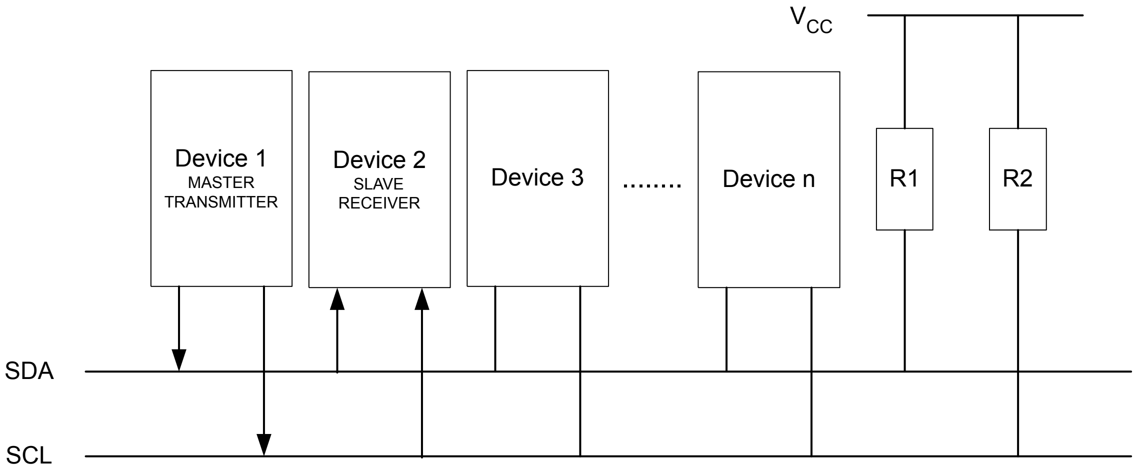
Circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSRn, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

When the TWINT Flag is set, the status code in TWSRn 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 below in the Status Code table for each mode. Note that the prescaler bits are masked to zero in these tables.

25.7.1 Master Transmitter Mode

In the Master Transmitter (MT) mode, a number of data bytes are transmitted to a Slave Receiver, see figure below. In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether MT or Master Receiver (MR) mode is to be entered: If SLA +W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-11. Data Transfer in Master Transmitter Mode



A START condition is sent by writing a value to the TWI Control Register n (TWCRn) of the type TWCRn=1x10x10x:

- The TWI Enable bit (TWCRn.TWEN) must be written to '1' to enable the 2-wire Serial Interface
- The TWI Start Condition bit (TWCRn.TWSTA) must be written to '1' to transmit a START condition
- The TWI Interrupt Flag (TWCRn.TWINT) must be written to '1' to clear the flag.

The TWI n will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSRn will be 0x08 (see Status Code table below). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to the TWI Data Register (TWDRn). Thereafter, the TWCRn.TWINT Flag should be cleared (by writing a '1' to it) to continue the transfer. This is accomplished by writing a value to TWRC of the type TWCR=1x00x10x.

When SLA+W have been transmitted and an acknowledgment 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 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in the Status Code table below.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCRn Register. After updating TWDRn, the TWINT bit should be cleared (by writing '1' to it) to continue the transfer. This is accomplished by writing again a value to TWCRn of the type TWCRn=1x00x10x.

This scheme is repeated until the last byte has been sent and the transfer is ended, either by generating a STOP condition or a by a repeated START condition. A repeated START condition is accomplished by writing a regular START value TWCRn=1x10x10x. A STOP condition is generated by writing a value of the type TWCRn=1x01x10x.

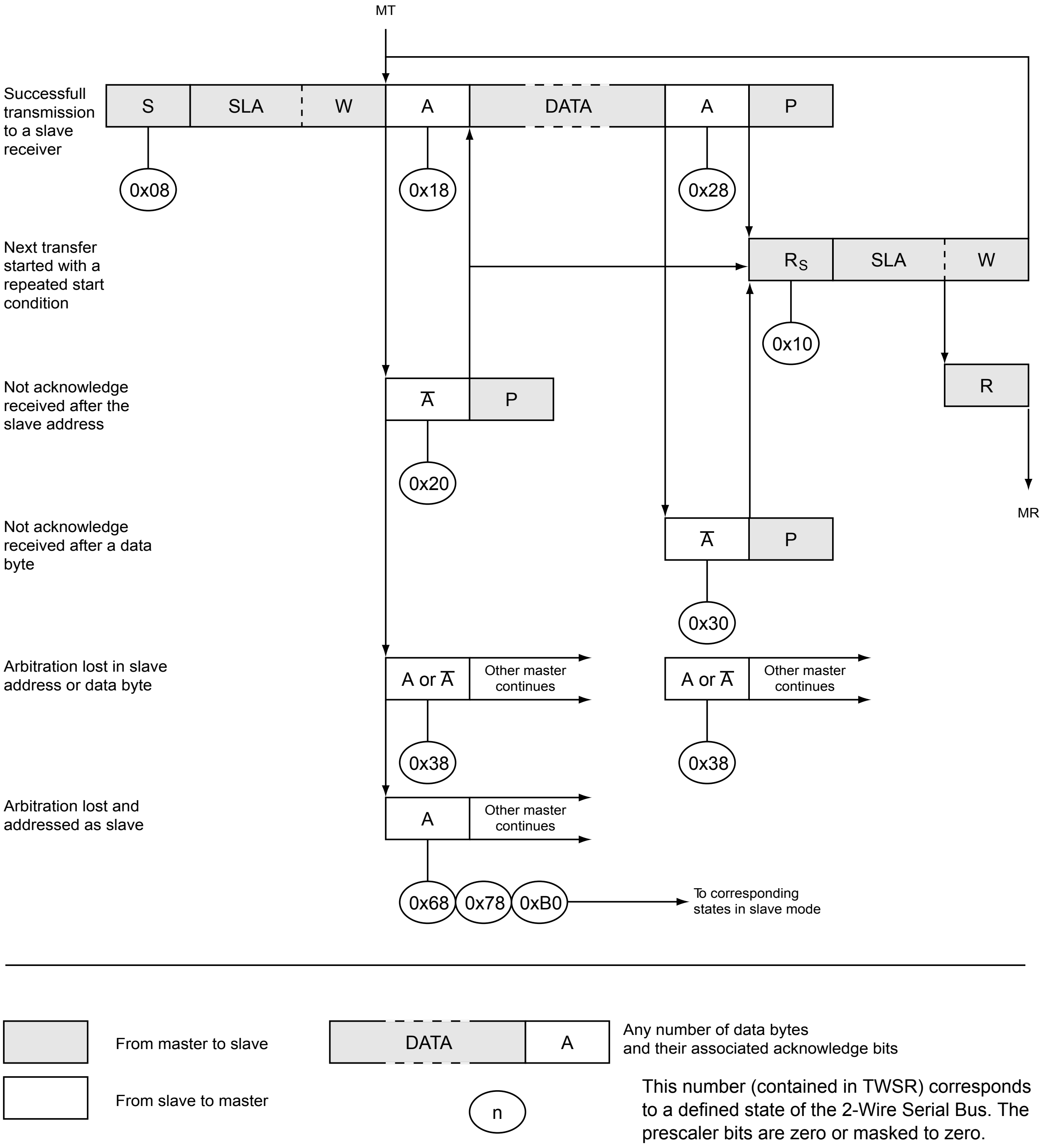
After a repeated START condition (status code 0x10), the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

Table 25-3. Status Codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCRn				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
0x18	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
0x20	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
0x28	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

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCRn				
			STA	STO	TWINT	TWEA	
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x30	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
0x38	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

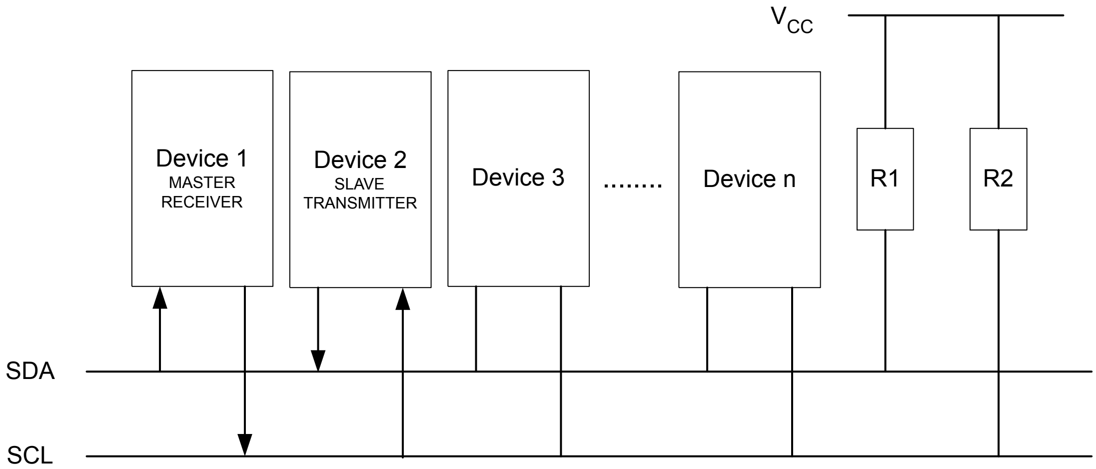
Figure 25-12. Formats and States in the Master Transmitter Mode



25.7.2 Master Receiver Mode

In the Master Receiver (MR) mode, a number of data bytes are received from a Slave Transmitter (see next figure). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter (MT) or MR mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-13. Data Transfer in Master Receiver Mode



A START condition is sent by writing to the TWI Control register (TWCRn) a value of the type TWCRn=1x10x10x:

- TWCRn.TWEN must be written to '1' to enable the 2-wire Serial Interface
- TWCRn.TWSTA must be written to '1' to transmit a START condition
- TWCRn.TWINT must be cleared by writing a '1' to it.

The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSRn will be 0x08 (see Status Code table below). In order to enter MR mode, SLA +R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter, the TWINT flag should be cleared (by writing '1' to it) to continue the transfer. This is accomplished by writing the a value to TWCRn of the type TWCRn=1x00x10x.

When SLA+R have been transmitted and an acknowledgment bit has been received, TWINT is set again and a number of status codes in TWSRn are possible. Possible status codes in Master mode are 0x38, 0x40, or 0x48. The appropriate action to be taken for each of these status codes is detailed in the table below. 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. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A repeated START condition is sent by writing to the TWI Control register (TWCRn) a value of the type TWCRn=1x10x10x again. A STOP condition is generated by writing TWCRn=1x01x10x:

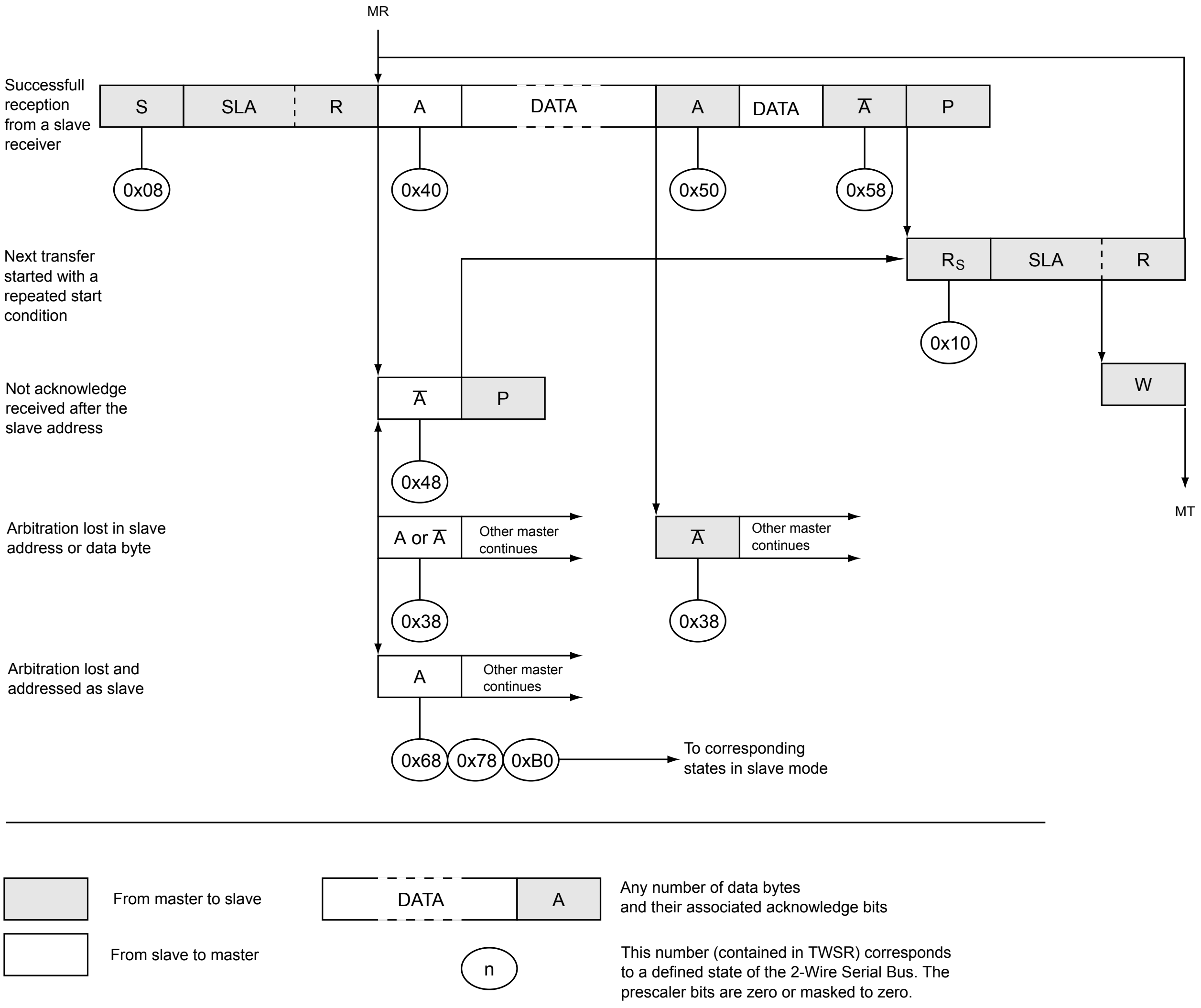
After a repeated START condition (status code 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 25-4. Status codes for Master Receiver Mode

Status Code (TWSRn) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWD	To TWCRn				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted

Status Code (TWSRn) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWD	To TWCRn				
			STA	STO	TWINT	TWEA	
							ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode will be entered
			1	0	1	X	A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action	0	0	1	0	Data byte will be received and NOT ACK will be returned
			0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received		1	0	1	X	Repeated START will be transmitted
			0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
			1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte	0	0	1	0	Data byte will be received and NOT ACK will be returned
			0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte	1	0	1	X	Repeated START will be transmitted
			0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
			1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

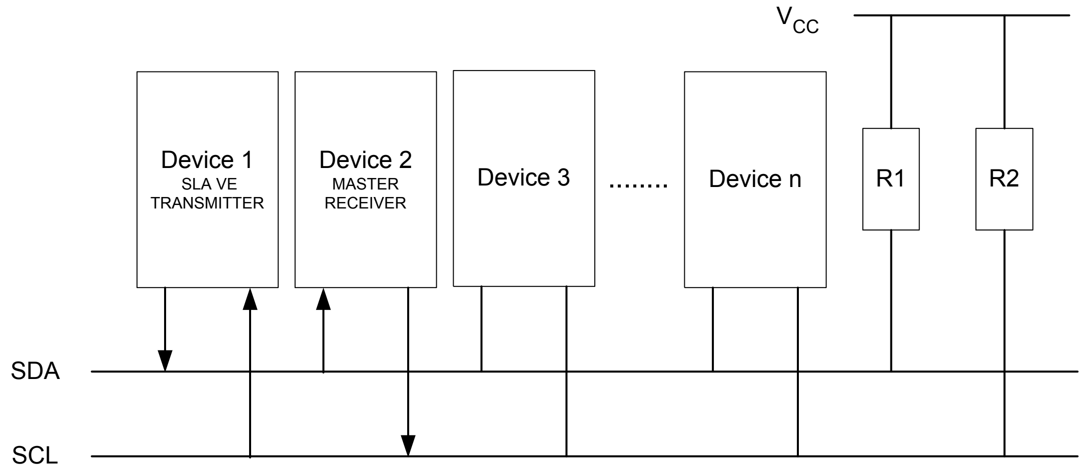
Figure 25-14. Formats and States in the Master Receiver Mode



25.7.3 Slave Transmitter Mode

In the Slave Transmitter (ST) mode, a number of data bytes are transmitted to a Master Receiver, as in the figure below. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-15. Data Transfer in Slave Transmitter Mode



To initiate the SR mode, the TWI (Slave) Address Register (TWARn) and the TWI Control Register (TWCRn) must be initialized as follows:

The upper seven bits of TWARn are the address to which the 2-wire Serial Interface will respond when addressed by a Master (TWARn.TWA[6:0]). If the LSB of TWARn is written to TWARn.TWGCI=1, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCRn must hold a value of the type TWCRn=0100010x - TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgment of the device’s own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWARn and TWCRn have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is “1” (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSRb. The status code is used to determine the appropriate sofTWARne action. The appropriate action to be taken for each status code is detailed in the table below. The ST mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWCRn.TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI 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. State 0xC8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWCRn.TWEA is zero, the TWI 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 TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing '1' to it). Further data transmission will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

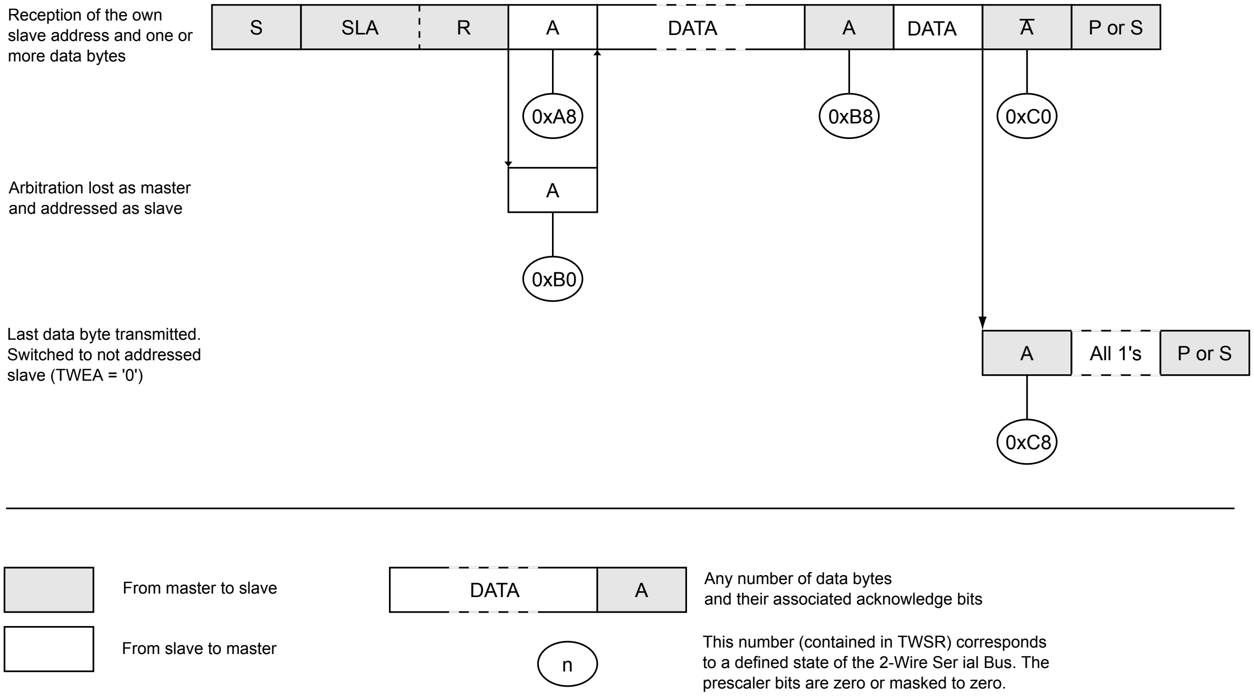
Note: The 2-wire Serial Interface Data Register (TWDRn) does not reflect the last byte present on the bus when waking up from these Sleep modes.

Table 25-5. Status Codes for Slave Transmitter Mode

Status Code (TWSRb) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application SofTWARne Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
			X	0	1	1	Data byte will be transmitted and ACK should be received
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
			X	0	1	1	Data byte will be transmitted and ACK should be received
0xB8	Data byte in TWDRn has been transmitted; ACK has been received	Load data byte	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
			X	0	1	1	Data byte will be transmitted and ACK should be received
0xC0	Data byte in TWDRn has been transmitted; NOT ACK has been received	No TWDRn action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			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
			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
0xC8	Last data byte in TWDRn has been transmitted (TWEA = “0”);	No TWDRn action	0	0	1	0	Switched to the not addressed Slave mode;

Status Code (TWSRb) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application SofTWARne Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
	ACK has been received						no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			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
			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

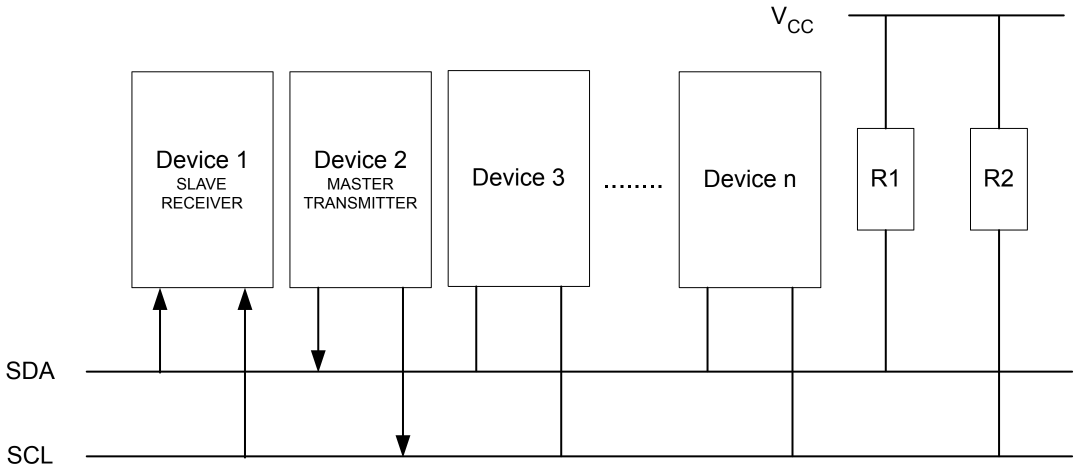
Figure 25-16. Formats and States in the Slave Transmitter Mode



25.7.4 Slave Receiver Mode

In the Slave Receiver (SR) mode, a number of data bytes are received from a Master Transmitter (see figure below). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-17. Data transfer in Slave Receiver mode



To initiate the SR mode, the TWI (Slave) Address Register n (TWARn) and the TWI Control Register n (TWCRn) must be initialized as follows:

The upper seven bits of TWARn are the address to which the 2-wire Serial Interface will respond when addressed by a Master (TWARn.TWA[6:0]). If the LSB of TWARn is written to TWARn.TWGCI=1, the TWI n will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCRn must hold a value of the type TWCRn=0100010x - TWCRn.TWEN must be written to '1' to enable the TWI. TWCRn.TWEA bit must be written to '1' to enable the acknowledgment of the device’s own slave address or the general call address. TWCRn.TWSTA and TWSTO must be written to zero.

When TWARn and TWCRn have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address, if enabled) followed by the data direction bit. If the direction bit is '0' (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action, as detailed in the table below. The SR mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

If the TWCRn.TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ('1') to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge 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 TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing '1' to it). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note: The 2-wire Serial Interface Data Register (TWDRn) does not reflect the last byte present on the bus when waking up from these Sleep modes.

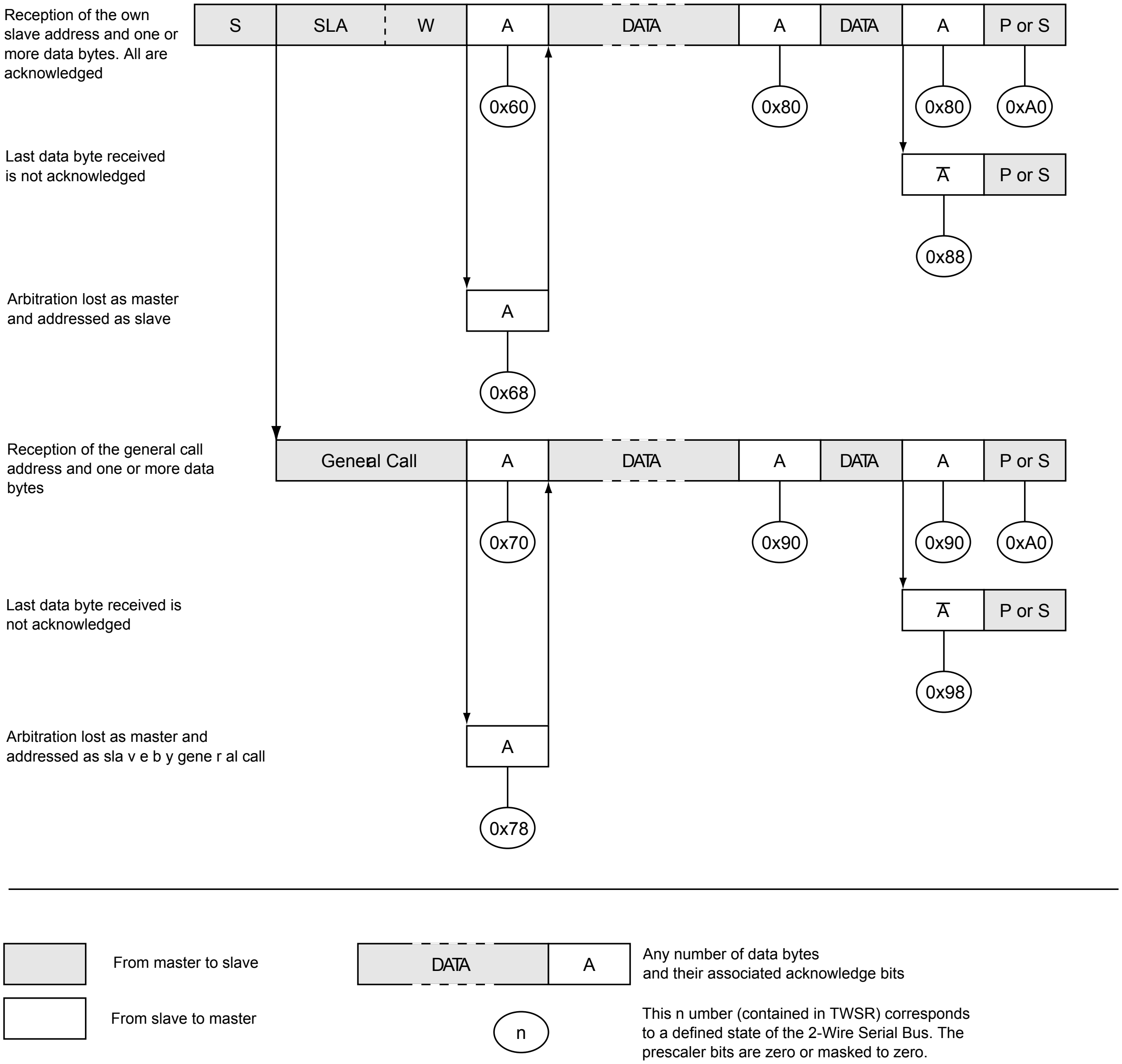
Table 25-6. Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application SofTWARne Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
0x60	Own SLA+W has been received; ACK has been returned	No TWDRn action	X	0	1	0	Data byte will be received and NOT ACK will be returned
			X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDRn action	X	0	1	0	Data byte will be received and NOT ACK will be returned
			X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address has been received; ACK has been returned	No TWDRn action	X	0	1	0	Data byte will be received and NOT ACK will be returned
			X	0	1	1	Data byte will be received and ACK will be returned
0x78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDRn action	X	0	1	0	Data byte will be received and NOT ACK will be returned
			X	0	1	1	Data byte will be received and ACK will be returned

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application SofTWARne Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte	X	0	1	0	Data byte will be received and NOT ACK will be returned
			X	0	1	1	Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			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
			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
0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte	X	0	1	0	Data byte will be received and NOT ACK will be returned
			X	0	1	1	Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized;

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application SofTWARne Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
							GCA will be recognized if TWGCE = “1”
			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
			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
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			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
			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 25-18. Formats and States in the Slave Receiver Mode



25.7.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see the table in this section.

Status 0xF8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status 0x00 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 TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCRn are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 25-7. Miscellaneous States

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
0xF8	No relevant state information available; TWINT = “0”	No TWDRn action	No TWCRn action				Wait or proceed current transfer
0x00	Bus error due to an illegal START or STOP condition	No TWDRn 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.

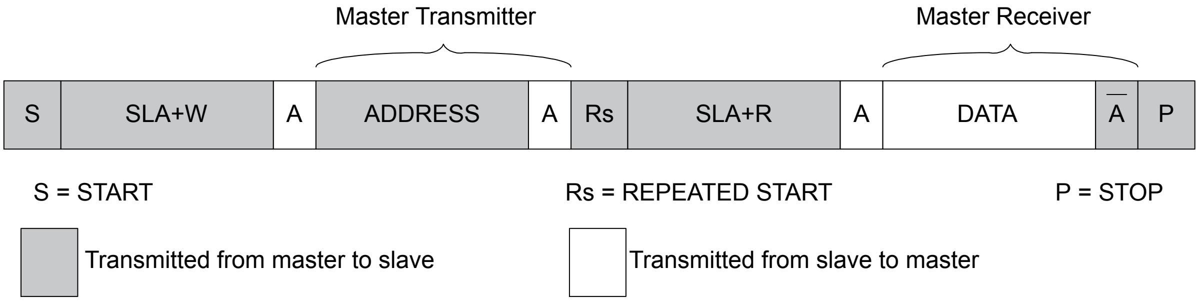
25.7.6 Combining Several TWI Modes

In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

1. The transfer must be initiated.
2. The EEPROM must be instructed what location should be read.
3. The reading must be performed.
4. The transfer must be finished.

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomical operation. If this principle is violated in a multi master system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The flow in this transfer is depicted in the following figure:

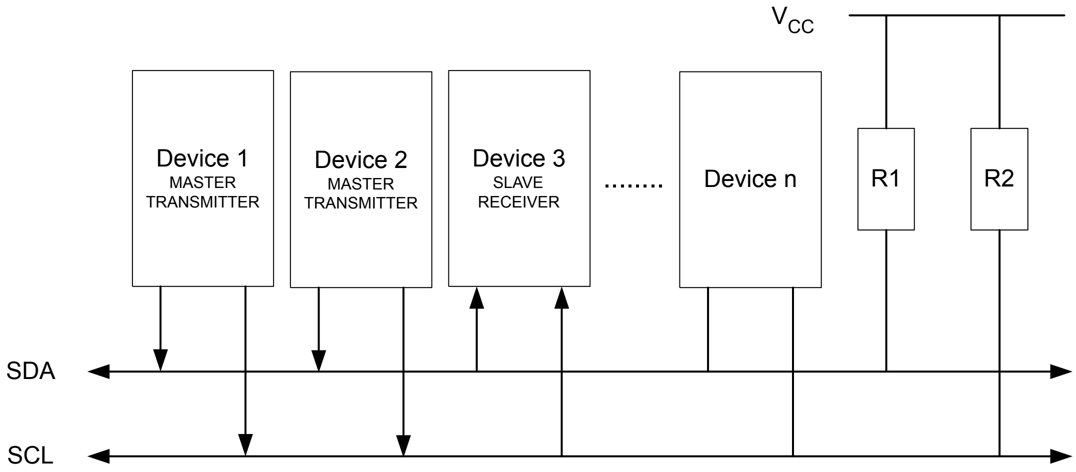
Figure 25-19. Combining Several TWI Modes to Access a Serial EEPROM



25.8 Multi-master Systems and Arbitration

If multiple masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two masters are trying to transmit data to a Slave Receiver.

Figure 25-20. An Arbitration Example

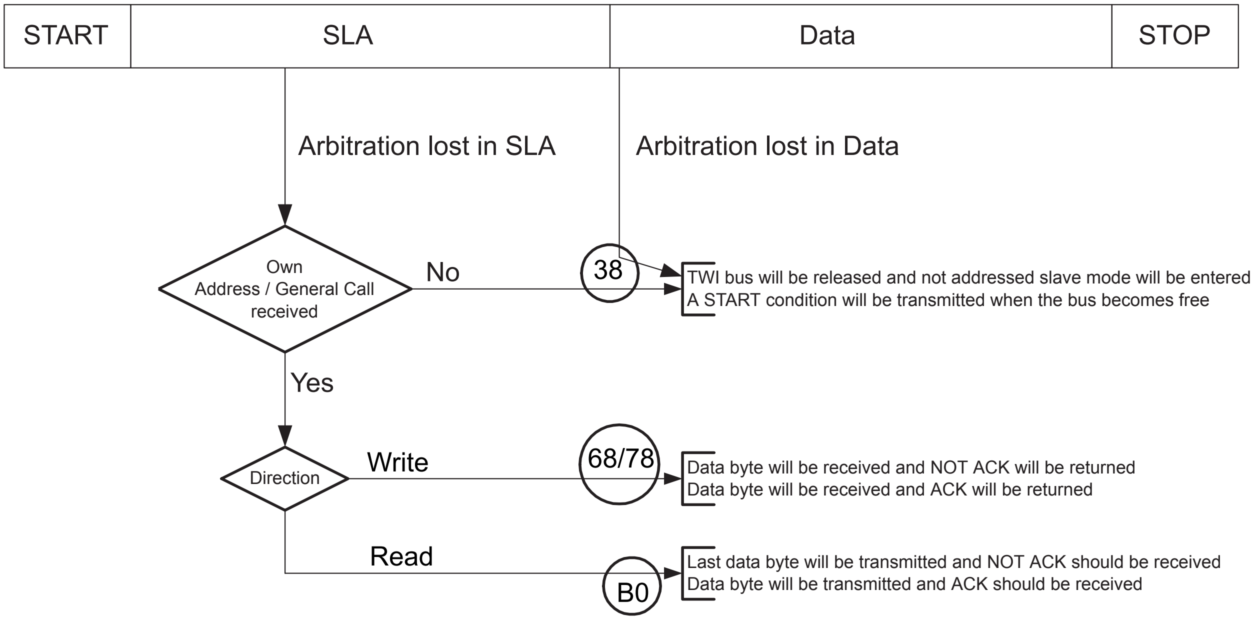


Several different scenarios may arise during arbitration, as described below:

- Two or more masters are performing identical communication with the same Slave. In this case, neither the Slave nor any of the masters will know about the bus contention.
- Two or more masters are accessing the same Slave with different data or direction bit. In this case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The masters trying to output a '1' on SDA while another Master outputs a zero will lose the arbitration. Losing masters will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.
- Two or more masters are accessing different slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a '1' on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

This is summarized in the next figure. Possible status values are given in circles.

Figure 25-21. Possible Status Codes Caused by Arbitration



25.9 Register Description

25.9.1 TWI n Bit Rate Register

Name: TWBR
Offset: 0xB8 + n*0x20 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TWBR [7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TWBR [7:0]: TWI 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.

25.9.2 TWI Status Register n

Name: TWSR
Offset: 0xB9 + n*0x20 [n=0..1]
Reset: 0xF8
Property: -

Bit	7	6	5	4	3	2	1	0
	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
Access	R	R	R	R	R		R/W	R/W
Reset	1	1	1	1	1		0	0

Bits 3, 4, 5, 6, 7 – TWS3, TWS4, TWS5, TWS6, TWS7: TWI Status Bit
The TWS[7:3] reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes are described in Transmission Modes. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

Bits 1:0 – TWPS[1:0]: TWI Prescaler
These bits can be read and written, and control the bit rate prescaler.

Table 25-8. TWI Bit Rate Prescaler

TWPS[1:0]	Prescaler Value
00	1
01	4
10	16
11	64

To calculate bit rates, refer to Bit Rate Generator Unit. The value of TWPS[1:0] is used in the equation.

25.9.3 TWI (Slave) Address Register n

The TWARn should be loaded with the 7-bit Slave address (in the seven most significant bits of TWARn) to which the TWI n will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. In multi master systems, TWARn must be set in masters which can be addressed as Slaves by other Masters.

The LSB of TWARn is used to enable recognition of the general call address (0x00). 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.

Name: TWAR
Offset: 0xBA + n*0x20 [n=0..1]
Reset: 0x02
Property: -

Bit	7	6	5	4	3	2	1	0
	TWA[6:0]							TWGCE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	0

Bits 7:1 – TWA[6:0]: TWI (Slave) Address
These seven bits constitute the slave address of the TWI n unit.

Bit 0 – TWGCE: TWI General Call Recognition Enable Bit
If set, this bit enables the recognition of a General Call given over the 2-wire Serial Bus n.

25.9.4 TWI Data Register n

In Transmit mode, TWDRn contains the next byte to be transmitted. In Receive mode, the TWDRn contains the last byte received. It is writable while the TWI n is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag in the TWI Control Register n (TWCRn.TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDRn remains stable as long as TWCRn.TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDRn always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI n interrupt. In this case, the contents of TWDRn is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

Name: TWDR
Offset: 0xBB + n*0x20 [n=0..1]
Reset: 0x01
Property: -

Bit	7	6	5	4	3	2	1	0
	TWD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – TWD[7:0]: TWI Data
These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-wire Serial Bus.

25.9.5 TWI Control Register n

The TWCRn is used to control the operation of the TWI n. It is used to enable the TWI n, 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 TWDRn. It also indicates a write collision if data is attempted written to TWDRn while the register is inaccessible.

Name: TWCR
Offset: 0xBC + n*0x20 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
Access	R/W	R/W	R/W	R/W	R	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit 7 – TWINT: TWI Interrupt Flag
This bit is set by hardware when the TWI n has finished its current job and expects application software response. If the I-bit in the Status Register (SREG.I) and the TWI Interrupt Enable bit in the TWI Control

Register n (TWCRn.TWIE) are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL 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 TWI n, so all accesses to the TWI Address Register (TWARn), TWI Status Register (TWSRn), and TWI Data Register (TWDRn) must be complete before clearing this flag.

Bit 6 – TWEA: TWI Enable Acknowledge
This bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI n 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 TWARn is set.
3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the 2-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

Bit 5 – TWSTA: TWI START Condition
The application writes the TWSTA bit to one when it desires TWI n to become a Master on the 2-wire Serial Bus. The TWI n 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 TWI n waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

Bit 4 – TWSTO: TWI STOP Condition
Writing the TWSTO bit to one in Master mode will generate a STOP condition on the 2-wire Serial Bus TWI n. 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. This will not generate a STOP condition, but the TWI n returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

Bit 3 – TWWC: TWI Write Collision Flag
The TWWC bit is set when attempting to write to the TWI n Data Register (TWDRn) when TWCRn.TWINT is low. This flag is cleared by writing the TWDRn register when TWINT is high.

Bit 2 – TWEN: TWI Enable
The TWEN bit enables TWI n operation and activates the TWI n interface. When TWEN is written to one, the TWI n takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI n is switched off and all transmissions of TWI n are terminated, regardless of any ongoing operation.

Bit 0 – TWIE: TWI Interrupt Enable
When this bit is written to one, and the I-bit in the Status Register (SREG.I) is set, the TWI n interrupt request will be activated for as long as the TWCRn.TWINT Flag is high.

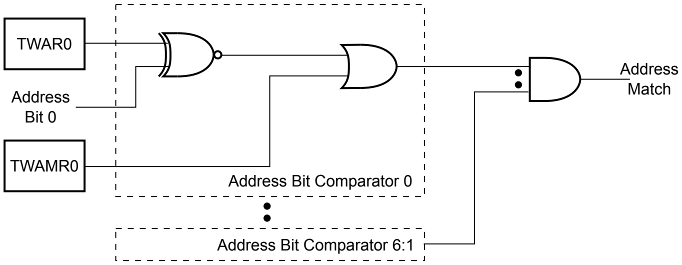
25.9.6 TWI (Slave) Address Mask Register n

Name: TWAMR
Offset: 0xBD + n*0x20 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TWAM[6:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bits 7:1 – TWAM[6:0]: TWI (Slave) Address
The TWAMRn can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMRn can mask (disable) the corresponding address bits in the TWI Address Register n (TWARn). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWARn.

Figure 25-22. Address Match Logic, Example For TWI0



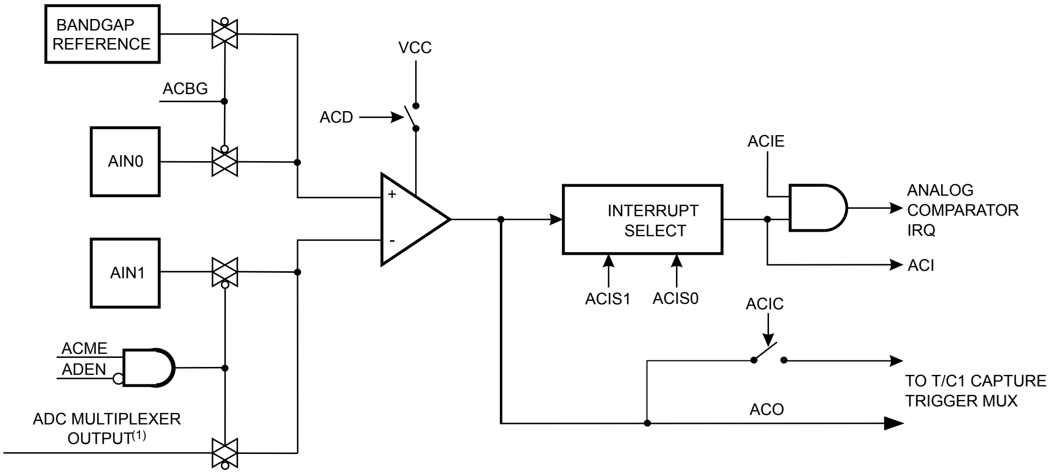
26. AC - Analog Comparator

26.1 Overview

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown below.

The Power Reduction ADC bit in the Power Reduction Register (PRR0.PRADC) must be written to '0' in order to be able to use the ADC input MUX.

Figure 26-1. Analog Comparator Block Diagram



Note: Refer to the *Pin Configuration* and the I/O Ports description for Analog Comparator pin placement

Related Links

I/O-Ports

PRR0

Pin Configurations

Power Management and Sleep Modes

Minimizing Power Consumption

26.2 Analog Comparator Multiplexed Input

It is possible to select any of the ADC[7:0] pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit in the ADC Control and Status Register B (ADCSRB.ACME) is '1' and the ADC is switched off (ADCSRA.ADEN=0), the three least significant Analog Channel Selection bits in the ADC Multiplexer Selection register (ADMUX.MUX[2:0]) select the input pin to replace the negative input to the Analog Comparator, as shown in the table below. When ADCSRB.ACME=0 or ADCSRA.ADEN=1, AIN1 is applied to the negative input of the Analog Comparator.

Table 26-1. Analog Comparator Multiplexed Input

ACME	ADEN	MUX[2:0]	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

26.3 Register Description

26.3.1 Analog Comparator Control and Status Register B

The Store Program Memory Control and Status Register contains the control bits needed to control the Boot Loader operations.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: ACSRB
Offset: 0x4F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2F

Bit	7	6	5	4	3	2	1	0
								ACOE
Access								R/W
Reset								0

Bit 0 – ACOE: Analog Comparator Output Enable
When this bit is set, the analog comparator output is connected to the ACO pin.

26.3.2 Analog Comparator Control and Status Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: ACSR
Offset: 0x50
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x30

Bit	7	6	5	4	3	2	1	0
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS [1:0]	
Access	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – ACD: Analog Comparator Disable
When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 – ACBG: Analog Comparator Bandgap Select
When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. When the bandgap reference is used as input to the Analog Comparator, it will take a certain time for the voltage to stabilize. If not stabilized, the first conversion may give a wrong value.

Bit 5 – ACO: Analog Comparator Output
The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

Bit 4 – ACI: Analog Comparator Interrupt Flag
This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

Bit 3 – ACIE: Analog Comparator Interrupt Enable
When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

Bit 2 – ACIC: Analog Comparator Input Capture Enable
When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

Bits 1:0 – ACIS [1:0]: Analog Comparator Interrupt Mode Select
These bits determine which comparator events that trigger the Analog Comparator interrupt.

Table 26-2. ACIS[1:0] 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.

26.3.3 Digital Input Disable Register 1

Name: DIDR1
Offset: 0x7F
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							AIN1D	AIN0D
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1 – AIN0D, AIN1D: AIN Digital Input Disable
When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

27. ADC - Analog to Digital Converter

27.1 Features

- 10-bit Resolution
- 0.5 LSB Integral Non-Linearity
- ±2 LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 15kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- Differential mode with selectable gain at 1x, 10x or 200x(1)
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- 2.7V - V_{CC} Differential ADC Voltage Range
- Selectable 1.1V or 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

Note:

1. The differential input channels are not tested for devices in PDIP Package. This feature is only guaranteed to work for devices in TQFP and VQFN/QFN/MLF Packages.

27.2 Overview

The device features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage. This provides amplification steps of 0 dB (1x), 20 dB (10x), or 46 dB (200x) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1x or 10x gain is used, 8-bit resolution can be expected. If 200x gain is used, 6- bit resolution can be expected. Note that internal references of 1.1V should not be used on 10x and 200x gain.

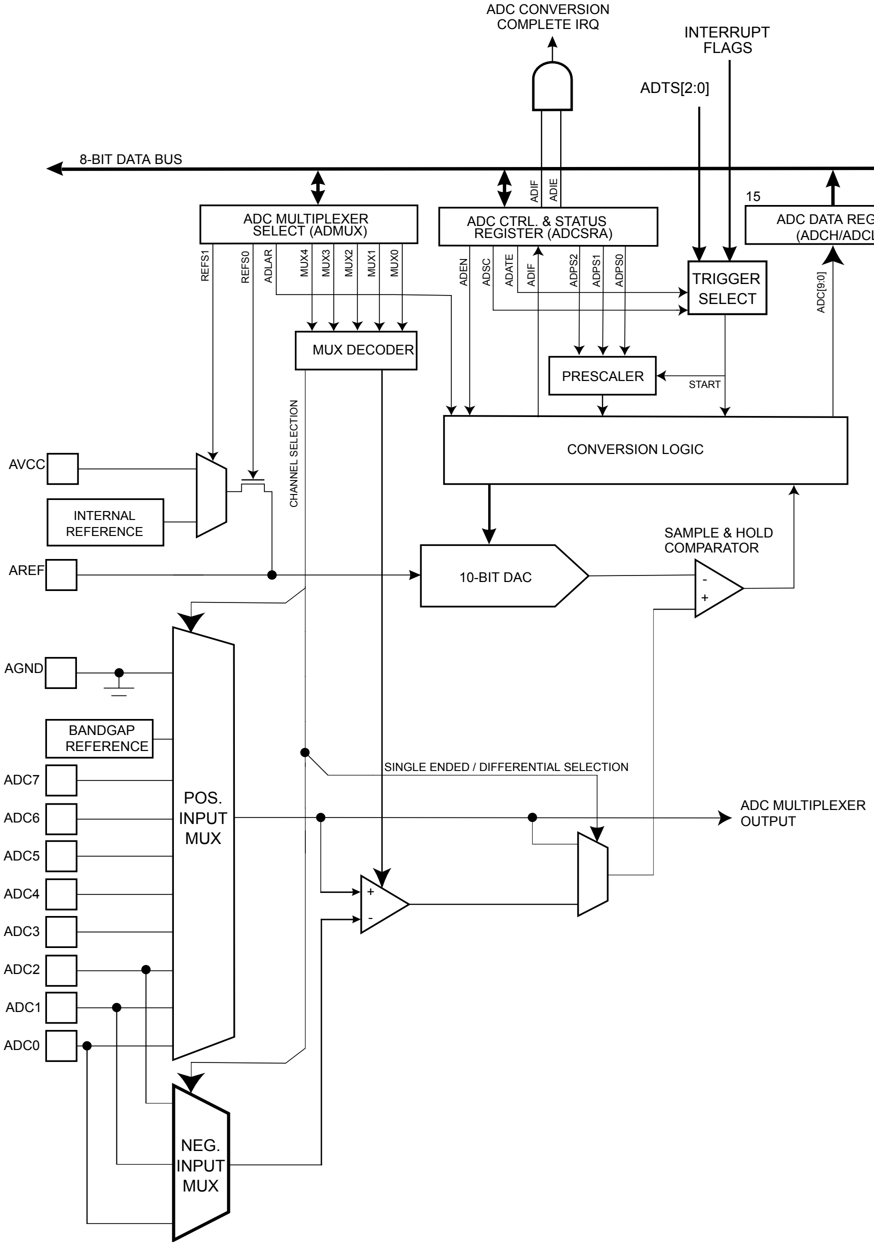
The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown below.

The ADC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ±0.3V from V_{CC}. See section ADC Noise Canceler on how to connect this pin.

The Power Reduction ADC bit in the Power Reduction Register (PRR0.PRADC) must be written to '0' in order to be enable the ADC.

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AV_{CC} or an internal 1.1V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference must be decoupled by an external capacitor at the AREF pin to improve noise immunity.

Figure 27-1. Analog to Digital Converter Block Schematic Operation



The analog input channel is selected by writing to the MUX bits in the ADC Multiplexer Selection register ADMUX.MUX[4:0]. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. The ADC is enabled by writing a '1' to the ADC Enable bit in the ADC Control and Status Register A (ADCSRA.ADEN). Voltage reference and input channel selections will not take 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.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by the selected gain factor. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

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

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 second conversion completes before ADCH is read, neither register is updated and the result from the second 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.

Related Links

Power Management and Sleep Modes
Power Reduction Register

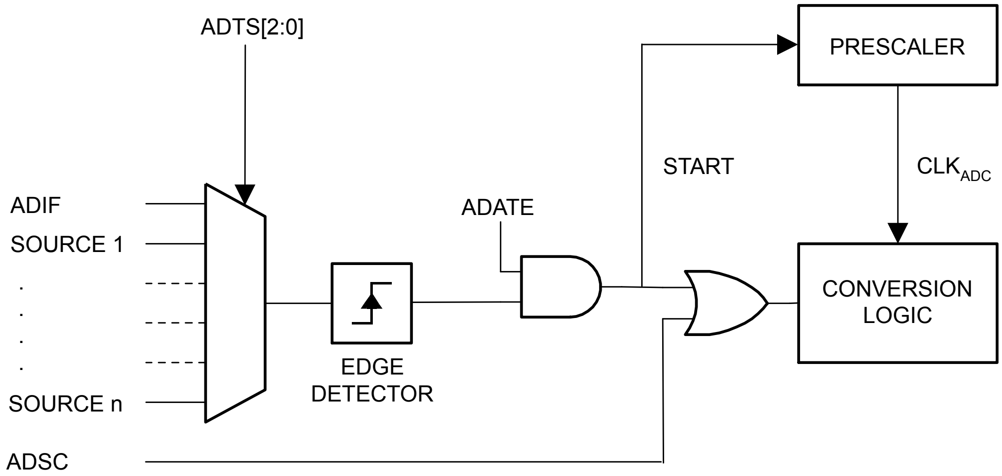
27.3 Starting a Conversion

A single conversion is started by writing a '0' to the Power Reduction ADC bit in the Power Reduction Register (PRR0.PRADC), and writing a '1' to the ADC Start Conversion bit in the ADC Control and Status Register A (ADCSRA.ADSC). ADCS will stay high as long as the conversion is in progress, and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit (ADCSRA.ADATE). The trigger source is selected by setting the ADC Trigger Select bits in the ADC Control and Status Register B (ADCSRB.ADTS). See the description of the ADCSRB.ADTS for a list of available trigger sources.

When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the Global Interrupt Enable bit in the AVR Status Register (SREG.I) is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 27-2. ADC Auto Trigger Logic

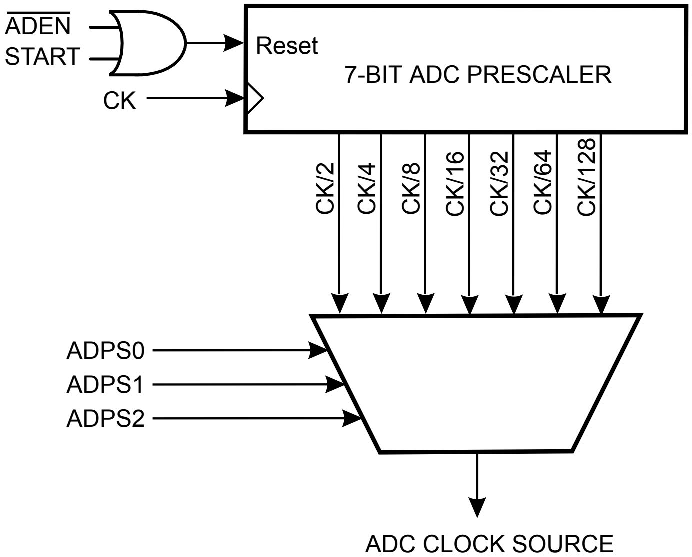


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a '1' to ADCSRA.ADSC. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag (ADIF) is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADCSRA.ADSC to '1'. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as '1' during a conversion, independently of how the conversion was started.

27.4 Prescaling and Conversion Timing

Figure 27-3. ADC Prescaler



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

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is selected by the ADC Prescaler Select bits in the ADC Control and Status Register A (ADCSRA.ADPS). The prescaler starts counting from the moment the ADC is switched on by writing the ADC Enable bit ADCSRA.ADEN to '1'. The prescaler keeps running for as long as ADEN=1, and is continuously reset when ADEN=0.

When initiating a single ended conversion by writing a '1' to the ADC Start Conversion bit (ADCSRA.ADSC), the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (i.e., ADCSRA.ADEN is written to '1') takes 25 ADC clock cycles in order to initialize the analog circuitry.

When the bandgap reference voltage is used as input to the ADC, it will take a certain time for the voltage to stabilize. If not stabilized, the first value read after the first conversion may be wrong.

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 first conversion. When a conversion is complete, the result is written to the ADC Data Registers (ADCL and ADCH), and the ADC Interrupt Flag (ADCSRA.ADIF) is set. In Single Conversion mode, ADCSRA.ADSC is cleared simultaneously. The software may then set ADCSRA.ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADCRSA.ADSC remains high. See also the ADC Conversion Time table below.

Figure 27-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

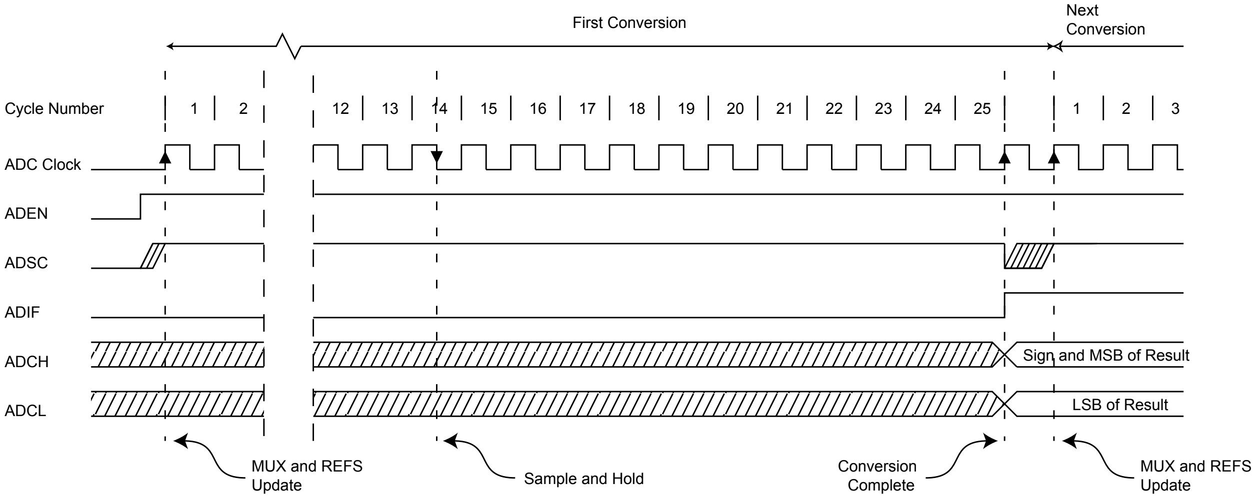


Figure 27-5. ADC Timing Diagram, Single Conversion

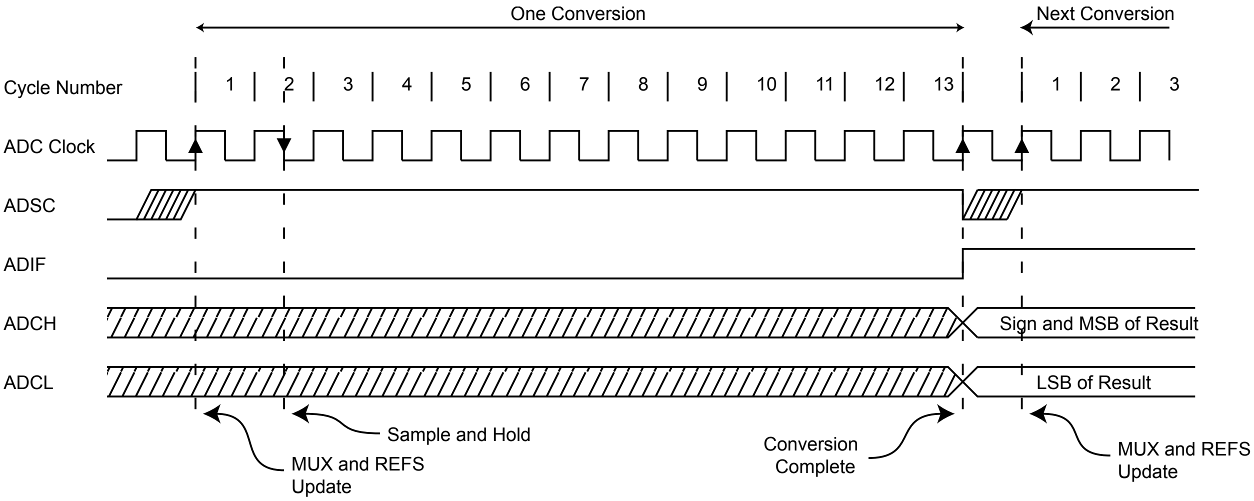


Figure 27-6. ADC Timing Diagram, Auto Triggered Conversion

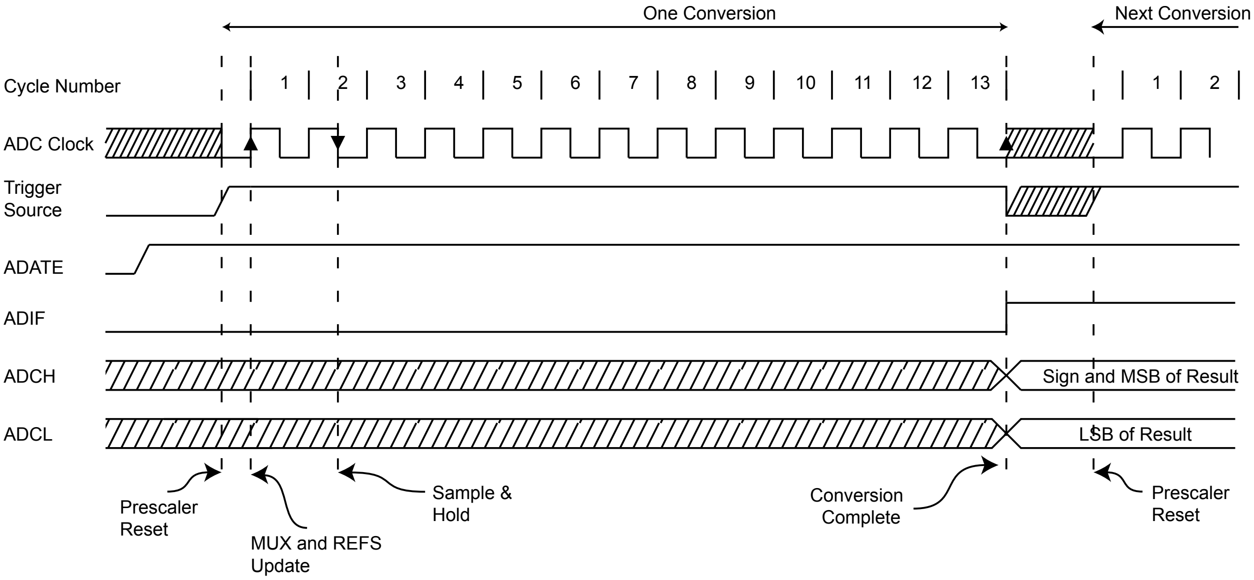


Figure 27-7. ADC Timing Diagram, Free Running Conversion

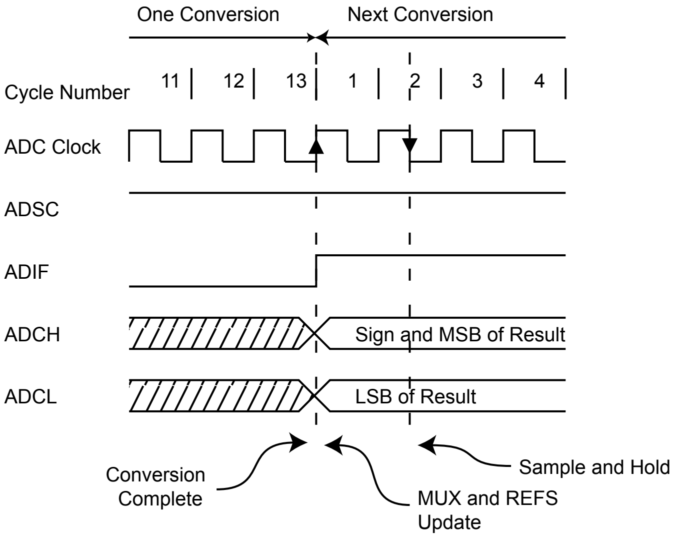


Table 27-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	14.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

27.4.1 Differential Gain Channels

When using differential gain channels, certain aspects of the conversion need to be taken into consideration. Note that the differential channels should not be used with an AREF < 2V.

Differential conversions are synchronized to the internal clock CK_{ADC2} equal to half the ADC clock. This synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CK_{ADC2}. A conversion initiated by the user (that is, all single conversions, and the first free running conversion) when CK_{ADC2} is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CK_{ADC2} is high will take 14 ADC clock cycles due to the synchronization mechanism. In Free Running mode, a new conversion is initiated immediately after the previous conversion completes, and since CK_{ADC2} is high at this time, all automatically started (that is, all but the first) free running conversions will take 14 ADC clock cycles.

The gain stage is optimized for a bandwidth of 4kHz at all gain settings. Higher frequencies may be subjected to non-linear amplification. An external low-pass filter should be used if the input signal contains higher frequency components than the gain stage bandwidth. Note that the ADC clock frequency is independent of the gain stage bandwidth limitation. For example, the ADC clock period may be 6 µs, allowing a channel to be sampled at 12kSPS, regardless of the bandwidth of this channel.

If differential gain channels are used and conversions are started by Auto Triggering, the ADC must be switched off between conversions. When Auto Triggering is used, the ADC prescaler is reset before the conversion is started. Since the gain stage is dependent of a stable ADC clock prior to the conversion, this conversion will not be valid. By disabling and then re-enabling the ADC between each conversion (writing ADEN in ADCSRA to “0” then to “1”), only extended conversions are performed. The result from the extended conversions will be valid. See *Prescaling and Conversion Timing* section

27.5 Changing Channel or Reference Selection

The Analog Channel Selection bits (MUX) and the Reference Selection bits (REFS) bits in the ADC Multiplexer Selection Register (ADMUX.MUX[4:0] and ADMUX.REFS[1:0]) are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (indicated by ADCSRA.ADIF set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after the ADC Start Conversion bit (ADCRSA.ADSC) was written.

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

If both the ADC Auto Trigger Enable and ADC Enable bits (ADCRSA.ADATE, ADCRSA.ADEN) are written to '1', an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

1. When ADATE or ADEN is cleared.
   1.1. During conversion, minimum one ADC clock cycle after the trigger event.
   1.2. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 μs to stabilize to the new value. Thus conversions should not be started within the first 125 μs after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded. The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS[1:0] bits in ADMUX).

27.5.1 ADC Input Channels

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

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

The user is advised not to write new channel or reference selection values during Free Running mode.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

27.5.2 ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AV_{CC}, internal 1.1V reference, or external AREF pin.

AV_{CC} is connected to the ADC through a passive switch. The internal 1.1V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedance voltmeter. Note that V_{REF} is a high impedance source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AV_{CC} and 1.1V as reference selection. The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

If differential channels are used, the selected reference should not be closer to AV_{CC} than indicated in ADC Characteristics of Electrical Characteristics chapter.

27.6 ADC Noise Canceler

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

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.
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. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note: The ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADCRSA.ADEN before entering such sleep modes to avoid excessive power consumption.

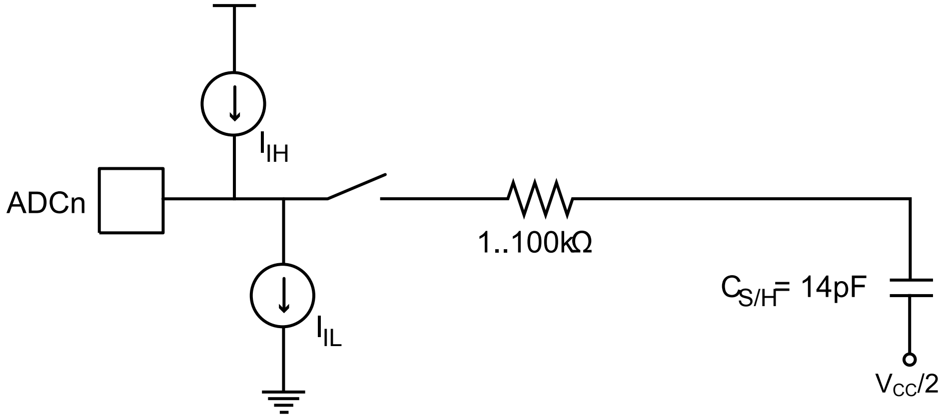
27.6.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated below. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 27-8. Analog Input Circuitry

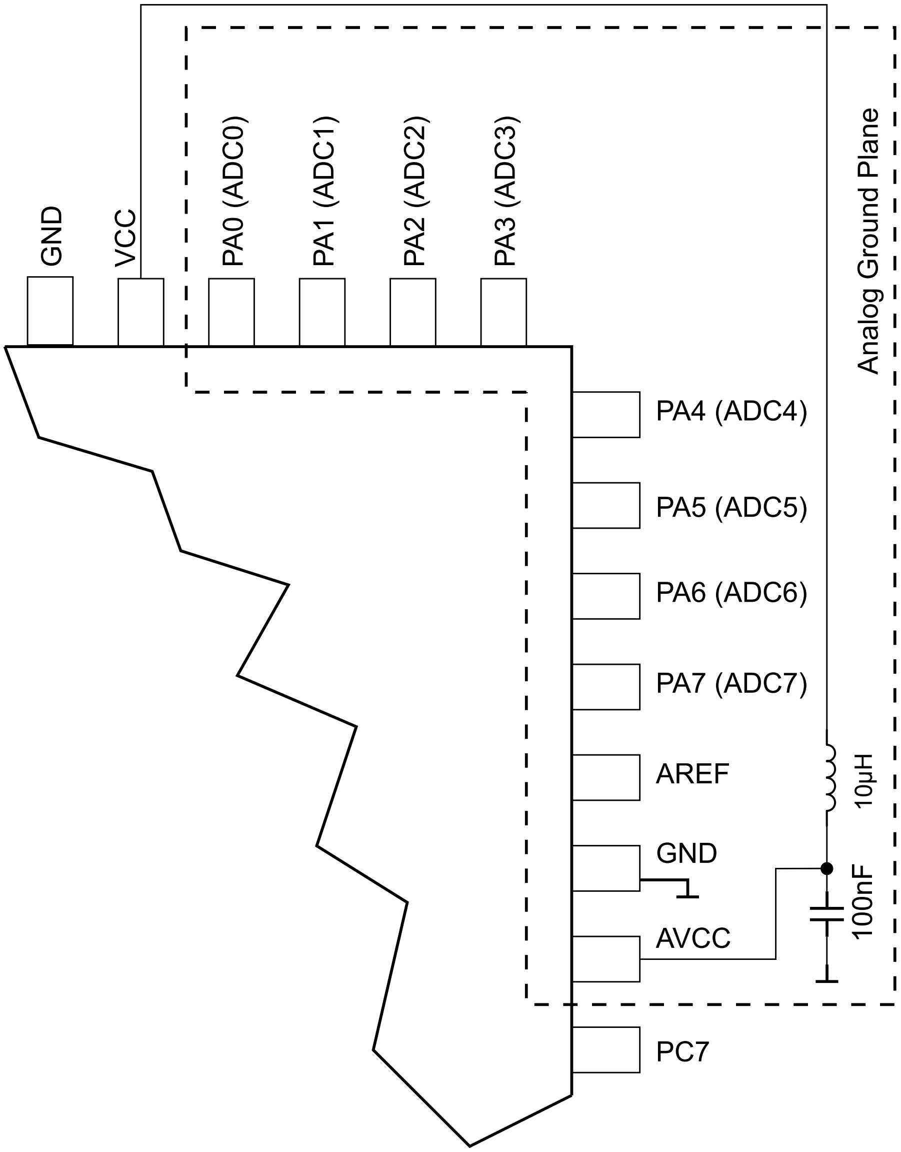


27.6.2 Analog Noise Canceling Techniques

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

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the ground plane, and keep them well away from high-speed switching digital tracks.
2. The AVCC pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in the figure below.
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Figure 27-9. ADC Power Connections



27.6.3 Offset Compensation Schemes

The gain stage has a built-in offset cancellation circuitry that nulls the offset of differential measurements as much as possible. The remaining offset in the analog path can be measured directly by selecting the same channel for both differential inputs. This offset residue can be then subtracted in software from the measurement results. Using this kind of software based offset correction, offset on any channel can be reduced below one LSB.

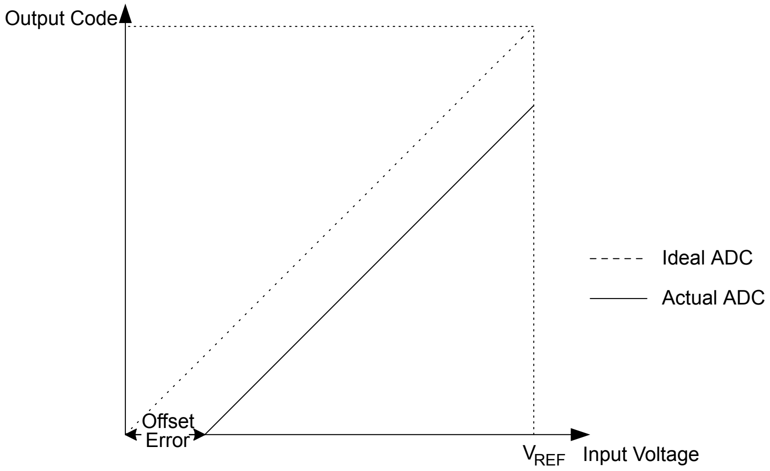
27.6.4 ADC Accuracy Definitions

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

Several parameters describe the deviation from the ideal behavior:

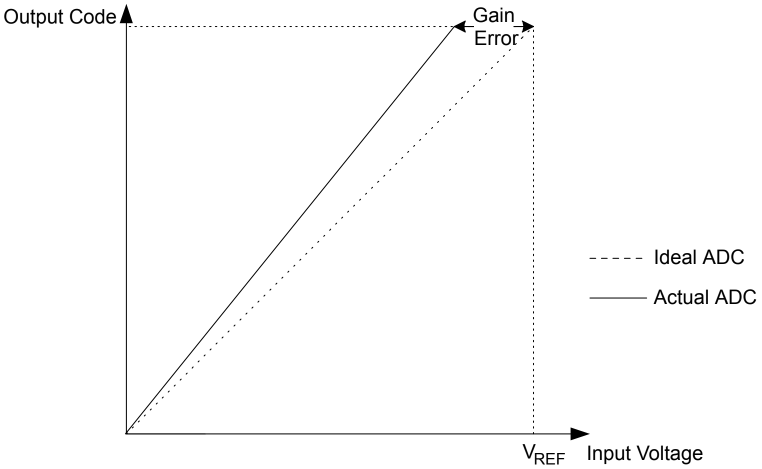
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 27-10. Offset Error



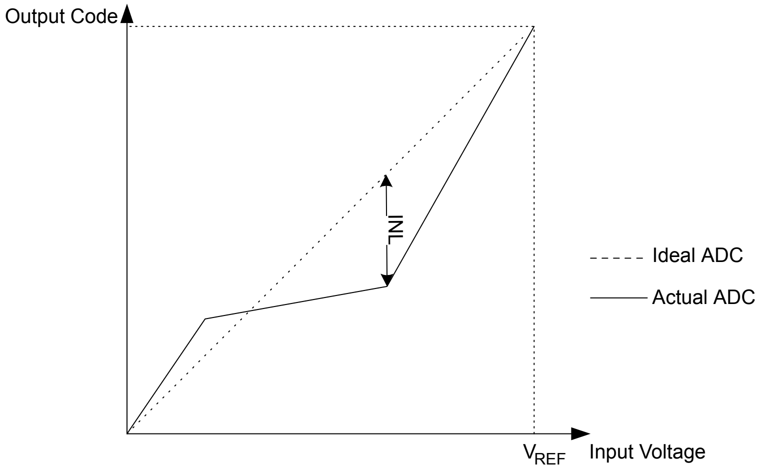
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

Figure 27-11. Gain Error



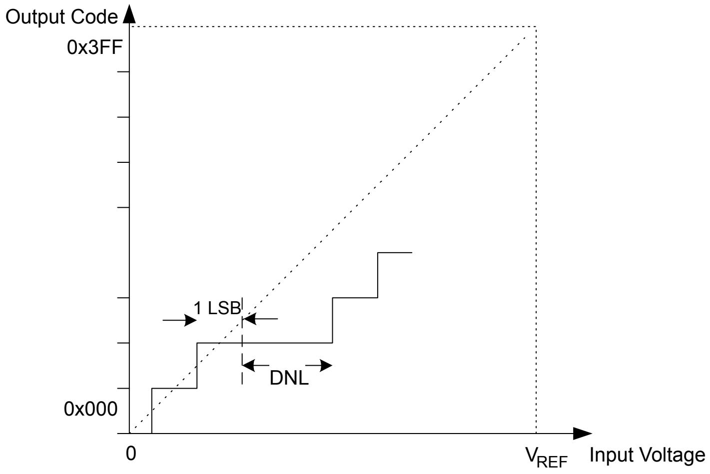
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 27-12. Integral Non-linearity (INL)



- Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 27-13. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ±0.5 LSB.
- Absolute accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ±0.5 LSB.

27.7 ADC Conversion Result

After the conversion is complete (ADCSRA.ADIF is set), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

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

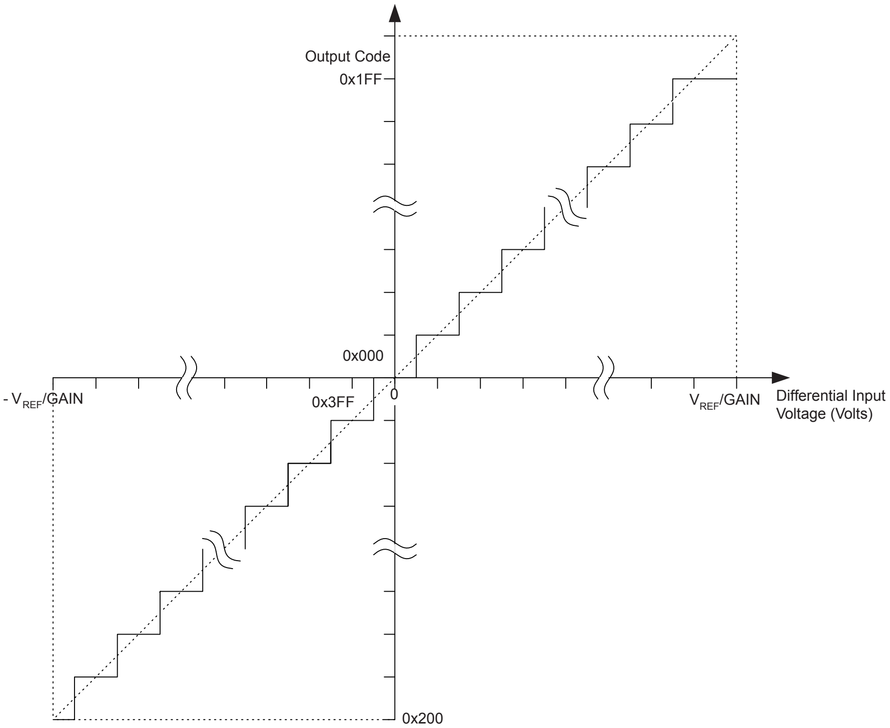
where V_{IN} is the voltage on the selected input pin, and V_{REF} the selected voltage reference (see also descriptions of ADMUX.REFSn and ADMUX.MUX). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is

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

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, GAIN the selected gain factor, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the results, it is sufficient to read the MSB of the result (ADC9 in ADCH). If this bit is one, the result is negative, and if this bit is zero, the result is positive. The figure below shows the decoding of the differential input range.

Figure 27-14. Differential Measurement Range



The table below shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a gain of GAIN and a reference voltage of VREF.

Table 27-2. Correlation between Input Voltage and Output Codes

V_{ADCn}	Read code	Corresponding Decimal Value
V_{ADCm} + V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 0.999 V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 0.998 V_{REF}/GAIN	0x1FE	510
...	...	...
V_{ADCm} + 0.001 V_{REF}/GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} - 0.001 V_{REF}/GAIN	0x3FF	-1
...	...	...
V_{ADCm} - 0.999 V_{REF}/GAIN	0x201	-511
V_{ADCm} - V_{REF}/GAIN	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10× gain, 2.56V reference, left adjusted result)

Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.

ADCR = 512 × 10 × (300 - 500) / 2560 = -400 = 0x270

ADCL will thus read 0x00, and ADCH will read 0x9C.

Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

27.8 Register Description

27.8.1 ADC Multiplexer Selection Register

Name: ADMUX
Offset: 0x7C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	REFS [1:0]		ADLAR	MUX [4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:6 – REFS [1:0]: Reference Selection
These bits select the voltage reference for the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 27-3. ADC Voltage Reference Selection

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal V_{ref} turned off
01	AV_{CC} with external capacitor at AREF pin
10	Internal 1.1V Voltage Reference with external capacitor at AREF pin
11	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Note: If differential channels are selected, only 2.56V should be used as Internal Voltage Reference.

Bit 5 – ADLAR: ADC Left Adjust Result
The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see the ADCL and ADCH.

Bits 4:0 – MUX [4:0]: Analog Channel and Gain Selection Bits
The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete. (ADIF in ADCSRA is set).

Table 27-4. Input Channel and Gain Selections

MUX[4:0]	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000	N/A	ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x

MUX[4:0]	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
11110	1.1V (V_{BG})	N/A		
11111	0V (GND)			

27.8.2 ADC Control and Status Register A

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS [2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – ADEN: ADC Enable
Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

Bit 6 – ADSC: ADC Start Conversion
In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

Bit 5 – ADATE: ADC Auto Trigger Enable
When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

Bit 4 – ADIF: ADC Interrupt Flag
This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

Bit 3 – ADIE: ADC Interrupt Enable
When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

Bits 2:0 – ADPS [2:0]: ADC Prescaler Select
These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 27-5. Input Channel Selection

ADPS[2:0]	Division Factor
000	2
001	2
010	4
011	8
100	16
101	32
110	64
111	128

27.8.3 ADC Data Register Low and High Byte (ADLAR=0)

The ADCL and ADCH register pair represents the 16-bit value, ADC Data Register. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

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

If differential channels are used, the result is presented in two's complement form.

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 and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set (ADLAR=1), the result is left adjusted. If ADLAR is cleared (ADLAR=0 which is the default value), the result is right adjusted.

Name: ADCL and ADCH
Offset: 0x78
Reset: 0x00
Property: ADLAR = 0

Bit	15	14	13	12	11	10	9	8
							ADC[9:8]	
Access							R	R
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	ADC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 9:0 – ADC[9:0]: ADC Conversion Result
These bits represent the result from the conversion. Refer to ADC Conversion Result for details.

27.8.4 ADC Data Register Low and High Byte (ADLAR=1)

The ADCL and ADCH register pair represents the 16-bit value, ADC Data Register. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

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

If differential channels are used, the result is presented in two's complement form.

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 and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set (ADLAR=1), the result is left adjusted. If ADLAR is cleared (ADLAR=0 which is the default value), the result is right adjusted.

Name: ADCL and ADCH
Offset: 0x78
Reset: 0x00
Property: ADLAR = 1

Bit	15	14	13	12	11	10	9	8
	ADC[9:2]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADC[1:0]							
Access	R	R						
Reset	0	0						

Bits 15:6 – ADC[9:0]: ADC Conversion Result

These bits represent the result from the conversion. Refer to ADC Conversion Result for details.

27.8.5 ADC Control and Status Register B

Name: ADCSRB
Offset: 0x7B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		ACME				ADTS [2:0]		
Access		R/W				R/W	R/W	R/W
Reset		0				0	0	0

Bit 6 – ACME: Analog Comparator Multiplexer Enable

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see *Analog Comparator Multiplexed Input*..

Bits 2:0 – ADTS [2:0]: ADC Auto Trigger Source

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS[2:0] settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 27-6. ADC Auto Trigger Source Selection

ADTS[2:0]	Trigger Source
000	Free Running mode
001	Analog Comparator
010	External Interrupt Request 0
011	Timer/Counter0 Compare Match A
100	Timer/Counter0 Overflow
101	Timer/Counter1 Compare Match B
110	Timer/Counter1 Overflow
111	Timer/Counter1 Capture Event

Related Links

Analog Comparator Multiplexed Input

27.8.6 Digital Input Disable Register 0

When the respective bits are written to logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7...0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Name: DIDR0
Offset: 0x7E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ADC0D, ADC1D, ADC2D, ADC3D, ADC4D, ADC5D, ADC6D, ADC7D: ADC Digital Input Disable

28. PTC - Peripheral Touch Controller

28.1 Overview

The Peripheral Touch Controller (PTC) acquires signals in order to detect touch on capacitive sensors. The external capacitive touch sensor is typically formed on a PCB, and the sensor electrodes are connected to the analog front end of the PTC through the I/O pins in the device. The PTC supports both self- and mutual-capacitance sensors.

In mutual-capacitance mode, sensing is done using capacitive touch matrices in various X-Y configurations, including indium tin oxide (ITO) sensor grids. The PTC requires one pin per X-line and one pin per Y-line.

In self-capacitance mode, the PTC requires only one pin (Y-line) for each touch sensor.

The number of available pins and the assignment of X- and Y-lines is depending on both package type and device configuration. Refer to the Configuration Summary and I/O Multiplexing table for details.

28.2 Features

- Low-power, high-sensitivity, environmentally robust capacitive touch buttons, sliders, wheels and proximity sensing
- Supports wake-up on touch from power-save sleep mode
- Supports mutual capacitance and self-capacitance sensing
  - Mix-and-match mutual-and self-capacitance sensors.
  - 256 buttons in mutual-capacitance mode

  Refer *Features* Section of this device for information on the number of buttons supported.
- One pin per electrode – no external components
- Load compensating charge sensing
  - Parasitic capacitance compensation and adjustable gain for superior sensitivity
- Zero drift over the temperature and V_{DD} range
  - Auto calibration and re-calibration of sensors
- Single-shot and free-running charge measurement
- Hardware noise filtering and noise signal de-synchronization for high conducted immunity
- Selectable channel change delay
  - Allows choosing the settling time on a new channel, as required
- Acquisition-start triggered by command or through auto-triggering feature
- Low CPU utilization through interrupt on acquisition-complete
  - 5% CPU utilization scanning 10 channels at 50ms scan rate

28.3 Block Diagram

Figure 28-1. PTC Block Diagram Mutual-Capacitance

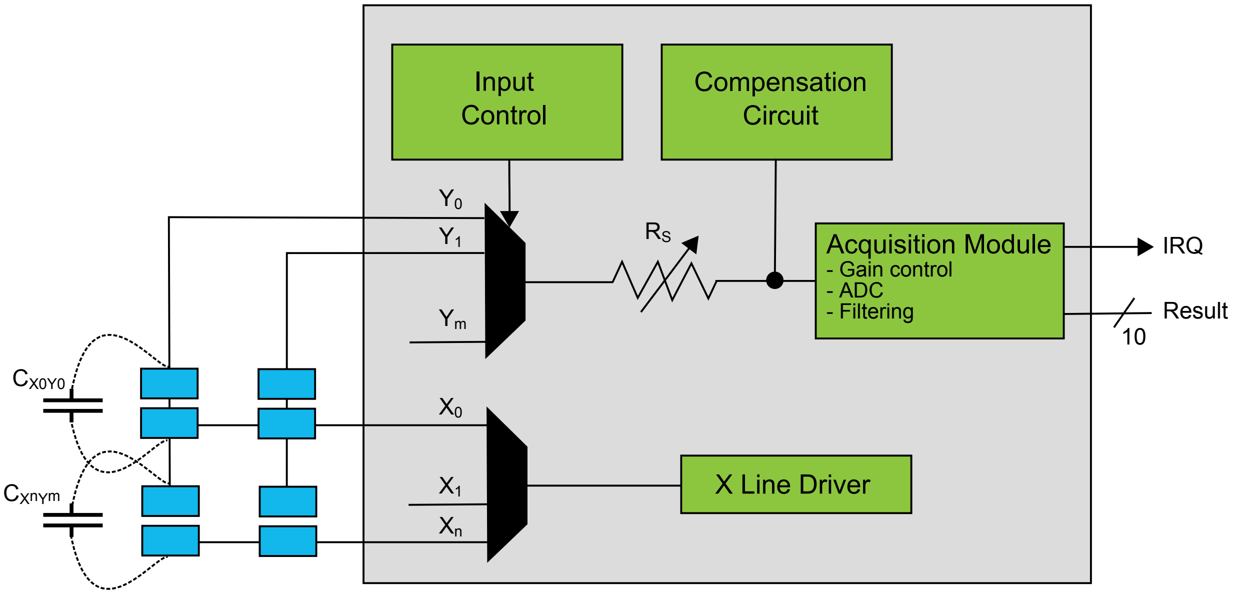


Figure 28-2. PTC Block Diagram Self-Capacitance

28.4 Signal Description

Name	Type	Description
X[n:0]	Digital	X-line (Output)
Y[m:0]	Analog	Y-line (Input/Output)

Note: The number of X and Y lines are device dependent. Refer to *Configuration Summary* for details.

Refer to *I/O Multiplexing and Considerations* for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

Related Links

I/O Multiplexing

28.5 System Dependencies

In order to use this Peripheral, configure the other components of the system as described in the following sections.

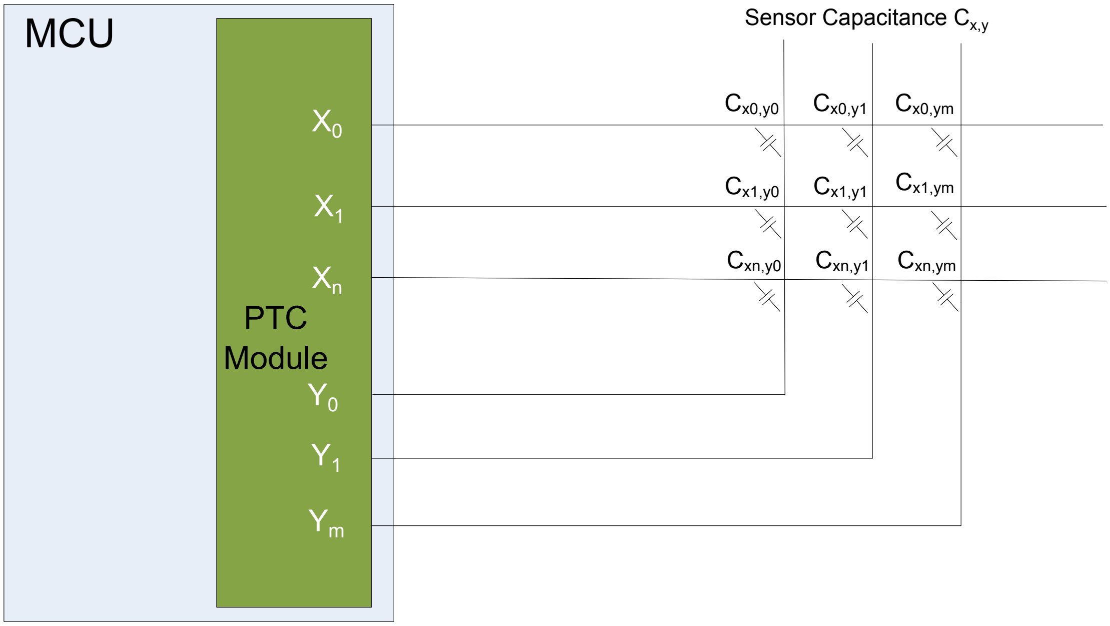
28.5.1 I/O Lines

The I/O lines used for analog X-lines and Y-lines must be connected to external capacitive touch sensor electrodes. External components are not required for normal operation. However, to improve the EMC performance, a series resistor of 1kΩ or more can be used on X-lines and Y-lines.

28.5.1.1 Mutual-Capacitance Sensor Arrangement

A mutual-capacitance sensor is formed between two I/O lines - an X electrode for transmitting and Y electrode for sensing. The mutual capacitance between the X and Y electrode is measured by the Peripheral Touch Controller.

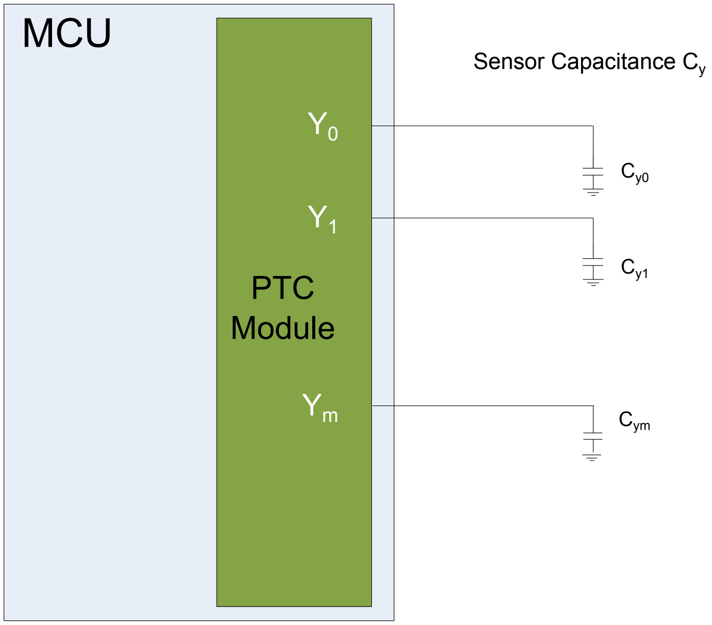
Figure 28-3. Mutual Capacitance Sensor Arrangement



28.5.1.2 Self-Capacitance Sensor Arrangement

A self-capacitance sensor is connected to a single pin on the Peripheral Touch Controller through the Y electrode for sensing the signal. The sense electrode capacitance is measured by the Peripheral Touch Controller.

Figure 28-4. Self-capacitance Sensor Arrangement

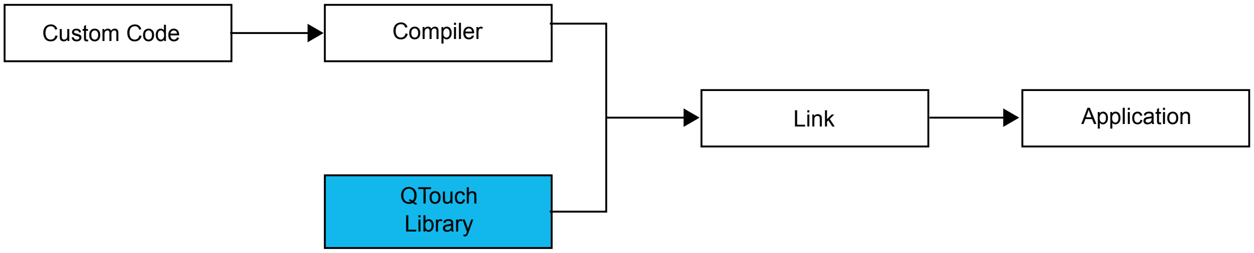


For more information about designing the touch sensor, refer to Buttons, Sliders and Wheels Touch Sensor Design Guide.

28.6 Functional Description

In order to access the PTC, the user must use the Atmel Start QTouch Configurator to configure and link the QTouch Library firmware with the application software. QTouch Library can be used to implement buttons, sliders, wheels in a variety of combinations on a single interface.

Figure 28-5. QTouch Library Usage



For more information about QTouch Library, refer to the QTouch Library Peripheral Touch Controller User Guide.

29. JTAG Interface and On-chip Debug System

29.1 Features

- JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the IEEE std. 1149.1 (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:
  - AVR *Break* Instruction
  - 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 Studio

29.2 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* and IEEE 1149.1 (JTAG) Boundary-scan, respectively. The On-chip Debug support is considered being private JTAG instructions and to selected third party vendors only.

Figure 29-1 shows 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.

The ID-Register, Bypass Register, and the Boundary-scan Chain are the data registers 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.

Related Links

Programming Via the JTAG Interface

29.3 TAP – Test Access Port

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 and the JTD bit in MCUCSR is cleared, the TAP input signals are internally pulled high and the JTAG is enabled for Boundary-scan and programming. In this case, the TAP output pin (TDO) is left floating in states where the JTAG TAP controller is not shifting data, and must therefore be connected to a pull-up resistor or other hardware having pull-ups (for instance the TDI-input of the next device in the scan chain). The device is shipped with this fuse programmed.

For the On-chip Debug system, in addition to the JTAG interface pins, 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 the Reset line are used in the application.

Figure 29-1. Block Diagram

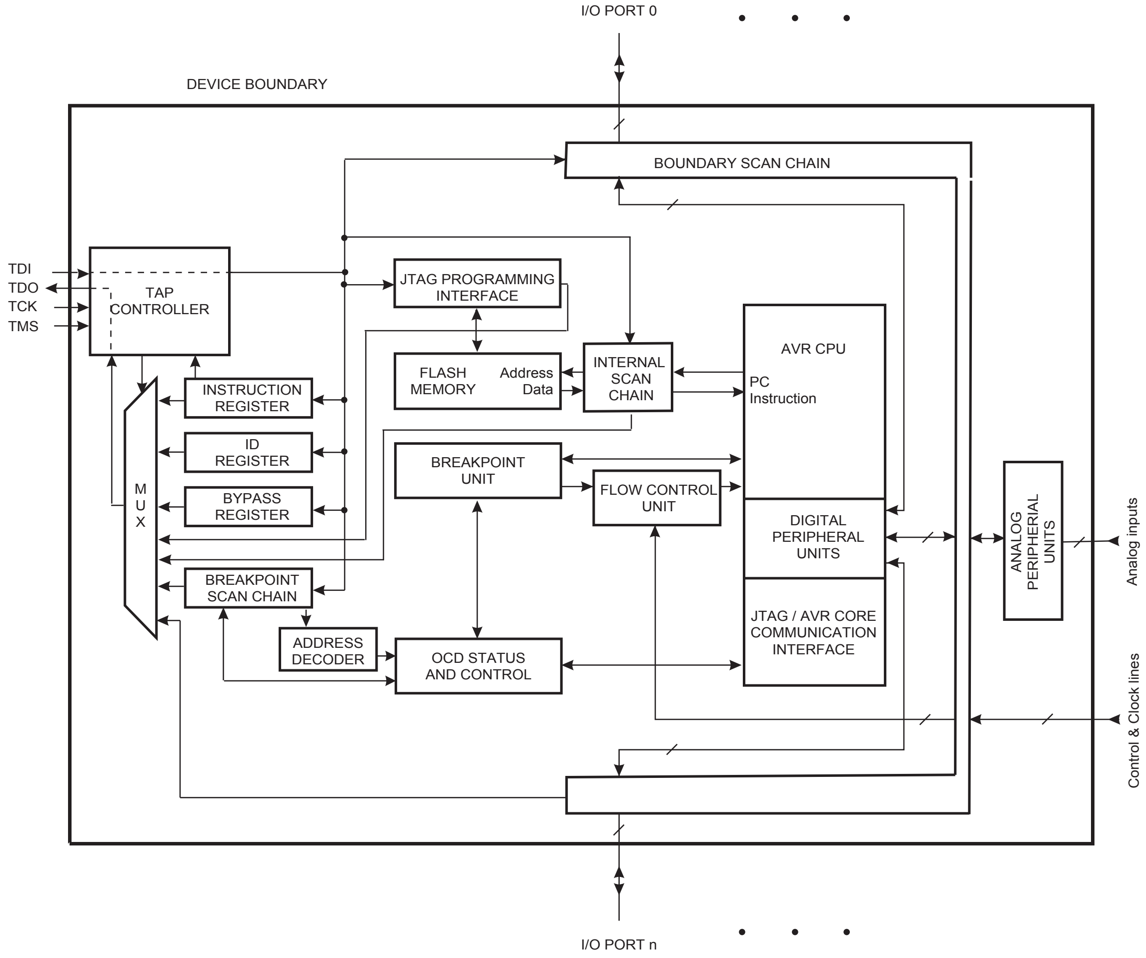
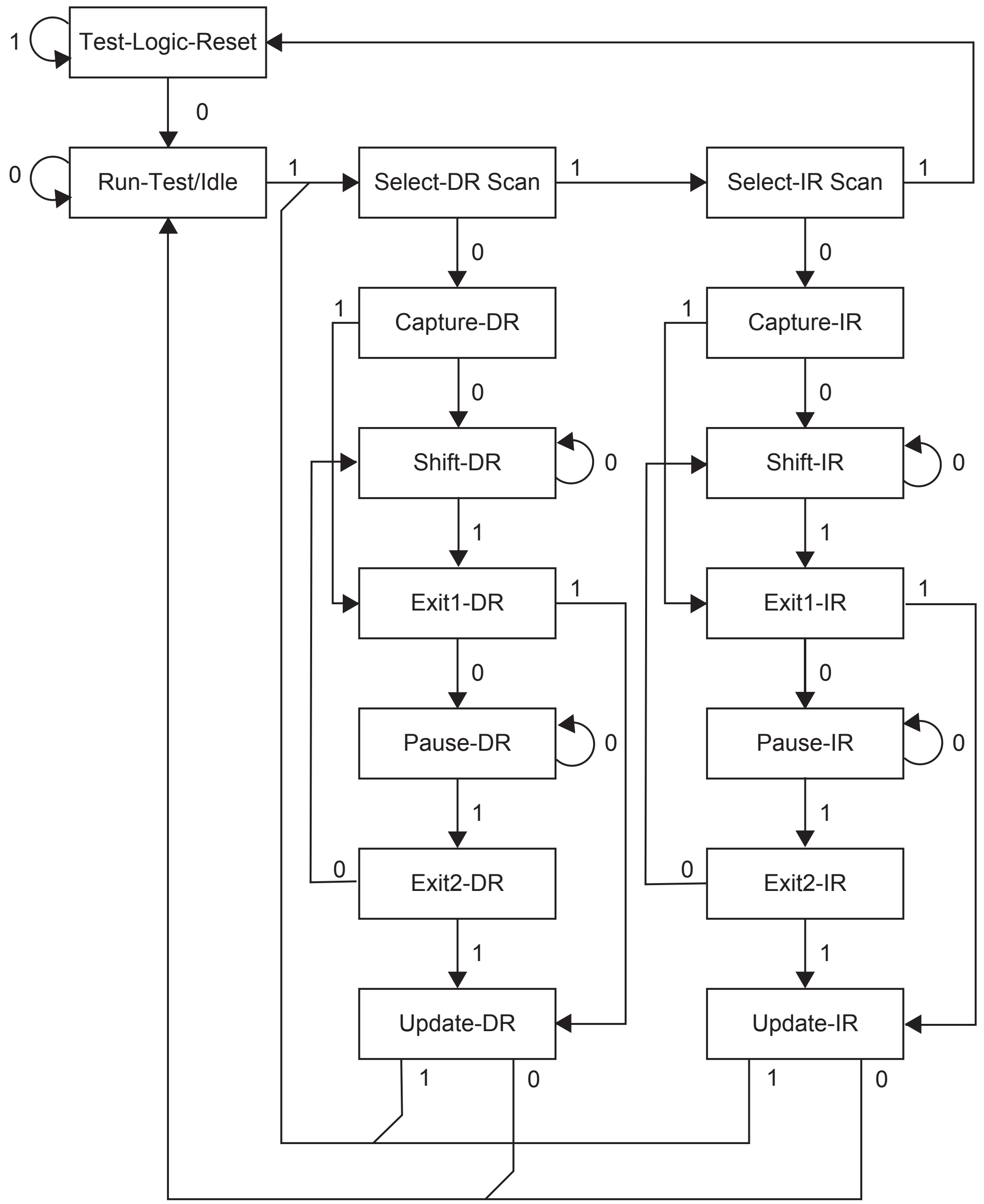


Figure 29-2. TAP Controller State Diagram



29.4 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 29-2 depend 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 in this state, shift the 4 bits of the JTAG instructions into the JTAG instruction register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted 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 in this state, 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. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted 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: 1. 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.

29.5 Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section IEEE 1149.1 (JTAG) Boundary-scan.

29.6 Using the On-chip Debug System

As shown in Figure 29-1, 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
- Break point 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 Break point Unit implements Break on Change of Program Flow, Single Step Break, two Program Memory Break points, and two combined break points. Together, the four break points can be configured as either:

- 4 Single Program Memory break points
- 3 Single Program Memory break points + 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 debugger, like the Studio®, may however use one or more of these resources for its internal purpose, leaving less flexibility to the end-user.

A list of the On-chip Debug specific JTAG instructions is given in On-chip Debug Specific JTAG Instructions.

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. As a security feature, the On-chip Debug system is disabled when any Lock bits are set. Otherwise, the On-chip Debug system would have provided a back-door into a secured device.

The 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. Studio supports source level execution of Assembly programs assembled with AVR Assembler and C programs compiled with third party vendors' compilers.

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

All necessary execution commands are available in 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 an unlimited number of code break points (using the BREAK instruction) and up to two data memory break points, alternatively combined as a mask (range) break point.

29.7 On-chip Debug Specific JTAG Instructions

The On-chip debug support is considered being private JTAG instructions and to selected third-party vendors only. Instruction opcodes are listed for reference.

PRIVATE0; 0x8

Private JTAG instruction for accessing On-chip Debug system.

PRIVATE1; 0x9

Private JTAG instruction for accessing On-chip Debug system.

PRIVATE2; 0xA

Private JTAG instruction for accessing On-chip Debug system.

PRIVATE3; 0xB

Private JTAG instruction for accessing On-chip Debug system.

29.8 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 MCUCSR 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.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section *Programming Via the JTAG Interface*.

Related Links

Programming Via the JTAG Interface

29.9 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

29.10 IEEE 1149.1 (JTAG) Boundary-scan

Related Links

Reset Sources

29.10.1 Features

- JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the JTAG Standard
- Full Scan of all Port Functions as well as Analog Circuitry having Off-chip Connections
- Supports the Optional IDCODE Instruction
- Additional Public AVR_RESET Instruction to Reset the AVR

29.10.2 System Overview

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. 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 four 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 MCUCSR 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.

29.11 Data Registers

The data registers relevant for Boundary-scan operations are:

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

29.11.1 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.

29.11.2 Device Identification Register

The figure below shows the structure of the Device Identification Register.

Figure 29-3. The format of the Device Identification Register

	MSB			LSB
Bit	31 28	27 12	11 1	0
Device ID	**Version**	**Part Number**	**Manufacturer ID**	**1**
	4 bits	16 bits	11 bits	1-bit

29.11.2.1 Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device, and wraps around at revision P (0xF). Revision A and Q is 0x0, revision B and R is 0x1 and so on.

29.11.2.2 Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for ATmega324PB is listed in the table below.

Table 29-1. AVR JTAG Part Number

Part Number	JTAG Part Number
ATmega324PB	0x9517

29.11.2.3 Manufacturer ID

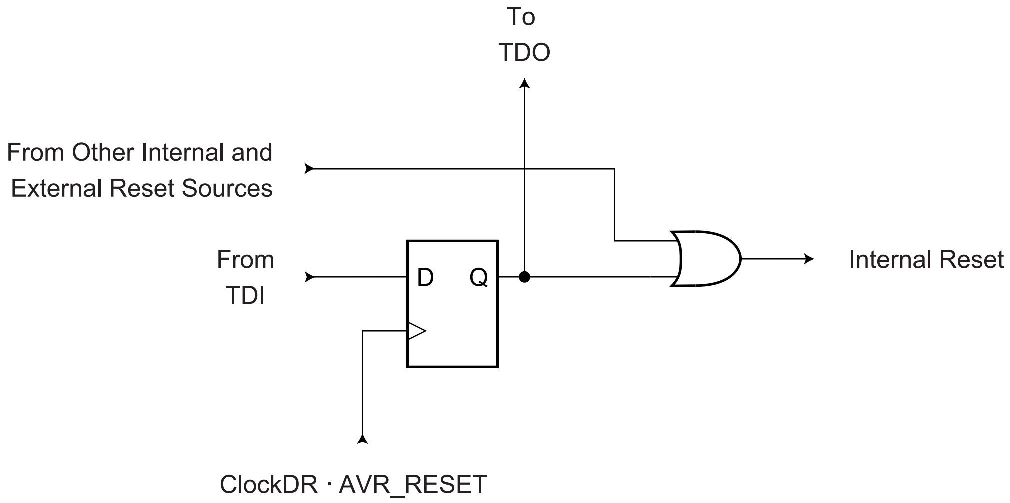
The Manufacturer ID is a 11-bit code identifying the manufacturer. The JTAG manufacturer ID is 0x01F.

29.11.3 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 *Clock Sources*) after releasing the Reset Register. The output from this Data Register is not latched, so the Reset will take place immediately, as shown in the figure below.

Figure 29-4. Reset Register



29.11.4 Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. Refer to Boundary-scan Chain for a complete description.

29.12 Boundry-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.

29.12.1 EXTEST; 0x0

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 with 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.

29.12.2 IDCODE; 0x1

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.

29.12.3 SAMPLE_PRELOAD; 0x2

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.

29.12.4 AVR_RESET; 0xC

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.

29.12.5 BYPASS; 0xF

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.

29.13 Boundary-scan Chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections.

29.13.1 Scanning the Digital Port Pins

The first figure below shows the Boundary-scan Cell for a bi-directional port pin with pull-up function. The cell consists of a standard Boundary-scan cell for the Pull-up Enable – PUExn – function, and a bi-directional pin cell that combines the three signals, Output Control – OCxn, Output Data – ODxn, and Input Data – IDxn, into only a two-stage Shift Register. The port and pin indexes are not used in the following description

The Boundary-scan logic is not included in the figures in the Data Sheet. Figure 29-6 shows a simple digital Port Pin as described in the section *I/O Ports*. The Boundary-scan details from the first figure below replaces the dashed box in Figure 29-6.

When no alternate port function is present, the Input Data – ID corresponds to the PINxn Register value (but ID has no synchronizer), Output Data corresponds to the PORT Register, Output Control corresponds to the Data Direction – DD Register, and the Pull-up Enable – PUExn – corresponds to logic expression $\overline{\text{PUD}} \cdot \overline{\text{DDxn}} \cdot \text{PORTxn}$.

Digital alternate port functions are connected outside the dotted box in Figure 29-6 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 circuitry.

Figure 29-5. Boundary-scan Cell for Bi-directional Port Pin with Pull-Up Function.

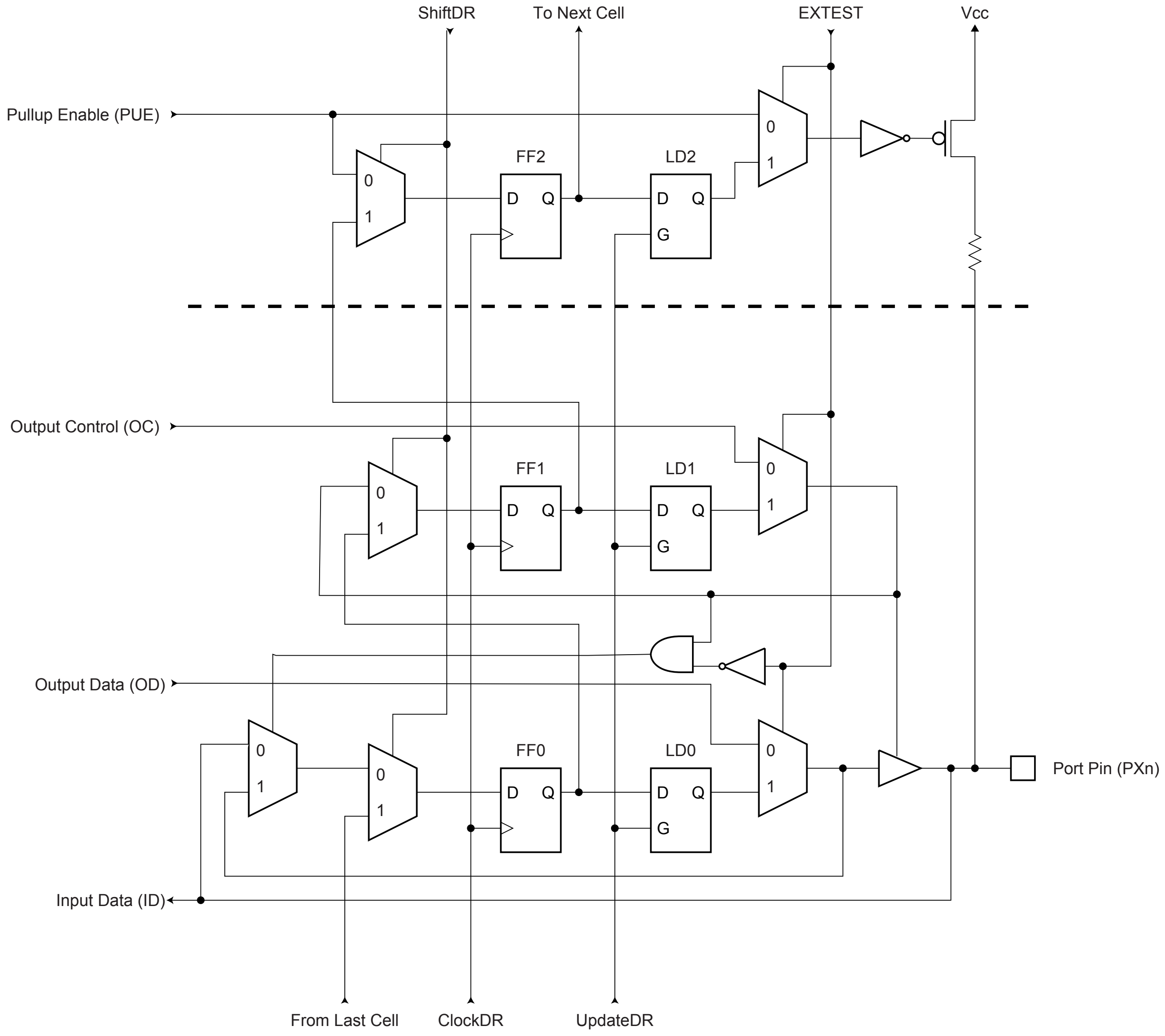
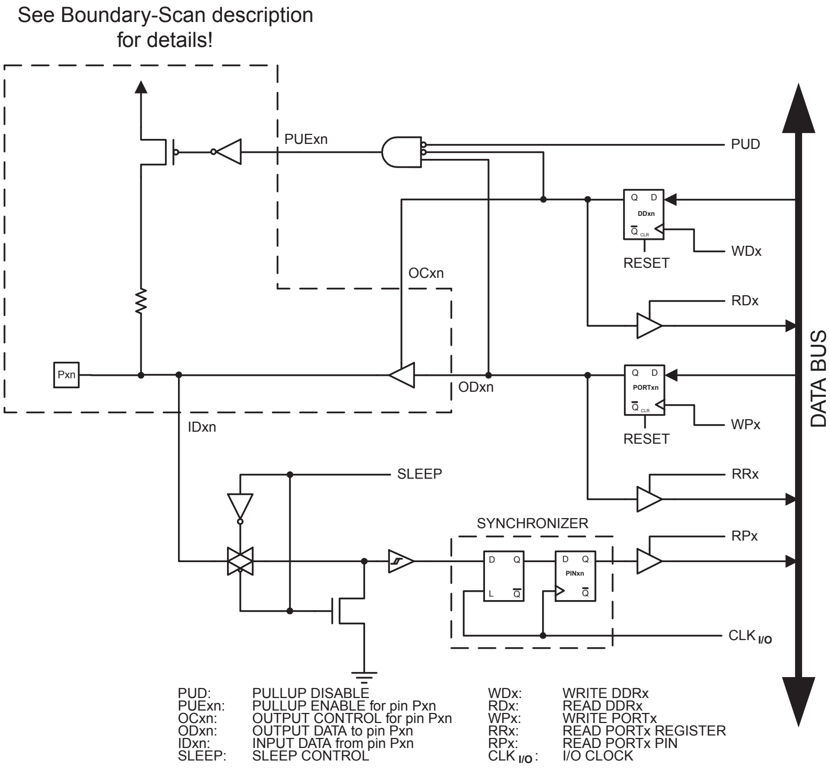


Figure 29-6. General Port Pin Schematic diagram



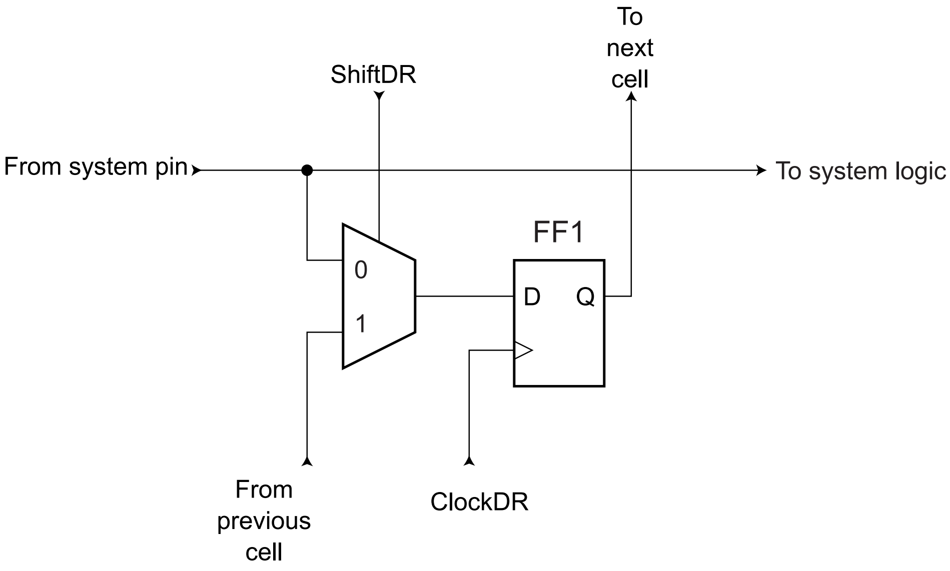
Related Links

I/O-Ports

29.13.2 Scanning the RESET Pin

The RESET pin accepts 5V active low logic for standard Reset operation, and 12V active high logic for High Voltage Parallel programming. An observe-only cell as shown in the figure below is inserted both for the 5V Reset signal; RSTT, and the 12V Reset signal; RSTHV.

Figure 29-7. Observe-only Cell



29.14 ATmega324PB Boundary-scan Order

The table below 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 are 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 29-5, PXn. Data corresponds to FF0, PXn. Control corresponds to FF1, and PXn. Pullup_enable 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 29-2. ATmega324PB Boundary-scan Order

Bit Number	Signal Name	Module
58	PB7.Data	Port B
57	PB7.Control	
56	RSTT	Reset Logic (Observe Only)
55	PE0.Data	Port E
54	PE0.Control	
53	PE1.Data	
52	PE1.Control	
51	PE2.Data	
50	PE2.Control	
49	PE3.Data	
48	PE3.Control	
47	PE4.Data	
46	PE4.Control	
45	PE5.Data	
44	PE5.Control	
43	PE6.Data	
42	PE6.Control	
41	PE7.Data	
40	PE7.Control	
39	PD0.Data	Port D
38	PD0.Control	
37	PD1.Data	
36	PD1.Control	

Bit Number	Signal Name	Module
35	PD2.Data	
34	PD2.Control	
33	PD3.Data	
32	PD3.Control	
31	PD4.Data	
30	PD4.Control	
29	PD5.Data	
28	PD5.Control	
27	PD6.Data	
26	PD6.Control	
25	PD7.Data	
24	PD7.Control	
23	PC0.Data	Port C
22	PC0.Control	
21	PC1.Data	
20	PC1.Control	
19	PC6.Data	
18	PC6.Control	
17	PC7.Data	
16	PC7.Control	
15	PA7.Data	Port A
14	PA7.Control	
13	PA6.Data	
12	PA6.Control	
11	PA5.Data	
10	PA5.Control	
9	PA4.Data	
8	PA4.Control	
7	PA3.Data	
6	PA3.Control	
5	PA2.Data	
4	PA2.Control	

Bit Number	Signal Name	Module
3	PA1.Data	
2	PA1.Control	
1	PA0.Data	
0	PA0.Control	

29.15 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.

29.16 Register Description

29.16.1 OCDR – On-chip Debug Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: OCDR
Offset: 0x51
Reset: 0x20
Property: When addressing as I/O Register: address offset is 0x31

Bit	7	6	5	4	3	2	1	0
	IDRD/OCDR7	OCDR 6	OCDR 5	OCDR 4	OCDR 3	OCDR 2	OCDR 1	OCDR 0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – IDRD/OCDR7: USART Receive Complete
The OCDR Register provides a communication channel from the running program in the microcontroller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an internal flag; I/O Debug Register Dirty – IDRD – is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information.
In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

- Bit 7 is MSB
- Bit 1 is LSB

Refer to the debugger documentation for further information on how to use this register.

Bits 0, 1, 2, 3, 4, 5, 6 – OCDR : On-chip Debug Register

29.16.2 MCU Control Register

The MCU Control Register controls the placement of the Interrupt Vector table in order to move interrupts between application and boot space.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: MCUCR
Offset: 0x55
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x35

Bit	7	6	5	4	3	2	1	0
	JTD	BODS	BODSE	PUD			IVSEL	IVCE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 7 – JTD
When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

Bit 6 – BODS: BOD Sleep
The BODS bit must be written to '1' in order to turn off BOD during sleep. Writing to the BODS bit is controlled by a timed sequence and the enable bit BODSE. To disable BOD in relevant sleep modes, both BODS and BODSE must first be written to '1'. Then, BODS must be written to '1' and BODSE must be written to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

Bit 5 – BODSE: BOD Sleep Enable
BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

Bit 4 – PUD: Pull-up Disable
When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

Bit 1 – IVSEL: Interrupt Vector Select
When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.

2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

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

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section.

Bit 0 – IVCE: Interrupt Vector Change Enable
The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example

```
Move_interrupts:
; Get MCUCR
in    r16, MCUCR
mov   r17, r16
; Enable change of Interrupt Vectors
ori   r16, (1<<IVCE)
out   MCUCR, r16
; Move interrupts to Boot Flash section
ori   r17, (1<<IVSEL)
out   MCUCR, r17
ret
```

C Code Example

```
void Move_interrupts(void)
{
uchar temp;
/* GET MCUCR*/
temp = MCUCR;
/* Enable change of Interrupt Vectors */
MCUCR = temp|(1<<IVCE);
/* Move interrupts to Boot Flash section */
MCUCR = temp|(1<<IVSEL);
}
```

29.16.3 MCU Status Register

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.
When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: MCUSR
Offset: 0x54 [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x34

Bit	7	6	5	4	3	2	1	0
				JTRF	WDRF	BORF	EXTRF	PORF
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

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 System Reset Flag
This bit is set if a Watchdog System Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

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 '0' to it.

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 '0' to it.

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 '0' to it.

30. BTLDR - Boot Loader Support – Read-While-Write Self-Programming

30.1 Features

- Read-While-Write Self-Programming
- Flexible Boot Memory Size
- High Security (Separate Boot Lock Bits for a Flexible Protection)
- Separate Fuse to Select Reset Vector
- Optimized Page(1) Size
- Code Efficient Algorithm
- Efficient Read-Modify-Write Support

Note: 1. A page is a section in the Flash consisting of several bytes (see Table. No. of Words in a Page and No. of Pages in the Flash in *Page Size*) used during programming. The page organization does not affect normal operation.

Related Links

Page Size

30.2 Overview

In this device, the Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

30.3 Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section. The size of the different sections is configured by the BOOTSZ Fuses. These two sections can have different level of protection since they have different sets of Lock bits.

30.3.1 Application Section

The Application section is the section of the Flash that is used for storing the application code. The protection level for the Application section can be selected by the application Boot Lock bits (Boot Lock bits 0). The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

30.3.2 BLS – Boot Loader Section

While the Application section is used for storing the application code, the Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1).

30.4 Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on which address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in the *Boot Loader Parameters* section and Figure 30-2. The main difference between the two sections is:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation

The user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax “Read-While-Write section” refers to which section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.

Related Links

Boot Loader Parameters

30.4.1 RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an on-going programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control and Status Register (SPMCSR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. Please refer to SPMCSR – Store Program Memory Control and Status Register in this chapter for details on how to clear RWWSB.

30.4.2 NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire Page Erase or Page Write operation.

Table 30-1. Read-While-Write Features

Which Section does the Z-pointer Address during the Programming?	Which Section can be read during Programming?	CPU Halted?	Read-While-Write Supported?
RWW Section	NRWW Section	No	Yes
NRWW Section	None	Yes	No

Figure 30-1. Read-While-Write vs. No Read-While-Write

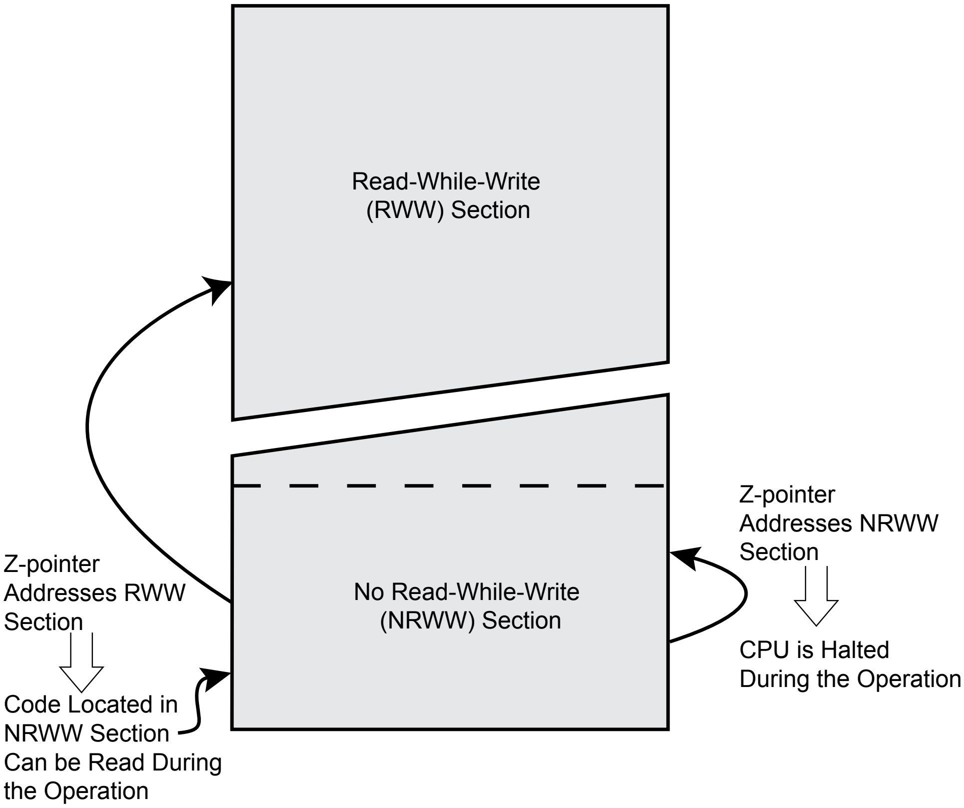
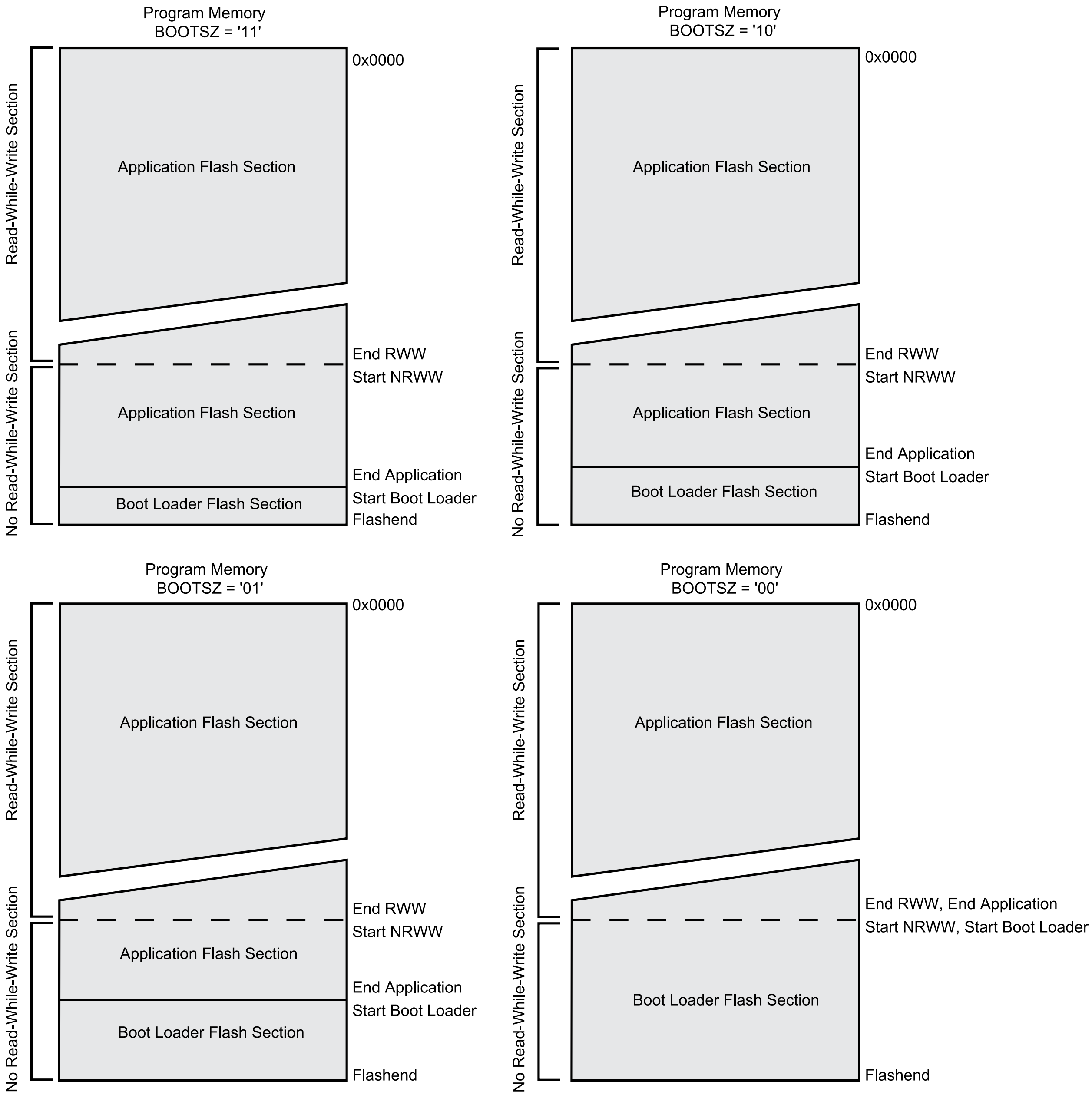


Figure 30-2. Memory Sections



Related Links

Boot Loader Parameters

30.5 Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. 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 30-2. Boot Reset Fuse

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset, as described by the Boot Loader Parameters

Note: '1' means unprogrammed, '0' means programmed.

30.6 Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU
- To protect only the Boot Loader Flash section from a software update by the MCU
- To protect only the Application Flash section from a software update by the MCU
- Allow software update in the entire Flash

The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 1) does not control reading nor writing by LPM/SPM, if it is attempted.

Table 30-3. 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 30-4. 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.

BLB1 Mode	BLB12	BLB11	Protection
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.

30.7 Addressing the Flash During Self-Programming

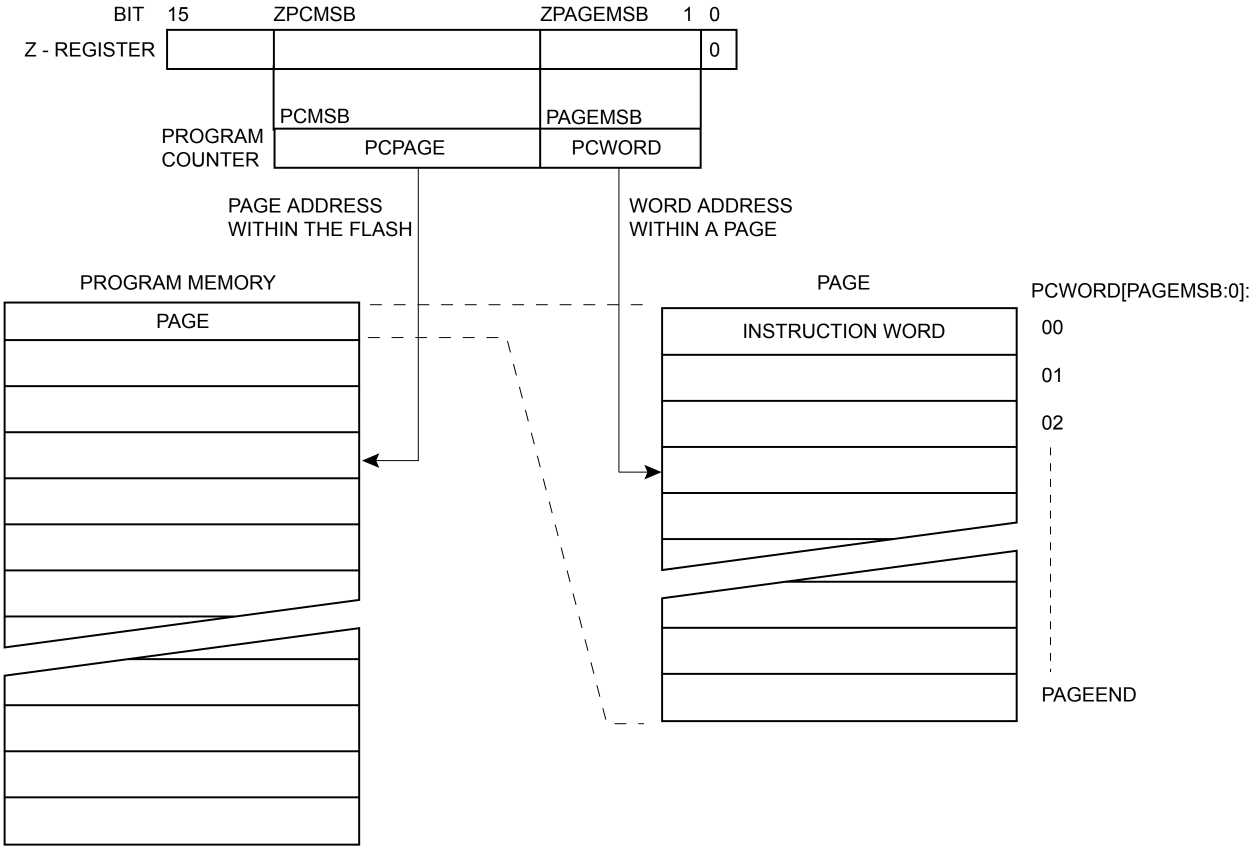
The Z-pointer is used to address the SPM commands. The Z pointer consists of the Z-registers ZL and ZH in the register file. The number of bits actually used is implementation dependent.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages, the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in the following figure. The Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 30-3. Addressing the Flash During SPM



Note: The different variables used in this figure are listed in the Related Links.

30.8 Self-Programming the Flash

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the Page Erase command or between a Page Erase and a Page Write operation:

Alternative 1, fill the buffer before a Page Erase

- Fill temporary page buffer
- Perform a Page Erase
- Perform a Page Write

Alternative 2, fill the buffer after Page Erase

- Perform a Page Erase
- Fill temporary page buffer
- Perform a Page Write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page. Please refer to Simple Assembly Code Example for a Boot Loader.

30.8.1 Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “0x0000011” to Store Program Memory Control and Status Register (SPMCSR) and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the Page Erase.
- Page Erase to the NRWW section: The CPU is halted during the operation.

30.8.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in [R1:R0], write “0x00000001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD ([Z5:Z1]) in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a Page Write operation or by writing the RWWSRE bit in SPMCSR (SPMCSR.RWWSRE). It is also erased after a system reset. It is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

30.8.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “0x0000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE ([Z5:Z1]). Other bits in the Z-pointer must be written to zero during this operation.

- Page Write to the RWW section: The NRWW section can be read during the Page Write
- Page Write to the NRWW section: The CPU is halted during the operation

30.8.4 Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCSR is cleared (SPMCSR.SPMEN). This means that the interrupt can be used instead of polling the SPMCSR Register in software. When using the SPM interrupt, the Interrupt Vectors should be moved to the Boot Loader Section (BLS) section to avoid that an interrupt is accessing the RWW section when it is blocked for reading. How to move the interrupts is described in *Interrupts* chapter.

Related Links

INT- Interrupts

30.8.5 Consideration While Updating Boot Loader Section (BLS)

Special care must be taken if the user allows the Boot Loader Section (BLS) to be updated by leaving Boot Lock bit11 unprogrammed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock bit11 to protect the Boot Loader software from any internal software changes.

30.8.6 Prevent Reading the RWW Section During Self-Programming

During Self-Programming (either Page Erase or Page Write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the self programming operation. The RWWSB in the SPMCSR (SPMCSR.RWWSB) will be set as long as the RWW section is busy. During Self-Programming the Interrupt Vector table should be moved to the BLS as described in *Watchdog Timer* chapter, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the SPMCSR.RWWSB by writing the SPMCSR.RWWSRE. Please refer to Simple Assembly Code Example for a Boot Loader for an example.

Related Links

Watchdog System Reset

30.8.7 Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock bits and general Lock Bits, write the desired data to R0, write “0x0001001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR.

Bit	7	6	5	4	3	2	1	0
R0	**1**	**1**	**BLB12**	**BLB11**	**BLB02**	**BLB01**	**LB2**	**LB1**

The tables in Boot Loader Lock Bits show how the different settings of the Boot Loader bits affect the Flash access.

If bits 5...0 in R0 are cleared (zero), the corresponding Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMEN are set in SPMCSR (SPMCSR.BLBSET and SPMCSR.SPMEN). The Z-pointer don’t care during this operation, but for future compatibility it is recommended to load the Z-pointer with 0x0001 (same as used for reading the IO$_{ck}$ bits). For future compatibility it is also recommended to set bits 7 and 6 in R0 to “1” when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

30.8.8 EEPROM Write Prevents Writing to SPMCSR

An EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR Register (EECR.EEPE) and verifies that the bit is cleared before writing to the SPMCSR Register.

30.8.9 Reading the Fuse and Lock Bits from Software

It is possible to read both the Fuse and Lock bits (LB) from software. To read the Lock bits, load the Z-pointer with 0x0001 and set the BLBSET and SPMEN bits in SPMCSR (SPMCSR.BLBSET and SPMCSR.SPMEN). When an LPM instruction is executed within three CPU cycles after the BLBSET and SPMEN bits are set in SPMCSR (SPMCSR.BLBSET and SPMCSR.SPMEN), the value of the Lock bits will be loaded in the destination register. The SPMCSR.BLBSET and SPMCSR.SPMEN will auto-clear upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When SPMCSR.BLBSET and SPMCSR.SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low byte (FLB) is similar to the one described above for reading the Lock bits. To read the Fuse Low byte, load the Z-pointer with 0x0000 and set the BLBSET and SPMEN bits in SPMCSR (SPMCSR.BLBSET and SPMCSR.SPMEN). When an LPM instruction is executed within three cycles after the SPMCSR.BLBSET and SPMCSR.SPMEN are set, the value of the Fuse Low byte (FLB) will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	**FLB7**	**FLB6**	**FLB5**	**FLB4**	**FLB3**	**FLB2**	**FLB1**	**FLB0**

Similarly, when reading the Fuse High byte (FHB), load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the SPMCSR.BLBSET and SPMCSR.SPMEN are set, the value of the Fuse High byte (FHB) will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

When reading the Extended Fuse byte (EFB), load 0x0002 in the Z-pointer. When an LPM instruction is executed within three cycles after the SPMCSR.BLBSET and SPMCSR.SPMEN are set, the value of the Extended Fuse byte (EFB) will be loaded in the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	EFB2	EFB1	EFB0

Fuse and Lock bits that are programmed read as '0'. Fuse and Lock bits that are unprogrammed, will read as '1'.

Related Links

Fuse Bits

30.8.10 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in the following table and set the SIGRD and SPMEN bits in SPMCSR (SPMCSR.SIGRD and SPMCSR.SPMEN). When an LPM instruction is executed within three CPU cycles after the SPMCSR.SIGRD and SPMCSR.SPMEN are set, the signature byte value will be loaded in the destination register. The SPMCSR.SIGRD and SPMCSR.SPMEN will auto-clear upon completion of reading the Signature Row Lock bits or if no LPM instruction is executed within three CPU cycles. When SPMCSR.SIGRD and SPMCSR.SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Related Links

Serial Number

30.8.11 Preventing Flash Corruption

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

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

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

1. If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
2. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
3. Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

30.8.12 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. The following table shows the typical programming time for Flash accesses from the CPU.

Table 30-5. SPM Programming Time

Symbol	Min. Programming Time	Max. Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7ms	4.5ms

Note: Minimum and maximum programming time is per individual operation.

30.8.13 Simple Assembly Code Example for a Boot Loader

```
;-the routine writes one page of data from RAM to Flash
; the first data location in RAM is pointed to by the Y pointer
; the first data location in Flash is pointed to by the Z-pointer
;-error handling is not included
 ;-the routine must be placed inside the Boot space
; (at least the Do_spm sub routine). Only code inside NRWW section can
; be read during Self-Programming (Page Erase and Page Write).
;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
; loophi (r25), spmcrval (r20)
; storing and restoring of registers is not included in the routine
; register usage can be optimized at the expense of code size
;-It is assumed that either the interrupt table is moved to the Boot
; loader section or that the interrupts are disabled.
.equ PAGESIZEB = PAGESIZE*2 ;PAGESIZEB is page size in BYTES, not words
.org SMALLBOOTSTART
Write_page:
      ; Page Erase
      ldi spmcrval, (1<<PGERS) | (1<<SPMEN)
      call Do_spm

      ; re-enable the RWW section
.
      ; must be avoided if the page buffer is pre-filled. Will flush the page
buffer.
      ldi spmcrval, (1<<RWWSRE) | (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 RWW section
     ldi spmcrval, (1<<RWWSRE) | (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, 1 ;use subi for PAGESIZEB<=256
     brne Rdloop

     ; return to RWW section
     ; verify that RWW section is safe to read
Return:
     in temp1, SPMCSR
     sbrs temp1, RWWSB ; If RWWSB is set, the RWW section is not ready yet
     ret
     ; re-enable the RWW section
```

```
     ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
     call Do_spm
     rjmp Return

Do_spm:
     ; check for previous SPM complete
Wait_spm:
     in temp1, SPMCSR
     sbrc temp1, SPMEN
     rjmp Wait_spm
     ; input: spmcrval determines SPM action
     ; disable interrupts if enabled, store status
     in temp2, SREG
     cli
     ; check that no EEPROM write access is present
Wait_ee:
     sbic EECR, EEPE
     rjmp Wait_ee
     ; SPM timed sequence
     out SPMCSR, spmcrval
     spm
     ; restore SREG (to enable interrupts if originally enabled)
     out SREG, temp2
     ret
```

30.8.14 Boot Loader Parameters

In the following tables, the parameters used in the description of the self programming are given.

Table 30-6. Boot Size Configuration

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	256 words	4	0x0000 - 0x3EFF	0x3F00 - 0x3FFF	0x3EFF	0x3F00
1	0	512 words	8	0x0000 - 0x3DFF	0x3E00 - 0x3FFF	0x3DFF	0x3E00

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
0	1	1024 words	16	0x0000 - 0x3BFF	0x3C00 - 0x3FFF	0x3BFF	0x3C00
0	0	2048 words	32	0x0000 - 0x37FF	0x3800 - 0x3FFF	0x37FF	0x3800

Note: The different BOOTSZ Fuse configurations are shown in Figure 30-2

Table 30-7. Read-While-Write Limit

Section	Pages	Address
Read-While-Write section (RWW)	224	0x0000 - 0x37FF
No Read-While-Write section (NRWW)	32	0x3800 - 0x3FFF

Note: For details about these two section, see NRWW – No Read-While-Write Section and RWW – Read-While-Write Section.

Table 30-8. Explanation of Different Variables used in Figure and the Mapping to the Z-pointer

Variable		Corresponding Variable [1]	Description
PCMSB	13		Most significant bit in the Program Counter. (The Program Counter is 14 bits PC[13:0])
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires 6 bits PC [5:0]).
ZPCMSB		Z14	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[13:6]	Z[14:7]	Program counter page address: Page select, for page erase and page write
PCWORD	PC[5:0]	Z[6:1]	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note:

1. Z[15]: always ignored. Z0: should be zero for all SPM commands, byte select for the LPM instruction.
   See Addressing the Flash During Self-Programming for details about the use of Z-pointer during Self- Programming.

30.9 Register Description

30.9.1 SPMCSR – Store Program Memory Control and Status Register

The Store Program Memory Control and Status Register contains the control bits needed to control the Boot Loader operations.

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPMCSR
Offset: 0x57 [ID-000004d0]
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x37

Bit	7	6	5	4	3	2	1	0
	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPMIE: SPM Interrupt Enable
When the SPMIE bit is written to one, and the I-bit in the Status Register is set (one), the SPM ready interrupt will be enabled. The SPM ready Interrupt will be executed as long as the SPMEN bit in the SPMCSR Register is cleared.

Bit 6 – RWWSB: Read-While-Write Section Busy
When a Self-Programming (Page Erase or Page Write) operation to the RWW section is initiated, the RWWSB will be set (one) by hardware. When the RWWSB bit is set, the RWW section cannot be accessed. The RWWSB bit will be cleared if the RWWSRE bit is written to one after a Self-Programming operation is completed. Alternatively the RWWSB bit will automatically be cleared if a page load operation is initiated.

Bit 5 – SIGRD: Signature Row Read
If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register. Please refer to *Reading the Fuse and Lock Bits from Software* in this chapter. An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect. This operation is reserved for future use and should not be used.

Bit 4 – RWWSRE: Read-While-Write Section Read Enable
When programming (Page Erase or Page Write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the Flash is busy with a Page Erase or a Page Write (SPMEN is set). If the RWWSRE bit is written while the Flash is being loaded, the Flash load operation will abort and the data loaded will be lost.

Bit 3 – BLBSET: Boot Lock Bit Set
If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets Boot Lock bits and Memory Lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the Lock bit set, or if no SPM instruction is executed within four clock cycles.

An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCSR Register (SPMCSR.BLBSET and SPMCSR.SPMEN), will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. Please refer to *Reading the Fuse and Lock Bits from Software* in this chapter.

Bit 2 – PGWRT: Page Write
If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Zpointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a Page Write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation if the NRWW section is addressed.

Bit 1 – PGERS: Page Erase
If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation if the NRWW section is addressed.

Bit 0 – SPMEN: Store Program Memory
This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During Page Erase and Page Write, the SPMEN bit remains high until the operation is completed.
Writing any other combination than “0x10001”, “0x01001”, “0x00101”, “0x00011” or “0x00001” in the lower five bits will have no effect.

31. MEMPROG- Memory Programming

31.1 Program And Data Memory Lock Bits

The devices provides six Lock bits. These can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in Table. Lock Bit Protection Modes in this section. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 31-1. Lock Bit Byte[(1)]

Lock Bit Byte	Bit No.	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note:

1. '1' means unprogrammed, '0' means programmed.

Table 31-2. Lock Bit Protection Modes(1)(2)

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.(1)
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Boot Lock bits and Fuse bits are locked in both Serial and Parallel Programming mode.(1)

Note:

1. Program the Fuse bits and Boot Lock bits before programming the LB1 and LB2.
2. '1' means unprogrammed, '0' means programmed.

Table 31-3. Lock Bit Protection - BLB0 Mode[(1)][(2)].

BLB0 Mode	BLB02	BLB01	
1	1	1	No restrictions for SPM or Load Program Memory (LPM) instruction 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.

Table 31-4. Lock Bit Protection - BLB1 Mode[(1)][(2)]

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 and Boot Lock bits before programming the LB1 and LB2.
2. '1' means unprogrammed; '0' means programmed.

31.2 Fuse Bits

The device has three Fuse bytes. The following tables describe briefly the functionality of all the fuses and how they are mapped into the Fuse bytes. Note that the fuses are read as logical zero, '0', if they are programmed.

Table 31-5. Extended Fuse Byte (0x0f)

Extended Fuse Byte	Bit No.	Description	Default Value
–	7	–	0
–	6	–	0

Extended Fuse Byte	Bit No.	Description	Default Value
–	5	–	0
–	4	–	0
CFD	3	Disable Clock Failure Detection	0 (programmed, CFD disable)
BODLEVEL2(1)	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1(1)	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0(1)	0	Brown-out Detector trigger level	1 (unprogrammed)

Note: 1. Please refer to Table. BODLEVEL Fuse Coding in *System and Reset Characteristics* for BODLEVEL Fuse decoding.

Table 31-6. Fuse High Byte. (0x99)

High Fuse Byte	Bit No.	Description	Default Value
OCDEN(1)	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN(2)	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON(3)	4	Watchdog Timer Always On	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed), EEPROM not reserved
BOOTSZ1(4)	2	Select Boot Size	0 (programmed)
BOOTSZ0(4)	1	Select Boot Size	0 (programmed)
BOOTRST	0	Boot Reset vector Enabled	1 (unprogrammed)

Note:

1. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. Please refer to *WDTCSR – Watchdog Timer Control Register* for details.
4. The default value of BOOTSZ[1:0] results in maximum Boot Size. See Boot size configuration table for details.

Table 31-7. Fuse Low Byte (0x62)

Low Fuse Byte	Bit No.	Description	Default Value
CKDIV8(4)	7	Divide clock by 8	0 (programmed)
CKOUT(3)	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)

Low Fuse Byte	Bit No.	Description	Default Value
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	1 (unprogrammed)(2)
CKSEL0	0	Select Clock source	0 (programmed)(2)

Note:

1. The default value of SUT[1:0] results in maximum start-up time for the default clock source. See Table. Start-up times for the internal calibrated RC Oscillator clock selection in *Calibrated Internal RC Oscillator* of System Clock and Clock Options chapter for details.
2. The default setting of CKSEL[3:0] results in internal RC Oscillator @ 8MHz. See Table 'Internal Calibrated RC Oscillator Operating Modes' in *Calibrated Internal RC Oscillator* of the System Clock and Clock Options chapter for details.
3. The CKOUT Fuse allows the system clock to be output on PORTB0. Please refer to *Clock Output Buffer* section in the System Clock and Clock Options chapter for details.
4. Please refer to *System Clock Prescaler* section in the System Clock and Clock Options chapter for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Related Links

Alternate Port Functions

Calibrated Internal RC Oscillator

WDTCSR

System and Reset Characteristics

31.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

31.3 Signature Bytes

All microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space. For the device the signature bytes are given in the following table.

Table 31-8. Device and JTAG ID

Part	Signature Bytes Address			JTAG	
	0x000	0x001	0x002	Part number	Manufacture ID
ATmega324PB	0x1E	0x95	0x17	9511	0x1F

31.4 Calibration Byte

The device has a byte calibration value for the Internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

31.5 Serial Number

The product has serial number which offer a unique ID to identify a specify part while it is in the field. It consists of several bytes which can be accessed from the signature address space.

Signature row includes factory-programmed data:

- ID for each device type
- Serial number for each device
- Calibration bytes for factory calibrated peripherals

31.5.1 Signature Row Summary - SIGROW

Offset	Name	Bit Pos.								
0x00	SIGROW_DEVICEI D0	7:0	DEVICEID[7:0]							
0x01	SIGROW_RCOC	7:0	RCOC[7:0]							
0x02	SIGROW_DEVICEI D1	7:0	DEVICEID[7:0]							
0x03	Reserved									
0x04	SIGROW_DEVICEI D2	7:0	DEVICEID[7:0]							
0x05 ... 0x0D	Reserved									
0x0E	SIGROW_SERNUM 0	7:0	SERNUM[7:0]							
0x0F	SIGROW_SERNUM 1	7:0	SERNUM[7:0]							
0x10	SIGROW_SERNUM 2	7:0	SERNUM[7:0]							
0x11	SIGROW_SERNUM 3	7:0	SERNUM[7:0]							
0x12	SIGROW_SERNUM 4	7:0	SERNUM[7:0]							
0x13	SIGROW_SERNUM 5	7:0	SERNUM[7:0]							
0x14	SIGROW_SERNUM 6	7:0	SERNUM[7:0]							
0x15	SIGROW_SERNUM 7	7:0	SERNUM[7:0]							
0x16	SIGROW_SERNUM 8	7:0	SERNUM[7:0]							
0x17	SIGROW_SERNUM 9	7:0	SERNUM[7:0]							

31.5.1.1 Device ID n

Name: SIGROW_DEVICEIDn

Offset: 0x00 + n*0x02 [n=0..2]

Reset: [Device ID]

Property: -

Bit	7	6	5	4	3	2	1	0
Register	DEVICEID[7:0]							

Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVICEID[7:0]: Byte n of the Device ID

31.5.1.2 RC Oscillator Calibration Byte

This signature row location is loaded to OSCCAL-register during start-up.

Name: SIGROW_RCOC

Offset: 0x01

Reset: [RC oscillator calibration]

Property: -

Bit	7	6	5	4	3	2	1	0
Register	RCOC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – RCOC[7:0]: RC Oscillator Calibration Byte

31.5.1.3 Serial Number Byte n

Each device has an individual serial number, representing a unique ID. This can be used to identify a specific device in the field. The serial number consists of ten bytes: SIGROW.SERNUM[9:0].

Name: SIGROW_SERNUMn

Offset: 0x0E + n*0x01 [n=0..9]

Reset: [device serial number]

Property: -

Bit	7	6	5	4	3	2	1	0
Register	SERNUM[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SERNUM[7:0]: Serial Number

Each device has an individual serial number, representing a unique ID. This can be used to identify a specific device in the field. The serial number consists of ten bytes.

31.5.1.4 Wafer Number

Name: SIGROW_WAFERNUM

Offset: 0x14

Reset: [wafer number]

Property: -

Bit	7	6	5	4	3	2	1	0
Register	WAFERNUM[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SERNUM[7:0]: WAFERNUM[7:0]: Wafer Number

31.5.1.5 X Coordinate

Name: SIGROW_XCOORDINATE

Offset: 0x16

Reset: [X coordinate]

Property: -

Bit	7	6	5	4	3	2	1	0
Register	XCOORDINATE[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – XCOORDINATE[7:0]: Device X Coordinate

31.5.1.6 Y Coordinate

Name: SIGROW_YCOORDINATE

Offset: 0x17

Reset: [Y coordinate]

Property: -

Bit	7	6	5	4	3	2	1	0
Register	YCOORDINATE[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – YCOORDINATE[7:0]: Device Y Coordinate

31.6 Page Size

Table 31-9. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega324PB	16K words (32Kbytes)	64 words	PC[5:0]	256	PC[13:6]	13

Table 31-10. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATmega324PB	1Kbytes	4bytes	EEA[1:0]	256	EEA[9:2]	9

31.7 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the device. Pulses are assumed to be at least 250ns unless otherwise noted.

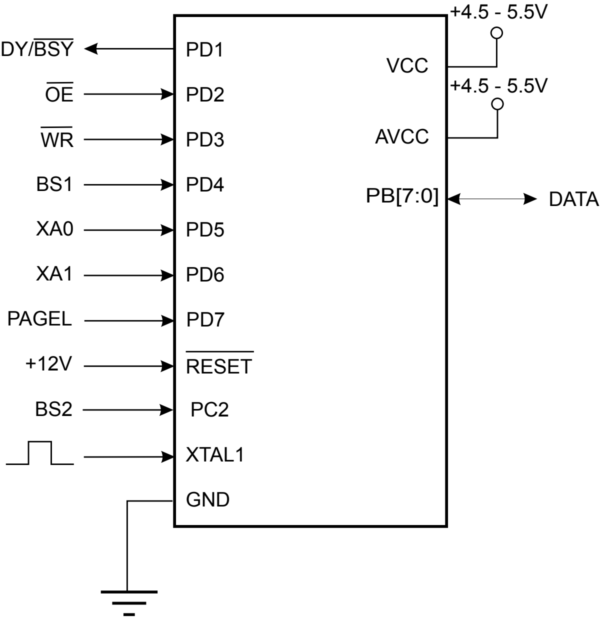
31.7.1 Signal Names

In this section, some pins of this device are referenced by signal names describing their functionality during parallel programming, please refer to Figure. Parallel Programming and Table. Pin Name Mapping in this section. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in the table, XA1 and XA0 Coding.

When pulsing $\overline{\text{WR}}$ or $\overline{\text{OE}}$, the command loaded determines the action executed. The different Commands are shown in the table, Command Byte Bit Coding Command Byte Command Executed.

Figure 31-1. Parallel Programming



Note: V_{CC} - 0.3V < AV_{CC} < V_{CC} + 0.3V, however, AV_{CC} should always be within 4.5 - 5.5V

Table 31-11. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{\text{BSY}}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{\text{OE}}$	PD2	I	Output Enable (Active low)

Signal Name in Programming Mode	Pin Name	I/O	Function
$\overline{\text{WR}}$	PD3	I	Write Pulse (Active low)
BS1	PD4	I	Byte Select 1 (“0” selects Low byte, “1” selects High byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
PAGEL	PD7	I	Program memory and EEPROM Data Page Load
BS2	PC2	I	Byte Select 2 (“0” selects Low byte, “1” selects 2’nd High byte)
DATA	PB[7:0]	I/O	Bi-directional Data bus (Output when OE is low)

Table 31-12. BS2 and BS1 encoding.

BS2	BS1	Flash / EEPROM address	Flash data loading / reading	Fuse programming	Reading fuse and lock bits
0	0	Low Byte	Low Byte	Low Byte	Fuse Low Byte
0	1	High Byte	High Byte	High Byte	Lockbits
1	0	Extended High Byte	Reserved	Extended Byte	Extended Fuse Byte
1	1	Reserved	Reserved	Reserved	Fuse High Byte

Table 31-13. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 31-14. 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 31-15. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits

Command Byte	Command Executed
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature Bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

31.8 Parallel Programming

31.8.1 Entering Programming Mode

Follow the steps below to put the device in Parallel (High-voltage) Programming mode:

1. Set the Prog_enable pins listed in the table *Pin Values Used to Enter Programming Mode* above to “0x0000”, $\overline{\text{RESET}}$ pin to 0V and V_{CC} to 0V.
2. Apply 4.5–5.5V between VCC and GND.
   Ensure that V_{CC} reaches at least 1.8V within the next 20μs.
3. Wait 20–60μs, and apply 11.5–12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10μs after the high voltage has been applied to ensure the Prog_enable signature has been latched.
5. Wait at least 300μs before giving any parallel programming commands.
6. Exit Programming mode by powering down the device or by bringing $\overline{\text{RESET}}$ pin to 0V.

If the rise time of V_{CC} is unable to fulfill the requirements listed above, the following alternative method can be used to put the device in Parallel (High-voltage) Programming mode:

1. Set the Prog_enable pins listed in the table *Pin Values Used to Enter Programming Mode* above to “0000”, $\overline{\text{RESET}}$ pin to 0V and V_{CC} to 0V.
2. Apply 4.5–5.5V between VCC and GND.
3. Monitor V_{CC}, and as soon as V_{CC} reaches 0.9–1.1V, apply 11.5–12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10μs after the high voltage has been applied to ensure the Prog_enable signature has been latched.
5. Wait until V_{CC} reaches 4.5–5.5V before giving any parallel programming commands.
6. Exit Programming mode by powering down the device or by bringing $\overline{\text{RESET}}$ pin to 0V.

31.8.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256byte EEPROM. This consideration also applies to Signature bytes reading.

31.8.3 Chip Erase

The Chip Erase will erase the Flash, the SRAM and the EEPROM memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

Note: The EEPROM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”:

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{\text{WR}}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{\text{BSY}}$ goes low.
6. Wait until RDY/$\overline{\text{BSY}}$ goes high before loading a new command.

31.8.4 Programming the Flash

The Flash is organized in pages as number of Words in a Page and number of Pages in the Flash. 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:

Step 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.

Step B. Load Address Low Byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

Step C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

Step D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

Step E. Latch Data

1. Set BS1 to “1”. This selects high data byte.

2. Give PAGEL a positive pulse. This latches the data bytes. (Please refer to the figure, Programming the Flash Waveforms, in this section for signal waveforms)

Step F. Repeat B Through E Until the Entire Buffer Is Filled or Until All Data Within the Page Is Loaded

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in the following figure, Addressing the Flash Which is Organized in Pages, in this section. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a Page Write.

Step G. Load Address High Byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

Step H. Program Page

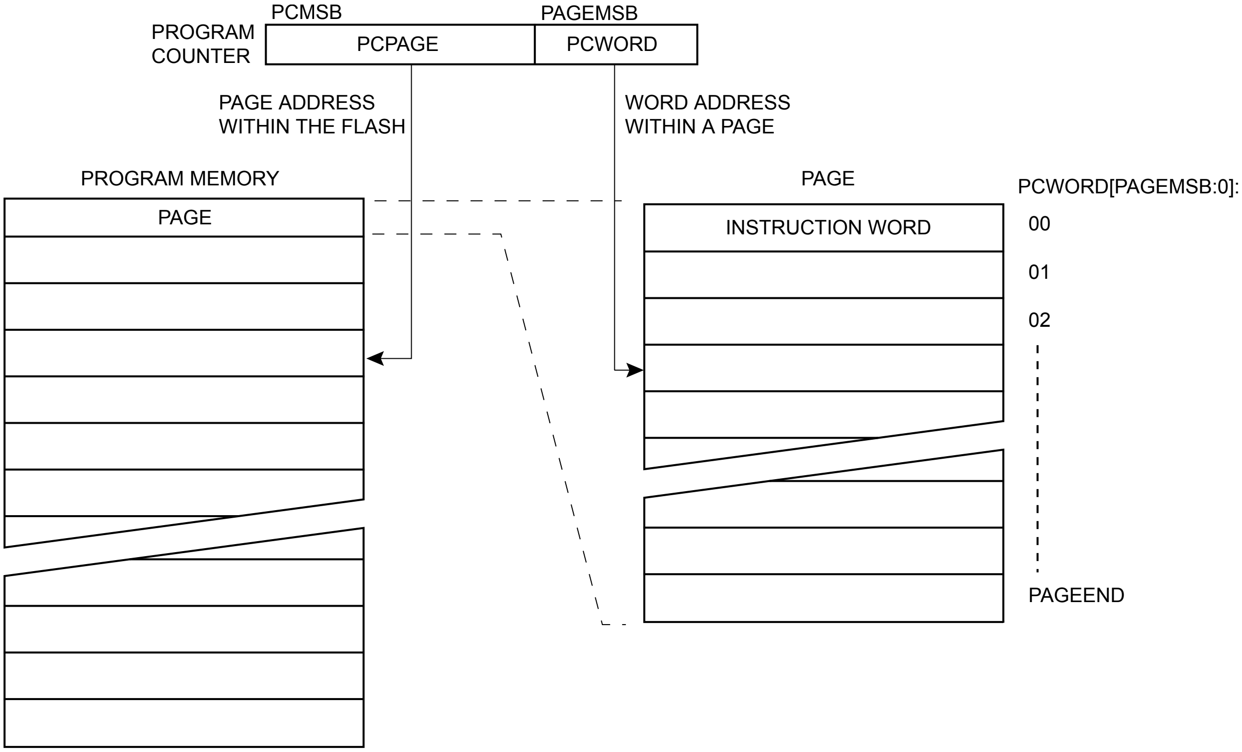
1. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{\text{BSY}}$ goes low.
2. Wait until RDY/$\overline{\text{BSY}}$ goes high (Please refer to the figure, Programming the Flash Waveforms, in this section for signal waveforms).

Step I. Repeat B Through H Until the Entire Flash Is Programmed or Until All Data Has Been Programmed

Step J. End Page Programming

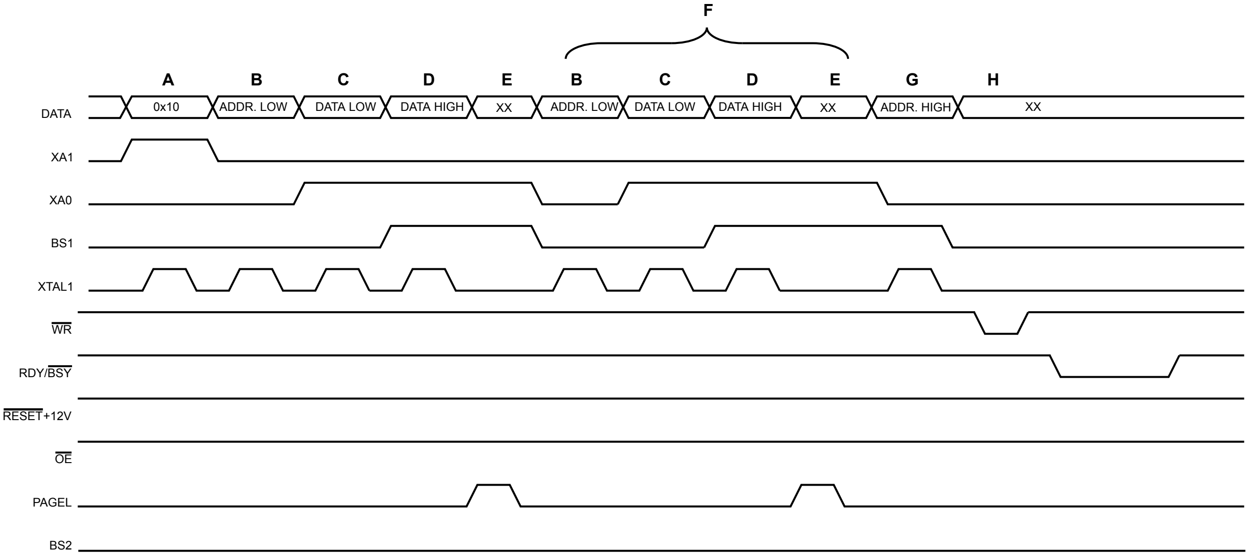
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 31-2. Addressing the Flash Which Is Organized in Pages



Note: PCPAGE and PCWORD are listed in the table of *No. of Words in a Page and No. of Pages* in the Flash in *Page Size* section.

Programming the Flash Waveforms



Note: “XX” is don’t care. The letters refer to the programming description above.

Related Links

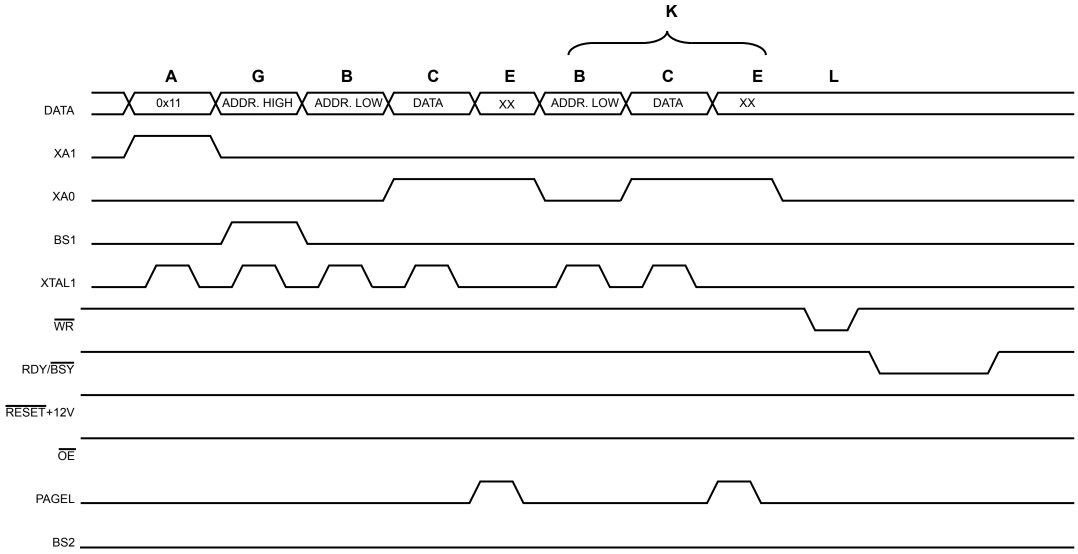
Page Size

31.8.5 Programming the EEPROM

The EEPROM is organized in pages, please refer to table, No. of Words in a Page and No. of Pages in the EEPROM, in the Page Size section. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (For details on Command, Address and Data loading, please refer to Programming the Flash):

1. Step A: Load Command "0001 0001".
2. Step G: Load Address High Byte (0x00 - 0xFF).
3. Step B: Load Address Low Byte (0x00 - 0xFF).
4. Step C: Load Data (0x00 - 0xFF).
5. Step E: Latch data (give PAGEL a positive pulse).
6. Step K:Repeat 3 through 5 until the entire buffer is filled.
7. Step L: Program EEPROM page
    - 7.1. Set BS1 to "0".
    - 7.2. Give WR a negative pulse. This starts programming of the EEPROM page. RDY/BSY goes low.
    - 7.3. Wait until to RDY/BSY goes high before programming the next page (Please refer to the following figure for signal waveforms).

Figure 31-3. Programming the EEPROM Waveforms



31.8.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (Please refer to Programming the Flash in this chapter for details on Command and Address loading):

1. Step A: Load Command "0000 0010".
2. Step G: Load Address High Byte (0x00 - 0xFF).
3. Step B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to "0", and BS1 to "0". The Flash word low byte can now be read at DATA.
5. Set BS1 to "1". The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to "1".

31.8.7 Reading the EEPROM

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

1. Step A: Load Command “0000 0011”.
2. Step G: Load Address High Byte (0x00 - 0xFF).
3. Step B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{\text{OE}}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{\text{OE}}$ to “1”.

31.8.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

1. Step A: Load Command “0100 0000”.
2. Step C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

31.8.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

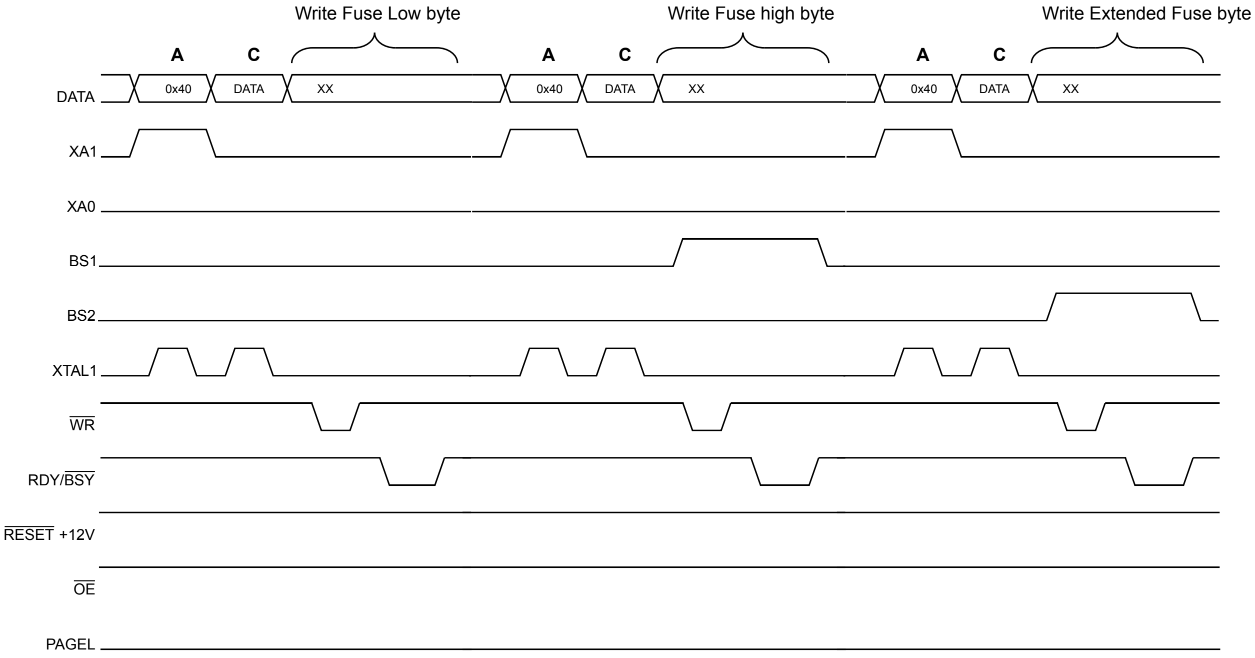
1. Step A: Load Command “0100 0000”.
2. Step C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give WR a negative pulse and wait for RDY/BSY to go high.
5. Set BS1 to “0”. This selects low data byte.

31.8.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

1. Step A: Load Command “0100 0000”.
2. Step C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.
5. Set BS2 to “0”. This selects low data byte.

Figure 31-4. Programming the FUSES Waveforms



31.8.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

1. Step A: Load Command “0010 0000”.
2. Step C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

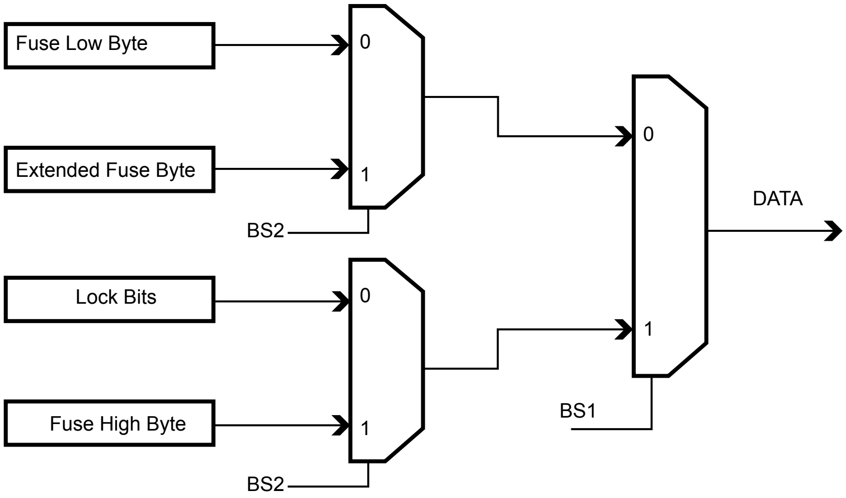
The Lock bits can only be cleared by executing Chip Erase.

31.8.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (Please refer to Programming the Flash for details on Command loading):

1. Step A: Load Command “0000 0100”.
2. Set $\overline{\text{OE}}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{\text{OE}}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set $\overline{\text{OE}}$ to “0”, BS2 to “1”, and BS1 to “0”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{\text{OE}}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{\text{OE}}$ to “1”.

Figure 31-5. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



31.8.13 Reading the Signature Bytes

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

1. Step A: Load Command “0000 1000”.
2. Step B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{\text{OE}}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{\text{OE}}$ to “1”.

31.8.14 Reading the Calibration Byte

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

1. Step A: Load Command “0000 1000”.
2. Step B: Load Address Low Byte, 0x00.
3. Set $\overline{\text{OE}}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{\text{OE}}$ to “1”.

31.8.15 Parallel Programming Characteristics

For characteristics of the Parallel Programming, please refer to *Parallel Programming Characteristics*.

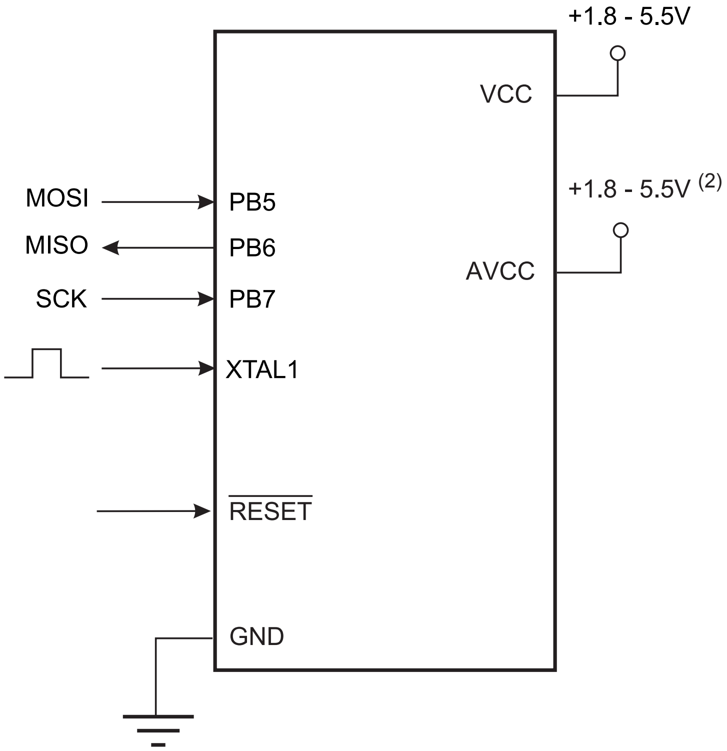
Related Links

Parallel Programming Characteristics

31.9 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Figure 31-6. Serial Programming and Verify



Note:

1. If the device is clocked by the internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.
2. V_{CC} - 0.3V < AVCC < V_{CC} + 0.3V, however, AVCC should always be within 1.8–5.5V .

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

- Low: > 2 CPU clock cycles for f_{ck} < 12MHz, , 3 CPU clock cycles for f_{ck} ≥ 12MHz
- High: > 2 CPU clock cycles for f_{ck} < 12MHz, 3 CPU clock cycles for f_{ck} ≥ 12MHz

31.9.1 Serial Programming Pin Mapping

Table 31-16. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB5	I	Serial Data in
MISO	PB6	O	Serial Data out
SCK	PB7	I	Serial Clock

Note: The pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

31.9.2 Serial Programming Algorithm

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

When reading data from the device, data is clocked on the falling edge of SCK. Please refer to the figure, Serial Programming Waveforms in SPI Serial Programming Characteristics section for timing details.

To program and verify the device in the serial programming mode, the following sequence is recommended (See Serial Programming Instruction set in Table 31-18:

1. Power-up sequence:
   Apply power between VCC and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, RESET must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load Program Memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program Memory Page is stored by loading the Write Program Memory Page instruction with the 7 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page . Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. A: The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The Memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load EEPROM Memory Page instruction. The EEPROM Memory Page is stored by loading the Write EEPROM Memory Page Instruction with the 7 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM Memory Page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next byte. In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.

   Turn V_{CC} power off.

Table 31-17. Typical Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	2.6ms
t_{WD_EEPROM}	3.6ms
t_{WD_ERASE}	10.5ms
t_{WD_FUSE}	4.5ms

31.9.3 Serial Programming Instruction Set

This section describes the Instruction Set.

Table 31-18. Serial Programming Instruction Set (Hexadecimal values)

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte 4
Programming Enable	0xAC	0x53	0x00	0x00
Chip Erase (Program Memory/EEPROM)	0xAC	0x80	0x00	0x00
Poll RDY/$\overline{\text{BSY}}$	0xF0	0x00	0x00	data byte out
Load Instructions				
Load Extended Address byte(1)	0x4D	0x00	Extended adr	0x00
Load Program Memory Page, High byte	0x48	0x00	adr LSB	high data byte in
Load Program Memory Page, Low byte	0x40	0x00	adr LSB	low data byte in
Load EEPROM Memory Page (page access)	0xC1	0x00	0000 000aa(2)	data byte in
Read Instructions(5)				
Read Program Memory, High byte	0x28	adr MSB	adr LSB	high data byte out
Read Program Memory, Low byte	0x20	adr MSB	adr LSB	low data byte out
Read EEPROM Memory	0xA0	0000 00aa(2)	aaaa aaaa(2)	data byte out
Read Lock bits(3)	0x58	0x00	0x00	data byte out
Read Signature Byte	0x30	0x00	0000 000aa(2)	data byte out
Read Fuse bits(3)	0x50	0x00	0x00	data byte out
Read Fuse High bits(3)	0x58	0x08	0x00	data byte out
Read Extended Fuse Bits(3)	0x50	0x08	0x00	data byte out
Read Calibration Byte	0x38	0x00	0x00	data byte out
Write Instructions(5)				
Write Program Memory Page(6)	0x4C	adr MSB(8)	adr LSB(8)	0x00
Write EEPROM Memory	0xC0	0000 00aa(2)	aaaa aaaa(2)	data byte in
Write EEPROM Memory Page (page access)	0xC2	0000 00aa(2)	aaaa aa00(2)	0x00

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte 4
Write Lock bits(3)(4)	0xAC	0xE0	0x00	data byte in
Write Fuse bits(3)(4)	0xAC	0xA0	0x00	data byte in
Write Fuse High bits(3)(4)	0xAC	0xA8	0x00	data byte in
Write Extended Fuse Bits(3)(4)	0xAC	0xA4	0x00	data byte in

Note:

1. Not all instructions are applicable for all parts.
2. a = address.
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’) .
5. Refer to the corresponding section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
6. Instructions accessing program memory use a word address. This address may be random within the page range.

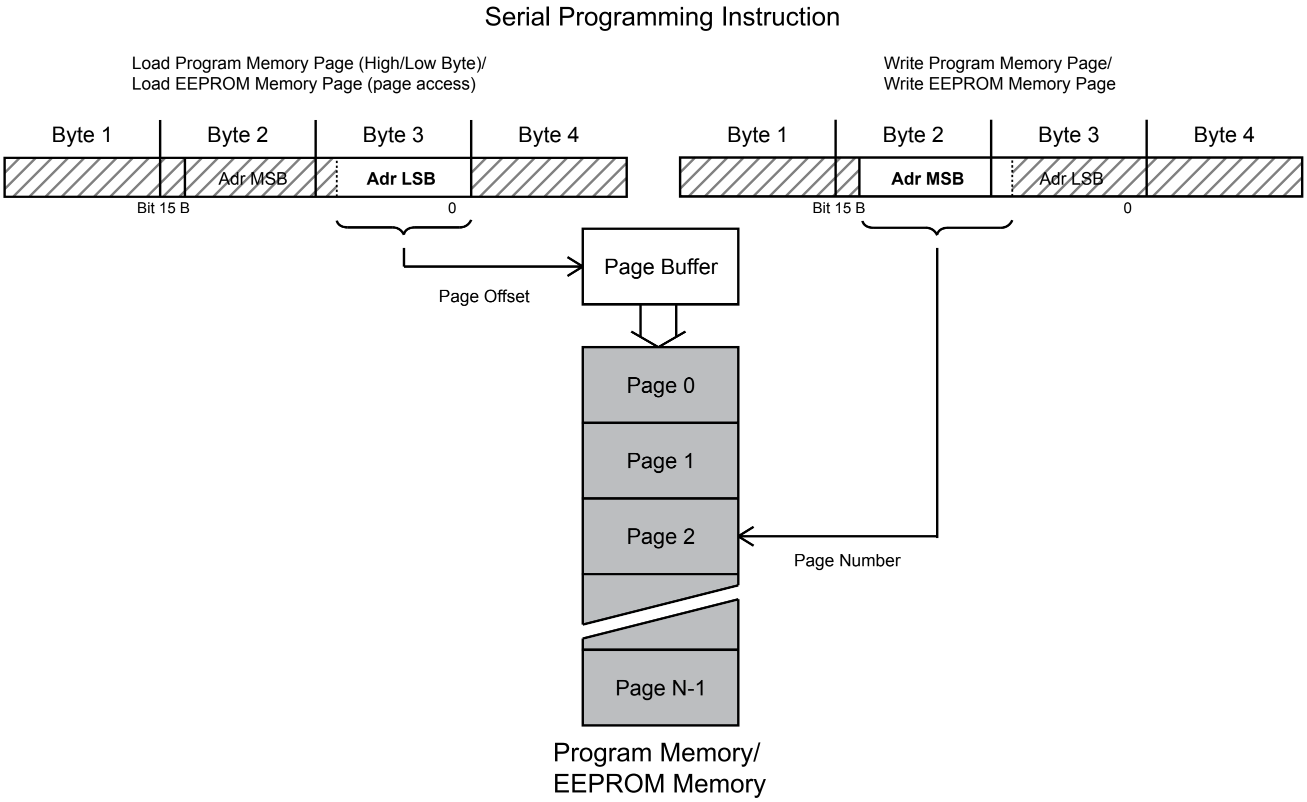
Note: See http://www.microchip.com/design-centers/8-bit/microchip-avr-mcus for Application Notes regarding programming and programmers.

If the LSB in RDY/$\overline{\text{BSY}}$ data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

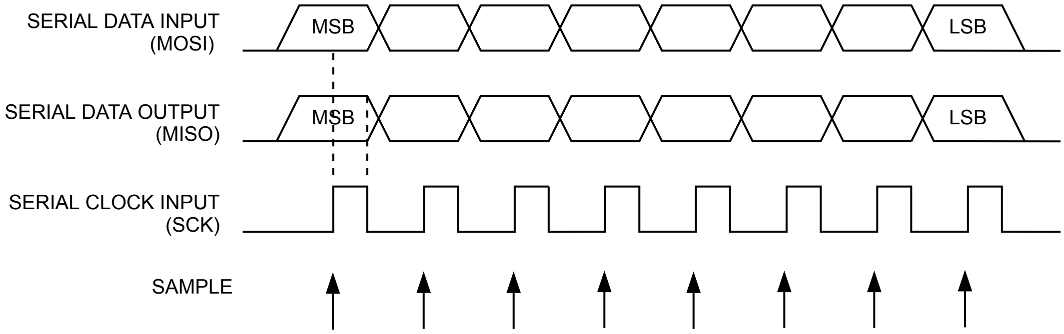
After data is loaded to the page buffer, program the EEPROM page, Please refer to the following figure.

Figure 31-7. Serial Programming Instruction example



31.9.4 SPI Serial Programming Characteristics

Figure 31-8. Serial Programming Waveforms



31.10 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.

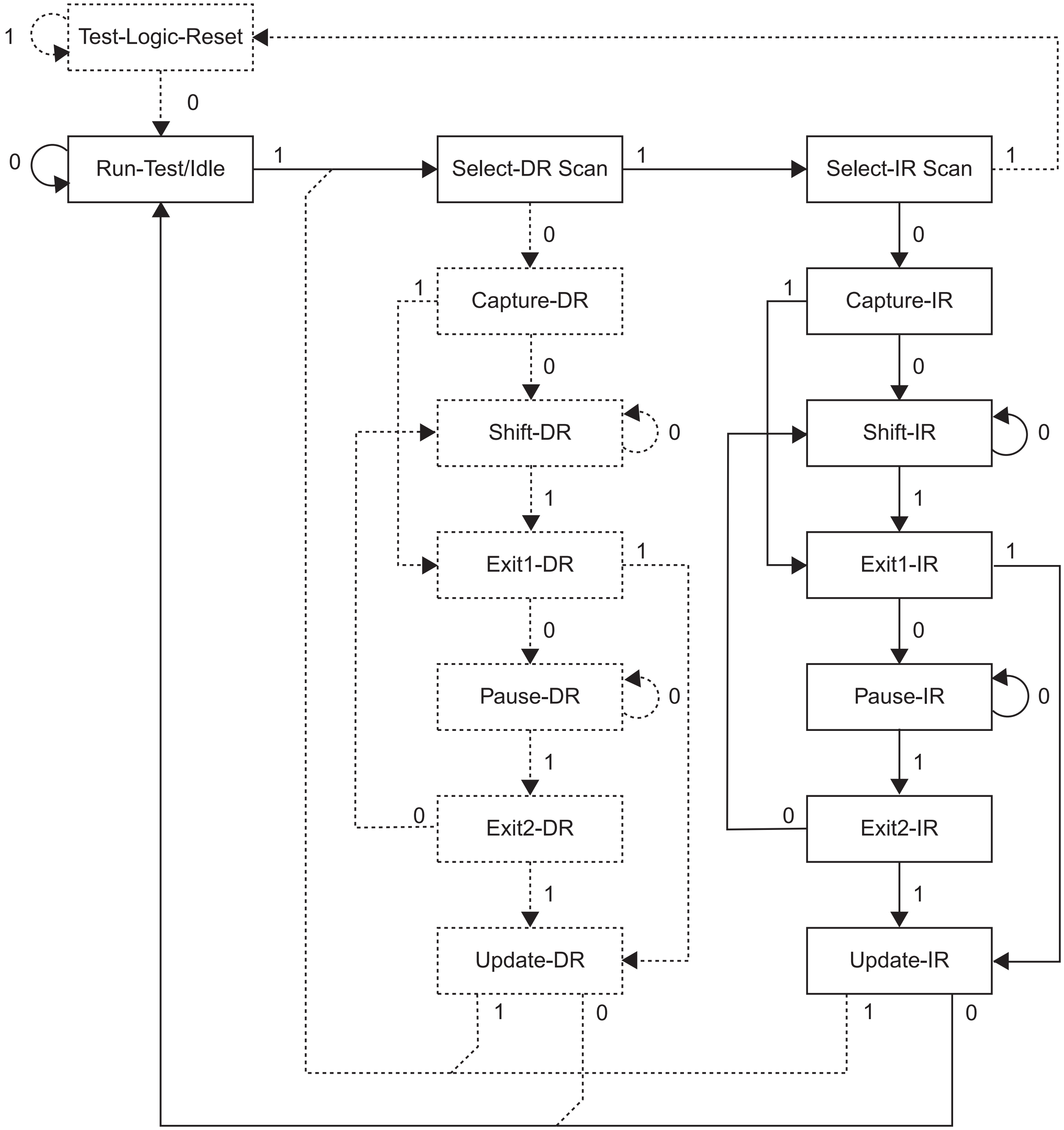
31.10.1 Programming Specific JTAG Instructions

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. The state machine sequence for changing the instruction word is shown in the figure below.

Figure 31-9. State Machine Sequence for Changing the Instruction Word



31.10.2 AVR_RESET (0xC)

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.

31.10.3 PROG_ENABLE (0x4)

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.

31.10.4 PROG_COMMANDS (0x5)

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.

31.10.5 PROG_PAGELOAD (0x6)

The AVR specific public JTAG instruction to directly load the Flash data page via the JTAG port. The 2048-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.

Note: 1. The JTAG instruction PROG_PAGELOAD can only be used if the AVR device is the first device in JTAG scan chain. If the AVR cannot be the first device in the scan chain, the byte-wise programming algorithm must be used.

31.10.6 PROG_PAGEREAD (0x7)

The AVR specific public JTAG instruction to read one full Flash data page via the JTAG port. The 2056-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.

Note: 1. The JTAG instruction PROG_PAGEREAD can only be used if the AVR device is the first device in JTAG scan chain. If the AVR cannot be the first device in the scan chain, the byte-wise programming algorithm must be used.

31.10.7 Data Registers

The data registers are selected by the JTAG instruction registers described in section Programming Specific JTAG Instructions. 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

31.10.8 Reset Register

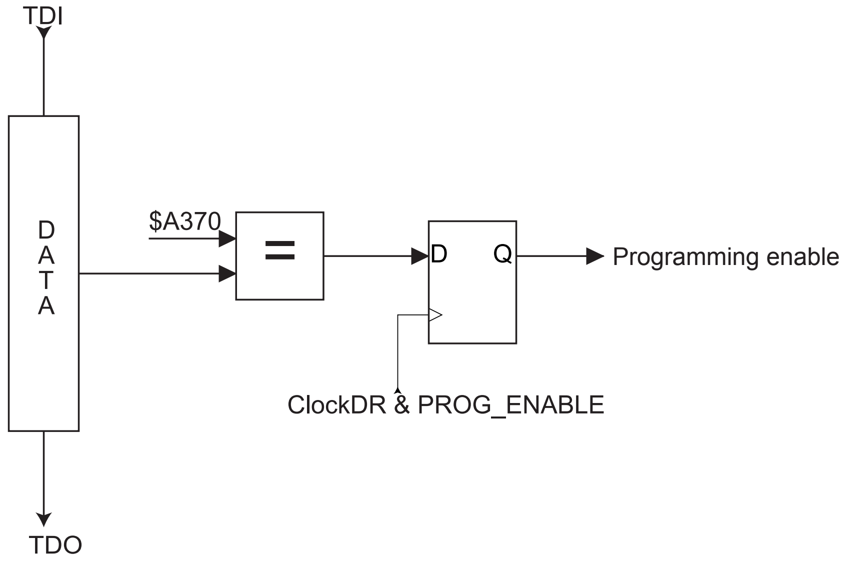
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering programming mode.

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 *Clock Sources*) 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 *Reset Register*.

31.10.9 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 31-10. Programming Enable Register



31.10.10 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 the following table. The state sequence when shifting in the programming commands is illustrated in State Machine Sequence for Changing/Reading the Data Word further down in this section.

Figure 31-11. Programming Command Register

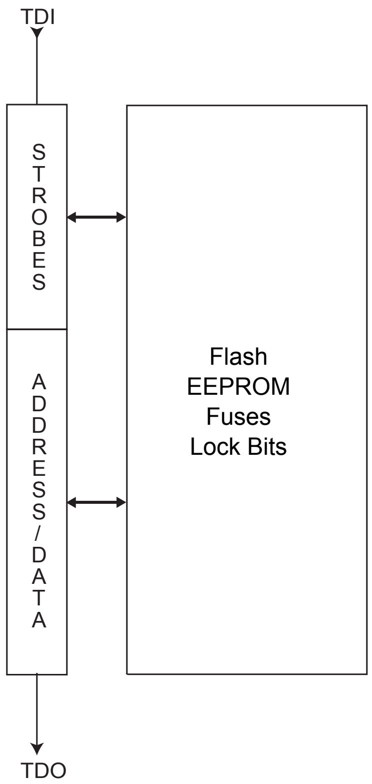


Table 31-19. JTAG Programming Instruction Set

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

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx 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_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
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 Flash Page	0110111_00000000 0110101_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)

Instruction	TDI sequence	TDO sequence	Notes
	0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
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_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
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_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
4c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Poll for Page Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
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 Low Byte(6)	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6c. Write Fuse Extended byte	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)

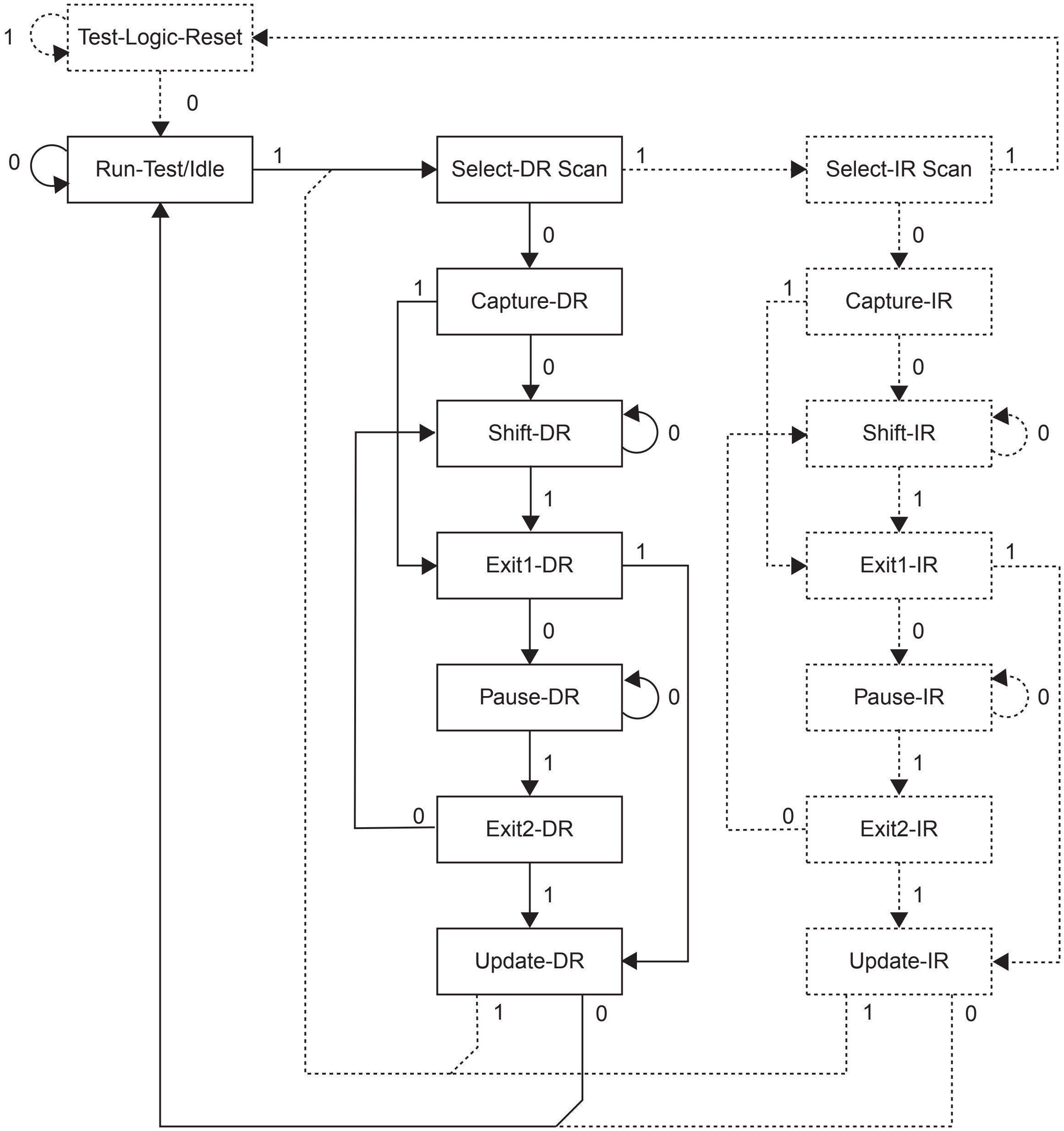
Instruction	TDI sequence	TDO sequence	Notes
6d. Poll for Fuse Write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
6e. Load Data Low Byte(7)	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6f. Write Fuse High byte	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6g. Poll for Fuse Write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
6h. Load Data Low Byte(7)	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6i. Write Fuse Low byte	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6j. Poll for Fuse Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
7a. Enter Lock bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte(9)	0010011_11iiiiii	xxxxxxx_xxxxxxxx	(4)
7c. Write Lock bits	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx 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 Extended Fuse Byte(6)	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8c. Read Fuse High Byte(7)	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8d. Read Fuse Low Byte(8)	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8e. Read Lock bits(9)	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxoooooo	(5)
8f. Read Fuses and Lock bits	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo	(5) fuse ext. byte fuse high byte fuse low byte lock bits

Instruction	TDI sequence	TDO sequence	Notes
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	

Note:

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. The bit mapping for Fuses Extended byte is listed in Extended Fuse Byte table of Fuse Bits section.
7. The bit mapping for Fuses High byte is listed in Fuse High Byte table of Fuse Bits section.
8. The bit mapping for Fuses Low byte is listed in Fuse Low Byte table of Fuse Bits section.
9. The bit mapping for Lock bits byte is listed in Lock Bit Byte table of Program and Data Memory Lock Bits section.
10. Address bits exceeding PCMSB and EEAMSB (Command Byte Bit Coding in Signal Names section and Page Size section) are don’t care

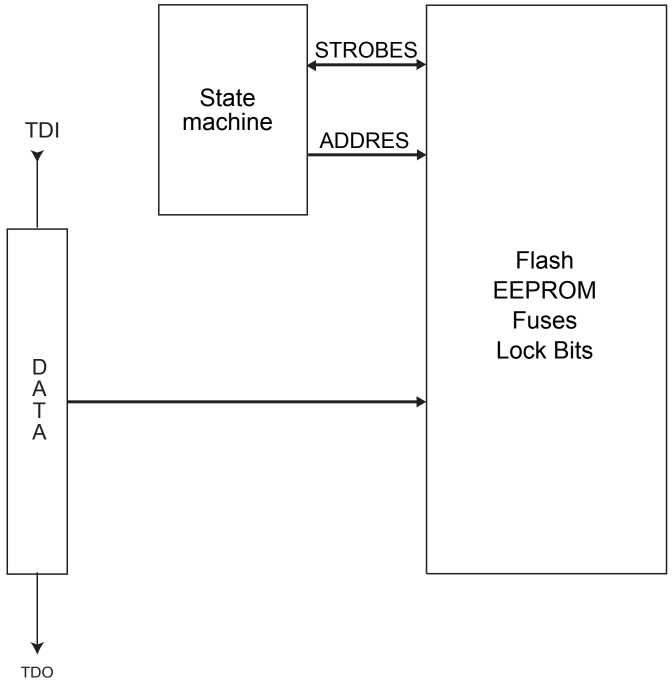
Figure 31-12. State Machine Sequence for Changing/Reading the Data Word



31.10.11 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. 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 first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to load the entire Flash page buffer before executing Page Write.

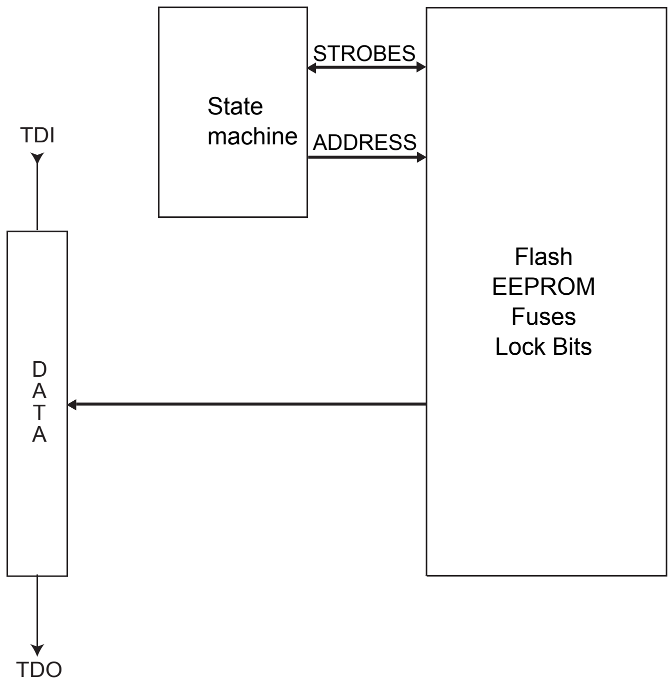
Figure 31-13. Virtual Flash Page Load Register



31.10.12 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. Internally the Shift Register is 8-bit, and the data are automatically transferred from the Flash data page byte by byte. The first eight 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 first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to read one full Flash page to verify programming.

Figure 31-14. Virtual Flash Page Read Register



31.10.13 Programming Algorithm

All references below of type “1a”, “1b”, and so on, refer to Table 31-19.

31.10.14 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.

31.10.15 Leaving Programming Mode

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 JTAG instruction AVR_RESET and shift 0 in the Reset Register.

31.10.16 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 *Command Byte Bit Coding* in section *Parallel Programming Parameters, Pin Mapping, and Commands*).

31.10.17 Programming the Flash

Before programming the Flash a Chip Erase must be performed. See Performing Chip Erase.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.

3. Load address high byte using programming instruction 2b.
4. Load address low byte using programming instruction 2c.
5. Load data using programming instructions 2d, 2e and 2f.
6. Repeat steps 4 and 5 for all instruction words in the page.
7. Write the page using programming instruction 2g.
8. Poll for Flash write complete using programming instruction 2h, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
9. 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. PCWORD (refer to Command Byte Bit Coding table in Signal Names section) is 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} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
9. Repeat steps 3 to 8 until all data have been programmed.

31.10.18 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. PCWORD (refer to table *Command Byte Bit Coding* in section *Parallel Programming Parameters, Pin Mapping, and Commands*) is 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 first instruction in the page and ending with the MSB of the last instruction in the page. Remember that the first 8 bits shifted out should be ignored.
6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

31.10.19 Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed. See Performing Chip Erase.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM write using programming instruction 4a.
3. Load address high byte using programming instruction 4b.
4. Load address low byte using programming instruction 4c.
5. Load data using programming instructions 4d and 4e.
6. Repeat steps 4 and 5 for all data bytes in the page.
7. Write the data using programming instruction 4f.
8. Poll for EEPROM write complete using programming instruction 4g, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
9. Repeat steps 3 to 8 until all data have been programmed.

Note that the PROG_PAGELOAD instruction can not be used when programming the EEPROM

31.10.20 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.

Note that the PROG_PAGEREAD instruction can not be used when reading the EEPROM

31.10.21 Programming the Fuses

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse write using programming instruction 6a.
3. Load data byte using programming instructions 6b. A bit value of “0” will program the corresponding fuse, a “1” will unprogram the fuse.
4. Write Extended Fuse byte using programming instruction 6c.
5. Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
6. Load data byte using programming instructions 6e. A bit value of “0” will program the corresponding fuse, a “1” will unprogram the fuse.
7. Write Fuse high byte using programming instruction 6f.
8. Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).
9. Load data byte using programming instructions 6h. A “0” will program the fuse, a “1” will unprogram the fuse.
10. Write Fuse low byte using programming instruction 6i.
11. Poll for Fuse write complete using programming instruction 6j, or wait for t_{WLRH} (refer to table *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).

31.10.22 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 *Parallel Programming Characteristics, VCC = 5V ±10%* in chapter *Parallel Programming Characteristics*).

31.10.23 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 8f.
   - To only read Extended Fuse byte, use programming instruction 8b.
   - To only read Fuse high byte, use programming instruction 8c.
   - To only read Fuse low byte, use programming instruction 8d.
   - To only read Lock bits, use programming instruction 8e.

31.10.24 Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address 0x00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address 0x01 and address 0x02 to read the second and third signature bytes, respectively.

31.10.25 Reading the Calibration Byte

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Calibration byte read using programming instruction 10a.
3. Load address 0x00 using programming instruction 10b.
4. Read the calibration byte using programming instruction 10c.

32. Electrical Characteristics

32.1 Absolute Maximum Ratings

Table 32-1. Absolute Maximum Ratings

Operating Temperature	-40°C to +105°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except RESET with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on RESET with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current VCC and GND Pins	100.0mA

Note: 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.

Note: During parallel programming, a 12V signal is connected to the Reset pin. There is therefore no internal protection diode from the Reset pin to V_{CC}. To achieve the same protection on the Reset pin as on other I/O pins external protection should be added.

32.2 DC Characteristics

Table 32-2. Common DC characteristics T_A = -40°C to 105°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and RESET pin	V_{CC} = 1.8V - 2.4V		-0.5		$0.2V_{CC}^{(1)}$	V
		V_{CC} = 2.4V - 5.5V		-0.5		$0.3V_{CC}^{(1)}$	
V_{IH}	Input High Voltage, except XTAL1 and RESET pins	V_{CC} = 1.8V - 2.4V		$0.7V_{CC}^{(2)}$		V_{CC} + 0.5	V
		V_{CC} = 2.4V - 5.5V		$0.6V_{CC}^{(2)}$		V_{CC} + 0.5	
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}^{(1)}$	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V		$0.8V_{CC}^{(2)}$		V_{CC} + 0.5	V
		V_{CC} = 2.4V - 5.5V		$0.7V_{CC}^{(2)}$		V_{CC} + 0.5	
V_{OL}	Output Low Voltage$^{(4)}$ except RESET pin	I_{OL} = 20mA, V_{CC} = 5V	T_A=85°C			0.9	V
			T_A=105°C			1.0	

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
		I_{OL} = 10mA, V_{CC} = 3V	T_A=85°C			0.6	
			T_A=105°C			0.7	
V_{OH}	Output High Voltage(3) except Reset pin	I_{OH} = 20mA, V_{CC} = 5V	T_A=85°C	4.0			V
			T_A=105°C	3.9			
		I_{OH} = 10mA, V_{CC} = 3V	T_A=85°C	2.1			
			T_A=105°C	2.0			
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)				1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)				1	µA
R_{RST}	Reset Pull-up Resistor			20		70	kΩ
R_{PU}	I/O Pin Pull-up Resistor			20		50	kΩ
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 1.8-5V, $0V<V_{in}<V_{CC}$-0.1V			<10	40	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC}=5V , V_{in} = V_{CC}/2		-100		100	nA
t_{ACID}	Analog Comparator Propagation Delay				400		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 source more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   3.1. The sum of all I_{OH}, for ports PA0-PA7, PE4 should not exceed 100mA.
   3.2. The sum of all I_{OH}, for ports PC0-7, PD0-7, PE0-3 should not exceed 100mA.
   3.3. The sum of all I_{OH}, for ports RESET, PB0-7, PE5-6 should not exceed 100mA.
   If II_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
4. Although each I/O port can sink more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   4.1. The sum of all I_{OL}, for ports PA0-PA7, PE4 should not exceed 100mA.
   4.2. The sum of all I_{OL}, for ports RESET, PB0-7, PE5-6 should not exceed 100mA.
   4.3. The sum of all I_{OL}, for ports PC0-7, PE2-3 should not exceed 100mA.
   4.4. The sum of all I_{OL}, for ports PD0-7, PE0-1 should not exceed 100mA.
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.

Related Links

Minimizing Power Consumption

32.2.1 Power Consumption

Table 32-3. ATmega324PB DC Characteristics - T_A = -40°C to 105°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition		Min.	Typ.[2]	Max.	Units
I_{CC}	Power Supply Current[1]	Active 1MHz, V_{CC} = 2V		-	0.24	0.5	mA
		Active 4MHz, V_{CC} = 3V		-	1.5	2.5	
		Active 8MHz, V_{CC} = 5V		-	5.2	8.0	
		Idle 1MHz, V_{CC} = 2V		-	0.06	0.16	
		Idle 4MHz, V_{CC} = 3V		-	0.35	0.7	
		Idle 8MHz, V_{CC} = 5V		-	1.4	3.0	
	Power-save mode[3]	32kHz TOSC enabled, V_{CC} = 1.8V	T = 25°C	-	1.0	-	µA
			T = 85°C	-	1.3	-	
			T = 105°C	-	2	-	
		32kHz TOSC enabled, V_{CC} = 3V	T = 25°C	-	1.1	-	
			T = 85°C	-	1.4	-	
			T = 105°C	-	2.2	-	
	Power-down mode[3]	WDT enabled, V_{CC} = 3V	T = 25°C	-	2.9	8	
			T = 85°C	-	3.5	8	
			T = 105°C	-	4.2	15	
		WDT disabled, V_{CC} = 3V	T = 25°C	-	0.2	2	
			T = 85°C	-	0.7	2	
			T = 105°C	-	1.3	5	

Note:

1. Values with *Minimizing Power Consumption* enabled (PRR0=PRR1=0xFF).
2. Typical values are at 25°C unless otherwise noted.
3. The current consumption values include input leakage current. Inputs are tied externally low or high.

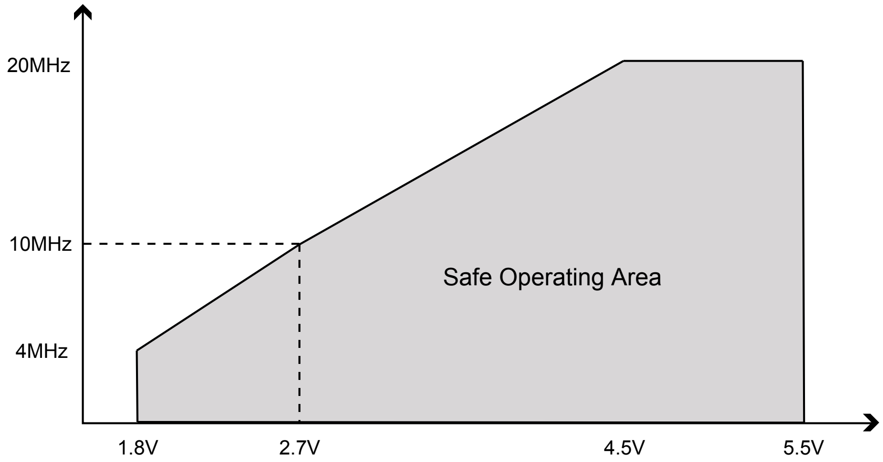
Related Links

Minimizing Power Consumption

32.3 Speed Grades

Maximum frequency is dependent on $V_{CC.}$ As shown in Figure. Maximum Frequency vs. V_{CC}, the Maximum Frequency vs. V_{CC} curve is linear between 1.8V < V_{CC} < 2.7V and between 2.7V < V_{CC} < 4.5V.

Figure 32-1. Maximum Frequency vs. V_{CC}



32.4 Clock Characteristics

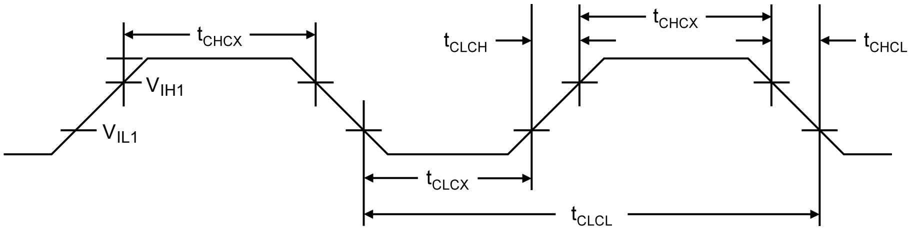
32.4.1 Calibrated Internal RC Oscillator Accuracy

Table 32-4. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{CC}	Temperature	Calibration Accuracy
Factory Calibration	8.0MHz	2.7V - 4.2V	0°C to +50°C	±2%
	8.0MHz	1.8V - 5.5V	0°C to +70°C	±3.5%
	8.0MHz	1.8V - 5.5V	-40°C to +105°C	±5%
User Calibration	Fixed frequency within: 7.3 - 8.1MHz	Fixed voltage within: 1.8V - 5.5V	Fixed temperature within: -40°C to +105°C	±1%

32.4.2 External Clock Drive Waveforms

Figure 32-2. External Clock Drive Waveforms



32.4.3 External Clock Drive

Table 32-5. External Clock Drive

Symbol	Parameter	V_{CC}= 1.8 - 5.5V		V_{CC}= 2.7 - 5.5V		V_{CC}= 4.5 - 5.5V		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	10	0	20	MHz
t_{CLCL}	Clock Period	250	-	100	-	50	-	ns
t_{CHCX}	High Time	100	-	40	-	20	-	ns
t_{CLCX}	Low Time	100	-	40	-	20	-	ns
t_{CLCH}	Rise Time	-	2.0	-	1.6	-	0.5	μs
t_{CHCL}	Fall Time	-	2.0	-	1.6	-	0.5	μs
Δt_{CLCL}	Change in period from one clock cycle to the next	-	2	-	2	-	2	%

32.5 System and Reset Characteristics

Table 32-6. Reset, Brown-out and Internal Voltage Characteristics(1)

Symbol	Parameter	Condition	Min.	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)		0.7	1.5	1.7	V
	Power-on Reset Threshold Voltage (falling)(2)		0.6	1.1	1.7	V
SR_{ON}	Power-on Slope Rate		0.01	-	9	V/ms
V_{RST}	RESET Pin Threshold Voltage		0.2 V_{CC}	-	0.9 V_{CC}	V
t_{RST}	Minimum pulse width on RESET Pin		-	-	2.5	μs
V_{HYST}	Brown-out Detector Hysteresis		-	50	-	mV
t_{BOD}	Min. Pulse Width on Brown-out Reset		-	2	-	μs
V_{BG}	Bandgap reference voltage		1.0	1.1	1.2	V
t_{BG}	Bandgap reference start-up time	V_{CC}=2.7 T_A=25°C	-	40	70	μs
I_{BG}	Bandgap reference current consumption	V_{CC}=2.7 T_A=25°C	-	10	-	μA

Note:

1. Values are guidelines only.
2. The Power-on Reset will not work unless the supply voltage is below V_{POT} (falling)

Table 32-7. BODLEVEL Fuse Coding(1)(2)

BODLEVEL [2:0] Fuses	Min. V_{BOT}	Typ. V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V

BODLEVEL [2:0] Fuses	Min. V_{BOT}	Typ. V_{BOT}	Max V_{BOT}	Units
101	2.5	2.7	2.9	
100	4.1	4.3	4.5	
011 - 000	Reserved			

Note: V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-Out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 110, 101 and 100.

32.6 SPI Timing Characteristics

Table 32-8. SPI Timing Parameters

Description	Mode	Min.	Typ	Max	Units
SCK period	Master	-	See Table. Relationship Between SCK and the Oscillator Frequency in "SPCR – SPI Control Register"	-	ns
SCK high/low	Master	-	50% duty cycle	-	
Rise/Fall time	Master	-	3.6	-	
Setup	Master	-	10	-	
Hold	Master	-	10	-	
Out to SCK	Master	-	$0.5 \cdot t_{sck}$	-	
SCK to out	Master	-	10	-	
SCK to out high	Master	-	10	-	
$\overline{\text{SS}}$ low to out	Slave	-	15	-	
SCK period	Slave	$4 \cdot t_{ck}$	-	-	
SCK high/low$^{(1)}$	Slave	$2 \cdot t_{ck}$	-	-	
Rise/Fall time	Slave	-	-	1600	
Setup	Slave	10	-	-	
Hold	Slave	t_{ck}	-	-	
SCK to out	Slave	-	15	-	
SCK to $\overline{\text{SS}}$ high	Slave	20	-	-	
$\overline{\text{SS}}$ high to tri-state	Slave	-	10	-	
$\overline{\text{SS}}$ low to SCK	Slave	$2 \cdot t_{ck}$	-	-	

Note:

1. In SPI Programming mode the minimum SCK high/low period is:

- 2 t_{CLCL} for f_{CK} < 12MHz
- 3 t_{CLCL} for f_{CK} > 12MHz

Figure 32-3. SPI Interface Timing Requirements (Master Mode)

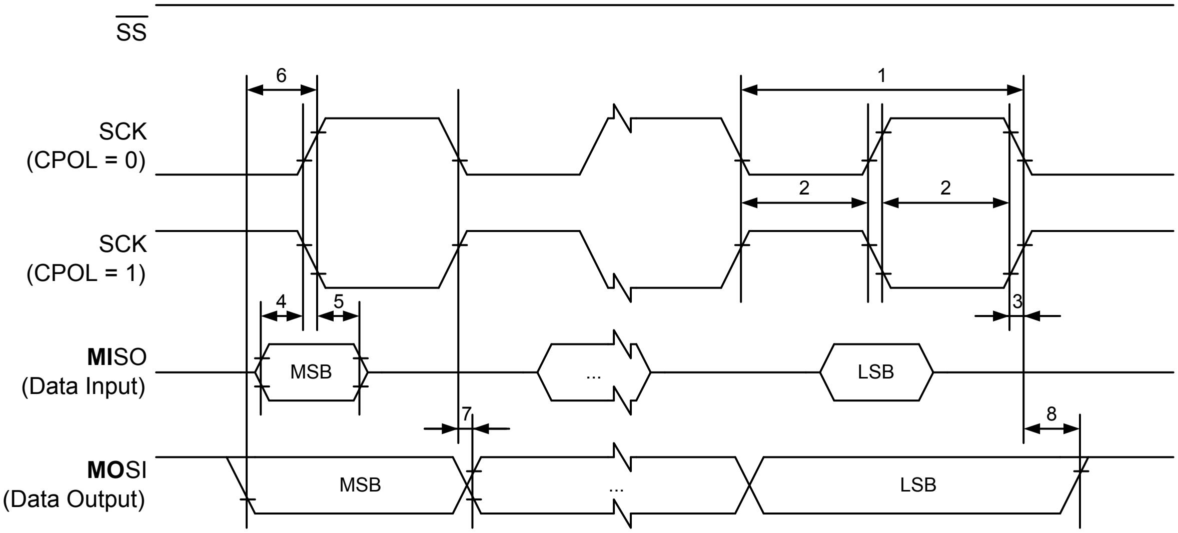
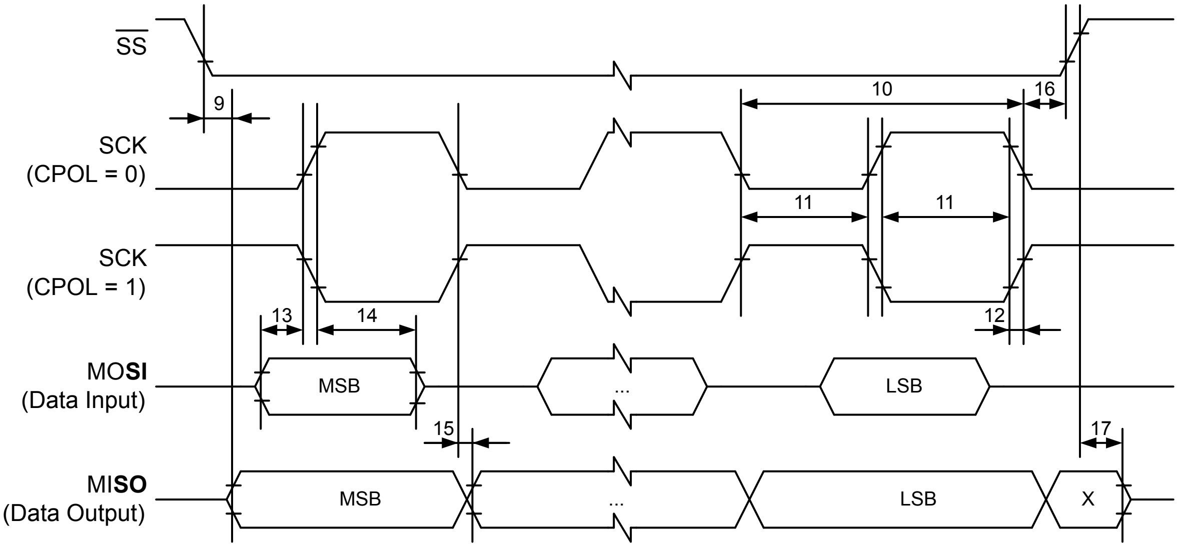


Figure 32-4. SPI Interface Timing Requirements (Slave Mode)



32.7 Two-wire Serial Interface Characteristics

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

Timing symbols refer to Figure 32-5.

Table 32-9. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min.	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.3 V_{CC}	V
V_{IH}	Input High-voltage		0.7 V_{CC}	V_{CC} + 0.5	V
V_{hys}(1)	Hysteresis of Schmitt Trigger Inputs		0.05 V_{CC}(2)	–	V
V_{OL}(1)	Output Low-voltage	3mA sink current	0	0.4	V

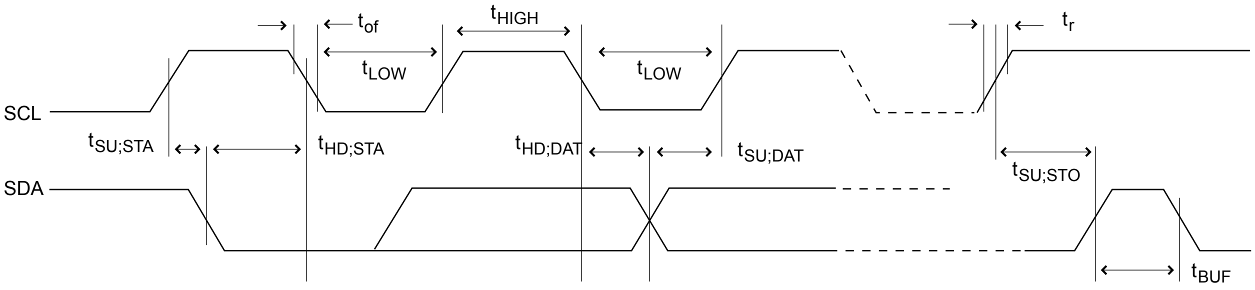
Symbol	Parameter	Condition	Min.	Max	Units
t_r(1)	Rise Time for both SDA and SCL		$20 + 0.1C_b$(3)(2)	300	ns
t_{of}(1)	Output Fall Time from V_{IHmin} to V_{ILmax}	$10pF < C_b < 400pF$(3)	$20 + 0.1C_b$(3)(2)	250	ns
t_{SP}(1)	Spikes Suppressed by Input Filter		0	50(2)	ns
I_i	Input Current each I/O Pin	$0.1V_{CC} < V_i < 0.9V_{CC}$	-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
Rp	Value of Pull-up resistor	$f_{SCL} \leq 100kHz$	$\frac{V_{CC} + -0}{3\text{mA}}$	$\frac{1000\text{ns}}{C_b}$	Ω
		$f_{SCL} > 100kHz$	$\frac{V_{CC} + -0}{3\text{mA}}$	$\frac{300\text{ns}}{C_b}$	Ω
$t_{HD;STA}$	Hold Time (repeated) START Condition	$f_{SCL} \leq 100kHz$	4.0	–	µs
		$f_{SCL} > 100kHz$	0.6	–	µs
t_{LOW}	Low Period of the SCL Clock	$f_{SCL} \leq 100kHz$	4.7	–	µs
		$f_{SCL} > 100kHz$	1.3	–	µs
t_{HIGH}	High period of the SCL clock	$f_{SCL} \leq 100kHz$	4.0	–	µs
		$f_{SCL} > 100kHz$	0.6	–	µs
$t_{SU;STA}$	Set-up time for a repeated START condition	$f_{SCL} \leq 100kHz$	4.7	–	µs
		$f_{SCL} > 100kHz$	0.6	–	µs
$t_{HD;DAT}$	Data hold time	$f_{SCL} \leq 100kHz$	0	3.45	µs
		$f_{SCL} > 100kHz$	0	0.9	µs
$t_{SU;DAT}$	Data setup time	$f_{SCL} \leq 100kHz$	250	–	ns
		$f_{SCL} > 100kHz$	100	–	ns
$t_{SU;STO}$	Setup time for STOP condition	$f_{SCL} \leq 100kHz$	4.0	–	µs
		$f_{SCL} > 100kHz$	0.6	–	µs
t_{BUF}	Bus free time between a STOP and START condition	$f_{SCL} \leq 100kHz$	4.7	–	µs
		$f_{SCL} > 100kHz$	1.3	–	µs

Note:

1. This parameter is characterized and not 100% tested.
2. Required only for $f_{SCL} > 100kHz$.
3. C_b = capacitance of one bus line in pF.
4. f_{CK} = CPU clock frequency.

5. This requirement applies to all 2-wire Serial Interface operation. Other devices connected to the 2-wire Serial Bus need only obey the general f_{SCL} requirement.

Figure 32-5. Two-wire Serial Bus Timing



32.8 ADC Characteristics

Table 32-10. ADC Characteristics, Single Ended Channel

Symbol	Parameter	Condition	Min.	Typ	Max	Units
	Resolution		-	10	-	Bits
TUE	Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 250kHz	-	3.5	7	LSB
		V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 1MHz	-	4	10	LSB
		V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 250kHz Noise Reduction Mode	-	3	7	LSB
		V_{REF} = 5V, V_{CC} = 4V, CLK_{ADC} = 1MHz Noise Reduction Mode	-	3.5	10	LSB
INL	Integral Non-Linearity	V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 250kHz	-	0.6	3	LSB
DNL	Differential Non-Linearity	V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 250kHz	-	0.3	1	LSB
	Gain Error	V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 250kHz	-7	3.5	7	LSB
	Offset Error	V_{REF} = 4V, V_{CC} = 5V, CLK_{ADC} = 250kHz	-5	-2.5	-0	LSB
	Conversion Time	Free Running Conversion	13	-	260	µs
	Clock Frequency		50	-	1000	kHz
AV_{CC}(1)	Analog Supply Voltage		V_{CC} - 0.3	-	V_{CC} + 0.3	V
V_{REF}	Reference Voltage		1.0	-	AV_{CC}	V
V_{IN}	Input Voltage		GND	-	V_{REF}	V
	Input Bandwidth		-	38.5	-	kHz

Symbol	Parameter	Condition	Min.	Typ	Max	Units
V_{INT}	Internal Voltage Reference 1.1V		1.0	1.1	1.2	V
	Internal Voltage Reference 2.56V	V_{DD} > 3V	2.3	2.56	2.8	V
R_{REF}	Reference Input Resistance		-	32	-	kΩ
R_{AIN}	Analog Input Resistance		-	100	-	MΩ

Note:

1. AV_{CC} absolute min./max: 1.8V/5.5V

Table 32-11. ADC Characteristics, Differential Channels

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution	Gain = 1×	-	10	-	Bits
		Gain = 10×	-	10	-	
		Gain = 200×	-	10	-	
TUE	Absolute Accuracy (Including INL, DNL Quantization Error and Offset Error)	Gain = 1×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	0	12	20	LSB
		Gain = 10×, V_{CC} = 5 V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-10	14	30	
		Gain = 200×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-10	10	30	
INL	Integral Non-linearity (1)	Gain = 1×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-	1	3.5	LSB
		Gain = 10×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-	1	3.5	
		Gain = 200×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-	2	4	

Symbol	Parameter	Condition	Min	Typ	Max	Units
DNL	Differential Non-linearity[1]	Gain = 1×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-	0.5	2	LSB
		Gain = 10×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-	0.75	2.5	
		Gain = 200×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-	0.75	2.5	
	Gain Error	Gain = 1×, V_{CC} = 5V, VREF = 4V, CLK_{ADC} = 250 kHz	-20	10	40	LSB
		Gain = 10×, VCC = 5V, VREF = 4V, CLK_{ADC} = 250 kHz	-20	10	40	
		Gain = 200×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-20	2	20	
	Midcode Offset	Gain = 1×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-10	-3.5	10	LSB
		Gain = 10×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-15	-3.5	15	
		Gain = 200×, V_{CC} = 5V, V_{REF} = 4V, CLK_{ADC} = 250 kHz	-20	3	20	
	Conversion Time		52	-	260	µs

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Clock Frequency		50	-	250	kHz
AVCC	Analog Supply Voltage		V_{CC} - 0.3	-	V_{CC} + 0.3	V
V_{REF}	Reference Voltage$^{(2)}$		2.0	-	AV_{CC} - 0.5	
V_{DIFF}	Input Differential Voltage		0	-	V_{REF}/Gain	
	ADC Conversion Output		-511	-	511	LSB
	Input Bandwidth		-	-	9.6	kHz
V_{INT}	Internal Voltage Reference 1.1V		1.0	1.1	1.2	V
	Internal Voltage Reference 2.56V	V_{DD}>3.3V (rational: V_{INT}(max)+0.5V)	2.3	2.56	2.8	
R_{REF}	Reference Input Resistance		-	32	-	kΩ

Note:

1. INL and DNL are calculated based on 90% of FSR (Full Scale Range).
2. It is not recommended to use an external AREF higher than (Vdd-1V) in differential mode, as this will affect ADC accuracy.

32.9 Parallel Programming Characteristics

Table 32-12. Parallel programming characteristics, VCC = 5V ±10%.

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{PP}	Programming Enable Voltage	11.5	-	12.5	V
I_{PP}	Programming Enable Current		-	250	μA
t_{DVXH}	Data and Control Valid before XTAL1 High	67	-	-	ns
t_{XLXH}	XTAL1 Low to XTAL1 High	300	-	-	ns
t_{XHXL}	XTAL1 Pulse Width High	150	-	-	ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67	-	-	ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0	-	-	ns
t_{XLPH}	XTAL1 Low to PAGEL high	0	-	-	ns
t_{PLXH}	PAGEL low to XTAL1 high	150	-	-	ns
t_{BVPH}	BS1 Valid before PAGEL High	67	-	-	ns
t_{PHPL}	PAGEL Pulse Width High	200	-	-	ns
t_{PLBX}	BS1 Hold after PAGEL Low	67	-	-	ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67	-	-	ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67	-	-	ns

Symbol	Parameter	Min.	Typ.	Max.	Units
t_{BVWL}	BS2/1 Valid to $\overline{\text{WR}}$ Low	67	-	-	ns
t_{WLWH}	$\overline{\text{WR}}$ Pulse Width Low	150	-	-	ns
t_{WLRL}	$\overline{\text{WR}}$ Low to RDY/$\overline{\text{BSY}}$ Low	0	-	1	μs
t_{WLRH}	WR Low to RDY/BSY High[1]	2	-	4.5	ms
t_{WLRH_CE}	$\overline{\text{WR}}$ Low to RDY/$\overline{\text{BSY}}$ High for Chip Erase[2]	7.5	-	12	
t_{XLOL}	XTAL1 Low to $\overline{\text{OE}}$ Low	0	-	-	ns
t_{BVDV}	BS1 Valid to DATA valid	0	-	500	
t_{OLDV}	$\overline{\text{OE}}$ Low to DATA Valid	-	-	500	
t_{OHDZ}	$\overline{\text{OE}}$ High to DATA Tri-stated	-	-	500	

Note:

1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.
2. t_{WLRH_CE} is valid for the Chip Erase command.

Figure 32-6. Parallel programming timing, including some general timing requirements.

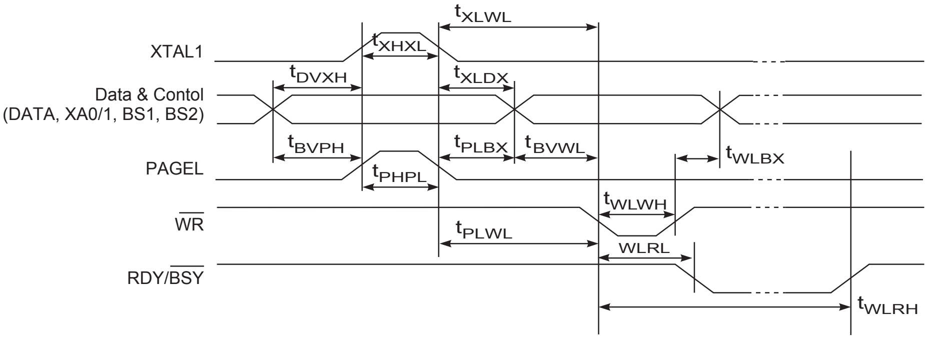
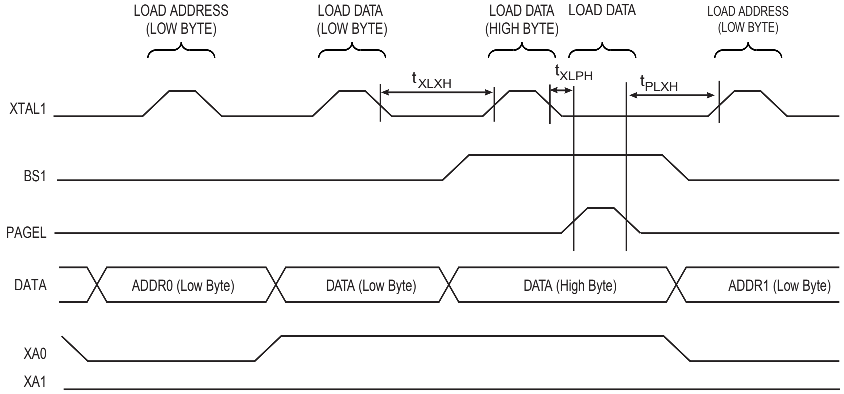
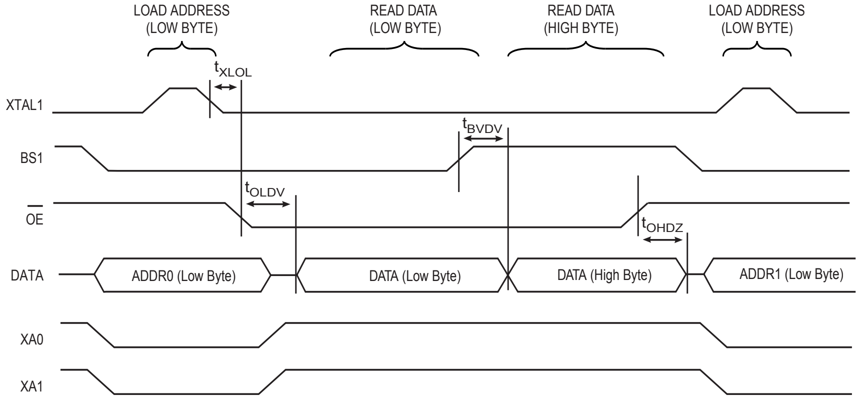


Figure 32-7. Parallel programming timing, loading sequence with timing requirements



Note: The timing requirements shown in Figure 32-6 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 32-8. Parallel programming timing, reading sequence (within the same page) with timing requirements



Note: The timing requirements shown in Figure 32-6 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

33. Typical Characteristics

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

All Active- and Idle current consumption measurements are done with all bits in the PRR registers set and thus, the corresponding I/O modules are turned off. Also the Analog Comparator is disabled during these measurements. 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.

33.1 Active Supply Current

Figure 33-1. ATmega324PB: Active Supply Current vs. Low Frequency (0.1-1.0MHz)

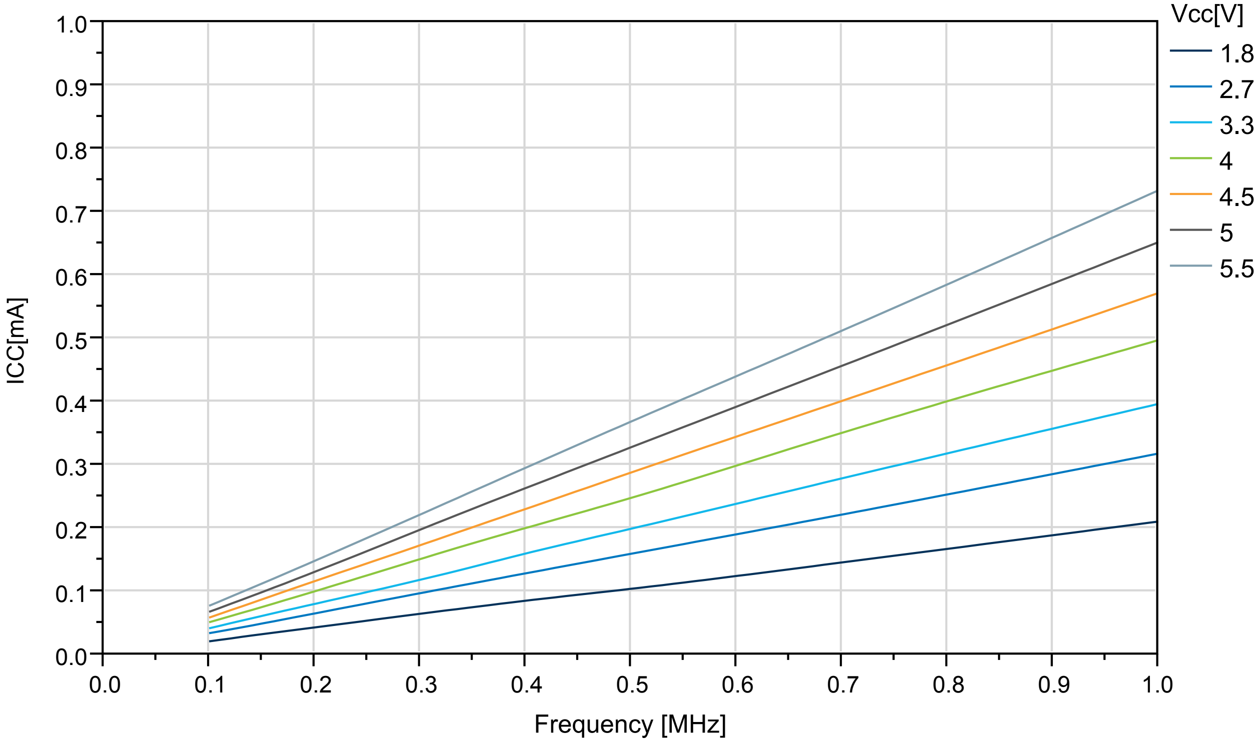


Figure 33-2. ATmega324PB: Active Supply Current vs. Frequency (1-20MHz)

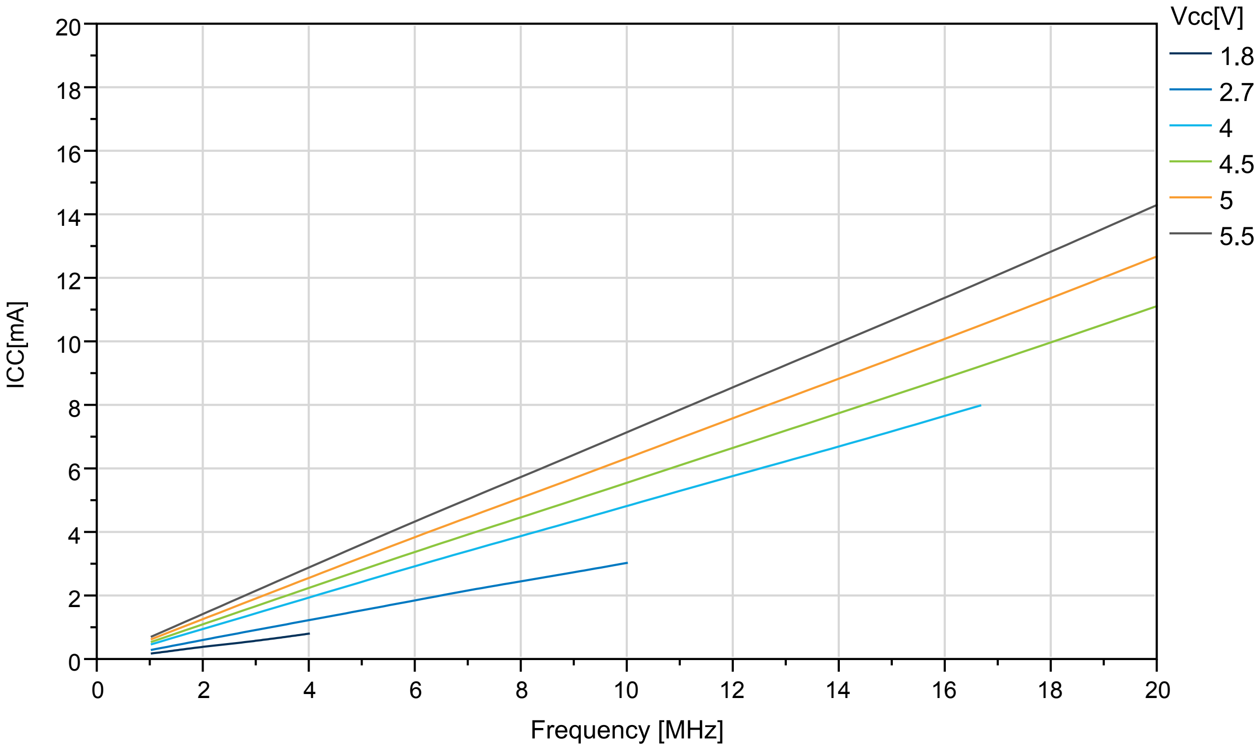


Figure 33-3. ATmega324PB: Active Supply Current vs. V_{CC} (Internal RC Oscillator, 128kHz)

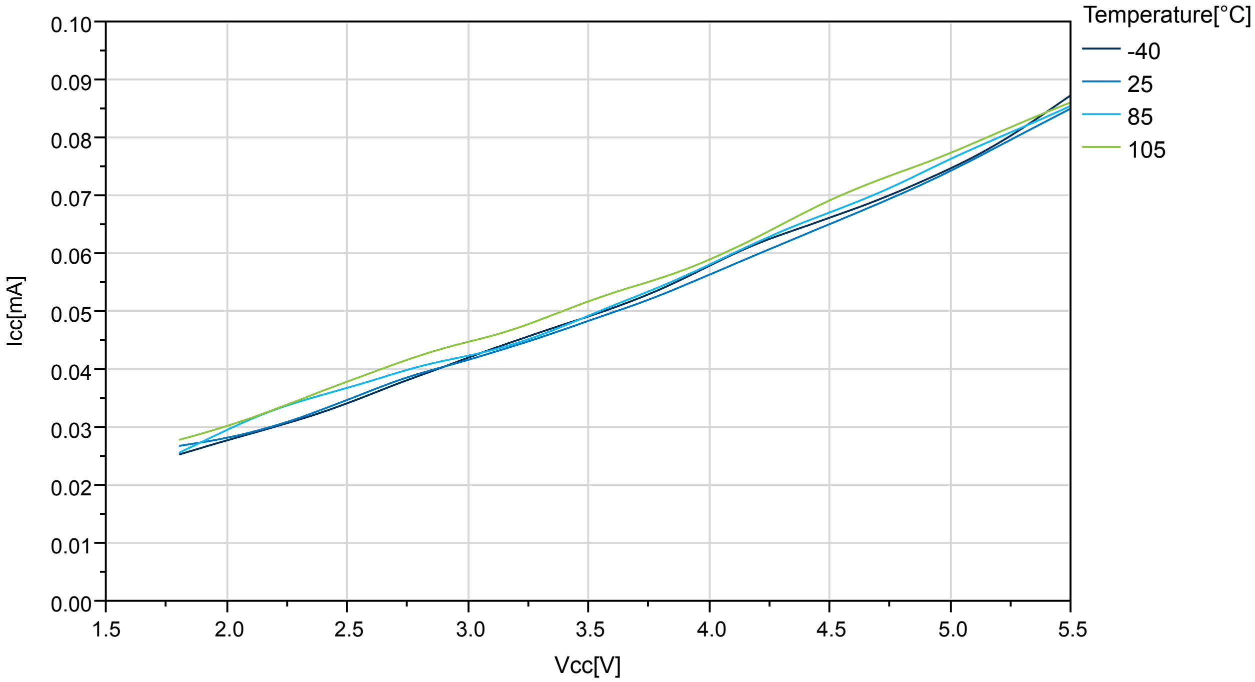


Figure 33-4. ATmega324PB: Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1MHz)

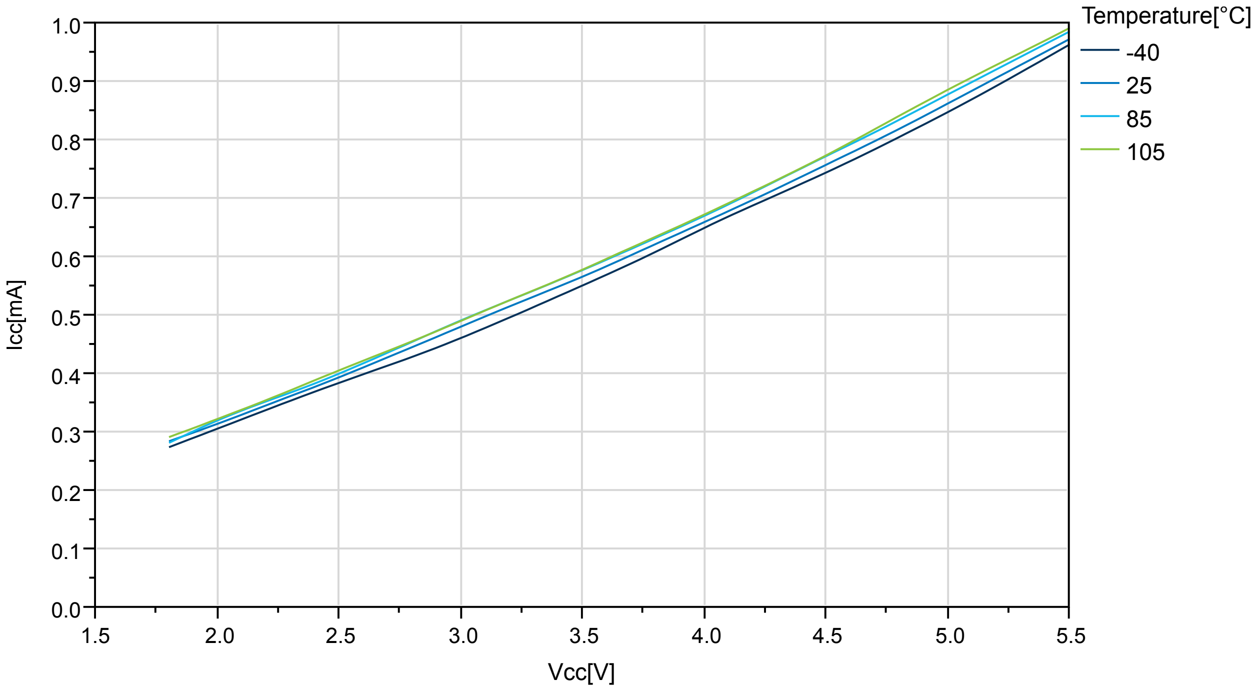


Figure 33-5. ATmega324PB: Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8MHz)

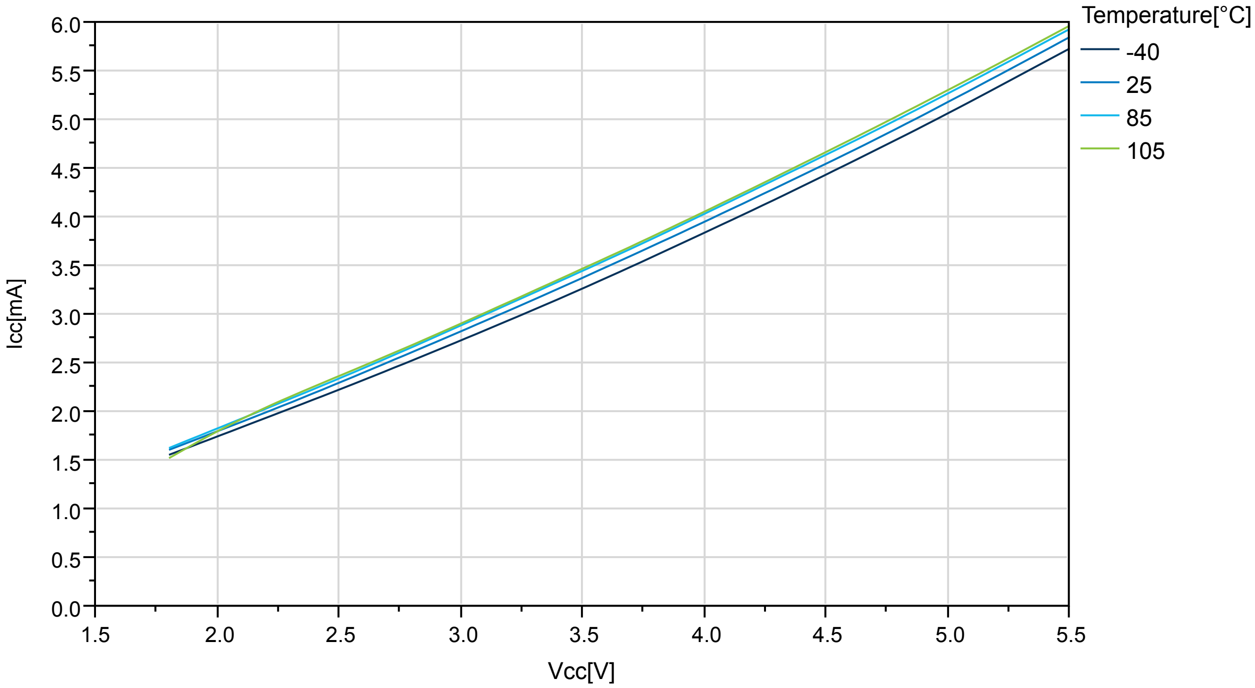


Figure 33-6. ATmega324PB: Active Supply Current vs. VCC (With ADC, Conversion 50kHz)

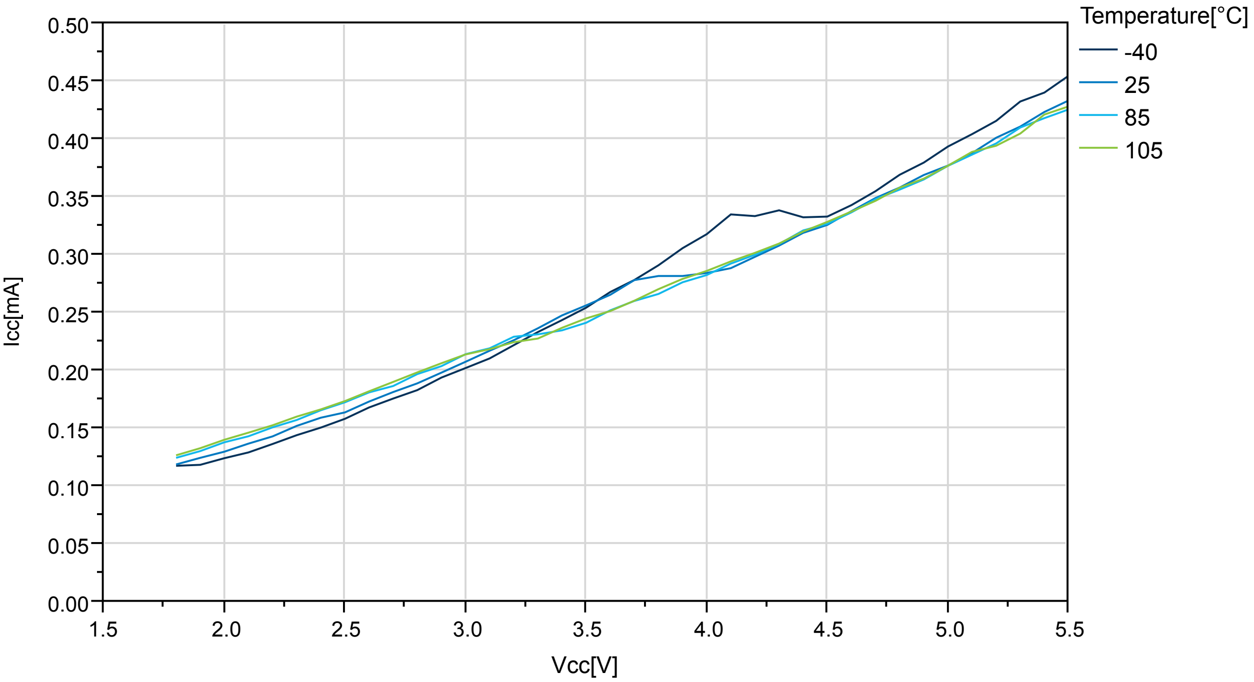
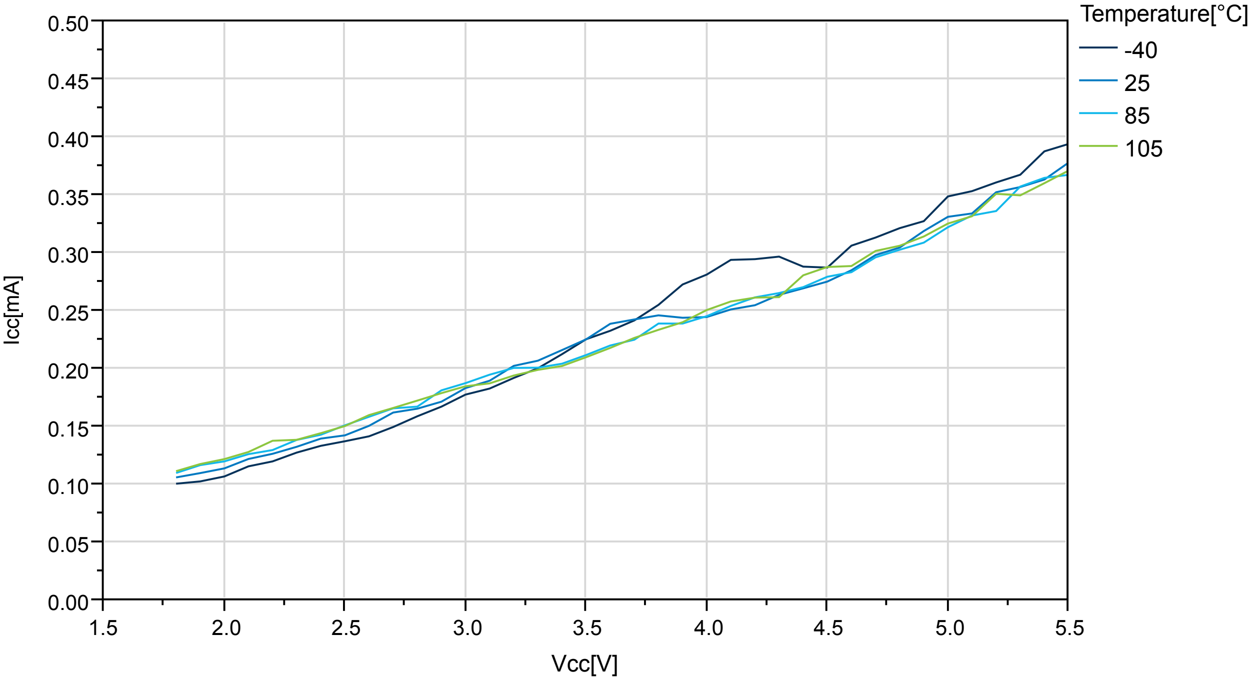


Figure 33-7. ATmega324PB: Active Supply Current vs. VCC (With ADC, No conversion)



33.2 Idle Supply Current

Figure 33-8. ATmega324PB: Idle Supply Current vs. Low Frequency (0.1-1.0MHz)

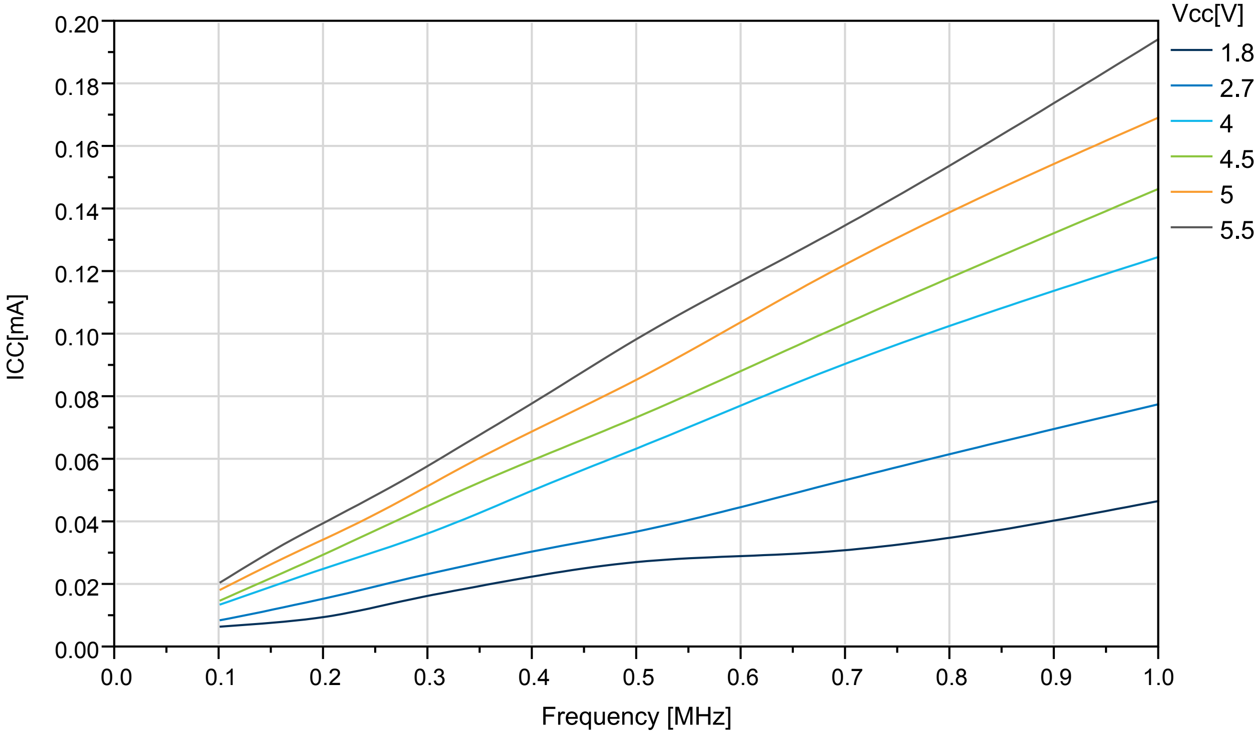


Figure 33-9. ATmega324PB: Idle Supply Current vs. Frequency (1-20MHz)

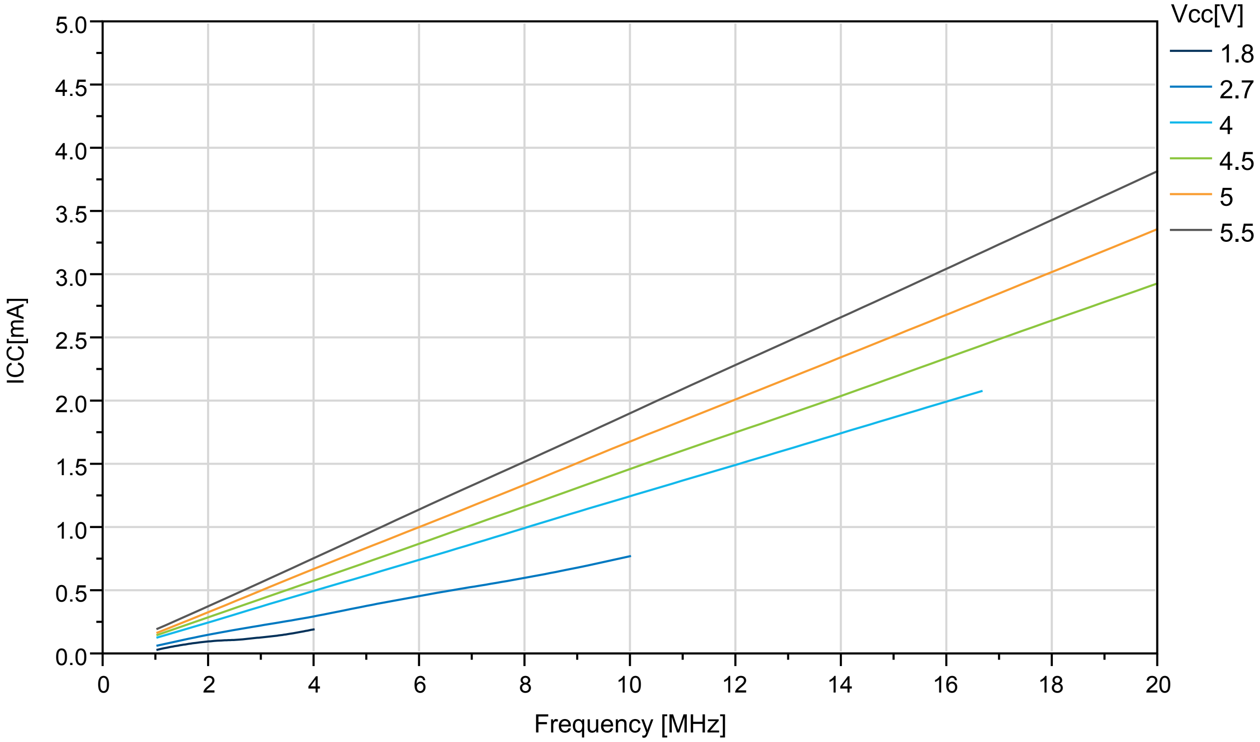


Figure 33-10. ATmega324PB: Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 128kHz)

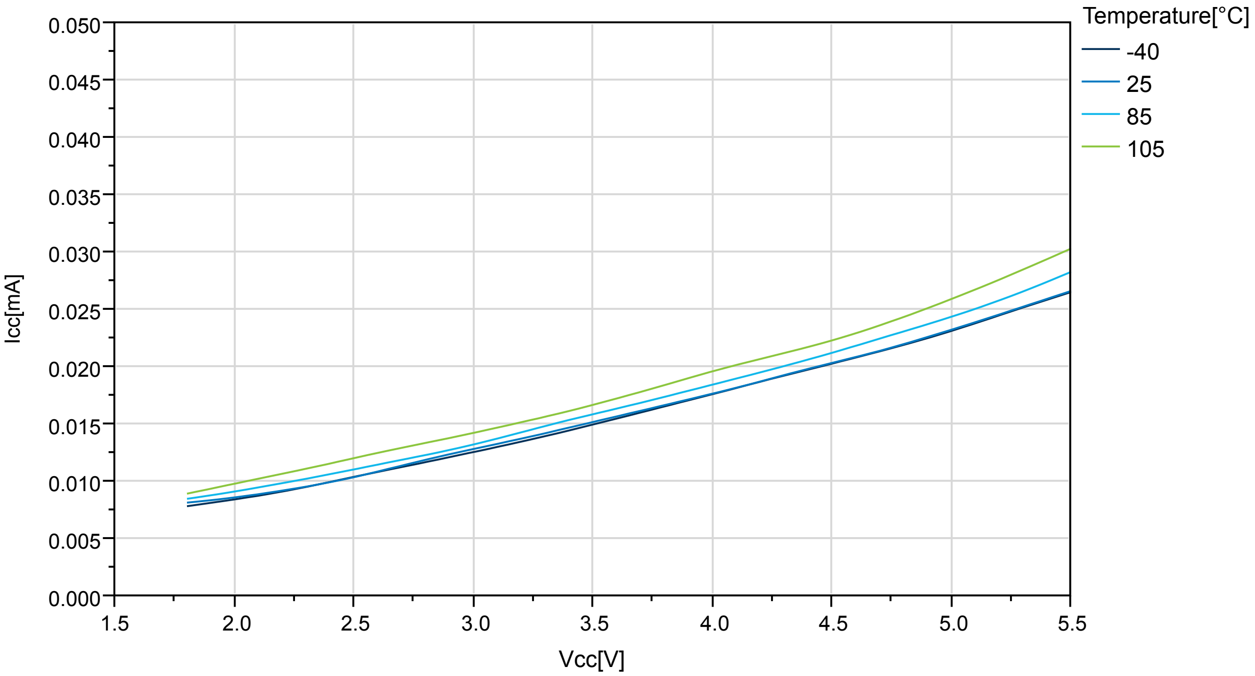


Figure 33-11. ATmega324PB: Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1MHz)

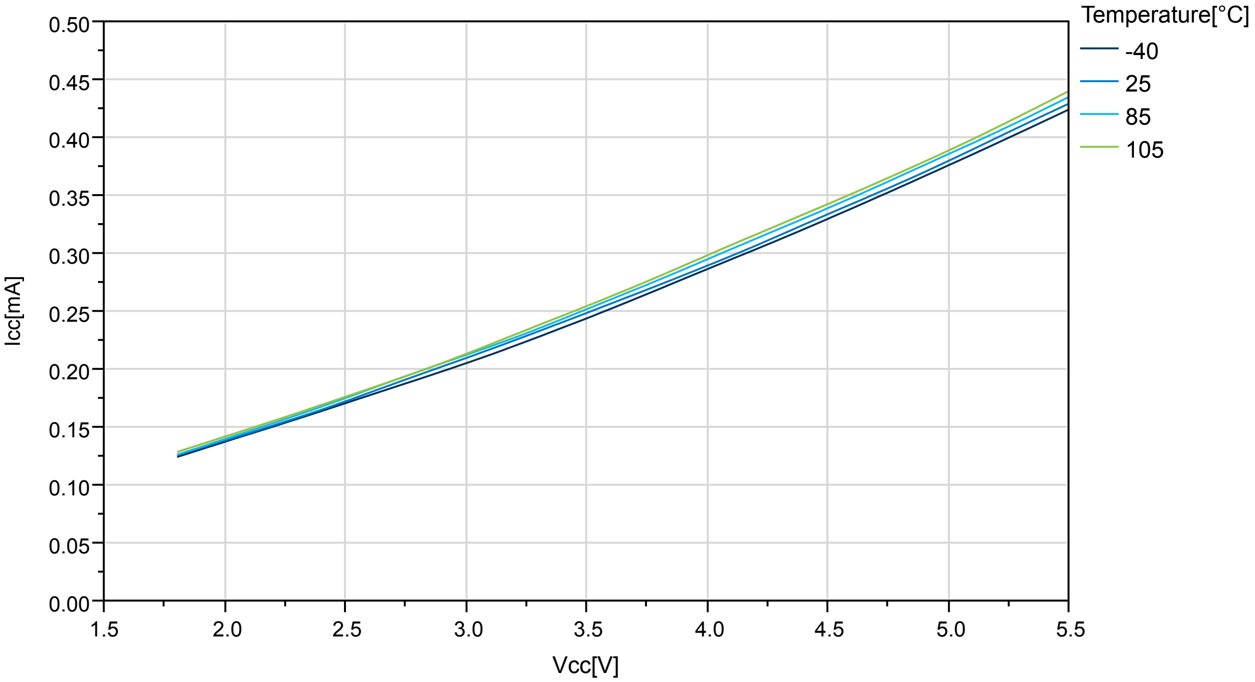
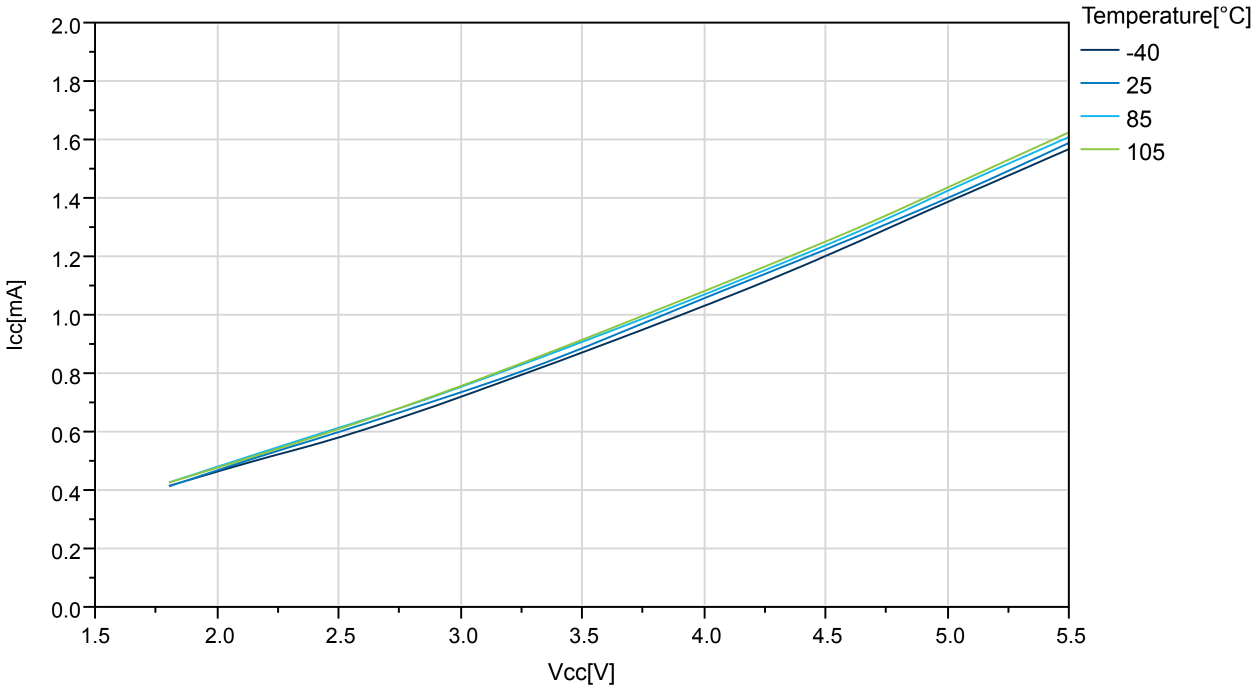


Figure 33-12. ATmega324PB: Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8MHz)



33.3 Supply Current of IO Modules

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules are controlled by the Power Reduction Register.

Table 33-1. Additional Current Consumption for the different I/O modules (absolute values)

PRR bit	Typical numbers		
	V_{CC} = 2V, F = 1MHz	V_{CC} = 3V, F = 4MHz	V_{CC} = 5V, F = 8MHz
PRUSART0	5.8µA	30µA	119µA
PRTWI	6.6µA	38µA	150µA
PRTIM2	7.4µA	43µA	170µA
PRTIM1	4.1µA	26µA	98µA
PRTIM0	2.9µA	13µA	50µA
PRSPI	5.3µA	35µA	145µA
PRADC	7.1µA	50µA	197µA
PRSPI1	2.8µA	25µA	104µA
PRUSART1	3.8µA	28µA	104µA
PRTIM3	4.6µA	28µA	107µA
PRTIM4	3.6µA	29µA	108µA
RUSART2	4.7µA	30µA	122µA

PRR bit	Typical numbers		
	V_{CC} = 2V, F = 1MHz	V_{CC} = 3V, F = 4MHz	V_{CC} = 5V, F = 8MHz
PRTWI1	6.5µA	41µA	158µA
PRPTC	39µA	248µA	990µA

Related Links

Power Reduction Registers

33.4 Power-down Supply Current

Figure 33-13. ATmega324PB: Power-Down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

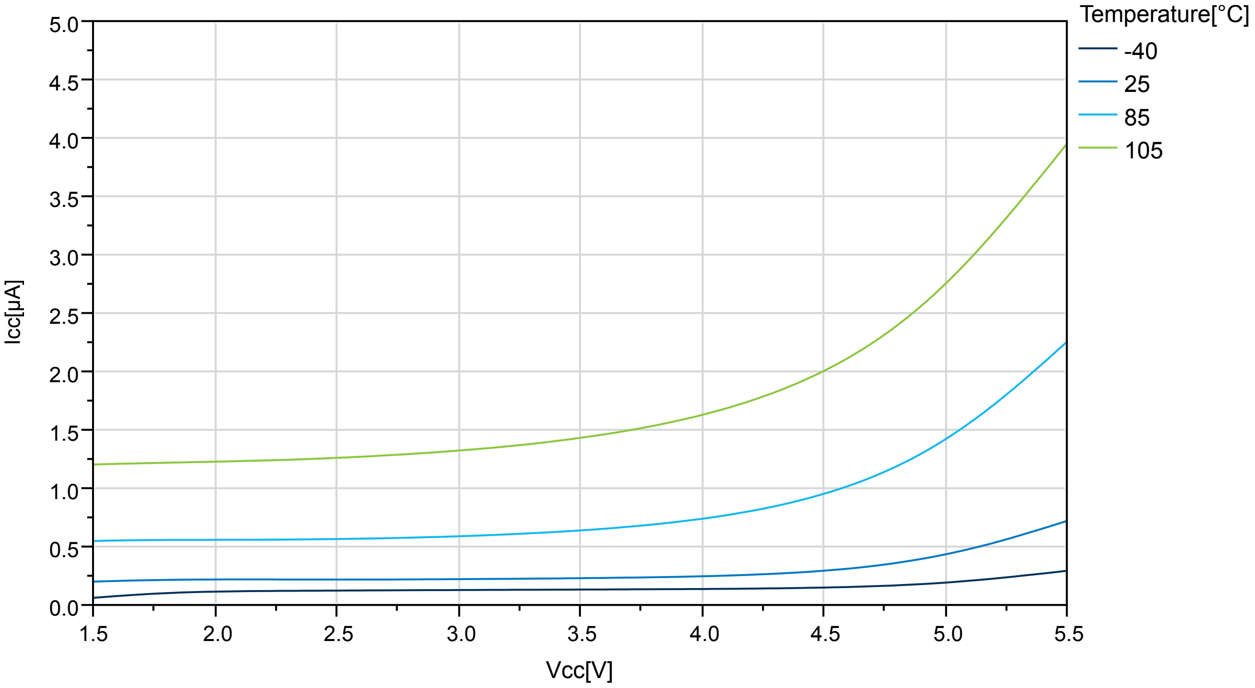


Figure 33-14. ATmega324PB: Power-Down Supply Current vs. V_{CC} (Watchdog Timer Enabled)

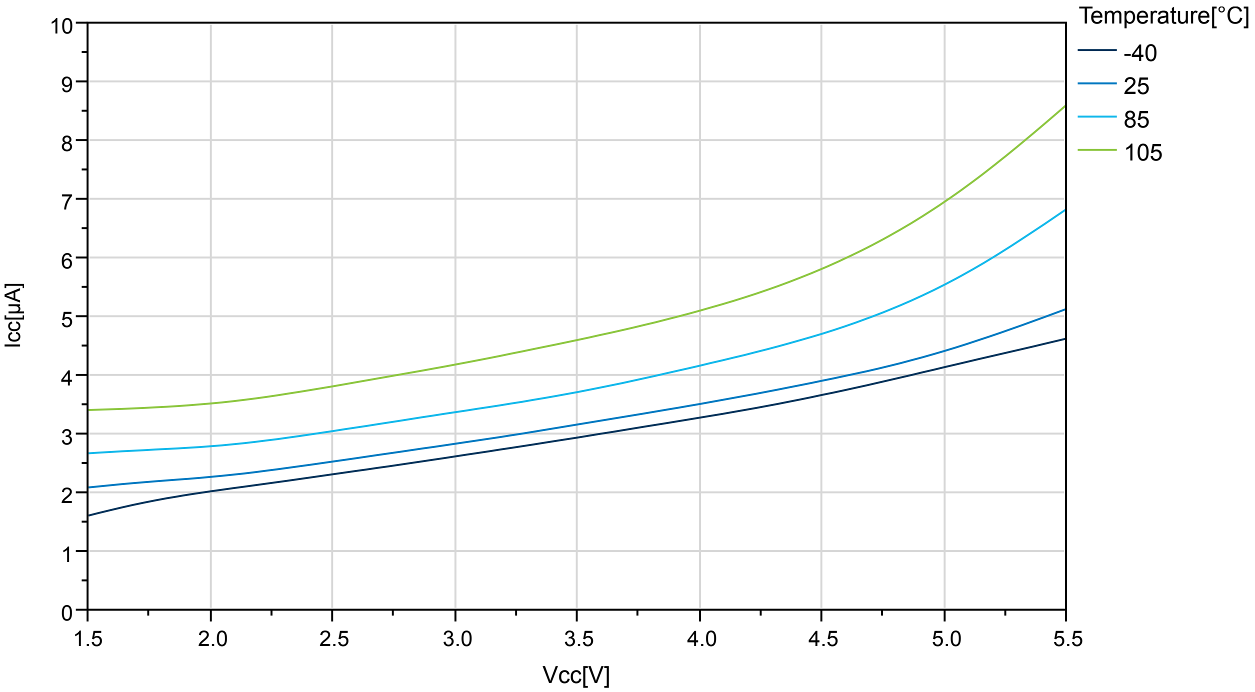
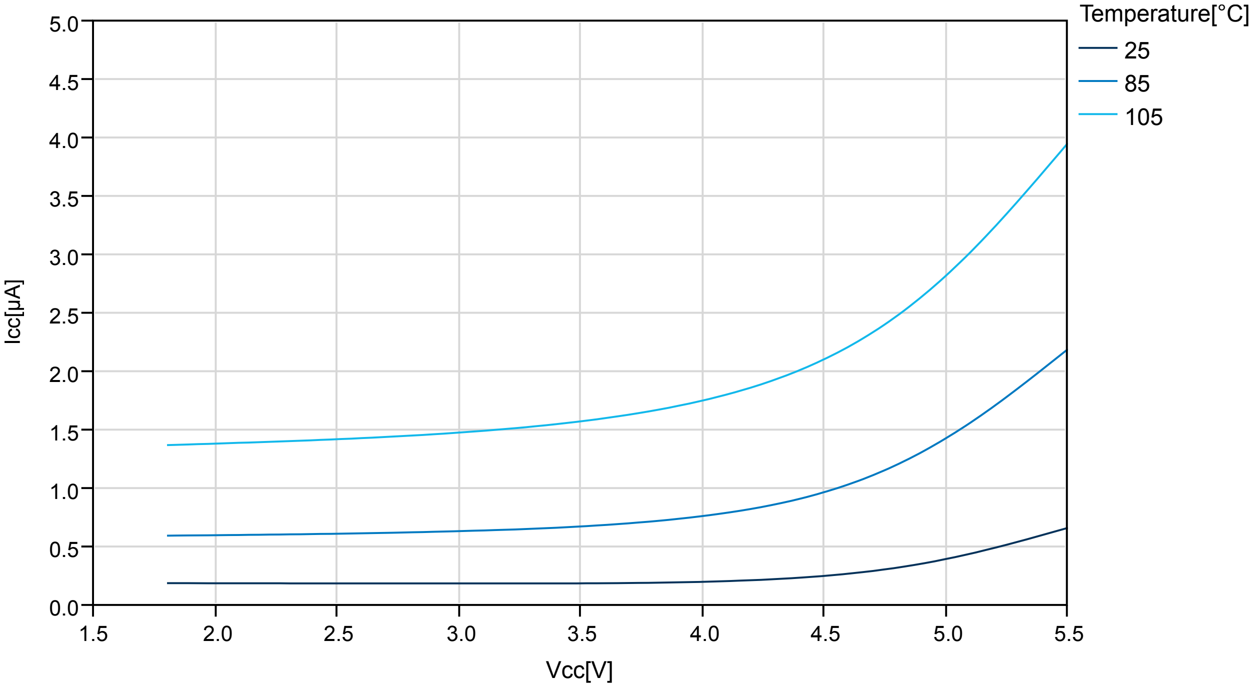


Figure 33-15. ATmega324PB: Power-Down Supply Current vs. VCC (AREF, VCCDIV2)



33.5 Pin Pull-Up

Figure 33-16. ATmega324PB: I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 1.8V)

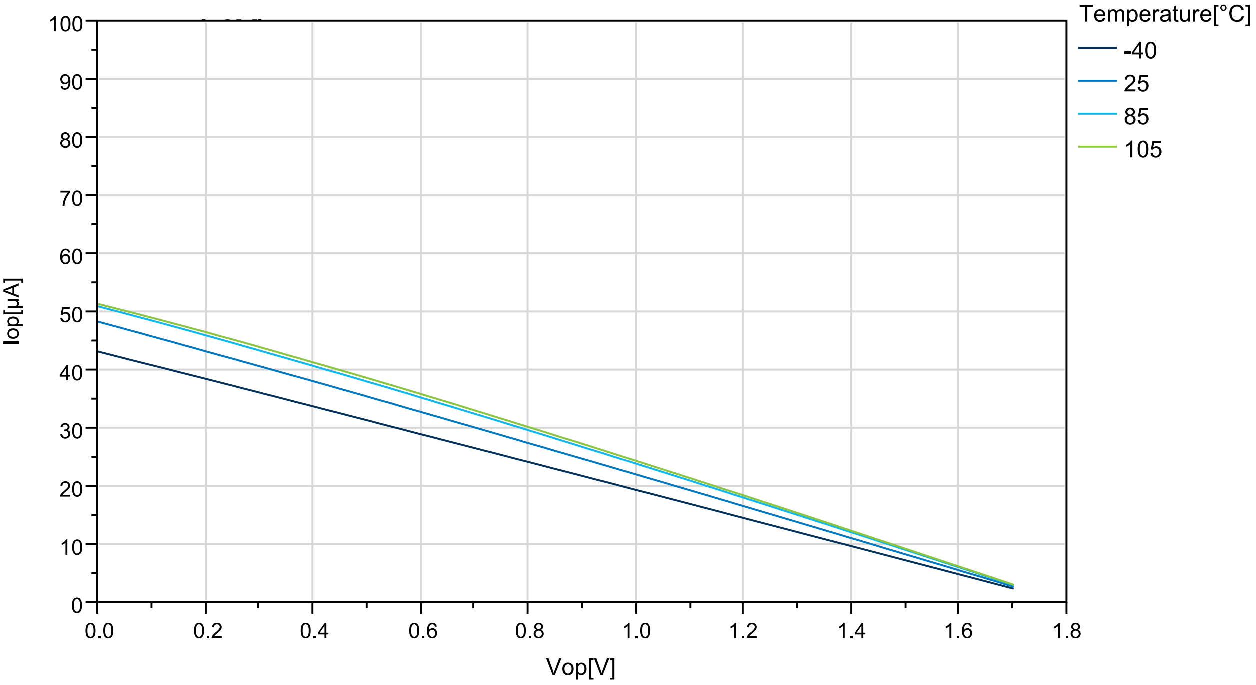


Figure 33-17. ATmega324PB: I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

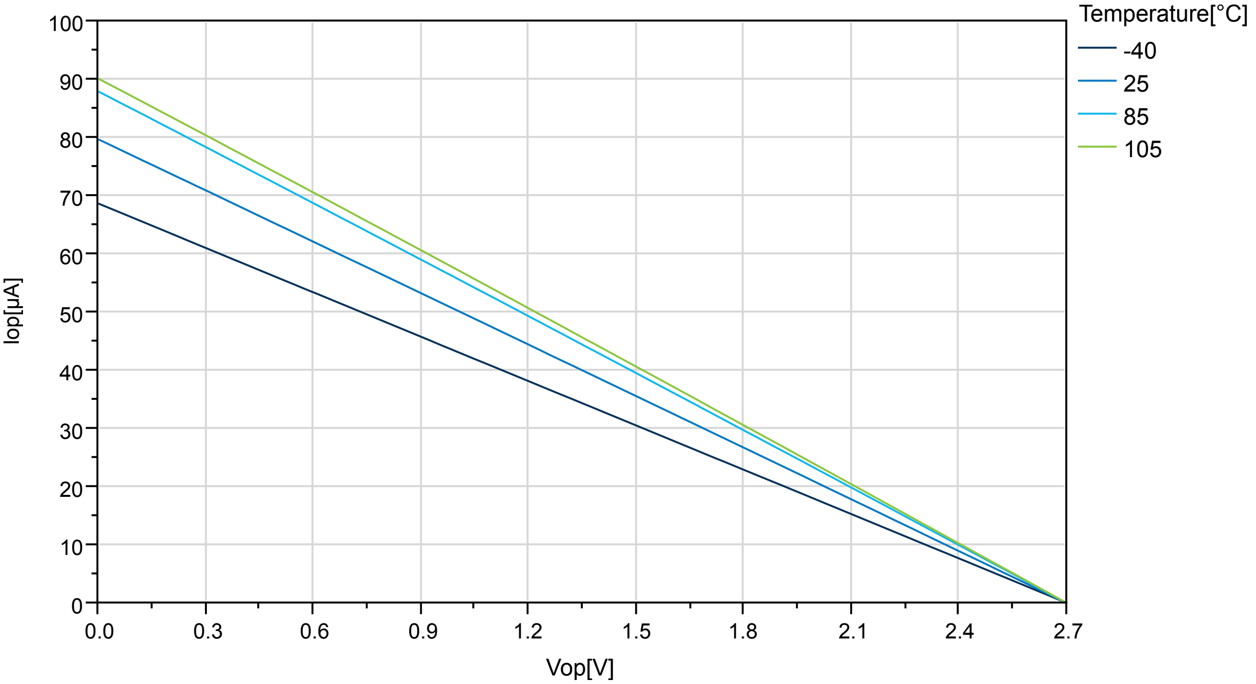


Figure 33-18. ATmega324PB: I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

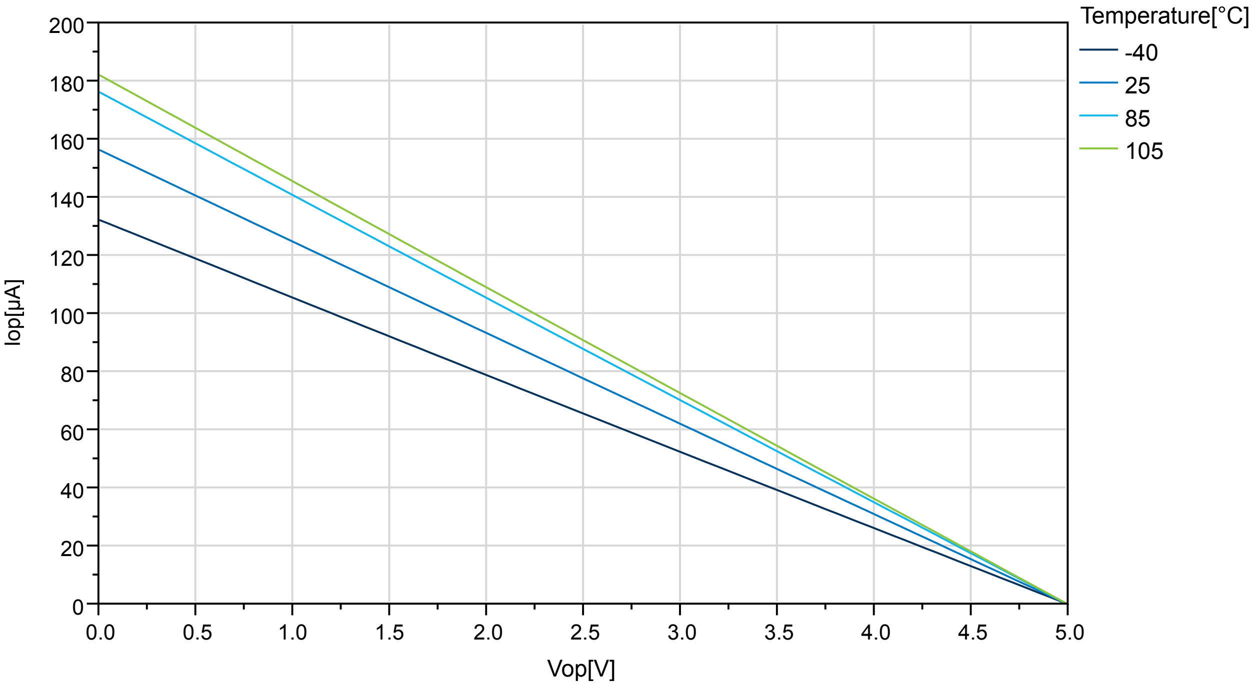


Figure 33-19. ATmega324PB: Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 1.8V)

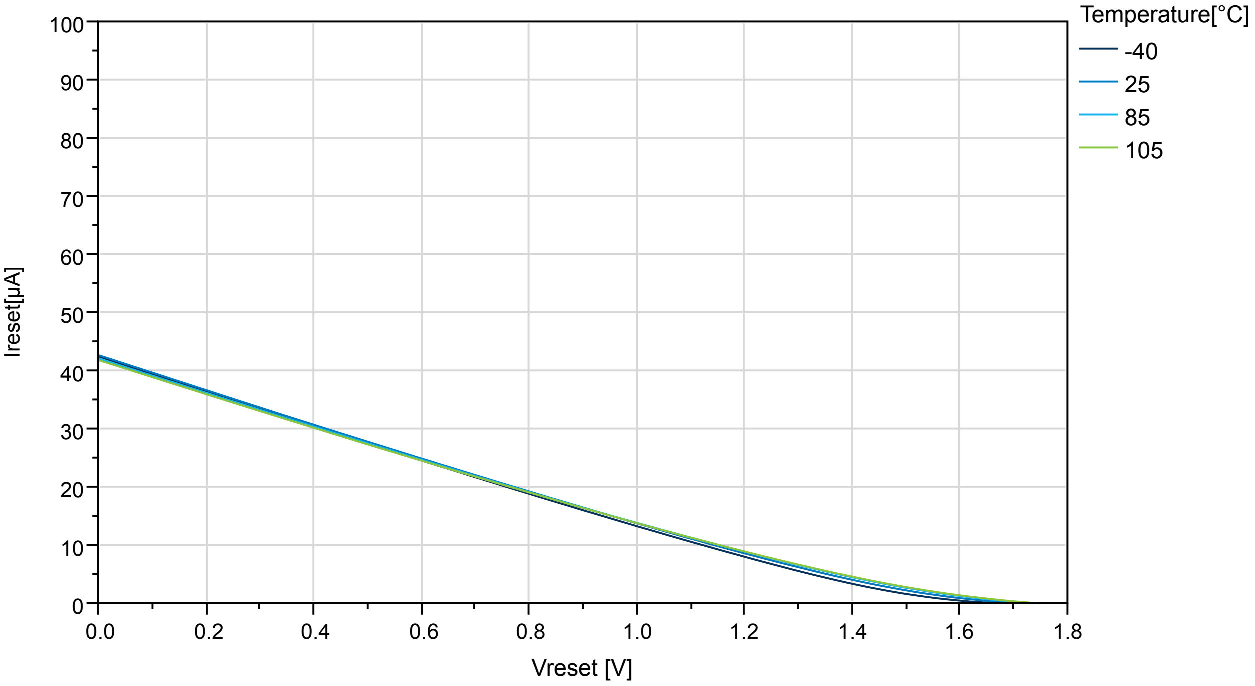


Figure 33-20. ATmega324PB: Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)

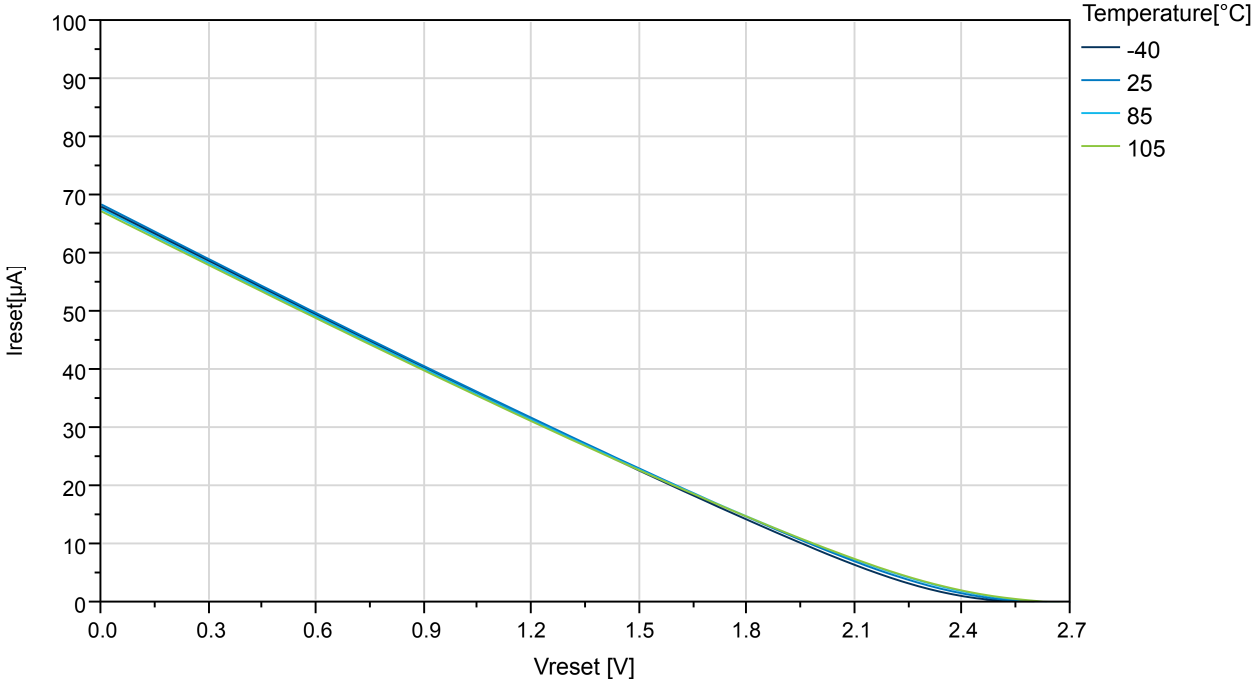
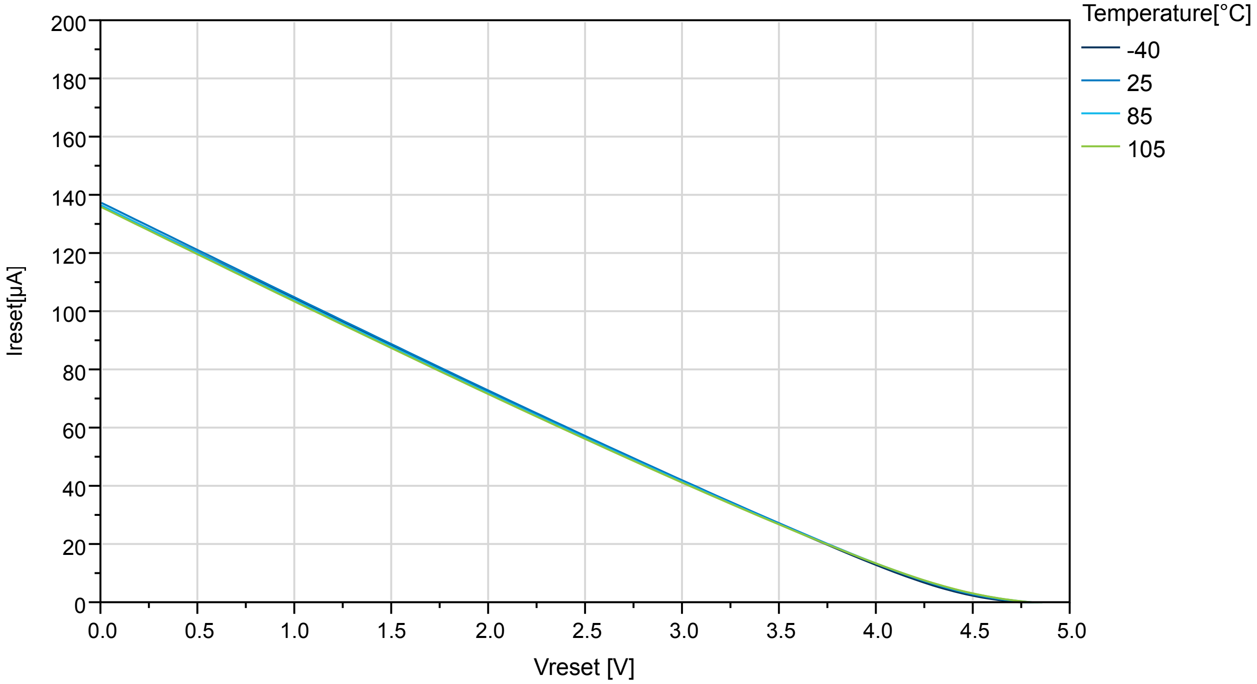


Figure 33-21. ATmega324PB: Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)



33.6 Pin Driver Strength

Figure 33-22. ATmega324PB: I/O Pin Output Voltage vs. Sink Current (V_{CC} = 3V)

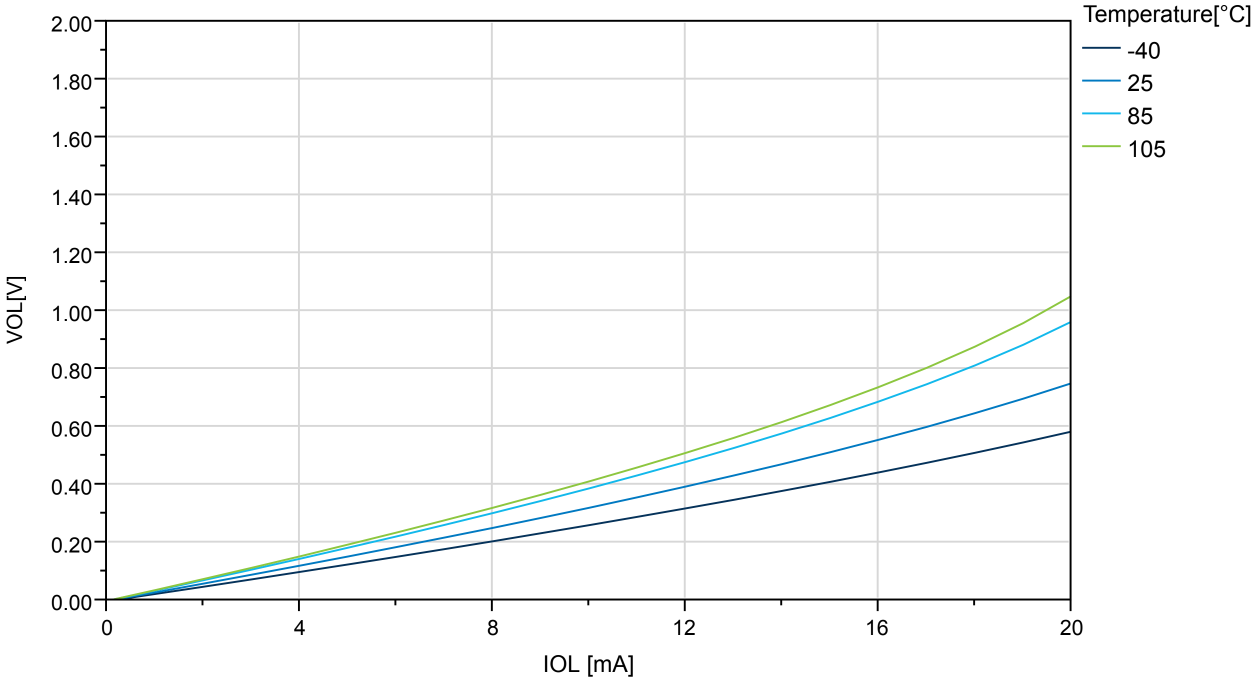


Figure 33-23. ATmega324PB: I/O Pin Output Voltage vs. Sink Current (V_{CC} = 5V)

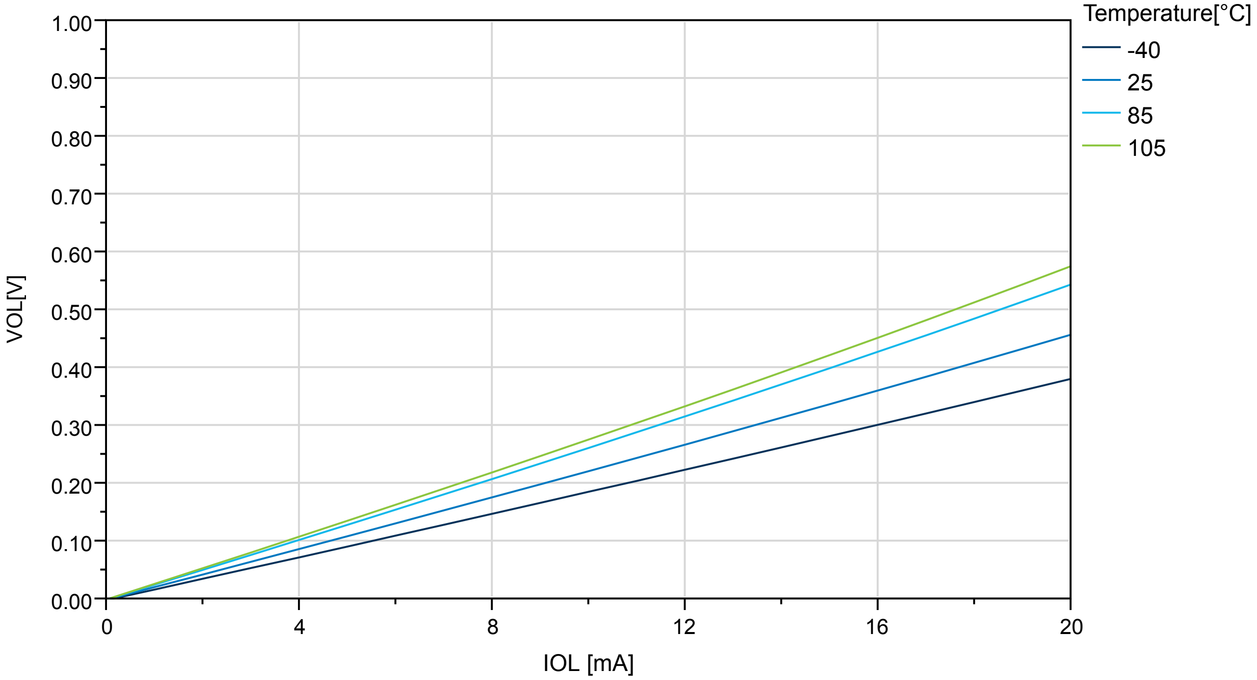


Figure 33-24. ATmega324PB: I/O Pin Output Voltage vs. Source Current (V_{CC} = 3V)

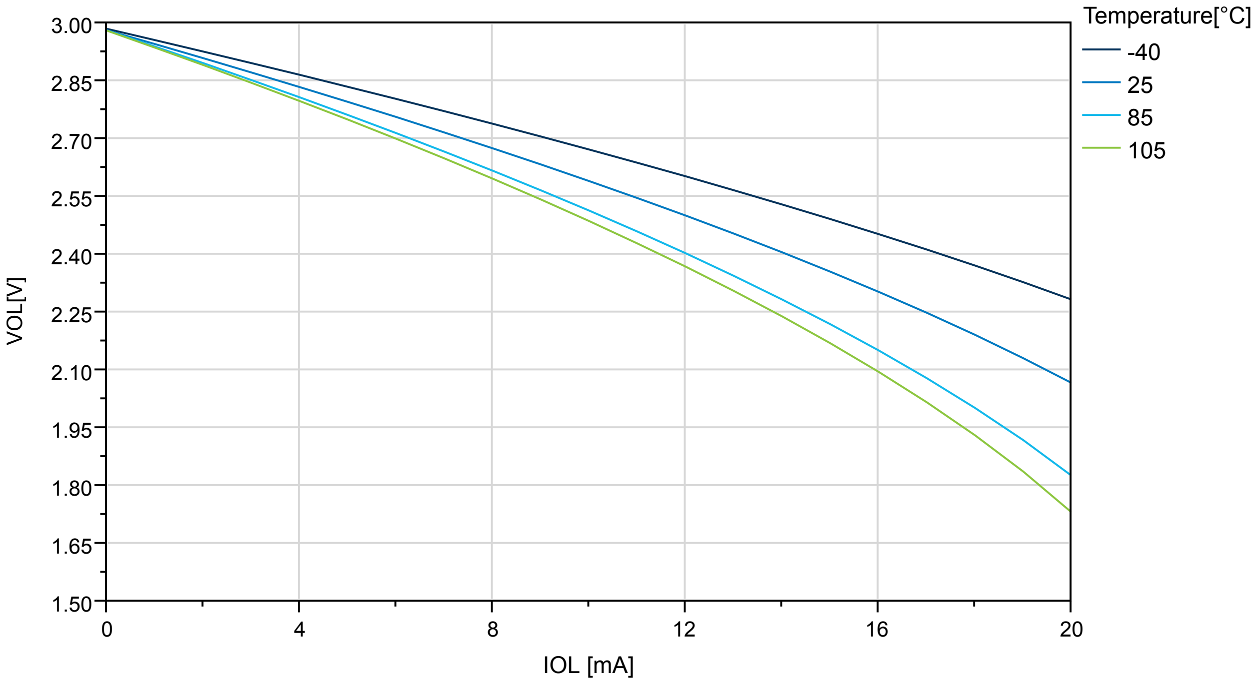
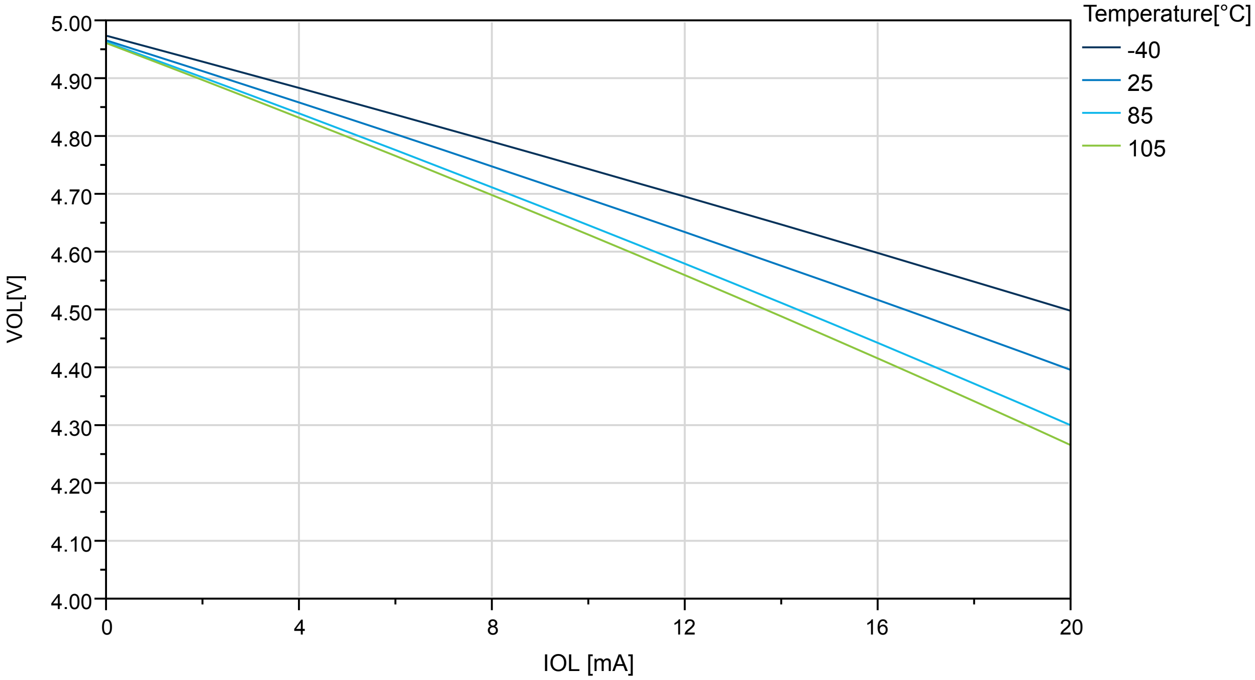


Figure 33-25. ATmega324PB: I/O Pin Output Voltage vs. Source Current (V_{CC} = 5V)



33.7 Pin Threshold and Hysteresis

Figure 33-26. ATmega324PB I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin read as ‘1’)

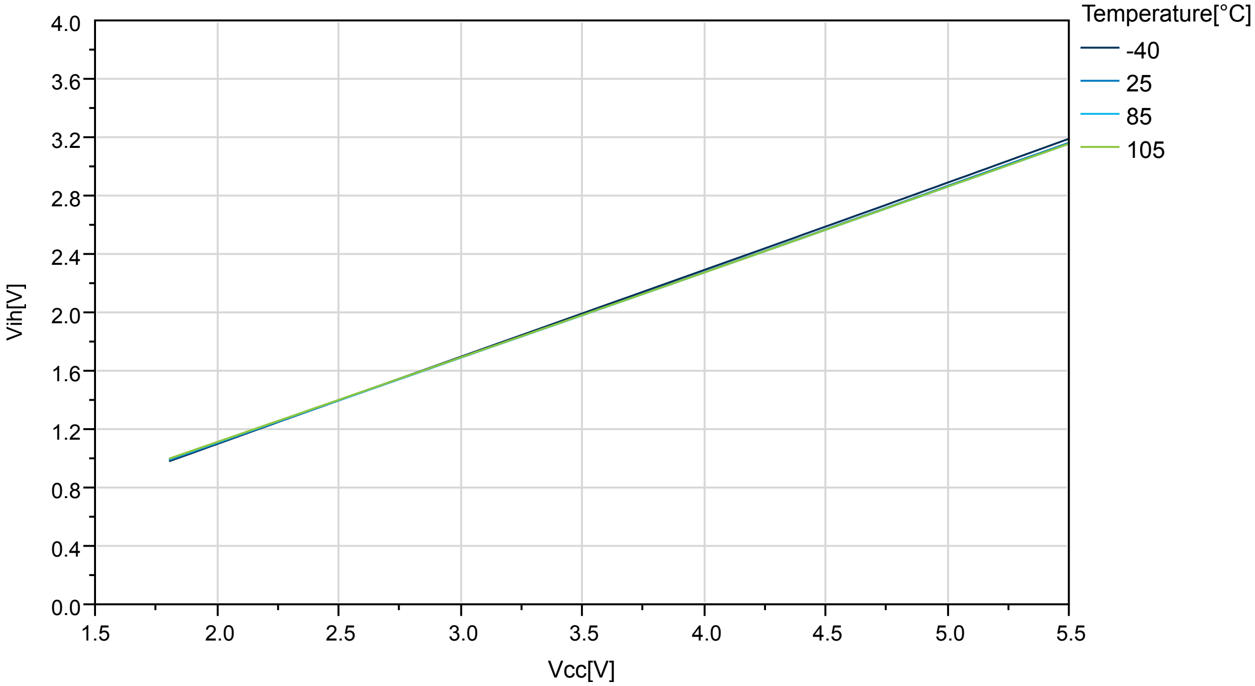


Figure 33-27. ATmega324PB I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin read as ‘0’)

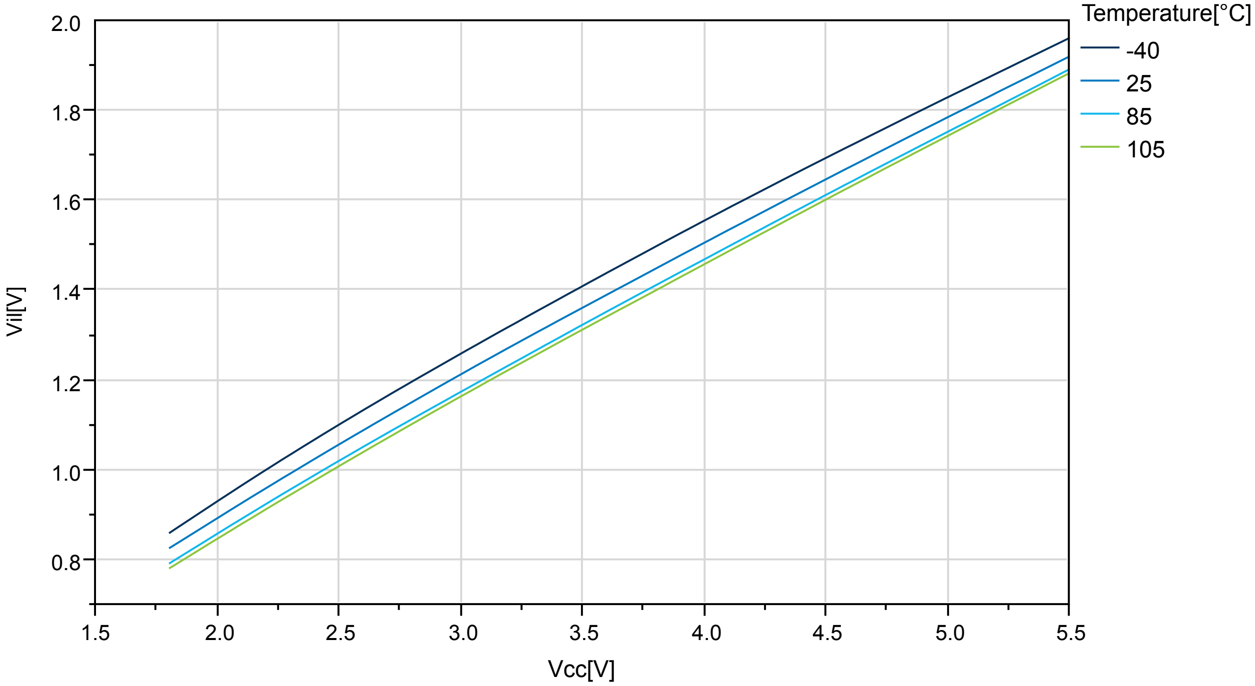


Figure 33-28. ATmega324PB I/O Pin Input Hysteresis vs. VCC

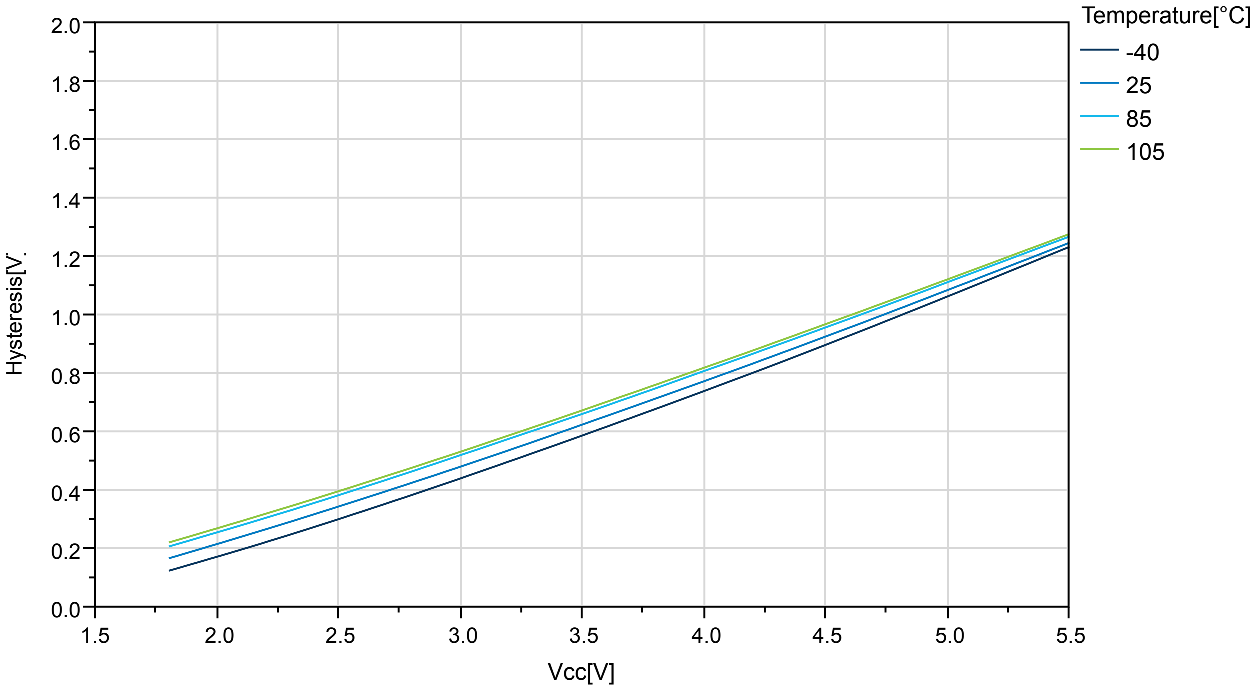


Figure 33-29. ATmega324PB Reset Input Threshold Voltage vs. VCC (VIH, I/O Pin read as '1')

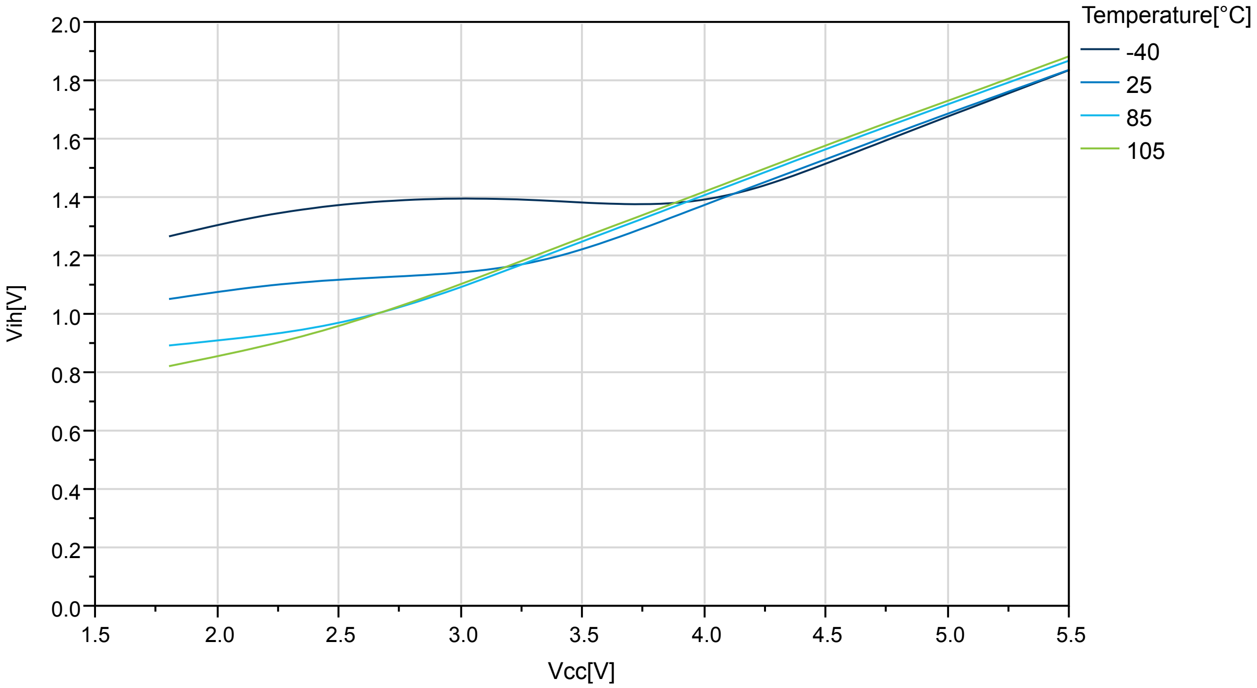


Figure 33-30. ATmega324PB Reset Input Threshold Voltage vs. VCC (VIL, I/O Pin read as ‘0’)

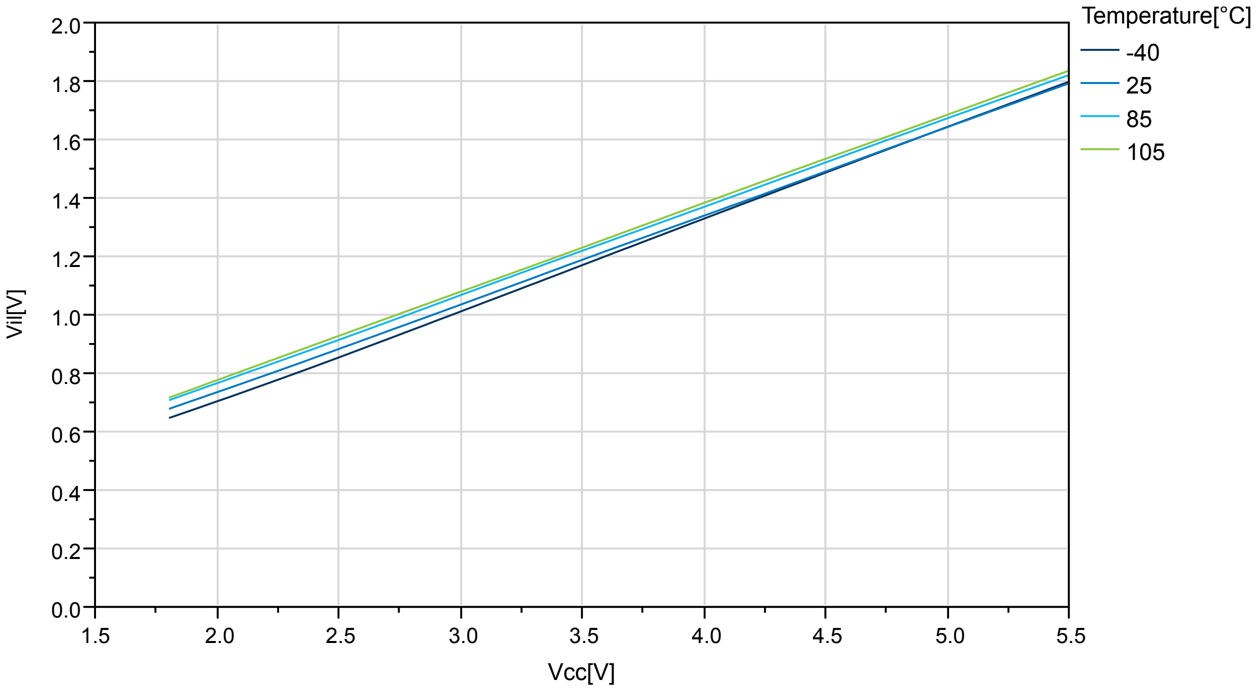
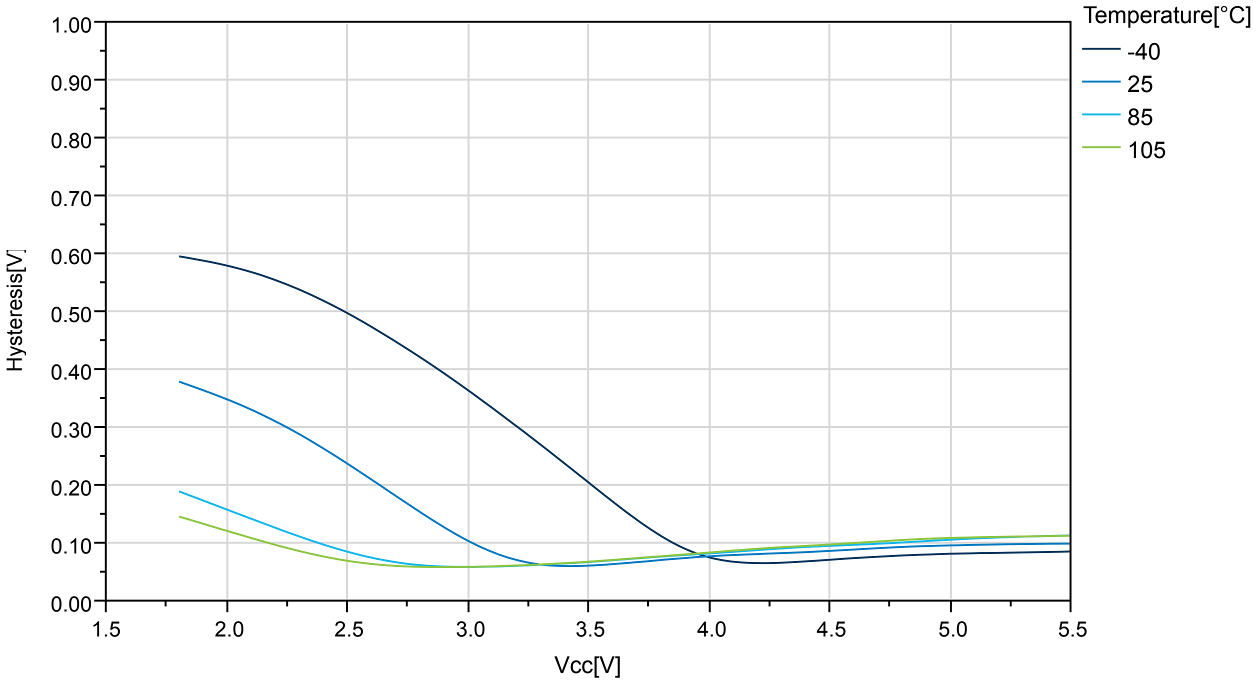


Figure 33-31. ATmega324PB Reset Pin Input Hysteresis vs. VCC



33.8 BOD Threshold

Figure 33-32. ATmega324PB: BOD Thresholds vs. Temperature (BODLEVEL is 1.8V)

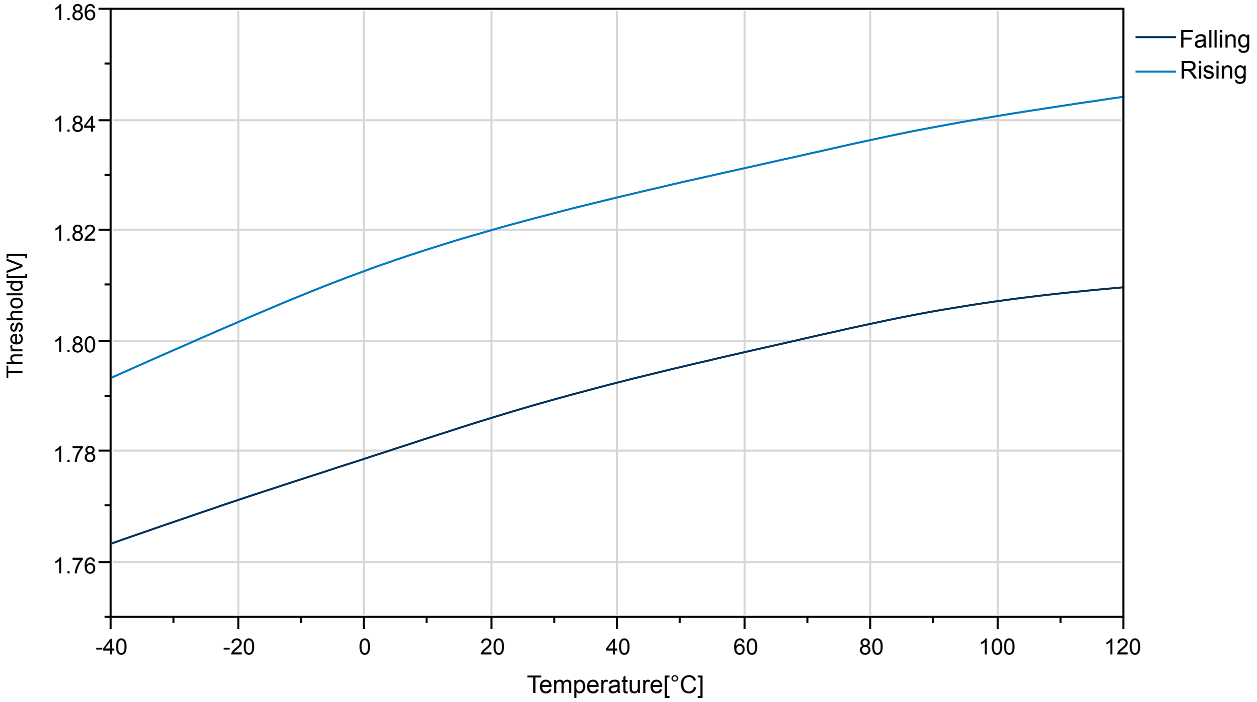


Figure 33-33. ATmega324PB: BOD Thresholds vs. Temperature (BODLEVEL is 2.7V)

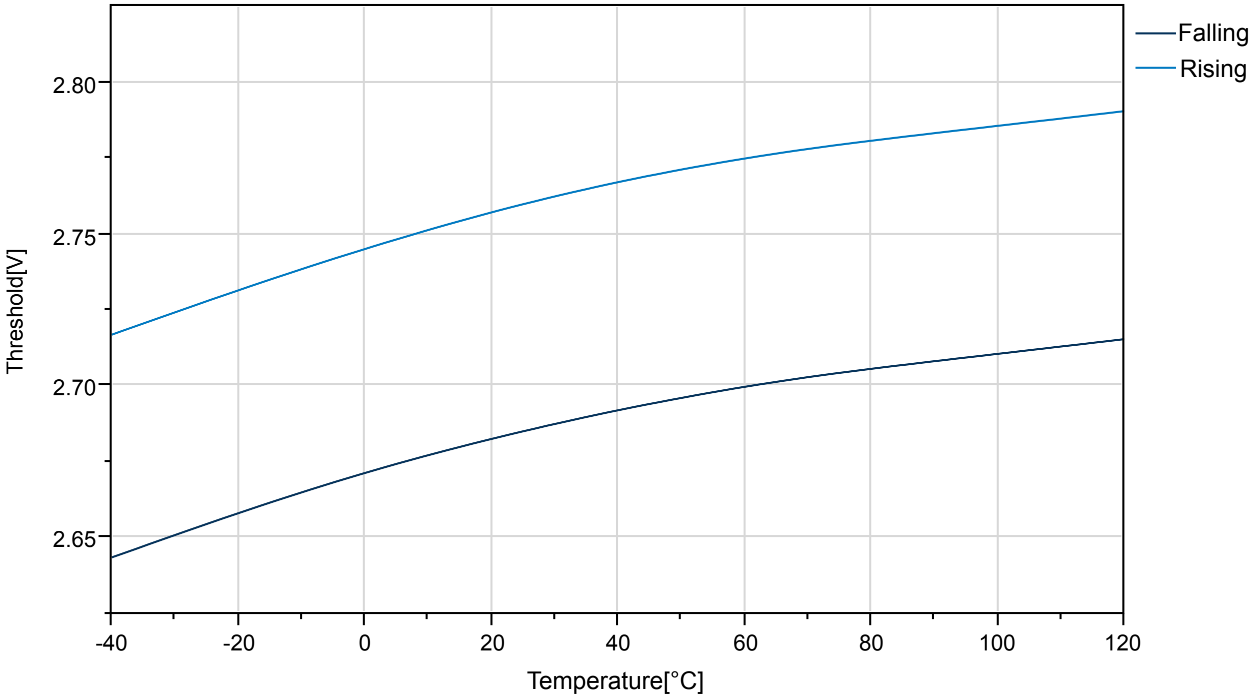


Figure 33-34. ATmega324PB: BOD Thresholds vs. Temperature (BODLEVEL is 4.3V)

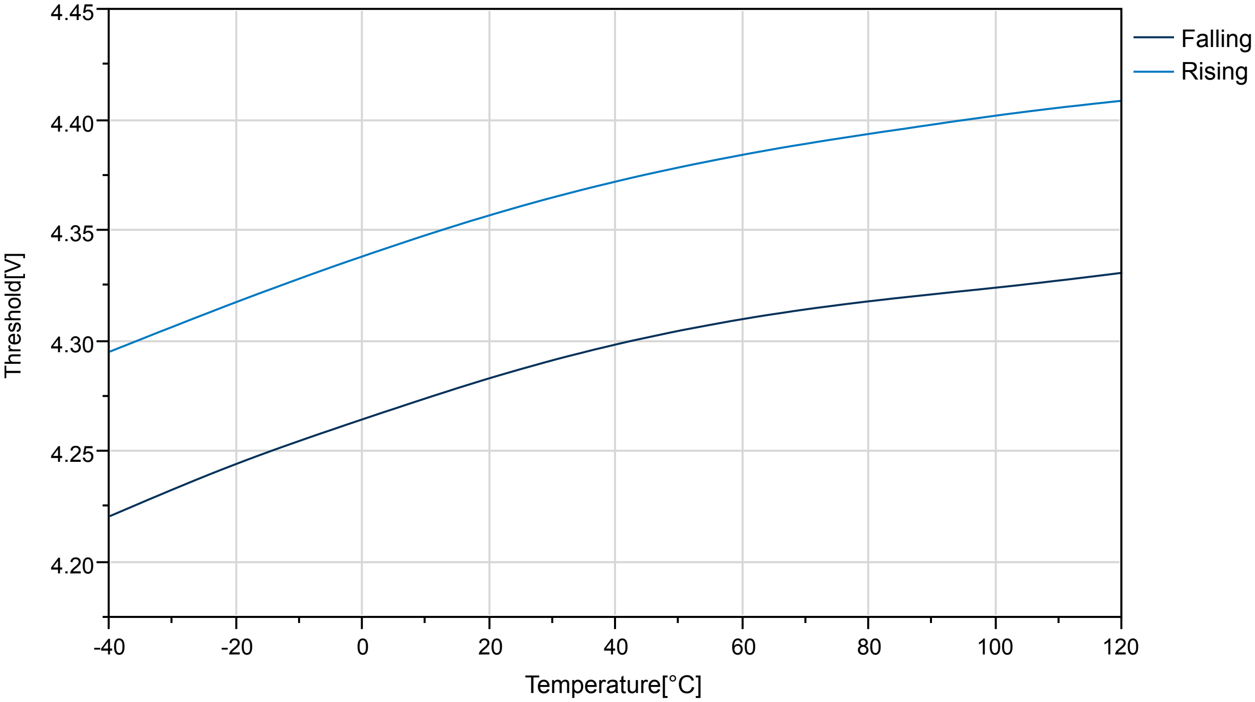


Figure 33-35. ATmega324PB: Calibrated Bandgap Voltage vs. Temperature

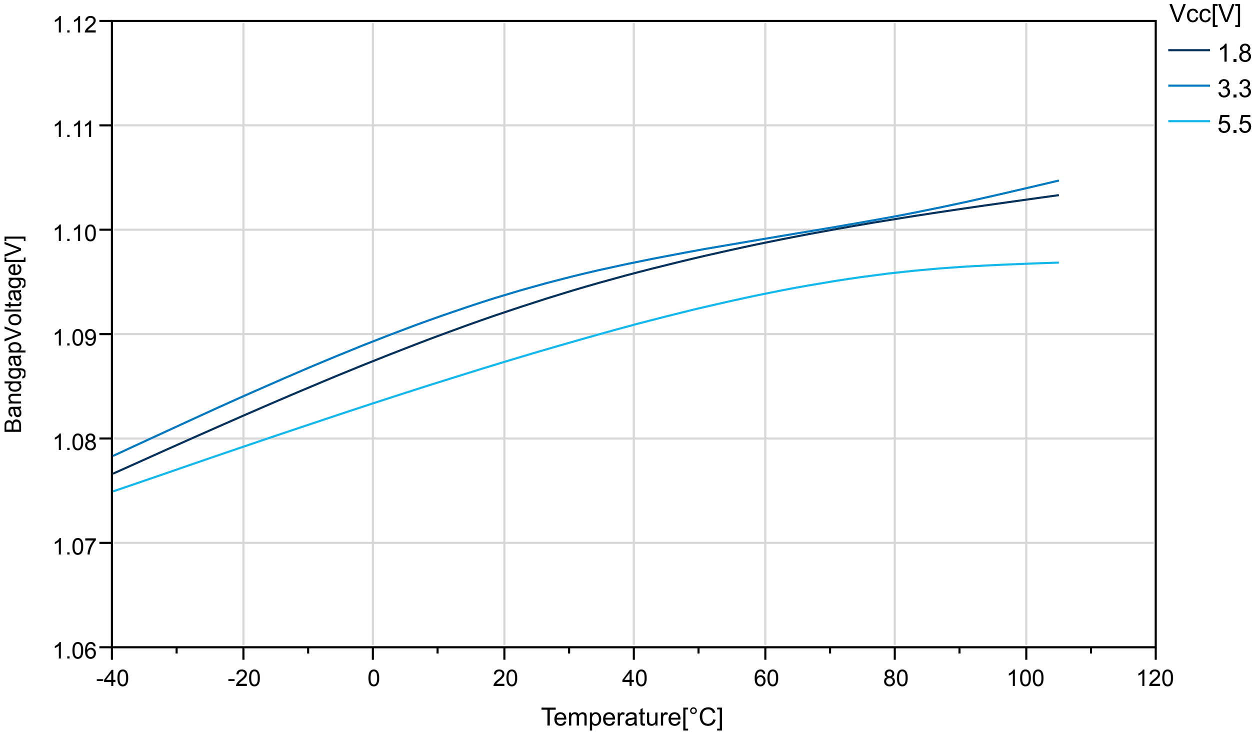


Figure 33-36. ATmega324PB: Calibrated Bandgap Voltage vs. V_{cc}

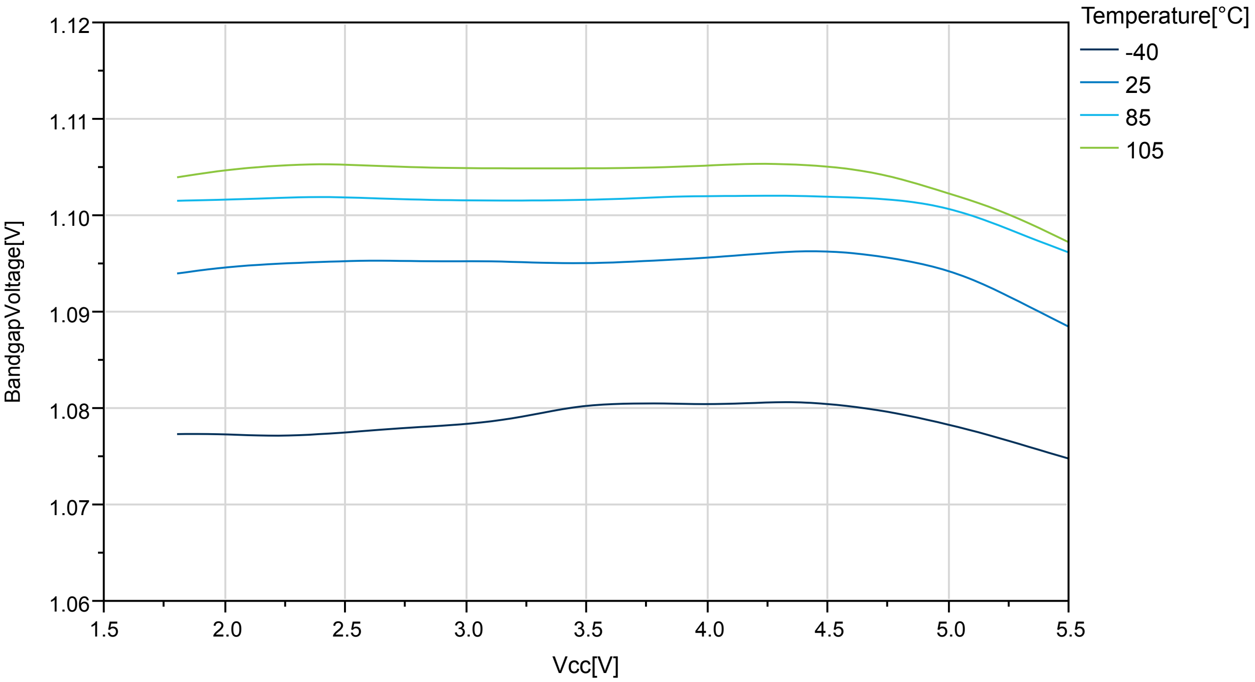
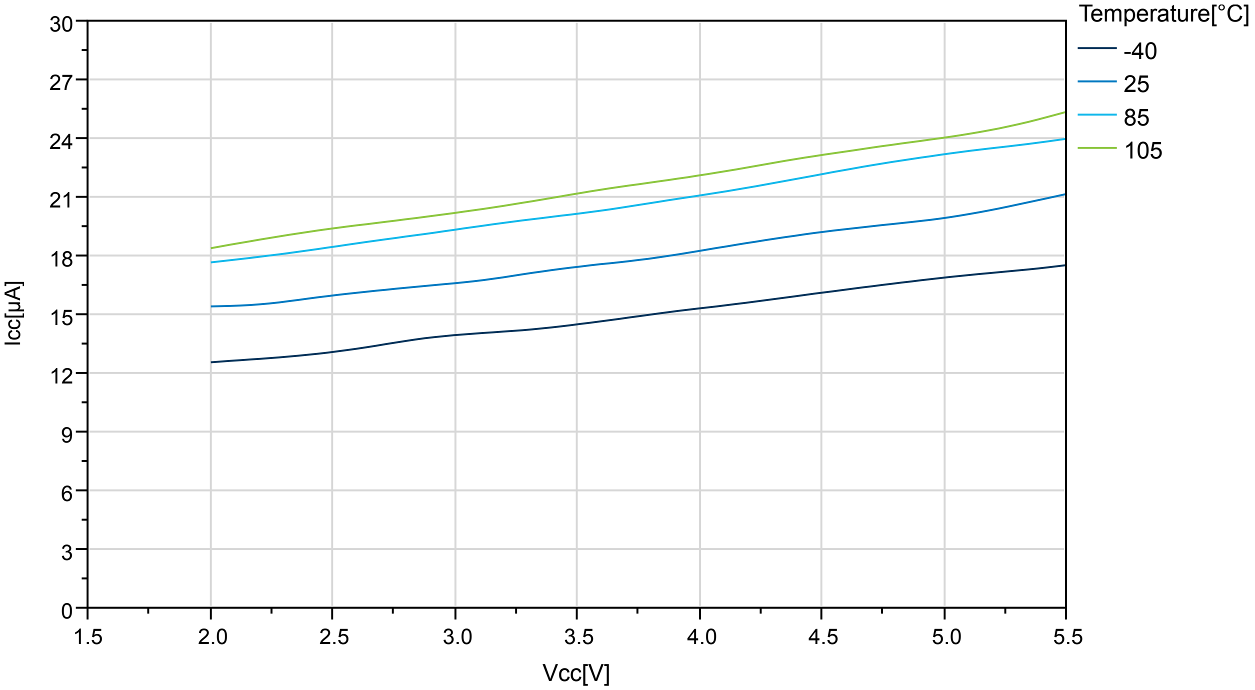


Figure 33-37. ATmega324PB: Brownout Detector Current vs. V_{cc}



33.9 Analog Comparator Offset

Figure 33-38. ATmega324PB: AC Offset vs. Common Voltage (V_{CC} = 1.8V)

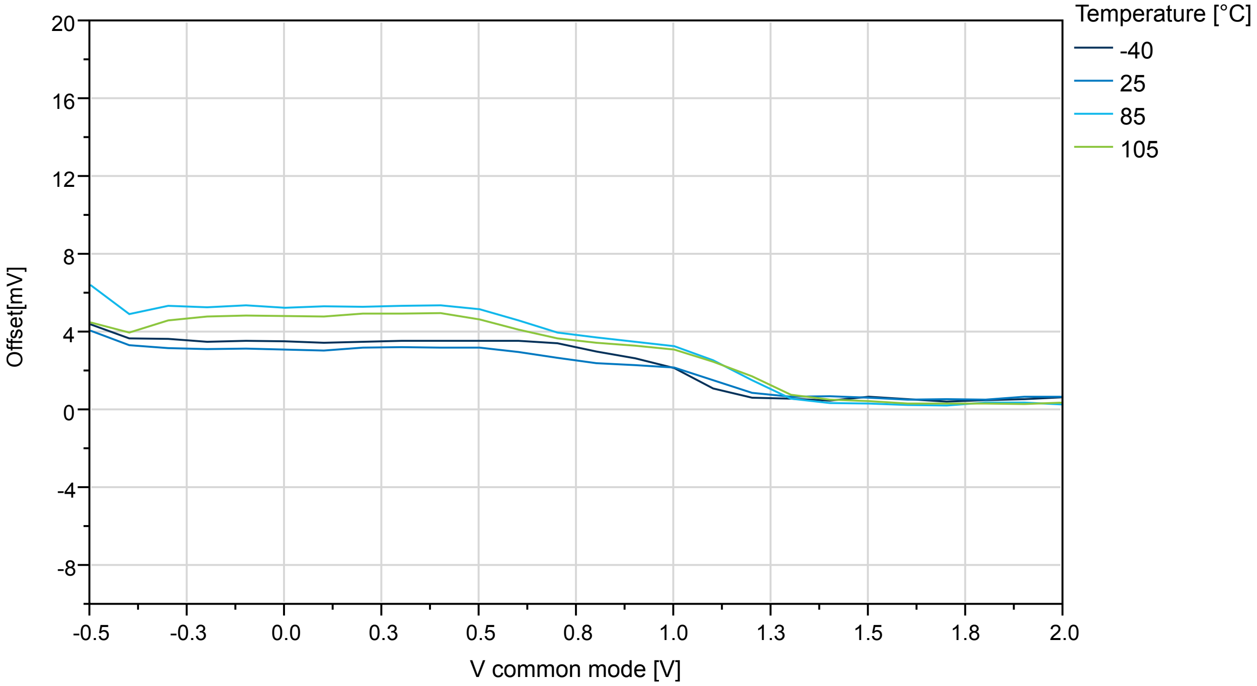


Figure 33-39. ATmega324PB: AC Offset vs. Common Voltage (V_{CC} = 3.0V)

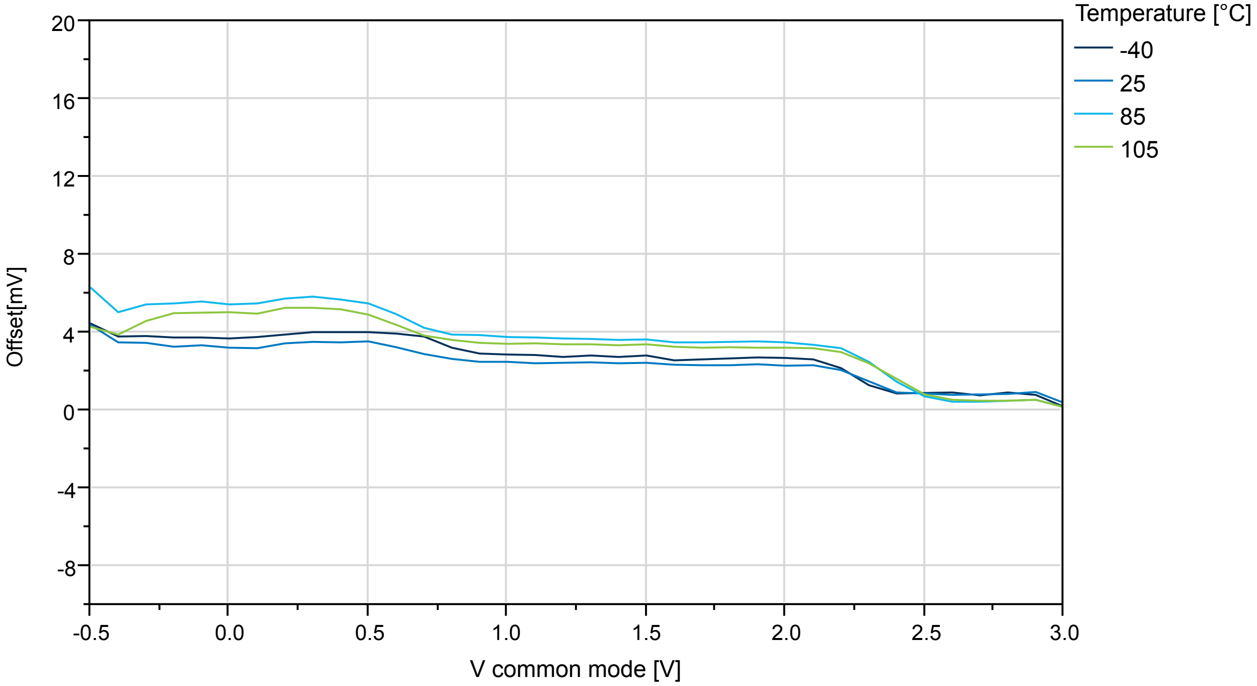
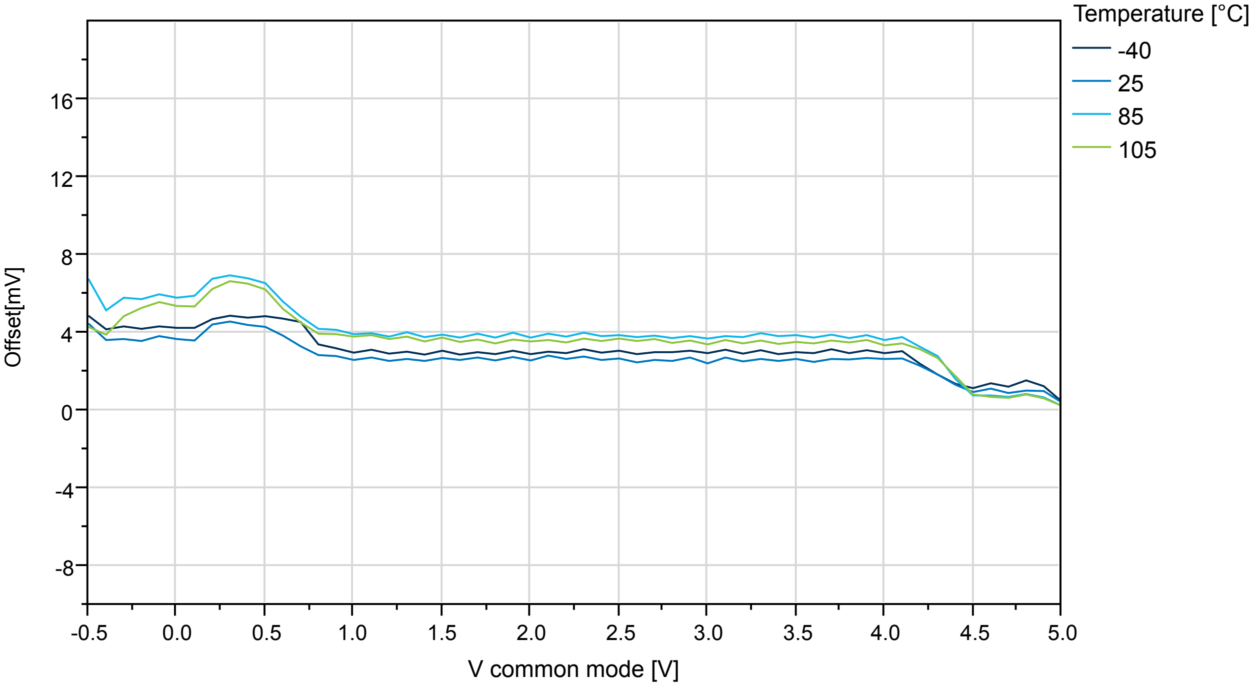


Figure 33-40. ATmega324PB: AC Offset vs. Common Voltage (V_{CC} = 5.0V)



33.10 Internal Oscillator Speed

Figure 33-41. ATmega324PB: Watchdog Oscillator Frequency vs. Temperature

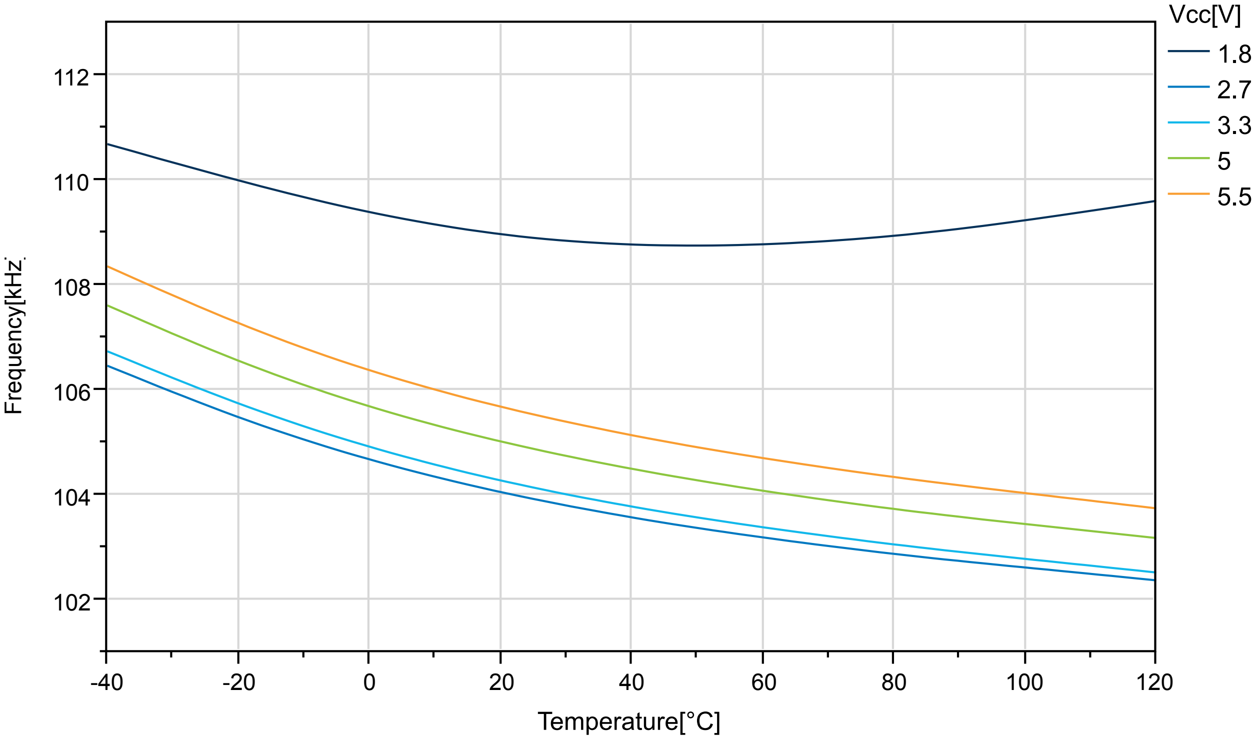


Figure 33-42. ATmega324PB: Watchdog Oscillator Frequency vs. V_{CC}

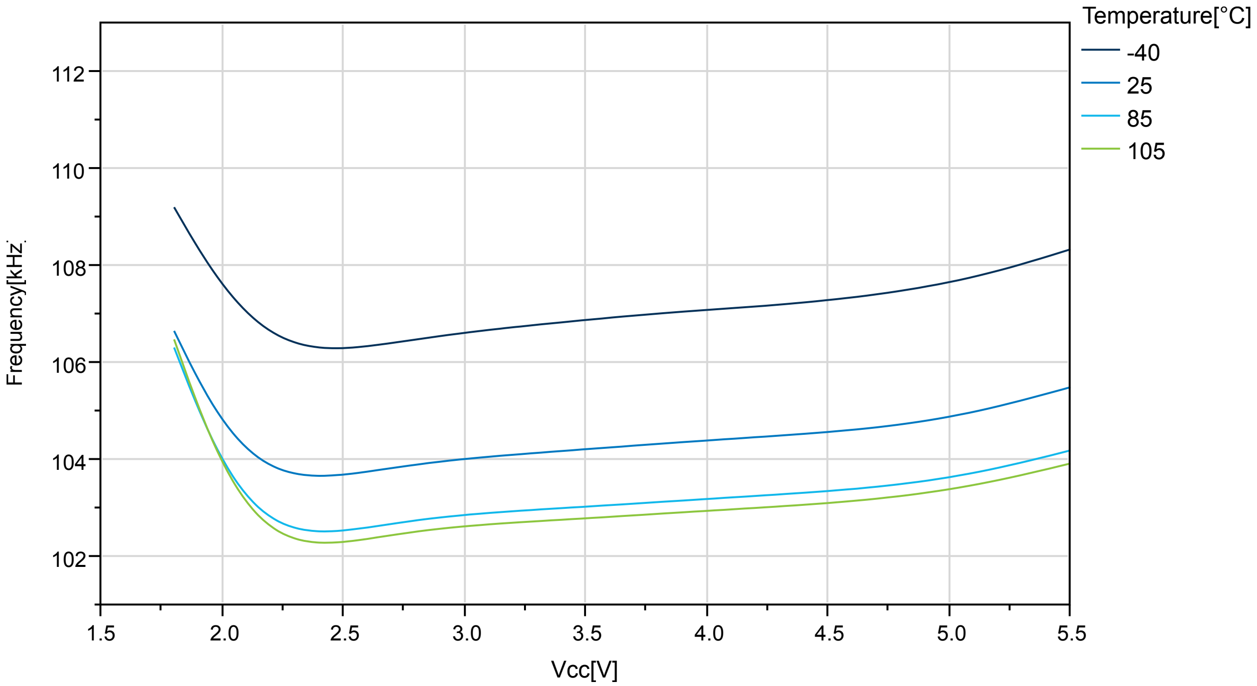


Figure 33-43. ATmega324PB: Calibrated 8MHz RC Oscillator Frequency vs. V_{CC}

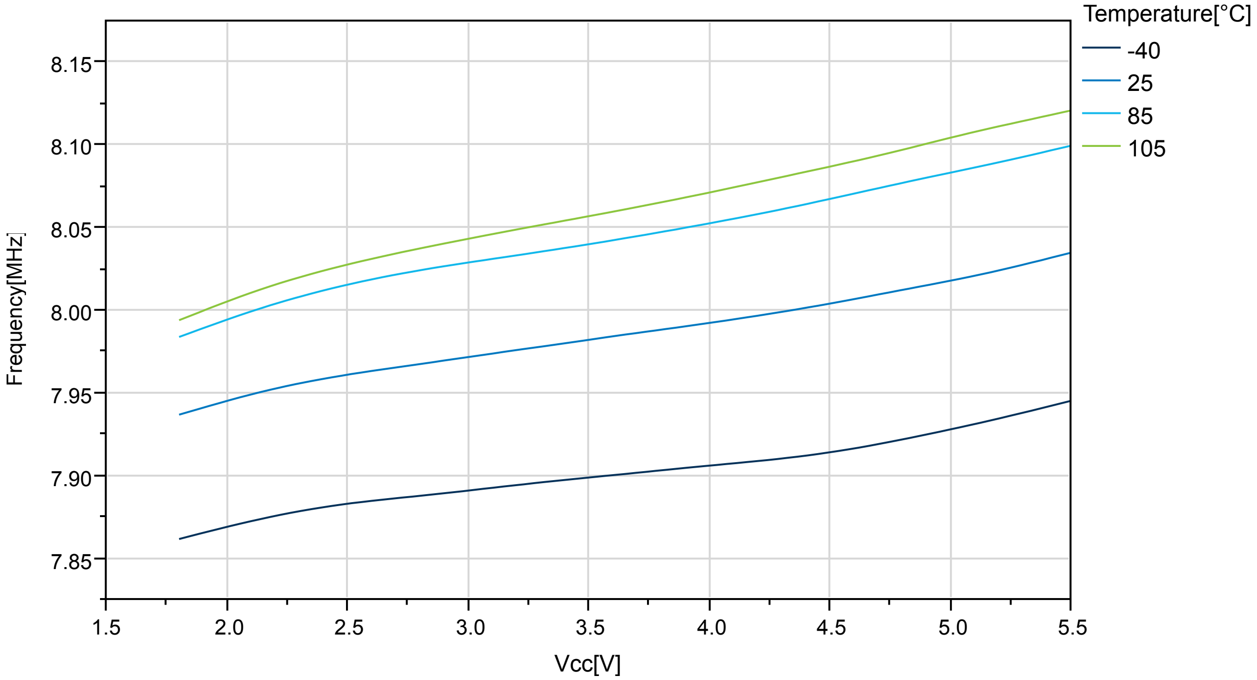


Figure 33-44. ATmega324PB: Calibrated 8MHz RC Oscillator Frequency vs. Temperature

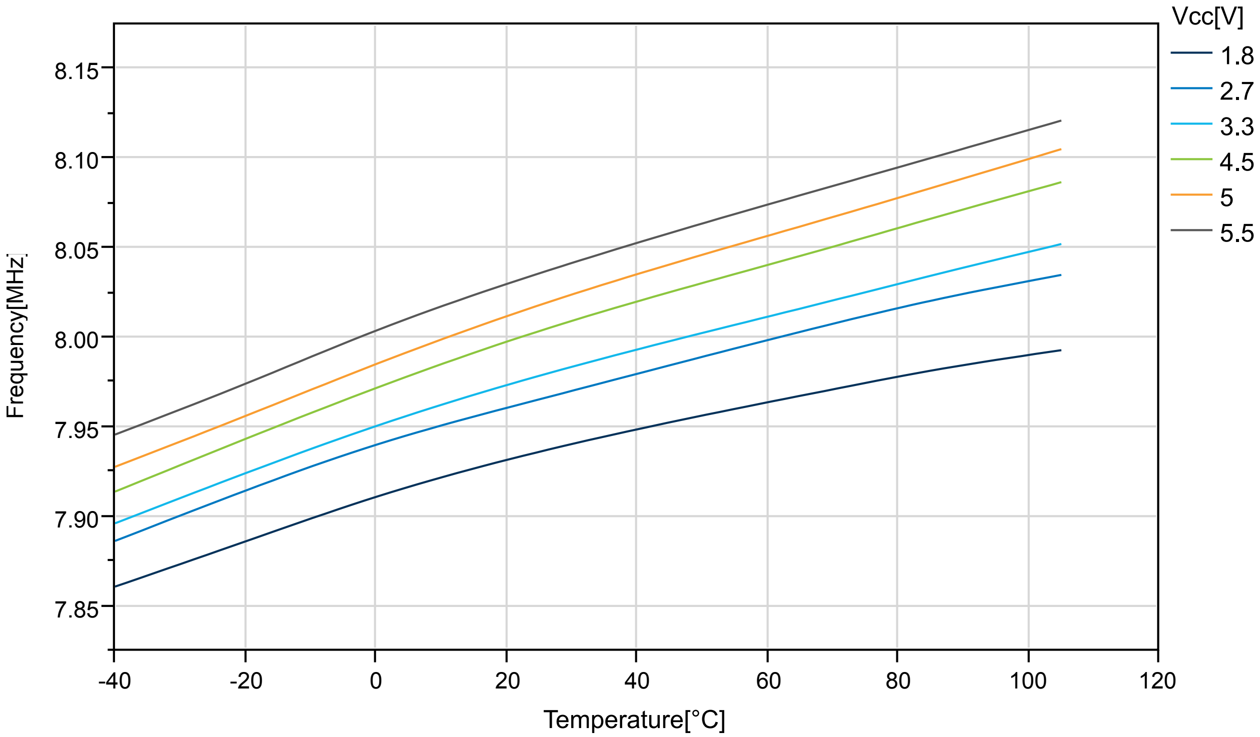


Figure 33-45. ATmega324PB: Calibrated 8MHz RC Oscillator Frequency vs. OSCCAL Value

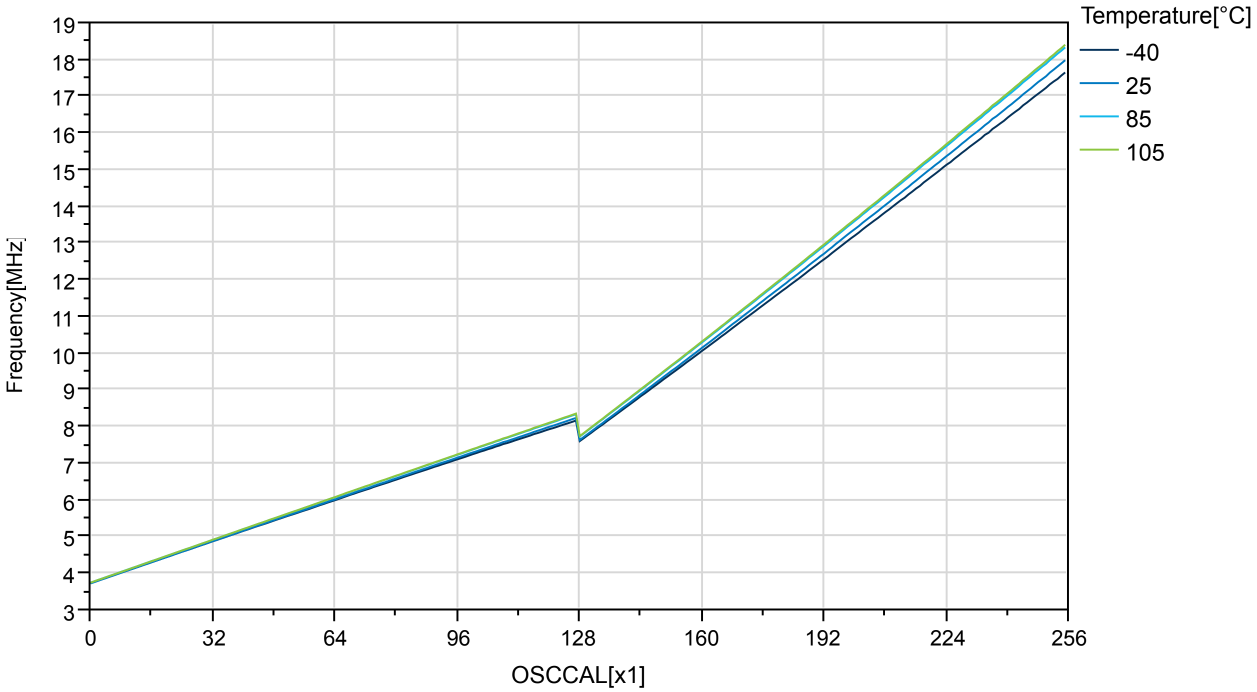
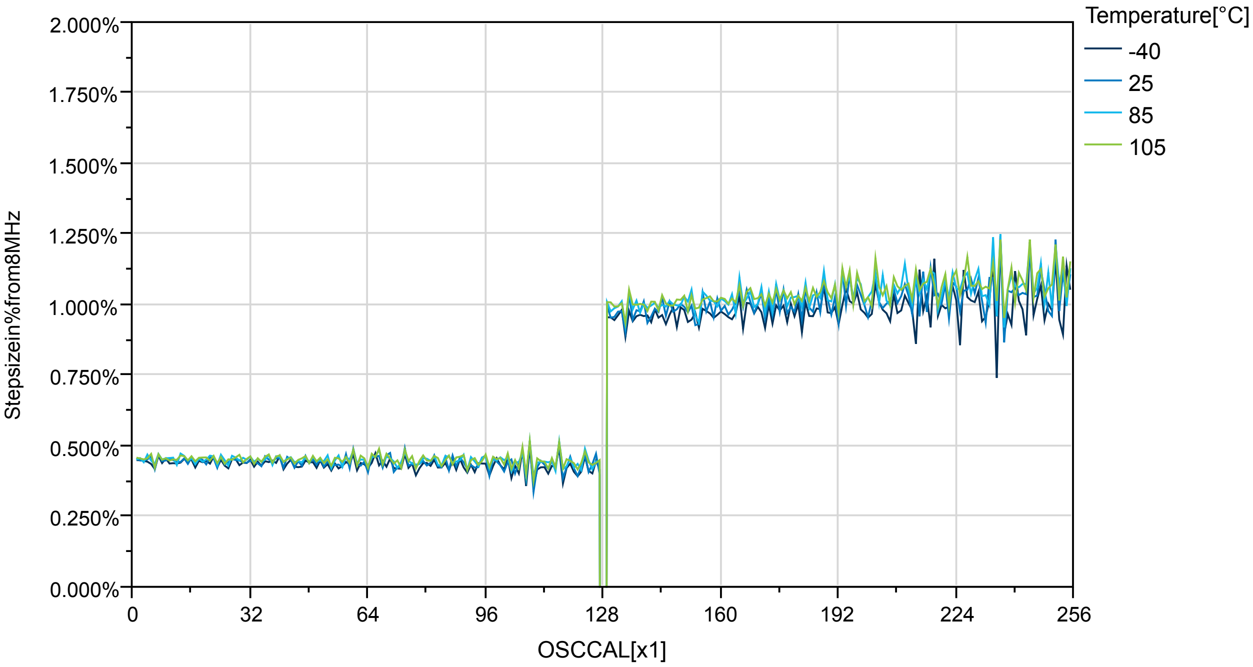


Figure 33-46. ATmega324PB: OSCCAL Value Step Size in %. Base Frequency = 8.0MHz



33.11 Current Consumption of Peripheral Units

Figure 33-47. ATmega324PB: ADC Current vs. V_{cc} (AREF = AV_{cc}), Conversion @50kHz

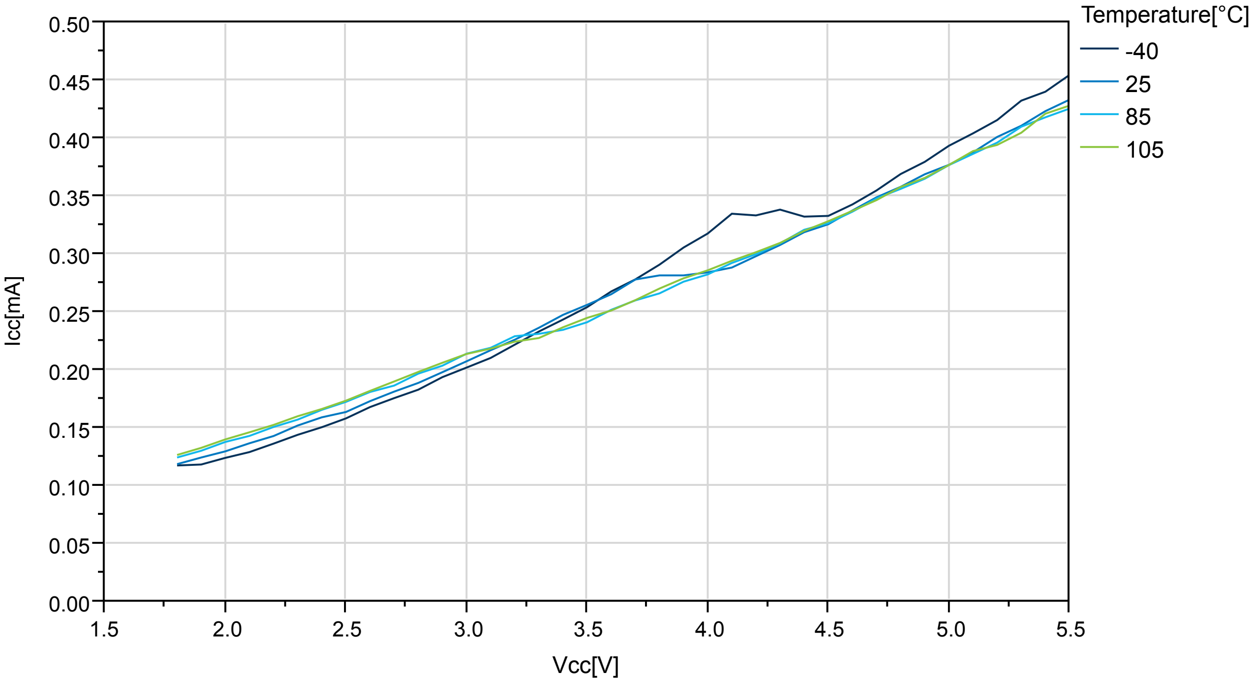


Figure 33-48. ATmega324PB: ADC Current vs. V_{cc} (AREF = AV_{cc}), No Conversion

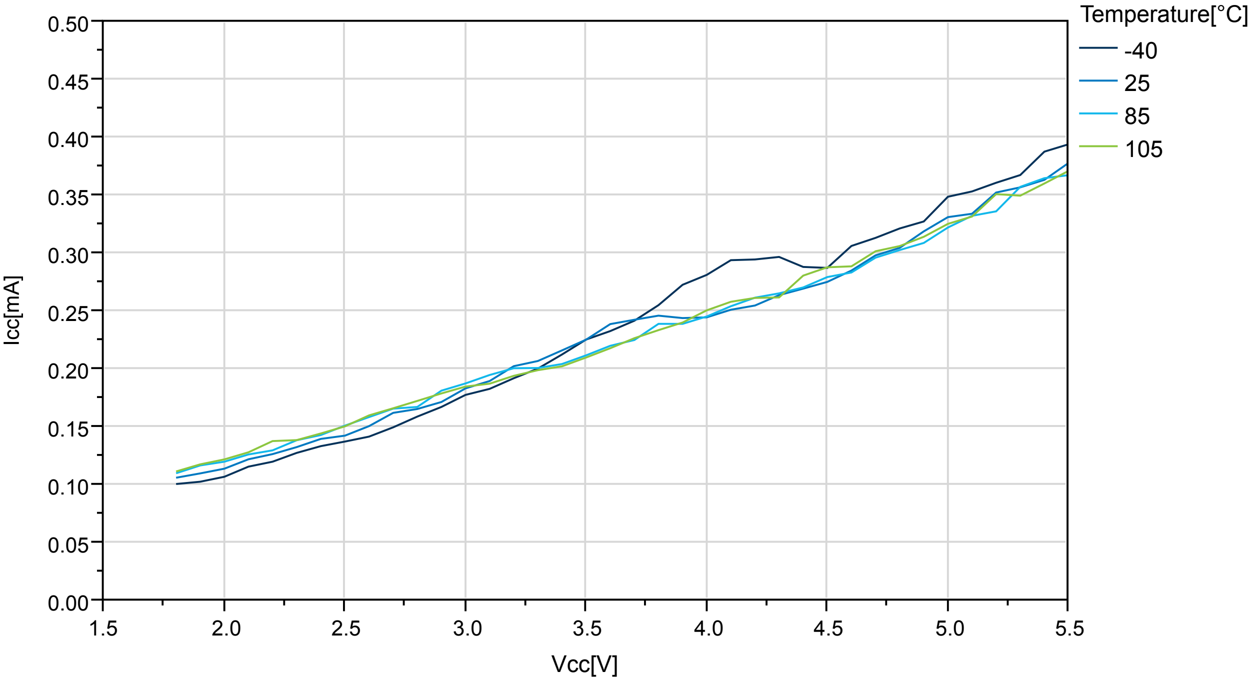


Figure 33-49. ATmega324PB: Analog Comparator Current vs. V_{cc}

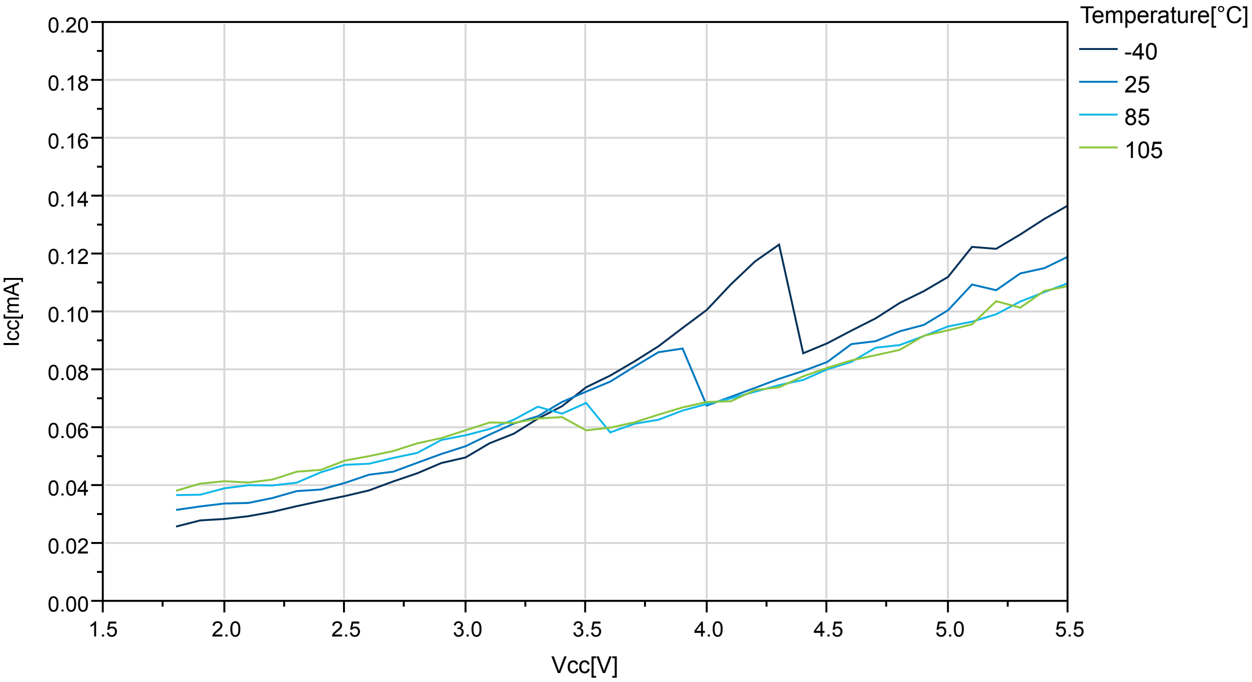


Figure 33-50. ATmega324PB: AREF External Reference Current vs. V_{CC}

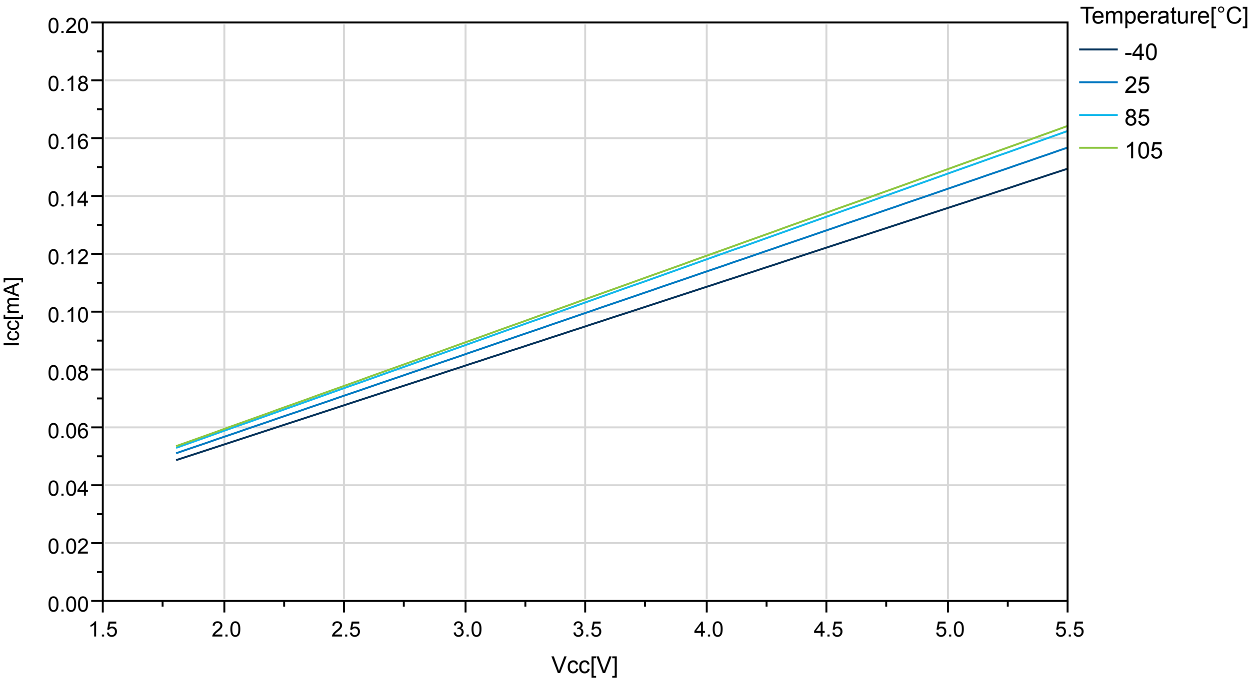
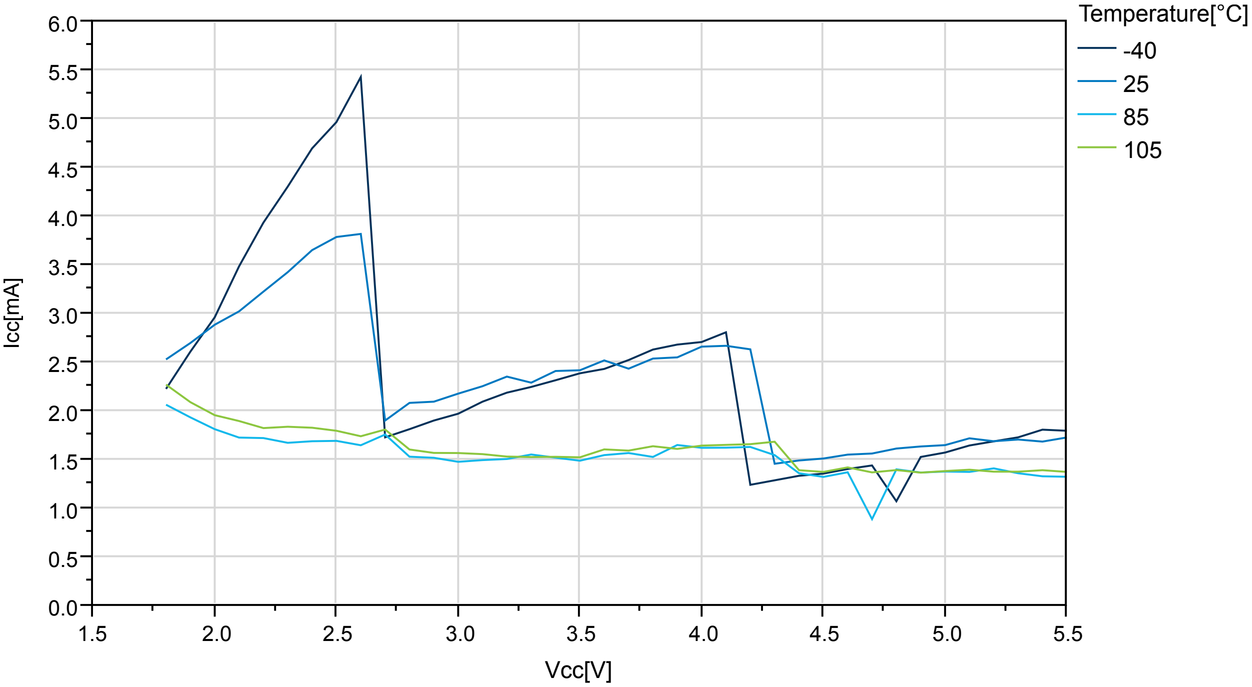


Figure 33-51. ATmega324PB: Programming Current vs. V_{CC}



33.12 Current Consumption in Reset and Reset Pulse Width

Figure 33-52. ATmega324PB: Reset Supply Current vs. Low Frequency (0.1MHz - 1.0MHz)

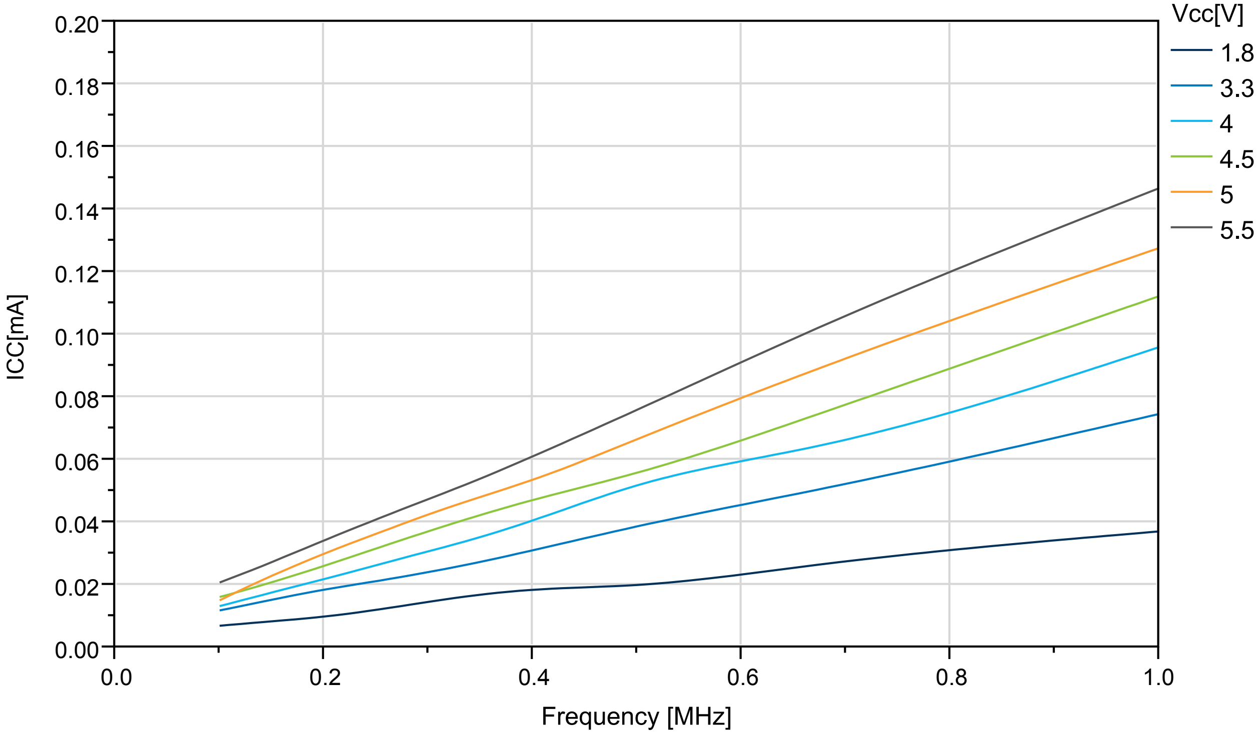


Figure 33-53. ATmega324PB: Reset Supply Current vs. Frequency (1MHz - 20MHz)

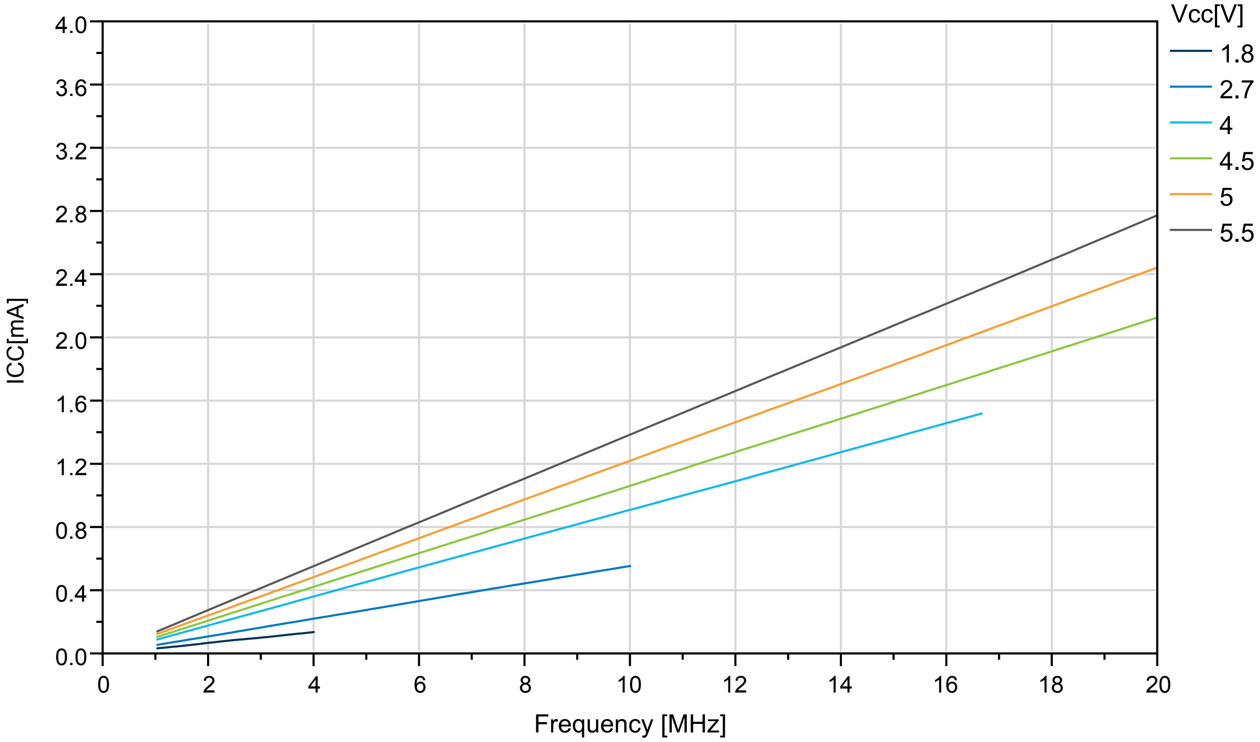


Figure 33-54. ATmega324PB: Reset Supply Current vs. V_{CC} (Excluding current through Reset Pullup)

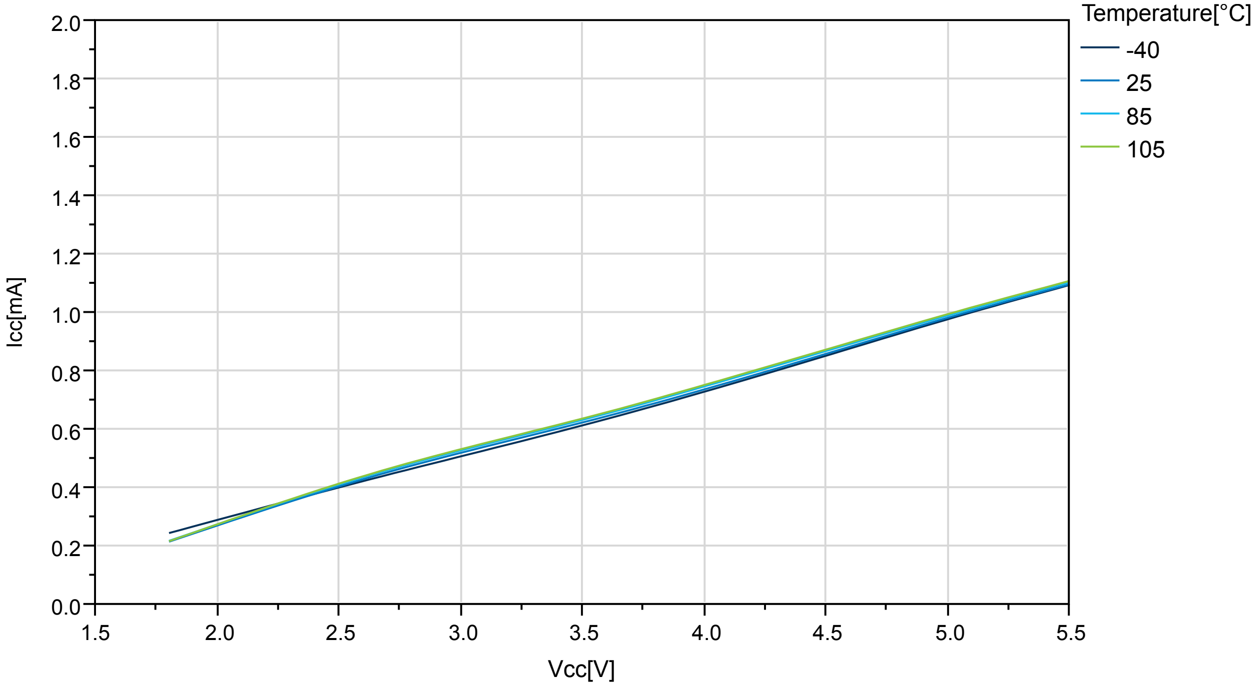
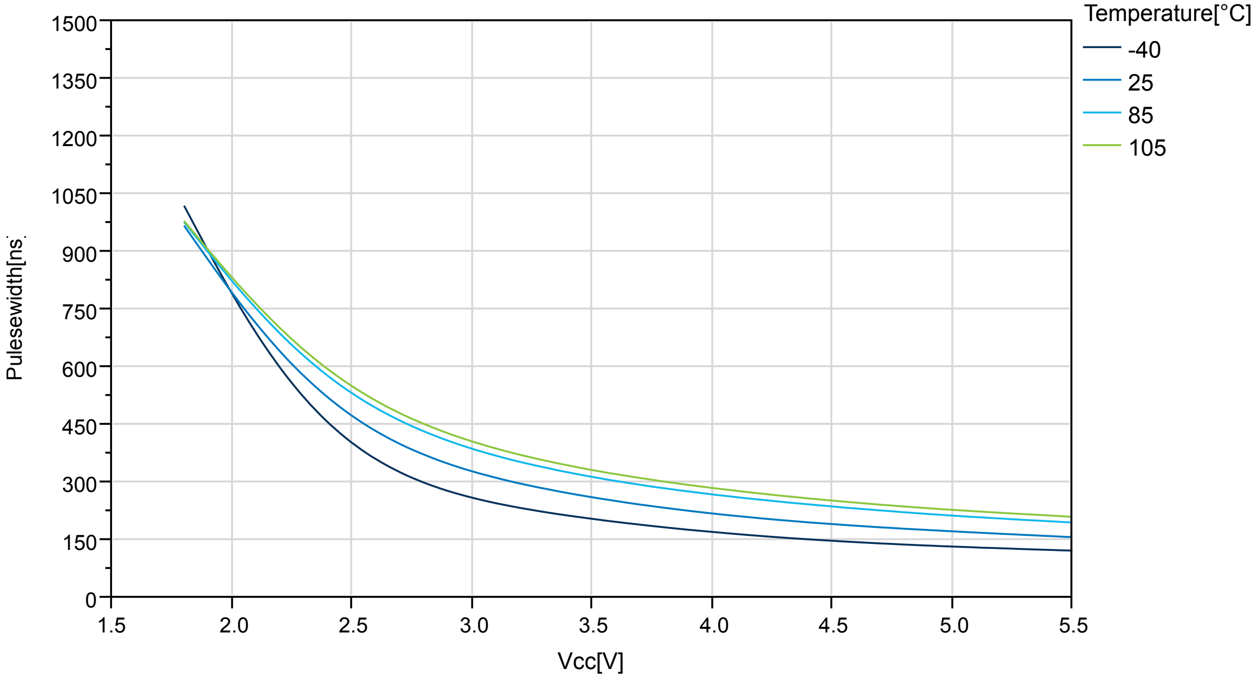


Figure 33-55. ATmega324PB: Minimum Reset Pulse Width vs. V_{CC}



34. Register Summary

Offset	Name	Bit Pos.								
0x20	PINA	7:0	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
0x21	DDRA	7:0	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
0x22	PORTA	7:0	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
0x23	PINB	7:0	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x24	DDRB	7:0	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
0x25	PORTB	7:0	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x26	PINC	7:0	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x27	DDRC	7:0	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
0x28	PORTC	7:0	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x29	PIND	7:0	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x2A	DDRD	7:0	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
0x2B	PORTD	7:0	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x2C	PINE	7:0		PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
0x2D	DDRE	7:0		DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
0x2E	PORTE	7:0		PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
0x2F ... 0x34	Reserved									
0x35	TIFR0	7:0						OCF0B	OCF0A	TOV0
0x36	TIFR1	7:0			ICF1			OCF1B	OCF1A	TOV1
0x37	TIFR2	7:0						OCF2B	OCF2A	TOV2
0x38	TIFR3	7:0			ICF3			OCF3B	OCF3A	TOV3
0x39	TIFR4	7:0			ICF4			OCF4B	OCF4A	TOV4
0x3A	Reserved									
0x3B	PCIFR	7:0				PCIF4	PCIF3	PCIF2	PCIF1	PCIF0
0x3C	EIFR	7:0						INTF2	INTF1	INTF0
0x3D	EIMSK	7:0						INT2	INT1	INT0
0x3E	GPIOR0	7:0	GPIOR0[7:0]							
0x3F	EECR	7:0			EEPM[1:0]		EERIE	EEMPE	EEPE	EERE
0x40	EEDR	7:0	EEDR[7:0]							
0x41	EEARL and EEARH	7:0	EEAR[7:0]							
0x42		15:8					EEAR[11:8]			
0x43	GTCCR	7:0	TSM						PSRASY	PSRSYNC
0x44	TCCR0A	7:0	COM0A[1:0]		COM0B [1:0]				WGM0[1:0]	
0x45	TCCR0B	7:0	FOC0A	FOC0B			WGM0 [2]	CS0[2:0]		
0x46	TCNT0	7:0	TCNT0[7:0]							
0x47	OCR0A	7:0	OCR0A[7:0]							
0x48	OCR0B	7:0	OCR0B[7:0]							
0x49	Reserved									
0x4A	GPIOR1	7:0	GPIOR1[7:0]							
0x4B	GPIOR2	7:0	GPIOR2[7:0]							
0x4C	SPCR0	7:0	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR0 [1:0]	
0x4D	SPSR0	7:0	SPIF0	WCOL0						SPI2X0
0x4E	SPDR0	7:0	SPID[7:0]							

Offset	Name	Bit Pos.								
0x4F	ACSRB	7:0								ACOE
0x50	ACSR	7:0	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS [1:0]	
0x51	OCDR	7:0	IDRD/OCDR7	OCDR 6	OCDR 5	OCDR 4	OCDR 3	OCDR 2	OCDR 1	OCDR 0
0x52	Reserved									
0x53	SMCR	7:0					SM[2:0]			SE
0x54	MCUSR	7:0				JTRF	WDRF	BORF	EXTRF	PORF
0x55	MCUCR	7:0	JTD	BODS	BODSE	PUD			IVSEL	IVCE
0x56	Reserved									
0x57	SPMCSR	7:0	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
0x58 ... 0x5C	Reserved									
0x5D	SPL and SPH	7:0	SP[7:0]							
0x5E		15:8					SP[11:8]			
0x5F	SREG	7:0	I	T	H	S	V	N	Z	C
0x60	WDTCSR	7:0	WDIF	WDIE	WDP[3]	WDCE	WDE	WDP[2:0]		
0x61	CLKPR	7:0	CLKPCE				CLKPS [3:0]			
0x62	XFDCSR	7:0							XFDIF	XFDIE
0x63	PRR2	7:0					PRPTC	PRUSART2	PRSPI1	PRTWI1
0x64	PRR0	7:0	PRTWI0	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI0	PRUSART0	PRADC
0x65	PRR1	7:0							PRTIM4	PRTIM3
0x66	OSCCAL	7:0	CAL [7:0]							
0x67	Reserved									
0x68	PCICR	7:0				PCIE4	PCIE3	PCIE2	PCIE1	PCIE0
0x69	EICRA	7:0			ISC2 [1:0]		ISC1 [1:0]		ISC0 [1:0]	
0x6A	Reserved									
0x6B	PCMSK0	7:0	PCINT 7	PCINT 6	PCINT 5	PCINT 4	PCINT 3	PCINT 2	PCINT 1	PCINT 0
0x6C	PCMSK1	7:0	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6D	PCMSK2	7:0	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x6E	TIMSK0	7:0						OCIE0B	OCIE0A	TOIE0
0x6F	TIMSK1	7:0			ICIE1			OCIE1B	OCIE1A	TOIE1
0x70	TIMSK2	7:0						OCIE2B	OCIE2A	TOIE2
0x71	TIMSK3	7:0			ICIE3			OCIE3B	OCIE3A	TOIE3
0x72	TIMSK4	7:0			ICIE4			OCIE4B	OCIE4A	TOIE4
0x73	PCMSK3	7:0	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
0x74	PCMSK4	7:0		PCINT38	PCINT37	PCINT36	PCINT35	PCINT34	PCINT33	PCINT32
0x75 ... 0x77	Reserved									
0x78	ADCL and ADCH	7:0	ADC[7:0]							
0x79		15:8							ADC[9:8]	
0x7A	ADCSRA	7:0	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS [2:0]		
0x7B	ADCSRB	7:0		ACME				ADTS [2:0]		
0x7C	ADMUX	7:0	REFS [1:0]		ADLAR	MUX [4:0]				
0x7D	Reserved									
0x7E	DIDR0	7:0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
0x7F	DIDR1	7:0							AIN1D	AIN0D

Offset	Name	Bit Pos.								
0x80	TCCR1A	7:0	COM1A[1:0]		COM1B[1:0]				WGM1[1:0]	
0x81	TCCR1B	7:0	ICNC1	ICES1		WGM1[3]	WGM1[2]	CS1[2:0]		
0x82	TCCR1C	7:0	FOC1A	FOC1B						
0x83	Reserved									
0x84	TCNT1L and TCNT1H	7:0	TCNT1[7:0]							
0x85		15:8	TCNT1[15:8]							
0x86	ICR1L and ICR1H	7:0	ICR1[7:0]							
0x87		15:8	ICR1[15:8]							
0x88	OCR1AL and OCR1AH	7:0	OCR1A[7:0]							
0x89		15:8	OCR1A[15:8]							
0x8A	OCR1BL and OCR1BH	7:0	OCR1B[7:0]							
0x8B		15:8	OCR1B[15:8]							
0x8C ... 0x8F	Reserved									
0x90	TCCR3A	7:0	COM3A[1:0]		COM3B[1:0]				WGM3[1:0]	
0x91	TCCR3B	7:0	ICNC3	ICES3		WGM3[3]	WGM3[2]	CS3[2:0]		
0x92	TCCR3C	7:0	FOC3A	FOC3B						
0x93	Reserved									
0x94	TCNT3L and TCNT3H	7:0	TCNT3[7:0]							
0x95		15:8	TCNT3[15:8]							
0x96	ICR3L and ICR3H	7:0	ICR3[7:0]							
0x97		15:8	ICR3[15:8]							
0x98	OCR3AL and OCR3AH	7:0	OCR3A[7:0]							
0x99		15:8	OCR3A[15:8]							
0x9A	OCR3BL and OCR3BH	7:0	OCR3B[7:0]							
0x9B		15:8	OCR3B[15:8]							
0x9C ... 0x9F	Reserved									
0xA0	TCCR4A	7:0	COM4A[1:0]		COM4B[1:0]				WGM4[1:0]	
0xA1	TCCR4B	7:0	ICNC4	ICES4		WGM4[3]	WGM4[2]	CS4[2:0]		
0xA2	TCCR4C	7:0	FOC4A	FOC4B						
0xA3	Reserved									
0xA4	TCNT4L and TCNT4H	7:0	TCNT4[7:0]							
0xA5		15:8	TCNT4[15:8]							
0xA6	ICR4L and ICR4H	7:0	ICR4[7:0]							
0xA7		15:8	ICR4[15:8]							
0xA8	OCR4AL and OCR4AH	7:0	OCR4A[7:0]							
0xA9		15:8	OCR4A[15:8]							
0xAA	OCR4BL and OCR4BH	7:0	OCR4B[7:0]							
0xAB		15:8	OCR4B[15:8]							
0xAC	SPCR1	7:0	SPIE1	SPE1	DORD1	MSTR1	CPOL1	CPHA1	SPR1 [1:0]	
0xAD	SPSR1	7:0	SPIF1	WCOL1						SPI2X1
0xAE	SPDR1	7:0	SPID1[7:0]							
0xAF	Reserved									
0xB0	TCCR2A	7:0	COM2A [1:0]		COM2B [1:0]				WGM2 [1:0]	

Offset	Name	Bit Pos.								
0xB1	TCCR2B	7:0	FOC2A	FOC2B			WGM2 [2]	CS2[2:0]		
0xB2	TCNT2	7:0	TCNT2[7:0]							
0xB3	OCR2A	7:0	OCR2A[7:0]							
0xB4	OCR2B	7:0	OCR2B[7:0]							
0xB5	Reserved									
0xB6	ASSR	7:0		EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
0xB7	Reserved									
0xB8	TWBR0	7:0	TWBR [7:0]							
0xB9	TWSR0	7:0	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
0xBA	TWAR0	7:0	TWA[6:0]							TWGCE
0xBB	TWDR0	7:0	TWD[7:0]							
0xBC	TWCR0	7:0	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
0xBD	TWAMR0	7:0	TWAM[6:0]							
0xBE ... 0xBF	Reserved									
0xC0	UCSR0A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xC1	UCSR0B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xC2	UCSR0C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xC3	UCSR0D	7:0	RXSIE	RXS	SFDE					
0xC4	UBRR0L and UBRR0H	7:0	UBRR[7:0]							
0xC5		15:8					UBRR[11:8]			
0xC6	UDR0	7:0	TXB / RXB[7:0]							
0xC7	Reserved									
0xC8	UCSR1A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xC9	UCSR1B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xCA	UCSR1C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xCB	UCSR1D	7:0	RXSIE	RXS	SFDE					
0xCC	UBRR1L and UBRR1H	7:0	UBRR[7:0]							
0xCD		15:8					UBRR[11:8]			
0xCE	UDR1	7:0	TXB / RXB[7:0]							
0xCF	Reserved									
0xD0	UCSR2A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xD1	UCSR2B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xD2	UCSR2C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xD3	UCSR2D	7:0	RXSIE	RXS	SFDE					
0xD4	UBRR2L and UBRR2H	7:0	UBRR[7:0]							
0xD5		15:8					UBRR[11:8]			
0xD6	UDR2	7:0	TXB / RXB[7:0]							
0xD7	Reserved									
0xD8	TWBR1	7:0	TWBR [7:0]							
0xD9	TWSR1	7:0	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
0xDA	TWAR1	7:0	TWA[6:0]							TWGCE
0xDB	TWDR1	7:0	TWD[7:0]							

Offset	Name	Bit Pos.								
0xDC	TWCR1	7:0	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
0xDD	TWAMR1	7:0	TWAM[6:0]							

35. Instruction Set Summary

ARITHMETIC AND LOGIC INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
ADD	Rd, Rr	Add two Registers without Carry	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add two Registers with Carry	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 two Registers with Carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract Constant from Reg with Carry.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd · Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd · K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd · (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd · Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
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					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP(1)	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3

BRANCH INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL(1)	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	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIS	A, b	Skip if Bit in I/O Register is Set	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
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

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C¬Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0...6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3...0)←Rd(7...4),Rd(7...4)¬Rd(3...0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow.	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1

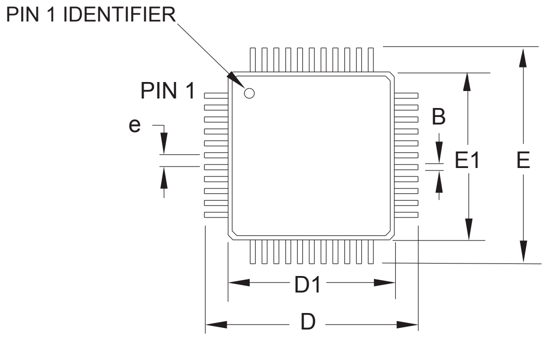
DATA TRANSFER INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2

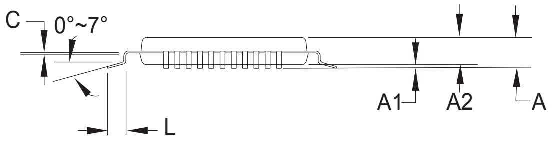
DATA TRANSFER INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
LD	Rd, - Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	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-Increment	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Decrement	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-Increment	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Decrement	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, A	In from I/O Location	Rd ← I/O (A)	None	1
OUT	A, Rr	Out to I/O Location	I/O (A) ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

MCU CONTROL INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
NOP		No Operation	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

36. Packaging Information

36.1 44-pin TQFP





COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	0.37	0.45	
C	0.09	(0.17)	0.20	
L	0.45	0.60	0.75	
e	0.80 TYP			

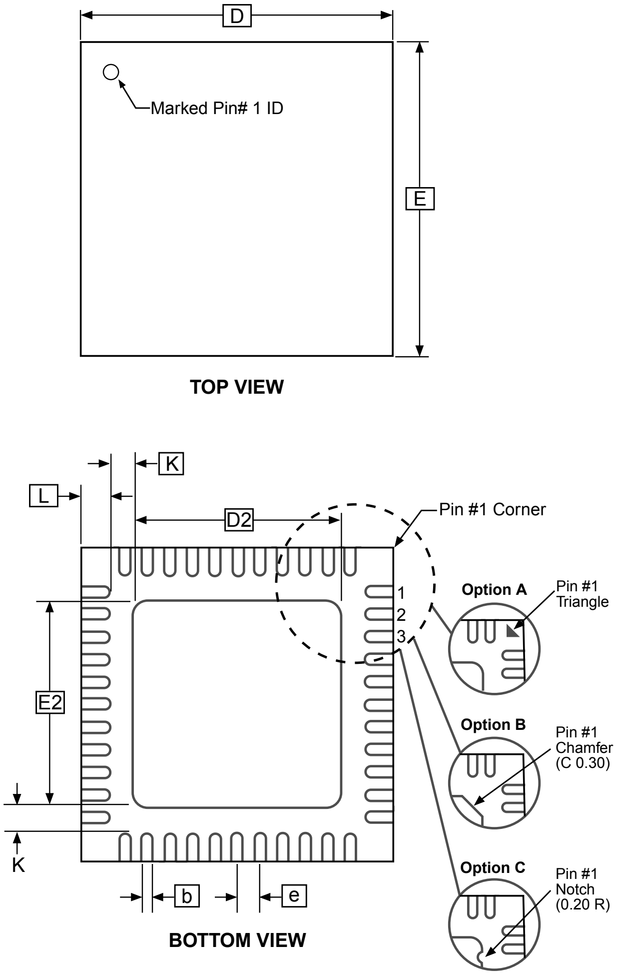
Notes:

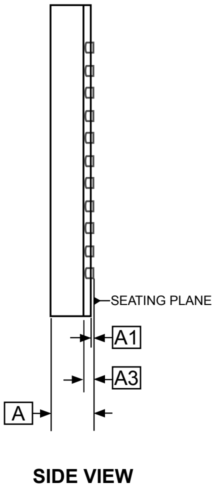
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10mm maximum.

06/02/2014

	TITLE	DRAWING NO.	REV.
Atmel **Package Drawing Contact:** **packagedrawings@atmel.com**	**44A,** 44-lead, 10 x 10mm body size, 1.0mm body thickness, 0.8 mm lead pitch, thin profile plastic quad flat package (TQFP)	44A	C

36.2 44-pin VQFN





Note: JEDEC Standard MO-220, Fig. 1 (SAW Singulation) VKKD-3.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	6.90	7.00	7.10	
D2	5.00	5.20	5.40	
E	6.90	7.00	7.10	
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.59	0.64	0.69	
K	0.20	0.26	0.41	

9/26/08

Package Drawing Contact: avr@atmel.com	**TITLE** **44M1,**44-pad, 7 x 7 x 1.0mm body, lead pitch 0.50mm, 5.20mm exposed pad, thermally enhanced plastic very thin quad flat no lead package (VQFN)	**GPC** ZWS	**DRAWING NO.** 44M1	**REV.** H

37. Errata

37.1 Rev.A-H

Rev. A to H are not sampled.

37.2 Rev. I

Sampled. Not released in production.

37.3 Rev. J - K

1. If the Peripheral Touch Controller (PTC) is enabled in sleep mode, the PTC might stop working.

Description:	If the Peripheral Touch Controller (PTC) is enabled in sleep mode, the PTC might stop working.
Fix/Workaround:	Use QTouch Library version 5.0.8 or later. OR Disable the PTC before entering sleep mode.

38. Datasheet Revision History

Doc Rev.	Date	Comments
DS40001908A	Jan. /2017	• Change of document style. • New Microchip document number. Previous version was Atmel document 42546 rev.C. • OCM - Output Compare Modulator – Update Figure 21-2 • ADC - Analog to Digital Converter – Update bits in ADCSRB • Electrical Characteristics – Updated I_{ACLK} to Min/Max [-100/100 nA] in DC Characteristics – Updated Min and Max values in DC Characteristics – Updated Calibrated Internal RC Oscillator Accuracy – Updated the Max value of SR_{ON} to 9V/ms in System and Reset Characteristics – Added BODLEVEL Fuse Coding in System and Reset Characteristics – Added *Symbols: TUE, INL and DNL* in ADC Characteristics • Typical Characteristics – Update Figure 33-15 – Update Figure 33-50
42546C	Oct. /2016	• In Power Save mode, the main clock is kept running when the PTC is enabled. • OCM - Output Compare Modulator – The share port is PB7. • Typical Characteristics: – Added Supply Current of IO Modules – Plots updated • Add Errata Rev. J - K.
42546B	Sept. /2016	• Ordering Information – Updated the dimensions of 44M1 package to 7 x 7 x 0.9 mm • JTAG Interface and On-chip Debug System – Updated ATmega324PB Boundary-scan Order table • MEMPROG- Memory Programming – Updated the description of BOOTRST in Table 31-6

Doc Rev.	Date	Comments
		- Electrical Characteristics: Updated the content inclunding conditions and values in the following sections – DC Characteristics – Power Consumption – Calibrated Internal RC Oscillator Accuracy – System and Reset Characteristics – ADC Characteristics – Parallel Programming Characteristics - Typical Characteristics: Plots updated
42546A	Nov. /2015	Initial document release.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-1684-5

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra’anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Microchip:

ATMEGA324PB-AU ATMEGA324PB-MU ATMEGA324PB-MUR ATMEGA324PB-AUR ATMEGA324PB-MN ATMEGA324PB-AN ATMEGA324PB-ANR ATMEGA324PB-MNR

MOSCHIP.RU
ВМЕСТЕ МЫ СОЗДАЕМ БУДУЩЕЕ

ПОСТАВКА
ЭЛЕКТРОННЫХ КОМПОНЕНТОВ

многоканальный
+7 495 668 12 70
info@moschip.ru

Общество с ограниченной ответственностью «МосЧип» ИНН 7719860671 / КПП 771901001
Адрес: 105318, г.Москва, ул.Щербаковская д.3, офис 1107

Данный компонент на территории Российской Федерации

Вы можете приобрести в компании MosChip.

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

http://moschip.ru/get-element

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

Офис по работе с юридическими лицами:

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:
moschip.ru
moschip.ru_4
moschip.ru_6
moschip.ru_9