ATmegaS64M1

Rad-Tol 8-bit AVR Microcontroller, 3.3V, 8 MHz with 64 KB Flash, 2 KB EEPROM, 4 KB SRAM, 10-bit ADC, 10-bit DAC, CAN, UART, 12-bit PSC, SPI, 8-bit and 16-bit Timer/Counter with PWM

Introduction

The ATmegaS64M1 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 ATmegaS64M1 achieves throughputs close to 1 MIPS per MHz. This empowers system designers to optimize the device for power consumption versus processing speed.

Features

- High-performance, Low-Power 8-bit AVR® Microcontroller
- Advanced RISC Architecture
  - 131 powerful instructions - most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 1 MIPS throughput per MHz
  - On-chip 2-cycle multiplier
- Data and Nonvolatile Program Memory
  - 64K Bytes flash of in-system programmable program memory
  - 2K Bytes of in-system programmable EEPROM
  - 4K Bytes internal SRAM
  - Write/erase cycles
    - Flash: 10,000 cycles
    - EEPROM: 20,000 cycles
  - Data retention
    - 15 years @ 70°C
    - 10 years @ 125°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 Flash program and EEPROM data security
- On-chip DebugInterface (debugWIRE)
- CAN 2.0A/B with Six Message Objects - ISO16845 certified
- 8-bit UART (supporting LIN 2.1 and 1.3 controller)

- One 12-bit High-speed Power Stage Controller (PSC)
  - Nonoverlapping inverted PWM output pins with flexible dead-time
  - Variable PWM duty cycle and frequency
  - Synchronous update of all PWM registers
  - Auto stop function for emergency event
- Peripheral Features
  - One 8-bit general-purpose timer/counter with separate prescaler, compare mode and capture mode
  - One 16-bit general-purpose timer/counter with separate prescaler, compare mode and capture mode
  - One master/slave SPI serial interface
  - 10-bit ADC
    - Up to 11 single-ended channels and three fully differential ADC channel pairs
    - Programmable gain (5×, 10×, 20×, 40×) on differential channels
    - Internal reference voltage
    - Direct power supply voltage measurement
  - 10-bit DAC for variable voltage reference (comparators, ADC)
  - Four analog comparators with variable threshold detection
  - 100 µA ±2% current source (LIN node identification)
  - Interrupt and wake-up on pin change
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip temperature sensor
- Special Microcontroller Features
  - Low power idle, noise reduction, and power down modes
  - Power on reset and programmable brown-out detection
  - In-system programmable via SPI port
  - High-precision crystal oscillator for CAN operations
  - 8 MHz internal calibrated RC oscillator
  - On-chip PLL for fast PWM
- Operating Range
  - Operating voltage: 3.0 V to 3.6 V
  - Temperature: -55 °C to +125 °C
- 8 MHz Core Speed Grade
- Radiation Tolerance
  - No Single Event Latch-up (SEL) below a LET threshold of 62.5 MeV/mg/cm^2@125 °C
  - Tested up to a Total Ionizing Dose (TID) of 30 KRads(Si) according to MIL-STD-883 Method 1019
- ESD Classification
  - HBM > 4000V (Class 3A)
  - CDM > 1000V (Class IV)
- Packages
  - 32-lead Ceramic Quad Flat package (CQFP) with mass equal to 0.847g
  - 32-lead Plastic Quad Flat package (TQFP) with mass equal to 0.145g

Table of Contents

1. Space Quality Grade

The ATmegaS64M1 has been developed and manufactured according to the most stringent requirements of MIL-PRF-38535 International Standards and Aerospace AEQA0239 specification. This datasheet provides limit values extracted from the results of extensive characterization (versus temperature and voltage). The quality and reliability of the ATmegaS64M1 have been verified during regular product qualification in compliance with MIL-PRF-38535 and MIL-STD-883 standards.

2. Description

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

The ATmegaS64M1 provides the following features: 64Kbytes of In-System Programmable Flash with Read-While-Write capabilities(1), 2Kbytes EEPROM, 4Kbytes SRAM, 27 general-purpose I/O lines, 32 general-purpose working registers, one Motor Power Stage Controller, two flexible Timer/Counters with compare modes and PWM, one UART with HW LIN, an 11-channel 10-bit ADC with two differential input stages with programmable gain, a 10-bit DAC, a programmable Watchdog Timer with internal individual oscillator, an SPI serial port, an On-chip Debug system and four software-selectable power saving modes.

The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI ports, CAN, LIN/UART and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset. The ADC Noise Reduction mode stops the CPU and all I/O modules except the ADC, thus minimizing 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 our 0.35 µm CMOS and high-density nonvolatile memory technology (AT35K4 process). 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 continues 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 ATmegaS64M1 is a powerful microcontroller that provides a highly flexible and cost-effective solution to many embedded control applications in a Space environment.

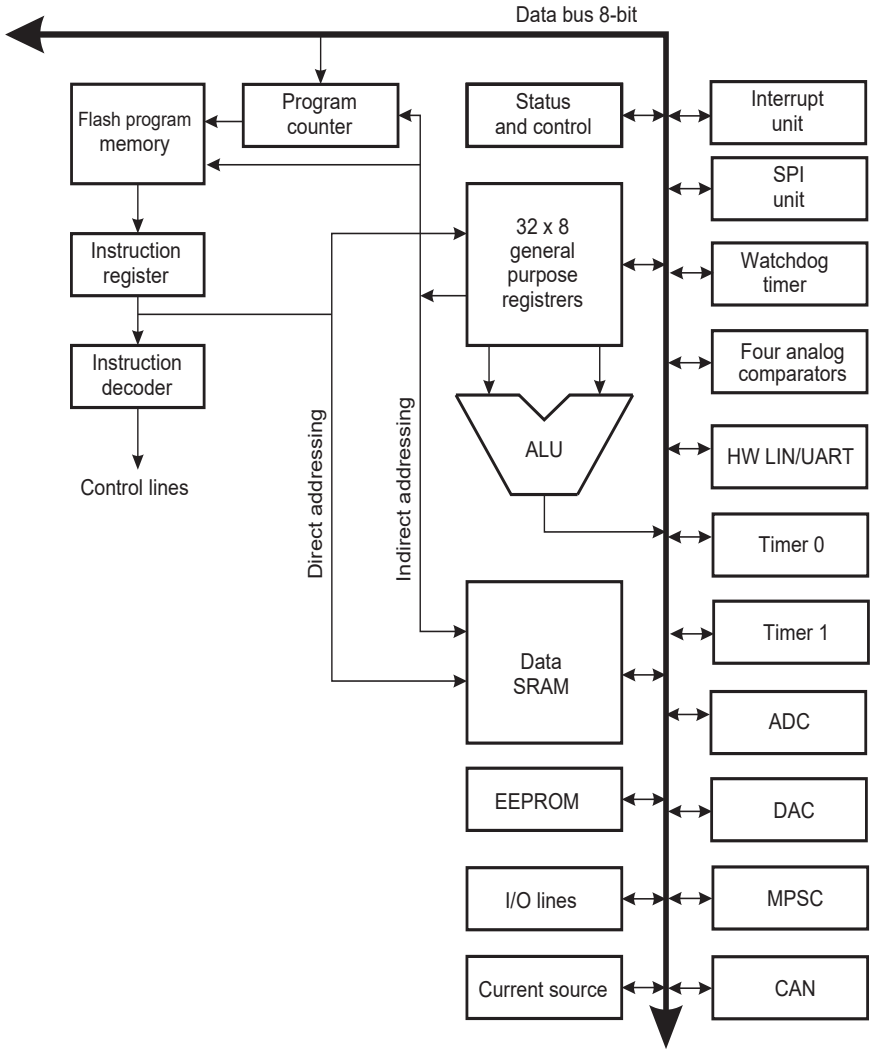
The ATmegaS64M1 is supported by a full suite of program and system development tools including C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Note:

1. Refer to the product radiation report and dedicated application notes.

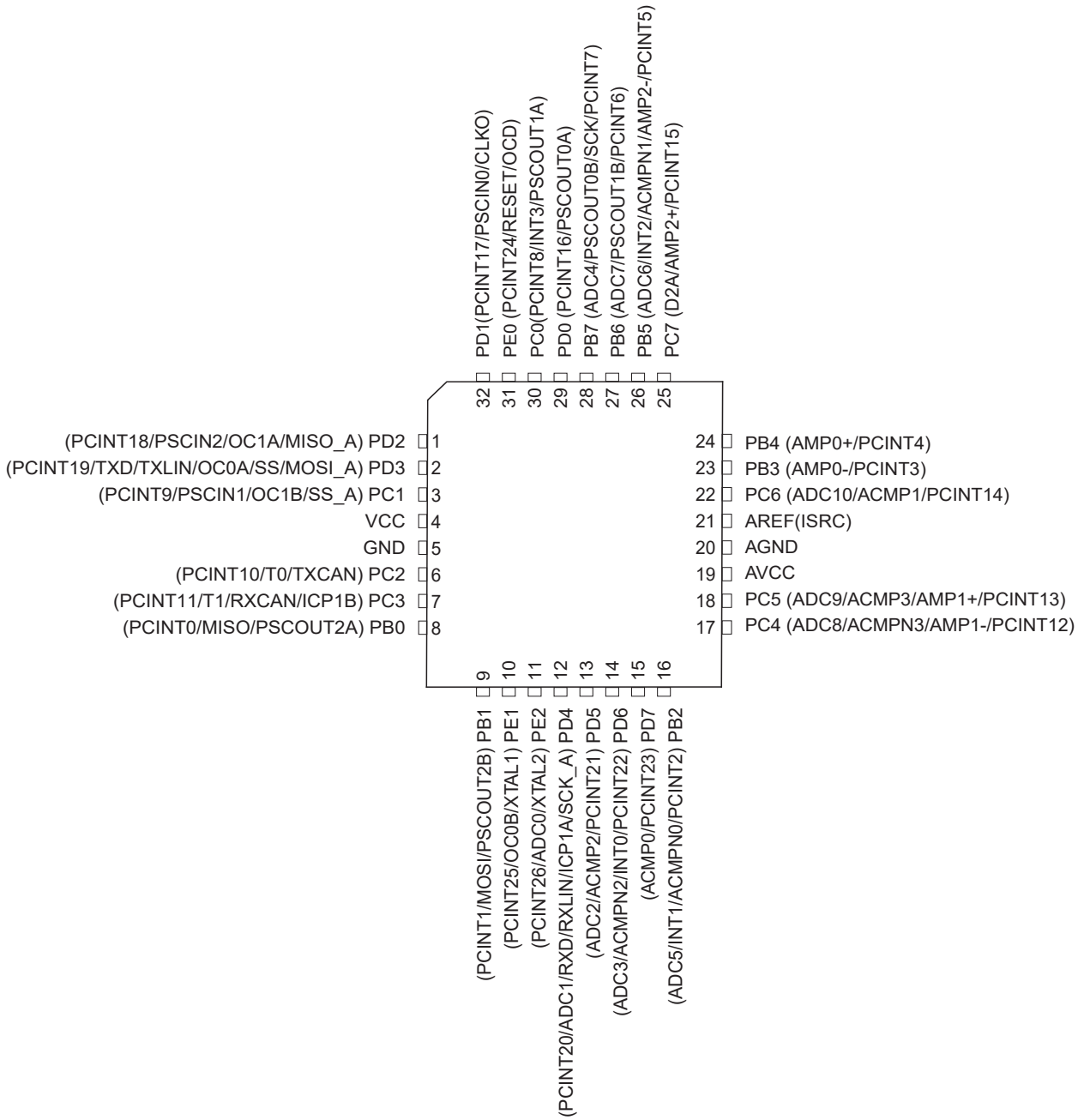
3. Block diagram

Figure 3-1. Block diagram.



4. Pin configurations

Figure 4-1. ATmegaS64M1 Pinout.



4.1 Pin Descriptions

Table 4-1. Pinout Description

Pin No.	Mnemonic	Type	Name, Function	
			Standard Function	Alternate Function
4	VCC	Power	Digital Power Supply	
5	GND	Power	Ground: 0V reference	
19	AVCC	Power	Analog Power Supply–this pin supplies the voltage for the A/D Converter, D/A Converter, Current source. It should be externally connected to V_{CC}, even if the ADC, DAC are not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.	
20	AGND	Power	Analog ground–0V reference for analog part.	
21	AREF	Power	Analog reference–reference for analog converter. This is the reference voltage of the A/D converter.	As output, can be used by external analog. ISRC (Current Source Output)
8	PB0	I/O	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.	MISO (SPI Master In Slave Out) PSCOUT2A (PSC Module 2 Output A) PCINT0 (Pin Change Interrupt 0)
9	PB1	I/O		MOSI (SPI Master Out Slave In) PSCOUT2B (PSC Module 2 Output B) PCINT1 (Pin Change Interrupt 1)
16	PB2	I/O		ADC5 (Analog Input Channel 5) INT1 (External Interrupt 1 Input) ACMPN0 (Analog Comparator 0 Negative Input) PCINT2 (Pin Change Interrupt 2)
23	PB3	I/O		AMP0- (Analog Differential Amplifier 0 Negative Input) PCINT3 (Pin Change Interrupt 3)
24	PB4	I/O		AMP0+ (Analog Differential Amplifier 0 Positive Input) PCINT4 (Pin Change Interrupt 4)
26	PB5	I/O		ADC6 (Analog Input Channel 6) INT2 (External Interrupt 2 Input) ACMPN1 (Analog Comparator 1 Negative Input) AMP2- (Analog Differential Amplifier 2 Negative Input)

Pin No.	Mnemonic	Type	Name, Function	
			Standard Function	Alternate Function
				PCINT5 (Pin Change Interrupt 5)
27	PB6	I/O		ADC7 (Analog Input Channel 7) PSCOUT1B (PSC Module 1 Output A) PCINT6 (Pin Change Interrupt 6)
28	PB7	I/O		ADC4 (Analog Input Channel 4) PSCOUT0B (PSC Module 0 Output B) SCK (SPI Clock) PCINT7 (Pin Change Interrupt 7)
30	PC0	I/O	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.	PSCOUT1A (PSC Module 1 Output A) INT3 (External Interrupt 3 Input) PCINT8 (Pin Change Interrupt 8)
3	PC1	I/O		PSCIN1 (PSC Digital Input 1) OC1B (Timer 1 Output Compare B) SS_A (Alternate SPI Slave Select) PCINT9 (Pin Change Interrupt 9)
6	PC2	I/O		T0 (Timer 0 clock input) TXCAN (CAN Transmit Output) PCINT10 (Pin Change Interrupt 10)
7	PC3	I/O		T1 (Timer 1 clock input) RXCAN (CAN Receive Input) ICP1B (Timer 1 input capture alternate B input) PCINT11 (Pin Change Interrupt 11)
17	PC4	I/O		ADC8 (Analog Input Channel 8) AMP1- (Analog Differential Amplifier 1 Negative Input) ACMPN3 (Analog Comparator 3 Negative Input) PCINT12 (Pin Change Interrupt 12)
18	PC5	I/O		ADC9 (Analog Input Channel 9) AMP1+ (Analog Differential Amplifier 1 Positive Input) ACMP3 (Analog Comparator 3 Positive Input)

Pin No.	Mnemonic	Type	Name, Function	
			Standard Function	Alternate Function
				PCINT13 (Pin Change Interrupt 13)
22	PC6	I/O		ADC10 (Analog Input Channel 10) ACMP1 (Analog Comparator 1 Positive Input) PCINT14 (Pin Change Interrupt 14)
25	PC7	I/O		D2A (DAC output) AMP2+ (Analog Differential Amplifier 2 Positive Input) PCINT15 (Pin Change Interrupt 15)
29	PD0	I/O	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.	PSCOUT0A (PSC Module 0 Output A) PCINT16 (Pin Change Interrupt 16)
32	PD1	I/O		PSCIN0 (PSC Digital Input 0) CLKO (System Clock Output) PCINT17 (Pin Change Interrupt 17)
1	PD2	I/O		OC1A (Timer 1 Output Compare A) PSCIN2 (PSC Digital Input 2) MISO_A (Programming & alternate SPI Master In Slave Out) PCINT18 (Pin Change Interrupt 18)
2	PD3	I/O		TXD (UART Tx data) TXLIN (LIN Transmit Output) OC0A (Timer 0 Output Compare A) SS (SPI Slave Select) MOSI_A (Programming & alternate Master Out SPI Slave In) PCINT19 (Pin Change Interrupt 19)
12	PD4	I/O		ADC1 (Analog Input Channel 1) RXD (UART Rx data) RXLIN (LIN Receive Input) ICP1A (Timer 1 input capture alternate A input) SCK_A (Programming & alternate SPI Clock) PCINT20 (Pin Change Interrupt 20)

Pin No.	Mnemonic	Type	Name, Function	
			Standard Function	Alternate Function
13	PD5	I/O		ADC2 (Analog Input Channel 2) ACMP2 (Analog Comparator 2 Positive Input) PCINT21 (Pin Change Interrupt 21)
14	PD6	I/O		ADC3 (Analog Input Channel 3) ACMPN2 (Analog Comparator 2 Negative Input) INT0 (External Interrupt 0 Input) PCINT22 (Pin Change Interrupt 22)
15	PD7	I/O		ACMP0 (Analog Comparator 0 Positive Input) PCINT23 (Pin Change Interrupt 23)
31	PE0	I/O or I	Port E is an 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the RSTDISBL Fuse is programmed, PE0 is used as an I/O pin. Note that the electrical characteristics of PE0 differ from those of the other pins of Port E. If the RSTDISBL Fuse is unprogrammed, PE0 is used as a 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. Depending on the clock selection fuse settings, PE1 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PE2 can be used as output from the inverting Oscillator amplifier.	RESET (Reset Input) OCD (On Chip Debug I/O) PCINT24 (Pin Change Interrupt 24)
10	PE1	I/O		XTAL1 (XTAL Input) OC0B (Timer 0 Output Compare B) PCINT25 (Pin Change Interrupt 25)
11	PE2	I/O		XTAL2 (XTAL Output) ADC0 (Analog Input Channel 0) PCINT26 (Pin Change Interrupt 26)

5. Ordering Information

Ordering Code	Speed (MHz)	Power Supply	Package	Flow
ATmegaS64M1-KH-E	8 MHz	3.0–3.6V	CQFP32	Engineering Samples
ATmegaS64M1-KH-MQ				QML-Q equivalent
ATmegaS64M1-KH-SV				QML-V equivalent
ATmegaS64M1-MA-HP			TQFP32	Hirel Plastic

6. 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. About code examples

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

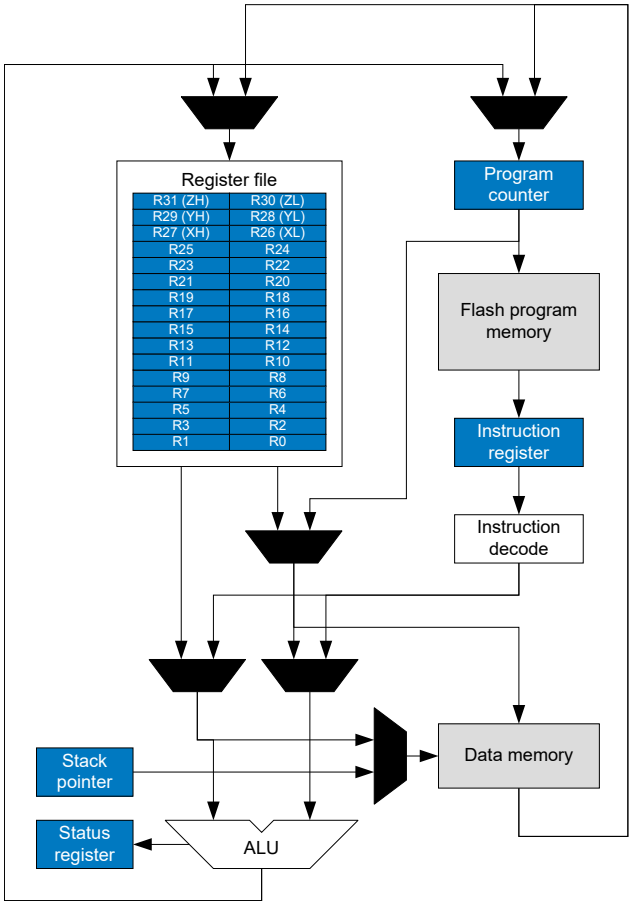
These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined 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 these address pointers can be used as an

address pointer for lookup tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

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

Program flow is provided by conditional and unconditional jump and call instructions, 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 into 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 Stack Pointer (SP) in the Reset routine (before subroutines or interrupts are executed). The 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 Arithmetic Logic Unit (ALU)

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 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

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

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.

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 ⊕ 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.

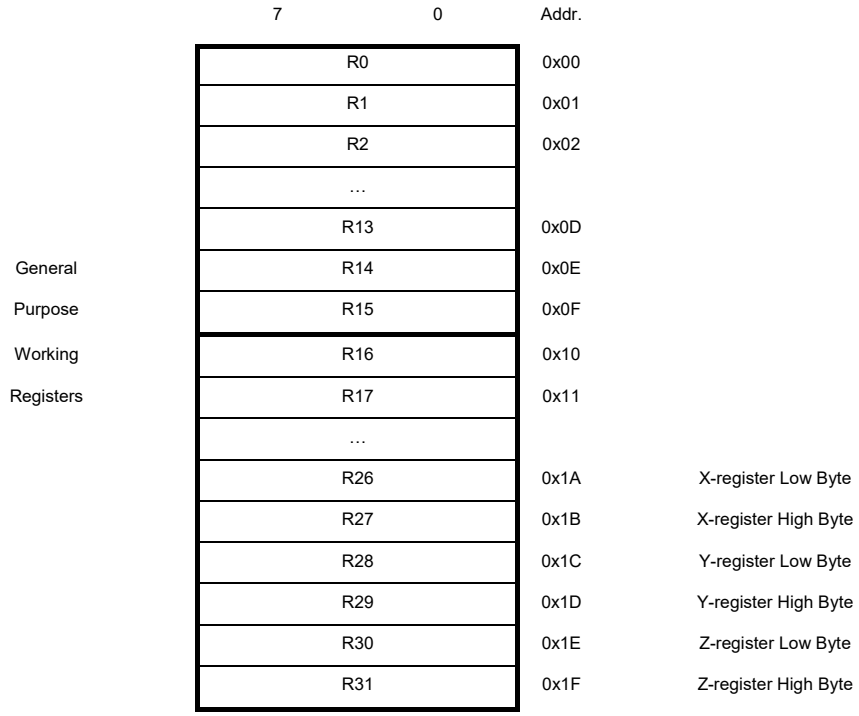
Bit 0 – C Carry Flag
The carry flag C indicates a carry in an arithmetic or logic operation. See the *Instruction Set Description* for detailed information.

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



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, local variables, and 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

Name: SPL and SPH
Offset: 0x5D
Reset: 0x10FF
Property: When addressing I/O Registers as data space the offset address is 0x3D

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.

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.

Bit	15	14	13	12	11	10	9	8
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 – SP Stack Pointer Register
SPL and SPH are combined into SP.

8.6 Accessing 16-Bit Registers

The AVR data bus has a width of 8 bit, 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.

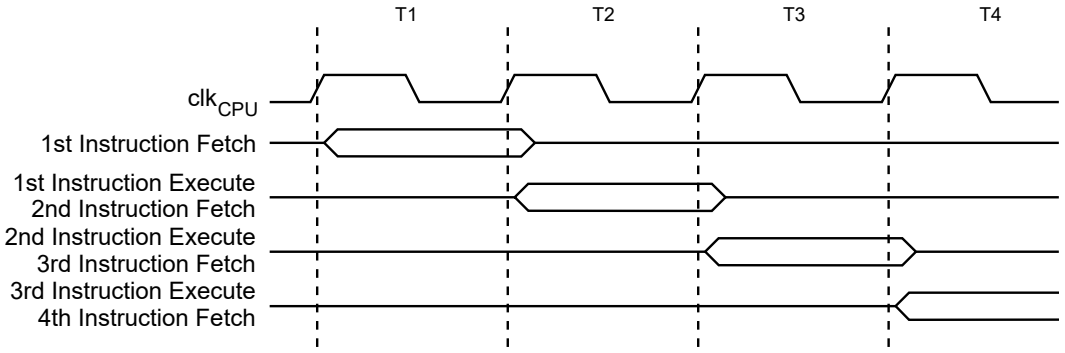
Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

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

8.7 Instruction Execution Timing

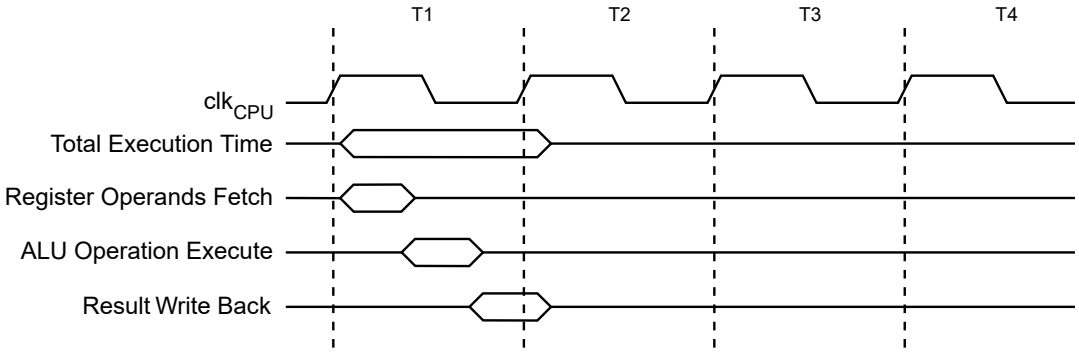
This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used. The figure below shows the parallel instruction fetches and instruction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

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 microcontroller (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

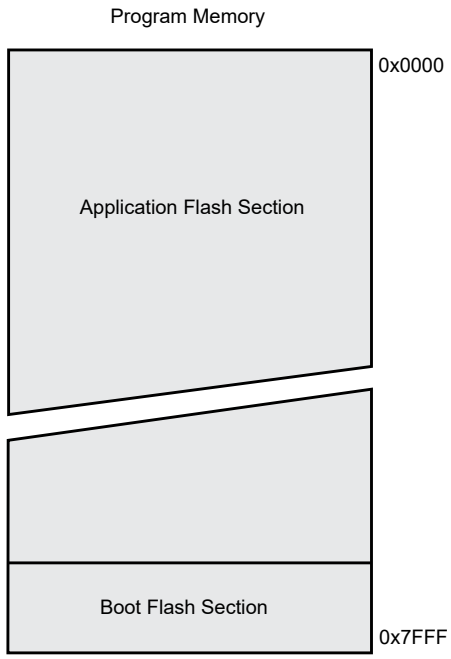
The ATmegaS64M1 contains 64Kbytes on-chip in-system reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 32K x 16.

The ATmegaS64M1 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 of Flash data serial downloading using the SPI pins.

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 ATmegaS64M1



Related Links

Boot Loader Support – Read-While-Write Self-programming (BTLDR)
Memory Programming (MEMPROG)
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.

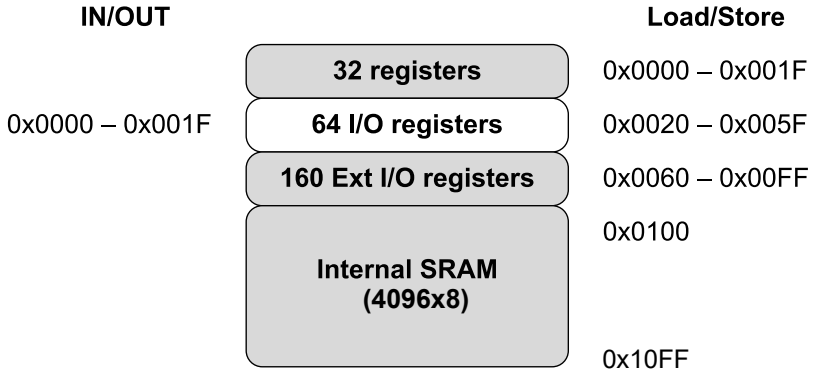
The lower 4352 data memory locations address both the register file, the I/O memory, extended I/O memory, and the internal data SRAM. The first 32 locations address the register file, the next 64 location the standard I/O memory, then 160 locations of extended I/O memory, and the next 4K locations address 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 4K bytes of internal data SRAM in the device are all accessible through all these addressing modes.

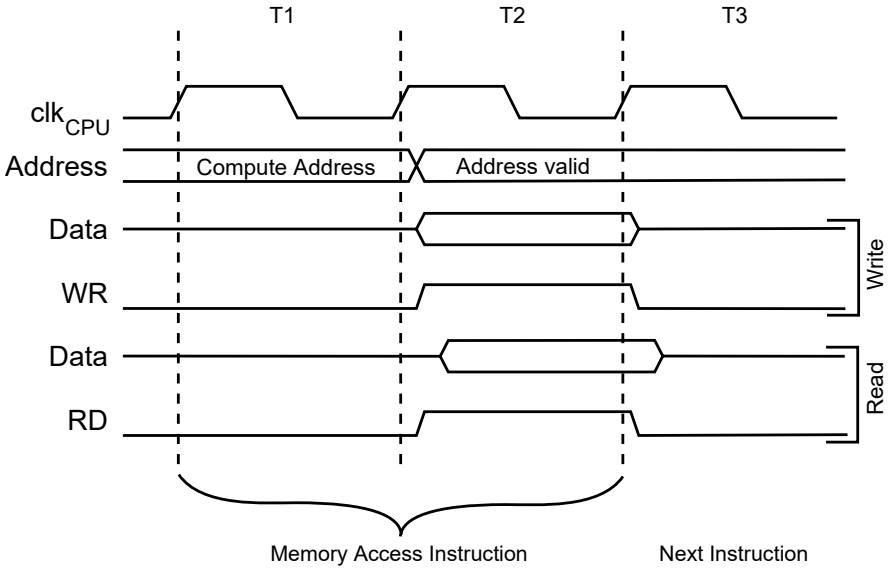
Figure 9-2. Data Memory Map with 4096 Byte Internal Data SRAM



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 ATmegaS64M1 contains 2KB of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. 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

Memory Programming (MEMPROG)

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 a minimum for the clock frequency used. 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 locations 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

Memory Programming (MEMPROG)

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 has a width of 8 bit, 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 be read and written directly from user software.

9.6.2 EEPROM Address Register Low and High Byte

Name: EEARL and EEARH
Offset: 0x41
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x21

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 in the section above.

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.

Bit	15	14	13	12	11	10	9	8
						EEAR[10:8]		
Access						R/W	R/W	R/W
Reset						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 10:0 – EEAR[10:0] EEPROM Address
The EEPROM Address Registers, EEARH and EEARL, specify the EEPROM address in the 2KB 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

Name: EEDR
Offset: 0x40
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x20

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.

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
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 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]	Typ. Programming Time	Operation
00	3.4 ms	Erase and Write in one operation (Atomic Operation)
01	1.8 ms	Erase Only
10	1.8 ms	Write Only
11	-	Reserved for future use

Bit 3 – EERIE EEPROM Ready Interrupt Enable
Writing EERIE to '1' 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 cannot 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	3.3 ms

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) 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

Name: GPIOR2
Offset: 0x3A
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2B

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.

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

Name: GPIOR1
Offset: 0x39
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2A

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.

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

Name: GPIOR0
Offset: 0x3E
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1E

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.

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

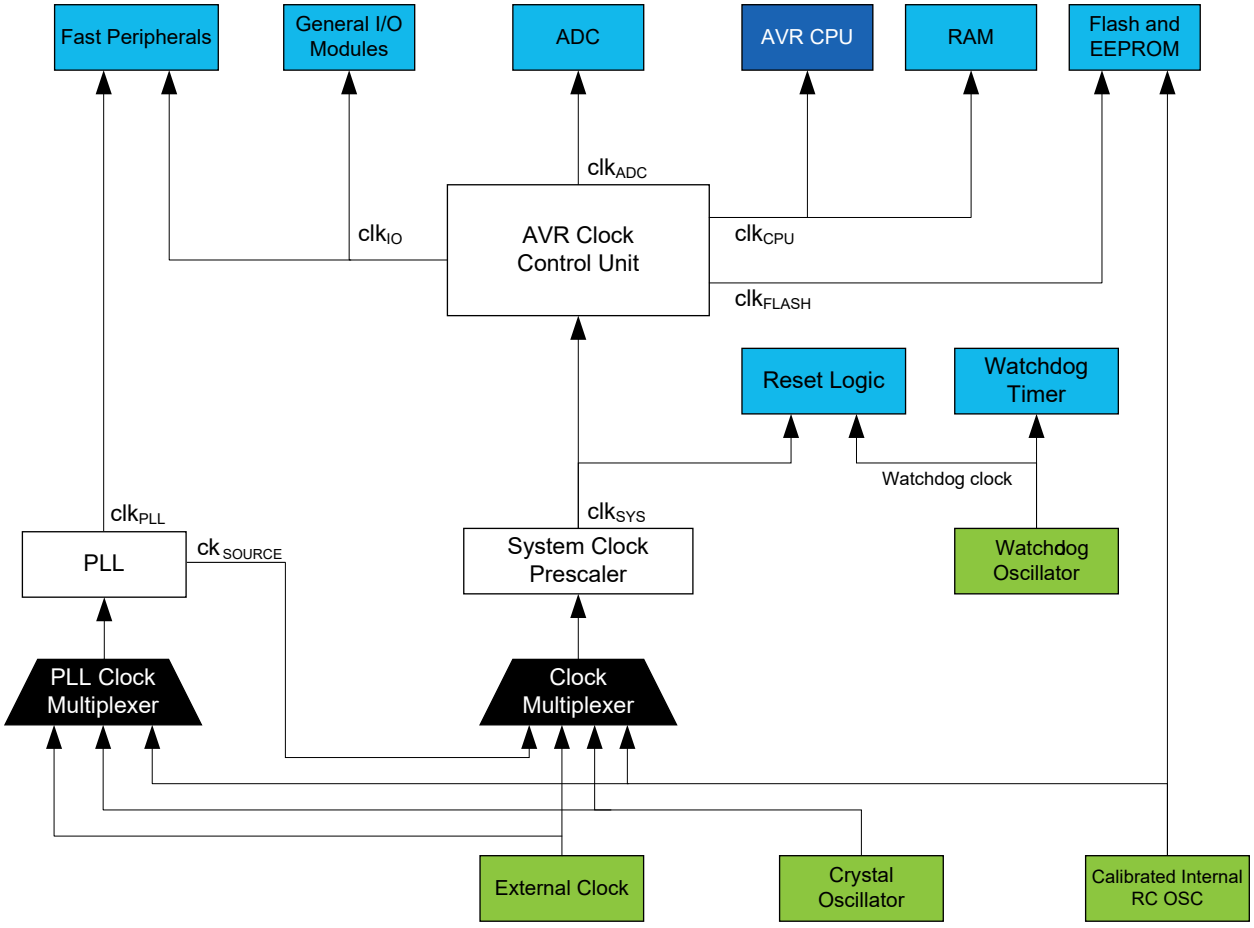
10. System Clock and Clock Options

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 Flash Clock – clk_{FLASH}

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

10.1.4 PLL Clock – clk_{PLL}

The PLL clock allows the fast peripherals to be clocked directly from a 64/32 MHz clock.

A clock derived from the PLL cannot be used to clock the CPU in the space environment.

10.1.5 Asynchronous Timer Clock – clk_{ASY}

The asynchronous timer clock allows asynchronous timer/counters to be clocked directly from an external clock. 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 Selector

Device Clocking Option	System Clock	PLL Input	CKSEL[3:0]
External crystal or ceramic resonator/ internal oscillator	Ext osc	RC osc	1111 - 1000
External crystal/ceramic resonator	Ext osc	Ext osc	0100
Reserved	N/A	N/A	0101
Reserved	N/A	N/A	0110
Reserved	N/A	N/A	0111
Reserved	N/A	N/A	0011
Calibrated internal RC oscillator	RC osc	RC osc	0010

Device Clocking Option	System Clock	PLL Input	CKSEL[3:0]
Reserved	N/A	N/A	0001
External clock	Ext clk	RC osc	0000

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

The choices for each clocking option are given in the following sections. When the CPU wakes up from power-down or power-save, the selected clock source is used to time the start-up, ensuring stable oscillator operation before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before starting normal operation. The Watchdog oscillator is used for timing this real-time part of the start-up time. The number of WDT oscillator cycles used for each time-out is shown below.

Table 10-2. Number of Watchdog Oscillator Cycles

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

10.3 Default Clock Source

The device is shipped with CKSEL = “0010”, SUT = “10”, and CKDIV8 unprogrammed. The default clock source setting is the Internal RC Oscillator with longest start-up time and an initial system clock prescaling of one. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel programmer.

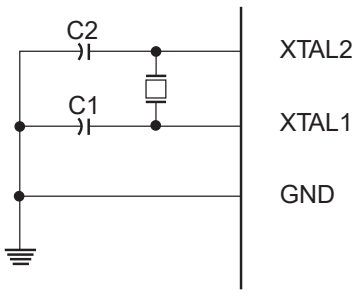
10.4 Low Power Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in the figure below. Either a quartz crystal or a ceramic resonator may be used.

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.

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 table below. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 10-2. Crystal Oscillator Connections



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

Table 10-3. Crystal Oscillator Operating Modes

CKSEL[3..1]	Frequency Range [MHz]	Recommended Range for Capacitors C1 and C2 for Use with Crystals [pF]
100 (see Note)	0.4–0.9	–
101	0.9–3.0	12–22
110	3.0–8.0	12–22
111	Reserved	

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

The CKSEL0 fuse together with the SUT[1..0] fuses select the startup times as shown below.

Table 10-4. Start-up Times for Oscillator Clock Selection

CKSEL0	SUT[1..0]	Startup Time from Power-down and Power-save	Additional Delay from Reset (VCC = 3.3V)	Recommended Usage
0	00	258 CK (see Note 1)	14CK + 4.1ms	Ceramic resonator, fast rising power
0	01	258 CK (see Note 1)	14CK + 65ms	Ceramic resonator, slow rising power
0	10	1K CK (see Note 2)	14CK	Ceramic resonator, BOD enabled
0	11	1K CK (see Note 2)	14CK + 4.1ms	Ceramic resonator, fast rising power
1	00	1K CK (see Note 2)	14CK + 65ms	Ceramic resonator, slow rising power
1	01	16K CK	14CK	Ceramic resonator, BOD enabled
1	10	16K CK	14CK + 4.1ms	Ceramic resonator, fast rising power
1	11	16K CK	14CK + 65ms	Ceramic resonator, slow rising power

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

PSC clock sources

10.5 Calibrated Internal RC Oscillator

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

This clock may be selected as the system clock by programming the CKSEL fuses as shown in the following table. During Reset, hardware loads the pre-programmed calibration value into the OSCCAL register and thereby automatically calibrates the RC oscillator. A 10K pull-down should be connected between XTAL1 and GROUND while XTAL2 should be left unconnected (NC).

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 10-5. Internal Calibrated RC Oscillator Operating Modes

Frequency Range(1) [MHz]	CKSEL[3:0]
8.0	0010(2)

Note:

1. If 8 MHz 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 10-6. 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} = 3.3V)	SUT[1:0]
BOD enabled	6 CK	14 CK	00
Fast rising power	6 CK	14 CK + 4 ms	01
Slow rising power	6 CK	14 CK + 65 ms	10(1)
Reserved			11

Note:

1. The device is shipped with this option selected.

Related Links

System Clock Prescaler

Calibration Byte

OSCCAL

10.6 PLL

10.6.1 Internal PLL

The internal PLL in the ATmegaS64M1 generates a clock frequency that is 64× multiplied from nominally 1MHz input. The source of the 1MHz PLL input clock is the output of the internal RC oscillator which is divided down to 1MHz.

The PLL is locked on the RC oscillator, and adjusting the RC oscillator in the OSCCAL register adjusts the fast peripheral clock at the same time. However, even if the possibly divided RC oscillator is taken to a higher frequency than 1MHz, the fast peripheral clock frequency saturates at 70MHz (worst case) and remains oscillating at the maximum frequency. Note that in this case the PLL is no longer locked with the RC oscillator clock.

Therefore it is recommended to keep OSCCAL adjustments to a frequency lower than 1MHz in order to keep the PLL in the correct operating range. The internal PLL is enabled only when the PLLE bit in the register PLLCSR is set. The bit PLOCK from the register PLLCSR is set when the PLL is locked.

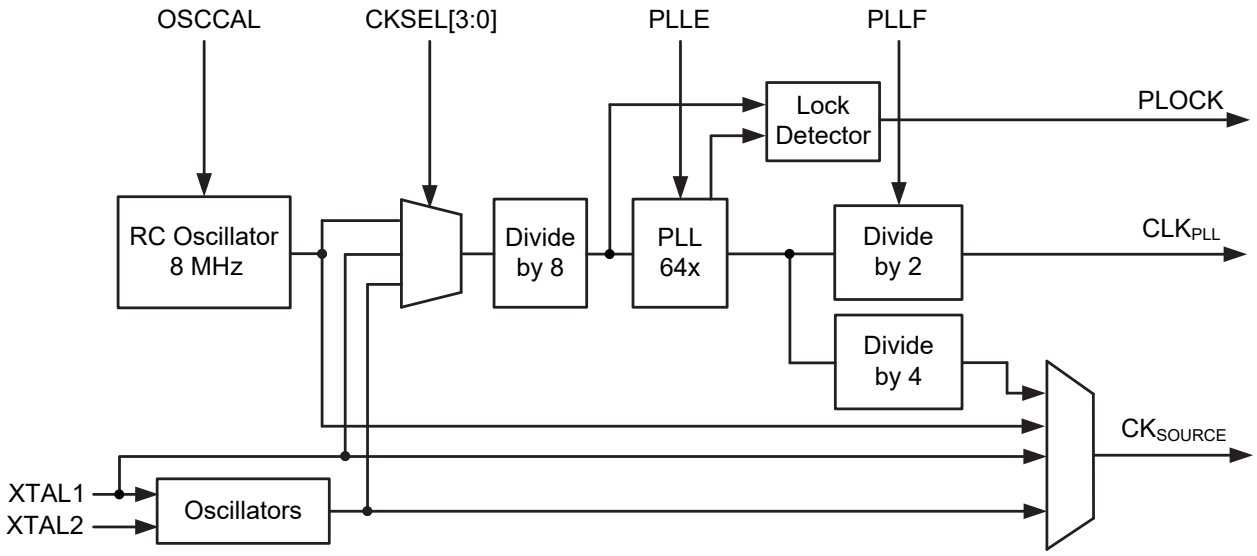
The internal 1MHz RC oscillator and the PLL are switched off in Power-down and Standby sleep modes.

Table 10-7. Startup Times with PLL Selected as System Clock

CKSEL[3..0]	SUT[1..0]	Startup Time from Power-down and Power-save	Additional Delay from Reset (VCC = 3.3V)
0011 RC Osc	00	1K CK	14CK
	01	1K CK	14CK + 4ms
	10	1K CK	14CK + 64ms
	11	16K CK	14CK
0101 Ext Osc	00	1K CK	14CK
	01	1K CK	14CK + 4ms
	10	16K CK	14CK + 4ms
	11	16K CK	14CK + 64ms
0001 Ext Clk	00	6 CK(1)	14CK
	01	6 CK(1)	14CK + 4ms
	10	6 CK(1)	14CK + 64ms
	Reserved		

Note: This value does not provide a proper restart; do not use PD in this clock scheme.

Figure 10-3. PLL Clocking System



10.7 128kHz Internal Oscillator

The 128kHz internal oscillator is a low power oscillator providing a clock of 128kHz. The frequency is nominal at 3V and 25°C. This clock is used by the Watchdog Oscillator.

10.8 External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown below. To run the device on an external clock, the CKSEL fuses must be programmed to “0000”.

Figure 10-4. External Clock Drive Configuration

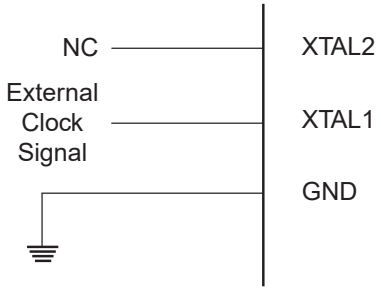


Table 10-8. External Clock Frequency

CKSEL[3:0]	Frequency
0000	0 – 8MHz

When this clock source is selected, startup times are determined by the SUT fuses:

Table 10-9. Startup Times for the External Clock Selection – SUT

SUT[1:0]	Startup Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 3.3V)	Recommended Usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4.1ms	Fast rising power
10	6 CK	14CK + 65ms	Slowly rising power
11	Reserved		

When applying an external clock, sudden changes must be avoided 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 a variation greater 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.

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 occur 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 away from the oscillator frequency. A preprogrammed calibration value is automatically written to this register during chip reset, giving the factory calibrated frequency as specified in the *Clock Characteristics* section of chapter *Electrical Characteristics*.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 chapter *Electrical Characteristics*. Calibration outside that range is not recommended.

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 8.8 MHz. 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 PLL Control and Status Register

Name: PLLCSR
Offset: 0x49
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
						PLLF	PLLE	PLOCK
Access						R/W	R/W	R
Reset						0	x	0

Bit 2 – PLLF PLL Factor
The PLLF bit is used to select the division factor of the PLL.

Value	Description
1	If PLLF is set, the PLL output is 64MHz.
0	If PLLF is cleared, the PLL output is 32MHz.

Bit 1 – PLLE PLL Enable
When the PLLE is set, the PLL is started and if not yet started the internal RC Oscillator is started as PLL reference clock. If PLL is selected as a system clock source the value for this bit is always 1.

Bit 0 – PLOCK PLL Lock Detector
When the PLOCK bit is set, the PLL is locked to the reference clock, and it is safe to enable CLKPLL for Fast Peripherals. After the PLL is enabled, it takes about 100ms for the PLL to lock..

10.11.3 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 10-10. 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

CLKPS[3:0]	Clock Division Factor
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

11. Power Management and Sleep Modes

11.1 Sleep Modes

The following Table shows the different sleep modes and their wake-up sources.

Table 11-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

Sleep Mode	Active Clock Domains					Oscillators	Wake-up Sources					
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{PLL}	Main Clock Source Enabled	INT3..0	PSC	SPM/EEPROM Ready	ADC	WDT	Other I/O
Idle			Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ADC Noise Reduction				Yes	Yes	Yes	Yes(2)	Yes	Yes	Yes	Yes	
Power-down							Yes(2)				Yes	
Standby (1)						Yes	Yes(2)				Yes	

1. **Note:** Only recommended with external crystal or resonator selected as clock source.
2. **Note:** Only level interrupt.

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.

11.2 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, two-wire serial interface, 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.

The 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.

Related Links

ACSR

11.3 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 two-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
- Two-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

16-bit Timer/Counter1 (TC1) with PWM

11.4 Power-Down Mode

When the SM[2:0] bits are written to '010', the SLEEP instruction makes the MCU enter the Power-Down mode. In this mode, the external oscillator is stopped, while the external interrupts, the two-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
- Two-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 the 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

PSC clock sources

Overview

11.5 Standby Mode

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

11.6 Power Reduction Register

The Power Reduction Register (PRR) 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 cannot 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.

11.7 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.

11.7.1 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.

11.7.2 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 the sleep mode.

11.7.3 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

11.7.4 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 (ADC). 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

11.7.5 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

11.7.6 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

11.7.7 On-chip Debug System

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

11.8 Register Description

11.8.1 Sleep Mode Control Register

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

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.

Bit	7	6	5	4	3	2	1	0
					SM2	SM1	SM0	SE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – SM2 Sleep Mode Select 2
The SM[2:0] bits select between the five available sleep modes.

Table 11-2. Sleep Mode Select

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

Note:
1. Standby mode is only recommended for use with external clock source.

Bit 2 – SM1 Sleep Mode Select 1
Refer to SM2.

Bit 1 – SM0 Sleep Mode Select 0
Refer to SM2.

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.

11.8.2 Power Reduction Register

Name: PRR
Offset: 0x64
Reset: 0x00
Property: R/W

Bit	7	6	5	4	3	2	1	0
		PRCAN	PRPSC	PRTIM1	PRTIM0	PRSPI	PRLIN	PRADC
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 6 – PRCAN Power Reduction CAN
Writing a logic one to this bit reduces the consumption of the CAN by stopping the clock to this module. When waking up the CAN again, the CAN should be re initialized to ensure proper operation.

Bit 5 – PRPSC Power Reduction PSC
Writing a logic one to this bit reduces the consumption of the PSC by stopping the clock to this module. When waking up the PSC again, the PSC should be re initialized to ensure proper operation.

Bit 4 – 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 3 – 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 2 – PRSPI Power Reduction Serial Peripheral Interface
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 – PRLIN Power Reduction LIN
Writing a logic one to this bit reduces the consumption of the LIN controller by stopping the clock to this module. When waking up the LIN controller again, the LIN controller 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. System Control and Reset

12.1 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 ATmegaS64M1. 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 the 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

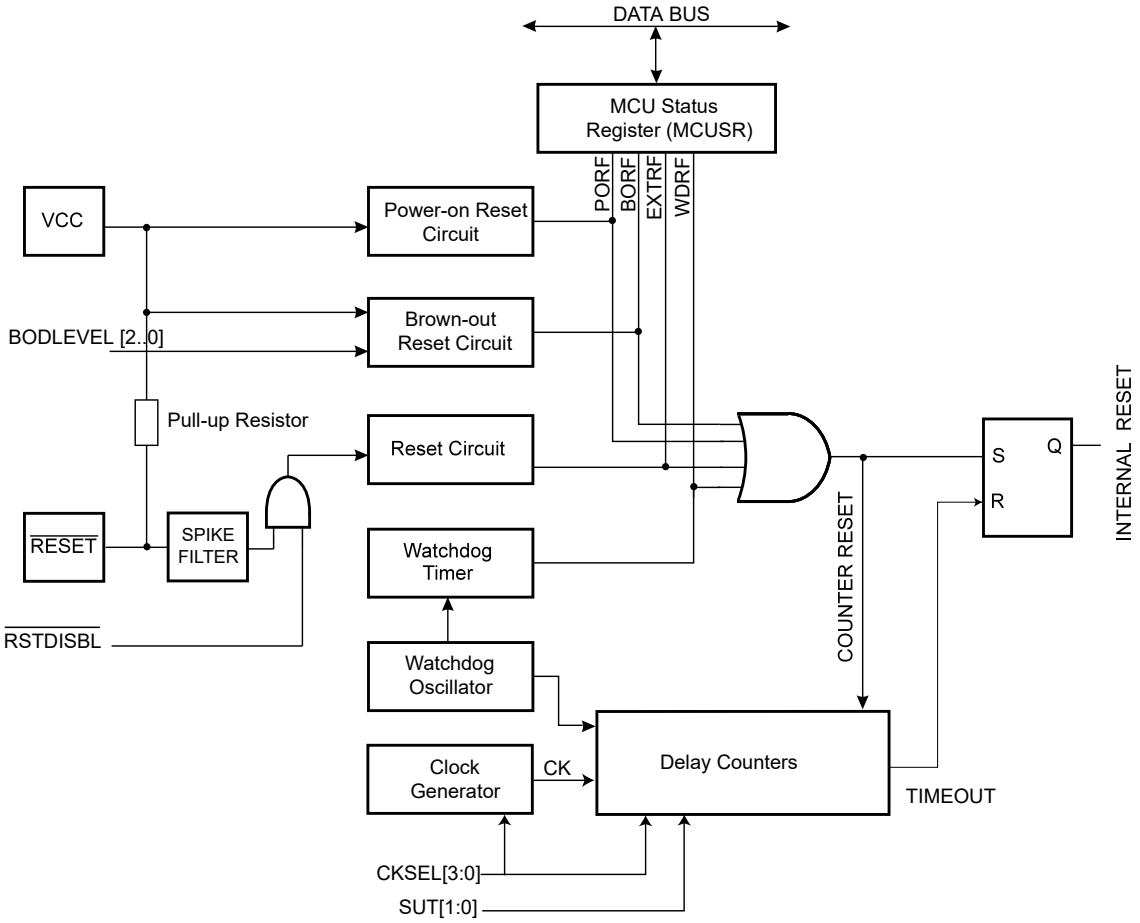
System Clock and Clock Options

12.2 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.

Figure 12-1. Reset Logic



12.3 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 POR circuit ensures that the device is reset from power-on. Reaching the POR 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 12-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

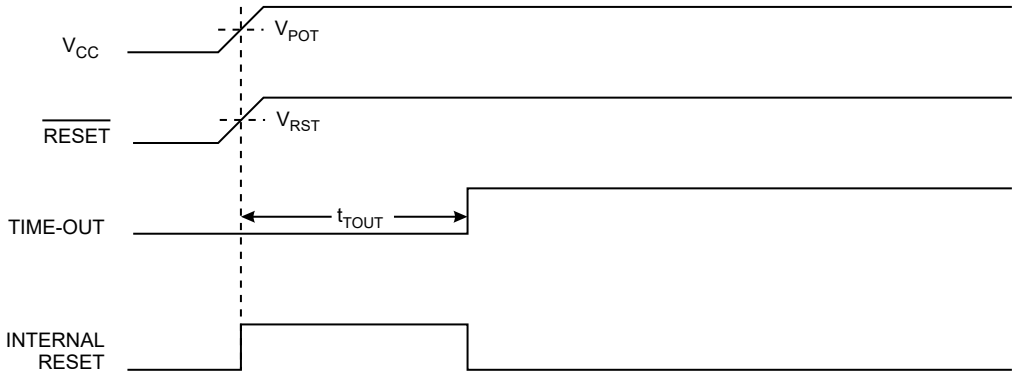
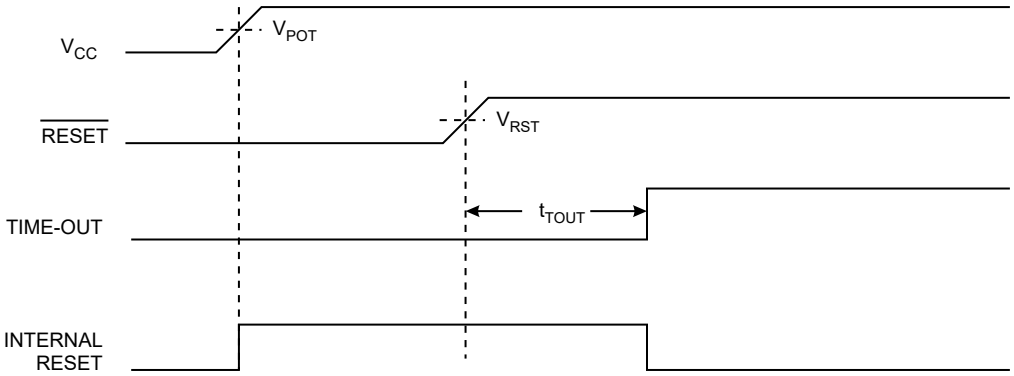


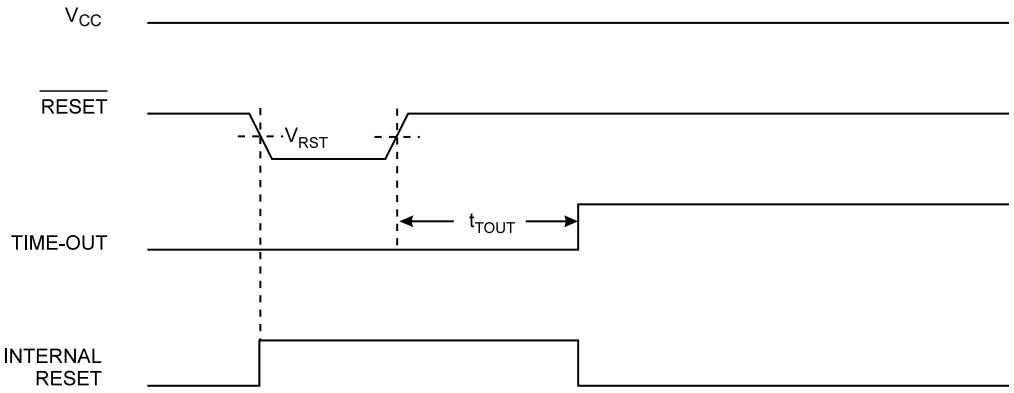
Figure 12-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



12.4 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 12-4. External Reset During Operation

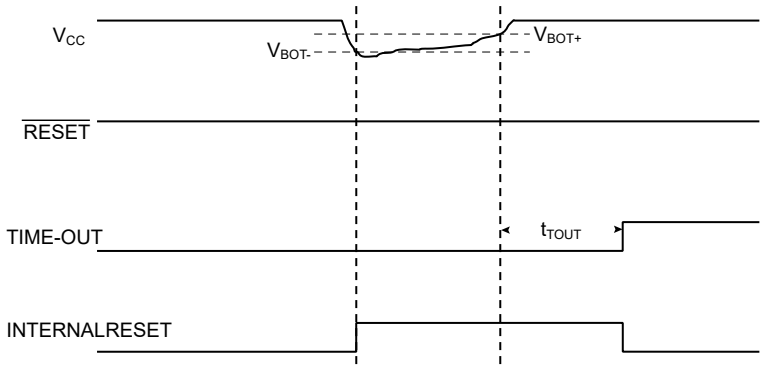


12.5 Brown-out Detection

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 BOD. 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}.

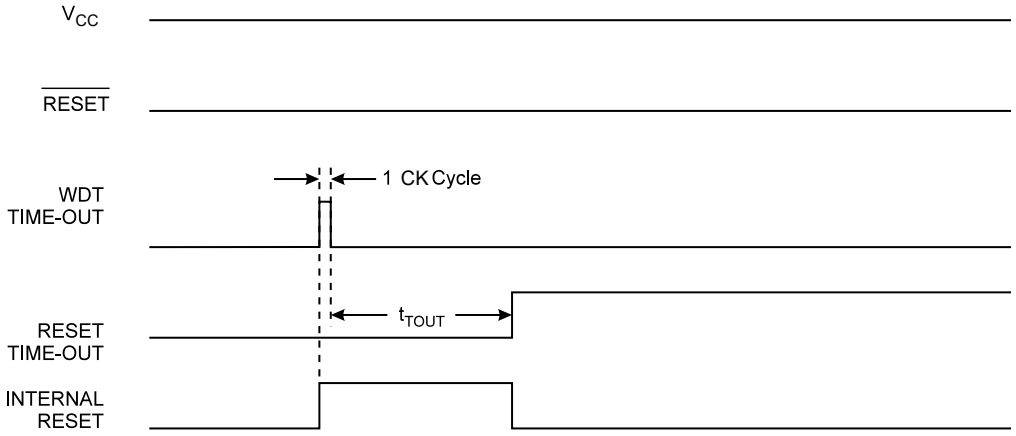
Figure 12-5. Brown-out Reset During Operation



12.6 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 12-6. Watchdog System Reset During Operation



12.7 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.

12.7.1 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 the Power-Down mode, the user can avoid the three conditions above to ensure that the reference is turned OFF before entering Power-Down mode.

12.8 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.

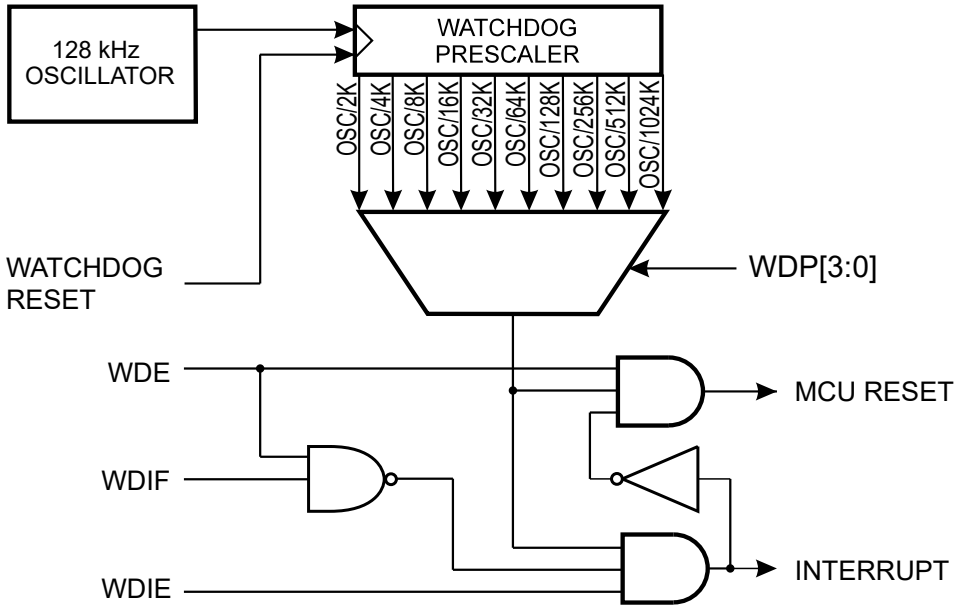
12.8.1 Features

- Clocked from Separate On-chip Oscillator
- Three Operating modes:
  - Interrupt
  - System Reset
  - Interrupt and System Reset
- Selectable Time-out Period from 16 ms to 8s
- Possible Hardware Fuse Watchdog Always ON (WDTON) for Fail-safe mode

12.8.2 Overview

The device has an Enhanced Watchdog Timer (WDT). The WDT is a timer counting cycles of a separate on-chip 128 kHz 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 12-7. 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 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.

12.9 Register Description

12.9.1 MCU Status Register

Name: MCUSR
Offset: 0x54
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x34

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.

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

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.

12.9.2 Watchdog Timer Control Register

Name: WDTCSR
Offset: 0x60
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
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 time out occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a '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 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 12-1. 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 – WDP3 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 in section *Watchdog Timer* for information on 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 0, 1, 2 – WDP 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 time out periods are shown in the following table.

Table 12-2. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator (Cycles)	Oscillator
0	0	0	0	2K (2048)	16 ms
0	0	0	1	4K (4096)	32 ms
0	0	1	0	8K (8192)	64 ms
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		
1	1	0	1		
1	1	1	0		
1	1	1	1		

13. 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 two instruction words for ATmegaS64M1
- 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

13.1 Interrupt Vectors in ATmegaS64M1

Table 13-1. Reset and Interrupt Vectors in ATmegaS64M1

Vector No	Program Address	Source	Interrupts definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog System Reset, and emulation AVR reset
2	0x0002	ANACOMP 0	Analog Comparator 0
3	0x0004	ANACOMP 1	Analog Comparator 1
4	0x0006	ANACOMP 2	Analog Comparator 2
5	0x0008	ANACOMP 3	Analog Comparator 3
6	0x000A	PSC FAULT	PSC fault
7	0x000C	PSC EC	PSC end of cycle
8	0x000E	INT0	External Interrupt Request 0
9	0x0010	INT1	External Interrupt Request 1
10	0x0012	INT2	External Interrupt Request 2
11	0x0014	INT3	External Interrupt Request 3
12	0x0016	TIMER1 CAPT	Timer/Counter1 capture event
13	0x0018	TIMER1 COMPA	Timer/Counter1 compare match A
14	0x001A	TIMER1 COMPB	Timer/Counter1 compare match B
15	0x001C	TIMER1 OVF	Timer/Counter1 overflow
16	0x001E	TIMER0 COMPA	Timer/Counter0 compare match A
17	0x0020	TIMER0 COMPB	Timer/Counter0 compare match B
18	0x0022	TIMER0 OVF	Timer/Counter0 overflow
19	0x0024	CAN INT	CAN MOB, burst, general errors
20	0x0026	CAN TOVF	CAN timer overflow
21	0x0028	LIN TC	LIN transfer complete

Vector No	Program Address	Source	Interrupts definition
22	0x002A	LIN ERR	LIN error
23	0x002C	PCINT0	Pin change interrupt request 0
24	0x002E	PCINT1	Pin change interrupt request 1
25	0x0030	PCINT2	Pin change interrupt request 2
26	0x0032	PCINT3	Pin change interrupt request 3
27	0x0034	SPI, STC	SPI serial transfer complete
28	0x0036	ADC	ADC conversion complete
29	0x0038	WDT	Watchdog time-out interrupt
30	0x003A	EE READY	EEPROM ready
31	0x003C	SPM READY	Store program memory ready

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 13-2. 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.

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

```
Address Labels Code Comments
0x000 jmp RESET ; Reset Handler
0x002 jmp ANA_COMP_0 ; Analog Comparator 0 Handler
0x004 jmp ANA_COMP_1 ; Analog Comparator 1 Handler
0x006 jmp ANA_COMP_2 ; Analog Comparator 2 Handler
0x008 jmp ANA_COMP_3 ; Analog Comparator 3 Handler
0x00A jmp PSC_FAULT ; PSC Fault Handler
0x00C jmp PSC_EC ; PSC End of Cycle Handler
0x00E jmp EXT_INT0 ; IRQ0 Handler
0x010 jmp EXT_INT1 ; IRQ1 Handler
0x012 jmp EXT_INT2 ; IRQ2 Handler
```

```
0x014 jmp EXT_INT3 ; IRQ3 Handler
0x016 jmp TIM1_CAPT ; Timer1 Capture Handler
0x018 jmp TIM1_COMPA ; Timer1 Compare A Handler
0x01A jmp TIM1_COMPB ; Timer1 Compare B Handler
0x01C jmp TIM1_OVF ; Timer1 Overflow Handler
0x01E jmp TIM0_COMPA ; Timer0 Compare A Handler
0x020 jmp TIM0_COMPB ; Timer0 Compare B Handler
0x022 jmp TIM0_OVF ; Timer0 Overflow Handler
0x024 jmp CAN_INT ; CAN MOB,Burst,General Errors Handler
0x026 jmp CAN_TOVF ; CAN Timer Overflow Handler
0x028 jmp LIN_TC ; LIN Transfer Complete Handler
0x02A jmp LIN_ERR ; LIN Error Handler
0x02C jmp PCINT0 ; Pin Change Int Request 0 Handler
0x02E jmp PCINT1 ; Pin Change Int Request 1 Handler
0x030 jmp PCINT2 ; Pin Change Int Request 2 Handler
0x032 jmp PCINT3 ; Pin Change Int Request 3 Handler
0x034 jmp SPI_STC ; SPI Transfer Complete Handler
0x036 jmp ADC ; ADC Conversion Complete Handler
0x038 jmp WDT ; Watchdog Timer Handler
0x03A jmp EE_RDY ; EEPROM Ready Handler
0x03C jmp SPM_RDY ; Store Program Memory Ready Handler
;
0x03ERESET: ldi r16, high(RAMEND); Main program start
0x03F out SPH,r16 ; Set Stack Pointer to top of RAM
0x040 ldi r16, low(RAMEND)
0x041 out SPL,r16
0x042 sei ; Enable interrupts
0x043 <instr> xxx
... ... ... ...
```

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2K 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
0x000 RESET: ldi r16,high(RAMEND); Main program start
0x001 out SPH,r16 ; Set Stack Pointer to top of RAM
0x002 ldi r16,low(RAMEND)
0x003 out SPL,r16
0x004 sei ; Enable interrupts
0x005 <instr> xxx
;
.org 0xC02
0xC02 jmp ANA_COMP_0 ; Analog Comparator 0 Handler
0xC04 jmp ANA_COMP_1 ; Analog Comparator 1 Handler
... ... ... ;
0xC3C jmp SPM_RDY ; Store Program Memory Ready Handler
```

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

```
Address Labels Code Comments
.org 0x002
0x002 jmp ANA_COMP_0 ; Analog Comparator 0 Handler
0x004 jmp ANA_COMP_1 ; Analog Comparator 1 Handler
... ... ... ;
0x03C jmp SPM_RDY ; Store Program Memory Ready Handler
;
.org 0xC00
0xC00 RESET: ldi r16,high(RAMEND); Main program start
0xC01 out SPH,r16 ; Set Stack Pointer to top of RAM
0xC02 ldi r16,low(RAMEND)
0xC03 out SPL,r16
0xC04 sei ; Enable interrupts
0xC05 <instr> xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 2K 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 0xC00
0xC00 jmp RESET ; Reset handler
0xC02 jmp ANA_COMP_0 ; Analog Comparator 0 Handler
0xC04 jmp ANA_COMP_1 ; Analog Comparator 1 Handler
... ... ... ;
0xC3C jmp SPM_RDY ; Store Program Memory Ready Handler
;
0xC3E RESET: ldi r16,high(RAMEND); Main program start
0xC3F out SPH,r16 ; Set Stack Pointer to top of RAM
0xC40 ldi r16,low(RAMEND)
0xC41 out SPL,r16
0xC42 sei ; Enable interrupts
0xC43 <instr> xxx
```

13.2 Register Description

13.2.1 Moving Interrupts Between Application and Boot Space

The MCU Control register controls the placement of the interrupt vector table.

13.2.2 MCU Control Register

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

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.

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

Bit 7 – SPIPS SPI Pin Redirection
Thanks to SPIPS (SPI Pin Select) in MCUCR Sfr, SPI pins can be redirected. Note that the programming port is always located on alternate SPI port.

Value	Description
0	When the SPIPS bit is written to zero, the SPI signals are directed on pins MISO, MOSI, SCK and SS
1	When the SPIPS bit is written to one,the SPI signals are directed on alternate SPI pins, MISO_A, MOSI_A, SCK_A and SS_A

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 same cycle as IVCE is written, and interrupts 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 programmed, 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 the 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);
}
```

14. External Interrupts (EXINT)

14.1 Overview

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 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 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.

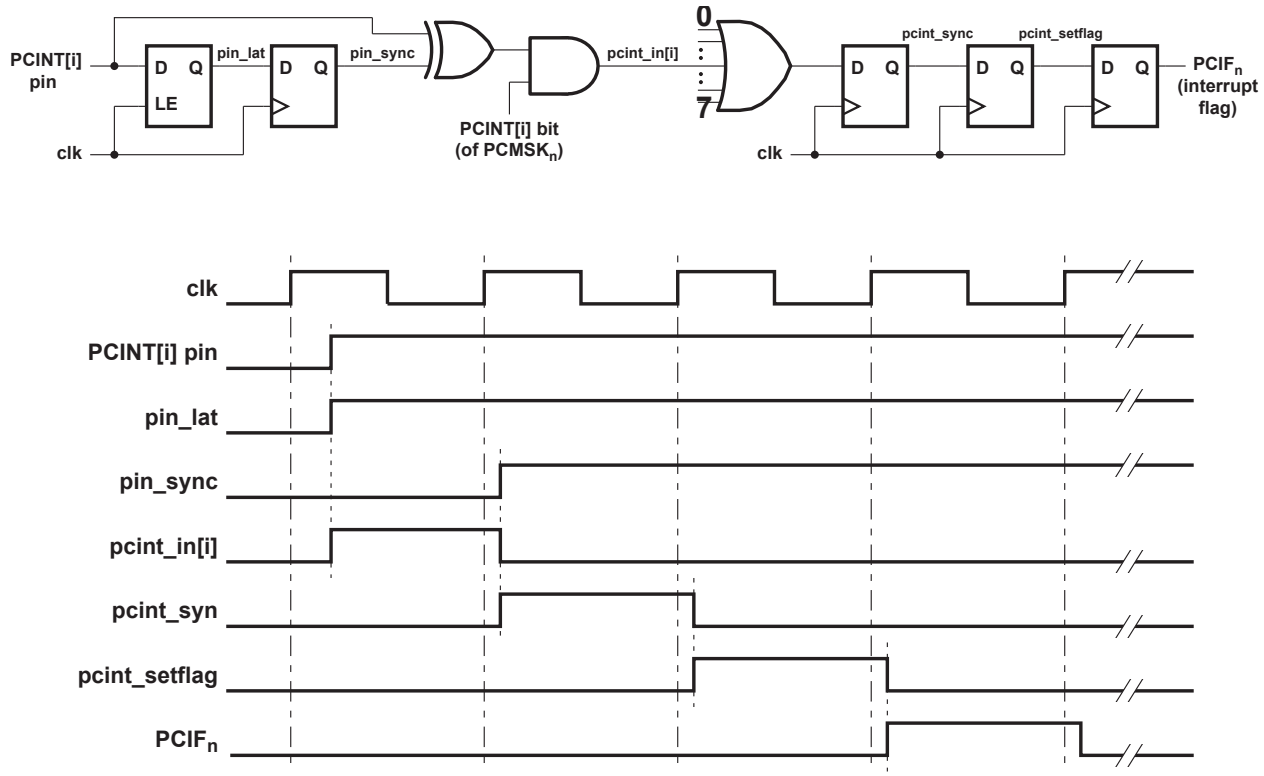
Related Links

System Clock and Clock Options

14.1.1 Pin Change Interrupt Timing

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

Figure 14-1. Timing of Pin Change Interrupts



14.2 Register Description

14.2.1 External Interrupt Control Register A

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

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

Bit	7	6	5	4	3	2	1	0
	ISC3[1:0]		ISC2[1:0]		ISC1[1:0]		ISC0[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

Bits 7:6 – ISC3[1:0] Interrupt Sense Control 3
The External Interrupt 3 is activated by the external pin INT3 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT3 pin that activate the interrupt are defined in table below. The value on the INT3 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 INT3 generates an interrupt request.
01	Any logical change on INT3 generates an interrupt request.
10	The falling edge of INT3 generates an interrupt request.
11	The rising edge of INT3 generates an interrupt request.

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.

14.2.2 External Interrupt Mask Register

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

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.

Bit	7	6	5	4	3	2	1	0
					INT3	INT2	INT1	INT0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – INT3 External Interrupt Request 3 Enable
When the INT3 bit is set and the I-bit in the Status Register (SREG) is set, the external pin interrupt is enabled. The Interrupt Sense Control3 bits 1/0 (ISC31 and ISC30) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT3 pin or level sensed. Activity on the pin will cause an interrupt request even if INT3 is configured as an output. The corresponding interrupt of External Interrupt Request 3 is executed from the INT3 Interrupt Vector.

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.

14.2.3 External Interrupt Flag Register

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

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.

Bit	7	6	5	4	3	2	1	0
					INTF3	INTF2	INTF1	INTF0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – INTF3 External Interrupt Flag 3
When an edge or logic change on the INT3 pin triggers an interrupt request, INTF3 will be set. If the I-bit in SREG and the INT3 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 INT3 is configured as a level interrupt.

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.

14.2.4 Pin Change Interrupt Control Register

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

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

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[27:24] pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI3 Interrupt Vector. PCINT[27:24] 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.

14.2.5 Pin Change Interrupt Flag Register

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

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.

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

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.

14.2.6 Pin Change Mask Register 3

Name: PCMSK3
Offset: 0x6D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						PCINT26	PCINT25	PCINT24
Access						R/W	R/W	R/W
Reset						0	0	0

Bits 0, 1, 2 – PCINT Pin Change Enable Mask
Each PCINT[26:24]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[26:24] is set and the PCIE3 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[26:24] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

14.2.7 Pin Change Mask Register 2

Name: PCMSK2
Offset: 0x6C
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 – PCINT 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.

14.2.8 Pin Change Mask Register 1

Name: PCMSK1
Offset: 0x6B
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 – PCINT 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.

14.2.9 Pin Change Mask Register 0

Name: PCMSK0
Offset: 0x6A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
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

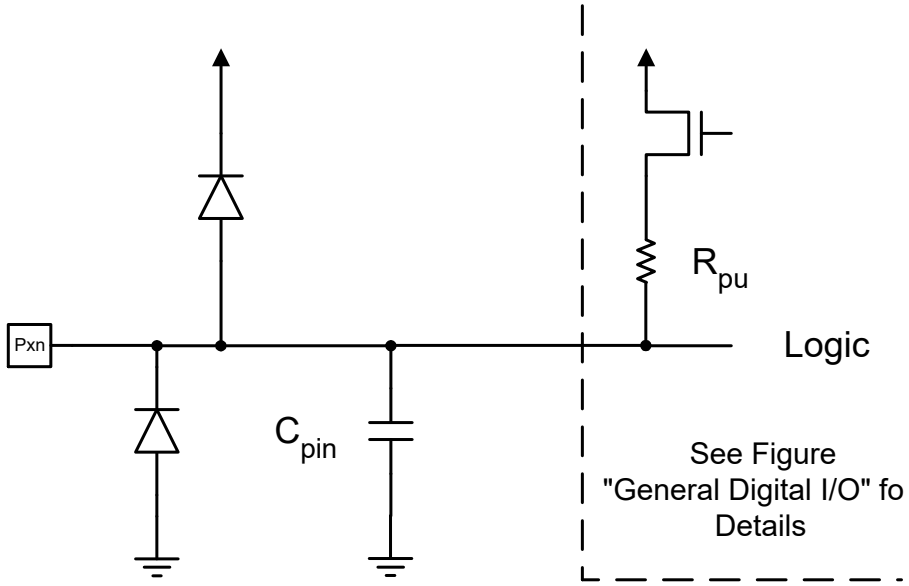
Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT Pin Change Enable Mask
Each PCINT[7:0] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is cleared,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

15. I/O-Ports

15.1 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 an output) or enabling/disabling of pull-up resistors (if configured as an 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 15-1. 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 number 3 in Port B, here documented generally as PORTxn.

Three I/O memory address locations are allocated for each port, one each for the Data Register (Portx), Data Direction Register (DDRx), and the Port Input Pins (PINx). The port input pins I/O location is read-only, while the data register and the data direction register are read/write. 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.

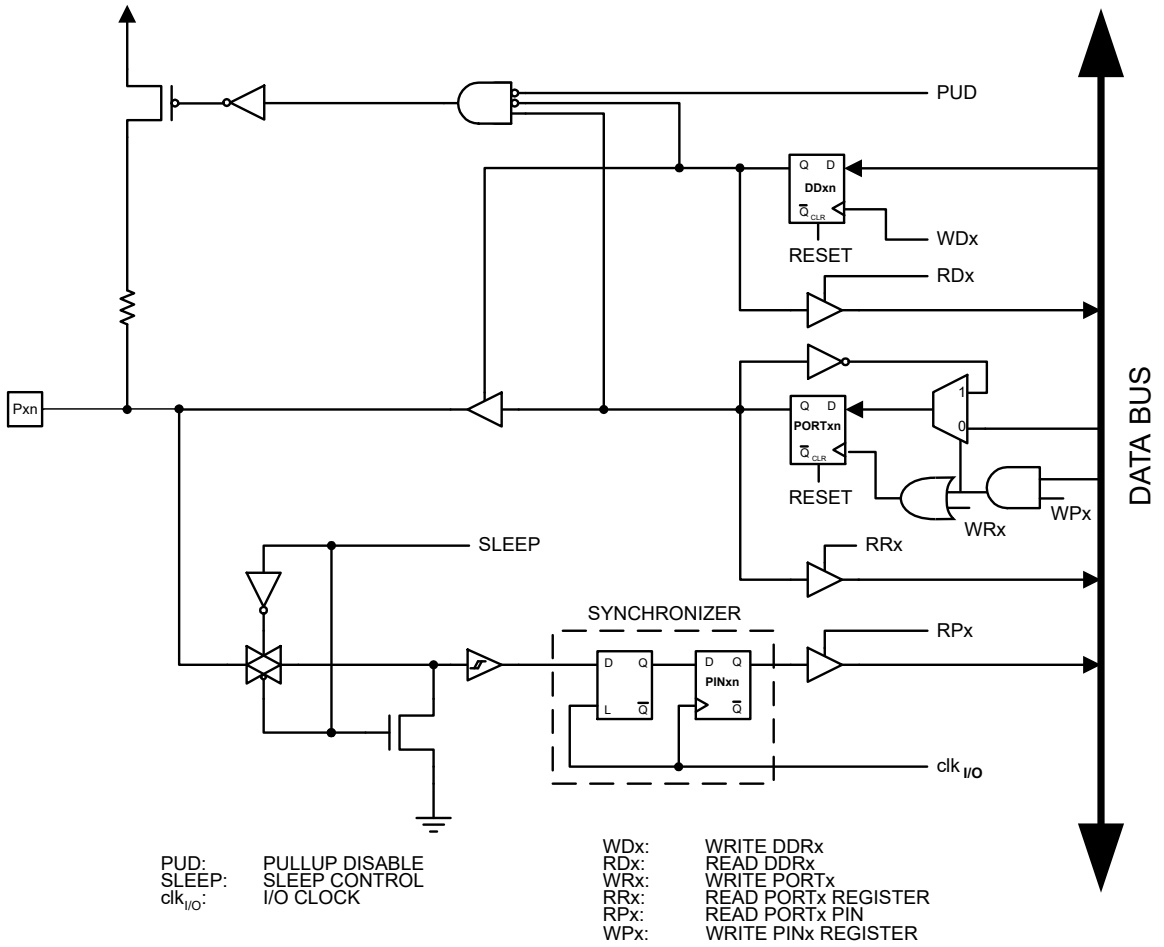
Using the I/O port as general digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer 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.

15.2 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 15-2. 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.

15.2.1 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 the 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.

15.2.2 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.

15.2.3 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.

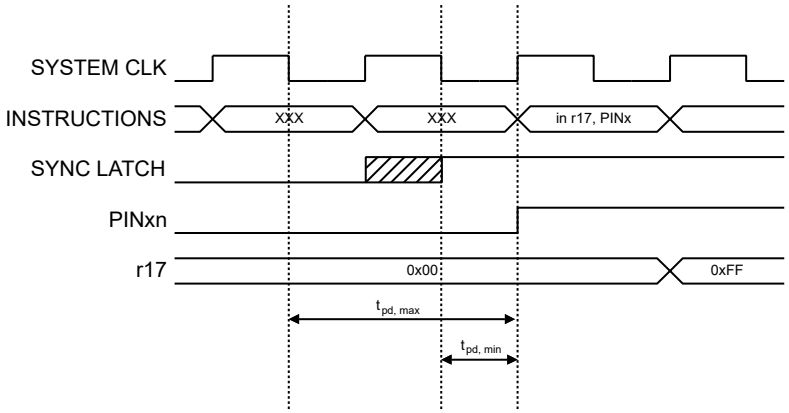
Table 15-1. Port Pin Configurations

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)

15.2.4 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.

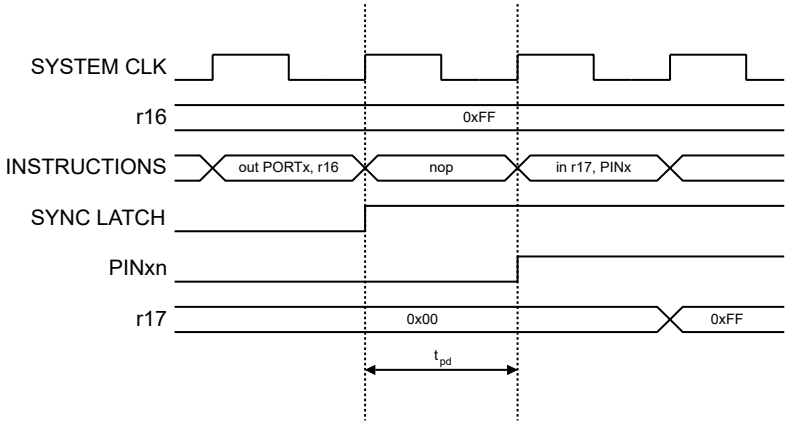
Figure 15-3. Synchronization when Reading an Externally Applied Pin value



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 $t_{pd,max}$ and $t_{pd,min}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software-assigned pin value, a nop instruction must be inserted as indicated in 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 one system clock period.

Figure 15-4. 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;
...
```

15.2.5 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 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.

15.2.6 Unconnected Pins

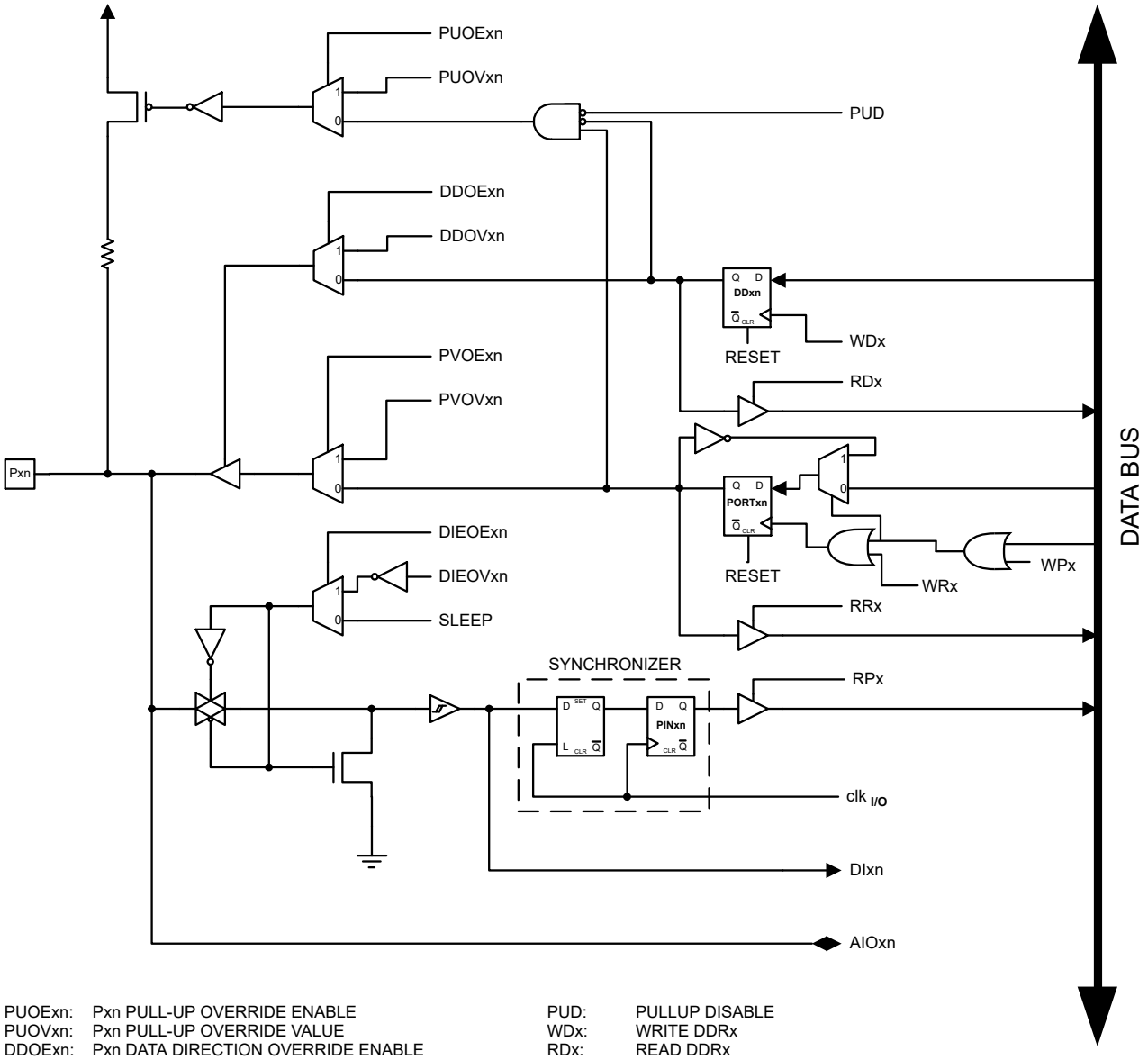
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

15.3 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 15-2 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 15-5. 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 the previous figure are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 15-2. 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.

15.3.1 Alternate Functions of Port B

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

Table 15-3. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	PSCOUT0B (PSC output 0B) ADC4 (Analog Input Channel 4) SCK (SPI Bus Serial Clock)

Port Pin	Alternate Functions
	PCINT7 (Pin Change Interrupt 7)
PB6	ADC7 (Analog Input Channel 7) PSCOUT1B (PSC output 1B) PCINT6 (Pin Change Interrupt 6)
PB5	ADC6 (Analog Input Channel 6) INT2 (External Interrupt 2) ACMPN1 (Analog Comparator 1 Negative Input) AMP2- (Analog Differential Amplicator 2 Negative Input) PCINT5 (Pin Change Interrupt 5)
PB4	AMP0+ (Analog Differential Amplifier 0 Positive Input) PCINT4 (Pin Change Interrupt 4)
PB3	AMP0- (Analog Differential Amplifier 0 Negative Input) PCINT3 (Pin Change Interrupt 3)
PB2	ADC5 (Analog Input Channel5) INT1 (External Interrupt 1) ACMPN0 (Analog Comparator 0 Negative Input) PCINT2 (Pin Change Interrupt 2)
PB1	MOSI (SPI Master Out Slave In) PSCOUT2B (PSC output 2B) PCINT1 (Pin Change Interrupt 1)
PB0	MISO (SPI Master In Slave Out) PSCOUT2A (PSC output 2A) PCINT0 (Pin Change Interrupt 0)

The alternate pin configuration is as follows:

- **ADC4/PSCOUT0B/SCK/PCINT7 – Bit 7**
  - PSCOUT0B, Output 0B of PSC.
  - ADC4, Analog to Digital Converter, input channel 4.
  - SCK, Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit.
  - PCINT7, Pin Change Interrupt 7.
- **ADC7/PSCOUT1B/PCINT6 – Bit 6**

  - ADC7, Analog to Digital Converter, input channel 7.
  - PSCOUT1B, Output 1B of PSC.
  - PCINT6, Pin Change Interrupt 6.
- **ADC6/INT2/ACMPN1/AMP2-/PCINT5 – Bit 5**
  - ADC6, Analog to Digital Converter, input channel 6.
  - INT2, External Interrupt source 2. This pin can serve as an External Interrupt source to the MCU.
  - ACMPN1, Analog Comparator 1 Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT5, Pin Change Interrupt 5.
- **APM0+/PCINT4 – Bit 4**
  - AMP0+, Analog Differential Amplifier 0 Positive Input Channel.
  - PCINT4, Pin Change Interrupt 4.
- **AMP0-/PCINT3 – Bit 3**
  - AMP0-, Analog Differential Amplifier 0 Negative Input Channel. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Amplifier.
  - PCINT3, Pin Change Interrupt 3.
- **ADC5/INT1/ACMPN0/PCINT2 – Bit 2**
  - ADC5, Analog to Digital Converter, input channel 5.
  - INT1, External Interrupt source 1. This pin can serve as an external interrupt source to the MCU.
  - ACMPN0, Analog Comparator 0 Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT2, Pin Change Interrupt 2.
- **PCINT1/MOSI/PSCOUT2B – Bit 1**
  - MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB1 When the SPI is enabled as a master, the data direction of this pin is controlled by DDB1. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB1 and PUD bits.
  - PSCOUT2B, Output 2B of PSC.
  - PCINT1, Pin Change Interrupt 1.
- **PCINT0/MISO/PSCOUT2A – Bit 0**
  - MISO, Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB0. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 and PUD bits.
  - PSCOUT2A, Output 2A of PSC.
  - PCINT0, Pin Change Interrupt 0.

The tables below relate the alternate functions of Port B to the overriding signals shown in Figure 15-5.

Table 15-4. Overriding Signals for Alternate Functions in PB7...PB4

Signal Name	PB7/ADC4/ PSCOUT0B/SCK/ PCINT7	PB6/ADC7/ PSCOUT1B/ PCINT6	PB5/ADC6/ INT2/ ACMPN1/ AMP2-/ PCINT5	PB4/AMP0+/ PCINT4
PUOE	SPE • $\overline{\text{MSTR}}$ • $\overline{\text{SPIPS}}$	0	0	0
PUOV	PB7 • $\overline{\text{PUD}}$ • $\overline{\text{SPIPS}}$	0	0	0
DDOE	SPE • $\overline{\text{MSTR}}$ • $\overline{\text{SPIPS}}$ + PSCEN01	PSCEN11	0	0
DDOV	PSCEN01	1	0	0
PVOE	SPE • MSTR • $\overline{\text{SPIPS}}$	PSCEN11	0	0
PVOV	PSCOUT01 • $\overline{\text{SPIPS}}$ + PSCOUT01 • PSCEN01 • $\overline{\text{SPIPS}}$ + PSCOUT01 • PSCEN01 • $\overline{\text{SPIPS}}$	PSCOUT11	0	0
DIEOE	ADC4D	ADC7D	ADC6D + IN2EN	AMP0ND
DIEOV	0	0	IN2EN	0
DI	SCKIN • $\overline{\text{SPIPS}}$ • $\overline{\text{IRESET}}$	ICP1B	INT2	–
AIO	ADC4	ADC7	ADC6	AMP0+

Table 15-5. 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	–	–
PUOV	0	0	–	–
DDOE	0	0	–	–
DDOV	0	0	–	–
PVOE	0	0	–	–
PVOV	0	0	–	–
DIEOE	AMP0ND	ADC5D + IN1EN	0	0
DIEOV	0	IN1EN	0	0
DI	0	INT1	MOSI_IN • $\overline{\text{SPIPS}}$ • $\overline{\text{IRESET}}$	MISO_IN • $\overline{\text{SPIPS}}$ • $\overline{\text{IRESET}}$
AIO	AMP0-	ADC5	-	-

15.3.2 Alternate Functions of Port C

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

Table 15-6. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	D2A (DAC output) AMP2+ (Analog Differential Amplifier 2 Positive Input) PCINT15 (Pin Change Interrupt 15)
PC6	ADC10 (Analog Input Channel 10) ACMP1 (Analog Comparator 1 Positive Input) PCINT14 (Pin Change Interrupt 14)
PC5	ADC9 (Analog Input Channel 9) AMP1+ (Analog Differential Amplifier 1 Input Channel) ACMP3 (Analog Comparator 3 Positive Input) PCINT13 (Pin Change Interrupt 13)
PC4	ADC8 (Analog Input Channel 8) AMP1- (Analog Differential Amplifier 1 Input Channel) ACMPN3 (Analog Comparator 3 Negative Input) PCINT12 (Pin Change Interrupt 12)
PC3	T1 (Timer 1 clock input) RXCAN (CAN Rx Data) ICP1B (Timer 1 input capture alternate input) PCINT11 (Pin Change Interrupt 11)
PC2	T0 (Timer 0 clock input) TXCAN (CAN Tx Data) PCINT10 (Pin Change Interrupt 10)
PC1	PSCIN1 (PSC 1 Digital Input) OC1B (Timer 1 Output Compare B) SS_A (Alternate SPI Slave Select) PCINT9 (Pin Change Interrupt 9)
PC0	PSCOUT1A (PSC output 2A) INT3 (External Interrupt 3) PCINT8 (Pin Change Interrupt 8)

The alternate pin configuration is as follows:

- **D2A/AMP2+/PCINT15 – Bit 7**

  - D2A, Digital to Analog output.
  - AMP2+, Analog Differential Amplifier 2 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Amplifier.
  - PCINT15, Pin Change Interrupt 15.
- **ADC10/ACMP1/PCINT14 – Bit 6**
  - ADC10, Analog to Digital Converter, input channel 10.
  - ACMP1, Analog Comparator 1 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT14, Pin Change Interrupt 14

.**ADC9/ACMP3/AMP1+/PCINT13 – Bit 5**
  - ADC9, Analog to Digital Converter, input channel 9.
  - ACMP3, Analog Comparator 3 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - AMP1+, Analog Differential Amplifier 1 Positive Input Channel. Configure the port pin as input with the internal pullup switched off to avoid the digital port function from interfering with the function of the Analog Amplifier.
  - PCINT13, Pin Change Interrupt 13.
- **ADC8/AMP1-/ACMPN3/PCINT12 – Bit 4**
  - ADC8, Analog to Digital Converter, input channel 8.
  - AMP1-, Analog Differential Amplifier 1 Negative Input Channel. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Amplifier.
  - ACMPN3, Analog Comparator 3 Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT12, Pin Change Interrupt 12.
- **PCINT11/T1/RXCAN/ICP1B – Bit 3**
  - T1, Timer/Counter1 counter source.
  - RXCAN, CAN Rx Data.
  - ICP1B, Input Capture Pin: The PC3 pin can act as an Input Capture Pin for Timer/Counter1.
  - PCINT11, Pin Change Interrupt 11.
- **PCINT10/T0/TXCAN – Bit 2**
  - T0, Timer/Counter0 counter source.
  - TXCAN, CAN Tx Data.
  - PCINT10, Pin Change Interrupt 10.
- **PCINT9/PSCIN1/OC1B/SS_A – Bit 1**
  - PCSIN1, PSC 1 Digital Input.
  - OC1B, Output Compare Match B output: This pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDC1 set “one”) to serve this function. This pin is also the output pin for the PWM mode timer function.
  - SS_A: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD0. As a slave, the SPI is activated when this pin is

driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDD0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD0 bit.

  - PCINT9, Pin Change Interrupt 9.

- **PCINT8/PSCOUT1A/INT3 – Bit 0**
  - PSCOUT1A, Output 1A of PSC.
  - INT3, External Interrupt source 3: This pin can serve as an external interrupt source to the MCU.
  - PCINT8, Pin Change Interrupt 8.

The tables below relate the alternate functions of Port C to the overriding signals shown in Figure 15-5.

Table 15-7. Overriding Signals for Alternate Functions in PC[7:4]

Signal Name	PC7/D2A/AMP2+/ PCINT15	PC6/ADC10/ ACMP1/ PCINT14	PC5/ADC9/ AMP1+/ ACMP3/ PCINT13	PC4/ADC8/ AMP1-/ ACMPN3/ PCINT12
PUOE	0	0	0	–
PUOV	0	0	0	–
DDOE	DAEN	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	–
PVOV	0	0	0	–
DIEOE	DAEN	ADC10D	ADC9D	ADC8D
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	–	ADC10 AMP1	ADC9 AMP1+	ADC8 AMP1- ACMPN3

Table 15-8. Overriding Signals for Alternate Functions in PC[3:0]

Signal Name	PC3/T1/RXCAN/ ICP1B/PCINT11	PC2/T0/TXCAN/ PCINT10	PC1/PSCIN1/ OC1B/ SS_A/ PCINT9	PC0/INT3/ PSCOUT1A/ PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	–	–	0	PSCEN10
DDOV	1	1	0	1
PVOE	–	–	OC1BEN	PSCEN10
PVOV	–	–	OC1B	PSCOUT10
DIEOE	–	–	–	IN3EN
DIEOV	–	–	–	IN3EN

Signal Name	PC3/T1/RXCAN/ ICP1B/PCINT11	PC2/T0/TXCAN/ PCINT10	PC1/PSCIN1/ OC1B/ SS_A/ PCINT9	PC0/INT3/ PSCOUT1A/ PCINT8
DI	T1	T0	PSCIN1 SS_A	INT3
AIO	–	–	–	–

15.3.3 Alternate Functions of Port D

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

Table 15-9. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	ACMP0 (Analog Comparator 0 Positive Input) PCINT23 (Pin Change Interrupt 23)
PD6	ADC3 (Analog Input Channel 3) ACMPN2 (Analog Comparator 2 Negative Input) INT0 (External Interrupt 0) PCINT22 (Pin Change Interrupt 22)
PD5	ADC2 (Analog Input Channel 2) ACMP2 (Analog Comparator 2 Positive Input) PCINT21 (Pin Change Interrupt 21)
PD4	ADC1 (Analog Input Channel 1) RXD/RXLIN (LIN/UART Rx data) ICP1A (Timer 1 input capture) SCK_A (Programming & alternate SPI Clock) PCINT20 (Pin Change Interrupt 20)
PD3	TXD/TXLIN (LIN/UART Tx data) OC0A (Timer 0 Output Compare A) SS (SPI Slave Select) MOSI_A (Programming & alternate SPI Master Out Slave In) PCINT19 (Pin Change Interrupt 19)
PD2	PSCIN2 (PSC Digital Input 2) OC1A (Timer 1 Output Compare A) MISO_A (Programming & alternate Master In SPI Slave Out) PCINT18 (Pin Change Interrupt 18)
PD1	PSCIN0 (PSC Digital Input 0)

Port Pin	Alternate Function
	CLKO (System Clock Output) PCINT17 (Pin Change Interrupt 17)
PD0	PSCOUT0A (PSC output 0A) PCINT16 (Pin Change Interrupt 16)

The alternate pin configuration is as follows:

- **ACMP0/PCINT23 – Bit 7**
  - ACMP0, Analog Comparator 0 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT23, Pin Change Interrupt 23.
- **ADC3/ACMPN2/INT0/PCINT22 – Bit 6**
  - ADC3, Analog to Digital Converter, input channel 3.
  - ACMPN2, Analog Comparator 2 Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - INT0, External Interrupt source 0. This pin can serve as an external interrupt source to the MCU.
  - PCINT22, Pin Change Interrupt 23.
- **ADC2/ACMP2/PCINT21 – Bit 5**
  - ADC2, Analog to Digital Converter, input channel 2.
  - ACMP2, Analog Comparator 1 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT21, Pin Change Interrupt 21.
- **PCINT20/ADC1/RXD/RXLIN/ICP1/SCK_A – Bit 4**
  - ADC1, Analog to Digital Converter, input channel 1.
  - RXD/RXLIN, LIN/UART Receive Pin. Receive Data (Data input pin for the LIN/UART). When the LIN/UART receiver is enabled this pin is configured as an input regardless of the value of DDRD4. When the UART forces this pin to be an input, a logical one in PORTD4 will turn on the internal pull-up.
  - ICP1, Input Capture Pin1: This pin can act as an input capture pin for Timer/Counter1.
  - SCK_A: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD4. When the SPI is enabled as a master, the data direction of this pin is controlled by DDD4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD4 bit.
  - PCINT20, Pin Change Interrupt 20.
- **PCINT19/TXD/TXLIN/OC0A/SS/MOSI_A, Bit 3**
  - TXD/TXLIN, LIN/UART Transmit pin. Data output pin for the LIN/UART. When the LIN/UART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD3.

  - OC0A, Output Compare Match A output: This pin can serve as an external output for the Timer/Counter0 Output Compare A. The pin has to be configured as an output (DDD3 set “one”) to serve this function. The OC0A pin is also the output pin for the PWM mode.
  - SS: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD3. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDD3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD3 bit.
  - MOSI_A: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD3 When the SPI is enabled as a master, the data direction of this pin is controlled by DDD3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD3 bit.
  - PCINT19, Pin Change Interrupt 19.
- **PCINT18/PSCIN2/OC1A/MISO_A, Bit 2**
  - PCSIN2, PSC Digital Input 2.
  - OC1A, Output Compare Match A output: This pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD2 set “one”) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.
  - MISO_A: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDD2. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDD2. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD2 bit.
  - PCINT18, Pin Change Interrupt 18.
- **PCINT17/PSCIN0/CLKO – Bit 1**
  - PCSIN0, PSC Digital Input 0.
  - CLKO, Divided System Clock: The divided system clock can be output on this pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTD1 and DDD1 settings. It will also be output during reset.
  - PCINT17, Pin Change Interrupt 17.
- **PCINT16/PSCOUT0A – Bit 0**
  - PSCOUT0A: Output 0 of PSC 0.
  - PCINT16, Pin Change Interrupt 16.

The tables below relate the alternate functions of Port D to the overriding signals shown in Figure 15-5.

Table 15-10. Overriding Signals for Alternate Functions PD[7:4]

Signal Name	PD7/ ACMP0/ PCINT23	PD6/ADC3/ ACMPN2/ INT0/ PCINT22	PD5/ADC2/ ACMP2/ PCINT21	PD4/ADC1/RXD/ RXLIN/ICP1A/ SCK_A/PCINT20
PUOE	0	0	0	RXEN + SPE • $\overline{\text{MSTR}}$ • SPIPS
PUO	0	0	0	PD4 • PUD
DDOE	0	0	0	RXEN + SPE • $\overline{\text{MSTR}}$ • SPIPS
DDOV	0	0	0	0
PVOE	0	0	0	SPE • MSTR • SPIPS

Signal Name	PD7/ ACMP0/ PCINT23	PD6/ADC3/ ACMPN2/ INT0/ PCINT22	PD5/ADC2/ ACMP2/ PCINT21	PD4/ADC1/RXD/ RXLIN/ICP1A/ SCK_A/PCINT20
PVOV	0	0	0	–
DIEOE	ACMP0D	ADC3D + IN0EN	ADC2D	ADC1D
DIEOV	0	IN0EN	0	0
DI	–	INT0	–	ICP1A
AIO	ACOMP0	ADC3 ACMPM	ADC2 ACOMP2	ADC1

Table 15-11. Overriding Signals for Alternate Functions in PD[3:0]

Signal Name	PD3/TXD/TXLIN/ OC0A/SS/MOSI_A/ PCINT19	PD2/PSCIN2/ OC1A/MISO_A/ PCINT18	PD1/PSCIN0/ CLKO/ PCINT17	PD0/PSCOUT0A/ XCK/ PCINT16
PUOE	TXEN + SPE • $\overline{\text{MSTR}}$ • SPIPS	–	0	SPE • $\overline{\text{MSTR}}$ • SPIPS
PUO	$\overline{\text{TXEN}}$ • SPE • $\overline{\text{MSTR}}$ • $\overline{\text{SPIPS}}$ • PD3 • PUD	–	0	PD0 • $\overline{\text{PUD}}$
DDOE	TXEN + SPE • $\overline{\text{MSTR}}$ • SPIPS	–	0	$\overline{\text{PSCEN00}}$ + SPE • MSTR • SPIPS
DDOV	TXEN	0	0	PSCEN00
PVOE	TXEN + OC0EN + SPE • MSTR • SPIPS	–	0	PSCEN00 + UMSEL
PVOV	TXEN • TXD + $\overline{\text{TXEN}}$ • (OC0EN • OC0 + OC0EN • SPIPS • MOSI)	–	0	–
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SS MOSI_AIN	–	–	–
AIO	–	–	–	–

15.3.4 Alternate Functions of Port E

The Port E pins with alternate functions are shown in the table below.

Table 15-12. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE2	XTAL2 (XTAL output) ADC0 (Analog Input Channel 0) PCINT26 (Pin Change Interrupt 26)
PE1	XTAL1 (XTAL input) OC0B (Timer 0 Output Compare B)

Port Pin	Alternate Function
	PCINT25 (Pin Change Interrupt 25)
PE0	RESET# (Reset input) OCD (On Chip Debug I/O) PCINT24 (Pin Change Interrupt 24)

The alternate pin configuration is as follows:

- **PCINT26/XTAL2/ADC0 – Bit 2**
  - XTAL2: Chip clock Oscillator pin 2. Used as clock pin for crystal Oscillator or Low-frequency crystal Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.
  - ADC0, Analog to Digital Converter, input channel 0.
  - PCINT26, Pin Change Interrupt 26.
- **PCINT25/XTAL1/OC0B – 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.
  - OC0B, Output Compare Match B output: This pin can serve as an external output for the Timer/Counter0 Output Compare B. The pin has to be configured as an output (DDE1 set “one”) to serve this function. This pin is also the output pin for the PWM mode timer function.
  - PCINT25, Pin Change Interrupt 25.
- **PCINT24/RESET/OCD – Bit 0**
  - RESET, Reset pin: When the RSTDISBL Fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on Power-on Reset and Brown-out Reset as its reset sources. When the RSTDISBL Fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.
  - If PE0 is used as a reset pin, DDE0, PORTE0 and PINE0 will all read 0.
  - PCINT24, Pin Change Interrupt 24.

The table below relates the alternate functions of Port E to the overriding signals shown in Figure 15-5.

Table 15-13. Overriding Signals for Alternate Functions in PE2...PE0

Signal Name	PE2/ADC0/XTAL2/ PCINT26	PE1/XTAL1/OC0B/ PCINT25	PE0/$\overline{\text{RESET}}$/ OCD/PCINT24
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	0	0
DDOV	0	0	0
PVOE	0	OC0BEN	0
PVOV	0	OC0B	0
DIEOE	ADC0D	0	0
DIEOV	0	0	0

Signal Name	PE2/ADC0/XTAL2/ PCINT26	PE1/XTAL1/OC0B/ PCINT25	PE0/$\overline{\text{RESET}}$/ OCD/PCINT24
DI			
AIO	Osc output ADC0	Osc/Clock input	

15.4 Register Description

15.4.1 MCU Control Register

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

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.

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

Bit 7 – SPIPS SPI Pin Redirection
Thanks to SPIPS (SPI Pin Select) in MCUCR Sfr, SPI pins can be redirected. Note that the programming port is always located on alternate SPI port.

Value	Description
0	When the SPIPS bit is written to zero, the SPI signals are directed on pins MISO, MOSI, SCK and SS
1	When the SPIPS bit is written to one,the SPI signals are directed on alternate SPI pins, MISO_A, MOSI_A, SCK_A and SS_A

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 same cycle as IVCE is written, and interrupts 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 programmed, 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 the 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);
}
```

15.4.2 Port B Data Register

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

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.

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

15.4.3 Port B Data Direction Register

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

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.

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.

15.4.4 Port B Input Pins Address

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

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.

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 I/O. Refer to Toggling the Pin.

15.4.5 Port C Data Register

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

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.

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

15.4.6 Port C Data Direction Register

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

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.

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.

15.4.7 Port C Input Pins Address

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

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.

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 I/O. Refer to Toggling the Pin.

15.4.8 Port D Data Register

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

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.

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

15.4.9 Port D Data Direction Register

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

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.

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.

15.4.10 Port D Input Pins Address

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

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.

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 I/O. Refer to Toggling the Pin.

15.4.11 Port E Data Register

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

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.

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

Bits 0, 1, 2 – PORTE Port E Data

15.4.12 Port E Data Direction Register

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

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.

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

Bits 0, 1, 2 – 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.

15.4.13 Port E Input Pins Address

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

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.

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

Bits 0, 1, 2 – PINE Port E Input Pins Address
Writing to the pin register provides toggle functionality for I/O. Refer to Toggling the Pin.

16. 8-bit Timer/Counter0 (TC0) with PWM

16.1 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)

16.2 Overview

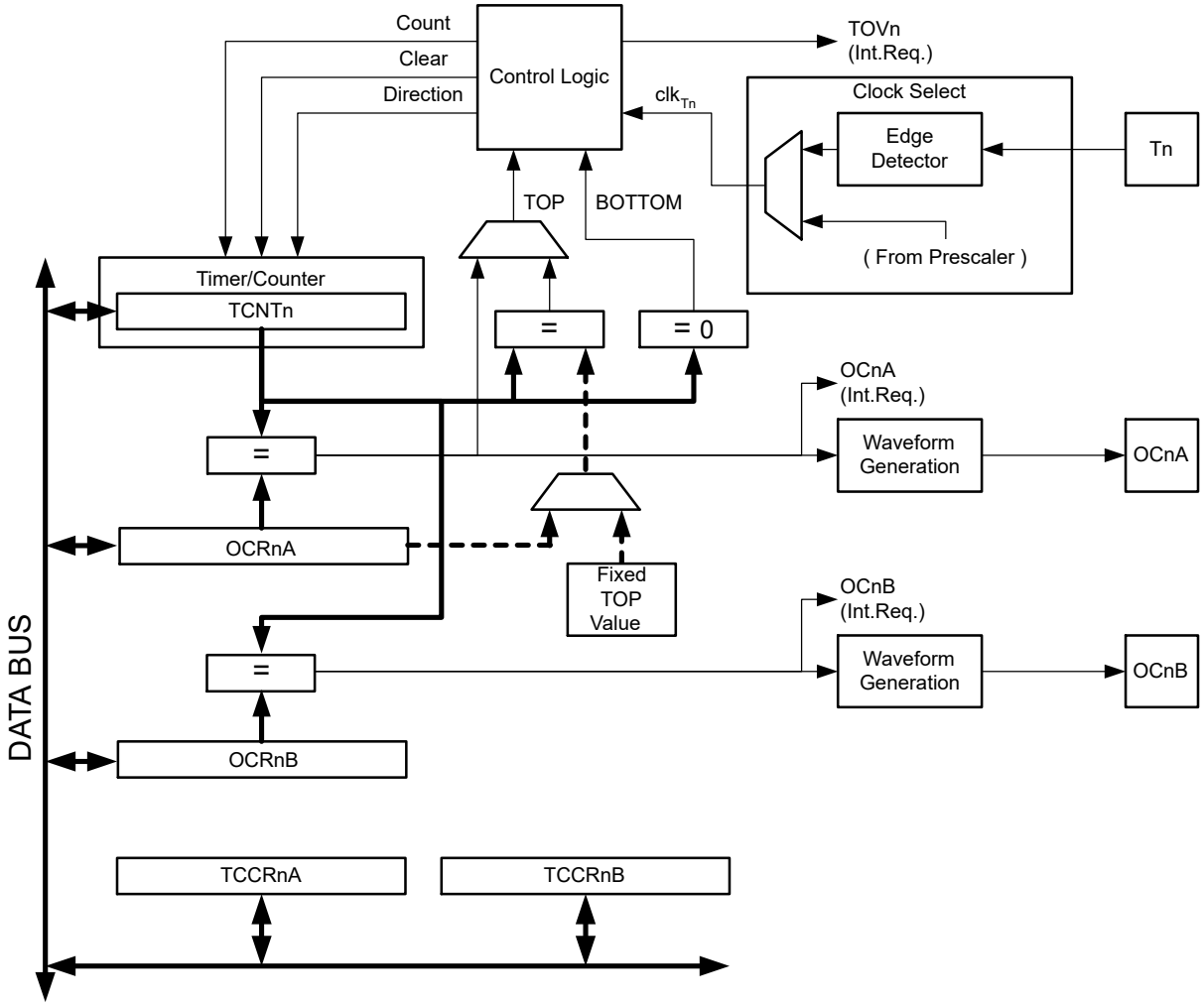
Timer/Counter0 (TC0) is a general purpose 8-bit timer/counter module, with two independent output compare units, and PWM support. It allows accurate program execution timing (event management) and wave generation.

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 (PRR.PRTIM0) is written to '1'.

Figure 16-1. 8-bit Timer/Counter Block Diagram



16.2.1 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 16-1. 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.

16.2.2 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 timer/counter (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 are 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 Prescalers

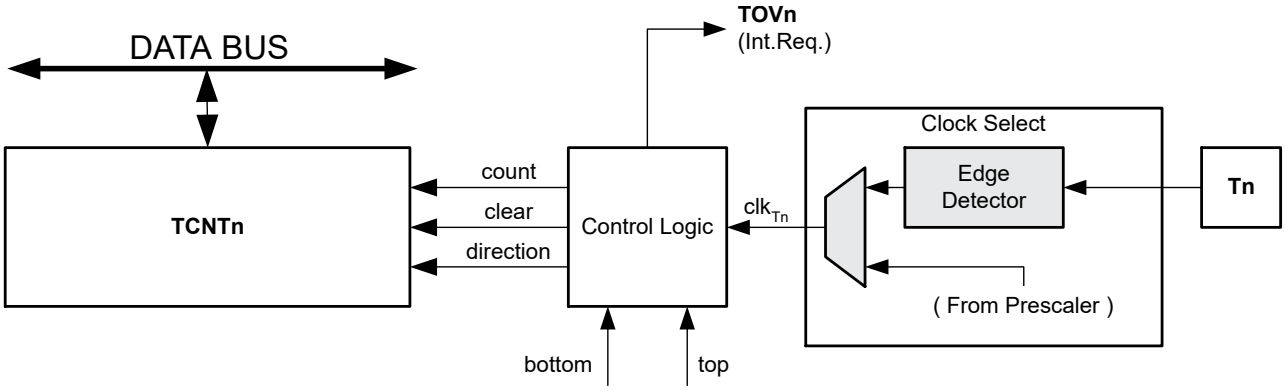
16.3 Timer/Counter Clock Sources

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).

16.4 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 16-2. 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 16-2. 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 on 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.

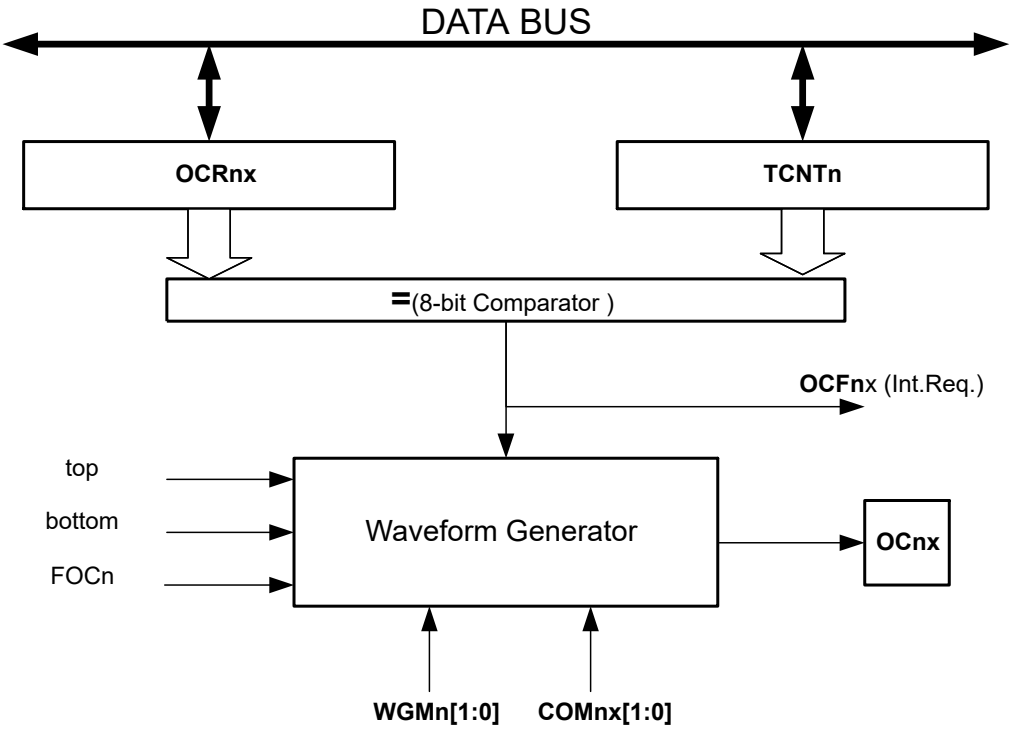
The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM0[2:0] bits. TOV0 can be used for generating a CPU interrupt.

16.5 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 maximum and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation.

Figure 16-3. 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.

16.5.1 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).

16.5.2 Compare Match Blocking by TCNTn Write

All CPU write operations to the TCNTn register will block any compare match that occurs 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.

16.5.3 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 counting down.

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.

16.6 Compare Match Output Unit

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

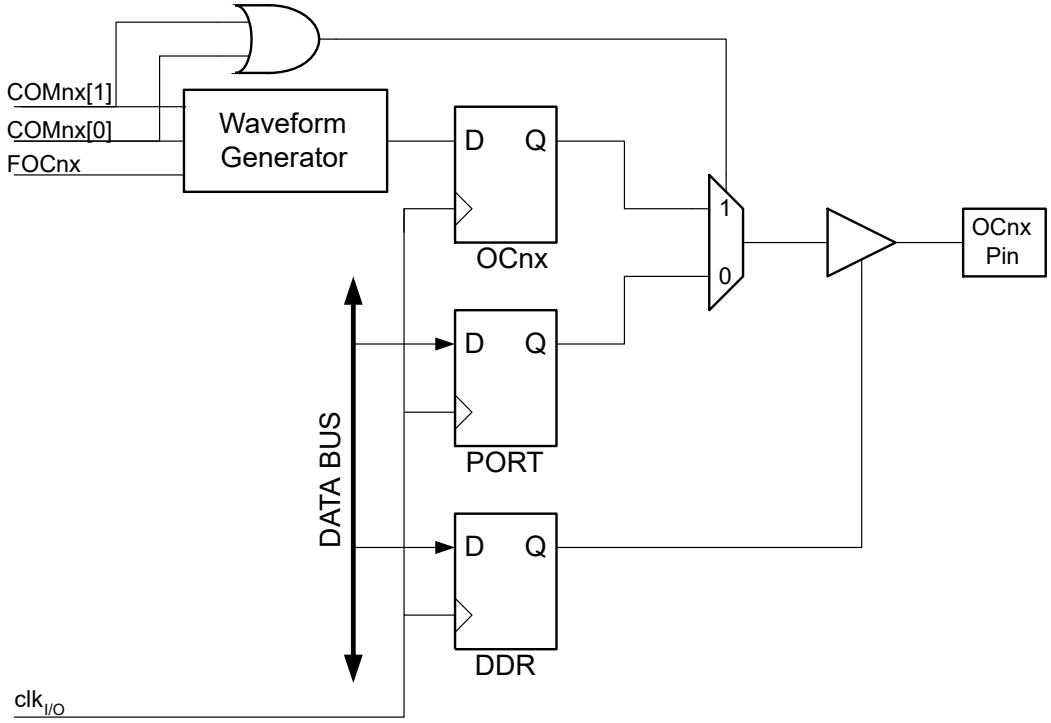
- The waveform generator uses the COM0x bits for defining the Output Compare (OC0x) register state at the next compare match.
- The COM0x bits control the OC0x pin output source

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 16-4. 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 DDR, the bit for the OC1x 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

16.6.1 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 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.

16.7 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 the previous section *Compare Match Output Unit*).

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

Related Links

Timer/Counter Timing Diagrams

16.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM[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 (TOV) will be set in the same clock cycle in which the TCNT becomes zero. In this case, the TOV 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 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 any time.

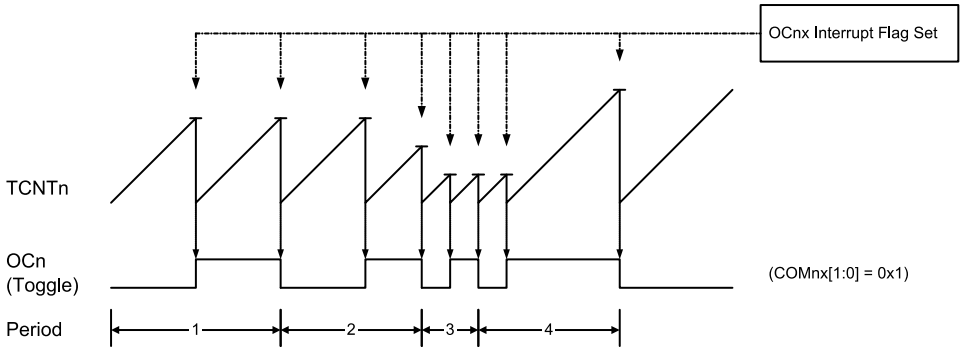
The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended since this will occupy too much of the CPU time.

16.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare (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 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 16-5. 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 an 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.

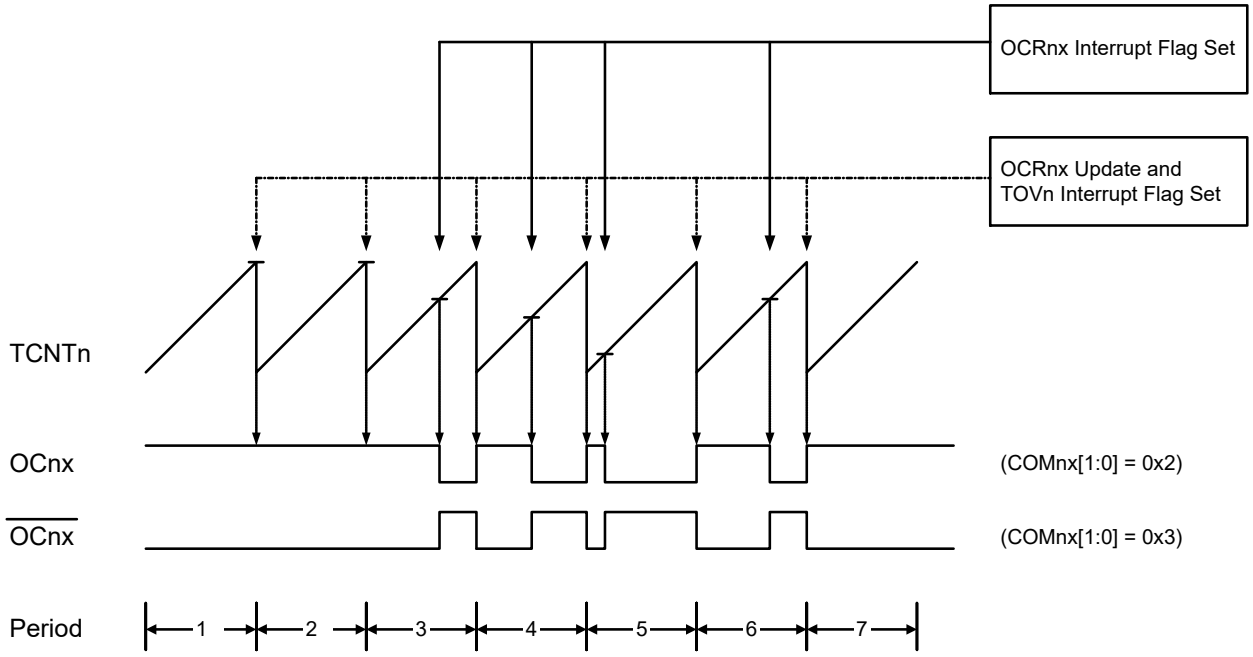
16.7.3 Fast PWM Mode

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 and 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 16-6. 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 represent 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.

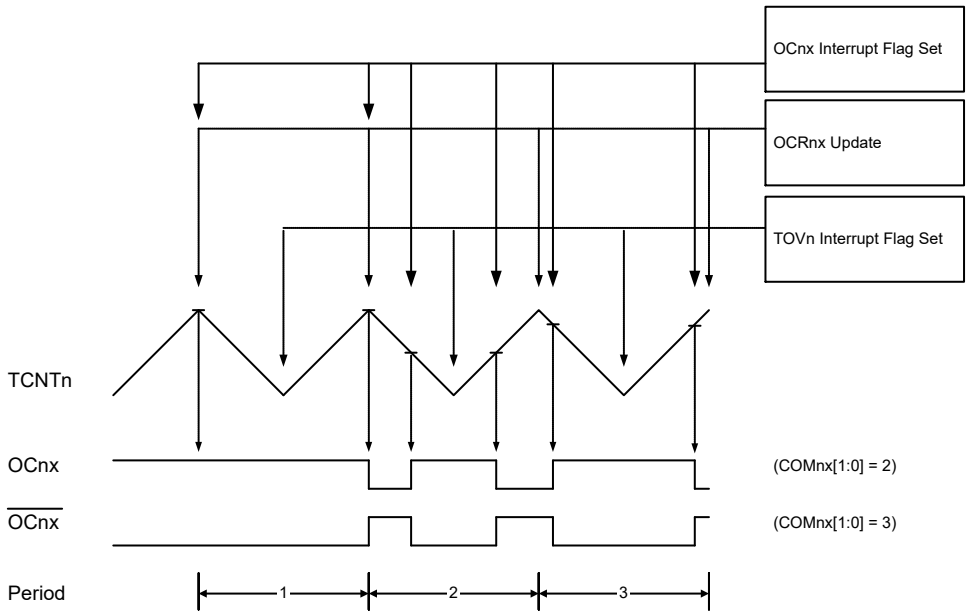
16.7.4 Phase Correct PWM Mode

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 16-7. 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:

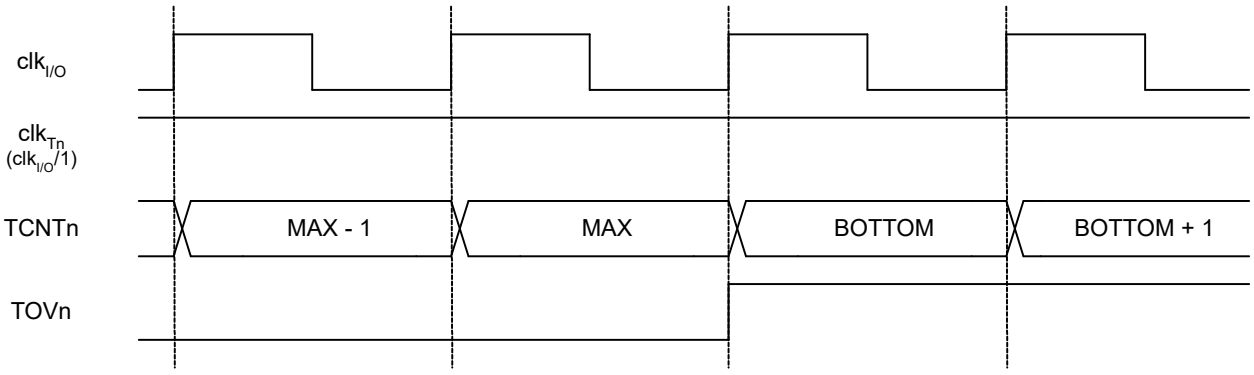
- OCR0x changes its value from MAX, as in the timing diagram. When the OCR0A value is MAX, the OC0 pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OC0x value at MAX must correspond to the result of an up-counting compare match.
- The timer starts 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.

16.8 Timer/Counter Timing Diagrams

The timer/counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. If the given instance of the TC0 supports an Asynchronous mode, $clk_{I/O}$ 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.

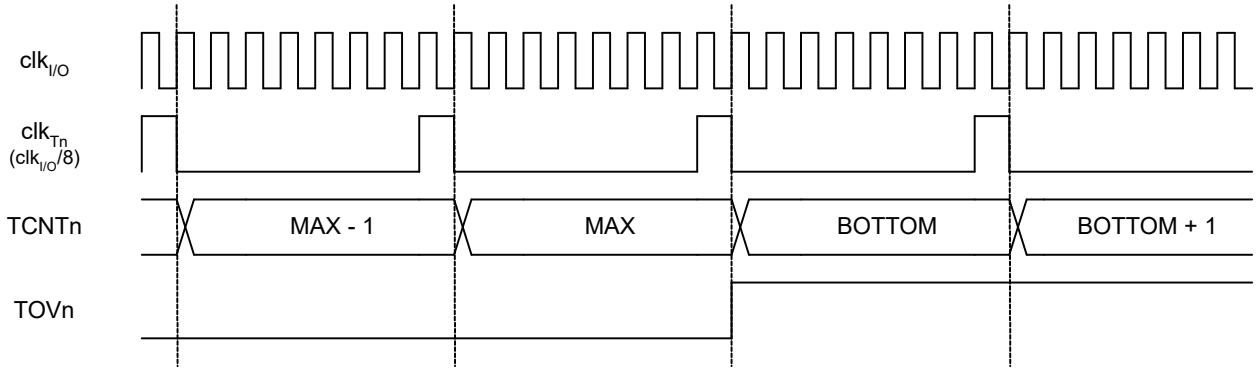
Figure 16-8. Timer/Counter Timing Diagram, no Prescaling



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.

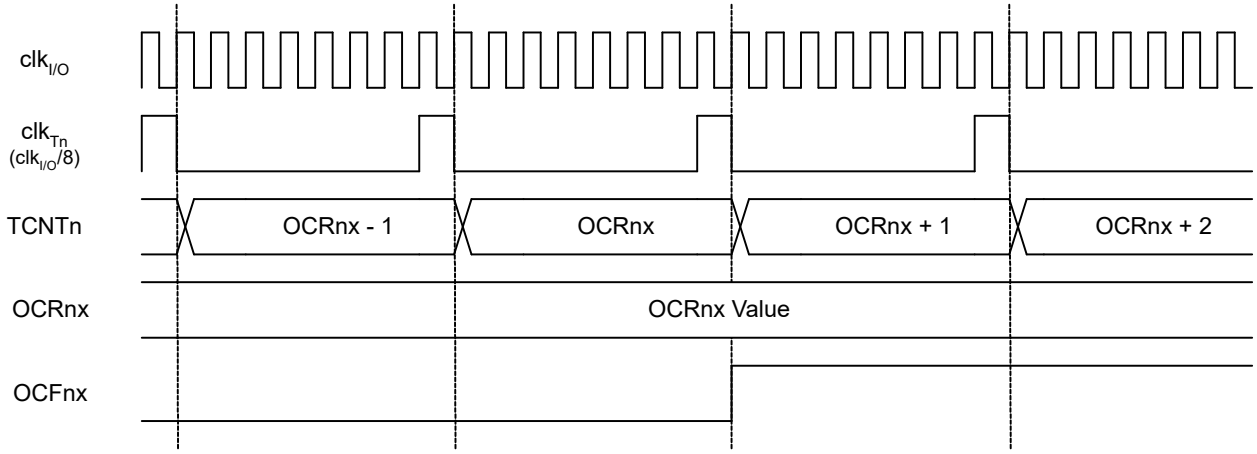
Figure 16-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



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).

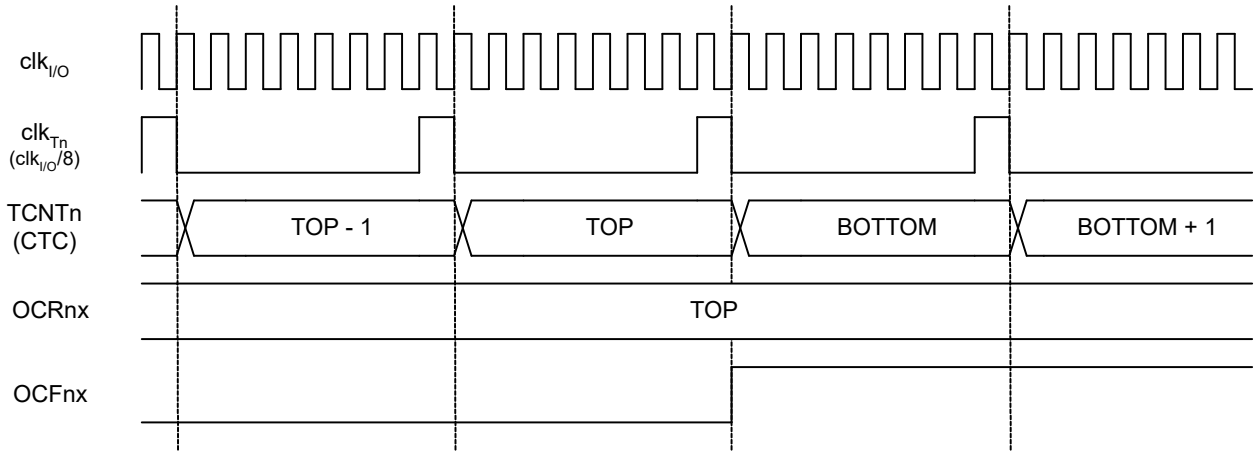
Figure 16-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)



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 16-11. 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).

16.9 Register Description

16.9.1 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 16-3. Compare Output Mode, Non-PWM

COM0A[1]	COM0A[0]	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match.
1	0	Clear OC0A on compare match.
1	1	Set OC0A on compare match.

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

Table 16-4. Compare Output Mode, Fast PWM(1)

COM0A[1]	COM0A[0]	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM0[2:0]: Normal port operation, OC0A disconnected. WGM0[2:1]: 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 COM0A[1] 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 16-5. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A[1]	COM0A[0]	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM0[2:0]: Normal port operation, OC0A disconnected. WGM0[2:1]: 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 COM0A[1] 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 16-6. Compare Output Mode, Non-PWM

COM0B[1]	COM0B[0]	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 16-7. Compare Output Mode, Fast PWM(1)

COM0B[1]	COM0B[0]	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved.

COM0B[1]	COM0B[0]	Description
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 16-8. Compare Output Mode, Phase Correct PWM Mode(1)

COM0B[1]	COM0B[0]	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 COM0B[1] 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 16-9. Waveform Generation Mode Bit Description

Mode	WGM0[2]	WGM0[1]	WGM0[0]	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	OCR0A	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCR0A	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCR0A	BOTTOM	TOP

Note:

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

16.9.2 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			WGM02	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 – WGM02 Waveform Generation Mode
Refer to TCCR0A register.

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 16-10. Clock Select Bit Description

CS0[2]	CS0[1]	CS0[0]	Description
0	0	0	No clock source (timer/counter stopped)
0	0	1	$clk_{I/O}$/1 (no prescaling)

CS0[2]	CS0[1]	CS0[0]	Description
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.

16.9.3 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.

16.9.4 General Timer/Counter Control Register

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

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.

Bit	7	6	5	4	3	2	1	0
	TSM	ICPSEL1						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 PSRSYNC bit 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, hardware clears the PSRSYNC bit, and the timer/counters start counting simultaneously.

Bit 6 – ICPSEL1 Timer 1 Input Capture selection
Timer 1 capture function has two possible inputs ICP1A (PD4) and ICP1B (PC3).

Value	Description
0	Select ICP1A as trigger for timer 1 input capture
1	Select ICP1B as trigger for timer 1 input capture

Bit 0 – PSRSYNC Prescaler Reset
When this bit is one, timer/counter 0, 1 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 share the same prescaler and a Reset of this prescaler will affect the mentioned timers.

16.9.5 TC0 Counter Value Register

Name: TCNT0
Offset: 0x46
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x26

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.

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.

16.9.6 TC0 Output Compare Register A

Name: OCR0A
Offset: 0x47
Reset: 0x00
Property: When addressing as I/O register: address offset is 0x27

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.

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.

16.9.7 TC0 Output Compare Register B

Name: OCR0B
Offset: 0x48
Reset: 0x00
Property: When addressing as I/O register: address offset is 0x28

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.

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.

16.9.8 TC0 Interrupt Flag Register

Name: TIFR0
Offset: 0x35
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x15

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.

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 on the WGM0[2:0] bit setting. Refer to bit description of WGM0 in *TCCR0A*.

Related Links
TCCR0A

17. 16-bit Timer/Counter1 (TC1) with PWM

17.1 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

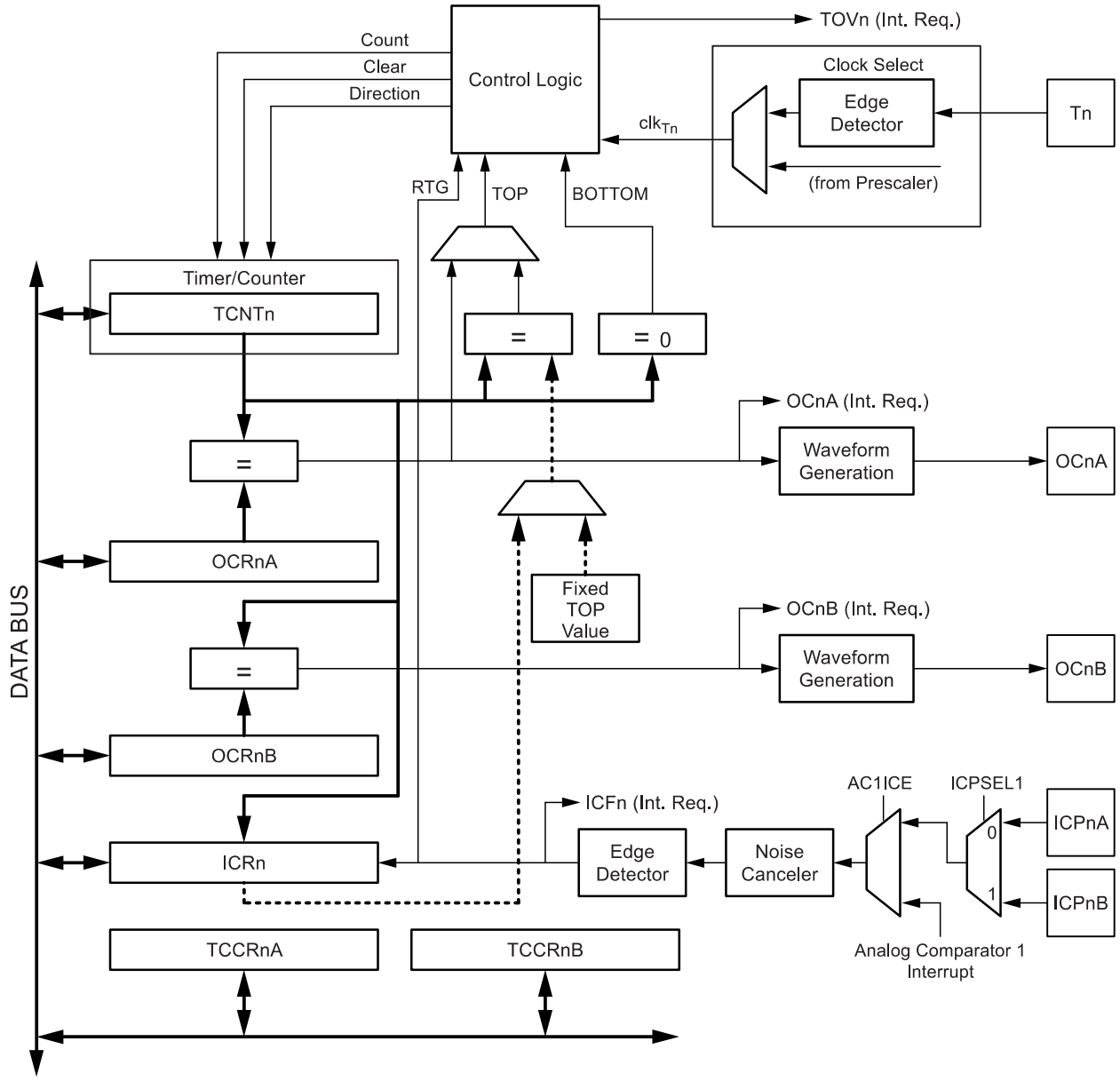
I/O-Ports

17.2 Features

- True 16-bit Design (i.e., allows 16-bit PWM)
- Two Independent Output Compare Units
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Retriggering Function by External Signal (ICP1A or ICP1B)
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse-Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Four Independent Interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)

17.3 Block Diagram

Figure 17-1. 16-bit Timer/Counter Block Diagram



See the related links for actual pin placement.

17.4 Definitions

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

- n=1 represents the timer/counter number
- x=A, B represents the output compare unit A or B

However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing timer/counter1 counter value.

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 one of the fixed values 0x00FF, 0x01FF or 0x03FF, or the value stored in the OCR1A or ICR1 register. The assignment is dependent on the mode of operation.

17.5 Registers

The Timer/Counter (TCNT1), Output Compare registers (OCR1A/B), and Input Capture Register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in section Accessing 16-bit Timer/Counter Registers.

The Timer/Counter Control Registers (TCCR1A/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 (TIFR1). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK1). TIFR1 and TIMSK1 are not shown in the block diagram.

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

The double buffered Output Compare Registers (OCR1A/B) are compared with the timer/counter value at all time. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare pin (OC1A/B). See Output Compare Units. The compare match event will also set the Compare Match Flag (OCF1A/B), which can be used to generate an output compare interrupt request.

The Input Capture register can capture the timer/counter value at a given external (edge triggered) event on either the Input Capture pin (ICP1) or on the analog comparator pins. The input capture unit includes a digital filtering unit (Noise canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum timer/counter value, can in some modes of operation be defined by either the OCR1A register, the ICR1 register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A register cannot 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 runtime. If a fixed TOP value is required, the ICR1 register can be used as an alternative, freeing the OCR1A to be used as PWM output.

17.6 Accessing 16-bit Timer/Counter Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be 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 high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

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

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 OCR1A/B and ICR1 registers. Note that when using C, the compiler handles the 16-bit access.

Assembly Code Example(1)

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

The assembly code example returns the TCNT**1** value in the r17:r16 register pair.

C Code Example(1)

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

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 is corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to perform an atomic read of the TCNT1 register contents. The OCR1A/B or ICR1 registers can be read using the same principle.

Assembly Code Example(1)

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

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

C Code Example(1)

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

Note:

1. 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 TCNT1 register contents. Writing any of the OCR1A/B or ICR1 registers can be done using the same principle.

Assembly Code Example(1)

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

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

C Code Example(1)

```
void TIM16_WriteTCNT1( unsigned int i )
{
```

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

Note:

1. 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`.

17.6.1 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.

17.7 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 (TCCR1B.CS[2:0]).

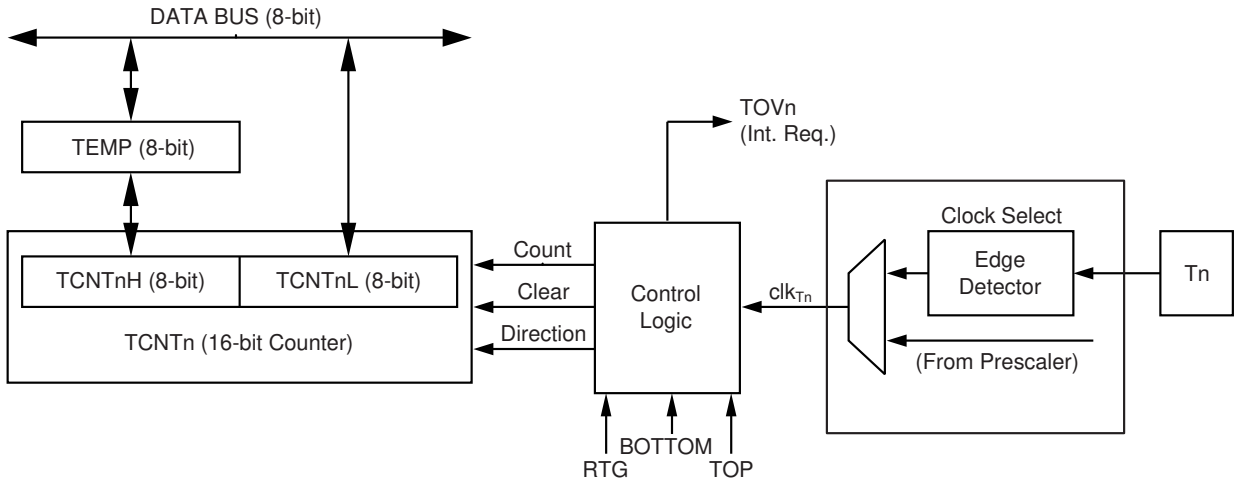
Related Links

Timer/Counter 0, 1 Prescalers

17.8 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 17-2. Counter Unit Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for timer/counter 1), and the “x” indicates output compare unit (A/B).

Table 17-2. Signal Description (Internal Signals)

Signal Name	Description
Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_{T1}	Timer/counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).
RTG	An external event (ICP1A or ICP1B) requests a TOP like action.

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT1H) containing the upper eight bits of the counter, and Counter Low (TCNT1L) containing the lower eight bits. The TCNT1H register can only be accessed indirectly by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus.
Note: That there are special cases when writing to the TCNT1 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_{T1}). The clock clk_{T1} can be generated from an external or internal clock source, as selected by the clock select bits in the Timer/Counter1 Control Register B (TCCR1B.CS[2:0]). When no clock source is selected (CS[2:0]=0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} is present or not. A CPU write overrides (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 (TCCR1B.WGM1[3:2] and TCCR1A.WGM1[1:0]). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OCR1A/B. For more details about advanced counting sequences and waveform generation, see Modes of Operation.

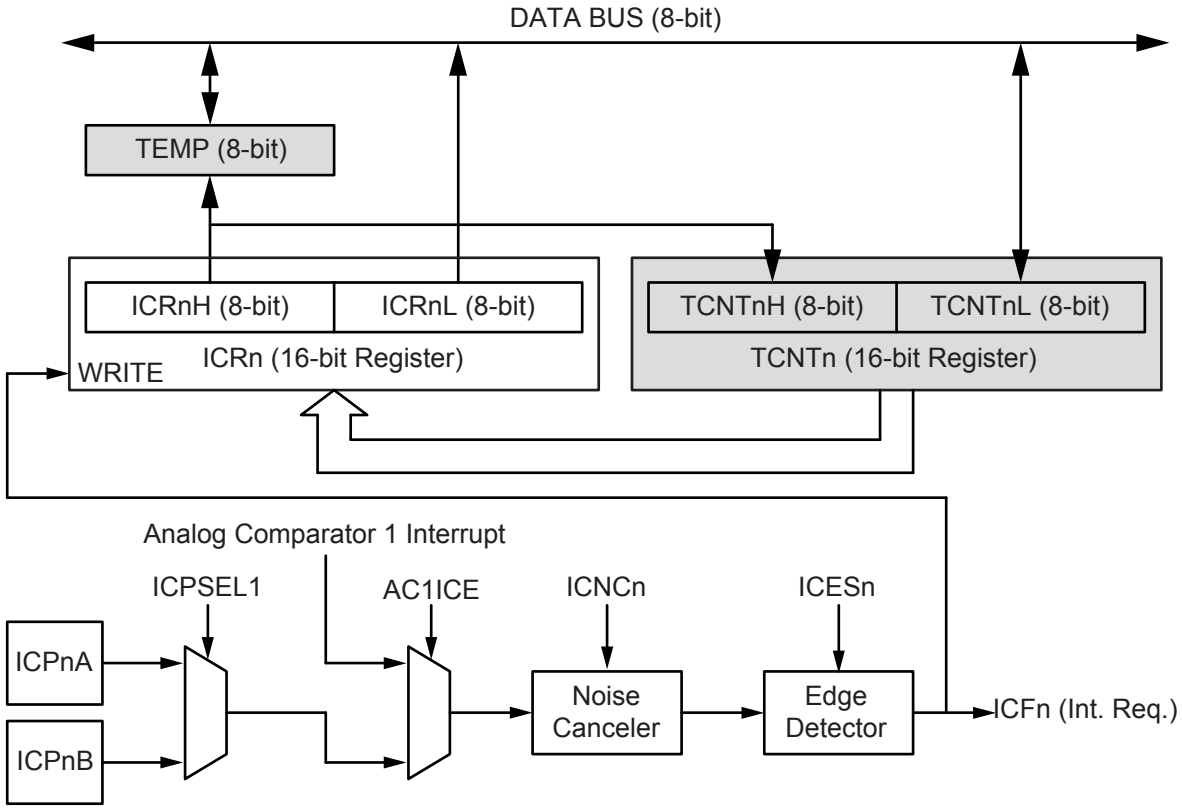
The timer/counter overflow flag in the TC1 Interrupt Flag Register (TIFR1.TOV1) is set according to the mode of operation selected by the WGM1[3:0] bits. TOV1 can be used for generating a CPU interrupt.

17.9 Input Capture Unit

The timer/counter1 incorporates an input capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle and other features of the signal applied. Alternatively, the time-stamps can be used for creating a log of the events.

The input capture unit is illustrated by the block diagram 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 17-3. Input Capture Unit Block Diagram for TC1



Note: The “n” in the register and bit names indicates the device number (n = 1 for timer/counter 1), and the “x” indicates output compare unit (A/B).

When a change of the logic level (an event) occurs on the input capture pin (ICP1), 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 (TCNT1) is written to the Input Capture Register (ICR1). The Input Capture Flag (ICF1) is set at the same system clock cycle as the TCNT1 value is copied into the ICR1 . If enabled (TIMSK1.ICIE1=1), the ICF generates an input capture interrupt. The ICF1 is automatically cleared when the interrupt is executed. Alternatively, the ICF can be cleared by software by writing '1' to its I/O bit location.

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

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

17.9.1 Input Capture Trigger Source

The main trigger source for the input capture unit is the Input Capture pin (ICP1). Timer/Counter1 can alternatively use the analog comparator output as trigger source for the input capture unit. The analog

comparator is selected as a trigger source by setting the Analog Comparator 1 Interrupt Capture Enable bit (AC1ICE) in the Analog Comparator 1 Control Register (AC1CON). Be aware that changing trigger source can trigger a capture. The input capture flag must, therefore, be cleared after the change.

Both the Input Capture Pin (ICP1) and the Analog Comparator Output (ACO) inputs are sampled using the same technique as for the T1 pin. The edge detector is 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 ICR1 to define TOP.

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

17.9.2 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 (TCCR1B.ICNC1). 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 ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

17.9.3 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 ICR1 before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

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

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

Measurement of an external signal’s duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 has been read. After a change of the edge, the ICF1 must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 is not required (if an interrupt handler is used).

17.9.3.1 Using the Input Capture Unit as TCNT1 Retrigger Input

TCNT1 counts from BOTTOM to TOP. The TOP value can be a fixed value, ICR1, or OCR1A. When enabled, the retrigger input forces the TOP value to be reached, indicating that ICF1 output is or'ed with the TOP signal.

17.10 Output Compare Units

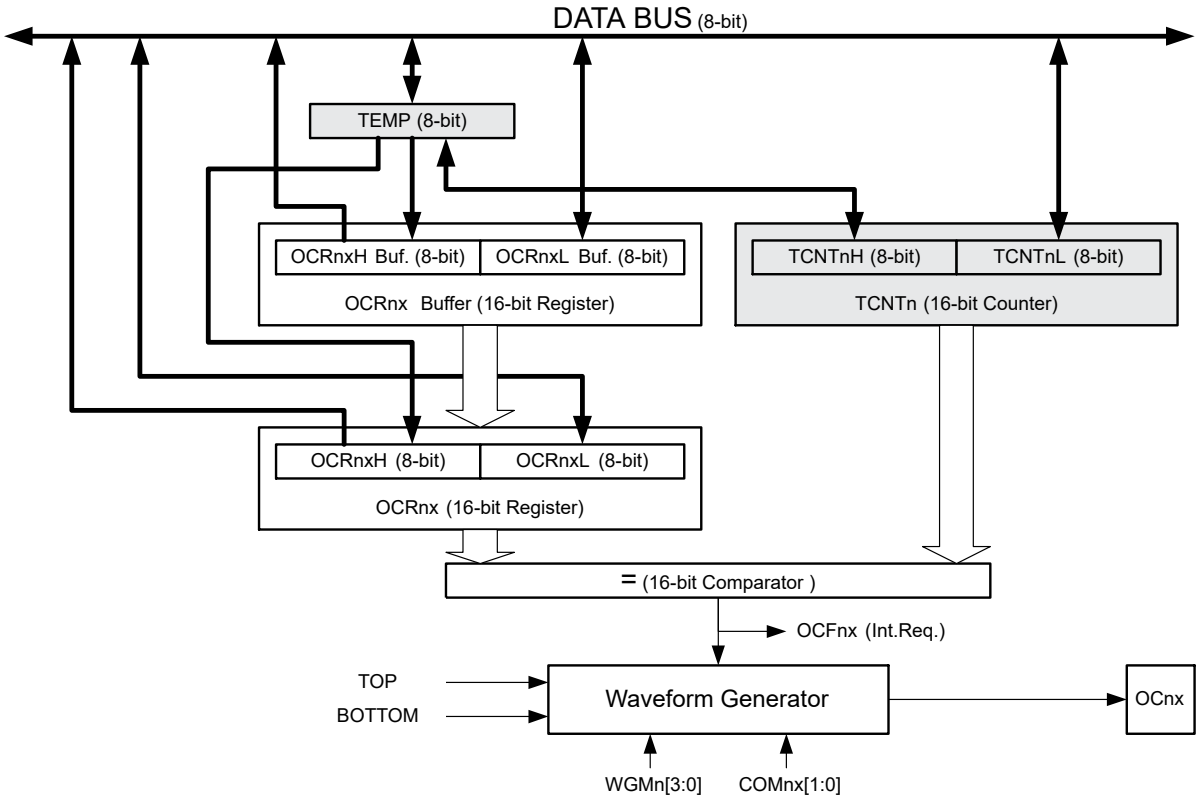
The 16-bit comparator continuously compares TCNT1 with the Output Compare Register (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the Output Compare Flag (TIFR1.OCF1x) at the next timer clock cycle. If enabled (TIMSK1.OCIE1x = 1), the output compare flag generates an output compare interrupt. OCF1x is automatically cleared when the interrupt is executed.

Alternatively, OCF1x 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 (WGM1[3:0]) bits and Compare Output mode (COM1x[1:0]) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some 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 17-4. Output Compare Unit, Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).

The OCR1x is double buffered when using any of the twelve Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x 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 OCR1x Buffer register. When double buffering is disabled, the CPU will access the OCR1x directly.

The content of the OCR1x (Buffer or Compare) register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1). Therefore OCR1x 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 OCR1x must be done via the TEMP register since the compare of all 16 bits is done continuously. The high byte (OCR1xH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP register will be updated by the value written. Then when the low byte (OCR1xL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x in the same system clock cycle.

Related Links

Accessing 16-bit Timer/Counter Registers

17.10.1 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 (TCCR1C.FOC1x) bit. Forcing compare match will not set the OCF1x Flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the TCCR1C.COM1x[1:0] bits settings define whether the OC1x pin is set, cleared or toggled).

17.10.2 Compare Match Blocking by TCNT1 Write

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

17.10.3 Using the Output Compare Unit

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

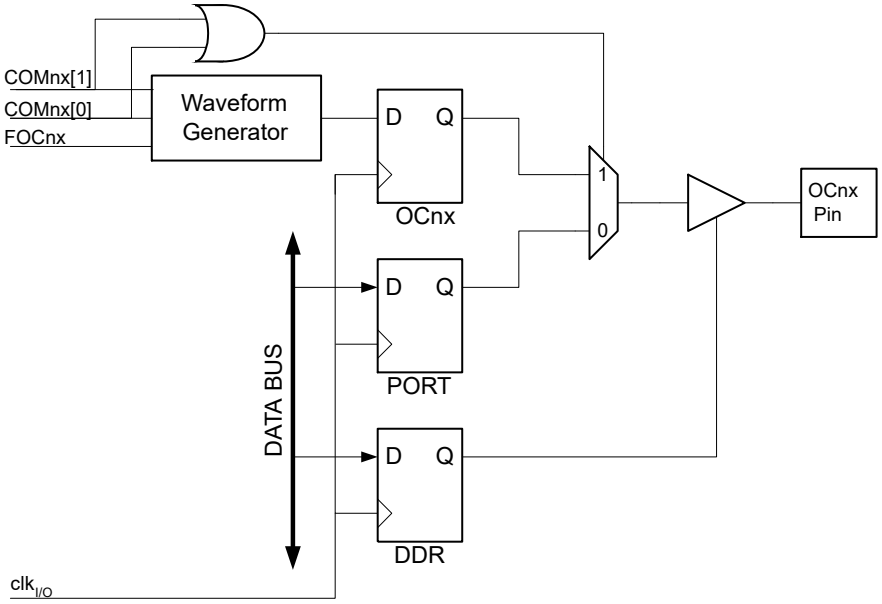
The setup of the OC1x should be performed before setting the Data Direction register for the port pin to output. The easiest way of setting the OC1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x register keeps its value even when changing between Waveform Generation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

Be aware that the TCCR1A.COM1x[1:0] bits are not double buffered together with the compare value. Changing the TCCR1A.COM1x[1:0] will take effect immediately.

17.11 Compare Match Output Unit

The Compare Output mode (TCCR1A.COM1x[1:0]) bits have two functions. The waveform generator uses the TCCR1A.COM1x[1:0] bits for defining the Output Compare (OC1x) state at the next compare match. Secondly the TCCR1A.COM1x[1:0] bits control the OC1x pin output source. The figure below shows a simplified schematic of the logic affected by the TCCR1A.COM1x[1:0] bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the TCCR1A.COM1x[1:0] bits are shown. When referring to the OC1x state, the reference is for the internal OC1x register, not the OC1x pin. If a System Reset occurs, the OC1x register is reset to “0”.

Figure 17-5. Compare Match Output Unit, Schematic



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OC1x) from the waveform generator if either of the TCCR1A.COM1x[1:0] bits are set. However, the OC1x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The DDR bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the waveform generation mode, but there are some exceptions.

The design of the output compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some TCCR1A.COM1x[1:0] bit settings are reserved for certain modes of operation.

The TCCR1A.COM1x[1:0] bits have no effect on the input capture unit.

17.11.1 Compare Output Mode and Waveform Generation

The waveform generator uses the TCCR1A.COM1x[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the TCCR1A.COM1x[1:0] = 0 tells the waveform generator that no action on the OC1x register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the TCCR1A.COM1x[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 TCCR1C.FOC1x strobe bits.

17.12 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 (WGM1[3:0]) and Compare Output mode (TCCR1A.COM1x[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The TCCR1A.COM1x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the TCCR1A.COM1x[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

17.12.1 Normal Mode

The simplest mode of operation is the Normal mode (TCCR1A.WGM1[3:0]=0). In this mode, the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX=0xFFFF) and then restarts from BOTTOM=0x0000. In normal operation, the Timer/Counter Overflow Flag (TIFR1.TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. In this case, the TOV1 flag behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written any time.

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

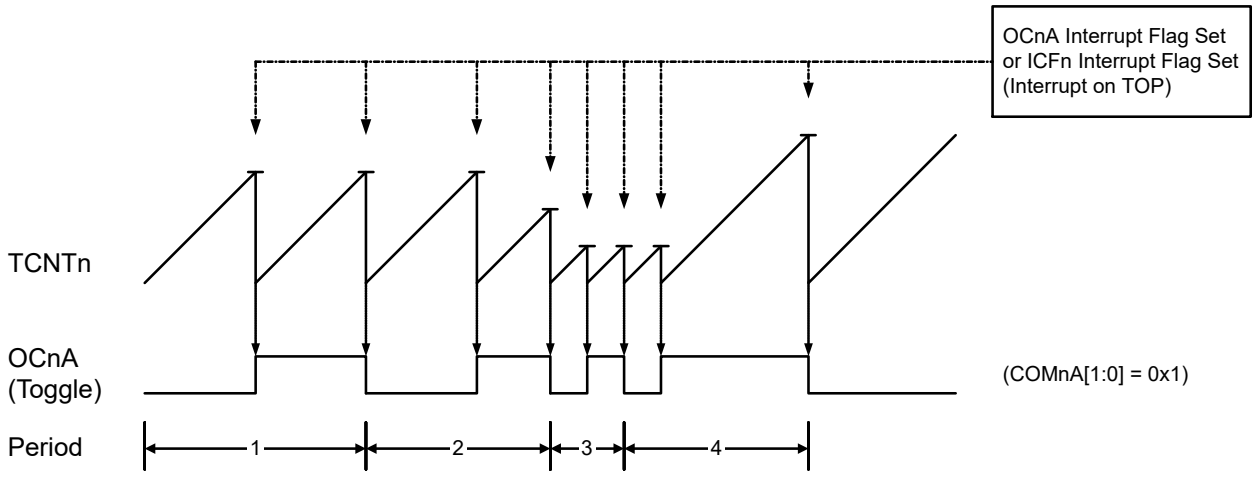
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended since this will occupy too much of the CPU time.

17.12.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare (CTC) modes (mode 4 or 12, WGM1[3:0]=0x4 or 0xC), the OCR1A or ICR1 registers are used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNT1) matches either the OCR1A (if WGM1[3:0]=0x4) or the ICR1 (WGM1[3:0]=0xC). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown below. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then TCNT1 is cleared.

Figure 17-6. CTC Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).

An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 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 OCR1A is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then count to its maximum value (0xFF for an 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 OCR1A for defining TOP (WGM1[3:0]=0xF), since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM1A[1:0]=0x1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A=1). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to ZERO (0x0000). The waveform frequency is defined by the following equation:

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

Note:

- The “n” indicates the device number (n = 1 for Timer/Counter 1), 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 TOV1 flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

17.12.3 Fast PWM Mode

The Fast Pulse Width Modulation or Fast PWM modes (modes 5, 6, 7, 14, and 15, WGM1[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 (OC1x) is cleared on the compare match between TCNT1 and OCR1x 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.

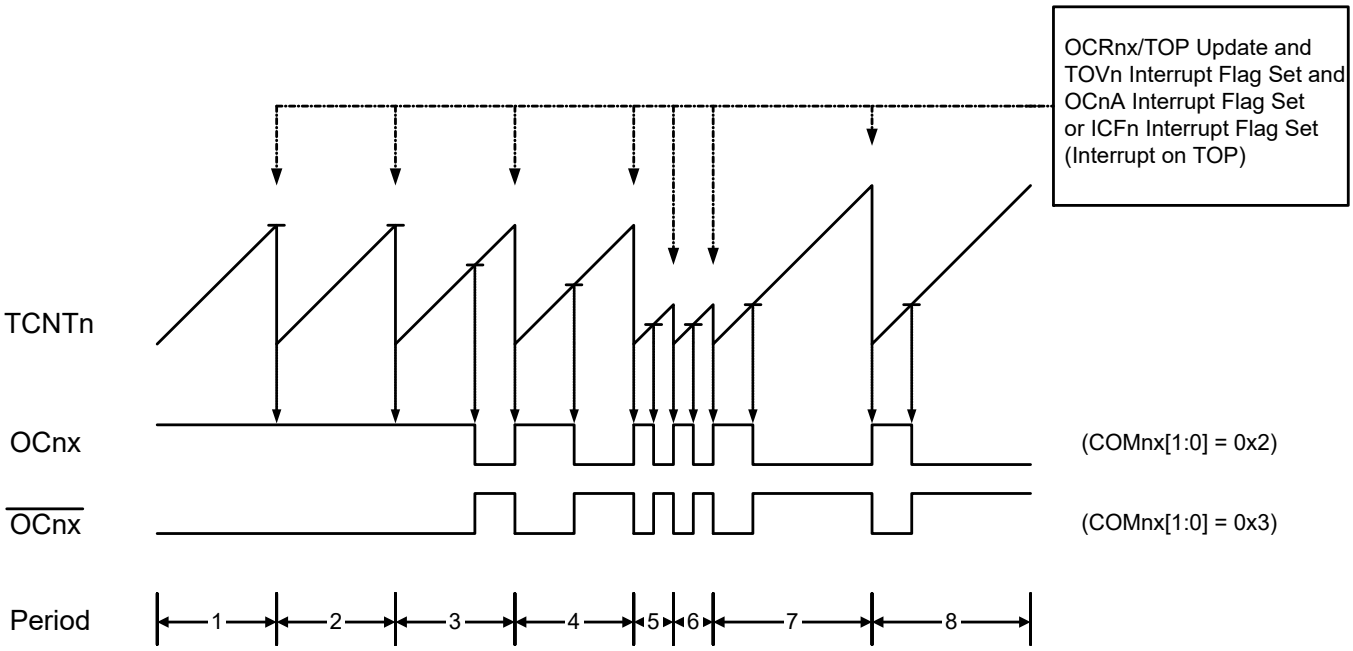
The PWM resolution for Fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A register set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A registers set to MAX). The PWM resolution in bits can be calculated 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 (WGM1[3:0] = 0x5, 0x6, or 0x7), the value in ICR1 (WGM1[3:0]=0xE), or the value in OCR1A (WGM1[3:0]=0xF). The counter is then cleared at the following timer clock cycle.

The timing diagram for the Fast PWM mode using OCR1A or ICR1 to define TOP is shown below. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT1 slopes mark compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 17-7. Fast PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).

The Timer/Counter Overflow flag (TOV1) is set each time the counter reaches TOP. In addition, when either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag is set at the same timer clock cycle TOV1 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 TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. 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 OCR1A Register, however, is double buffered. This feature allows the OCR1A I/O location to be written any time. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer register. The OCR1A Compare register will then be updated with the value in the Buffer register at the next timer clock cycle the TCNT1 matches TOP. The update is performed at the same timer clock cycle as the TCNT1 is cleared and the TOV1 flag is set.

Using the ICR1 register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency

is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In Fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Writing the COM1x[1:0] bits to 0x2 will produce an inverted PWM and a non-inverted PWM output can be generated by writing the COM1x[1:0] to 0x3. The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

Note:

- The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), 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 OCR1x registers represent special cases when generating a PWM waveform output in the Fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output which is controlled by COM1x[1:0]).

A frequency waveform output with 50% duty cycle can be achieved in Fast PWM mode by selecting OC1A to toggle its logical level on each compare match (COM1A[1:0]=0x1). This applies only if OCR1A is used to define the TOP value (WGM1[3:0]=0xF). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the Fast PWM mode.

17.12.4 Phase Correct PWM Mode

The Phase Correct Pulse Width Modulation or Phase Correct PWM modes (WGM1[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 (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting, and set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

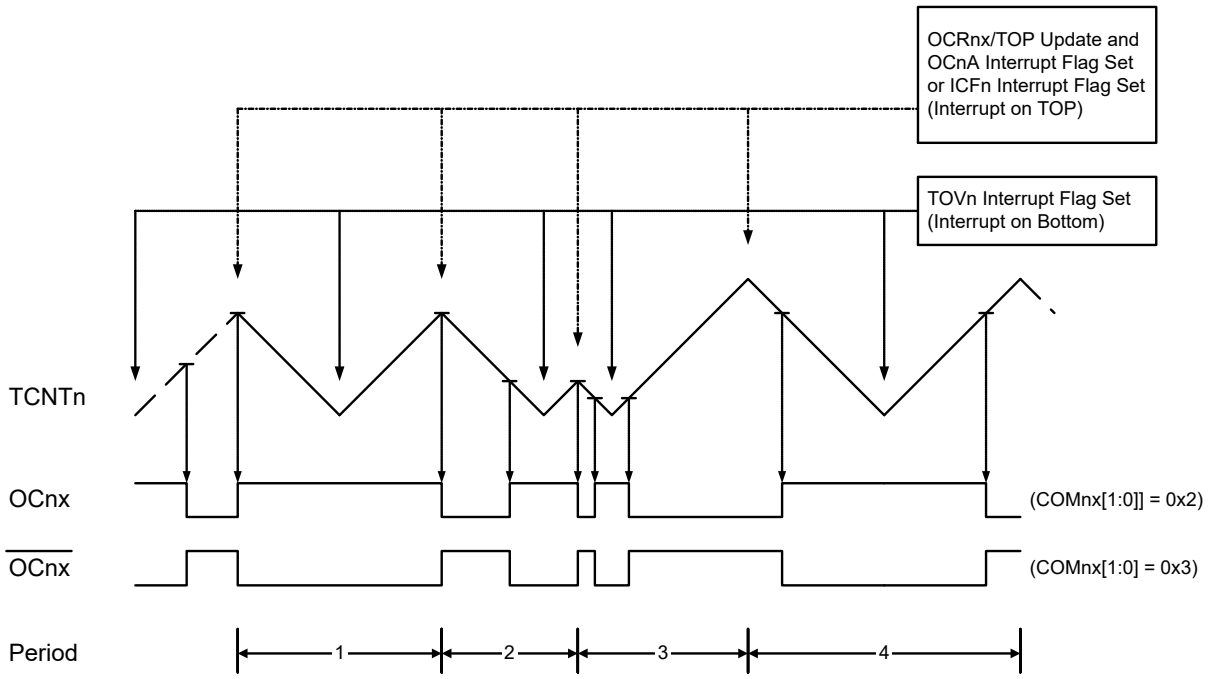
The PWM resolution for the Phase Correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated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

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

In Phase Correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM1[3:0]= 0x1, 0x2, or 0x3), the value in ICR1

(WGM1[3:0]=0xA), or the value in OCR1A (WGM1[3:0]=0xB). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the Phase Correct PWM mode is shown below, using OCR1A or ICR1 to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT1 slopes mark compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 17-8. Phase Correct PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).

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

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x registers 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 OCR1x. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

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

In Phase Correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Writing COM1x[1:0] bits to 0x2 will produce a non-inverted PWM. An inverted PWM output can be generated by writing the COM1x[1:0] to 0x3. The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using Phase Correct PWM can be calculated by the following equation:

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

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

The extreme values for the OCR1x represent special cases when generating a PWM waveform output in the Phase Correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM1[3:0]=0xB) and COM1A[1:0]=0x1, the OC1A output will toggle with a 50% duty cycle.

17.12.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation, or phase and frequency correct PWM mode (WGM1[3:0] = 0x8 or 0x9) provides a high-resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode are, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while up-counting, and set on the compare match while down-counting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

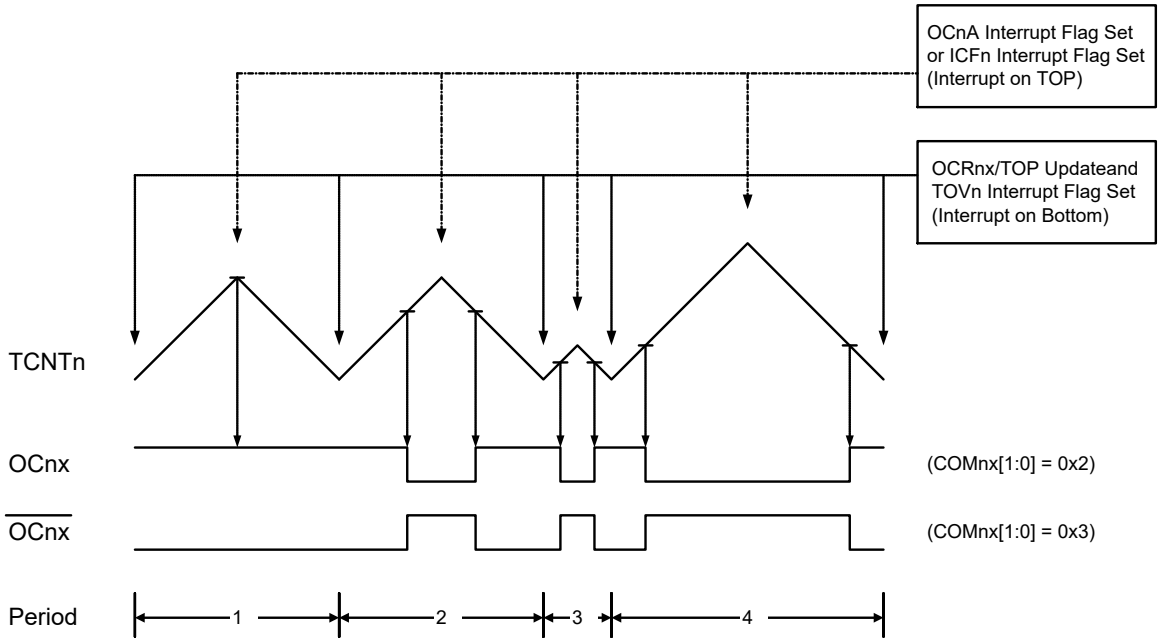
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x is updated by the OCR1x Buffer register, (see Figure 17-8 and the Timing Diagram below).

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

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

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM1[3:0]=0x8), or the value in OCR1A (WGM1[3:0]=0x9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown below. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 17-9. Phase and Frequency Correct PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).

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

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

As shown in the timing diagram above, the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is, therefore, frequency correct.

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

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x[1:0] bits to 0x2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x[1:0] to 0x3 (see the description of TCCRA.COM1x). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

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

Note:

- The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates output compare unit (A/B).
- *N* represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM1[3:0]=0x9) and COM1A[1:0]=0x1, the OC1A output will toggle with a 50% duty cycle.

17.13 Timer/Counter 0, 1 Prescalers

The 8-bit Timer/Counter0 (TC0) and the 16-bit Timer/Counter1 (TC1) share the same prescaler module, but the timer/counters can have different prescaler settings. The following description applies to TC0, TC1.

Related Links

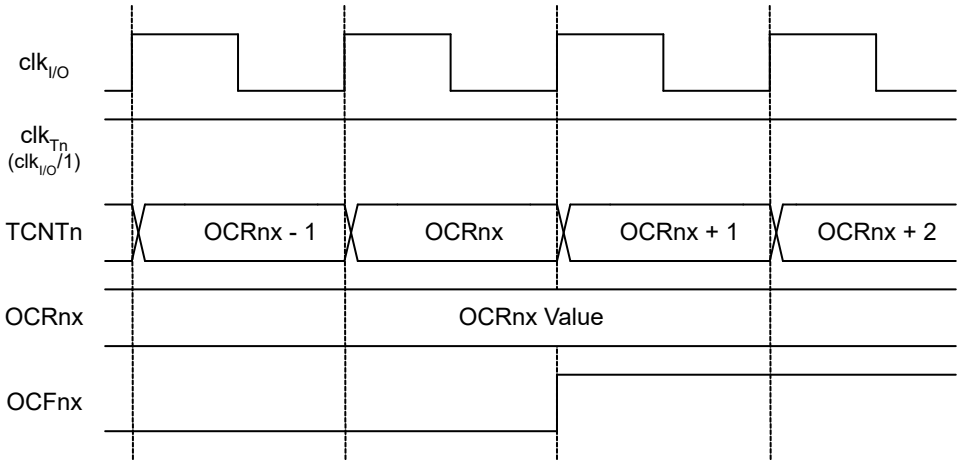
8-bit Timer/Counter0 (TC0) with PWM

16-bit Timer/Counter1 (TC1) with PWM

17.14 Timer/Counter Timing Diagrams

The timer/counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR1x is updated with the OCR1x buffer value (only for modes utilizing double buffering). The first figure shows a timing diagram for the setting of OCF1x.

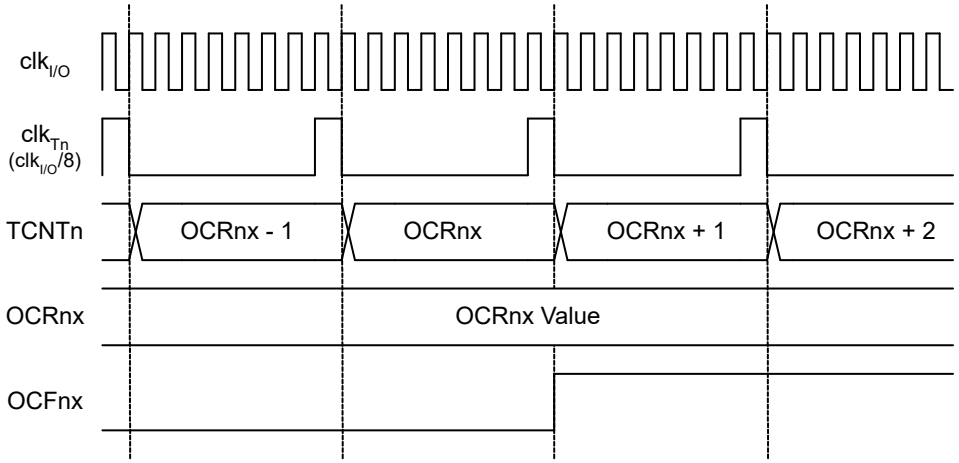
Figure 17-10. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling



Note: The “n” in the register and bit names indicates the device number (n = 1 for timer/counter 1), and the “x” indicates output compare unit (A/B).

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

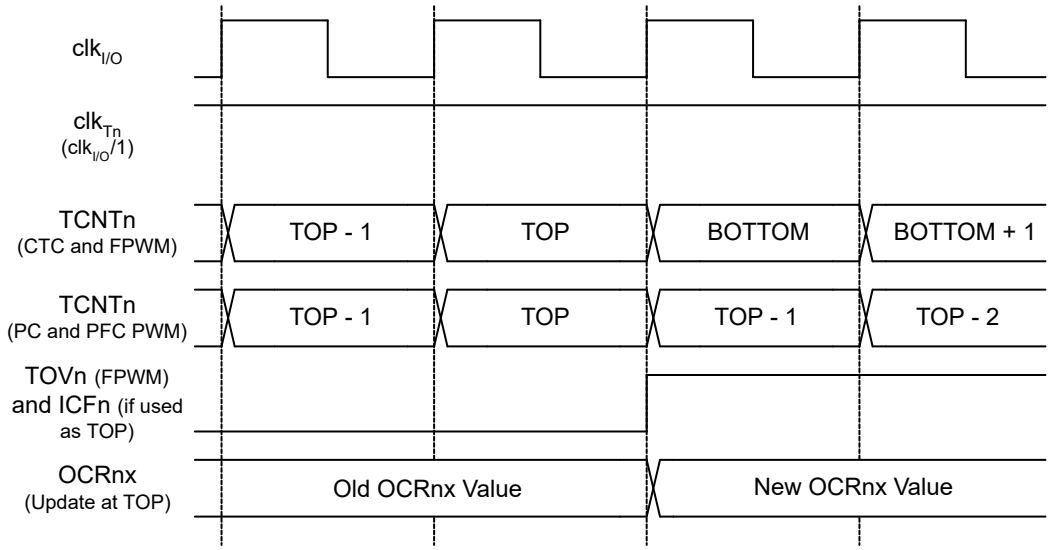
Figure 17-11. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}/8$)



Note: The “n” in the register and bit names indicates the device number (n = 1 for timer/counter 1), and the “x” indicates output compare unit (A/B).

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

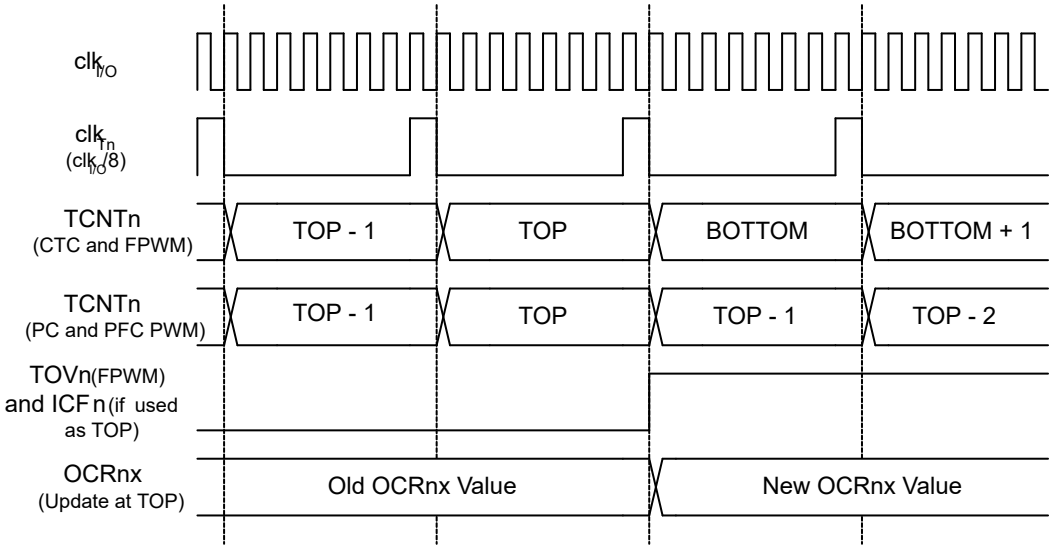
Figure 17-12. Timer/Counter Timing Diagram, no Prescaling.



Note: The “n” in the register and bit names indicates the device number (n = 1 for timer/counter 1), and the “x” indicates output compare unit (A/B).

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

Figure 17-13. 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 for timer/counter 1), and the “x” indicates output compare unit (A/B).

17.15 Register Description

17.15.1 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 – COM1 Compare Output Mode for Channel
The COM1A[1:0] and COM1B[1:0] control the behavior of the output compare pins OC1A and OC1B, respectively. If one or both of the COM1A[1:0] bits are written to one, the OC1A output overrides the normal port functionality of the connected I/O pin. If one or both of the COM1B[1:0] bits are written to one, the OC1B output overrides the normal port functionality of the connected I/O pin. 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 on the setting of WGM1[3:0]. The table below shows COM1x[1:0] functionality when WGM1[3:0] are set to a Normal or to a CTC mode (non-PWM).

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

COM1A[1]/ COM1B[1]	COM1A[0]/ COM1B[0]	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 COM1x[1:0] functionality when WGM1[3:0] are set to the fast PWM mode.

Table 17-4. Compare Output Mode, Fast PWM

COM1A[1]/ COM1B[1]	COM1A[0]/ COM1B[0]	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.

COM1A[1]/ COM1B[1]	COM1A[0]/ COM1B[0]	Description
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at TOP (Non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at TOP (Inverting mode)

Note:

1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A[1]/COM1B[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 COM1x[1:0] functionality when WGM1[3:0] are set to the phase correct or the phase and frequency correct, PWM mode.

Table 17-5. Compare Output Mode, Phase Correct, and Phase and Frequency Correct PWM

COM1A[1]/ COM1B[1]	COM1A[0]/ COM1B[0]	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 8, 9, 10 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 COM1A[1]/COM1B[1] are 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 17-6. Waveform Generation Mode Bit Description

Mode	WGM1[3]	WGM1[2] (CTC1)(1)	WGM1[1] (PWM1[1])(1)	WGM1[0] (PWM1[0])(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

Mode	WGM1[3]	WGM1[2] (CTC1)[1]	WGM1[1] (PWM1[1])[1]	WGM1[0] (PWM1[0])[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
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	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	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	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Note:

1. The CTC1 and PWM1[1:0] bit definition names are obsolete. Use the WGM1[2:0] definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

17.15.2 TC1 Control Register B

Name: TCCR1B
Offset: 0x81
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	RTGEN	WGM13	WGM12	CS1[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 – 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 a 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.

Bit 5 – RTGEN Retrigger Input Enable
Set this bit to enable the ICP1A as a Timer/Counter retrigger input. (This bit is reserved for future use. To ensure compatibility with future devices, this bit must be written to zero when TCCR1B is written.)

Bits 3, 4 – WGM1 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 17-10 and Figure 17-11.

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

Table 17-7. Clock Select Bit Description

CS1[2]	CS1[1]	CS1[0]	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)

CS1[2]	CS1[1]	CS1[0]	Description
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.

17.15.3 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 – FOC1 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.

17.15.4 TC1 Counter Value Low and High byte

Name: TCNT1L and TCNT1H
Offset: 0x84
Reset: 0x00
Property: -

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 Timer/Counter Registers*.

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.

Related Links
Accessing 16-bit Timer/Counter Registers

17.15.5 Input Capture Register 1 Low and High byte

Name: ICR1L and ICR1H
Offset: 0x86
Reset: 0x00
Property: -

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 Timer/Counter Registers*.

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.

Related Links
Accessing 16-bit Timer/Counter Registers

17.15.6 Output Compare Register 1 A Low and High byte

Name: OCR1AL and OCR1AH
Offset: 0x88
Reset: 0x00
Property: -

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 Timer/Counter Registers*.

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.

Related Links
Accessing 16-bit Timer/Counter Registers

17.15.7 Output Compare Register 1 B Low and High byte

Name: OCR1BL and OCR1BH
Offset: 0x8A
Reset: 0x00
Property: -

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 Timer/Counter Registers*.

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.

Related Links
Accessing 16-bit Timer/Counter Registers

17.15.8 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 TOV1 flag, located in TIFR1, is set.

17.15.9 TC1 Interrupt Flag Register

Name: TIFR1
Offset: 0x36
Reset: 0x00
Property: When addressing as I/O register: address offset is 0x16

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.

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 on 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. Timer/Counter 0, 1 Prescalers

The 8-bit Timer/Counter0 (TC0) and the 16-bit Timer/Counter1 (TC1) share the same prescaler module, but the timer/counters can have different prescaler settings. The following description applies to TC0, TC1.

Related Links

8-bit Timer/Counter0 (TC0) with PWM

16-bit Timer/Counter1 (TC1) with PWM

18.1 Internal Clock Source

The timer/counter can be clocked directly by the system clock (by setting the CSn[2:0]=0x01). 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 $f_{CLK\_I/O}/1024$.

18.2 Prescaler Reset

The prescaler is free-running, i.e., it 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 (0x06 > CSn[2:0] > 0x01). 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).

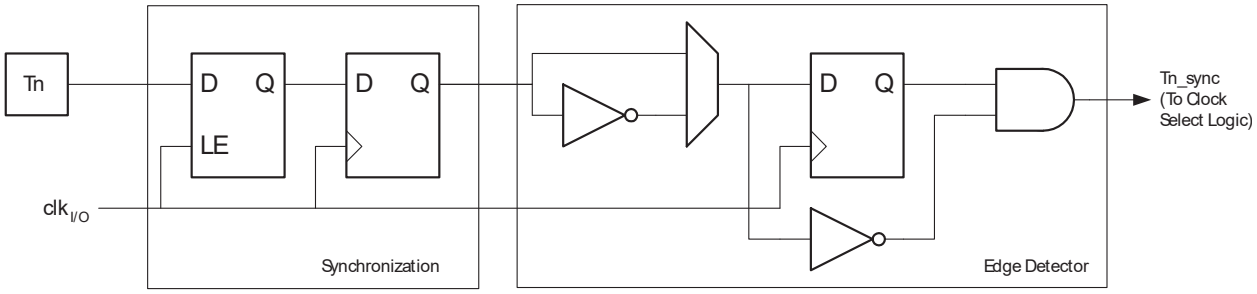
It is possible to use the prescaler Reset for synchronizing the timer/counter to program execution. However, care must be taken if the other timer/counter that shares the same prescaler also uses prescaling. A prescaler Reset will affect the prescaler period for all timer/counters it is connected to.

18.3 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 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 18-1. T1/T0 Pin Sampling



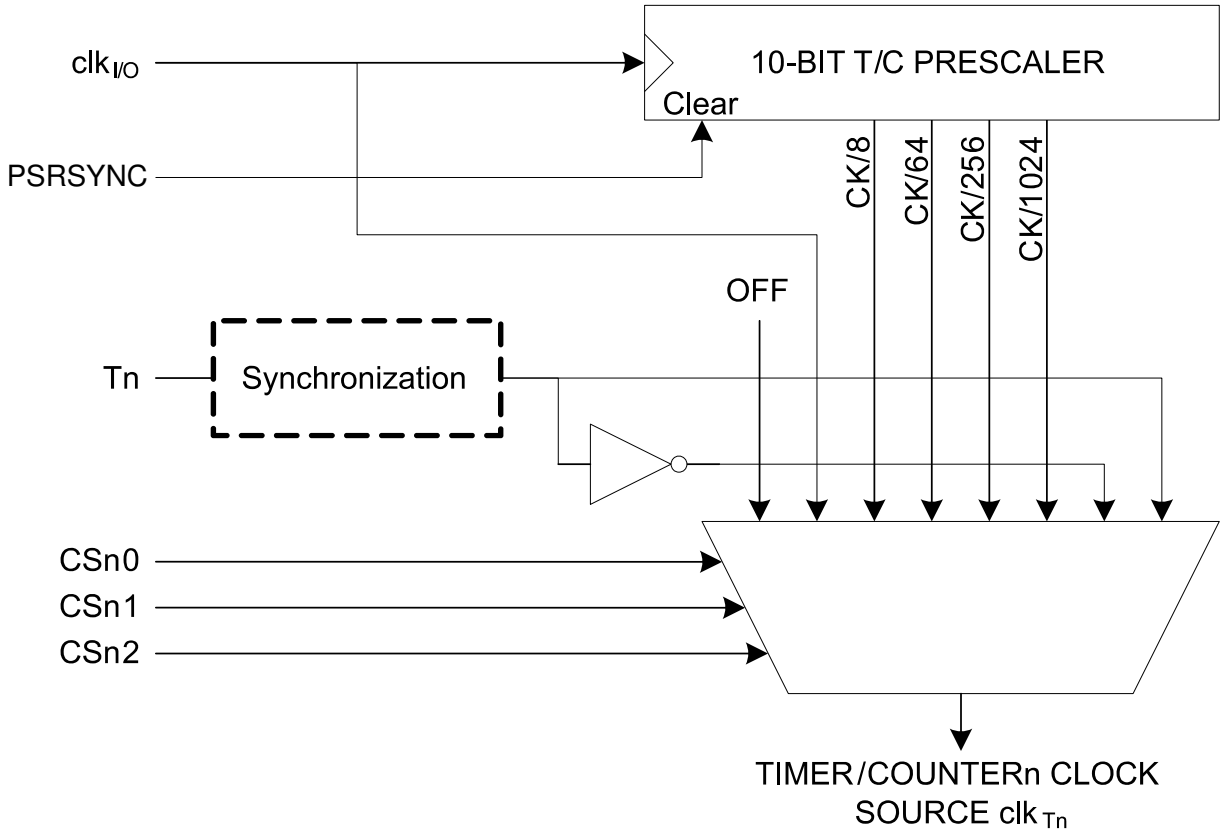
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

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 cannot be prescaled.

Figure 18-2. 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.

18.4 Register Description

18.4.1 General Timer/Counter Control Register

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

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.

Bit	7	6	5	4	3	2	1	0
	TSM	ICPSEL1						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 PSRSYNC bit 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, hardware clears the PSRSYNC bit, and the timer/counters start counting simultaneously.

Bit 6 – ICPSEL1 Timer 1 Input Capture selection
Timer 1 capture function has two possible inputs ICP1A (PD4) and ICP1B (PC3).

Value	Description
0	Select ICP1A as trigger for timer 1 input capture
1	Select ICP1B as trigger for timer 1 input capture

Bit 0 – PSRSYNC Prescaler Reset
When this bit is one, timer/counter 0, 1 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 share the same prescaler and a Reset of this prescaler will affect the mentioned timers.

19. PSC – Power Stage Controller

19.1 Features

- PWM waveform generating function with six complementary programmable outputs (able to control three half-bridges)
- Programmable dead time control
- PWM up to 12-bit resolution
- PWM clock frequency up to 64MHz (via PLL)
- Programmable ADC trigger
- Automatic Overlap protection
- Failsafe emergency inputs - 3 (to force all outputs to high impedance or in inactive state - fuse configurable)
- Center aligned and edge aligned modes synchronization

19.2 Overview

The Power Stage Controller is a high performance waveform controller.

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

- A lower case “n” replaces the PSC module number, in this case 0, 1 or 2. However, when using the register or bit defines in a program, the precise form must be used, that is, POCR0SAH for accessing module 0 POCRnSAH register and so on
- A lower case “x” replaces the PSC part , in this case A or B. However, when using the register or bit defines in a program, the precise form must be used, that is, OCR0SAH for accessing part A OCR0SxH register and so on

The purpose of the Power Stage Controller (PSC) is to control an external power interface. It has six outputs to drive for example a three half-bridge. This feature allows you to generate three phase waveforms for applications such as Asynchronous or BLDC motor drives, lighting systems...

The PSC also has three inputs, the purpose of which is to provide fast emergency stop capability.

The PSC outputs are programmable as “active high” or “active low”. All the timing diagrams in the following examples are given in the “active high” polarity.

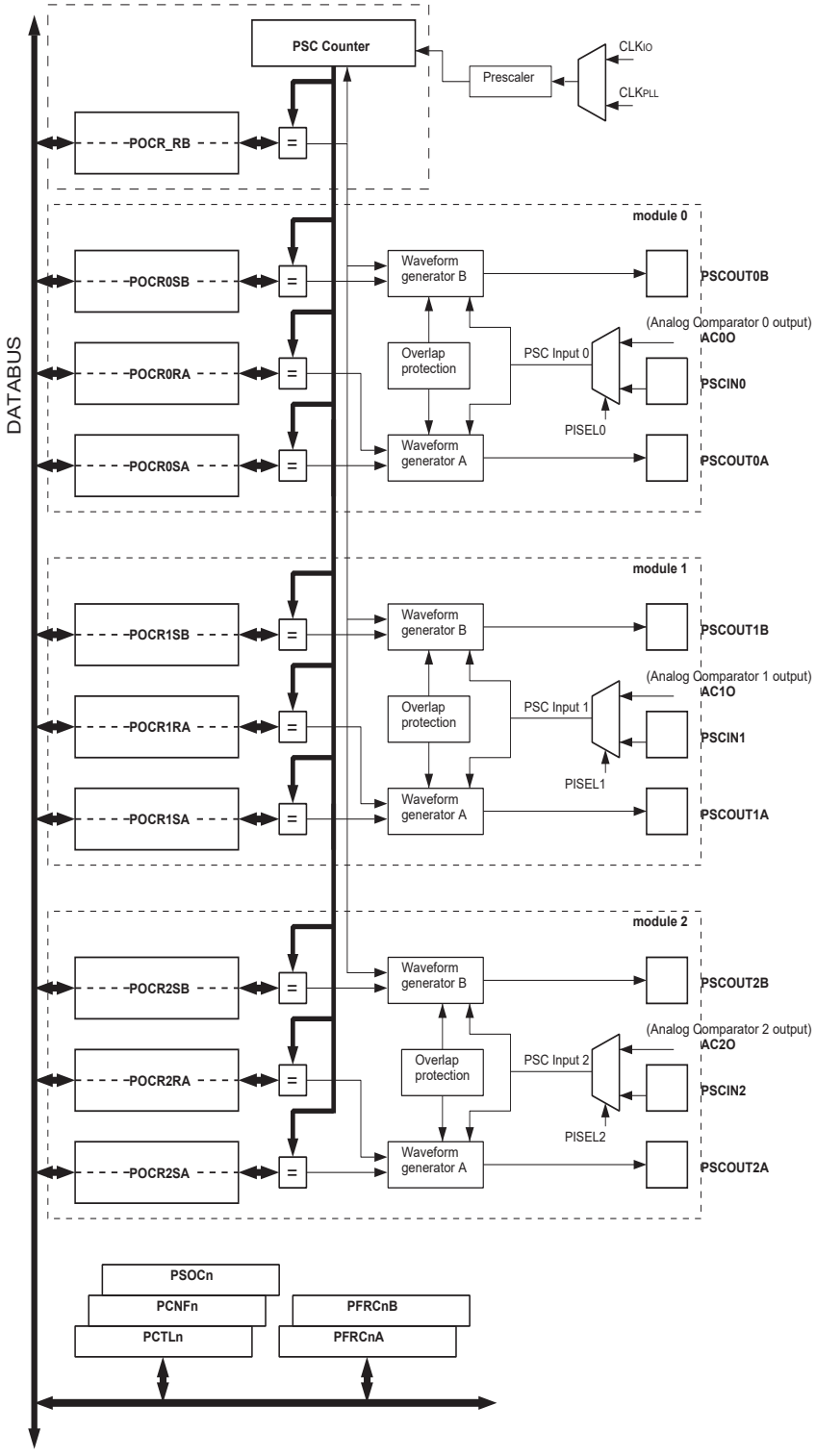
19.3 Accessing 16-bit registers

Some PSC registers are 16-bit registers. These registers can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit registers must be byte accessed using two read or write operations. The PSC has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all PSC 16-bit registers. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

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

19.4 PSC description

Figure 19-1. Power Stage Controller block diagram.



The PSC is based on the use of a free-running 12-bit counter (PSC counter). This counter is able to count up to a top value determined by the contents of POCR_RB register and then according to the selected running mode, count down or reset to zero for another cycle.

As can be seen from the block diagram, the PSC is composed of three modules.

Each of the three PSC modules can be seen as two symmetrical entities. One entity named part A which generates the output PSCOUTnA and the second one named part B which generates the PSCOUTnB output.

Each module has its own PSC Input circuitry which manages the corresponding input.

19.5 Functional description

19.5.1 Generating control waveforms

In general, the drive of a 3-phase motor requires generating six PWM signals. The duty cycle of these signals must be independently controlled to adjust the speed or torque of the motor or to produce the wanted waveform on the three voltage lines (trapezoidal, sinusoidal, and so on).

In case of cross conduction or overtemperature, having inputs which can immediately disable the waveform generator's outputs is desirable.

These considerations are common for many systems which require PWM signals to drive power systems such as lighting, DC/DC converters, and so on.

19.5.2 Waveform cycles

Each of the three modules has two waveform generators which jointly compose the output signal.

The first part of the waveform is relative to part A or PSCOUTnA output. This waveform corresponds to sub-cycle A in the following figure.

The second part of the waveform is relative to part B or PSCOUTnB output. This waveform corresponds to sub-cycle B in the following figure.

The complete waveform is terminated at the end of the sub-cycle B, whereupon any changes to the settings of the waveform generator registers will be implemented, for the next cycle.

The PSC can be configured in one of two modes (1Ramp Mode or Centered Mode). This configuration will affect the operation of all the waveform generators.

Figure 19-2. Cycle presentation in One Ramp mode.

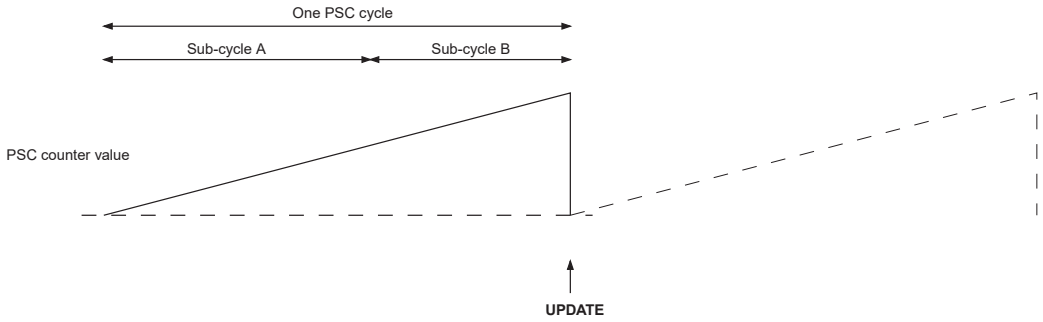
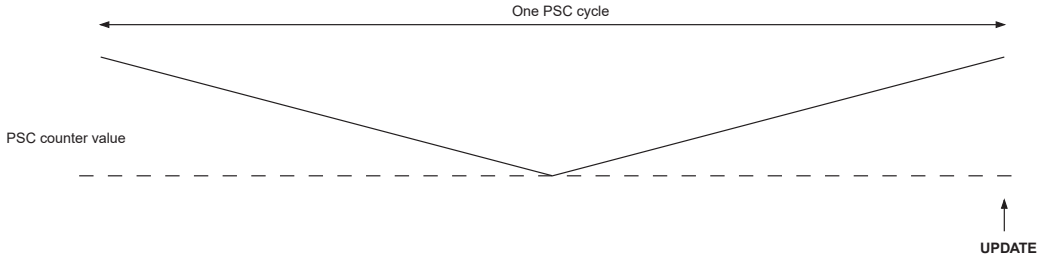


Figure 19-3. Cycle presentation in Centered mode.



The figures above graphically illustrate the values held in the PSC counter. Centered mode is like One Ramp mode which counts down and then up.

Notice that the update of the waveform generator registers is done regardless of ramp mode at the end of the PSC cycle.

19.5.3 Operation mode descriptions

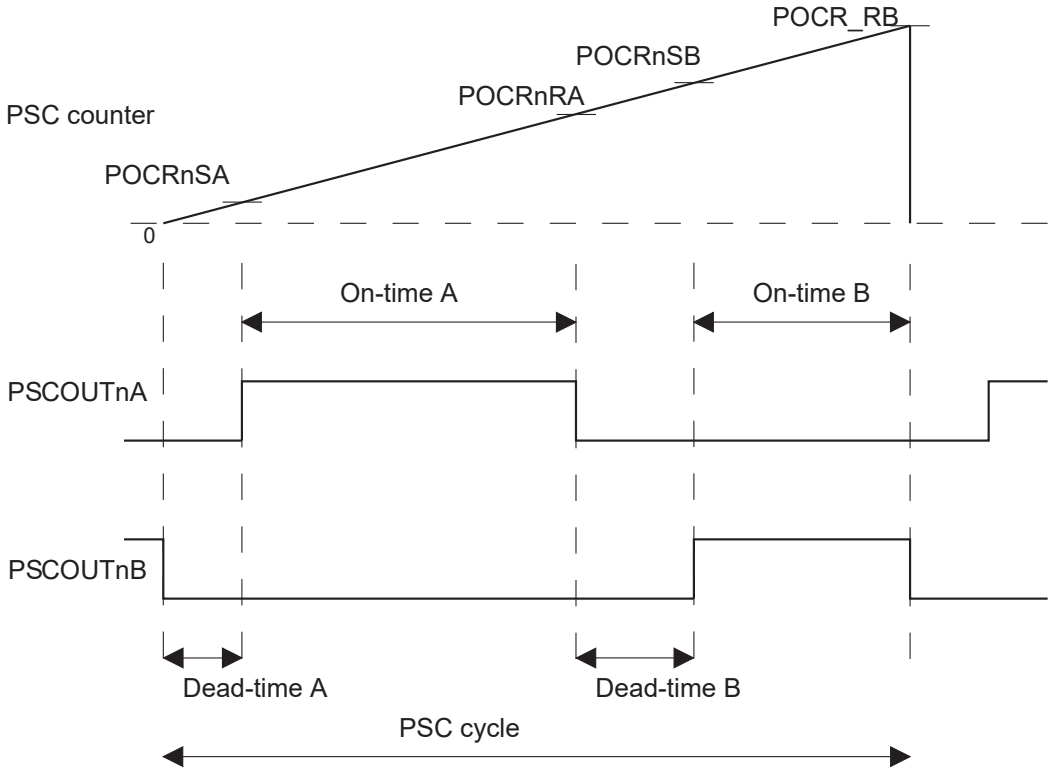
Waveforms and duration of output signals are determined by parameters held in the registers (POCRnSA, POCRnRA, POCRnSB, POCR_RB) and by the running mode. Two modes are possible:

- One Ramp Mode. In this mode, all the three PSCOUTnB outputs are edge-aligned and the three PSCOUTnA can be also edge-aligned when setting the same values in the dedicated registers.
  In this mode, the PWM frequency is twice the Center Aligned mode PWM frequency
- Center Aligned Mode. In this mode, all the six PSC outputs are aligned at the center of the period. Except when using the same duty cycles on the three modules, the edges of the outputs are not aligned. So the PSC outputs do not commute at the same time, thus the system which is driven by these outputs will generate less commutation noise.
  In this mode, the PWM frequency is twice as slow as in One Ramp mode

19.5.3.1 One Ramp Mode (Edge Aligned)

The following figure shows the resultant outputs PSCOUTnA and PSCOUTnB operating in One Ramp mode over a PSC cycle.

Figure 19-4. PSCOUTnA & PSCOUTnB Basic Waveforms in One Ramp Mode



On-Time A = (POCRnRAH/L - POCRnSAH/L) × $1/f_{CLKPSC}$

On-Time B = (POCR_RBH/L - POCRnSBH/L) × $1/f_{CLKPSC}$

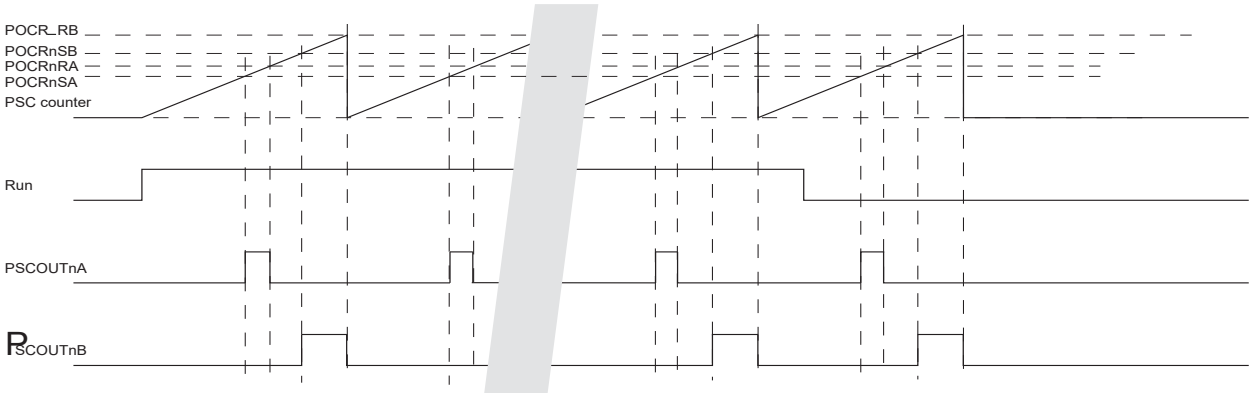
Dead-Time A = (POCRnSAH/L + 1) × $1/f_{CLKPSC}$

Dead-Time B = (POCRnSBH/L - POCRnRAH/L) × $1/f_{CLKPSC}$

Note: Minimal value for Dead-time A = $1/f_{CLKPSC}$.

If the overlap protection is disabled, in One-Ramp mode, PSCOUTnA and PSCOUTnB outputs can be configured to overlap each other, though in normal use this is not desirable.

Figure 19-5. Controlled Start and Stop Mechanism in One Ramp Mode

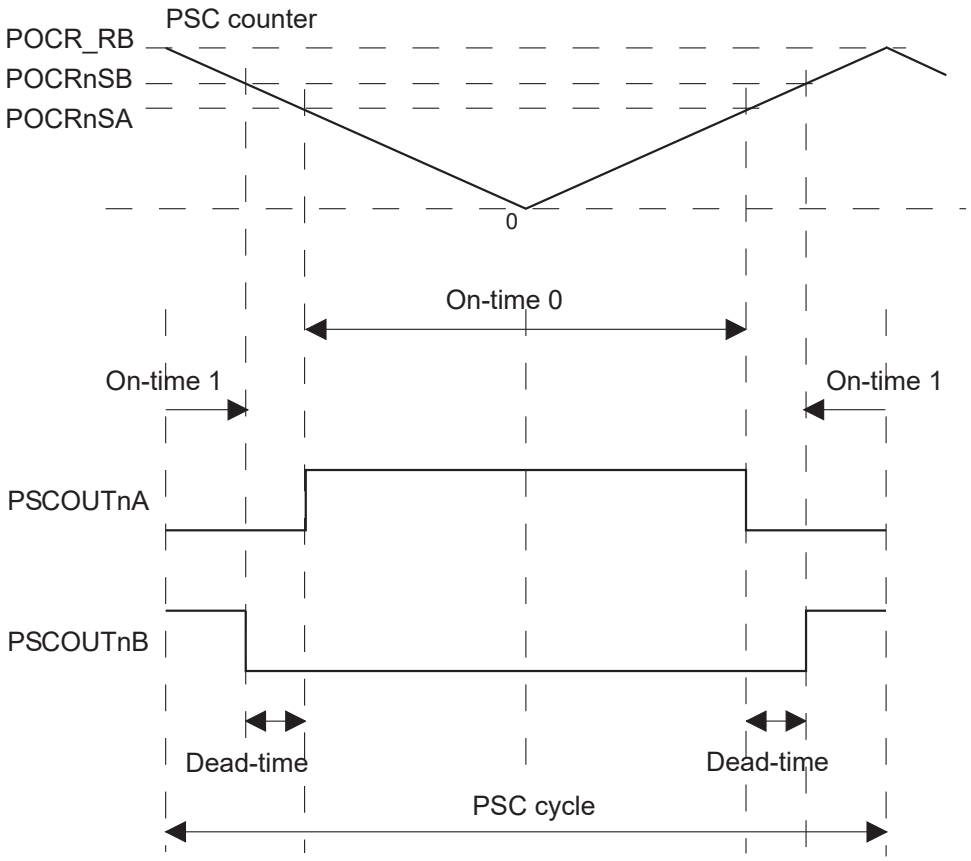


Note: See PCTL. (PCCYC = 1).

19.5.3.2 Center Aligned Mode

In Center Aligned mode, the center of PSCOUTnA and PSCOUTnB signals are centered.

Figure 19-6. PSCOUTnA & PSCOUTnB Basic Waveforms in Center Aligned Mode



On-Time 0 = 2 × POCRnSAH/L × $1/f_{CLKPSC}$

On-Time 1 = 2 × (POCR_RBH/L - POCRnSBH/L + 1) × $1/f_{CLKPSC}$

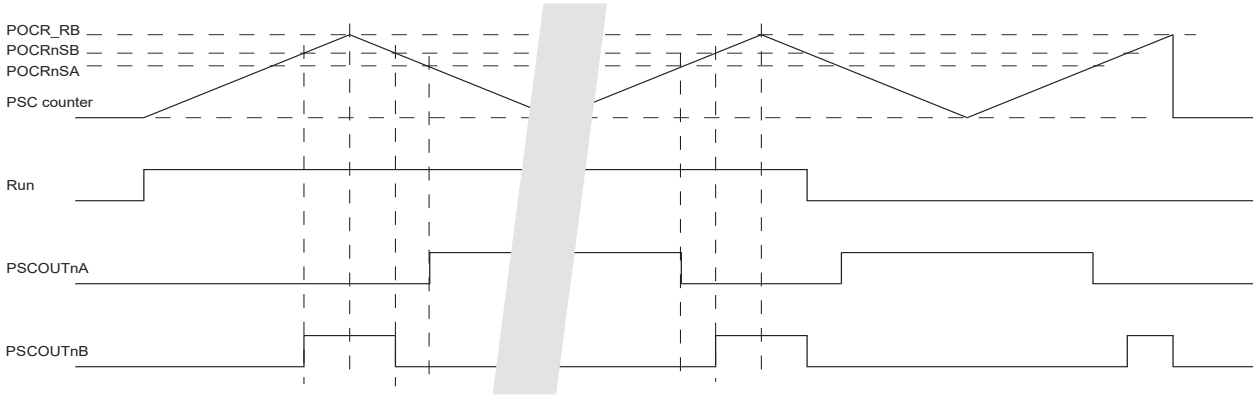
Dead-Time = (POCRnSBH/L - POCRnSAH/L) × $1/f_{CLKPSC}$

PSC Cycle = 2 × (POCR_RBH/L + 1) × $1/f_{CLKPSC}$

Note: Minimal value for PSC Cycle = 2 × $1/f_{CLKPSC}$

Note that in center aligned mode, POCRnRAH/L is not required (as it is in one ramp mode) to control PSC Output waveform timing. This allows POCRnRAH/L to be freely used to adjust ADC synchronization. See Analog Synchronization

Figure 19-7. Controlled Start and Stop Mechanism in Centered Mode

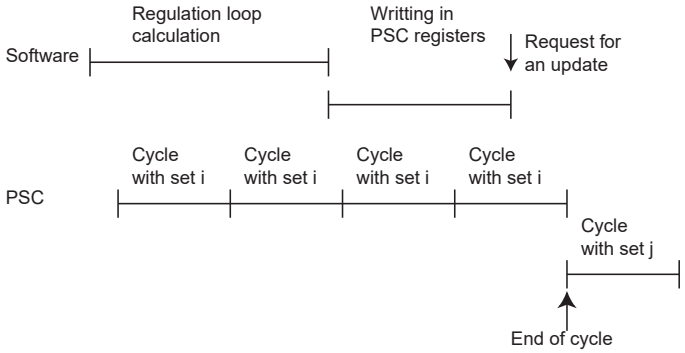


Note: See PCTL. (PCCYC = 1).

19.6 Update of values

To avoid asynchronous and incoherent values in a cycle, if an update of one of several values is necessary, all values are updated at the same time at the end of the cycle by the PSC. The new set of values is calculated by software and the update is initiated by software.

Figure 19-8. Update at the end of complete PSC cycle.



The software can stop the cycle before the end to update the values and restart a new PSC cycle.

19.6.1 Value update synchronization

New timing values or PSC output configuration can be written during the PSC cycle. Thanks to the LOCK configuration bit, the new whole set of values can be taken into account after the end of the PSC cycle.

When LOCK configuration bit is set, there is no update. The update of the PSC internal registers will be done at the end of the PSC cycle if the LOCK bit is released to zero.

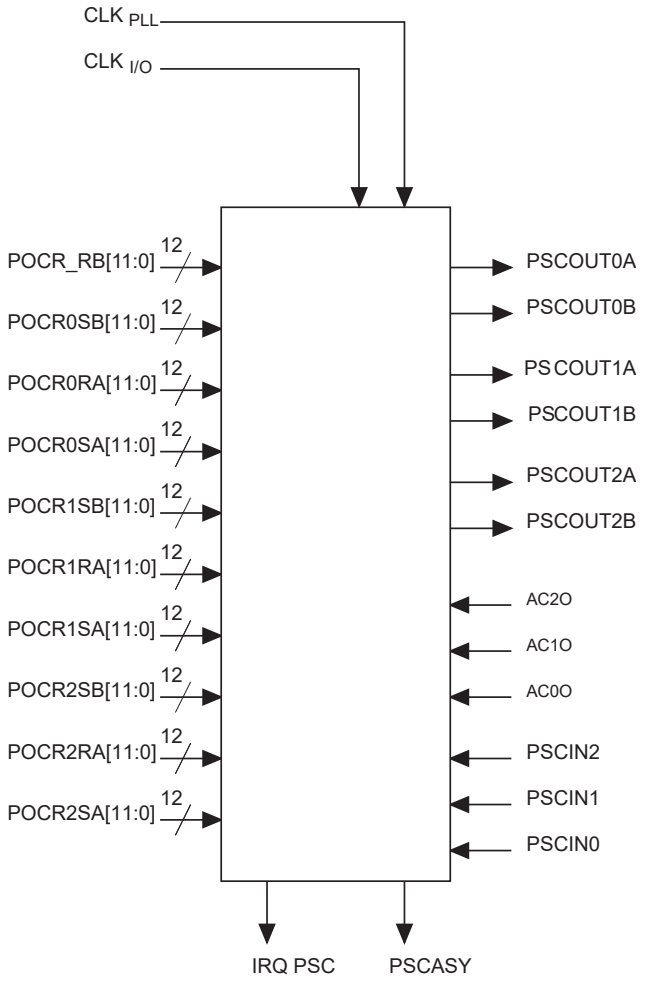
The registers which update is synchronized thanks to LOCK are POC, POCRnSA, POCRnRA, POCRnSB and POCR_RB.

19.7 Overlap Protection

Thanks to Overlap Protection two outputs on a same module cannot be active at the same time. So it cannot generate cross conduction. This feature can be disactivated thanks to POVEn (PSC Overlap Enable).

19.8 Signal Description

Figure 19-9. PSC External Block View



19.8.1 Input Description

Table 19-1. Internal inputs

Name	Description	Type width
POCR_RB[11:0]	Compare value which reset signal on Part B (PSCOUTnB)	Register 12 bits
POCRnSB[11:0]	Compare value which set signal on Part B (PSCOUTnB)	Register 12 bits
POCRnRA[11:0]	Compare value which reset signal on Part A (PSCOUTnA)	Register 12 bits
POCRnSA[11:0]	Compare value which set signal on Part A (PSCOUTnA)	Register 12 bits
CLK I/O	Clock input from I/O clock	Signal
CLK PLL	Clock input from PLL	Signal

Name	Description	Type width
AC0O	Analog Comparator 0 Output	Signal
AC1O	Analog Comparator 1 Output	Signal
AC2O	Analog Comparator 2 Output	Signal

Table 19-2. Block Inputs

Name	Description	Type width
PSCIN0	Input 0 used for fault function	Signal
PSCIN1	Input 1 used for fault function	Signal
PSCIN2	Input 2 used for fault function	Signal

19.8.2 Output description

Table 19-3. Block Outputs

Name	Description	Type width
PSCOUT0A	PSC Module 0 Output A	Signal
PSCOUT0B	PSC Module 0 Output B	Signal
PSCOUT1A	PSC Module 1 Output A	Signal
PSCOUT1B	PSC Module 1 Output B	Signal
PSCOUT2A	PSC Module 2 Output A	Signal
PSCOUT2B	PSC Module 2 Output B	Signal

Table 19-4. Internal Outputs

Name	Description	Type width
IRQPSCn	PSC interrupt request: two souces, overflow, fault	Signal
PSCASY	ADC synchronization (+ Amplifier Syncho) **Note:** See Analog Synchronization	Signal

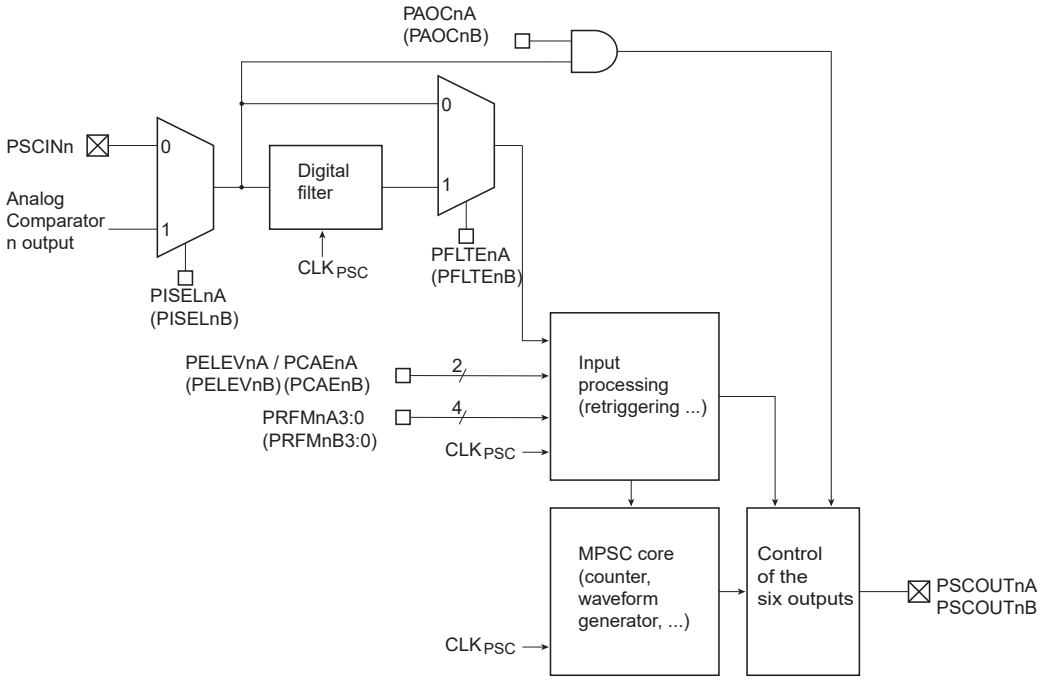
19.9 PSC Input

For detailed information on the PSC, refer to the Application Note ‘AVR138: ATmega32M1 Family PSC Cookbook’, available on our web site.

Each module 0, 1 and 2 of PSC has its own system to take into account one PSC input configurable in the PSC Module n Input Control Register. See “PMICn – PSC Module n Input Control Register”. PSCINn input can act as a Retrigger or Fault input.

Each block A or B is also configured by this PSC Module n Input Control Register (PMICn).

Figure 19-10. PSC Input Module



19.9.1 PSC input configuration

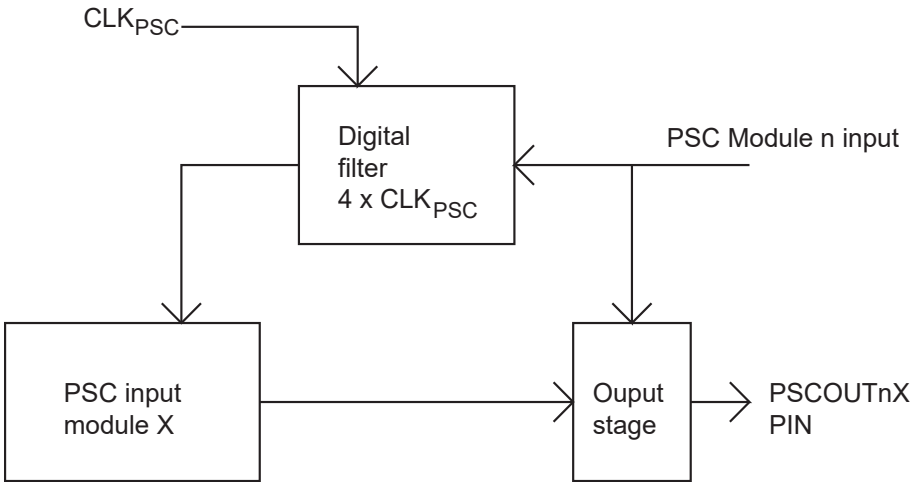
The PSC input configuration is done by programming bits in configuration registers.

19.9.1.1 Filter enable

If the “Filter Enable” bit is set, a digital filter of four cycles is inserted before evaluation of the signal. The disable of this function is mainly needed for prescaled PSC clock sources, where the noise cancellation gives too high latency.

Important: If the digital filter is active, the level sensitivity is true also with a disturbed PSC clock to deactivate the outputs (emergency protection of external component). Likewise when used as fault input, PSC Module n Input A or Input B have to go through PSC to act on PSCOUTn0/1/2 outputs. This way needs that CLK_{PSC} is running. So thanks to PSC Asynchronous Output Control bit (PAOCnA/B), PSCINn input can desactivate directly the PSC outputs. Notice that in this case, input is still taken into account as usually by Input Module System as soon as CLK_{PSC} is running.

Figure 19-11. PSC input filtering.



19.9.1.2 Signal polarity

One can select the active edge (edge modes) or the active level (level modes). See PELEVnx bit description in Section "PMICn – PSC Module n Input Control Register", page 144.

If PELEVnx bit set, the significant edge of PSCn Input A or B is rising (edge modes) or the active level is high (level modes) and vice versa for unset/falling/low.

- In 2- or 4-ramp mode, PSCn Input A is taken into account only during Dead-Time0 and On-Time0 period (respectively Dead-Time1 and On-Time1 for PSCn Input B)

- In 1-ramp-mode PSC Input A or PSC Input B act on the whole ramp

19.9.1.3 Input mode operation

Thanks to four configuration bits (PRFM3:0), it is possible to define the mode of the PSC inputs.

Table 19-5. PSC Input mode operation.

PRFMn2:0	Description
000b	No action, PSC Input is ignored
001b	Disactivate module n Outputs A
010b	Disactivate module n Output B
011b	Disactivate module n Output A & B
10x	Disactivate all PSC Output
11xb	Halt PSC and Wait for Software Action

Note: All following examples are given with rising edge or high level active inputs.

19.10 PSC input modes 001b to 10xb: Deactivate outputs without changing timing

Figure 19-12. PSC behavior versus PSCn input in mode 001b to 10xb

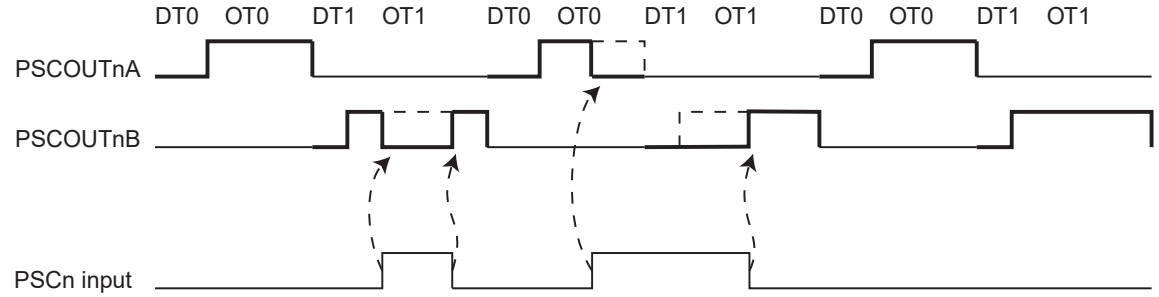
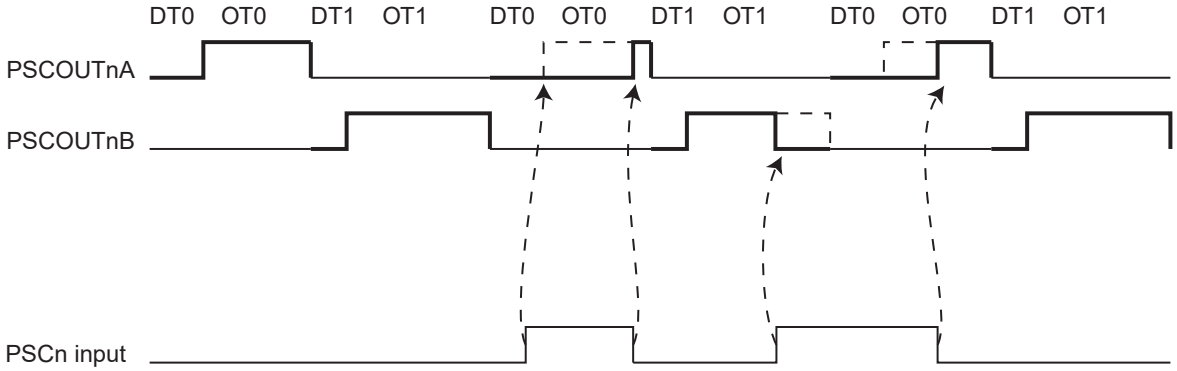


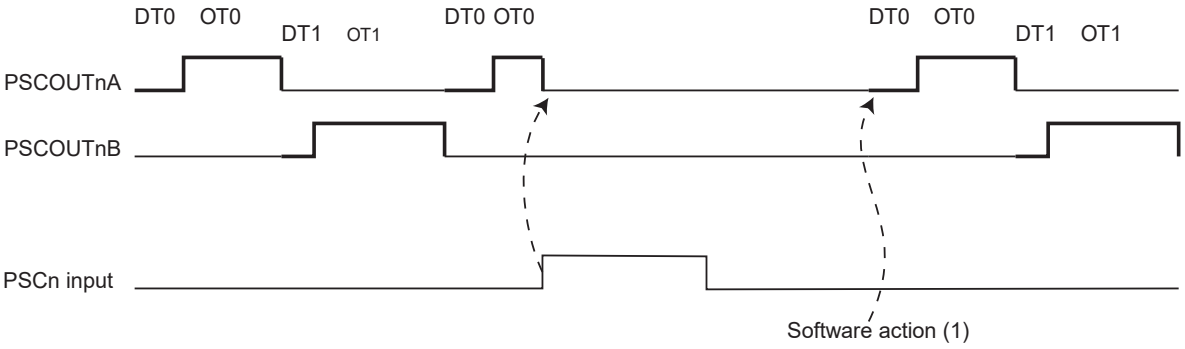
Figure 19-13. PSC behavior versus PSCn Input A or Input B in fault mode 4



PSCn input acts indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

19.11 PSC Input Mode 11xb: Halt PSC and wait for software action

Figure 19-14. PSC behavior versus PSCn Input A in fault mode 11xb



Note: Software action is the setting of the PRUNn bit in PCTLn register.

Used in fault mode 7, PSCn Input A or PSCn Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

19.12 Analog Synchronization

Each PSC module generates a signal to synchronize the ADC sample and hold; synchronization is mandatory for measurements.

This signal can be selected between all falling or rising edges of PSCOUTnA or PSCOUTnB outputs.

In center-aligned mode, POCRnRAH/L is not used, so it can be used to specify the synchronization of the ADC. It this case, its minimum value is 1.

19.13 Interrupt handling

As each PSC module can be dedicated for one function, each PSC has its own interrupt system.

List of interrupt sources:

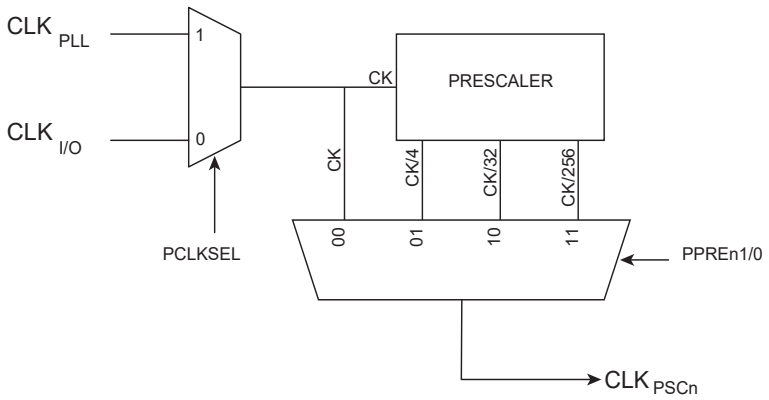
- Counter reload (end of On Time 1)
- PSC Input event (active edge or at the beginning of level configured event)
- PSC Mutual Synchronization Error

19.14 PSC clock sources

Each PSC has two clock inputs:

- CLK PLL from the PLL
- CLK I/O

Figure 19-15. Clock selection.



PCLKSELn bit in PSC Control Register (PCTL) is used to select the clock source.

PPREn1/0 bits in PSC Control Register (PCTL) are used to select the divide factor of the clock.

Table 19-6. Output clock versus selection and prescaler.

PCLKSELn	PPREn1	PPREn0	CLKPSCn output
0	0	0	CLK I/O
0	0	1	CLK I/O / 4
0	1	0	CLK I/O / 32
0	1	1	CLK I/O / 256
1	0	0	CLK PLL
1	0	1	CLK PLL / 4
1	1	0	CLK PLL / 32
1	1	1	CLK PLL / 256

19.15 Interrupts

This section describes the specifics of the interrupt handling as performed in the ATmegaS64M1.

19.15.1 Interrupt vector

PSC provides two interrupt vectors:

- PSC_End (End of Cycle): When enabled and when a match with POCR_RB occurs
- PSC_Fault (Fault Event): When enabled and when a PSC input detects a Fault event

19.15.2 PSC interrupt vectors in ATmegaS64M1

Table 19-7. PSC interrupt vectors.

Vector no.	Program address	Source	Interrupt definition
-	-	-	-
5	0x0004	PSC_Fault	PSC fault event
6	0x0005	PSC_End	PSC end of Cycle
-	-	-	-
-	-	-	-

19.16 Register Description

19.16.1 PSC Output Configuration

Name: POC
Offset: 0xB6
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
			POEN2B	POEN2A	POEN1B	POEN1A	POEN0B	POEN0A
Access								
Reset			0	0	0	0	0	0

Bit 5 – POEN2B PSC Output 2B Enable

Value	Description
0	I/O pin affected to PSCOUT2B acts as a standard port.
1	I/O pin affected to PSCOUT2B is connected to the PSC module 2 waveform generator B output and is set and clear according to the PSC operation.

Bit 4 – POEN2A PSC Output 2A Enable

Value	Description
0	I/O pin affected to PSCOUT2A acts as a standard port.
1	I/O pin affected to PSCOUT2A is connected to the PSC module 2 waveform generator A output and is set and clear according to the PSC operation.

Bit 3 – POEN1B PSC Output 2B Enable

Value	Description
0	I/O pin affected to PSCOUT1B acts as a standard port.
1	I/O pin affected to PSCOUT1B is connected to the PSC module 1 waveform generator B output and is set and clear according to the PSC operation.

Bit 2 – POEN1A PSC Output 1A Enable

Value	Description
0	I/O pin affected to PSCOUT1A acts as a standard port.
1	I/O pin affected to PSCOUT1A is connected to the PSC module 1 waveform generator A output and is set and clear according to the PSC operation.

Bit 1 – POEN0B PSC Output 0B Enable

Value	Description
0	I/O pin affected to PSCOUT0B acts as a standard port.
1	I/O pin affected to PSCOUT0B is connected to the PSC module 0 waveform generator B output and is set and clear according to the PSC operation.

Bit 0 – POEN0A PSC Output 0A Enable

Value	Description
0	I/O pin affected to PSCOUT0A acts as a standard port.
1	I/O pin affected to PSCOUT0A is connected to the PSC module 0 waveform generator A output and is set and clear according to the PSC operation.

19.16.2 PSC Synchro Configuration

Name: PSYNC
Offset: 0xB4
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
			PSYNC2[1:0]		PSYNC1[1:0]		PSYNC0[1:0]	
Access								
Reset			0	0	0	0	0	0

Bits 0:1, 2:3, 4:5 – PSYNC Synchronization Out for ADC Selection
Select the polarity and signal source for generating a signal which will be sent from module n to the ADC for synchronization.

Value	Name	Description
00	One Ramp mode	Send signal on leading edge of PSCOUTnA (match with POCRnSA)
01	One Ramp mode	Send signal on trailing edge of PSCOUTnA (match with POCRnRA or fault/retrigger on part A)
10	One Ramp mode	Send signal on leading edge of PSCOUTnB (match with POCRnSB)
11	One Ramp mode	Send signal on trailing edge of PSCOUTnB (match with POCR_RB or fault/retrigger on part B)
00	Centered mode	Send signal on match with POCRnRA (during counting down of PSC). The min value of POCRnRA must be 1
01	Centered mode	Send signal on match with POCRnRA (during counting up of PSC). The min value of POCRnRA must be 1
10	Centered mode	No synchronization signal
11	Centered mode	No synchronization signal

19.16.3 PSC Output Compare SA Register

Name: POCRnSA
Offset: 0xA0 + n*0x06 [n=0..2]
Reset: 0x0
Property: R/W

Note: n = 0 to 2 according to the module number

The Output Compare Registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSC counter value. A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the associated pin.

The Output Compare Registers are 16-bit and 12-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.

Bit	15	14	13	12	11	10	9	8
					POCRnSAH[3:0]			
Access								
Reset					0	0	0	0

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

Bits 11:8 – POCRnSAH[3:0] PSC Output Compare SA High Bits

Bits 7:0 – POCRnSAL[7:0] PSC Output Compare SA Low Byte

19.16.4 PSC Output Compare RA Register

Name: POCRnRA
Offset: 0xA2 + n*0x06 [n=0..2]
Reset: 0x0
Property: R/W

Note: n = 0 to 2 according to module number.

The Output Compare Registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSC counter value. A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the associated pin.

The Output Compare Registers are 16-bit and 12-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.

Bit	15	14	13	12	11	10	9	8
					POCRnRAH[3:0]			
Access								
Reset					0	0	0	0

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

Bits 11:8 – POCRnRAH[3:0] PSC Output Compare RA High Bits

Bits 7:0 – POCRnRAL[7:0] PSC Output Compare RA Low Byte

19.16.5 PSC Output Compare SB Register

Name: POCRnSB
Offset: 0xA4 + n*0x06 [n=0..2]
Reset: 0x0
Property: R/W

Note: n = 0 to 2 according to module number.

The Output Compare Registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSC counter value. A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the associated pin.

The Output Compare Registers are 16-bit and 12-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.

Bit	15	14	13	12	11	10	9	8
					POCRnSBH[3:0]			
Access								
Reset					0	0	0	0

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

Bits 11:8 – POCRnSBH[3:0] PSC Output Compare SB High bits

Bits 7:0 – POCRnSBL[7:0] PSC Output Compare SB Low byte

19.16.6 PSC Output Compare RB Register

Name: POCR_RB
Offset: 0xb2
Reset: 0x0
Property: R/W

The Output Compare Registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSC counter value. A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the associated pin.

The Output Compare Registers are 16-bit and 12-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.

Bit	15	14	13	12	11	10	9	8
					POCR_RBH[11:8]			
Access								
Reset					0	0	0	0

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

Bits 11:8 – POCR_RBH[11:8] PSC Output Compare RB High bits

Bits 7:0 – POCR_RBL[7:0] PSC Output Compare RB Low byte

19.16.7 PSC Configuration Register

Name: PCNF
Offset: 0xB5
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
			PULOCK	PMODE	POPB	POPA		
Access								
Reset			0	0	0	0		

Bit 5 – PULOCK PSC Update Lock
When this bit is set, the Output Compare Registers POCRnRA, POCRnSA, POCRnSB, POCR_RB and the PSC Output Configuration Registers POC can be written without disturbing the PSC cycles. The update of the PSC internal registers will be done if the PULOCK bit is released to zero.

Bit 4 – PMODE PSC Mode
Select the mode of PSC.

Value	Description
0	One Ramp mode (edge aligned)
1	Center Aligned mode

Bit 3 – POPB PSC B Output Polarity

Value	Description
0	PSC outputs B are active Low.
1	PSC outputs B are active High.

Bit 2 – POPA PSC A Output Polarity

Value	Description
0	PSC outputs A are active Low.
1	PSC outputs A are active High.

19.16.8 PSC Control Register

Name: PCTL
Offset: 0xB7
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	PPRE[1:0]		PCLKSEL				PCCYC	PRUN
Access								
Reset	0	0	0				0	0

Bits 7:6 – PPRE[1:0] PSC Prescaler Select
These two bits select the PSC input clock division factor. All generated waveforms will be modified by this factor.

Value	Description
00	No divider on PSC input clock
01	Divide the PSC input clock by 4
10	Divide the PSC input clock by 32
11	Divide the PSC clock by 256

Bit 5 – PCLKSEL PSC Input Clock Select
This bit is used to select between CLK_{PLL} or CLK_{IO} clocks.

Value	Description
1	Set this bit to select the fast clock input (CLK_{PLL}).
0	Clear this bit to select the slow clock input (CLK_{IO}).

Bit 1 – PCCYC PSC Complete Cycle
When this bit is set, the PSC completes the entire waveform cycle before halt operation requested by clearing PRUN.

Bit 0 – PRUN PSC Run
Writing this bit to one starts the PSC.

19.16.9 PSC Module n Input Control Register

Name: PMICn
Offset: 0xB8 + n*0x01 [n=0..2]
Reset: 0x0
Property: R/W

The Input Control Registers are used to configure the two PSC's Retrigger/Fault block A & B. The two blocks are identical, thus are configured in the same way.

Bit	7	6	5	4	3	2	1	0
	POVENn	PISELn	PELEVn	PFLTEn	PAOCn	PRFMn[2:0]		
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – POVENn PSC Module n Overlap Enable
Set this bit to deactivate the Overlap Protection. Refer to Overlap Protection..

Bit 6 – PISELn PSC Module n Input Select

Value	Description
0	Selects PSCINn as module n input.
1	Selects Comparator n output as module n input.

Bit 5 – PELEVn PSC Module n Input Level Selector

Value	Description
0	The low level of selected input generates the significant event for fault function.
1	The high level of selected input generates the significant event for fault function.

Bit 4 – PFLTEn PSC Module n Input Filter Enable
Setting this bit (to one) activates the input noise canceler. When the noise canceler is activated, the input from the input pin is filtered. The filter function requires four successive equal valued samples of the input pin to change its output. The input is therefore delayed by four oscillator cycles when the noise canceler is enabled.

Bit 3 – PAOCn PSC Module n 0 Asynchronous Output Control
When this bit is clear, fault input can act directly to PSC module n outputs A & B. Refer to PSC input configuration.

Bits 2:0 – PRFMn[2:0] PSC Module n Input Mode
These three bits define the mode of operation of the PSC inputs.

Value	Description
0b000	No action, PSC input is ignored.
0b001	Deactivates module n Outputs A.
0b010	Deactivates module n Output B.
0b011	Deactivates module n Output A & B.
0b10x	Deactivates all PSC Output.
0b11x	Halts PSC and waits for software action.

19.16.10 PSC Interrupt Mask Register

Name: PIM
Offset: 0xBB
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
					PEVE2	PEVE1	PEVE0	PEOPE
Access								
Reset					0	0	0	0

Bit 3 – PEVE2 PSC External Event 2 Interrupt Enable
When this bit is set, an external event which can generates a fault on module 2 generates also an interrupt.

Bit 2 – PEVE1 PSC External Event 1 Interrupt Enable
When this bit is set, an external event which can generates a fault on module 1 generates also an interrupt.

Bit 1 – PEVE0 PSC External Event 0 Interrupt Enable
When this bit is set, an external event which can generates a fault on module 0 generates also an interrupt.

Bit 0 – PEOPE PSC End Of Cycle Interrupt Enable
When this bit is set, an interrupt is generated when PSC reaches the end of the whole cycle.

19.16.11 PSC Interrupt Flag Register

Name: PIFR
Offset: 0xBC
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
					PEV2	PEV1	PEV0	PEOP
Access								
Reset					0	0	0	0

Bit 3 – PEV2 PSC External Event 2 Interrupt
This bit is set by hardware when an external event which can generates a fault on module 2 occurs. Must be cleared by software by writing a one to its location. This bit can be read even if the corresponding interrupt is not enabled (PEVE2 bit = 0).

Bit 2 – PEV1 PSC External Event 1 Interrupt
This bit is set by hardware when an external event which can generates a fault on module 1 occurs. Must be cleared by software by writing a one to its location. This bit can be read even if the corresponding interrupt is not enabled (PEVE1 bit = 0).

Bit 1 – PEV0 PSC External Event 0 Interrupt
This bit is set by hardware when an external event which can generates a fault on module 0 occurs. Must be cleared by software by writing a one to its location. This bit can be read even if the corresponding interrupt is not enabled (PEVE0 bit = 0).

Bit 0 – PEOP PSC End Of Cycle Interrupt
This bit is set by hardware when an “end of PSC cycle” occurs. Must be cleared by software by writing a one to its location. This bit can be read even if the corresponding interrupt is not enabled (PEOPE bit = 0).

20. Serial Peripheral Interface (SPI)

20.1 Features

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

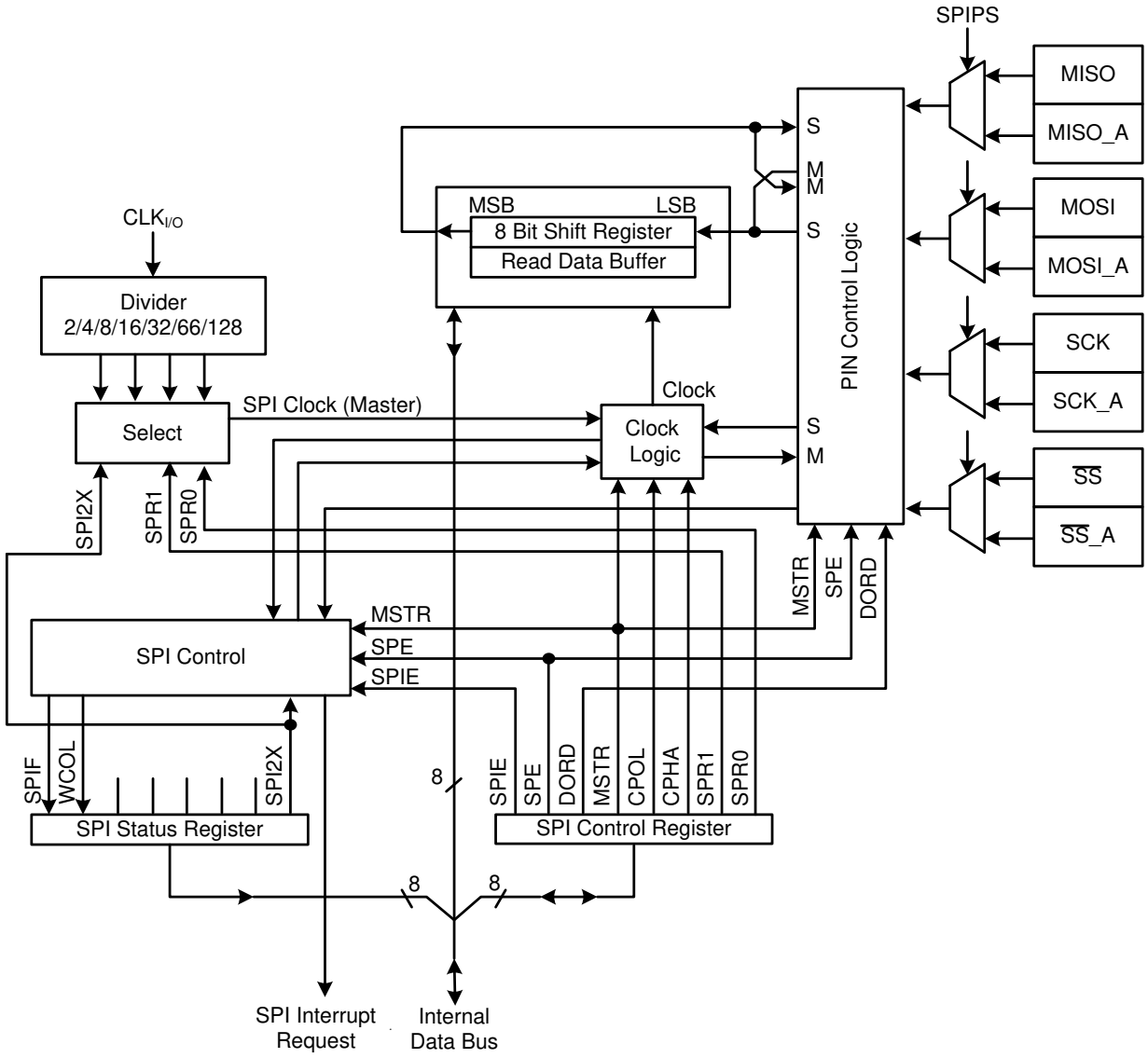
20.2 Overview

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 be used in Master SPI mode, refer to chapter *USART in SPI Mode*.

To enable the SPI module, Power Reduction Serial Peripheral Interface bit in the Power Reduction Register (PRR.PRSPI) must be written to '0'.

Figure 20-1. SPI Block Diagram



Note: Refer to the pinout description and the I/O Port description for SPI pin placement.

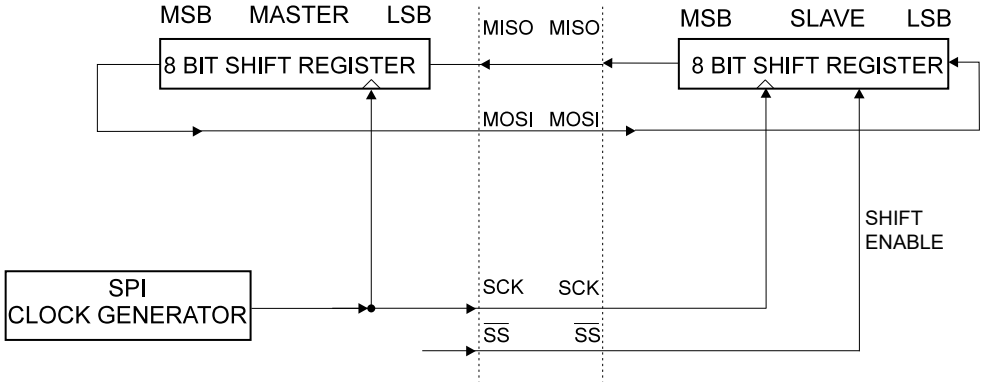
The interconnection between master and slave CPUs with SPI is shown in the figure below. The system consists of two shift registers and a master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired slave. Master and slave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

When configured as a master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data register starts the SPI clock generator, and the hardware shifts the 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, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer register for later use.

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

Figure 20-2. 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 frequency of the SPI clock should never exceed $f_{CLKIO}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to the table below. For more details on automatic port overrides, refer to the I/O Ports description.

Table 20-1. SPI Pin Overrides

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

Note: 1. See the I/O Ports description for details on defining 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, for example, if MOSI is placed on pin PB1, replace `DD_MOSI` with `DDB1` 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;
}
```

20.3 $\overline{SS}$ Pin Functionality

20.3.1 Slave Mode

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

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

20.3.2 Master Mode

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

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

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

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

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

20.4 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 20-2. 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)
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 20-3. SPI Transfer Format with CPHA = 0

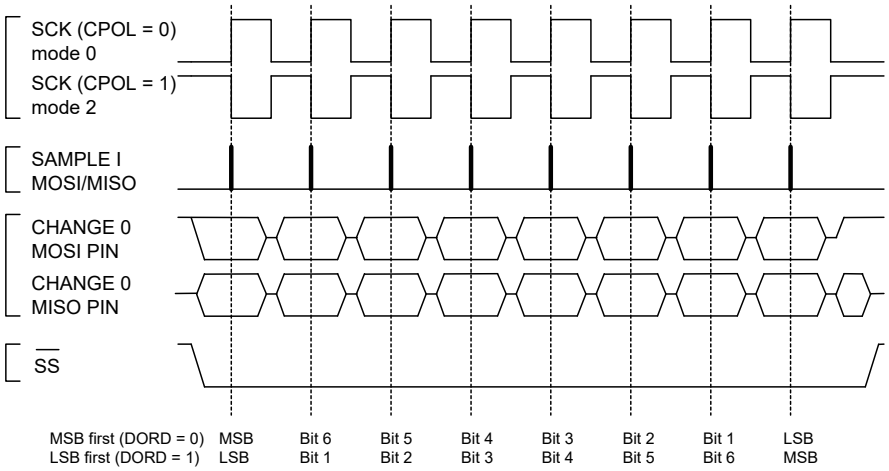
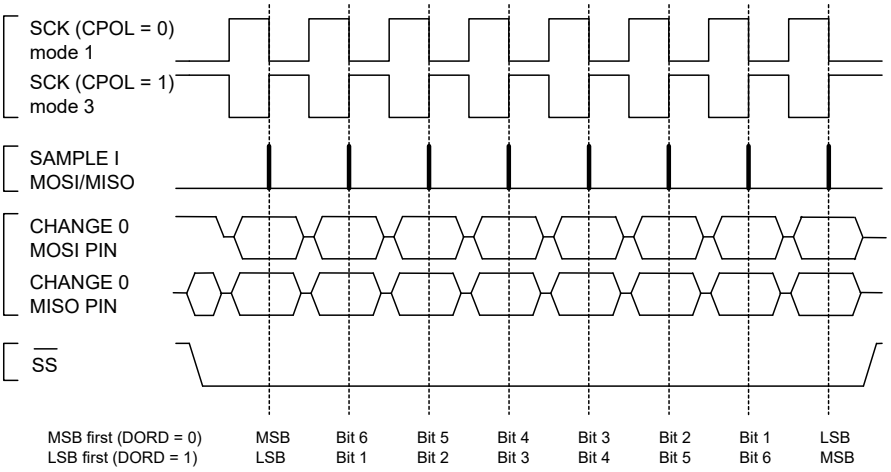


Figure 20-4. SPI Transfer Format with CPHA = 1



20.5 Register Description

20.5.1 MCU Control Register

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

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.

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

Bit 7 – SPIPS SPI Pin Redirection
Thanks to SPIPS (SPI Pin Select) in MCUCR Sfr, SPI pins can be redirected. Note that the programming port is always located on alternate SPI port.

Value	Description
0	When the SPIPS bit is written to zero, the SPI signals are directed on pins MISO, MOSI, SCK and SS
1	When the SPIPS bit is written to one,the SPI signals are directed on alternate SPI pins, MISO_A, MOSI_A, SCK_A and SS_A

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 same cycle as IVCE is written, and interrupts 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 programmed, 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 the 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);
}
```

20.5.2 SPI Control Register

Name: SPCR
Offset: 0x4C
Reset: 0x00
Property: When addressing as I/O register: address offset is 0x2C

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.

Bit	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR[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 – SPIE SPI Interrupt Enable
This bit causes the SPI interrupt to be executed if the SPIF bit in the SPSR register is set and if the global interrupt enable bit in SREG is set.

Bit 6 – SPE SPI Enable
When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 – DORD Data Order
When the DORD bit is written to one, the LSB of the data word is transmitted first.

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

Bit 4 – MSTR Master/Slave Select
This bit selects the Master SPI mode when written to one, and the 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 Master SPI mode.

Bit 3 – CPOL Clock Polarity
When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 20-3 and Figure 20-4 for an example. The CPOL functionality is summarized below:

Table 20-3. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Bit 2 – CPHA Clock Phase
The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 20-3 and Figure 20-4 for an example. The CPHA functionality is summarized below:

Table 20-4. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Bits 1:0 – SPR[1:0] SPI Clock Rate Select 1 and 0
These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the CLKIO frequency f_{CLKIO} is shown in the table below.

Table 20-5. Relationship Between SCK and Oscillator Frequency

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

20.5.3 SPI Status Register

Name: SPSR
Offset: 0x4D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2D

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.

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL						SPI2X
Access	R	R						R/W
Reset	0	0						0

Bit 7 – SPIF SPI Interrupt Flag
When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Bit 6 – WCOL Write Collision Flag
The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

Bit 0 – SPI2X Double SPI Speed Bit
When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (refer to Table 20-5). 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 $f_{CLKIO}/4$ or lower.

The SPI interface is also used for program memory and EEPROM downloading or uploading. See *Serial Downloading* for serial programming and verification.

20.5.4 SPI Data Register

Name: SPDR
Offset: 0x4E
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x2E

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.

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.

21. CAN – Controller Area Network

21.1 Features

- Full CAN controller
- Fully compliant with CAN standard rev 2.0 A and rev 2.0 B
- Six MOb (Message Object) with their own:
  - 11 bits of Identifier Tag (rev 2.0 A), 29 bits of Identifier Tag (rev 2.0 B)
  - 11 bits of Identifier Mask (rev 2.0 A), 29 bits of Identifier Mask (rev 2.0 B)
  - Eight bytes data buffer (static allocation)
  - Tx, Rx, frame buffer or automatic reply configuration
  - Time stamping
- 1Mbit/s maximum transfer rate at 8MHz
- TTC timer
- Listening mode (for spying or autobaud)

21.2 Overview

The Controller Area Network (CAN) protocol is a real-time, serial, broadcast protocol with a very high level of security. The ATmegaS64M1 CAN controller is fully compatible with the CAN Specification 2.0 Part A and Part B. It delivers the features required to implement the kernel of the CAN bus protocol according to the ISO/OSI Reference Model:

- The data link layer
  - the Logical Link Control (LLC) sublayer
  - the Medium Access Control (MAC) sublayer
- The physical layer
  - the Physical Signalling (PLS) sublayer
  - not supported - the Physical Medium Attach (PMA)
  - not supported - the Medium Dependent Interface (MDI)

The CAN controller is able to handle all types of frames (data, remote, error and overload) and achieves a bitrate of 1Mbit/s.

21.3 CAN protocol

The CAN protocol is an international standard defined in the ISO 11898 for high speed and ISO 11519-2 for low speed.

21.3.1 Principles

CAN is based on a broadcast communication mechanism. This broadcast communication is achieved by using a message oriented transmission protocol. These messages are identified by using a message identifier. Such a message identifier has to be unique within the whole network and it defines not only the content but also the priority of the message.

The priority at which a message is transmitted compared to another less urgent message is specified by the identifier of each message. The priorities are laid down during system design in the form of

corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority.

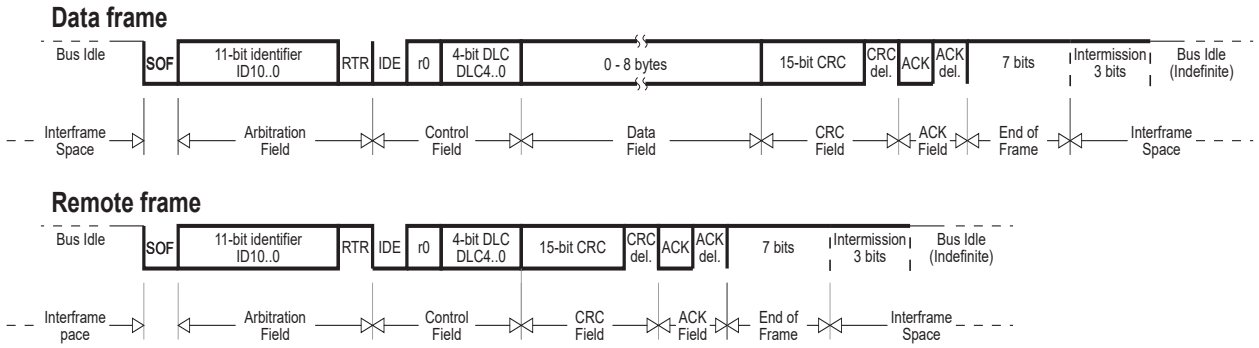
Bus access conflicts are resolved by bit-wise arbitration on the identifiers involved by each node observing the bus level bit for bit. This happens in accordance with the “wired and” mechanism, by which the dominant state overwrites the recessive state. The competition for bus allocation is lost by all nodes with recessive transmission and dominant observation. All the "losers" automatically become receivers of the message with the highest priority and do not re-attempt transmission until the bus is available again.

21.3.2 Message formats

The CAN protocol supports two message frame formats, the only essential difference being in the length of the identifier. The CAN standard frame, also known as CAN 2.0 A, supports a length of 11 bits for the identifier, and the CAN extended frame, also known as CAN 2.0 B, supports a length of 29 bits for the identifier.

21.3.2.1 CAN standard frame

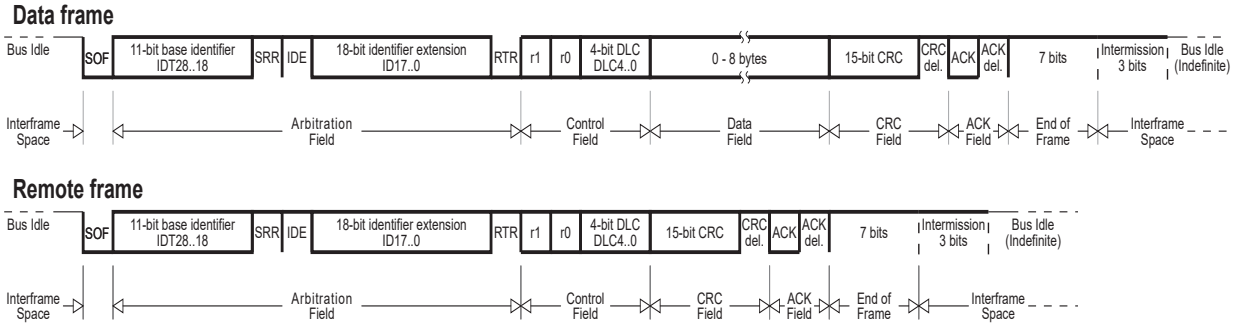
Figure 21-1. CAN standard frames



A message in the CAN standard frame format begins with the "Start Of Frame (SOF)", this is followed by the "Arbitration field" which consist of the identifier and the "Remote Transmission Request (RTR)" bit used to distinguish between the data frame and the data request frame called remote frame. The following "Control field" contains the "IDentifier Extension (IDE)" bit and the "Data Length Code (DLC)" used to indicate the number of following data bytes in the "Data field". In a remote frame, the DLC contains the number of requested data bytes. The "Data field" that follows can hold up to 8 data bytes. The frame integrity is guaranteed by the following "Cyclic Redundant Check (CRC)" sum. The "ACKnowledge (ACK) field" compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by the receivers which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by "End Of Frame (EOF)". The "Intermission Frame Space (IFS)" is the minimum number of bits separating consecutive messages. If there is no following bus access by any node, the bus remains idle.

21.3.2.2 CAN extended frame

Figure 21-2. CAN extended frames



A message in the CAN extended frame format is likely the same as a message in CAN standard frame format. The difference is the length of the identifier used. The identifier is made up of the existing 11-bit identifier (base identifier) and an 18-bit extension (identifier extension). The distinction between CAN standard frame format and CAN extended frame format is made by using the IDE bit which is transmitted as dominant in case of a frame in CAN standard frame format, and transmitted as recessive in the other case.

21.3.2.3 Format co-existence

As the two formats have to co-exist on one bus, it is laid down which message has higher priority on the bus in the case of bus access collision with different formats and the same identifier / base identifier: The message in CAN standard frame format always has priority over the message in extended format.

There are three different types of CAN modules available:

- 2.0A - Considers 29 bit ID as an error
- 2.0B Passive - Ignores 29 bit ID messages
- 2.0B Active - Handles both 11 and 29 bit ID Messages

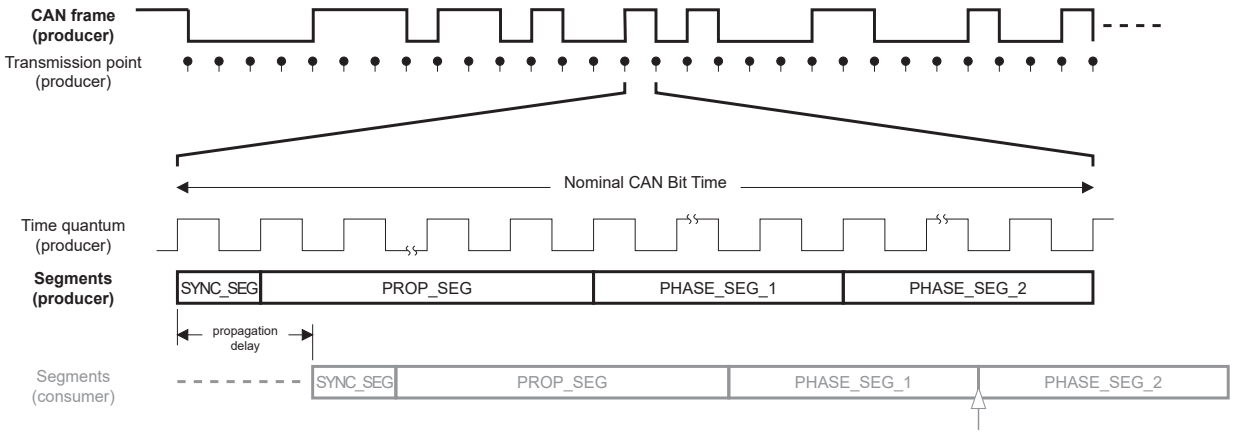
21.3.3 CAN bit timing

To ensure correct sampling up to the last bit, a CAN node needs to re-synchronize throughout the entire frame. This is done at the beginning of each message with the falling edge SOF and on each recessive to dominant edge.

21.3.3.1 Bit construction

One CAN bit time is specified as four non-overlapping time segments. Each segment is constructed from an integer multiple of the Time Quantum. The Time Quantum or TQ is the smallest discrete timing resolution used by a CAN node.

Figure 21-3. CAN bit construction



21.3.3.2 Synchronization segment

The first segment is used to synchronize the various bus nodes.

On transmission, at the start of this segment, the current bit level is output. If there is a bit state change between the previous bit and the current bit, then the bus state change is expected to occur within this segment by the receiving nodes.

21.3.3.3 Propagation time segment

This segment is used to compensate for signal delays across the network.

This is necessary to compensate for signal propagation delays on the bus line and through the transceivers of the bus nodes.

21.3.3.4 Phase Segment 1

Phase Segment 1 is used to compensate for edge phase errors.

This segment may be lengthened during re-synchronization.

21.3.3.5 Sample point

The sample point is the point of time at which the bus level is read and interpreted as the value of the respective bit. Its location is at the end of Phase Segment 1 (between the two Phase Segments).

21.3.3.6 Phase Segment 2

This segment is also used to compensate for edge phase errors.

This segment may be shortened during re-synchronization, but the length has to be at least as long as the Information Processing Time (IPT) and may not be more than the length of Phase Segment 1.

21.3.3.7 Information Processing Time

Information processing time (IPT) is the time required for the logic to determine the bit level of a sampled bit.

IPT begins at the sample point, is measured in TQ and is fixed at 2TQ for the CAN. Since Phase Segment 2 also begins at the sample point and is the last segment in the bit time, PS2 minimum shall not be less than the IPT.

21.3.3.8 Bit lengthening

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened to compensate for oscillator tolerances. If, for example, the transmitter oscillator is slower than

the receiver oscillator, the next falling edge used for resynchronization may be delayed. So Phase Segment 1 is lengthened in order to adjust the sample point and the end of the bit time.

21.3.3.9 Bit shortening

If, on the other hand, the transmitter oscillator is faster than the receiver one, the next falling edge used for resynchronization may be too early. So Phase Segment 2 in bit N is shortened in order to adjust the sample point for bit N+1 and the end of the bit time.

21.3.3.10 Synchronization jump width

The limit to the amount of lengthening or shortening of the Phase Segments is set by the Resynchronization Jump Width.

This segment may not be longer than Phase Segment 2.

21.3.3.11 Programming the sample point

Programming of the sample point allows "tuning" of the characteristics to suit the bus.

Early sampling allows more Time Quanta in the Phase Segment 2 so the Synchronization Jump Width can be programmed to its maximum. This maximum capacity to shorten or lengthen the bit time decreases the sensitivity to node oscillator tolerances, so that lower cost oscillators such as ceramic resonators may be used.

Late sampling allows more Time Quanta in the Propagation Time Segment which allows a poorer bus topology and maximum bus length.

21.3.3.12 Synchronization

Hard synchronization occurs on the recessive-to-dominant transition of the start bit. The bit time is restarted from that edge.

Re-synchronization occurs when a recessive-to-dominant edge doesn't occur within the Synchronization Segment in a message.

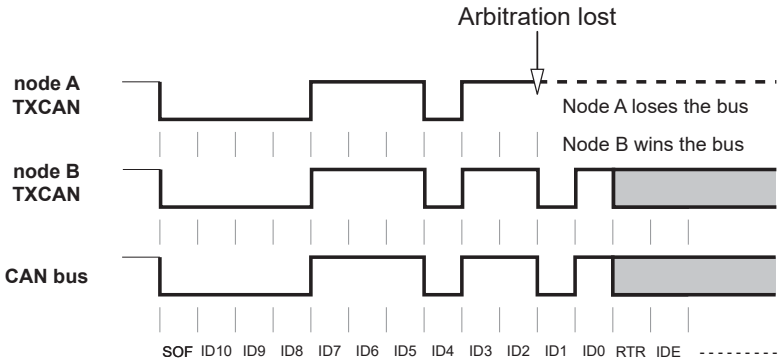
21.3.4 Arbitration

The CAN protocol handles bus accesses according to the concept called “Carrier Sense Multiple Access with Arbitration on Message Priority”.

During transmission, arbitration on the CAN bus can be lost to a competing device with a higher priority CAN Identifier. This arbitration concept avoids collisions of messages whose transmission was started by more than one node simultaneously and makes sure the most important message is sent first without time loss.

The bus access conflict is resolved during the arbitration field mostly over the identifier value. If a data frame and a remote frame with the same identifier are initiated at the same time, the data frame prevails over the remote frame (c.f. RTR bit).

Figure 21-4. Bus arbitration.



21.3.5 Errors

The CAN protocol signals any errors immediately as they occur. Three error detection mechanisms are implemented at the message level and two at the bit level:

21.3.5.1 Error at message level

- Cyclic Redundancy Check (CRC)
  The CRC safeguards the information in the frame by adding redundant check bits at the transmission end. At the receiver these bits are re-computed and tested against the received bits. If they do not agree there has been a CRC error.
- Frame Check
  This mechanism verifies the structure of the transmitted frame by checking the bit fields against the fixed format and the frame size. Errors detected by frame checks are designated "format errors".
- ACK Errors
  As already mentioned frames received are acknowledged by all receivers through positive acknowledgement. If no acknowledgement is received by the transmitter of the message an ACK error is indicated.

21.3.5.2 Error at bit level

- Monitoring
  The ability of the transmitter to detect errors is based on the monitoring of bus signals. Each node which transmits also observes the bus level and thus detects differences between the bit sent and the bit received. This permits reliable detection of global errors and errors local to the transmitter.
- Bit Stuffing
  The coding of the individual bits is tested at bit level. The bit representation used by CAN is "Non Return to Zero (NRZ)" coding, which guarantees maximum efficiency in bit coding. The synchronization edges are generated by means of bit stuffing.

21.3.5.3 Error signalling

If one or more errors are discovered by at least one node using the above mechanisms, the current transmission is aborted by sending an "error flag". This prevents other nodes accepting the message and thus ensures the consistency of data throughout the network. After transmission of an erroneous message that has been aborted, the sender automatically re-attempts transmission.

21.4 CAN Controller

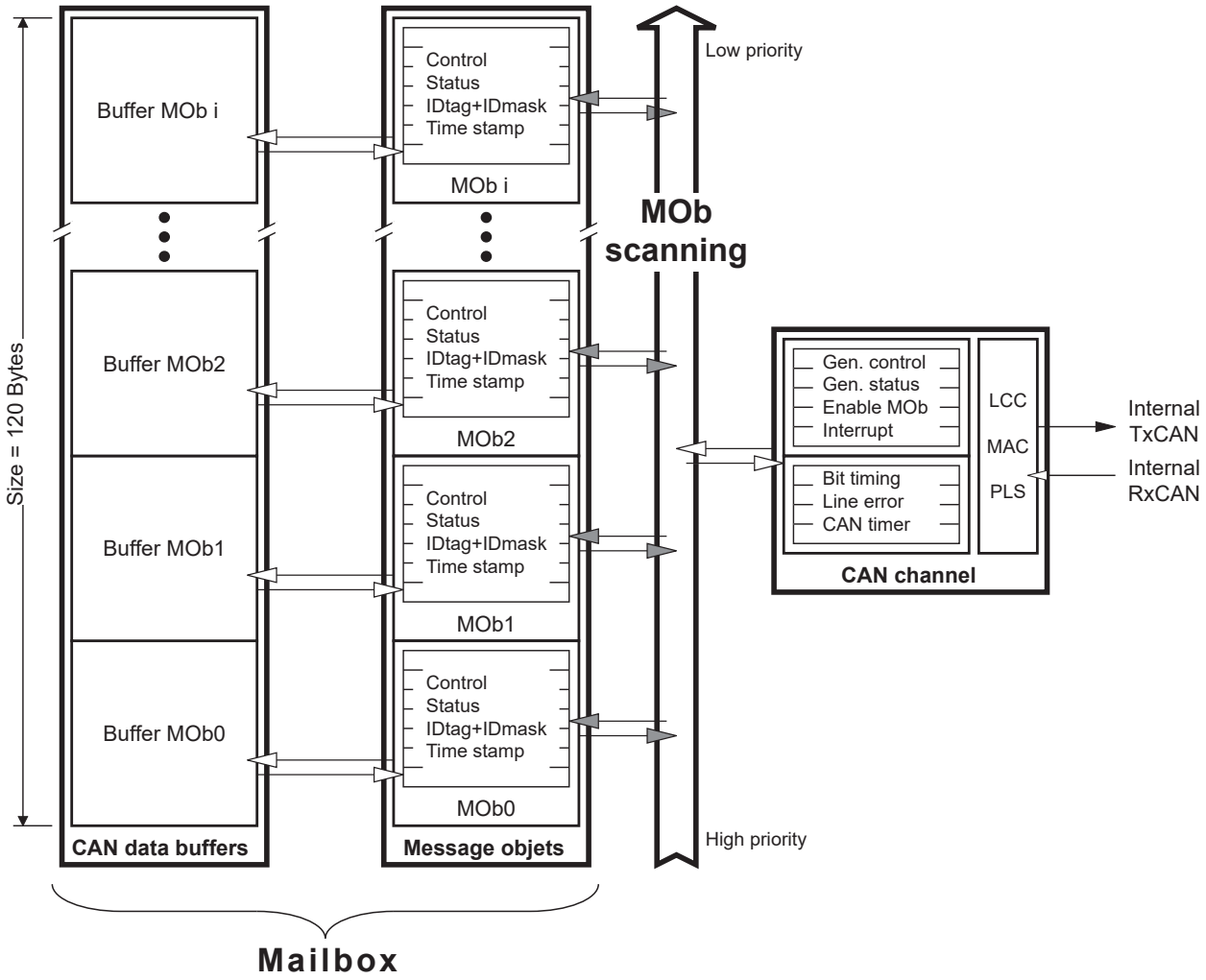
The CAN controller implemented in the ATmegaS64M1 offers V2.0B Active.

This full-CAN controller provides the whole hardware for convenient acceptance filtering and message management. For each message to be transmitted or received this module contains one so called message object in which all information regarding the message (for example identifier, data bytes etc.) are stored.

During the initialization of the peripheral, the application defines which messages are to be sent and which are to be received. Only if the CAN controller receives a message whose identifier matches with one of the identifiers of the programmed (receive-) message objects the message is stored and the application is informed by interrupt. Another advantage is that incoming remote frames can be answered automatically by the full-CAN controller with the corresponding data frame. In this way, the CPU load is strongly reduced compared to a basic-CAN solution.

Using full-CAN controller, high baud rates and high bus loads with many messages can be handled.

Figure 21-5. CAN Controller Structure



21.5 CAN channel

21.5.1 Configuration

The CAN channel can be in:

- Enabled mode

In this mode:

- the CAN channel (internal TxCAN & RxCAN) is enabled
- the input clock is enabled

- Standby mode

In standby mode:

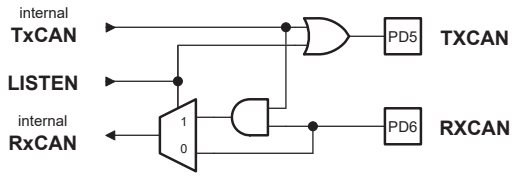
- the transmitter constantly provides a recessive level (on internal TxCAN) and the receiver is disabled
- input clock is enabled
- the registers and pages remain accessible

- Listening mode

This mode is transparent for the CAN channel:

- enables a hardware loop back, internal TxCAN on internal RxCAN
- provides a recessive level on TXCAN output pin
- does not disable RXCAN input pin
- freezes TEC and REC error counters

Figure 21-6. Listening mode.



21.5.2 Bit timing

FSM's (Finite State Machine) of the CAN channel need to be synchronous to the time quantum. So, the input clock for bit timing is the clock used into CAN channel FSM's.

Field and segment abbreviations:

- BRP: Baud Rate Prescaler
- TQ: Time Quantum (output of Baud Rate Prescaler)
- SYNS: SYNchronization Segment is 1 TQ long
- PRS: PRopagation time Segment is programmable to be 1, 2, ..., 8 TQ long
- PHS1: PHase Segment 1 is programmable to be 1, 2, ..., 8 TQ long
- PHS2: PHase Segment 2 is programmable to be ≤ PHS1 and ≥ INFORMATION PROCESSING TIME
- INFORMATION PROCESSING TIME is 2 TQ
- SJW: (Re) Synchronization Jump Width is programmable between 1 and min (4, PHS1)

The total number of TQ in a bit time has to be programmed at least from 8 to 25.

Figure 21-7. Sample and transmission point

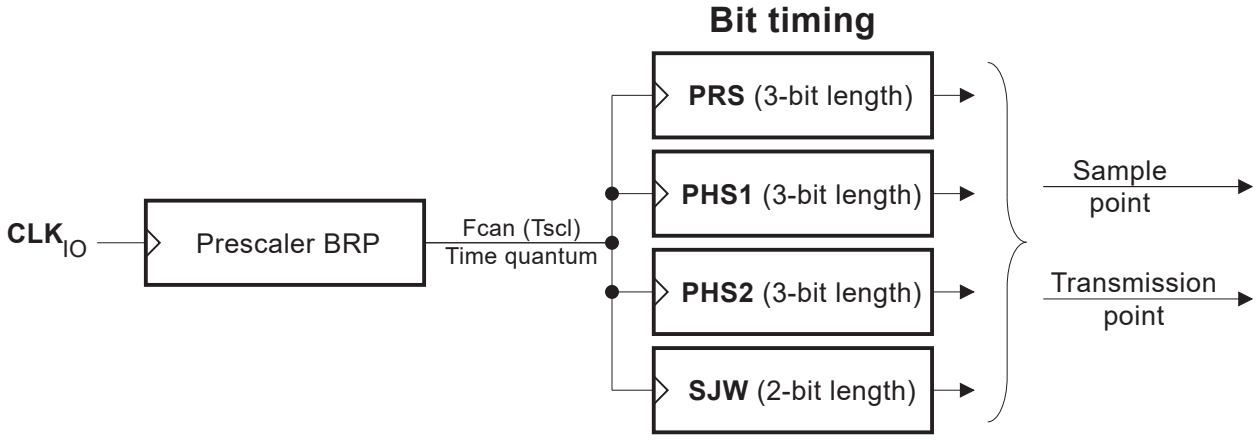
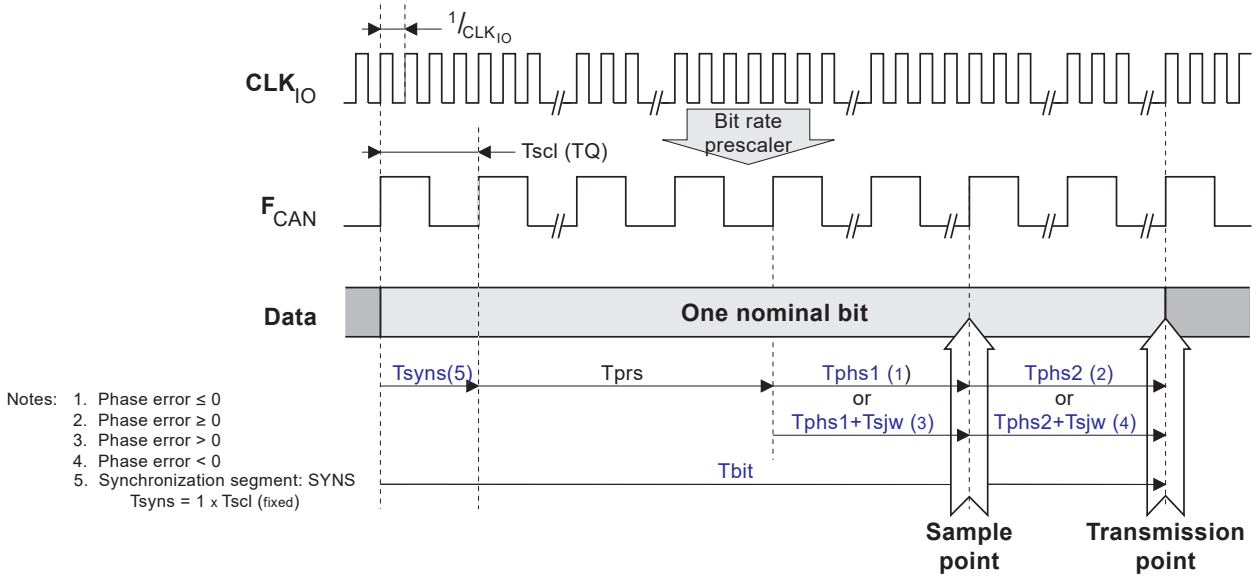


Figure 21-8. General Structure of a bit period



21.5.3 Baud rate

With no baud rate prescaler (BRP[5..0]=0) the sampling point comes one time quantum too early. This leads to a fail according the ISO16845 Test plan. It is necessary to lengthen the Phase Segment 1 by one time quantum and to shorten the Phase Segment 2 by one time quantum to compensate. The baud rate selection is made by T_{bit} calculation:

Tbit (1) = Tsyns + Tprs + Tphs1 + Tphs2

1. Tsyns = 1 × Tscl = (BRP[5..0] + 1)/clk_{IO} (= 1TQ)
2. Tprs = (1 to 8) × Tscl = (PRS[2..0] + 1) × Tscl
3. Tphs1 = (1 to 8) × Tscl = (PHS1[2..0] + 1) × Tscl
4. Tphs2 = (1 to 8) × Tscl = (PHS2[2..0] (2)+ 1) × Tscl
5. Tsjw = (1 to 4) × Tscl = (SJW[1..0] + 1) × Tscl

Notes: 1. The total number of Tscl (Time Quanta) in a bit time must be from 8 to 25.

2. PHS2[2..0] 2 is programmable to be ≤ PHS1[2..0] and ≥1.

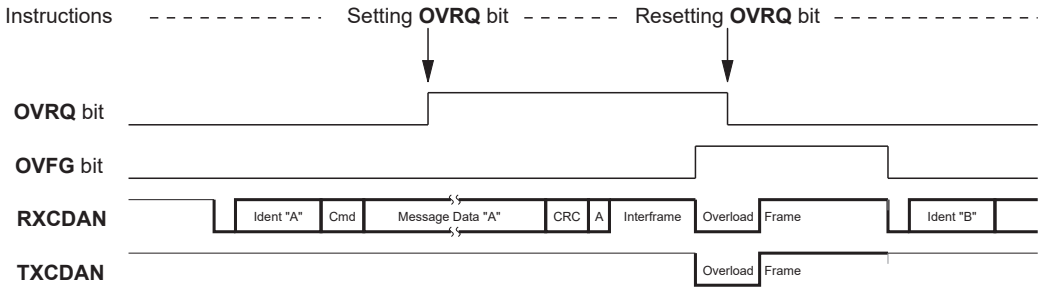
21.5.4 Fault confinement

See Error management.

21.5.5 Overload frame

An overload frame is sent by setting an overload request (OVRQ). After the next reception, the CAN channel sends an overload frame in accordance with the CAN specification. A status or flag is set (OVRF) as long as the overload frame is sent.

Figure 21-9. Overload frame.



21.6 Message objects

The MOb is a CAN frame descriptor. It contains all information to handle a CAN frame. This means that a MOb has been outlined to allow to describe a CAN message like an object. The set of MObs is the front end part of the “mailbox” where the messages to send and/or to receive are pre-defined as well as possible to decrease the work load of the software.

The MObs are independent but priority is given to the lower one in case of multi matching. The operating modes are:

- Disabled mode
- Transmit mode
- Receive mode
- Automatic reply
- Frame buffer receive mode

21.6.1 Number of MObs

This device has six MObs, they are numbered from 0 up to 5 (i = 5).

21.6.2 Operating modes

There is no default mode after RESET.
Every MOb has its own fields to control the operating mode. Before enabling the CAN peripheral, each MOb must be configured (ex: disabled mode - CONMOB=00).

Table 21-1. MOb configuration.

MOb configuration		Reply valid	RTR tag	Operating mode
0	0	x	x	Disabled
0	1	x	0	Tx data frame
		x	1	Tx remote frame

MOb configuration		Reply valid	RTR tag	Operating mode
1	0	x	0	Rx data frame
		0	1	Rx remote frame
		1		Rx remote frame then, Tx data frame (reply)
1	1	x	x	Frame Buffer Receive mode

21.6.2.1 Disabled

In this mode, the MOb is “free”.

21.6.2.2 Tx data and remote frame

1. Several fields must be initialized before sending:
   - Identifier tag (IDT)
   - Identifier extension (IDE)
   - Remote transmission request (RTRTAG)
   - Data length code (DLC)
   - Reserved bit(s) tag (RBnTAG)
   - Data bytes of message (MSG)
2. The MOb is ready to send a data or a remote frame when the MOb configuration is set (CONMOB).
3. Then, the CAN channel scans all the MObs in Tx configuration, finds the MOb having the highest priority and tries to send it.
4. When the transmission is completed the TXOK flag is set (interrupt).
5. All the parameters and data are available in the MOb until a new initialization.

21.6.2.3 Rx data and remote frame

1. Several fields must be initialized before receiving:
   - Identifier tag (IDT)
   - Identifier mask (IDMSK)
   - Identifier extension (IDE)
   - Identifier extension mask (IDEMSK)
   - Remote transmission request (RTRTAG)
   - Remote transmission request mask (RTRMSK)
   - Data length code (DLC)
   - Reserved bit(s) tag (RBnTAG)
2. The MOb is ready to receive a data or a remote frame when the MOb configuration is set (CONMOB).
3. When a frame identifier is received on CAN network, the CAN channel scans all the MObs in receive mode, tries to find the MOb having the highest priority which is matching.
4. On a hit, the IDT, the IDE and the DLC of the matched MOb are updated from the incoming (frame) values.
5. Once the reception is completed, the data bytes of the received message are stored (not for remote frame) in the data buffer of the matched MOb and the RXOK flag is set (interrupt).
6. All the parameters and data are available in the MOb until a new initialization.

21.6.2.4 Automatic reply

A reply (data frame) to a remote frame can be automatically sent after reception of the expected remote frame.

1. Several fields must be initialized before receiving the remote frame:
   – Reply valid (RPLV) in a identical flow to the one described inSection 19.6.2.3 “Rx data and remote frame” on page 167.
2. When a remote frame matches, automatically the RTRTAG and the reply valid bit (RPLV) are reset. No flag (or interrupt) is set at this time. Since the CAN data buffer has not been used by the incoming remote frame, the MOb is then ready to be in transmit mode without any more setting. The IDT, the IDE, the other tags and the DLC of the received remote frame are used for the reply.
3. When the transmission of the reply is completed the TXOK flag is set (interrupt).
4. All the parameters and data are available in the MOb until a new initialization.

21.6.2.5 Frame buffer receive mode

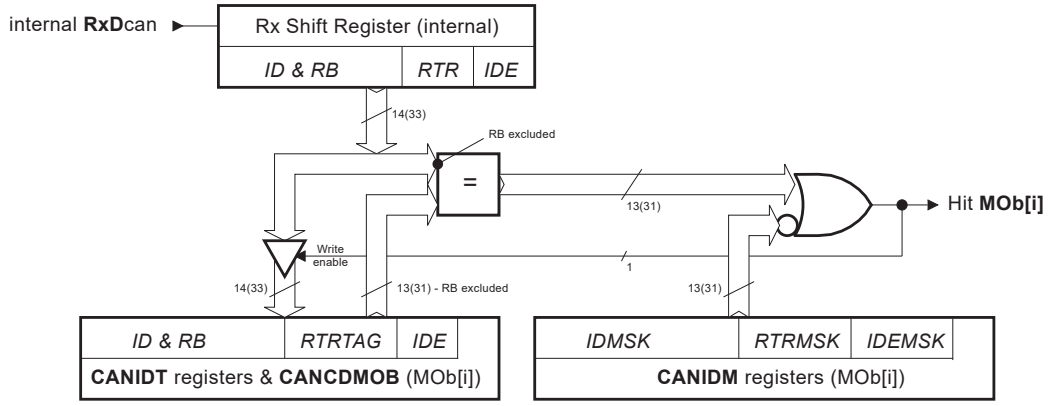
This mode is useful to receive multi frames. The priority between MObs offers a management for these incoming frames. One set MObs (including non-consecutive MObs) is created when the MObs are set in this mode. Due to the mode setting, only one set is possible. A frame buffer completed flag (or interrupt) - BXOK - will rise only when all the MObs of the set will have received their dedicated CAN frame.

1. MObs in frame buffer receive mode need to be initialized as MObs in standard receive mode.
2. The MObs are ready to receive data (or a remote) frames when their respective configurations are set (CONMOB).
3. When a frame identifier is received on CAN network, the CAN channel scans all the MObs in receive mode, tries to find the MOb having the highest priority which is matching.
4. On a hit, the IDT, the IDE and the DLC of the matched MOb are updated from the incoming (frame) values.
5. Once the reception is completed, the data bytes of the received message are stored (not for remote frame) in the data buffer of the matched MOb and the RXOK flag is set (interrupt).
6. When the reception in the last MOb of the set is completed, the frame buffer completed BXOK flag is set (interrupt). BXOK flag can be cleared only if all CONMOB fields of the set have been re-written before.
7. All the parameters and data are available in the MObs until a new initialization.

21.6.3 Acceptance filter

Upon a reception hit (that is, a good comparison between the ID + RTR + RBn + IDE received and an IDT + RTRTAG + RBnTAG + IDE specified while taking the comparison mask into account) the IDT + RTRTAG + RBnTAG + IDE received are updated in the MOb (written over the registers).

Figure 21-10. Acceptance filter block diagram.



Examples:

Full filtering: to accept only ID = 0x317 in part A.

- ID MSK = 111 1111 1111 $_b$
- ID TAG = 011 0001 0111 $_b$

Partial filtering: to accept ID from 0x310 up to 0x317 in part A.

- ID MSK = 111 1111 1000 $_b$
- ID TAG = 011 0001 0xxx $_b$

No filtering: to accept all ID's from 0x000 up to 0x7FF in part A.

- ID MSK = 000 0000 0000 $_b$
- ID TAG = xxx xxxx xxxx $_b$

21.6.4 MOb page

Every MOb is mapped into a page to save place. The page number is the MOb number. This page number is set in CANPAGE register. The other numbers are reserved for factory tests.

CANHPMOB register gives the MOb having the highest priority in CANSIT registers. It is formatted to provide a direct entry for CANPAGE register. Because CANHPMOB codes CANSIT registers, it will be only updated if the corresponding enable bits (ENRX, ENTX, ENERR) are enabled. See Figure 21-14

21.6.5 CAN data buffers

To preserve register allocation, the CAN data buffer is seen such as a FIFO (with address pointer accessible) into a MOb selection.This also allows to reduce the risks of un-controlled accesses.

There is one FIFO per MOb. This FIFO is accessed into a MOb page thanks to the CAN message register.

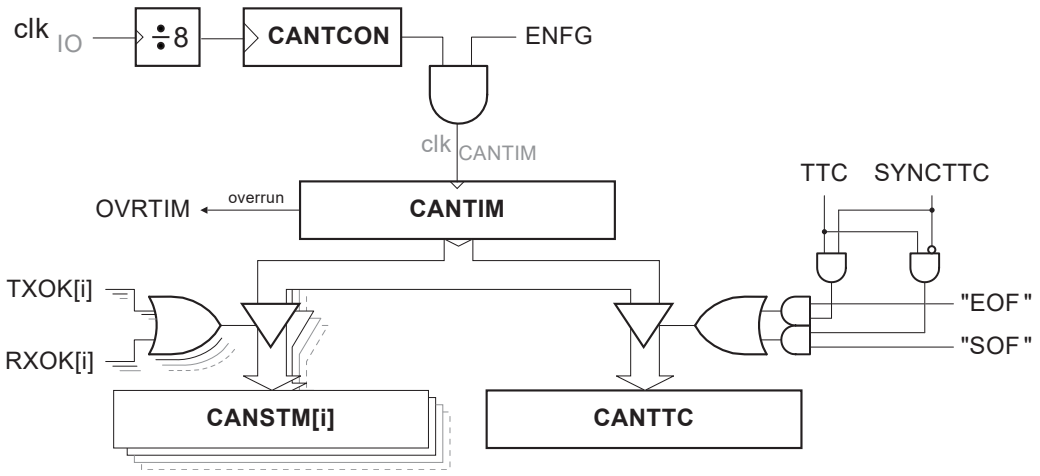
The data index (INDX) is the address pointer to the required data byte. The data byte can be read or write. The data index is automatically incremented after every access if the AINC* bit is reset. A roll-over is implemented, after data index=7 it is data index=0.

The first byte of a CAN frame is stored at the data index=0, the second one at the data index=1, ...

21.7 CAN timer

A programmable 16-bit timer is used for message stamping and time trigger communication (TTC).

Figure 21-11. CAN timer block diagram.



21.7.1 Prescaler

An 8-bit prescaler is initialized by CANTCON register. It receives the clk_{IO} frequency divided by 8. It provides clk_{CANTIM} frequency to the CAN Timer if the CAN controller is enabled.

$Tclk_{CANTIM} = Tclk_{IO} \times 8 \times (CANTCON\ [7:0] + 1)$

21.7.2 16-bit timer

This timer starts counting from 0x0000 when the CAN controller is enabled (ENFG bit). When the timer rolls over from 0xFFFF to 0x0000, an interrupt is generated (OVRTIM).

21.7.3 Time triggering

Two synchronization modes are implemented for TTC (TTC bit):

- synchronization on Start of Frame (SYNCTTC=0)
- synchronization on End of Frame (SYNCTTC=1)

In TTC mode, a frame is sent once, even if an error occurs.

21.7.4 Stamping message

The capture of the timer value is done in the MOb which receives or sends the frame. All managed MOb are stamped, the stamping of a received (sent) frame occurs on RxOk (TXOK).

21.8 Error management

21.8.1 Fault confinement

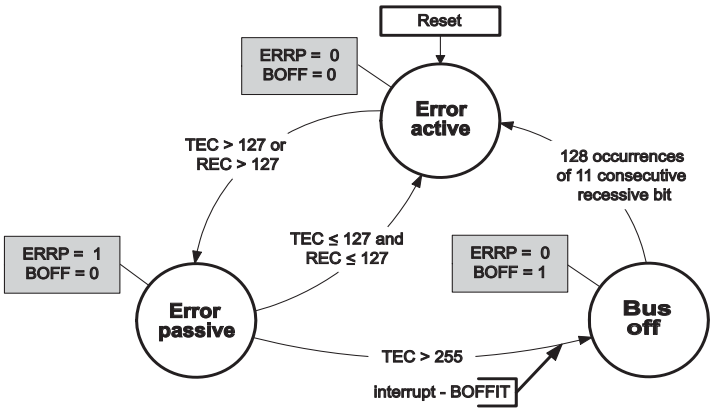
The CAN channel may be in one of the three following states:

- Error active (default):
  The CAN channel takes part in bus communication and can send an active error frame when the CAN macro detects an error

- Error passive:
  The CAN channel cannot send an active error frame. It takes part in bus communication, but when an error is detected, a passive error frame is sent. Also, after a transmission, an error passive unit will wait before initiating further transmission
- Bus off:
  The CAN channel is not allowed to have any influence on the bus

For fault confinement, a transmit error counter (TEC) and a receive error counter (REC) are implemented. BOFF and ERRP bits give the information of the state of the CAN channel. Setting BOFF to one may generate an interrupt.

Figure 21-12. Line Error mode.

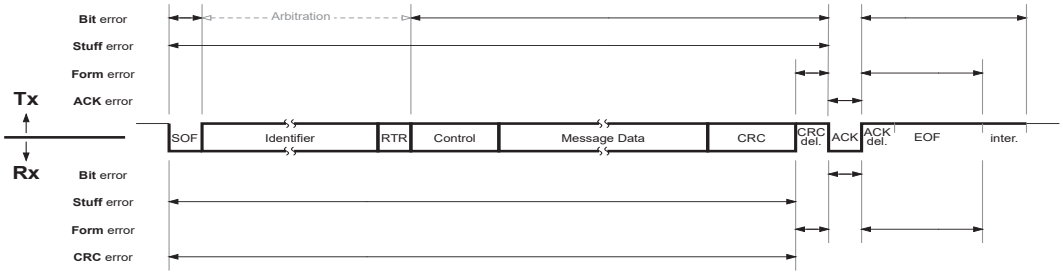


Note: More than one REC/TEC change may apply during a given message transfer.

21.8.2 Error types

- BERR: Bit error. The bit value which is monitored is different from the bit value sent
  Note: Exceptions:
  - Recessive bit sent monitored as dominant bit during the arbitration field and the acknowledge slot
  - Detecting a dominant bit during the sending of an error frame
- SERR: Stuff error. Detection of more than five consecutive bit with the same polarity
- CERR: CRC error (Rx only). The receiver performs a CRC check on every destuffed received message from the start of frame up to the data field. If this checking does not match with the destuffed CRC field, an CRC error is set
- FERR: Form error. The form error results from one (or more) violations of the fixed form of the following bit fields:
  - CRC delimiter
  - acknowledgment delimiter
  - end-of-frame
  - error delimiter
  - overload delimiter
- AERR: Acknowledgment error (Tx only). No detection of the dominant bit in the acknowledge slot

Figure 21-13. Error detection procedures in a data frame.



21.8.3 Error setting

The CAN channel can detect some errors on the CAN network.

- In transmission:
  The error is set at MOb level
- In reception:
  - The identified has matched:
  The error is set at MOb level
  - The identified has not or not yet matched:
  The error is set at general level

After detecting an error, the CAN channel sends an error frame on network. If the CAN channel detects an error frame on network, it sends its own error frame.

21.9 Interrupts

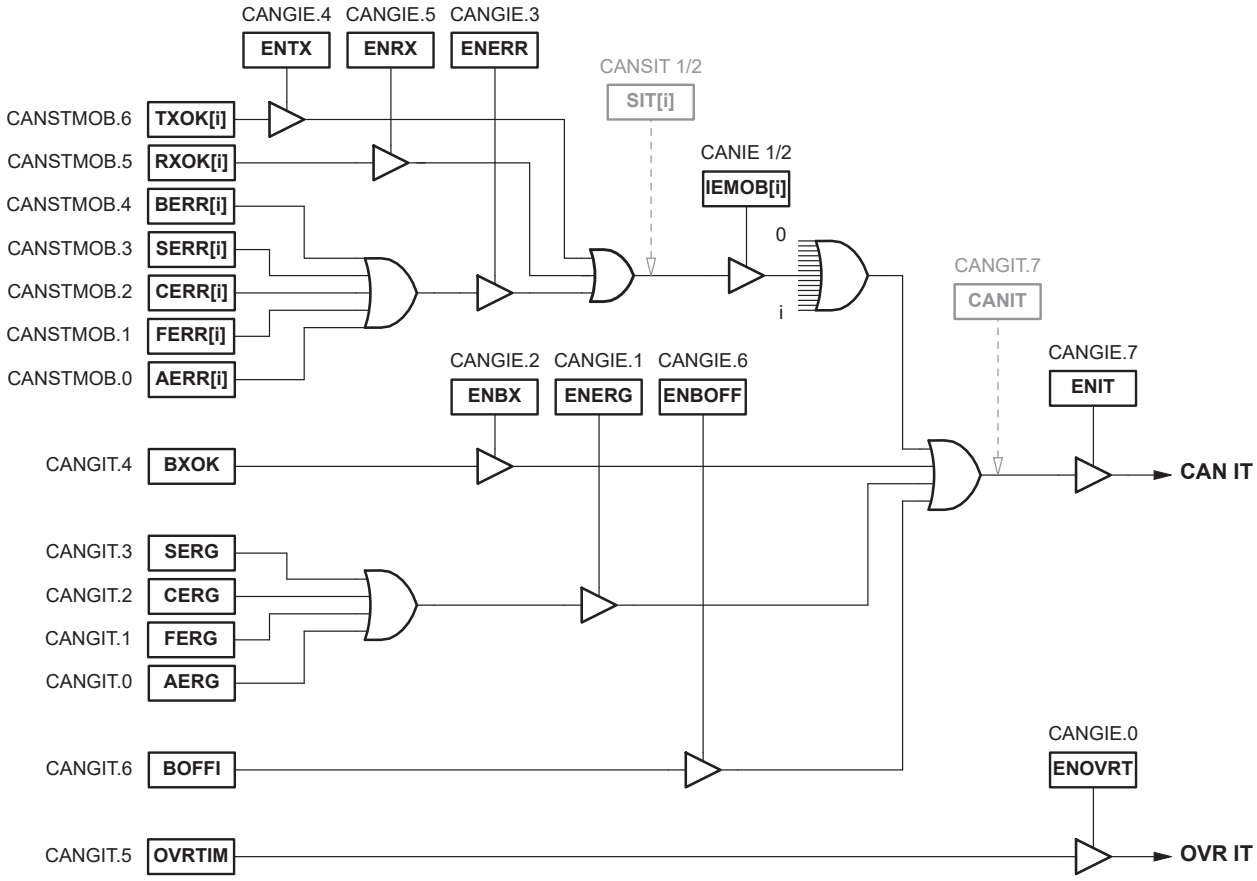
21.9.1 Interrupt organization

The different interrupts are:

- Interrupt on receive completed OK
- Interrupt on transmit completed OK
- Interrupt on error (bit error, stuff error, CRC error, form error, acknowledge error)
- Interrupt on frame buffer full
- Interrupt on “Bus Off” setting
- Interrupt on overrun of CAN timer

The general interrupt enable is provided by ENIT bit and the specific interrupt enable for CAN timer overrun is provided by ENORVT bit.

Figure 21-14. CAN controller interrupt structure



21.9.2 Interrupt behavior

When an interrupt occurs, an interrupt flag bit is set in the corresponding MOb-CANSTMOB register or in the general CANGIT register. If in the CANIE register, ENRX / ENTX / ENERR bit are set, then the corresponding MOb bit is set in the CANSITn register.

To acknowledge a MOb interrupt, the corresponding bits of CANSTMOB register (RXOK, TXOK,...) must be cleared by the software application. This operation needs a read-modify-write software routine.

To acknowledge a general interrupt, the corresponding bits of CANGIT register (BXOK, BOFFIT,...) must be cleared by the software application. This operation is made writing a logical one in these interrupt flags (writing a logical zero doesn't change the interrupt flag value).

OVRTIM interrupt flag is reset as the other interrupt sources of CANGIT register and is also reset entering in its dedicated interrupt handler.

When the CAN node is in transmission and detects a Form Error in its frame, a bit Error will also be raised. Consequently, two consecutive interrupts can occur, both due to the same error.

When a MOb error occurs and is set in its own CANSTMOB register, no general error is set in CANGIT register.

21.10 Examples of CAN Baud Rate Setting

The CAN bus requires very accurate timing especially for high baud rates. It is recommended to use only an external crystal for CAN operations.

See *Related Links* below for more information.

Table 21-2. Examples of CAN Baud Rate Settings for Common Frequencies

$fCLK_{IO}$ [MHz]	CAN Rate [Kbps]	Description			Segments				Registers		
		Sampling Point	TQ [s]	Tbit [TQ]	Tprs [TQ]	Tph1 [TQ]	Tph2 [TQ]	Tsjw [TQ]	CANBT1	CANBT2	CANBT3
8.000	1000	63% (1)		x	---nodata---						
			0.125	8	3	2	2	1	0x00	0x04	0x12 (2)
	500	69% (1)	0.125	16	7	4	4	1	0x00	0x0C	0x36 (2)
		75%	0.250	8	3	2	2	1	0x02	0x04	0x13
	250	75%	0.250	16	7	4	4	1	0x02	0x0C	0x37
			0.500	8	3	2	2	1	0x06	0x04	0x13
	200	75%	0.250	20	8	6	5	1	0x02	0x0E	0x4B
			0.625	8	3	2	2	1	0x08	0x04	0x13
	125	75%	0.500	16	7	4	4	1	0x06	0x0C	0x37
			1.000	8	3	2	2	1	0x0E	0x04	0x13
	100	75%	0.625	16	7	4	4	1	0x08	0x0C	0x37
			1.250	8	3	2	2	1	0x12	0x04	0x13
6.000	1000	---notapplicable---									
	500	67% (1)	0.166666	12	5	3	3	1	0x00	0x08	0x24 (2)
				x	---nodata---						
	250	75%	0.333333	12	5	3	3	1	0x02	0x08	0x25
			0.500	8	3	2	2	1	0x04	0x04	0x13
	200	80%	0.333333	15	7	4	3	1	0x02	0x0C	0x35
			0.500	10	4	3	2	1	0x04	0x06	0x23
	125	75%	0.500	16	7	4	4	1	0x04	0x0C	0x37
			1.000	8	3	2	2	1	0x0A	0x04	0x13
	100	75%	0.500	20	8	6	5	1	0x04	0x0E	0x4B
			0.833333	12	5	3	3	1	0x08	0x08	0x25
4.000	1000	---notapplicable---									
	500	63% (1)		x	---nodata---						
			0.250	8	3	2	2	1	0x00	0x04	0x12 (2)
	250	69% (1)	0.250	16	7	4	4	1	0x00	0x0C	0x36 (2)
		75%	0.500	8	3	2	2	1	0x02	0x04	0x13
	200	70% (1)	0.250	20	8	6	5	1	0x00	0x0E	0x4A (2)
				x	---nodata---						
	125	75%	0.500	16	7	4	4	1	0x02	0x0C	0x37
			1.000	8	3	2	2	1	0x06	0x04	0x13
	100	75%	0.500	20	8	6	5	1	0x02	0x0E	0x4B
			1.250	8	3	2	2	1	0x08	0x04	0x13

Note:

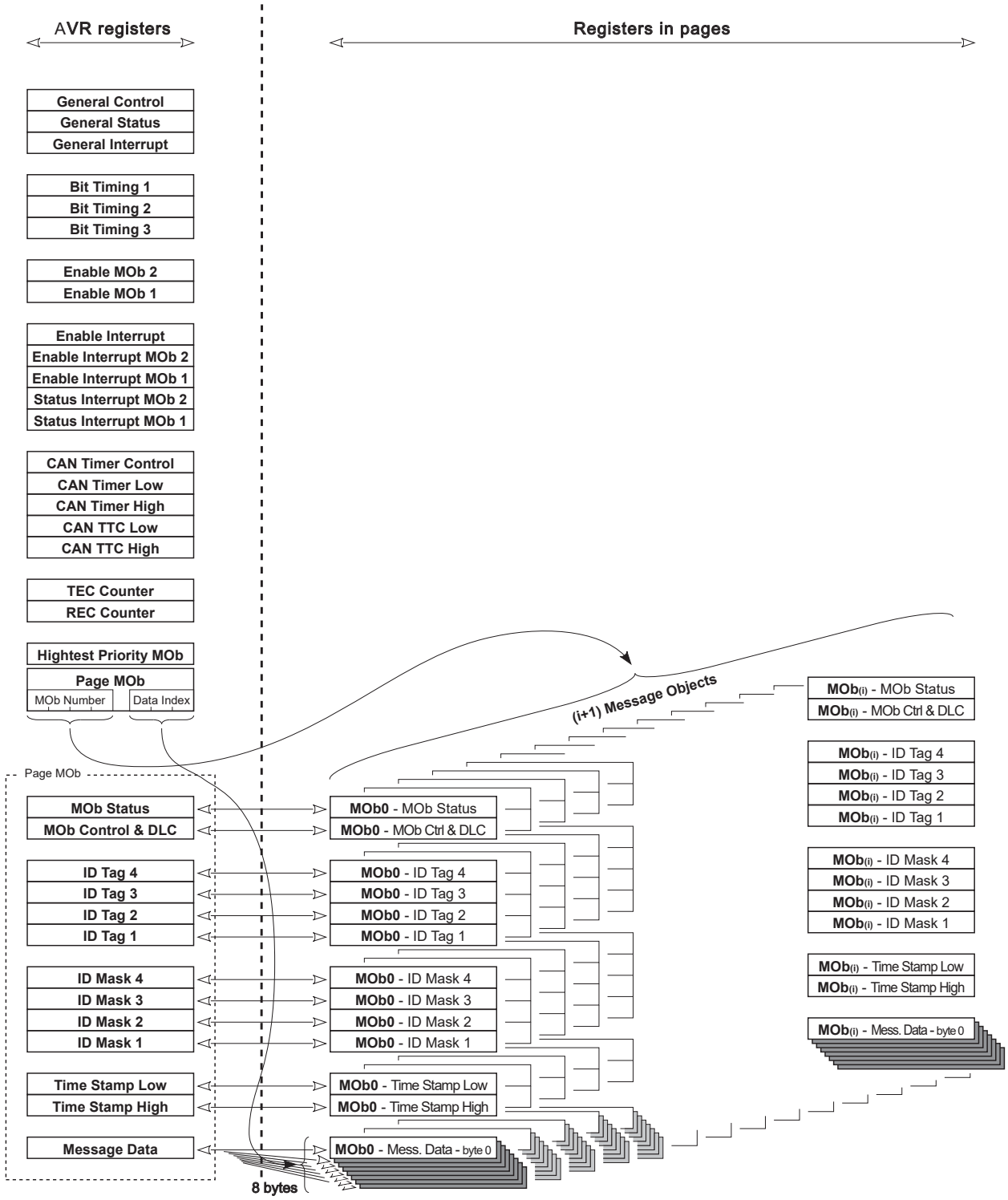
1. See section *Baud rate* for more details.
2. Refer to the description of bit SMP: Sample Point(s) in *CAN Bit Timing Register 3 (CANBT3)*.

Related Links

Bit timing
Baud rate
CANBT1
CANBT2
CANBT3

21.11 Register Description

Figure 21-15. Registers organization



21.11.1 CAN General Control Register

Name: CANGCON
Offset: 0xD8
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	ABRQ	OVRQ	TTC	SYNTTC	LISTEN	TEST	ENA/STB	SWRES
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – ABRQ Abort Request
This is not an auto resettable bit.

Value	Description
0	No request
1	Abort request: a reset of CANEN1 and CANEN2 registers is done. The pending communications are immediately disabled and the on-going one will be normally terminated, setting the appropriate status flags. **Note:** The CANCDMOB register remains unchanged

Bit 6 – OVRQ Overload Frame Request
This is not an auto resettable bit. The overload frame can be traced observing OVFG in CANGSTA register

Value	Description
0	No request
1	Overload frame request: send an overload frame after the next received frame

Bit 5 – TTC Time Trigger Communication

Value	Description
0	No TTC
1	TTC mode

Bit 4 – SYNTTC Synchronization of TTC
This bit is only used in TTC mode.

Value	Description
0	The TTC timer is caught on SOF
1	The TTC timer is caught on the last bit of the EOF

Bit 3 – LISTEN Listening Mode

Value	Description
0	No listening mode
1	Listening mode

Bit 2 – TEST Test Mode
Note: CAN may malfunction if this bit is set.

Value	Description
0	No test mode
1	Test mode: intend for factory testing and not for customer use

Bit 1 – ENA/$\overline{\text{STB}}$ Enable / Standby Mode
Because this bit is a command and is not immediately effective, the ENFG bit in CANGSTA register gives the true state of the chosen mode.

Value	Description
0	Standby mode: The on-going transmission (if exists) is normally terminated and the CAN channel is frozen (the CONMOB bits of every MOb do not change). The transmitter constantly provides a recessive level. In this mode, the receiver is not enabled but all the registers and mailbox remain accessible from CPU. In this mode, the receiver is not enabled but all the registers and mailbox remain accessible from CPU **Note:** A standby mode applied during a reception may corrupt the on-going reception or set the controller in a wrong state. The controller will restart correctly from this state if a software reset (SWRES) is applied. If no reset is considered, a possible solution is to wait for a lake of a receiver busy (RXBSY) before to enter in stand-by mode. The best solution is first to apply an abort request command (ABRQ) and then wait for the lake of the receiver busy (RXBSY) before to enter in stand-by mode. In any cases, this standby mode behavior has no effect on the CAN bus integrity.
1	Enable mode: The CAN channel enters in enable mode once 11 recessive bits has been read

Bit 0 – SWRES Software Reset Request
This auto resettable bit only resets the CAN controller.

Value	Description
0	No reset
1	Reset: this reset is “ORed” with the hardware reset

21.11.2 CAN General Status Register

Name: CANGSTA
Offset: 0xD9
Reset: 0x0
Property: R

Bit	7	6	5	4	3	2	1	0
		OVRG		TXBSY	RXBSY	ENFG	BOFF	ERRP
Access								
Reset		0		0	0	0	0	0

Bit 6 – OVRG Overload Frame Flag
This flag does not generate an interrupt.

Value	Description
0	No overload frame
1	Overload frame: set by hardware as long as the produced overload frame is sent

Bit 4 – TXBSY Transmitter Busy
This flag does not generate an interrupt.

Value	Description
0	Transmitter not busy
1	Transmitter busy: set by hardware as long as a frame (data, remote, overload or error frame) or an ACK field is sent. Also set when an inter frame space is sent

Bit 3 – RXBSY Receiver Busy
This flag does not generate an interrupt.

Value	Description
0	Receiver not busy
1	Receiver busy: set by hardware as long as a frame is received or monitored

Bit 2 – ENFG Enable Flag
This flag does not generate an interrupt.

Value	Description
0	CAN controller disable: because an enable/$\overline{\text{standby}}$ command is not immediately effective, this status gives the true state of the chosen mode
1	CAN controller enable

Bit 1 – BOFF Bus Off Mode
BOFF gives the information of the state of the CAN channel. Only entering in bus off mode generates the BOFFIT interrupt.

Value	Description
0	No bus off mode
1	Bus off mode

Bit 0 – ERRP Error Passive Mode
ERRP gives the information of the state of the CAN channel. This flag does not generate an interrupt.

Value	Description
0	No error passive mode
1	Error passive mode

21.11.3 CAN General Interrupt Register

Name: CANGIT
Offset: 0xDA
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	CANIT	BOFFIT	OVRTIM	BXOK	SERG	CERG	FERG	AERG
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – CANIT General Interrupt Flag
This is a read only bit.

Value	Description
0	No interrupt
1	CAN interrupt: image of all the CAN controller interrupts except for OVRTIM interrupt. This bit can be used for polling method

Bit 6 – BOFFIT Bus Off Interrupt Flag
Writing a logical one resets this interrupt flag. BOFFIT flag is only set when the CAN enters in bus off mode (coming from error passive mode).

Value	Description
0	No interrupt
1	Bus off interrupt when the CAN enters in bus off mode

Bit 5 – OVRTIM Overrun CAN Timer
Writing a logical one resets this interrupt flag. Entering in CAN timer overrun interrupt handler also reset this interrupt flag

Value	Description
0	No interrupt
1	CAN timer overrun interrupt: set when the CAN timer switches from 0xFFFF to 0

Bit 4 – BXOK Frame Buffer Receive Interrupt
Writing a logical one resets this interrupt flag. BXOK flag can be cleared only if all CONMOB fields of the MOb's of the buffer have been re-written before.

Value	Description
0	No interrupt
1	Burst receive interrupt: set when the frame buffer receive is completed

Bit 3 – SERG Stuff Error General
Writing a logical one resets this interrupt flag.

Value	Description
0	No interrupt
1	Stuff error interrupt: detection of more than 5 consecutive bits with the same polarity

Bit 2 – CERG CRC Error General
Writing a logical one resets this interrupt flag.

Value	Description
0	No interrupt
1	CRC error interrupt: the CRC check on destuffed message does not fit with the CRC field

Bit 1 – FERG Form Error General
Writing a logical one resets this interrupt flag.

Value	Description
0	No interrupt
1	Form error interrupt: one or more violations of the fixed form in the CRC delimiter, acknowledgment delimiter or EOF

Bit 0 – AERG Acknowledgment Error General
Writing a logical one resets this interrupt flag.

Value	Description
0	No interrupt
1	acknowledgment error interrupt: no detection of the dominant bit in acknowledge slot

21.11.4 CAN General Interrupt Enable Register

Name: CANGIE
Offset: 0xDB
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	ENIT	ENBOFF	ENRX	ENTX	ENERR	ENBX	ENERG	ENOVRT
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – ENIT Enable all Interrupts
Except for CAN Timer Overrun Interrupt

Value	Description
0	Interrupt disabled
1	CANIT interrupt enabled

Bit 6 – ENBOFF Enable Bus Off Interrupt

Value	Description
0	Interrupt disabled
1	Bus off interrupt enabled

Bit 5 – ENRX Enable Receive Interrupt

Value	Description
0	Interrupt disabled
1	Receive interrupt enabled

Bit 4 – ENTX Enable Transmit Interrupt

Value	Description
0	Interrupt disabled
1	Transmit interrupt enabled

Bit 3 – ENERR Enable MOb Errors Interrupt

Value	Description
0	Interrupt disabled
1	MOb errors interrupt enabled

Bit 2 – ENBX Enable Frame Buffer Interrupt

Value	Description
0	Interrupt disabled
1	Frame buffer interrupt enabled

Bit 1 – ENERG Enable General Errors Interrupt

Value	Description
0	Interrupt disabled
1	General errors interrupt enabled

Bit 0 – ENOVRT Enable CAN Timer Overrun Interrupt

Value	Description
0	Interrupt disabled
1	CAN timer interrupt overrun enabled

21.11.5 CAN Enable MOb Registers

Name: CANEN
Offset: 0xDC
Reset: 0x0
Property: R

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			ENMOB5	ENMOB4	ENMOB3	ENMOB2	ENMOB1	ENMOB0
Access								
Reset			0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5 – ENMOB Enable MOb
This bit provides the availability of the MOb. It is set to one when the MOb is enabled (that is, CONMOB1:0 of CANCDMOB register). Once TXOK or RXOK is set to one (TXOK for automatic reply), the corresponding ENMOB is reset. ENMOB is also set to zero configuring the MOb in disabled mode, applying abortion or standby mode.

Value	Description
0	Message object disabled: MOb available for a new transmission or reception
1	Message object enabled: MOb in use

21.11.6 CAN Enable Interrupt MOb Registers

Name: CANIE
Offset: 0xDE
Reset: 0x0
Property: R/W

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			IEMOB5	IEMOB4	IEMOB3	IEMOB2	IEMOB1	IEMOB0
Access								
Reset			0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5 – IEMOB Enable MOb
Example: CANIE2 = 0000 1100b : enable of interrupts on MOb 2 and 3

Value	Description
0	Interrupt disabled
1	MOb interrupt enabled

21.11.7 CAN Status Interrupt MOb Registers

Name: CANSIT
Offset: 0xE0
Reset: 0x0
Property: R

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			SIT5	SIT4	SIT3	SIT2	SIT1	SIT0
Access								
Reset			0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5 – SIT Enable MOb
Example: CANSIT2 = 0010 0001b : MOb 0 and 5 interrupts

Value	Description
0	No interrupt
1	MOb interrupt

21.11.8 CAN Bit Timing Register 1

Name: CANBT1
Offset: 0xE2
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
		BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	
Access								
Reset		0	0	0	0	0	0	

Bits 1, 2, 3, 4, 5, 6 – BRP Baud Rate Prescaler
The period of the CAN controller system clock Tscl is programmable and determines the individual bit timing.

$$T_{\mathrm{SCL}} = \frac{\mathrm{BRP}[5:0]+1}{\mathrm{clk_{IO}frequency}}$$

21.11.9 CAN Bit Timing Register 2

Name: CANBT2
Offset: 0xE3
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
		SJW1	SJW0		PRS2	PRS1	PRS0	
Access								
Reset		0	0		0	0	0	

Bits 5, 6 – SJW Re-Synchronization Jump Width
To compensate for phase shifts between clock oscillators of different bus controllers, the controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum number of clock cycles. A bit period may be shortened or lengthened by a re-synchronization.

$$T_{\mathrm{sjw}} = T_{\mathrm{scl}} \times (\mathrm{SJW}[1{:}0]+1)$$

Bits 1, 2, 3 – PRS Propagation Time Segment
This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal propagation time on the bus line, the input comparator delay and the output driver delay.

$$T_{\mathrm{prs}} = T_{\mathrm{scl}} \times (\mathrm{PRS}[2{:}0]+1)$$

21.11.10 CAN Bit Timing Register 3

Name: CANBT3
Offset: 0xE4
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
		PHS22	PHS21	PHS20	PHS12	PHS11	PHS10	SMP
Access								
Reset		0	0	0	0	0	0	0

Bits 4, 5, 6 – PHS2 Phase Segment 2
This phase is used to compensate for phase edge errors. This segment may be shortened by the re-synchronization jump width. PHS2[2:0] shall be ≥1 and ≤PHS1[2..0]

$$T_{\mathrm{phs2}} = T_{\mathrm{scl}} \times (\mathrm{PHS2}[2{:}0]+1)$$

Bits 1, 2, 3 – PHS1 Phase Segment 1
This phase is used to compensate for phase edge errors. This segment may be lengthened by the re-synchronization jump width.

$$T_{\mathrm{phs1}} = T_{\mathrm{scl}} \times (\mathrm{PHS1}[2{:}0]+1)$$

Bit 0 – SMP Sample Point(s)
This option allows to filter possible noise on TxCAN input pin.

'SMP=1' configuration is not compatible with 'BRP[5:0]=0' because TQ = $Tclk_{IO}$. If BRP = 0, SMP must be cleared.

Value	Description
0	The sampling will occur once at the user configured sampling point - SP
1	With three-point sampling configuration the first sampling will occur two $Tclk_{IO}$ clocks before the user configured sampling point - SP, again at one $Tclk_{IO}$ clock before SP and finally at SP. Then the bit level will be determined by a majority vote of the three samples

21.11.11 CAN Timer Control Register

Name: CANTCON
Offset: 0xE5
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	TPRSC7	TPRSC6	TPRSC5	TPRSC4	TPRSC3	TPRSC2	TPRSC1	TPRSC0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – TPRSC CAN Timer Prescaler
Prescaler for the CAN timer upper counter range 0 to 255. It provides the clock to the CAN timer if the CAN controller is enabled.

$Tclk_{CANTIM} = Tclk_{IO} \times 8 \times (CANTCON[7:0]+1)$

21.11.12 CAN Timer Registers

Name: CANTIM
Offset: 0xE6
Reset: 0x0
Property: R

Bit	15	14	13	12	11	10	9	8
	CANTIM15	CANTIM14	CANTIM13	CANTIM12	CANTIM11	CANTIM10	CANTIM9	CANTIM8
Access								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CANTIM7	CANTIM6	CANTIM5	CANTIM4	CANTIM3	CANTIM2	CANTIM1	CANTIM0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 – CANTIM CAN Timer Count
CAN timer counter range 0 to 65,535.

21.11.13 CAN Timer Registers

Name: CANTTC
Offset: 0xE8
Reset: 0x0
Property: R

Bit	15	14	13	12	11	10	9	8
	TIMTTC15	TIMTTC14	TIMTTC13	TIMTTC12	TIMTTC11	TIMTTC10	TIMTTC9	TIMTTC8
Access								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TIMTTC7	TIMTTC6	TIMTTC5	TIMTTC4	TIMTTC3	TIMTTC2	TIMTTC1	TIMTTC0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 – TIMTTC TTC Timer Count
CAN TTC timer counter range 0 to 65,535.

21.11.14 CAN Transmit Error Counter Register

Name: CANTEC
Offset: 0xEA
Reset: 0x0
Property: R

Bit	7	6	5	4	3	2	1	0
	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – TEC Transmit Error Count
CAN transmit error counter range 0 to 255.

21.11.15 CAN Receive Error Counter Register

Name: CANREC
Offset: 0xEB
Reset: 0x0
Property: R

Bit	7	6	5	4	3	2	1	0
	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – REC Receive Error Count
CAN receive error counter range 0 to 255.

21.11.16 CAN Highest Priority MOb Register

Name: CANHPMOB
Offset: 0xEC
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	HPMOB3	HPMOB2	HPMOB1	HPMOB0	CGP3	CGP2	CGP1	CGP0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 4, 5, 6, 7 – HPMOB Highest Priority MOb Number
MOb having the highest priority in CANSIT registers.

If CANSIT = 0 (no MOb), the return value is 0xF.

Note: Do not confuse “MOb priority” and “Message ID priority”

Bits 0, 1, 2, 3 – CGP CAN General Purpose Bits
These bits can be pre-programmed to match with the wanted configuration of the CANPAGE register (that is, $\overline{\text{AINC}}$ and INDX2:0 setting).

21.11.17 CAN Page MOb Register

Name: CANPAGE
Offset: 0xED
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	MOBNB3	MOBNB2	MOBNB1	MOBNB0	$\overline{\text{AINC}}$	INDX2	INDX1	INDX0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 4, 5, 6, 7 – MOBNB MOb Number
Selection of the MOb number, the available numbers are from 0 to 5.

Note: MOBNB3 always must be written to zero for compatibility with all AVR CAN devices

Bit 3 – $\overline{\text{AINC}}$ Auto Increment of the FIFO CAN Data Buffer Index (Active Low)

Value	Description
0	Auto increment of the index (default value)
1	No auto increment of the index

Bits 0, 1, 2 – INDX FIFO CAN Data Buffer Index
Byte location of the CAN data byte into the FIFO for the defined MOb.

21.11.18 CAN MOb Status Register

Name: CANSTMOB
Offset: 0xEE
Property: R/W

Bit	7	6	5	4	3	2	1	0
	DLCW	TXOK	RXOK	BERR	SERR	CERR	FERR	AERR
Access								
Reset								

Bit 7 – DLCW Data Length Code Warning
The incoming message does not have the DLC expected. Whatever the frame type, the DLC field of the CANCDMOB register is updated by the received DLC.

Bit 6 – TXOK Transmit OK
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The communication enabled by transmission is completed. TxOK rises at the end of EOF field. When the controller is ready to send a frame, if two or more message objects are enabled as producers, the lower MOb index (0 to 14) is supplied first.

Bit 5 – RXOK Receive OK
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The communication enabled by reception is completed. RxOK rises at the end of the 6th bit of EOF field. In case of two or more message object reception hits, the lower MOb index (0 to 14) is updated first.

Bit 4 – BERR Bit Error (Only in Transmission)
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The bit value monitored is different from the bit value sent.

Exceptions: the monitored recessive bit sent as a dominant bit during the arbitration field and the acknowledge slot detecting a dominant bit during the sending of an error frame.

Bit 3 – SERR Stuff Error
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

Detection of more than five consecutive bits with the same polarity. This flag can generate an interrupt.

Bit 2 – CERR CRC Error
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The receiver performs a CRC check on every de-stuffed received message from the start of frame up to the data field. If this checking does not match with the de-stuffed CRC field, a CRC error is set.

Bit 1 – FERR Form Error
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The form error results from one or more violations of the fixed form in the following bit fields:

- CRC delimiter
- Acknowledgment delimiter
- EOF

Bit 0 – AERR Acknowledgment Error
This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

No detection of the dominant bit in the acknowledge slot.

21.11.19 CAN MOb Control and DLC Register

Name: CANCDMOB
Offset: 0xEF
Property: R/W

Bit	7	6	5	4	3	2	1	0
	CONMOB1	CONMOB0	RPLV	IDE	DLC3	DLC2	DLC1	DLC0
Access								
Reset								

Bits 6, 7 – CONMOB Configuration of Message Object
These bits set the communication to be performed (no initial value after RESET).

These bits are not cleared once the communication is performed. The user must re-write the configuration to enable a new communication.

- This operation is necessary to be able to reset the BXOK flag
- This operation also set the corresponding bit in the CANEN registers

Value	Description
00	Disable
01	Enable transmission
10	Enable reception
11	Enable frame buffer reception

Bit 5 – RPLV Reply Valid
Used in the automatic reply mode after receiving a remote frame.

Value	Description
0	Reply not ready
1	Reply ready and valid

Bit 4 – IDE Identifier Extension
IDE bit of the remote or data frame to send. This bit is updated with the corresponding value of the remote or data frame received.

Value	Description
0	CAN standard rev 2.0 A (identifiers length = 11 bits)
1	CAN standard rev 2.0 B (identifiers length = 29 bits)

Bits 0, 1, 2, 3 – DLC Data Length Code
Number of Bytes in the data field of the message.

DLC field of the remote or data frame to send. The range of DLC is from 0 up to 8. If DLC field >8 then effective DLC=8.

This field is updated with the corresponding value of the remote or data frame received. If the expected DLC differs from the incoming DLC, a DLC warning appears in the CANSTMOB register.

21.11.20 CAN Identifier Tag Registers

Name: CANIDT
Offset: 0xF0
Property: R/W

V2.0 part A

Bit	31	30	29	28	27	26	25	24
	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5	IDT4	IDT3
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IDT2	IDT1	IDT0					
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						RTRTAG		RB0TAG
Access								
Reset								

Bits 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – IDT Identifier Tag
Identifier field of the remote or data frame to send. This field is updated with the corresponding value of the remote or data frame received.

Bit 2 – RTRTAG Remote Transmission Request Tag
RTR bit of the remote or data frame to send. This tag is updated with the corresponding value of the remote or data frame received. In case of Automatic Reply mode, this bit is automatically reset before sending the response.

Bit 0 – RB0TAG Reserved Bit 0 Tag
RB0 bit of the remote or data frame to send. This tag is updated with the corresponding value of the remote or data frame received.

21.11.21 CAN Identifier Tag Registers

Name: CANIDT
Offset: 0xF0
Property: R/W

V2.0 part B

Bit	31	30	29	28	27	26	25	24
	IDT28	IDT27	IDT26	IDT25	IDT24	IDT23	IDT22	IDT21
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IDT20	IDT19	IDT18	IDT17	IDT16	IDT15	IDT14	IDT13
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	IDT12	IDT11	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	IDT4	IDT3	IDT2	IDT1	IDT0	RTRTAG	RB1TAG	RB0TAG
Access								
Reset								

Bits 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – IDT Identifier Tag
Identifier field of the remote or data frame to send. This field is updated with the corresponding value of the remote or data frame received.

Bit 2 – RTRTAG Remote Transmission Request Tag
RTR bit of the remote or data frame to send. This tag is updated with the corresponding value of the remote or data frame received. In case of Automatic Reply mode, this bit is automatically reset before sending the response.

Bit 1 – RB1TAG Reserved Bit 1 Tag
RB1 bit of the remote or data frame to send. This tag is updated with the corresponding value of the remote or data frame received.

Bit 0 – RB0TAG Reserved Bit 0 Tag
RB0 bit of the remote or data frame to send. This tag is updated with the corresponding value of the remote or data frame received.

21.11.22 CAN Identifier Mask Registers

Name: CANIDM
Offset: 0xF4
Property: R/W

V2.0 part A

Bit	31	30	29	28	27	26	25	24
	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5	IDMSK4	IDMSK3
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IDMSK2	IDMSK1	IDMSK0					
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						RTRMSK		IDEMSK
Access								
Reset								

Bits 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – IDMSK Identifier Mask

Value	Description
0	Comparison true forced
1	Bit comparison enabled

Bit 2 – RTRMSK Remote Transmission Request Mask

Value	Description
0	Comparison true forced
1	Bit comparison enabled

Bit 0 – IDEMSK Identifier Extension Mask

Value	Description
0	Comparison true forced
1	Bit comparison enabled

21.11.23 CAN Identifier Mask Registers

Name: CANIDM
Offset: 0xF4
Property: R/W

V2.0 part B

Bit	31	30	29	28	27	26	25	24
	IDMSK28	IDMSK27	IDMSK26	IDMSK25	IDMSK24	IDMSK23	IDMSK22	IDMSK21
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IDMSK20	IDMSK19	IDMSK18	IDMSK17	IDMSK16	IDMSK15	IDMSK14	IDMSK13
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	IDMSK12	IDMSK11	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	IDMSK4	IDMSK3	IDMSK2	IDMSK1	IDMSK0	RTRMSK		IDEMSK
Access								
Reset								

Bits 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – IDMSK Identifier Mask

Value	Description
0	Comparison true forced
1	Bit comparison enabled

Bit 2 – RTRMSK Remote Transmission Request Mask

Value	Description
0	Comparison true forced
1	Bit comparison enabled

Bit 0 – IDEMSK Identifier Extension Mask
RB0 bit of the remote or data frame to send. This tag is updated with the corresponding value of the remote or data frame received.

Value	Description
0	Comparison true forced
1	Bit comparison enabled

21.11.24 CAN Time Stamp Registers

Name: CANSTM
Offset: 0xF8
Property: R

Bit	15	14	13	12	11	10	9	8
	TIMSTM15	TIMSTM14	TIMSTM13	TIMSTM12	TIMSTM11	TIMSTM10	TIMSTM9	TIMSTM8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	TIMSTM7	TIMSTM6	TIMSTM5	TIMSTM4	TIMSTM3	TIMSTM2	TIMSTM1	TIMSTM0
Access								
Reset								

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 – TIMSTM Time Stamp Count
CAN time stamp counter range 0 to 65,535.

21.11.25 CAN Data Message Register

Name: CANMSG
Offset: 0xFA
Property: R

Bit	7	6	5	4	3	2	1	0
	MSG7	MSG6	MSG5	MSG4	MSG3	MSG2	MSG1	MSG0
Access								
Reset								

Bits 0, 1, 2, 3, 4, 5, 6, 7 – MSG Message Data
This register contains the CAN data byte pointed at the page MOb register.

After writing in the page MOb register, this byte is equal to the specified message location of the pre-defined identifier + index. If auto-incrementation is used, at the end of the data register writing or reading cycle, the index is auto-incremented.

The range of the counting is 8 with no end of loop (0, 1,..., 7, 0,...).

22. LIN / UART - Local Interconnect Network Controller or UART

22.1 Features

22.1.1 LIN

• Hardware implementation of LIN 2.1 (LIN 1.3 Compatibility)

• Small, CPU efficient and independent Master/Slave routines based on “LIN Work Flow Concept” of LIN 2.1 specification

• Automatic LIN header handling and filtering of irrelevant LIN frames

• Automatic LIN response handling

• Extended LIN error detection and signalling

• Hardware frame time-out detection

• “Break-in-data” support capability

• Automatic re-synchronization to ensure proper frame integrity

• Fully flexible extended frames support capabilities

22.1.2 UART

• Full duplex operation (independent serial receive and transmit processes)

• Asynchronous operation

• High resolution baud rate generator

• Hardware support of eight data bits, odd/even/no parity bit, and one stop bit frames

• Data over-run and framing error detection

22.2 Overview

The LIN (Local Interconnect Network) is a serial communications protocol which efficiently supports the control of mechatronics nodes in distributed automotive applications. The main properties of the LIN bus are:

• Single master with multiple slaves concept

• Low cost silicon implementation based on common UART/SCI interface

• Self synchronization in slave node

• Deterministic signal transmission with signal propagation time computable in advance

• Low cost single-wire implementation

• Speed up to 20Kbit/s

LIN provides a cost efficient bus communication where the bandwidth and versatility of CAN are not required. The specification of the line driver/receiver needs to match the ISO9141 NRZ-standard.

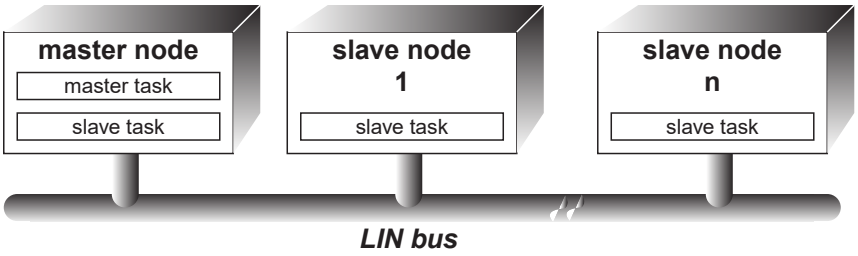
If LIN is not required, the controller alternatively can be programmed as Universal Asynchronous serial Receiver and Transmitter (UART).

22.3 LIN protocol

22.3.1 Master and slave

A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only.

Figure 22-1. LIN cluster with one master node and “n” slave nodes.



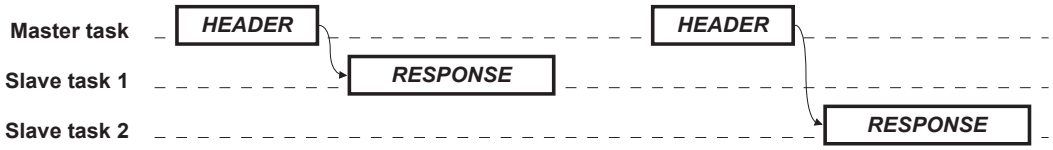
The master task decides when and which frame shall be transferred on the bus. The slave tasks provide the data transported by each frame. Both the master task and the slave task are parts of the Frame handler.

22.3.2 Frames

A frame consists of a header (provided by the master task) and a response (provided by a slave task).

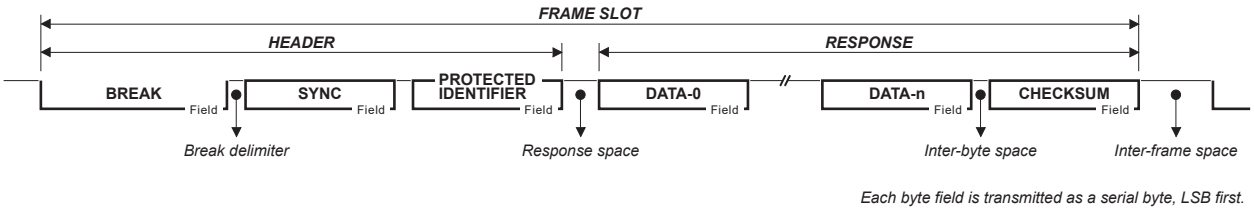
The header consists of a BREAK and SYNC pattern followed by a PROTECTED IDENTIFIER. The identifier uniquely defines the purpose of the frame. The slave task appointed for providing the response associated with the identifier transmits it. The response consists of a DATA field and a CHECKSUM field.

Figure 22-2. Master and slave tasks behavior in LIN frame.



The slave tasks waiting for the data associated with the identifier receives the response and uses the data transported after verifying the checksum.

Figure 22-3. Structure of a LIN frame



Each byte field is transmitted as a serial byte, LSB first.

22.3.3 Data transport

Two types of data may be transported in a frame; signals or diagnostic messages.

- Signals
  Signals are scalar values or byte arrays that are packed into the data field of a frame. A signal is always present at the same position in the data field for all frames with the same identifier.
- Diagnostic messages
  Diagnostic messages are transported in frames with two reserved identifiers. The interpretation of the data field depends on the data field itself as well as the state of the communicating nodes.

22.3.4 Schedule table

The master task (in the master node) transmits frame headers based on a schedule table. The schedule table specifies the identifiers for each header and the interval between the start of a frame and the start of the following frame. The master application may use different schedule tables and select among them.

22.3.5 Compatibility with LIN 1.3

LIN 2.1 is a super-set of LIN 1.3.

A LIN 2.1 master node can handle clusters consisting of both LIN 1.3 slaves and/or LIN 2.1 slaves. The master will then avoid requesting the new LIN 2.1 features from a LIN 1.3 slave:

- Enhanced checksum
- Re-configuration and diagnostics
- Automatic baud rate detection
- “Response error” status monitoring

LIN 2.1 slave nodes can not operate with a LIN 1.3 master node (for example the LIN1.3 master does not support the enhanced checksum).

The LIN 2.1 physical layer is backwards compatible with the LIN1.3 physical layer. But not the other way around. The LIN 2.1 physical layer sets greater requirements, that is, a master node using the LIN 2.1 physical layer can operate in a LIN 1.3 cluster.

22.4 LIN / UART controller

The LIN/UART controller is divided in three main functions:

- Tx LIN Header function
- Rx LIN Header function
- LIN Response function

These functions mainly use two services:

- Rx service
- Tx service

Because these two services are basically UART services, the controller is also able to switch into an UART function.

22.4.1 LIN Overview

The LIN/UART controller is designed to match as closely as possible to the LIN software application structure. The LIN software application is developed as independent tasks, several slave tasks and one master task (c.f. Schedule table). The ATmegaS64M1 conforms to this perspective. The only link between the master task and the slave task will be at the cross-over point where the interrupt routine is called once a new identifier is available. Thus, in a master node, housing both master and slave task, the Tx LIN Header function will alert the slave task of an identifier presence. In the same way, in a slave node, the Rx LIN Header function will alert the slave task of an identifier presence.

When the slave task is warned of an identifier presence, it has first to analyze it to know what to do with the response. Hardware flags identify the presence of one of the specific identifiers from 60 (0x3C) up to 63 (0x3F).

For LIN communication, only four interrupts need to be managed:

- LIDOK: New LIN identifier available
- LRXOK: LIN response received
- LTXOK: LIN response transmitted
- LERR: LIN Error(s)

The wake-up management can be automated using the UART wake-up capability and a node sending a minimum of five low bits (0xF0) for LIN 2.1 and 8 low bits (0x80) for LIN 1.3. Pin change interrupt on LIN wake-up signal can be also used to exit the device of one of its sleep modes.

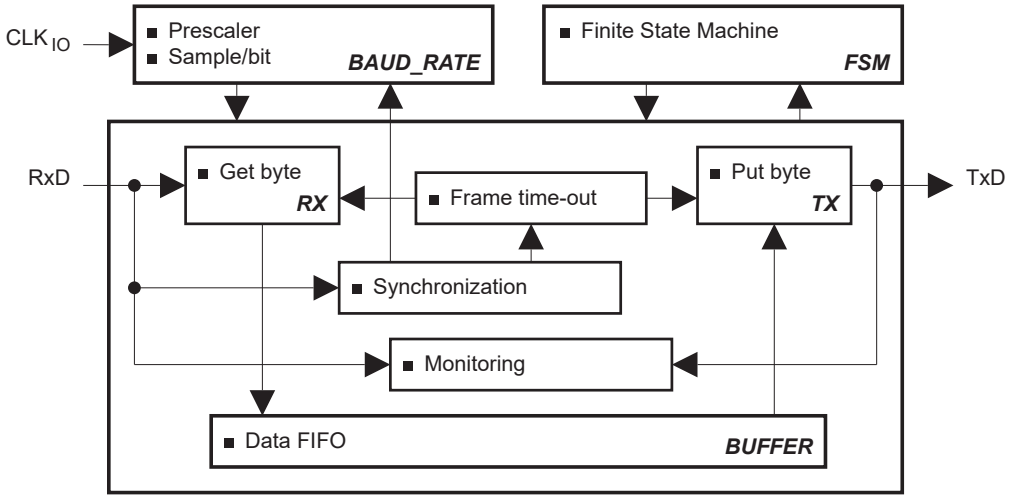
Extended frame identifiers 62 (0x3E) and 63 (0x3F) are reserved to allow the embedding of user-defined message formats and future LIN formats. The byte transfer mode offered by the UART will ensure the upwards compatibility of LIN slaves with accommodation of the LIN protocol.

22.4.2 UART overview

The LIN/UART controller can also function as a conventional UART. By default, the UART operates as a full duplex controller. It has local loop back circuitry for test purposes. The UART has the ability to buffer one character for transmit and two for receive. The receive buffer is made of one 8-bit serial register followed by one 8-bit independent buffer register. Automatic flag management is implemented when the application puts or gets characters, thus reducing the software overhead. Because transmit and receive services are independent, the user can save one device pin when one of the two services is not used. The UART has an enhanced baud rate generator providing a maximum error of 2% whatever the clock frequency and the targeted baud rate.

22.4.3 LIN/UART controller structure

Figure 22-4. LIN/UART controller block diagram.



22.4.4 LIN/UART command overview

Figure 22-5. LIN/UART command dependencies.

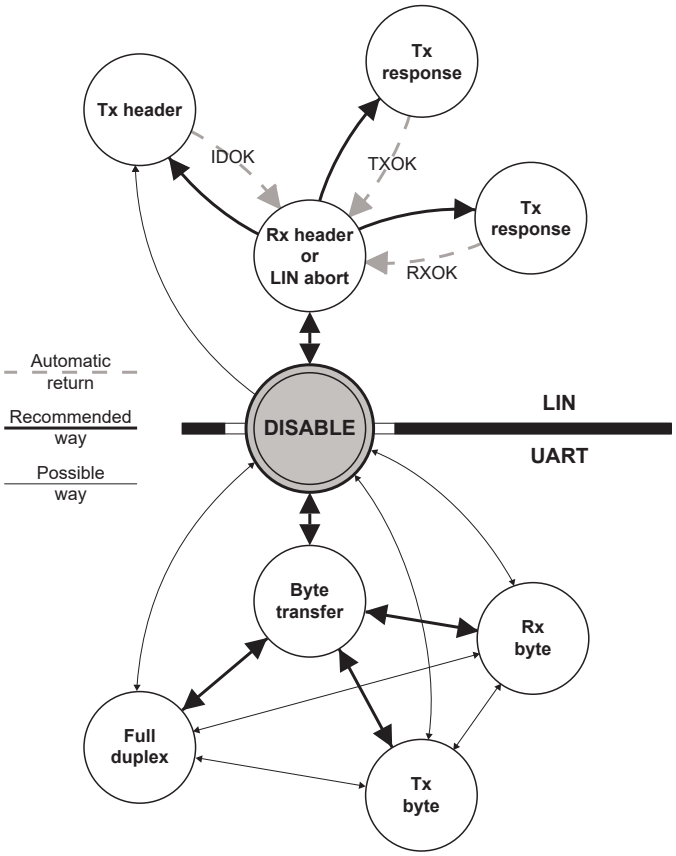


Table 22-1. LIN/UART command list.

LENA	LCMD[2]	LCMD[1]	LCMD[0]	Command	Comment
0	x	x	x	Disable peripheral	
1	0	0	0	Rx Header - LIN abort	LIN withdrawal
			1	Tx header	LCMD[2..0]=000 after Tx
		1	0	Rx response	LCMD[2..0]=000 after Rx
			1	Tx response	LCMD[2..0]=000 after Tx
	1	0	0	Byte transfer	no CRC, no time-out LTXDL=LRXDL=0 (LINDLR: read only register)
		1	0	Rx byte	
		0	1	Tx byte	
		1	1	Full duplex	

22.4.5 Enable / disable

Setting the LENA bit in LINCR register enables the LIN/UART controller. To disable the LIN/UART controller, LENA bit must be written to 0. No wait states are implemented, so, the disable command is taken into account immediately.

22.4.6 LIN commands

Clearing the LCMD[2] bit in LINCR register enables LIN commands.

As shown in Table 22-1, four functions controlled by the LCMD[1..0] bits of LINCR register are available (see Figure 22-5).

22.4.6.1 Rx header / LIN abort function

This function (or state) is mainly the withdrawal mode of the controller.

When the controller has to execute a master task, this state is the start point before enabling a Tx Header command.

When the controller has only to execute slave tasks, LIN header detection/acquisition is enabled as background function. At the end of such an acquisition (Rx Header function), automatically the appropriate flags are set, and in LIN 1.3, the LINDLR register is set with the uncoded length value.

This state is also the start point before enabling the Tx or the Rx Response command.

A running function (that is Tx Header, Tx or Rx Response) can be aborted by clearing LCMD[1..0] bits in LINCR register. In this case, an abort flag - LABORT - in LINERR register will be set to inform the other software tasks. No wait states are implemented, so, the abort command is taken into account immediately.

Rx Header function is responsible for:

- The BREAK field detection
- The hardware re-synchronization analyzing the SYNCH field
- The reception of the PROTECTED IDENTIFIER field, the parity control and the update of the LINDLR register in case of LIN 1.3
- The starting of the Frame_Time_Out
- The checking of the LIN communication integrity

22.4.6.2 Tx header function

In accordance with the LIN protocol, only the master task must enable this function. The header is sent in the appropriate timed slots at the programmed baud rate (c.f. LINBRR & LINBTR registers).

The controller is responsible for:

- The transmission of the BREAK field - 13 dominant bits
- The transmission of the SYNCH field - character 0x55
- The transmission of the PROTECTED IDENTIFIER field. It is the full content of the LINIDR register (automatic check bits included)

At the end of this transmission, the controller automatically returns to Rx Header / LIN Abort state (that is, LCMD[1..0] = 00) after setting the appropriate flags. This function leaves the controller in the same setting as after the Rx Header function. This means that, in LIN 1.3, the LINDLR register is set with the uncoded length value at the end of the Tx Header function.

During this function, the controller is also responsible for:

- The starting of the Frame_Time_Out
- The checking of the LIN communication integrity

22.4.6.3 Rx & TX response functions

These functions are initiated by the slave task of a LIN node. They must be used after sending an header (master task) or after receiving an header (considered as belonging to the slave task). When the TX

Response order is sent, the transmission begins. A Rx Response order can be sent up to the reception of the last serial bit of the first byte (before the stop-bit).

In LIN 1.3, the header slot configures the LINDLR register. In LIN 2.1, the user must configure the LINDLR register, either LRXDL[3..0] for Rx Response either LTXDL[3..0] for Tx Response.

When the command starts, the controller checks the LIN13 bit of the LINCR register to apply the right rule for computing the checksum. Checksum calculation over the DATA bytes and the PROTECTED IDENTIFIER byte is called enhanced checksum and it is used for communication with LIN 2.1 slaves. Checksum calculation over the DATA bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Note that identifiers 60 (0x3C) to 63 (0x3F) shall always use classic checksum.

At the end of this reception or transmission, the controller automatically returns to Rx Header / LIN Abort state (that is LCMD[1..0] = 00) after setting the appropriate flags.

If an LIN error occurs, the reception or the transmission is stopped, the appropriate flags are set and the LIN bus is left to recessive state.

During these functions, the controller is responsible for:

- The initialization of the checksum operator
- The transmission or the reception of ‘n’ data with the update of the checksum calculation
- The transmission or the checking of the CHECKSUM field
- The checking of the Frame_Time_Out
- The checking of the LIN communication integrity

While the controller is sending or receiving a response, BREAK and SYNCH fields can be detected and the identifier of this new header will be recorded. Of course, specific errors on the previous response will be maintained with this identifier reception.

22.4.6.4 Handling data of LIN response

A FIFO data buffer is used for data of the LIN response. After setting all parameters in the LINSEL register, repeated accesses to the LINDAT register perform data read or data write (see Data management).

Note that LRXDL[3..0] and LTXDL[3..0] are not linked to the data access.

22.4.7 UART commands

Setting the LCMD[2] bit in LINENR register enables UART commands. Tx Byte and Rx Byte services are independent as shown in Table 22-1.

- Byte Transfer: the UART is selected but both Rx and Tx services are disabled
- Rx Byte: only the Rx service is enable but Tx service is disabled
- Tx Byte: only the Tx service is enable but Rx service is disabled
- Full Duplex: the UART is selected and both Rx and Tx services are enabled

This combination of services is controlled by the LCMD[1..0] bits of LINENR register (see Figure 22-5).

22.4.7.1 Data handling

The FIFO used for LIN communication is disabled during UART accesses. LRXDL[3..0] and LTXDL[3..0] values of LINDLR register are then irrelevant. LINDAT register is then used as data register and LINSEL register is not relevant.

22.4.7.2 Rx service

Once this service is enabled, the user is warned of an in-coming character by the LRXOK flag of LINSIR register. Reading LINDAT register automatically clears the flag and makes free the second stage of the buffer. If the user considers that the in-coming character is irrelevant without reading it, he directly can clear the flag (see specific flag management described in Section 20.6.2 on page 208).

The intrinsic structure of the Rx service offers a 2-byte buffer. The fist one is used for serial to parallel conversion, the second one receives the result of the conversion. This second buffer byte is reached reading LINDAT register. If the 2-byte buffer is full, a new in-coming character will overwrite the second one already recorded. An OVRERR error in LINERR register will then accompany this character when read.

A FERR error in LINERR register will be set in case of framing error.

22.4.7.3 Tx service

If this service is enabled, the user sends a character by writing in LINDAT register. Automatically the LTXOK flag of LINSIR register is cleared. It will rise at the end of the serial transmission. If no new character has to be sent, LTXOK flag can be cleared separately (see specific flag management described in “LINSIR – LIN Status and Interrupt Register” on page 208).

There is no transmit buffering.

No error is detected by this service.

22.5 LIN / UART description

22.5.1 Reset

The AVR core reset logic signal also resets the LIN/UART controller. Another form of reset exists, a software reset controlled by LSWRES bit in LINCR register. This self-reset bit performs a partial reset.

Table 22-2. Reset of LIN/UART registers.

Register	Name	Reset Value	LSWRES value	Comment
LIN control reg.	LINCR	0000 0000 $_b$	0000 0000 $_b$	x=unknown u=unchanged
LIN status & interrupt reg.	LINSIR	0000 0000 $_b$	0000 0000 $_b$	
LIN enable interrupt reg.	LINENIR	0000 0000 $_b$	xxxx 0000 $_b$	
LIN error reg.	LINERR	0000 0000 $_b$	0000 0000 $_b$	
LIN bit timing reg.	LINBTR	0010 0000 $_b$	0010 0000 $_b$	
LIN baud rate reg. low	LINBRRL	0000 0000 $_b$	uuuu uuuu $_b$	
LIN baud rate reg. high	LINBRRH	0000 0000 $_b$	xxxx uuuu $_b$	
LIN data length reg.	LINDLR	0000 0000 $_b$	0000 0000 $_b$	
LIN identifier reg.	LINIDR	1000 0000 $_b$	1000 0000 $_b$	
LIN data buffer selection	LINSEL	0000 0000 $_b$	xxxx 0000 $_b$	
LIN data	LINDAT	0000 0000 $_b$	0000 0000 $_b$	

22.5.2 Clock

The I/O clock signal ($clk_{i/o}$) also clocks the LIN/UART controller. It is its unique clock.

22.5.3 LIN protocol selection

LIN13 bit in LINCR register is used to select the LIN protocol:

- LIN13 = 0 (default): LIN 2.1 protocol
- LIN13 = 1: LIN 1.3 protocol

The controller checks the LIN13 bit in computing the checksum (enhanced checksum in LIN2.1 / classic checksum in LIN 1.3). See Rx & TX response functions.

This bit is irrelevant for UART commands.

22.5.4 Configuration

Depending on the mode (LIN or UART), LCONF[1..0] bits of the LINCR register set the controller in the following configuration:

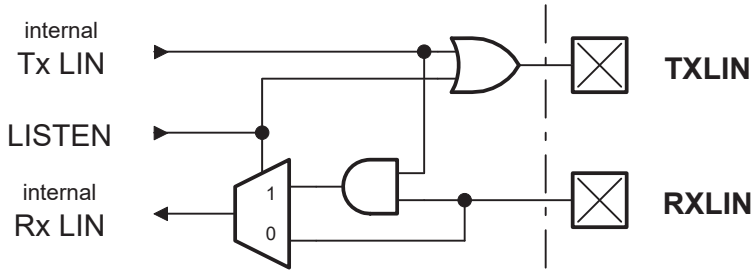
Table 22-3. Configuration table versus mode.

Mode	LCONF[1..0]	Configuration
LIN	00_b	LIN standard configuration (default)
	01_b	No CRC field detection or transmission
	10_b	Frame_Time_Out disable
	11_b	Listening mode
UART	00_b	8-bit data, no parity & 1 stop-bit
	01_b	8-bit data, even parity & 1 stop-bit
	10_b	8-bit data, odd parity & 1 stop-bit
	11_b	Listening mode, 8-bit data, no parity & 1 stop-bit

The LIN configuration is independent of the programmed LIN protocol.

The listening mode connects the internal Tx LIN and the internal Rx LIN together. In this mode, the TXLIN output pin is disabled and the RXLIN input pin is always enabled. The same scheme is available in UART mode.

Figure 22-6. Listening mode.

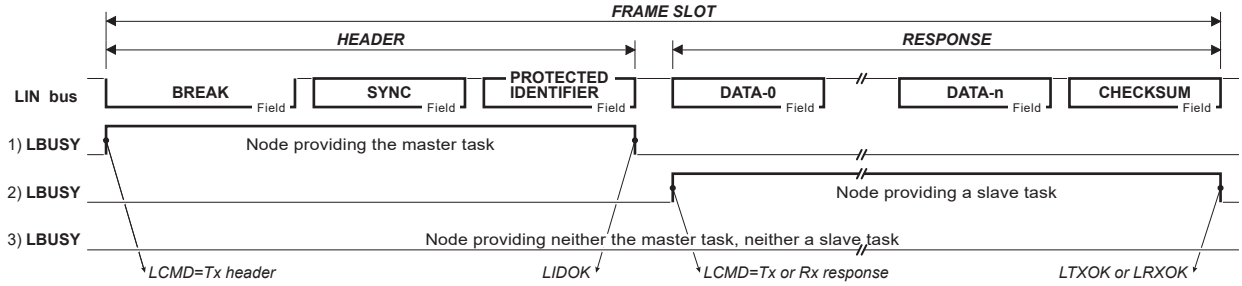


22.5.5 Busy signal

LBUSY bit flag in LINSIR register is the image of the BUSY signal. It is set and cleared by hardware. It signals that the controller is busy with LIN or UART communication.

22.5.5.1 Busy signal in LIN mode

Figure 22-7. Busy signal in LIN mode



When the busy signal is set, some registers are locked, user writing is not allowed:

- “LIN Control Register” - LINCR - except LCMD[2..0], LENA & LSWRES
- “LIN Baud Rate Registers” - LINBRRL & LINBRRH
- “LIN Data Length Register” - LINDLR
- “LIN Identifier Register” - LINIDR
- “LIN Data Register” - LINDAT

If the busy signal is set, the only available commands are:

- LCMD[1..0] = 00_b, the abort command is taken into account at the end of the byte
- LENA = 0 and/or LCMD[2] = 0, the kill command is taken into account immediately
- LSWRES = 1, the reset command is taken into account immediately

Note that, if another command is entered during busy signal, the new command is not validated and the LOVRERR bit flag of the LINERR register is set. The on-going transfer is not interrupted.

22.5.5.2 Busy signal in UART mode

During the byte transmission, the busy signal is set. This locks some registers from being written:

- “LIN Control Register” - LINCR - except LCMD[2..0], LENA & LSWRES
- “LIN Data Register” - LINDAT

The busy signal is not generated during a byte reception.

22.5.6 Bit timing

22.5.6.1 Baud rate generator

The baud rate is defined to be the transfer rate in bits per second (bps):

- BAUD: Baud rate (in bps)
- $clk_{i/o}$: System I/O clock frequency
- LDIV[11..0]: Contents of LINBRRH & LINBRRL registers - (0-4095), the pre-scaler receives $clk_{i/o}$ as input clock
- LBT[5..0]: Least significant bits of - LINBTR register- (0-63) is the number of samplings in a LIN or UART bit (default value 32)

Equation for calculating baud rate:

$BAUD = fclk_{i/o} / LBT[5..0] \times (LDIV[11..0] + 1)$

Equation for setting LINDIV value:

$LDIV[11..0] = (fclk_{i/o} / LBT[5..0] \times BAUD) - 1$

Note that in reception a majority vote on three samplings is made.

22.5.6.2 Re-synchronization in LIN mode

When waiting for Rx Header, LBT[5..0] = 32 in LINBTR register. The re-synchronization begins when the BREAK is detected. If the BREAK size is not in the range (11 bits min., 28 bits max. — 13 bits nominal), the BREAK is refused. The re-synchronization is done by adjusting LBT[5..0] value to the SYNCH field of the received header (0x55). Then the PROTECTED IDENTIFIER is sampled using the new value of LBT[5..0]. The re-synchronization implemented in the controller tolerates a clock deviation of ±20% and adjusts the baud rate in a ±2% range.

The new LBT[5..0] value will be used up to the end of the response. Then, the LBT[5..0] will be reset to 32 for the next header.

The LINBTR register can be used to re-calibrate the clock oscillator.

The re-synchronization is not performed if the LIN node is enabled as a master.

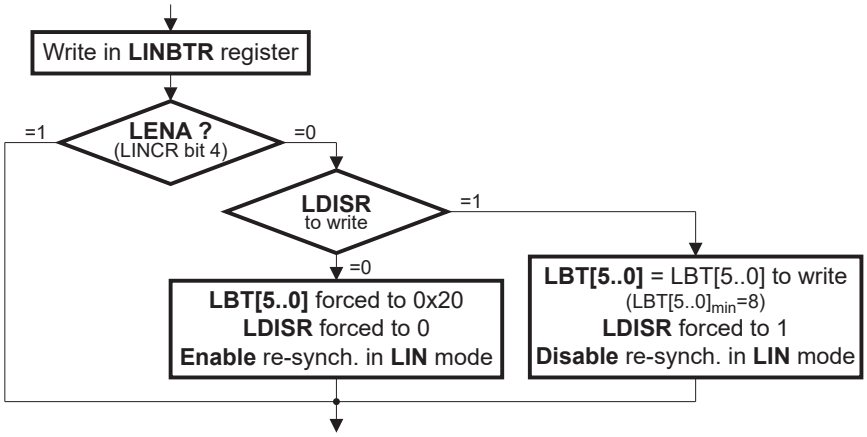
22.5.6.3 Handling LBT[5:0]

LDISR bit of LINBTR register is used to:

- To enable the setting of LBT[5:0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of eight is required for LBT[5:0] due to the sampling operation
- Disable the re-synchronization in LIN Slave Mode for test purposes

Note that the LENA bit of LINCR register is important for this handling.

Figure 22-8. Handling LBT[5:0].



22.5.7 Data length

LIN commands describes how to set or how are automatically set the LRXDL[3..0] or LTXDL[3..0] fields of LINDLR register before receiving or transmitting a response.

In the case of Tx Response the LRXDL[3..0] will be used by the hardware to count the number of bytes already successfully sent.

In the case of Rx Response the LTXDL[3..0] will be used by the hardware to count the number of bytes already successfully received.

If an error occurs, this information is useful to the programmer to recover the LIN messages.

22.5.7.1 Data length in LIN 2.1

- If LTXDL[3..0]=0 only the CHECKSUM will be sent

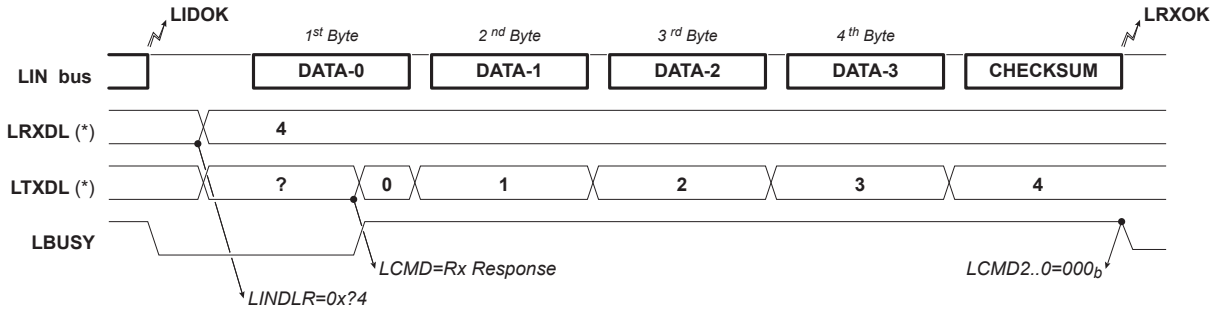
- If LRXDL[3..0]=0 the first byte received will be interpreted as the CHECKSUM
- If LTXDL[3..0] or LRXDL[3..0] >8, values will be forced to eight after the command setting and before sending or receiving of the first byte

22.5.7.2 Data length in LIN 1.3

- LRXDL and LTXDL fields are both hardware updated before setting LIDOK by decoding the data length code contained in the received PROTECTED IDENTIFIER (LRXDL = LTXDL)
- Via the above mechanism, a length of 0 or >8 is not possible

22.5.7.3 Data length in Rx Response

Figure 22-9. LIN2.1 - Rx Response - no error

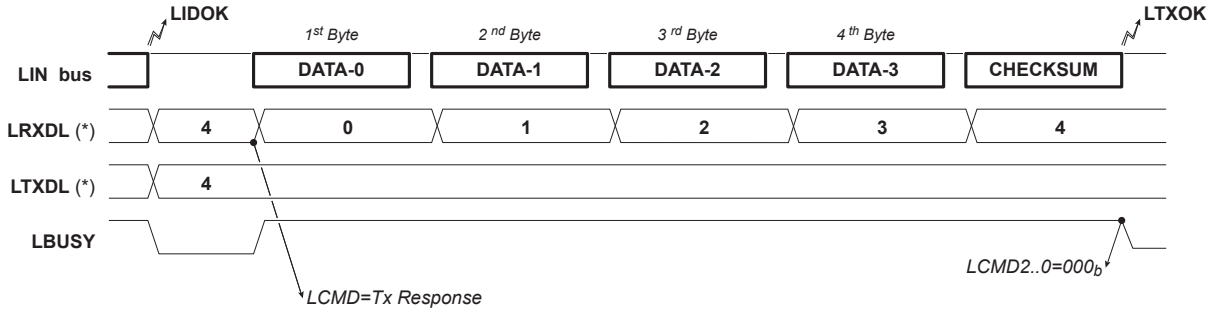


(*) : LRXDL & LTXDL updated by user

- The user initializes LRXDL field before setting the Rx Response command
- After setting the Rx Response command, LTXDL is reset by hardware
- LRXDL field will remain unchanged during Rx (during busy signal)
- LTXDL field will count the number of received bytes (during busy signal)
- If an error occurs, Rx stops, the corresponding error flag is set and LTXDL will give the number of received bytes without error
- If no error occurs, LRXOK is set after the reception of the CHECKSUM, LRXDL will be unchanged (and LTXDL = LRXDL)

22.5.7.4 Data length in Tx Response

Figure 22-10. LIN1.3 - Tx Response - no error



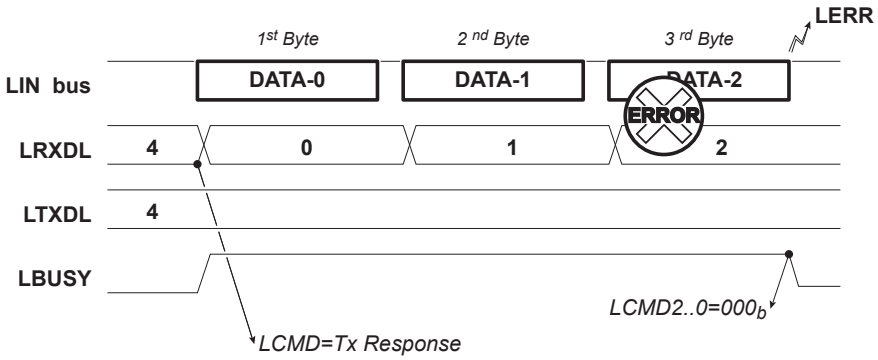
(*) : LRXDL & LTXDL updated by Rx Response or Tx Response task

- The user initializes LTXDL field before setting the Tx Response command
- After setting the Tx Response command, LRXDL is reset by hardware
- LTXDL will remain unchanged during Tx (during busy signal)
- LRXDL will count the number of transmitted bytes (during busy signal)

- If an error occurs, Tx stops, the corresponding error flag is set and LRXDL will give the number of transmitted bytes without error
- If no error occurs, LTXOK is set after the transmission of the CHECKSUM, LTXDL will be unchanged (and LRXDL = LTXDL)

22.5.7.5 Data length after error

Figure 22-11. Tx Response - error.



Note: Information on response (ex: error on byte) is only available at the end of the serialization/de-serialization of the byte.

22.5.7.6 Data length in UART mode

- The UART mode forces LRXDL and LTXDL to 0 and disables the writing in LINDLR register
- Note that after reset, LRXDL and LTXDL are also forced to 0

22.5.8 xxOK flags

There are three xxOK flags in LINSIR register:

- LIDOK: LIN IDentifier OK
  It is set at the end of the header, either by the Tx Header function or by the Rx Header. In LIN 1.3, before generating LIDOK, the controller updates the LRXDL & LTXDL fields in LINDLR register.
  It is not driven in UART mode.
- LRXOK: LIN RX response complete
  It is set at the end of the response by the Rx Response function in LIN mode and once a character is received in UART mode.
- LTXOK: LIN TX response complete
  It is set at the end of the response by the Tx Response function in LIN mode and once a character has been sent in UART mode.

These flags can generate interrupts if the corresponding enable interrupt bit is set in the LINENIR register (see Interrupts).

22.5.9 xxERR flags

LERR bit of the LINSIR register is an logical ‘OR’ of all the bits of LINERR register (see Interrupts). There are eight flags:

- LBERR = LIN Bit ERRor
  . A unit that is sending a bit on the bus also monitors the bus. A LIN bit error will be flagged when the bit value that is monitored is different from the bit value that is sent. After detection of a LIN bit error the transmission is aborted.

- LCERR = LIN Checksum ERRor. A LIN checksum error will be flagged if the inverted modulo-256 sum of all received data bytes (and the protected identifier in LIN 2.1) added to the checksum does not result in 0xFF.
- LPERR = LIN Parity ERRor (identifier). A LIN parity error in the IDENTIFIER field will be flagged if the value of the parity bits does not match with the identifier value. (See LP[1:0] bits in LINIDR. A LIN slave application does not distinguish between corrupted parity bits and a corrupted identifier. The hardware does not undertake any correction. However, the LIN slave application has to solve this as:
  - known identifier (parity bits corrupted)
  - or corrupted identifier to be ignored
  - or new identifier
- LSERR = LIN Synchronization ERRor. A LIN synchronization error will be flagged if a slave detects the edges of the SYNCH field outside the given tolerance.
- LFERR = LIN Framing ERRor. A framing error will be flagged if dominant STOP bit is sampled. Same function in UART mode.
- LTOERR = LIN Time Out ERRor. A time-out error will be flagged if the MESSAGE frame is not fully completed within the maximum length $T_{Frame\_Maximum}$ by any slave task upon transmission of the SYNCH and IDENTIFIER fields (see Section Frame time out).
- LOVERR = LIN OVerrun ERRor. Overrun error will be flagged if a new command (other than LIN Abort) is entered while ‘Busy signal’ is present.
  In UART mode, an overrun error will be flagged if a received byte overwrites the byte stored in the serial input buffer.
- LABORT. LIN abort transfer reflects a previous LIN Abort command (LCMD[2..0] = 000) while ‘Busy signal’ is present.

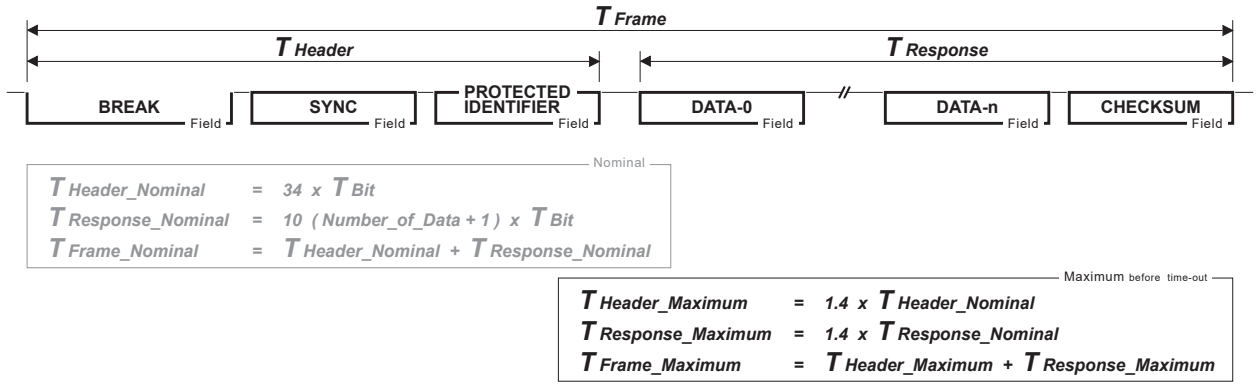
After each LIN error, the LIN controller stops its previous activity and returns to its withdrawal mode (LCMD[2..0] = 000_b) as illustrated in Figure 22-11.

Writing 1 in LERR of LINSIR register resets LERR bit and all the bits of the LINERR register.

22.5.10 Frame time out

According to the LIN protocol, a frame time-out error is flagged if: $T_{Frame} > T_{Frame\_Maximum}$. This feature is implemented in the LIN/UART controller.

Figure 22-12. LIN timing and frame time-out



22.5.11 Break-in-data

According to the LIN protocol, the LIN/UART controller can detect the BREAK/SYNC field sequence even if the break is partially superimposed with a byte of the response. When a BREAK/SYNC field sequence happens, the transfer in progress is aborted and the processing of the new frame starts.

- On slave node(s), an error is generated (that is, LBERR in case of Tx Response or LFERR in case of Rx Response). Information on data error is also available, refer to Data length after error.
- On master node, the user (code) is responsible for this aborting of frame. To do this, the master task has first to abort the on-going communication (clearing LCMD bits - LIN Abort command) and then to apply the Tx Header command. In this case, the abort error flag - LABORT - is set

On the slave node, the BREAK detection is processed with the synchronization setting available when the LIN/UART controller processed the (aborted) response. But the re-synchronization restarts as usual. Due to a possible difference of timing reference between the BREAK field and the rest of the frame, the time-out values can be slightly inaccurate.

22.5.12 Checksum

The last field of a frame is the checksum.

In LIN 2.1, the checksum contains the inverted eight bit sum with carry over all data bytes and the protected identifier. This calculation is called enhanced checksum.

$$\text{CHECKSUM} = 255 - \left(\text{unsigned char}\left(\left(\sum_0^n \text{DATA}_n\right) + \text{PROTECTED ID.}\right)\right.$$
$$\left. + \text{unsigned char}\left(\left(\left(\sum_0^n \text{DATA}_n\right) + \text{PROTECTED ID.}\right) \gg 8\right)\right)$$

In LIN 1.3, the checksum contains the inverted eight bit sum with carry over all data bytes. This calculation is called classic checksum.

$$\text{CHECKSUM} = 255 - \left(\text{unsigned char}\left(\sum_0^n \text{DATA}_n\right) + \text{unsigned char}\left(\left(\sum_0^n \text{DATA}_n\right) \gg 8\right)\right)$$

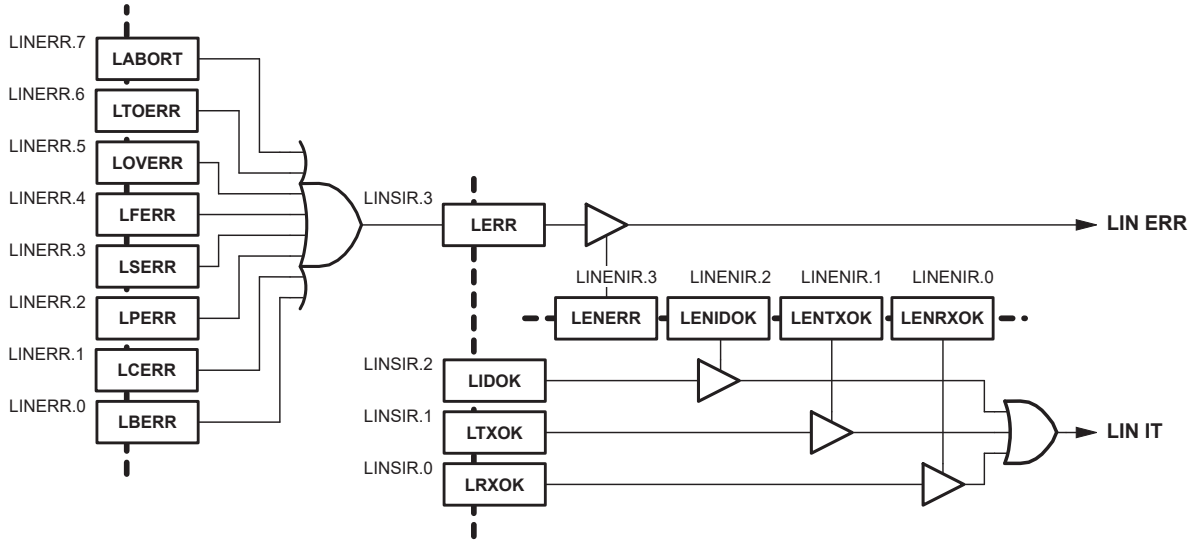
Frame identifiers 60 (0x3C) to 61 (0x3D) shall always use classic checksum.

22.5.13 Interrupts

As shown in the figure below, the four communication flags of the LINSIR register are combined to drive two interrupts. Each of these flags have their respective enable interrupt bit in LINENIR register.

See xxOK flags and xxERR flags.

Figure 22-13. LIN interrupt mapping



22.5.14 Message filtering

Message filtering based upon the whole identifier is not implemented. Only a status for frame headers having 0x3C, 0x3D, 0x3E and 0x3F as identifier is available in the LINSIR register.

Table 22-4. Frame status

LIDST[2..0]	Frame status
$0xx_b$	No specific identifier
100_b	60 (0x3C) identifier
101_b	61 (0x3D) identifier
110_b	62 (0x3E) identifier
111_b	63 (0x3F) identifier

The LIN protocol says that a message with an identifier from 60 (0x3C) up to 63 (0x3F) uses a classic checksum (sum over the data bytes only). Software will be responsible for switching correctly the LIN13 bit to provide/check this expected checksum (the insertion of the ID field in the computation of the CRC is set - or not - just after entering the Rx or Tx Response command).

22.5.15 Data management

22.5.15.1 LIN FIFO data buffer

To preserve register allocation, the LIN data buffer is seen as a FIFO (with address pointer accessible). This FIFO is accessed via the LINDX[2..0] field of LINSEL register through the LINDAT register.

LINDX[2..0], the data index, is the address pointer to the required data byte. The data byte can be read or written. The data index is automatically incremented after each LINDAT access if the LAINC (active low) bit is cleared. A roll-over is implemented, after data index=7 it is data index=0. Otherwise, if LAINC bit is set, the data index needs to be written (updated) before each LINDAT access.

The first byte of a LIN frame is stored at the data index=0, the second one at the data index=1, and so on. Nevertheless, LINSEL must be initialized by the user before use.

22.5.15.2 UART data register

The LINDAT register is the data register (no buffering - no FIFO). In write access, LINDAT will be for data out and in read access, LINDAT will be for data in.

In UART mode the LINSEL register is unused.

22.5.16 OCD support

This chapter describes the behavior of the LIN/UART controller stopped by the OCD (that is I/O view behavior in Atmel Studio®).

1. LINCR
   - LINCR[6..0] are R/W accessible
   - LSWRES always is a self-reset bit (needs one micro-controller cycle to execute)
2. LINSIR
   - LIDST[2..0] and LBUSY are always Read accessible
   - LERR & LxxOK bit are directly accessible (unlike in execution, set or cleared directly by writing 1 or 0)
   - Note that clearing LERR resets all LINERR bits and setting LERR sets all LINERR bits
3. LINENR
   - All bits are R/W accessible
4. LINERR
   - All bits are R/W accessible
   - Note that LINERR bits are ORed to provide the LERR interrupt flag of LINSIR
5. LINBTR
   - LBT[5..0] are R/W access only if LDISR is set
   - If LDISR is reset, LBT[5..0] are unchangeable
6. LINBRRH & LINBRRL
   - All bits are R/W accessible
7. LINDLR
   - All bits are R/W accessible
8. LINIDR
   - LID[5..0] are R/W accessible
   - LP[1..0] are Read accessible and are always updated on the fly
9. LINSEL
   - All bits are R/W accessible
10. LINDAT
    - All bits are R/W accessible
    - Note that LAINC has no more effect on the auto-incrementation and the access to the full FIFO is done setting LINDX[2..0] of LINSEL
    - **Note:** When a debugger break occurs, the state machine of the LIN/UART controller is stopped (included frame time-out) and further communication may be corrupted.

22.6 Register Description

22.6.1 LIN Control Register

Name: LINCR
Offset: 0xC8
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	LSWRES	LIN13	LCONF[1:0]		LENA	LCMD[2:0]		
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – LSWRES Software Reset

Value	Description
0	No action
1	Software reset (this bit is self-reset at the end of the reset procedure)

Bit 6 – LIN13 LIN 1.3 mode

Value	Description
0	LIN 2.1 (default)
1	LIN 1.3

Bits 5:4 – LCONF[1:0] Configuration
The configuration settings for LIN and UART mode are shown below

Value	Name	Description
00	LIN Mode	LIN Standard configuration (listen mode “off”, CRC “on” & Frame_Time_Out “on”) (default)
01	LIN Mode	No CRC, no Time out (listen mode “off”)
10	LIN Mode	No Frame_Time_Out (listen mode “off” & CRC “on”)
11	LIN Mode	Listening mode (CRC “on” & Frame_Time_Out “on”)
00	UART Mode	8-bit, no parity (listen mode “off”) (default)
01	UART Mode	8-bit, even parity (listen mode “off”)
10	UART Mode	8-bit, odd parity (listen mode “off”)
11	UART Mode	Listening mode, 8-bit, no parity

Bit 3 – LENA Enable

Value	Description
0	Disable (both LIN and UART modes)
1	Enable (both LIN and UART modes)

Bits 2:0 – LCMD[2:0] Command and mode
The command is only available if LENA is set

Value	Description
000	LIN Rx Header - LIN abort
001	LIN Tx Header

Value	Description
010	LIN Rx Response
011	LIN Tx Response
100	UART Rx & Tx Byte disable
11x	UART Rx Byte enable
1x1	UART Tx Byte enable

22.6.2 LIN Status and Interrupt Register

Name: LINSIR
Offset: 0xC9
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	LIDST[2:0]			LBUSY	LERR	LIDOK	LTXOK	LRXOK
Access								
Reset	0	0	0	0	0	0	0	0

Bits 7:5 – LIDST[2:0] LIN Identifier Status

Value	Description
0xx	No specific identifier
100	Identifier 60 (0x3C)
101	Identifier 61 (0x3D)
110	Identifier 62 (0x3E)
111	Identifier 63 (0x3F)

Bit 4 – LBUSY Busy Signal

Value	Description
0	Not busy
1	Busy (receiving or transmitting)

Bit 3 – LERR Error Interrupt
It is a logical OR of LINERR register bits. This bit generates an interrupt if its respective enable bit - LENERR - is set in LINENIR.

The user clears this bit by writing 1 in order to reset this interrupt. Resetting LERR also resets all LINERR bits. In UART mode, this bit is also cleared by reading LINDAT.

Value	Description
0	No error
1	An error has occurred

Bit 2 – LIDOK Identifier Interrupt
This bit generates an interrupt if its respective enable bit - LENIDOK - is set in LINENIR.

The user clears this bit by writing 1, in order to reset this interrupt.

Value	Description
0	No identifier
1	Slave task: Identifier present, master task: Tx Header complete

Bit 1 – LTXOK Transmit Performed Interrupt
This bit generates an interrupt if its respective enable bit - LENTXOK - is set in LINENIR.

The user clears this bit by writing 1, in order to reset this interrupt.

In UART mode, this bit is also cleared by writing LINDAT.

Value	Description
0	No Tx
1	Tx Response complete

Bit 0 – LRXOK Receive Performed Interrupt
This bit generates an interrupt if its respective enable bit - LENRXOK - is set in LINENIR.

The user clears this bit by writing 1, in order to reset this interrupt.

In UART mode, this bit is also cleared by writing LINDAT.

Value	Description
0	No Rx
1	Rx Response complete

22.6.3 LIN Enable Interrupt Register

Name: LINENIR
Offset: 0xCA
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
					LENERR	LENIDOK	LENTXOK	LENRXOK
Access								
Reset					0	0	0	0

Bit 3 – LENERR Enable Error Interrupt

Value	Description
0	Error interrupt masked
1	Error interrupt enabled

Bit 2 – LENIDOK Enable Identifier Interrupt

Value	Description
0	Identifier interrupt masked
1	Identifier interrupt enabled

Bit 1 – LENTXOK Enable Transmit Performed Interrupt

Value	Description
0	Transmit performed interrupt masked
1	Transmit performed interrupt enabled

Bit 0 – LENRXOK Enable Receive Performed Interrupt

Value	Description
0	Receive performed interrupt masked
1	Receive performed interrupt enabled

22.6.4 LIN Error Register

Name: LINERR
Offset: 0xCB
Reset: 0x0
Property: R

Bit	7	6	5	4	3	2	1	0
	LABORT	LTOERR	LOVERR	LFERR	LSERR	LPERR	LCERR	LBERR
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – LABORT Abort Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No warning
1	LIN abort command occurred

Bit 6 – LTOERR Frame_Time_Out Error Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Frame_Time_Out error

Bit 5 – LOVERR Overrun Error Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Overrun error

Bit 4 – LFERR Framing Error Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Framing error

Bit 3 – LSERR LSERR:
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Synchronization error

Bit 2 – LPERR Parity Error Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Parity error

Bit 1 – LCERR Checksum Error Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Checksum error

Bit 0 – LBERR Bit Error Flag
This bit is cleared when LERR bit in LINSIR is cleared.

Value	Description
0	No error
1	Bit error

22.6.5 LIN Bit Timing Register

Name: LINBTR
Offset: 0xCC
Reset: 0x20
Property: R/W

Bit	7	6	5	4	3	2	1	0
	LDISR		LBT[5:0]					
Access								
Reset	0		1	0	0	0	0	0

Bit 7 – LDISR Disable Bit Timing Re synchronization

Value	Description
0	Bit timing re-synchronization enabled (default)
1	Bit timing re-synchronization disabled

Bits 5:0 – LBT[5:0] LIN Bit Timing
Gives the number of samples of a bit. Sample-time = $(1/fclk_{i/o})$ × (LDIV[11..0] + 1). Default value: LBT[6:0]=32 — Min. value: LBT[6:0]=8 — Max. value: LBT[6:0]=63

22.6.6 LIN Baud Rate Register

Name: LINBRR
Offset: 0xCD
Reset: 0x0
Property: R/W

Bit	15	14	13	12	11	10	9	8
					LDIV11	LDIV10	LDIV9	LDIV8
Access								
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LDIV7	LDIV6	LDIV5	LDIV4	LDIV3	LDIV2	LDIV1	LDIV0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – LDIV Scaling of $clk_{i/o}$ Frequency
The LDIV value is used to scale the entering $clk_{i/o}$ frequency to achieve appropriate LIN or UART baud rate.

22.6.7 LIN Data Length Register

Name: LINDLR
Offset: 0xCF
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	LTXDL[3:0]				LRXDL[3:0]			
Access								
Reset	0	0	0	0	0	0	0	0

Bits 7:4 – LTXDL[3:0] LIN Transmit Data Length
In LIN mode, this field gives the number of bytes to be transmitted (clamped to 8 Max).

In UART mode this field is unused.

Bits 3:0 – LRXDL[3:0] LIN Receive Data Length
In LIN mode, this field gives the number of bytes to be received (clamped to 8 Max).

In UART mode this field is unused.

22.6.8 LIN Identifier Register

Name: LINIDR
Offset: 0xD0
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	LP1	LP0	LDL1	LDL0	LID3	LID2	LID1	LID0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 6, 7 – LP Parity
In LIN mode:

```
LP0 = LID4 ^ LID2 ^ LID1 ^ LID0
LP1 = ! ( LID1 ^ LID3 ^ LID4 ^ LID5 )
```

In UART mode this field is unused.

Bits 4, 5 – LDL LIN 1.3 Data Length or LIN 2.1 Identifier
In LIN 1.3 mode as in the table below.

In LIN 2.1 mode these 2 bits belong to the 6-bit identifier (no length transported).

In UART mode this field is unused.

Value	Description
00	2-byte response
01	2-byte response
10	4-byte response
11	8-byte response

Bits 0, 1, 2, 3, 4, 5 – LID LIN 1.3 Identifier and 2.1 Identifier
In LIN 1.3 mode LID is a 4-bit identifier.

In LIN 2.1 mode LID is a 6-bit identifier (no length transported).

In UART mode this field is unused.

Value	Description
00	2-byte response
01	2-byte response
10	4-byte response
11	8-byte response

22.6.9 LIN Data Buffer Selection Register

Name: LINSEL
Offset: 0xD1
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
					$\overline{\text{LAINC}}$	LINDX2	LINDX1	LINDX0
Access								
Reset					0	0	0	0

Bit 3 – $\overline{\text{LAINC}}$ $\overline{\text{Auto Increment}}$ of Data Buffer Index
In LIN mode as in the table below

In UART mode this field is unused.

Value	Description
0	Auto incrementation of FIFO data buffer index (default)
1	No auto incrementation

Bits 0, 1, 2 – LINDX FIFO LIN Data Buffer Index
In LIN mode: location (index) of the LIN response data byte into the FIFO data buffer. The FIFO data buffer is accessed through LINDAT.

In UART mode this field is unused.

22.6.10 LIN Data Register

Name: LINDAT
Offset: 0xD2
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	$\overline{\text{LDATA7}}$	$\overline{\text{LDATA6}}$	$\overline{\text{LDATA5}}$	$\overline{\text{LDATA4}}$	$\overline{\text{LDATA3}}$	$\overline{\text{LDATA2}}$	$\overline{\text{LDATA1}}$	$\overline{\text{LDATA0}}$
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – $\overline{\text{LDATA}}$ LIN Data In / Data out
In LIN mode: FIFO data buffer port.

In UART mode: data register (no data buffer - no FIFO).

- In Write access, data out
- In Read access, data in

23. Analog-to-Digital Converter (ADC)

23.1 Features

- 10-bit Resolution
- 0.5 LSB Integral Non-Linearity
- ±2 LSB Absolute Accuracy
- 8 - 250µs Conversion Time
- Up to 120kSPS at Maximum Resolution
- 11 Multiplexed Single Ended Input Channels
- Three differential input channels with accurate (5%) programmable gain 5, 10, 20 and 40
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC start conversion by auto triggering on interrupt sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler
- Temperature Sensor
- LiN address sense (ISRC voltage measurement)
- VCC voltage measurement

23.2 Overview

The device features a 10-bit successive approximation ADC. The ADC is connected to a 15-channel Analog Multiplexer which allows 11 single-ended voltage inputs. The single-ended voltage inputs refer to 0V (GND).

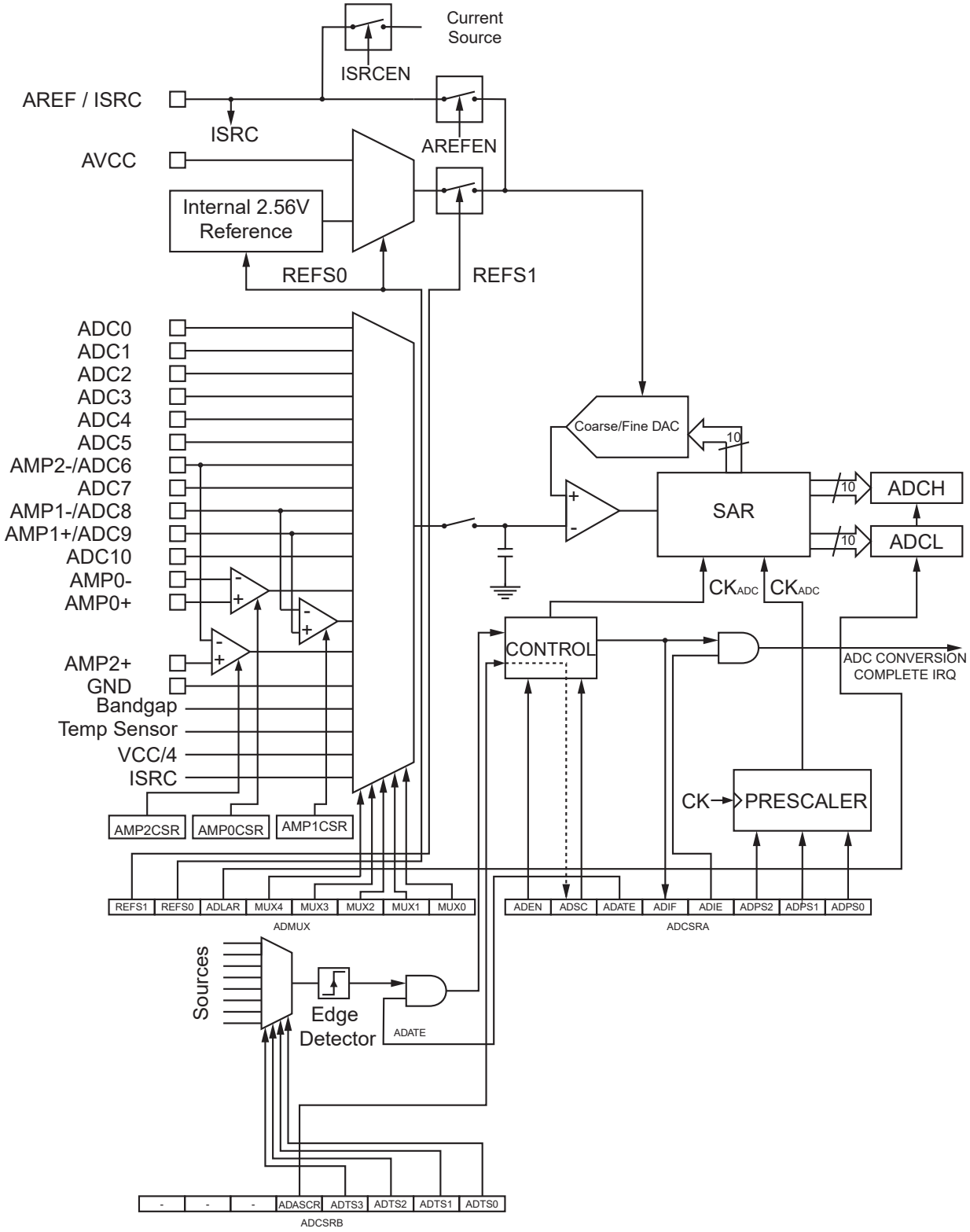
The device also supports three differential voltage input amplifiers which are equipped with a programmable gain stage, providing amplification steps of 14dB (5×), 20dB (10×), 26dB (20×), or 32dB (40×) on the differential input voltage before the A/D conversion. On the amplified channels, 8-bit resolution can be expected.

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, AVCC. 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 (PRR.PRADC) must be written to '0' in order to be enable the ADC.

Figure 23-1. Analog to Digital Converter Block Schematic Operation



Related Links

Power Management and Sleep Modes
Power Reduction Register

23.3 Description

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, AVCC 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 may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel are selected by writing to the MUX bits in ADMUX. 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 setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference is set by the REFS1 and REFS0 bits in ADMUX register, whatever the ADC is enabled or not. 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.

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 ADLAR bit in ADMUX.

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

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

23.4 Starting a Conversion

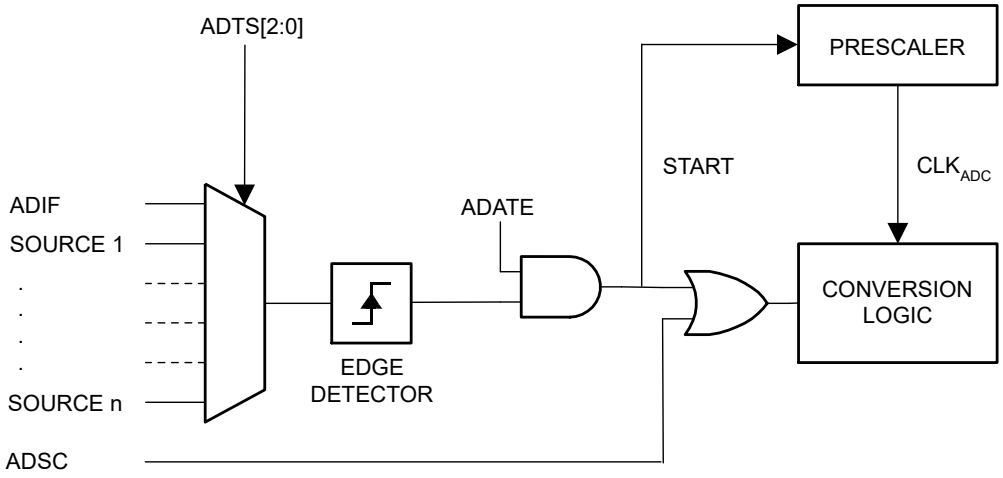
A single conversion is started by writing a '0' to the Power Reduction ADC bit in the Power Reduction Register (PRR.PRADC), and writing a '1' to the ADC Start Conversion bit in the ADC Control and Status Register A (ADCSRA.ADSC). ADCS will stay high as long as the conversion is in progress, and will 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 23-2. ADC Auto Trigger Logic

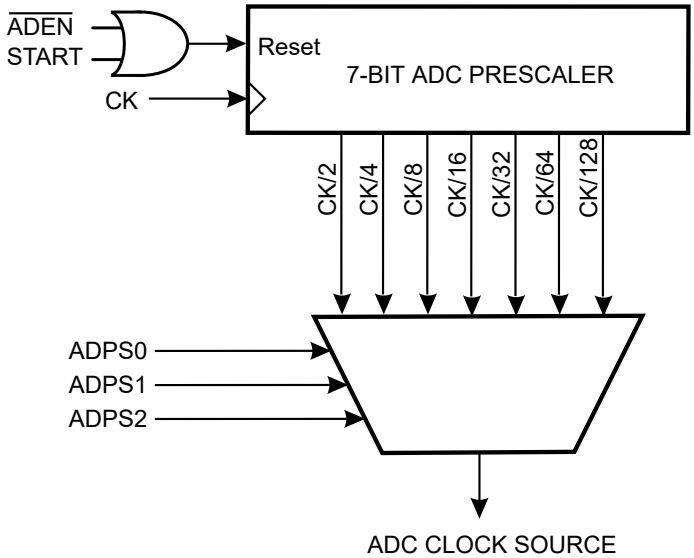


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a '1' to ADCSRA.ADSC. In this mode, the ADC will perform successive conversions independently of whether the ADC Interrupt Flag (ADIF) is cleared or not.The free running mode is not allowed on the amplified channels.

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.

23.5 Prescaling and Conversion Timing

Figure 23-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 2MHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 2MHz 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 3.5 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 23-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

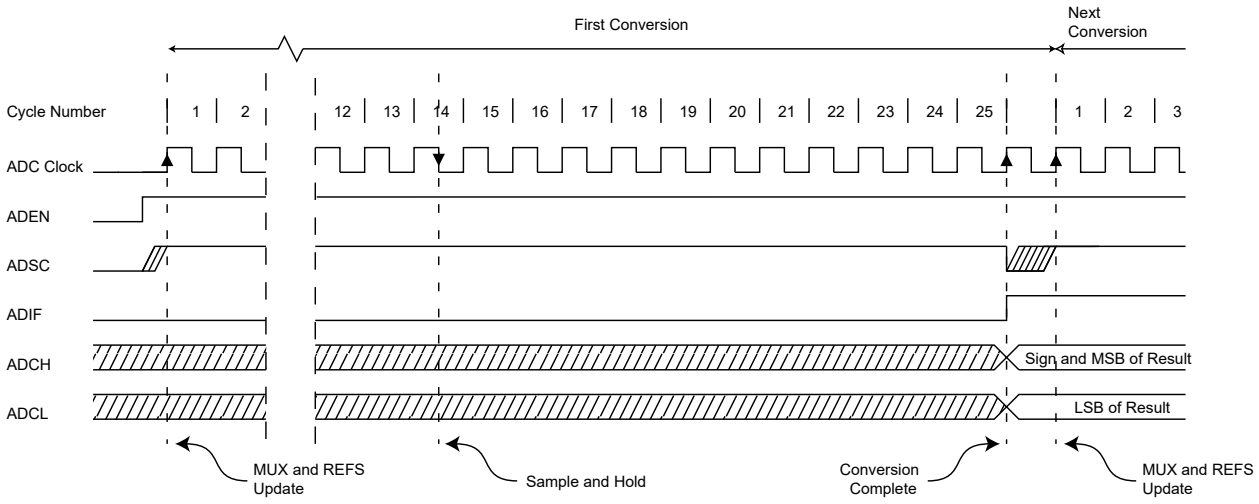


Figure 23-5. ADC Timing Diagram, Single Conversion

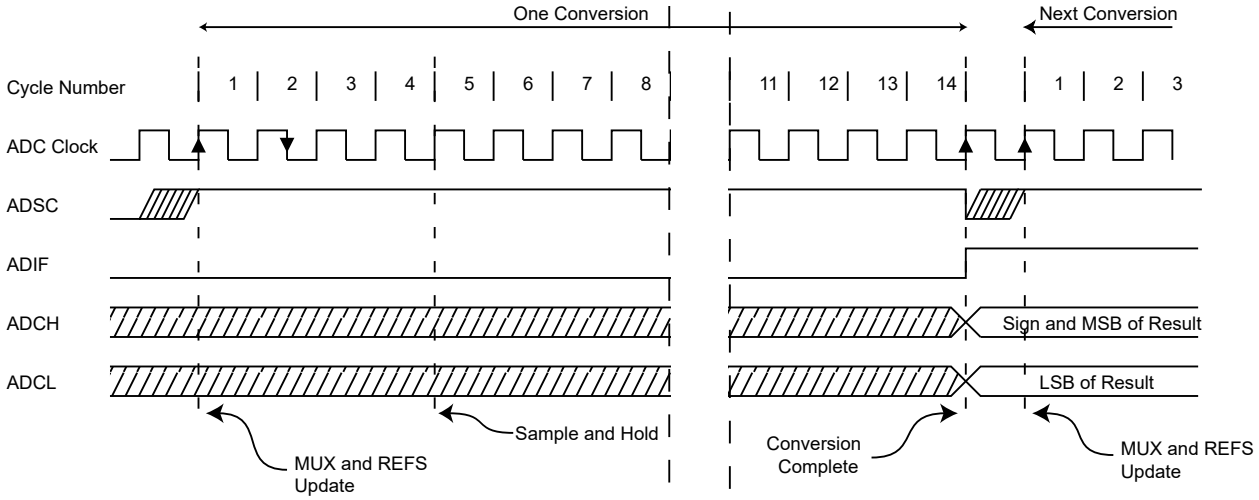


Figure 23-6. ADC Timing Diagram, Auto Triggered Conversion

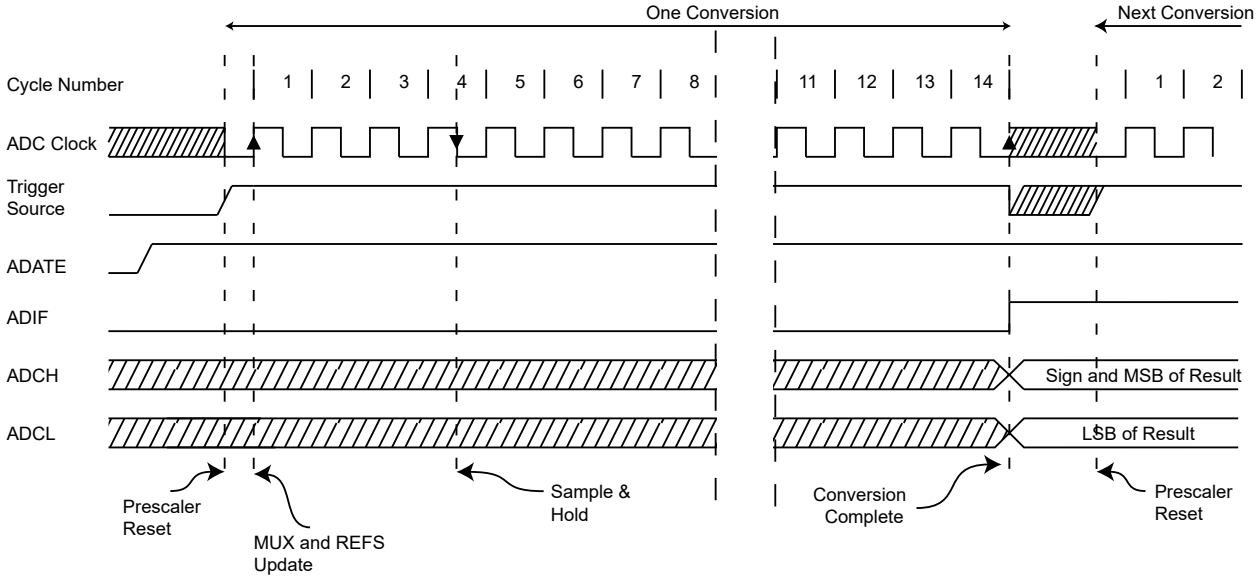


Figure 23-7. ADC Timing Diagram, Free Running Conversion

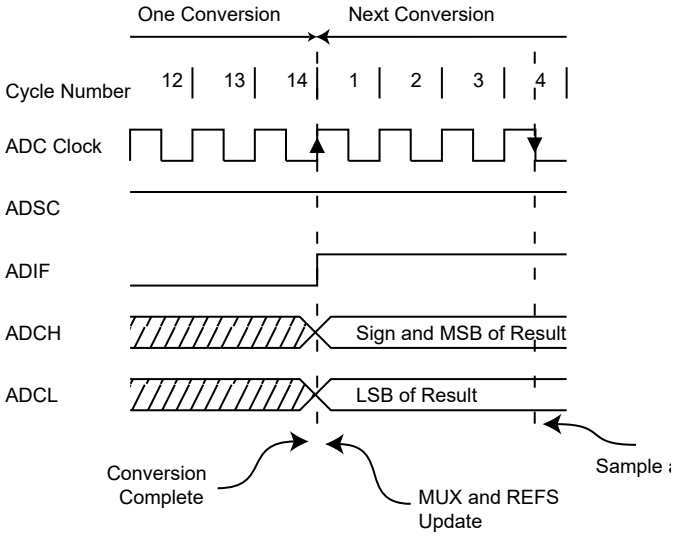


Table 23-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	3.5	15.5
Auto Triggered conversions	2	16

23.6 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 and ADMUX.REFS) 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 bits in ADMUX).

23.6.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

- In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing 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.
- In Free Running mode, because the amplifier clear the ADSC bit at the end of an amplified conversion, it is not possible to use the free running mode, unless ADSC bit is set again by soft at the end of each conversion.

The user is advised not to write new channel or reference selection values during the 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.

23.6.2 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 to 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.

AREF pin is alternate function with ISRC Current Source output. When current source is selected, the AREF pin is not connected to the internal reference voltage network. See AREFEN and ISRCEN bits in ADCSRB – ADC Control and Status Register B.

If differential channels are used, the selected reference should not be closer to AV_{CC} than indicated in ADC Characteristics of Electrical Characteristics chapter.

23.7 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.

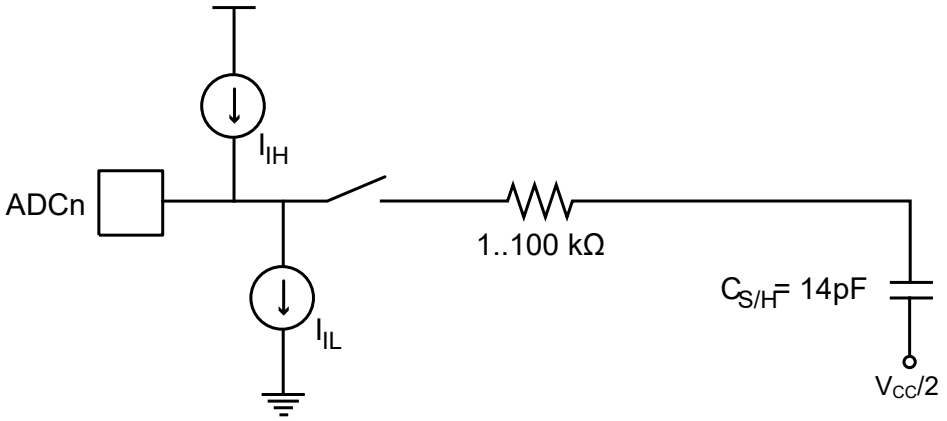
23.7.1 Analog Input Circuitry

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 of a time the source needs to charge the S/H capacitor, which can vary widely. It is recommended to use only 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 23-8. Analog Input Circuitry

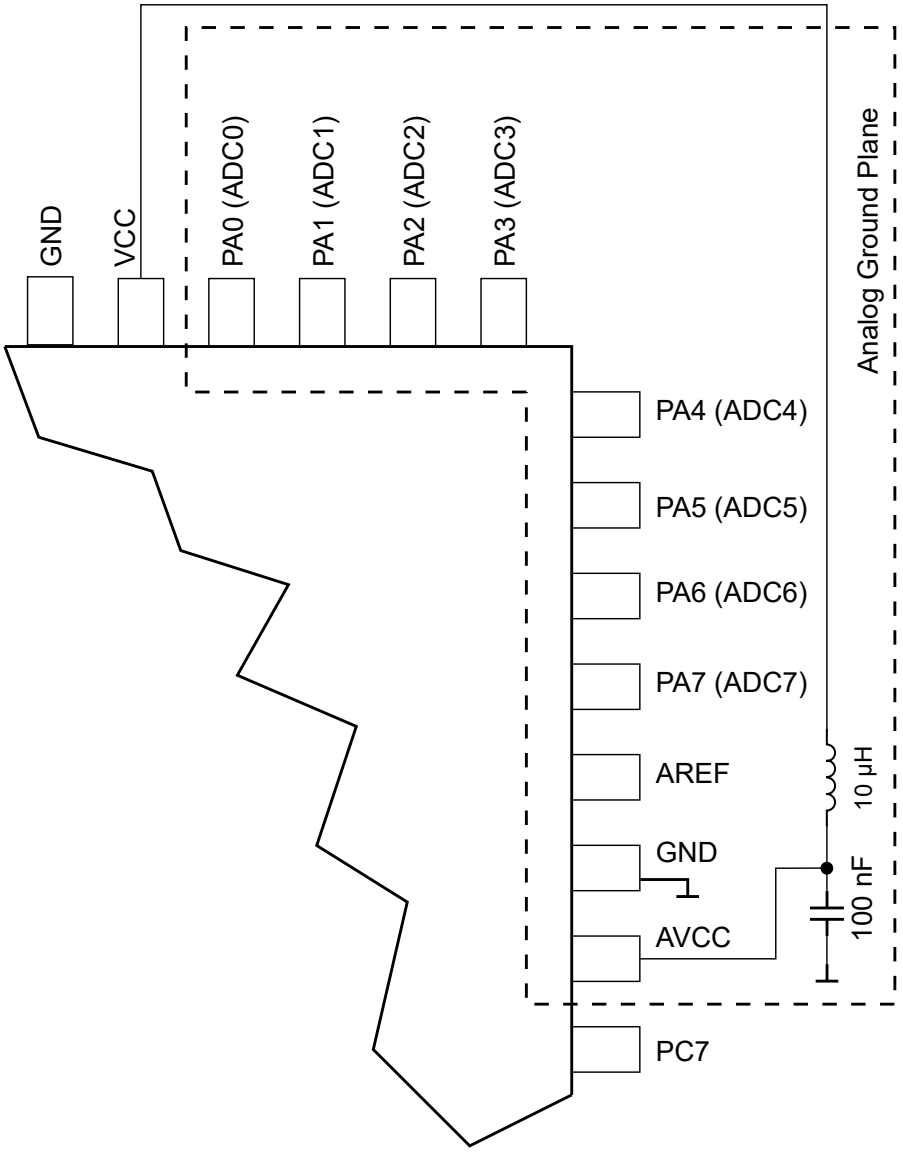


23.7.2 Analog Noise Canceling Techniques

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 AV_{CC} 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 23-9. ADC Power Connections



Note: If the resistivity in the inductor is too high, the AV_{CC} may exceed its range, V_{CC} - 0.3V < AV_{CC} < V_{CC} + 0.3V

23.7.3 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.

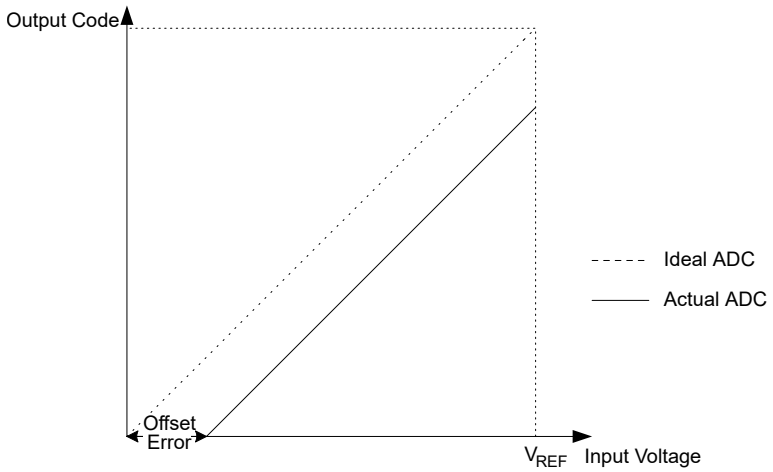
23.7.4 ADC Accuracy Definitions

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:

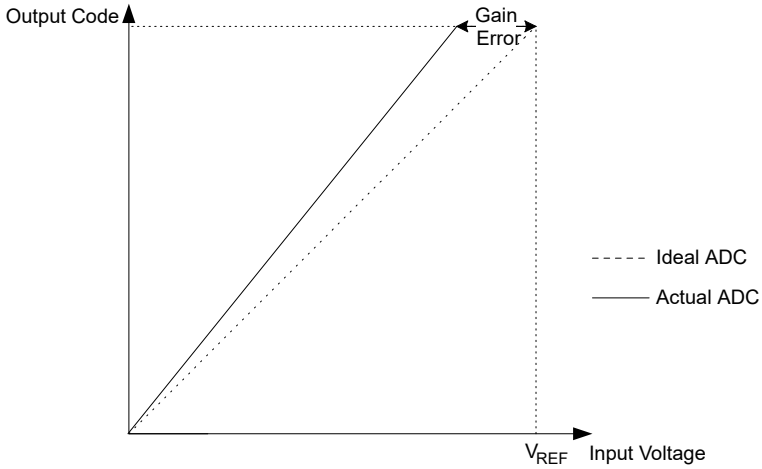
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 23-10. Offset Error



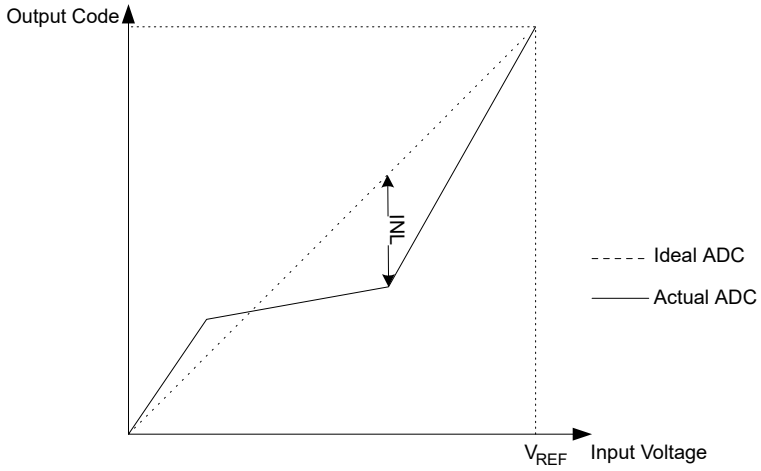
- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

Figure 23-11. Gain Error



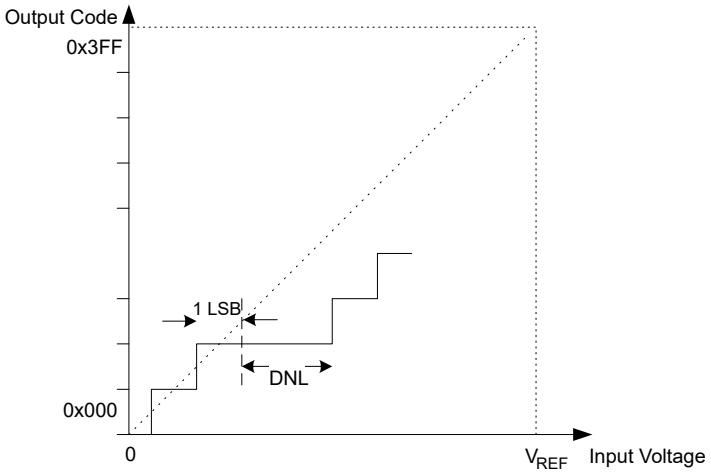
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 23-12. 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 23-13. 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.

23.8 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

$$\mathrm{ADC} = \frac{V_{\mathrm{IN}} \cdot 1024}{V_{\mathrm{REF}}}$$

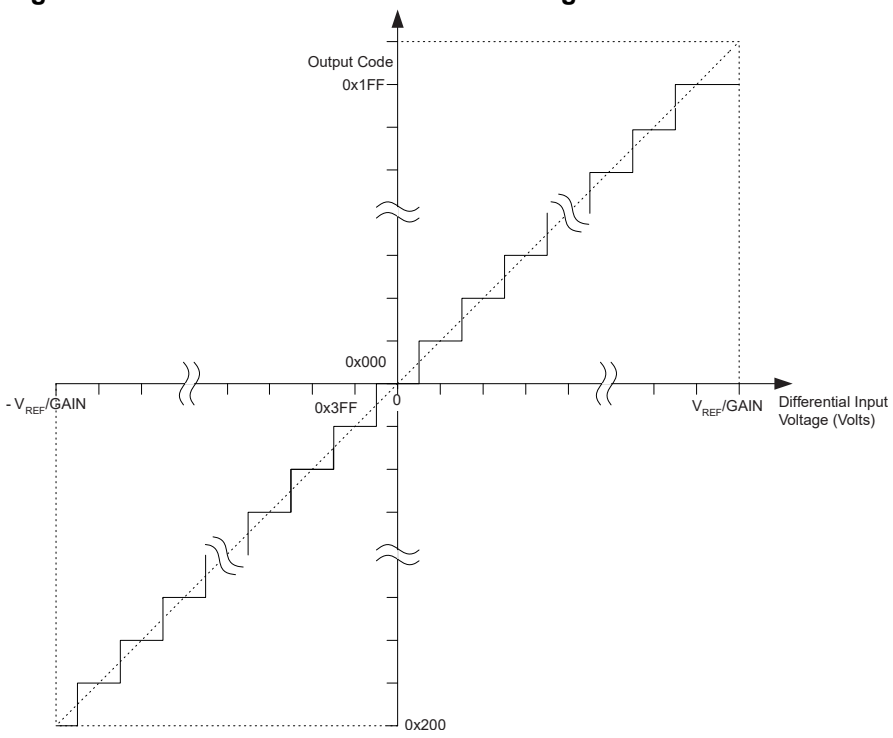
where V_{IN} is the voltage on the selected input pin, and V_{REF} the selected voltage reference (see also descriptions of ADMUX.REFSn and ADMUX.MUX). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is

$$\mathrm{ADC} = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{\mathrm{REF}}}$$

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 23-14. 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 23-2. 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_{ADCn}	Read Code	Corresponding Decimal Value
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

23.9 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended Temperature sensor channel. Selecting the Temperature sensor channel by writing ADMUX.MUX[4:0] to '10000' enables the temperature sensor. The internal 2.56V voltage reference must also be selected for the ADC voltage reference source in the temperature sensor measurement. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor.

The measured voltage has a linear relationship to the temperature as described in the following table. The voltage sensitivity is approximately 2.5mV/°C and the accuracy depends on the method of user calibration. Typically, the measurement accuracy after a single temperature calibration is ±10°C, assuming calibration at room temperature. Better accuracies are achieved by using two temperature points for calibration.

Table 23-3. Temperature vs. Sensor Output Voltage (Typical Case)

Temperature	-40°C	+25°C	+85°C	+125°C
Voltage	600mV	762mV	912mV	1012mV

The values described in the table above are typical values. However, due to process variations the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results the temperature measurement can be calibrated in the application software. The software calibration requires that a calibration value is measured and stored in a register or EEPROM for each chip, as a part of the production test. The software calibration can be done utilizing the formula:

$T = k \times [(ADCH << 8) \mid ADCL] - T_{OS}$

where ADCH and ADCL are the ADC data registers, T is temperature in Kelvin, *k* is a fixed coefficient and T_{OS} is the temperature sensor offset value determined and stored into EEPROM as a part of the production test. Typically, k is very close to 1.0 and in single-point calibration the coefficient may be omitted. Where higher accuracy is required the slope coefficient should be evaluated based on measurements at two temperatures.

23.9.1 User Calibration

The software calibration requires that a calibration value is measured and stored in a register or EEPROM for each chip. The software calibration can be done utilizing the formula:

$T = \{[(ADCH << 8) \mid ADCL] - T_{OS}\}/k$

where ADCH and ADCL are the ADC data registers, k is a fixed coefficient and T_{OS} is the temperature sensor offset value determined and stored into EEPROM.

23.9.2 Manufacturing Calibration

One can also use the calibration values available in the signature row.

The calibration values are determined from values measured during test at hot temperature which is approximately +85°C.

The temperature in Celsius degrees can be calculated utilizing the formula:

T = {[(ADCH << 8) | ADCL] × TSGAIN} + TSOFFSET - 273

Where:

- ADCH and ADCL are the ADC data registers.
- TSGAIN is the temperature sensor gain (constant 1, or unsigned fixed point number, 0x80 = decimal 1.0).
- TSOFFSET is the temperature sensor offset correction term (2. complement signed byte).

23.10 Amplifier

The ATmegaS64M1 features three differential amplified channels with programmable 5, 10, 20, and 40 gain stage.

Because the amplifiers are switching capacitor amplifiers, they need to be clocked by a synchronization signal called in this document the amplifier synchronization clock. To ensure an accurate result, the amplifier input needs to have a quite stable input value during at least four Amplifier synchronization clock periods.

To ensure an accurate result, the amplifier input needs to have a quite stable input value at the sampling point during at least four amplifier synchronization clock periods.

Amplified conversions can be synchronized to PSC events or to the internal clock CK_{ADC} equal to eighth the ADC clock frequency. In case the synchronization is done the ADC clock divided by 8, 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.

The normal way to use the amplifier is to select a synchronization clock via the AMPxTS1:0 bits in the AMPxCSR register. Then the amplifier can be switched on, and the amplification is done on each synchronization event.

In order to start an amplified Analog to Digital Conversion on the amplified channel, the ADMUX must be configured as specified on .

The ADC starting requirement is done by setting the ADSC bit of the ADCSRA Register.

Until the conversion is not achieved, it is not possible to start a conversion on another channel.

In order to have a better understanding of the functioning of the amplifier synchronization, two timing diagram examples are shown below.

As soon as a conversion is requested thanks to the ADSC bit, the Analog to Digital Conversion is started. In case the amplifier output is modified during the sample phase of the ADC, the on-going conversion is aborted and restarted as soon as the output of the amplifier is stable. This ensure a fast response time. The only precaution to take is to be sure that the trig signal (PSC) frequency is lower than ADCclk/4.

Figure 23-15. Amplifier synchronization timing diagram. With change on analog input signal

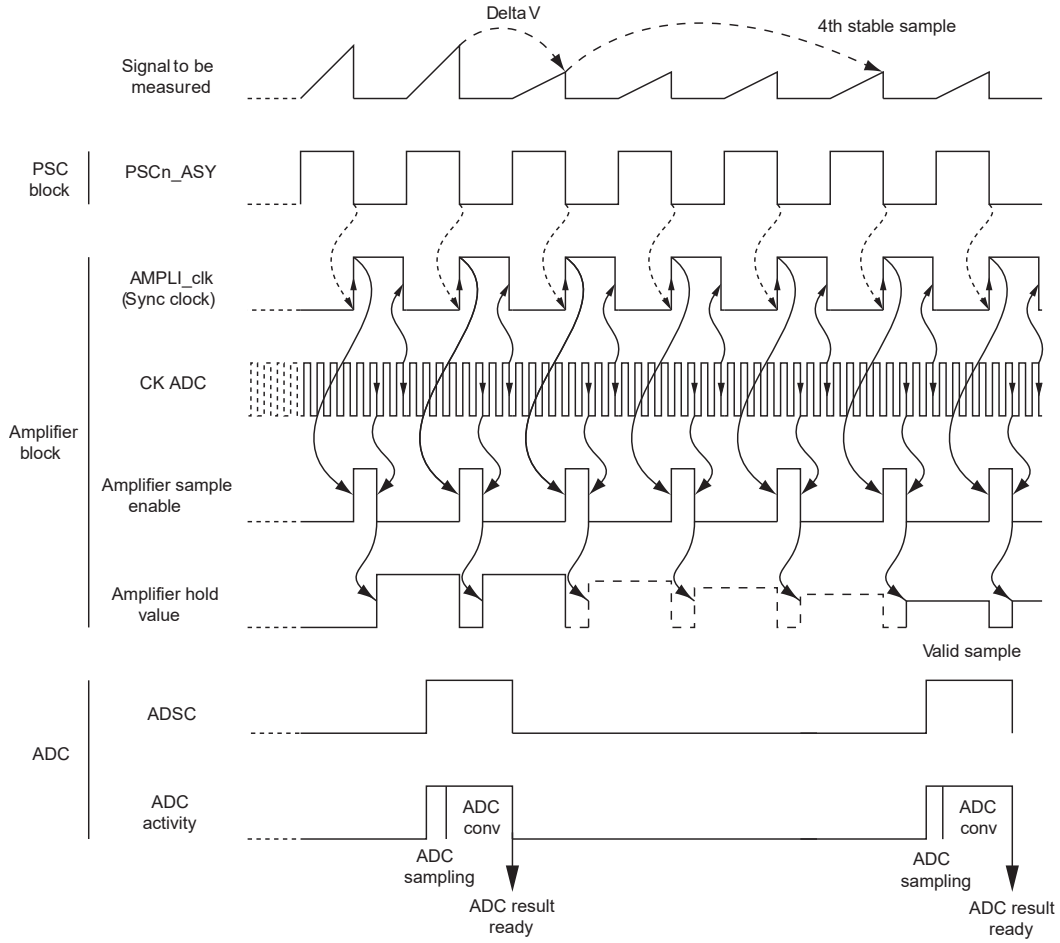
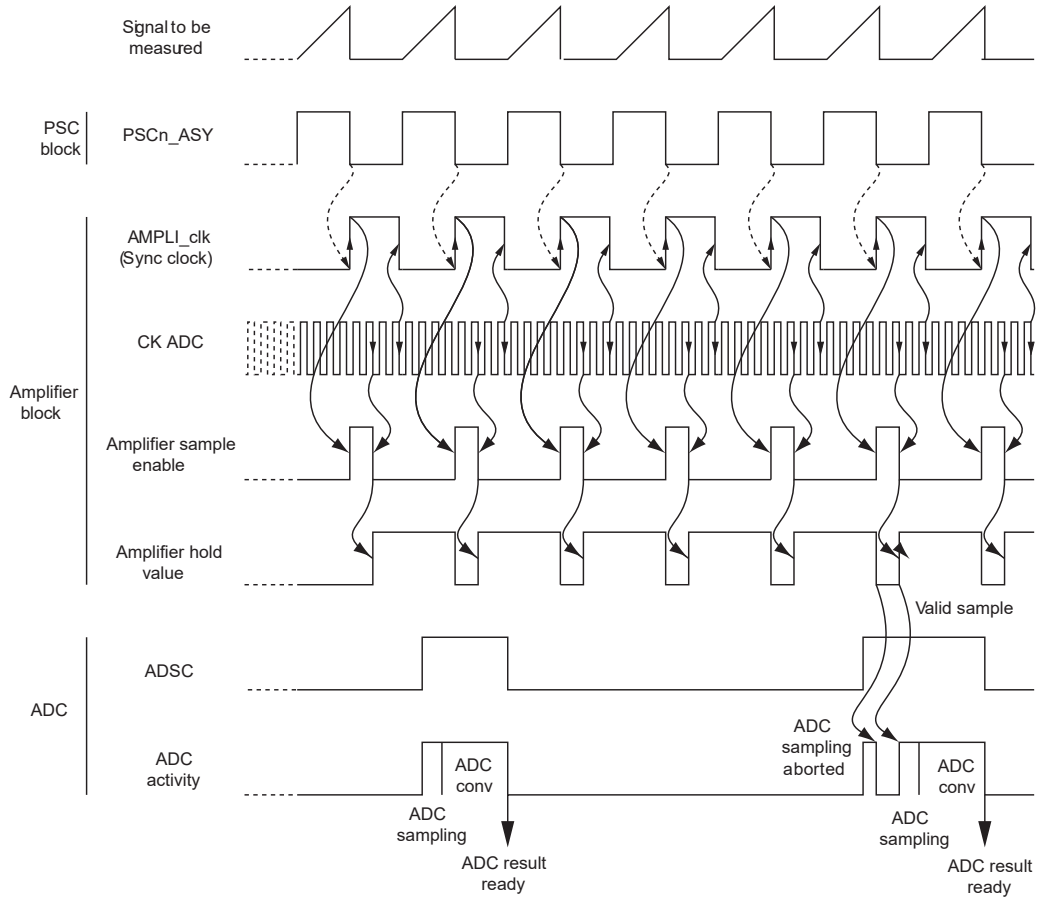


Figure 23-16. Amplifier synchronization timing diagram. ADSC is set when the amplifier output is changing due to the amplifier clock switch.

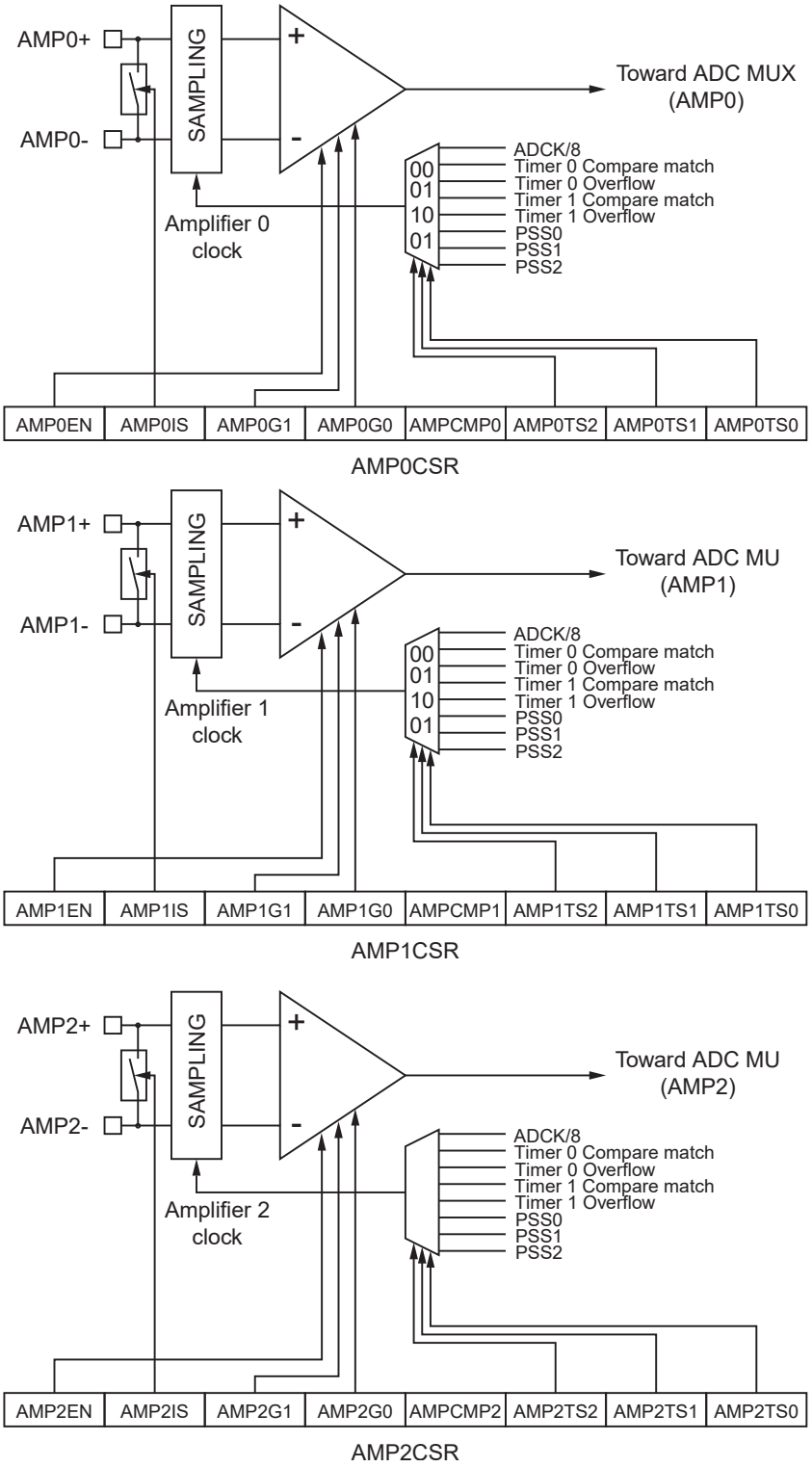


In order to have a better understanding of the functioning of the amplifier synchronization, a timing diagram example is shown in the figure above.

It is also possible to auto trigger conversion on the amplified channel. In this case, the conversion is started at the next amplifier clock event following the last auto trigger event selected thanks to the ADTS bits in the ADCSRB register. In auto trigger conversion, the free running mode is not possible unless the ADSC bit in ADCSRA is set by soft after each conversion.

The block diagram of the two amplifiers is shown below.

Figure 23-17. Amplifiers block diagram



23.11 Register Description

23.11.1 ADC Multiplexer Selection Register

Name: ADMUX
Offset: 0x7C
Reset: 0x00
Property: R/W

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] ADC Reference Selection Bits
These bits select the voltage reference for the ADC. If these bits are changed during a conversion, the change does not go into effect until this conversion is complete (ADCSRA.ADIF is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 23-4. ADC Voltage Reference Selection

AREFEN	ISRCEN	REFS[1:0]	Voltage Reference Selection
1	0	00	External V_{REF} on AREF pin; Internal V_{REF} is switched off
1	0	01	AV_{CC} with external capacitor connected on the AREF pin
0	0	01	AV_{CC} (no external capacitor connected on the AREF pin)
1	0	10	Reserved
1	0	11	Internal 2.56V reference voltage with external capacitor connected on the AREF pin
0	x	11	Internal 2.56V reference voltage

If bits REFS1 and REFS0 are changed during a conversion, the change does not take effect until this conversion is completed (i.e., ADCSRA.ADIF is set). In case the internal V_{REF} is selected, it is turned ON as soon as an analog feature needing it is set.

Bit 5 – ADLAR ADC Left Adjust Result
The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data register. Writing a '1' to ADLAR left-adjusts the result. Otherwise, the result is right-adjusted. Changing the ADLAR bit

affects the ADC Data register immediately, regardless of any ongoing conversions. For a complete description of this bit, see ADCL and ADCH (ADLAR = 0).

Bits 4:0 – MUX[4:0] ADC Channel Selection Bits
These four bits determine which analog inputs are connected to the ADC input. If these bits are changed during a conversion, the change does not go into effect until this conversion is completed. (ADCSRA.ADIF is set).

Table 23-5. ADC Input Channel Selection

MUX[4:0]	Single Ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7
01000	ADC8
01001	ADC9
01010	ADC10
01011	Temperature sensor
01100	VCC/4
01101	ISRC
01110	AMP0
01111	AMP1 (- is ADC8, + is ADC9)
10000	AMP2 (- is ADC6)
10001	Bandgap
10010	GND
10011	Reserved
101xx	Reserved
1xxxx	Reserved

23.11.2 ADC Control and Status Register A

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: r/w

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 take effect at the end of the 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 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. Clear it to return in single conversion mode. 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 23-6. Input Channel Selection

ADPS[2:0]	Division Factor
000	2
001	2

ADPS[2:0]	Division Factor
010	4
011	8
100	16
101	32
110	64
111	128

23.11.3 ADC Control and Status Register B

Name: ADCSRB
Offset: 0x7B
Reset: 0x00
Property: R/W

Bit	7	6	5	4	3	2	1	0
	ADHSM	ISRCEN	AREFEN		ADTS[3:0]			
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 – ADHSM ADC High Speed Mode
Writing this bit to one enables the ADC High Speed mode. Set this bit if you wish to convert with an ADC clock frequency higher than 200kHz.

Bit 6 – ISRCEN Current Source Enable
Set this bit to source a 100µA current to the AREF pin. Clear this bit to use AREF pin as Analog Reference pin.

Bit 5 – AREFEN Analog Reference pin Enable
Set this bit to connect the internal AREF circuit to the AREF pin. Clear this bit to disconnect the internal AREF circuit from the AREF pin.

Bits 3:0 – ADTS[3:0] ADC Auto Trigger Source Selection Bits
These bits are only necessary in case the ADC works in auto trigger mode. It means if ADATE bit in ADCSRA register is set.

These three bits select the interrupt event which will generate the trigger of the start of conversion. The start of conversion will be generated by the rising edge of the selected interrupt flag whether the interrupt is enabled or not. In case of trig on PSCnASY event, there is no flag. So in this case a conversion will start each time the trig event appears and the previous conversion is completed.

Value	Description
0000	Free Running mode
0001	External interrupt request 0
0010	Timer/Counter0 compare match
0011	Timer/Counter0 overflow
0100	Timer/Counter1 compare Match B
0101	Timer/Counter1 overflow
0110	Timer/Counter1 capture event
0111	PSC Module 0 synchronization signal
1000	PSC Module 1 synchronization signal
1001	PSC Module 2 synchronization signal
1010	Analog comparator 0
1011	Analog comparator 1
1100	Analog comparator 2
1101	Analog comparator 3

Value	Description
1110	Reserved
1111	Reserved

23.11.4 ADC Data Register Low and High Byte (ADLAR=0)

Name: ADCL and ADCH (ADLAR = 0)
Offset: 0x78
Reset: 0x00

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 Timer/Counter 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.

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.

23.11.5 ADC Data Register Low and High Byte (ADLAR=1)

Name: ADCL and ADCH (ADLAR = 1)
Offset: 0x78
Reset: 0x00

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 Timer/Counter 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.

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.

23.11.6 Digital Input Disable Register 0

Name: DIDR0
Offset: 0x7E
Reset: 0x00
Property: -

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.

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 – ADCD ADC Digital Input Disable

23.11.7 Digital Input Disable Register 1

Name: DIDR1
Offset: 0x7F
Reset: 0x00
Property: -

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 an analog 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.

Bit	7	6	5	4	3	2	1	0
						ADC10D	ADC9D	ADC8D
Access						R/W	R/W	R/W
Reset						0	0	0

Bits 0, 1, 2 – ADCD ADC Digital Input Disable

23.11.8 Amplifier n Control and Status Register

Name: AMPnCSR
Offset: 0x75 + n*0x01 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	AMPnEN	AMPnIS	AMPnG[1:0]		AMPCMPn	AMPnTS[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 – AMPnEN Amplifier n Enable
Set this bit to enable the Amplifier n. Clear this bit to disable the Amplifier n. Clearing this bit while a conversion is running takes effect at the end of the conversion.

WARNING Always clear AMPnTS0:1 when clearing AMPnEN.

Bit 6 – AMPnIS Amplifier n Input Shunt
Set this bit to short-circuit the Amplifier n input. Clear this bit to use the Amplifier n normally.

Bits 5:4 – AMPnG[1:0] Amplifier n Gain Selection
These bits determine the gain of the amplifier n. To ensure an accurate result, after the gain value has been changed, the amplifier input value must be stable input during at least four amplifier synchronization clock periods.

Value	Description
00	Gain 5
01	Gain 10
10	Gain 20
11	Gain 40

Bit 3 – AMPCMPn Amplifier n - Comparator n connection
Set this bit to connect the amplifier n to the comparator n positive input. In this configuration, the comparator clock is adapted to the amplifier clock and AMPnTS[2:0] bits have no effect. Clear this bit to use the Amplifier n normally.

Bits 2:0 – AMPnTS[2:0] Amplifier n Clock Source Selection
These bits select the event which generates the clock for the amplifier n. This clock source is necessary to start the conversion on the amplified channel.

Value	Description
000	ADC Clock/8
001	Timer/Counter0 compare match
010	Timer/Counter0 overflow
011	Timer/Counter1 compare Match B

Value	Description
100	Timer/Counter1 overflow
101	PSC Module 0 synchronization signal (PSS0)
110	PSC Module 1 synchronization signal (PSS1)
111	PSC Module 2 synchronization signal (PSS2)

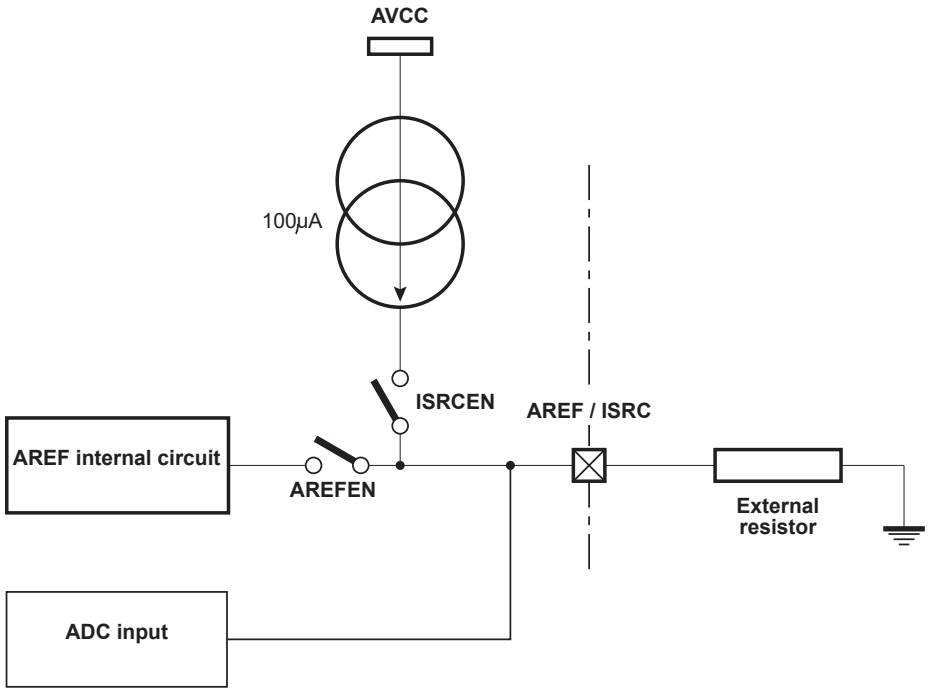
24. ISRC - Current Source

24.1 Features

- 100µA constant current source
- ±2% absolute accuracy

The ATmegaS64M1 features a 100µA ±2% current source. After RESET or up on request, the current is flowing through an external resistor. The voltage can be measured on the dedicated pin shared with the ADC. Using a resistor in series with a ≤0.5% tolerance is recommended. To protect the device against big values, the ADC must be configured with AV_{CC} as internal reference to perform the first measurement. Afterwards, another internal reference can be chosen according to the previous measured value to refine the result. When ISRCEN bit is set, the ISRC pin sources 100µA. Otherwise this pin keeps its initial function.

Figure 24-1. Current source block diagram



24.2 Typical Applications

24.2.1 LIN Current Source

During the configuration of a LIN node in a cluster, it may be necessary to attribute dynamically a unique physical address to every cluster node. The LIN protocol does not describe a procedure for this.

The current source is an effective way to associate a physical address to the application supported by the LIN node. A full dynamic node configuration can be used to set up the LIN nodes in a cluster.

The ATmegaS64M1 uses an external resistor in conjunction with the current source. The device measures the voltage to the boundaries of the resistance via the Analog-to-Digital Converter (ADC). The

resulting voltage defines the physical address that the communication handler uses when the node participates in LIN communication.

In applications where the distributed voltages may be very disturbed, the internal current source solution immunizes the address detection against any kind of voltage variations.

Table 24-1. Example of Resistor Values (±5%) for an 8-address System (AV_{CC} = 3.3V)

Physical Address	Resistor Value R_{LOAD} [Ohm] (1)	Minimum Voltage Measured [V]	Typical Voltage Measured [V]	Maximum Voltage Measured [V]
0	1 000	–	0.1	–
1	2 200	–	0.22	–
2	3 300	–	0.33	–
3	4 700	–	0.47	–
4	6 800	–	0.68	–
5	10 000	–	1	–
6	15 000	–	1.5	–
7	22 000	–	2.2	–

Table 24-2. Example of Resistor Values (±1%) for a 16-address System (AV_{CC} = 3.3V)

Physical Address	Resistor Value R_{LOAD} [Ohm] (1)	Minimum Voltage Measured [V]	Typical Voltage Measured [V]	Maximum Voltage Measured [V]
0	2 000	–	0.2	–
1	2 400	–	0.24	–
2	2 700	–	0.27	–
3	3 300	–	0.33	–
4	3 900	–	0.39	–
5	4 700	–	0.47	–
6	5 600	–	0.56	–
7	6 800	–	0.68	–
8	8 200	–	0.82	–
9	9 100	–	0.91	–
10	11 000	–	1.1	–
11	13 000	–	1.3	–
12	15 000	–	1.5	–

Note:

1. 3V range: Max. R_{LOAD} 15KΩ.

24.2.2 Voltage Reference for External Devices

An external resistor used in conjunction with the current source can be used as voltage reference for external devices. Using a resistor in series with a lower tolerance than the Current Source accuracy (≤2%) is recommended. The tables above give examples of voltage references using standard values of resistors.

24.2.3 Threshold Reference for Internal Analog Comparator

An external resistor used in conjunction with the current source can be used as threshold reference for the internal Analog Comparator. This can be connected to AIN0 (negative Analog Compare input pin) as well as AIN1 (positive Analog Compare input pin). Using a resistor in series with a lower tolerance than the Current Source accuracy (≤2%) is recommended. The tables above give examples of threshold references using standard values of resistors.

24.3 Register Description

24.3.1 ADC Control and Status Register B

Name: ADCSRB
Offset: 0x7B
Reset: 0x00
Property: R/W

Bit	7	6	5	4	3	2	1	0
	ADHSM	ISRCEN	AREFEN		ADTS[3:0]			
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 – ADHSM ADC High Speed Mode
Writing this bit to one enables the ADC High Speed mode. Set this bit if you wish to convert with an ADC clock frequency higher than 200kHz.

Bit 6 – ISRCEN Current Source Enable
Set this bit to source a 100µA current to the AREF pin. Clear this bit to use AREF pin as Analog Reference pin.

Bit 5 – AREFEN Analog Reference pin Enable
Set this bit to connect the internal AREF circuit to the AREF pin. Clear this bit to disconnect the internal AREF circuit from the AREF pin.

Bits 3:0 – ADTS[3:0] ADC Auto Trigger Source Selection Bits
These bits are only necessary in case the ADC works in auto trigger mode. It means if ADATE bit in ADCSRA register is set.

These three bits select the interrupt event which will generate the trigger of the start of conversion. The start of conversion will be generated by the rising edge of the selected interrupt flag whether the interrupt is enabled or not. In case of trig on PSCnASY event, there is no flag. So in this case a conversion will start each time the trig event appears and the previous conversion is completed.

Value	Description
0000	Free Running mode
0001	External interrupt request 0
0010	Timer/Counter0 compare match
0011	Timer/Counter0 overflow
0100	Timer/Counter1 compare Match B
0101	Timer/Counter1 overflow
0110	Timer/Counter1 capture event
0111	PSC Module 0 synchronization signal
1000	PSC Module 1 synchronization signal
1001	PSC Module 2 synchronization signal
1010	Analog comparator 0
1011	Analog comparator 1
1100	Analog comparator 2
1101	Analog comparator 3

Value	Description
1110	Reserved
1111	Reserved

25. AC – analog comparator

25.1 Features

- Four analog comparators
- High speed clocked comparators
- ±30mV hysteresis
- Four reference levels
- Generating configurable interrupts

25.2 Overview

The ATmegaS64M1 features four fast analog comparators. The Analog Comparator compares the input values on the positive pin ACMPx and negative pin ACMPM or ACMPMx.

Each comparator has a dedicated input on the positive input, and the negative input of each comparator can be configured as:

- a steady value among the four internal reference levels defined by the V_{REF} selected thanks to the REFS1:0 bits in ADMUX register
- a value generated from the internal DAC
- an external analog input ACMPMx

When the voltage on the positive ACMPn pin is higher than the voltage selected by the ACnM multiplexer on the negative input, the Analog Comparator output, ACnO, is set.

The comparator is a clocked comparator. A new comparison is done on the falling edge of $CLK_{I/O}$ or $CLK_{I/O}/2$. Depending on ACCKDIV fit of ACSR register, see ACSR.

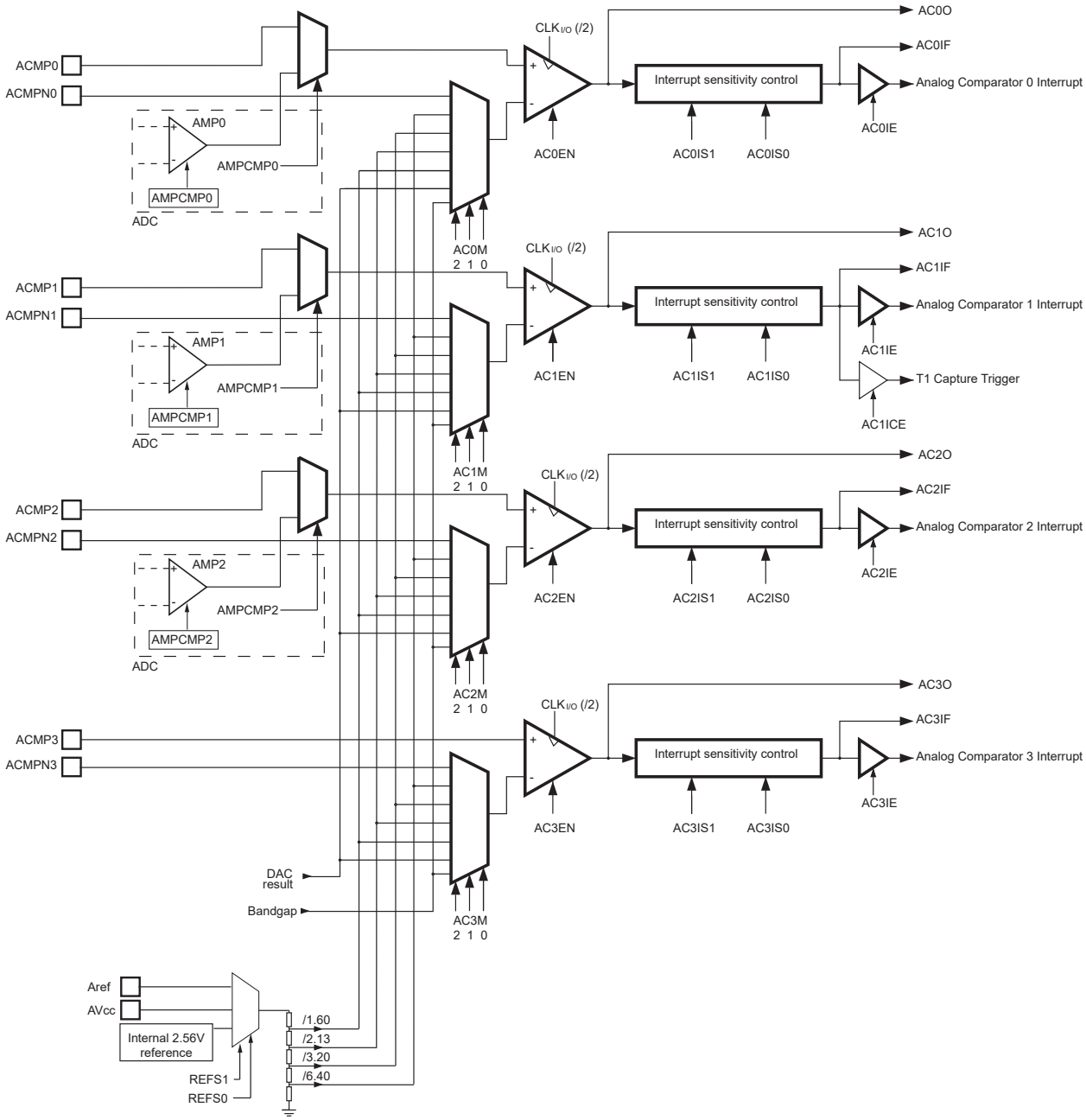
Each comparator can trigger a separate interrupt, exclusive to the Analog Comparator. In addition, the user can select Interrupt triggering on comparator output rise, fall or toggle.

The interrupt flags can also be used to synchronize ADC or DAC conversions.

Moreover, the comparator's output of the Comparator 1 can be set to trigger the Timer/Counter1 Input Capture function.

A block diagram of the four comparators and their surrounding logic is shown below.

Figure 25-1. Analog comparator block diagram



Note: ADC multiplexer output: see teh ADMUX register.

Note: Refer to the Pin Configuration for the analog comparator pin placement.

Note: The voltage on V_{REF} is defined in the ADMUX register.

Related Links

ADMUX

25.3 Use of ADC amplifiers

Thanks to AMPCMP0 configuration bit, Comparator 0 positive input can be connected to Amplifier O output. In that case, the clock of Comparator 0 is adapted to the Amplifier 0 clock.

Thanks to AMPCMP1 configuration bit, Comparator 1 positive input can be connected to Amplifier 1 output. In that case, the clock of Comparator 1 is adapted to the Amplifier 1 clock.

Thanks to AMPCMP2 configuration bit, Comparator 2 positive input can be connected to Amplifier 2 output. In that case, the clock of Comparator 2 is adapted to the Amplifier 2 clock.

Related Links

AMPnCSR

25.4 Register Description

25.4.1 Analog Comparator 0 Control Register

Name: AC0CON
Offset: 0x94
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	AC0EN	AC0IE	AC0IS[1:0]		ACCKSEL	AC0M[2:0]		
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – AC0EN Analog Comparator 0 Enable Bit

Value	Description
1	Enable the Analog Comparator 0.
0	Disable the Analog Comparator 0.

Bit 6 – AC0IE Analog Comparator 0 Interrupt Enable bit

Value	Description
1	Enable the Analog Comparator 0 interrupt.
0	Disable the Analog Comparator 0 interrupt.

Bits 5:4 – AC0IS[1:0] Analog Comparator 0 Interrupt Select bit
These two bits determine the sensitivity of the interrupt trigger.

Value	Description
00	Comparator Interrupt on output toggle
01	Reserved
10	Comparator interrupt on output falling edge
11	Comparator interrupt on output rising edge

Bit 3 – ACCKSEL Analog Comparator Clock Select

Value	Description
1	Use the PLL output as comparator clock.
0	use the CLK_{IO} as comparator clock.

Bits 2:0 – AC0M[2:0] Analog Comparator 0 Multiplexer register
These three bits determine the input of the negative input of the analog comparator.

Value	Description
000	V_{REF}/6.40
001	V_{REF}/3.20
010	V_{REF}/2.13
011	V_{REF}/1.60
100	Bandgap (1.1V)
101	DAC result

Value	Description
110	Analog comparator negative input (ACMPM pin)
111	Reserved

25.4.2 Analog Comparator 1 Control Register

Name: AC1CON
Offset: 0x95
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	AC1EN	AC1IE	AC1IS[1:0]		AC1ICE	AC1M[2:0]		
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – AC1EN Analog Comparator 1 Enable Bit

Value	Description
1	Enable the Analog Comparator 1.
0	Disable the Analog Comparator 1.

Bit 6 – AC1IE Analog Comparator 1 Interrupt Enable bit

Value	Description
1	Enable the Analog Comparator 1 interrupt.
0	Disable the Analog Comparator 1 interrupt.

Bits 5:4 – AC1IS[1:0] Analog Comparator 1 Interrupt Select bit
These two bits determine the sensitivity of the interrupt trigger.

Value	Description
00	Comparator Interrupt on output toggle
01	Reserved
10	Comparator interrupt on output falling edge
11	Comparator interrupt on output rising edge

Bit 3 – AC1ICE Analog Comparator 1 Interrupt Capture Enable bit
Set this bit to enable the input capture of the Timer/Counter1 on the analog comparator event. 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. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

In case ICES1 bit is set high, the rising edge of AC1O is the capture/trigger event of the Timer/Counter1, in case ICES1 is set to zero, it is the falling edge which is taken into account.

Clear this bit to disable this function. In this case, no connection between the analog comparator and the input capture function exists.

Bits 2:0 – AC1M[2:0] Analog Comparator 1 Multiplexer register
These three bits determine the input of the negative input of the analog comparator.

Value	Description
000	V_{REF}/6.40
001	V_{REF}/3.20
010	V_{REF}/2.13
011	V_{REF}/1.60
100	Bandgap (1.1V)
101	DAC result
110	Analog comparator negative input (ACMPM pin)
111	Reserved

25.4.3 Analog Comparator 2 Control Register

Name: AC2CON
Offset: 0x96
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	AC2EN	AC2IE	AC2IS[1:0]			AC2M[2:0]		
Access								
Reset	0	0	0	0		0	0	0

Bit 7 – AC2EN Analog Comparator 2 Enable

Value	Description
1	Enables the Analog Comparator 2.
0	Disables the Analog Comparator 2.

Bit 6 – AC2IE Analog Comparator 2 Interrupt Enable

Value	Description
1	Enables the Analog Comparator 2 interrupt.
0	Disables the Analog Comparator 2 interrupt.

Bits 5:4 – AC2IS[1:0] Analog Comparator 2 Interrupt Select
These two bits determine the sensitivity of the interrupt trigger.

Value	Description
00	Comparator Interrupt on output toggle
01	Reserved
10	Comparator interrupt on output falling edge
11	Comparator interrupt on output rising edge

Bits 2:0 – AC2M[2:0] Analog Comparator 2 Multiplexer
These three bits determine the input of the negative input of the analog comparator.

Value	Description
000	V_{REF}/6.40
001	V_{REF}/3.20
010	V_{REF}/2.13
011	V_{REF}/1.60
100	Bandgap (1.1V)
101	DAC result
110	Analog comparator negative input (ACMPM pin)
111	Reserved

25.4.4 Analog Comparator 3 Control Register

Name: AC3CON
Offset: 0x97
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	AC3EN	AC3IE	AC3IS[1:0]			AC3M[2:0]		
Access								
Reset	0	0	0	0		0	0	0

Bit 7 – AC3EN Analog Comparator 3 Enable Bit

Value	Description
1	Enable the Analog Comparator 3.
0	Disable the Analog Comparator 3.

Bit 6 – AC3IE Analog Comparator 3 Interrupt Enable bit

Value	Description
1	Enable the Analog Comparator 3 interrupt.
0	Disable the Analog Comparator 3 interrupt.

Bits 5:4 – AC3IS[1:0] Analog Comparator 3 Interrupt Select bit
These two bits determine the sensitivity of the interrupt trigger.

Value	Description
00	Comparator Interrupt on output toggle
01	Reserved
10	Comparator interrupt on output falling edge
11	Comparator interrupt on output rising edge

Bits 2:0 – AC3M[2:0] Analog Comparator 3 Multiplexer register
These three bits determine the input of the negative input of the analog comparator.

Value	Description
000	V_{REF}/6.40
001	V_{REF}/3.20
010	V_{REF}/2.13
011	V_{REF}/1.60
100	Bandgap (1.1V)
101	DAC result
110	Analog comparator negative input (ACMPM pin)
111	Reserved

25.4.5 Analog Comparator Status Register

Name: ACSR
Offset: 0x50
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	AC3IF	AC2IF	AC1IF	AC0IF	AC3O	AC2O	AC1O	AC0O
Access								
Reset	0	0	0	0	0	0	0	0

Bit 7 – AC3IF Analog Comparator 3 Interrupt Flag Bit
This bit is set by hardware when Comparator 3 output event triggers off the interrupt mode defined by AC3IS1 and AC3IS0 bits in AC3CON register.

This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC3IE in AC3CON register is set. Anyway, this bit is cleared by writing a logical one on it.

This bit can also be used to synchronize ADC or DAC conversions.

Bit 6 – AC2IF Analog Comparator 2 Interrupt Flag Bit
This bit is set by hardware when Comparator 2 output event triggers off the interrupt mode defined by AC2IS1 and AC2IS0 bits in AC2CON register.

This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC2IE in AC2CON register is set. Anyway, this bit is cleared by writing a logical one on it.

This bit can also be used to synchronize ADC or DAC conversions.

Bit 5 – AC1IF Analog Comparator 1 Interrupt Flag Bit
This bit is set by hardware when Comparator 1 output event triggers off the interrupt mode defined by AC1IS1 and AC1IS0 bits in AC1CON register.

This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC1IE in AC1CON register is set. Anyway, this bit is cleared by writing a logical one on it.

This bit can also be used to synchronize ADC or DAC conversions.

Bit 4 – AC0IF Analog Comparator 0 Interrupt Flag Bit
This bit is set by hardware when Comparator 0 output event triggers off the interrupt mode defined by AC0IS1 and AC0IS0 bits in AC0CON register.

This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC3IE in AC3CON register is set. Anyway, this bit is cleared by writing a logical one on it.

This bit can also be used to synchronize ADC or DAC conversions.

Bit 3 – AC3O AC3O bit is directly the output of the Analog Comparator 3.

Value	Description
1	The output of the comparator is high.
0	The output comparator is low.

Bit 2 – AC2O AC2O bit is directly the output of the Analog Comparator 2.

Value	Description
1	The output of the comparator is high.
0	The output comparator is low.

Bit 1 – AC1O AC1O bit is directly the output of the Analog Comparator 1.

Value	Description
1	The output of the comparator is high.
0	The output comparator is low.

Bit 0 – AC0O AC0O bit is directly the output of the Analog Comparator 0.

Value	Description
1	The output of the comparator is high.
0	The output comparator is low.

25.4.6 Digital Input Disable Register 0

Name: DIDR0
Offset: 0x7E
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
		ACMPN1D	ACMPN0D		ACMPN2D	ACMP2D		ACMPN3D
Access								
Reset		0	0		0	0		0

Bit 6 – ACMPN1D ACMPN1 Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

Bit 5 – ACMPN0D ACMPN0 Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

Bit 3 – ACMPN2D ACMPN2 Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

Bit 2 – ACMP2D ACMP2 Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

Bit 0 – ACMPN3D ACMPN3 Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

25.4.7 Digital Input Disable Register 1

Name: DIDR1
Offset: 0x7F
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
			ACMP0D			ACMP1D	ACMP3D	
Access								
Reset			0			0	0	

Bit 5 – ACMP0D ACMP0 Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

Bits 1, 2 – ACMPxD ACMPx Digital Input Disable
When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

26. DAC – Digital to Analog Converter

26.1 Features

- 10-bits resolution
- 8-bits linearity
- ±0.5LSB accuracy between 100mV and AV_{CC} - 100mV
- V_{out} = DAC × V_{REF}/1023
- The DAC could be connected to the negative inputs of the analog comparators and/or to a dedicated output driver
- The output impedance of the driver is around 100Ohm. So the driver is able to load a 1nF capacitance in parallel with a resistor higher than 33K with a time constant around 1µs

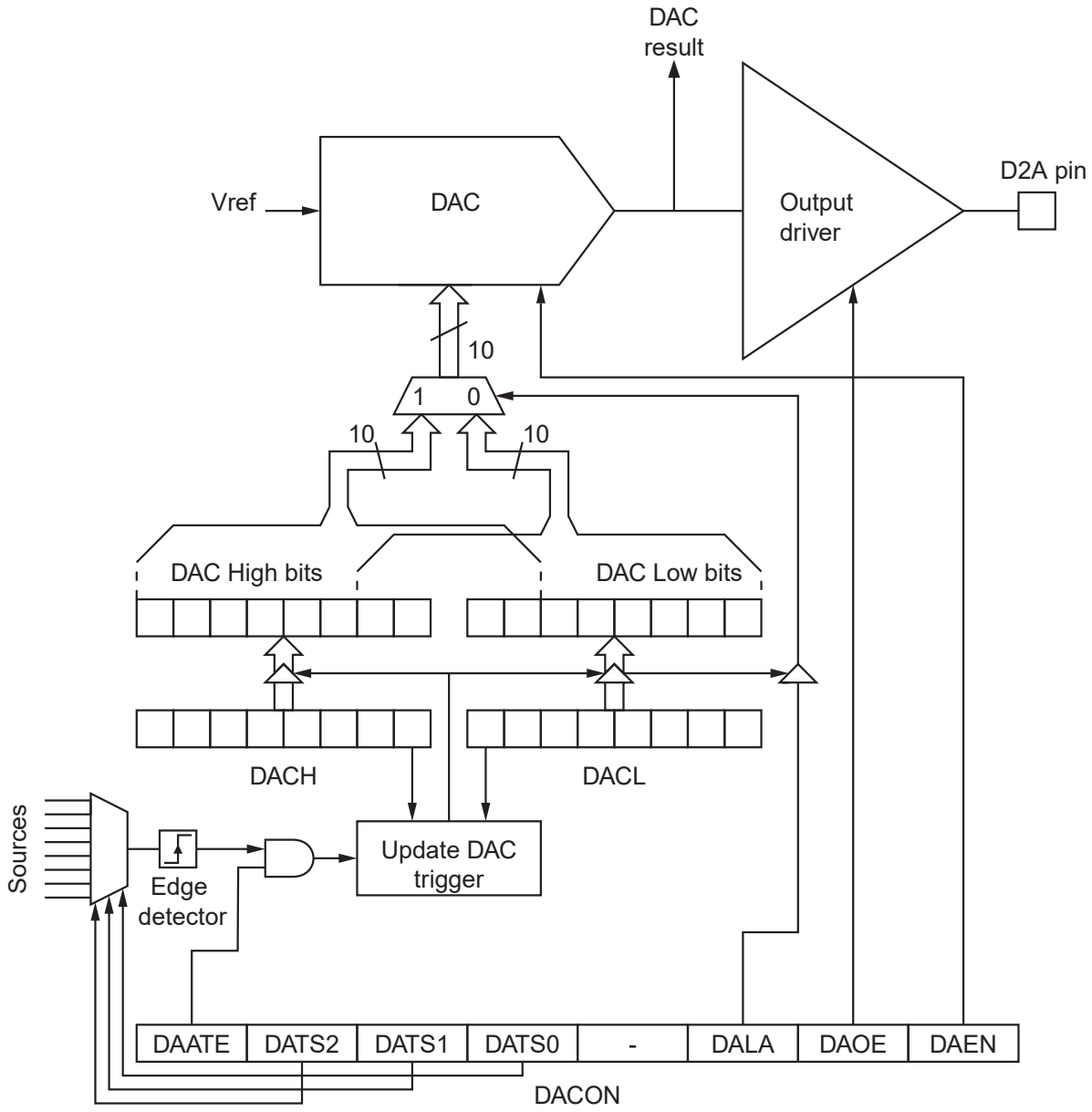
26.2 Overview

The ATmegaS64M1 features a 10-bit Digital to Analog Converter. This DAC can be used for the analog comparators and/or can be output on the D2A pin of the microcontroller via a dedicated driver.

The DAC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ±0.3V from V_{CC}. See the ADC description for how to connect this pin.

The reference voltage is the same as the one used for the ADC, see the description in the ADMUX register. These nominally 2.56V V_{REF} or AV_{CC} are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

Figure 26-1. Digital to analog converter block schematic



Related Links

ADC Noise Canceler
Analog Input Circuitry
Analog Noise Canceling Techniques
Offset Compensation Schemes
ADC Accuracy Definitions
ADMUX

26.3 Operation

The digital to analog converter generates an analog signal proportional to the value of the DAC registers value.

In order to have an accurate sampling frequency control, there is the possibility to update the DAC input values through different trigger events.

26.4 Starting a conversion

The DAC is configured thanks to the DACON register. As soon as the DAEN bit in DACON register is set, the DAC converts the value present on the DACH and DACL registers in accordance with the register DACON setting.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the DAC Auto Trigger Enable bit, DAATE in DACON. The trigger source is selected by setting the DAC Trigger Select bits, DATS in DACON (See description of the DATS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the DAC converts the value present on the DACH and DACL registers in accordance with the register DACON setting. This provides a method of starting conversions at fixed intervals. If the trigger signal is still 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 SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the interrupt flag must be cleared in order to trigger a new conversion at the next interrupt event.

26.4.1 DAC voltage reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the DAC. V_{REF} can be selected as either AV_{CC}, internal 2.56V reference, or external AREF pin.

AV_{CC} is connected to the DAC through a passive switch. The internal 2.56V 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 DAC, 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 2.56V as reference selection. The first DAC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

26.5 Register Description

26.5.1 Digital to Analog Conversion Control Register

Name: DACON
Offset: 0x90
Reset: 0x0
Property: R/W

Bit	7	6	5	4	3	2	1	0
	DAATE	DATS[2:0]				DALA	DAOE	DAEN
Access								
Reset	0	0	0	0		0	0	0

Bit 7 – DAATE DAC Auto Trigger Enable
Set this bit to update the DAC input value on the positive edge of the trigger signal selected with the DACTS2-0 bit in DACON register. Clear it to automatically update the DAC input when a value is written on DACH register.

Bits 6:4 – DATS[2:0] DAC Trigger Selection
These bits are only necessary in case the DAC works in auto trigger mode. It means if DAATE bit is set. These three bits select the interrupt event which will generate the update of the DAC input values. The update will be generated by the rising edge of the selected interrupt flag whether the interrupt is enabled or not.

Value	Description
000	Analog Comparator 0
001	Analog Comparator 1
010	External Interrupt Request 0
011	Timer/Counter0 compare Match
100	Timer/Counter0 Overflow
101	Timer/Counter1 compare Match B
110	Timer/Counter1 overflow
111	Timer/Counter1 capture event

Bit 2 – DALA Digital to Analog Left Adjust
Set this bit to left adjust the DAC input data. Clear it to right adjust the DAC input data. The DALA bit affects the configuration of the DAC data registers. Changing this bit affects the DAC output on the next DACH writing.

Bit 1 – DAOE Digital to Analog Output Enable
Set this bit to output the conversion result on D2A. Clear it to use the DAC internally.

Bit 0 – DAEN Digital to Analog Enable
Set this bit to enable the DAC. Clear it to disable the DAC

26.5.2 Digital to Analog Converter Input Register

Name: DAC (DALA=0)
Offset: 0x91
Reset: 0x0
Property: R/W

DACH and DACL registers contain the value to be converted into analog voltage. Writing the DACL register prohibits the update of the input value until DACH has not been written. To write a 10-bit value in the DAC register, first write DACL then write DACH. In order to work easily with only eight bits, there is the possibility to left adjust the input value. By doing so, it is sufficient to write DACH to update the DAC value.

To work with the 10-bit DAC, two registers must be updated. In order to avoid intermediate value, the DAC input values, which are really converted into analog signals, are buffered into unreachable registers. In normal mode, the update of the shadow register is done when the register DACH is written.

If DAATE is set, the DAC input values are updated on the trigger event selected via DATS.

In order to avoid incorrect DAC input values, the update can only be done after writing, respectively, DACL and DACH registers. It is possible to work with an 8-bit configuration by writing the DACH value only. In this case, update is done at each trigger event.

If DAATE is cleared, the DAC is in an automatic update mode. Writing the DACH register automatically updates the DAC input values with the DACH and DACL register values.

Regardless of the configuration of DAATE, changing the DACL register has no effect on the DAC output until the DACH register has also been updated. To work with a 10-bit configuration, DACL must be written before DACH. To work with an 8-bit configuration, writing DACH allows the update of the DAC.

Bit	15	14	13	12	11	10	9	8
							DAC9	DAC8
Access								
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – DAC DAC Value
DALA = 0

26.5.3 Digital to Analog Converter Input Register

Name: DAC (DALA=1)
Offset: 0x91
Reset: 0x0
Property: R/W

DACH and DACL registers contain the value to be converted into analog voltage. Writing the DACL register prohibits the update of the input value until DACH has not been written. To write a 10-bit value in the DAC register, first write DACL then write DACH. In order to work easily with only eight bits, there is the possibility to left adjust the input value. By doing so, it is sufficient to write DACH to update the DAC value.

To work with the 10-bit DAC, two registers must be updated. In order to avoid intermediate value, the DAC input values, which are really converted into analog signals, are buffered into unreachable registers. In normal mode, the update of the shadow register is done when the register DACH is written.

If DAATE is set, the DAC input values are updated on the trigger event selected via DATS.

In order to avoid incorrect DAC input values, the update can only be done after writing, respectively, DACL and DACH registers. It is possible to work with an 8-bit configuration by writing the DACH value only. In this case, update is done at each trigger event.

If DAATE is cleared, the DAC is in an automatic update mode. Writing the DACH register automatically updates the DAC input values with the DACH and DACL register values.

Regardless of the configuration of DAATE, changing the DACL register has no effect on the DAC output until the DACH register has also been updated. To work with a 10-bit configuration, DACL must be written before DACH. To work with an 8-bit configuration, writing DACH allows the update of the DAC.

Bit	15	14	13	12	11	10	9	8
	DAC9	DAC8	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2
Access								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DAC1	DAC0						
Access								
Reset	0	0						

Bits 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 – DAC DAC Value
DALA = 1

27. debugWIRE On-chip Debug System

27.1 Features

- Complete Program Flow Control
- Emulates All On-chip Functions, Both Digital and Analog, except RESET Pin
- Real-time Operation
- Symbolic Debugging Support (Both at C and Assembler Source Level, or for Other HLLs)
- Unlimited Number of Program Break Points (Using Software Break Points)
- Non-intrusive Operation
- Electrical Characteristics Identical to Real Device
- Automatic Configuration System
- High-speed Operation
- Programming of Nonvolatile Memories

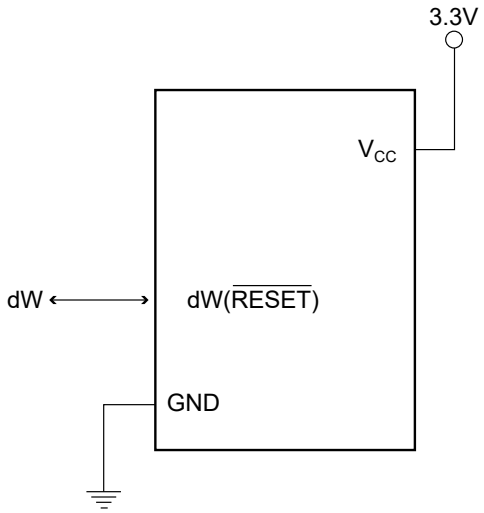
27.2 Overview

The debugWIRE on-chip debug system uses a wire with bi-directional interface to control the program flow and execute AVR instructions in the CPU and to program the different nonvolatile memories.

27.3 Physical Interface

When the debugWIRE Enable (DWEN) bit is programmed to '0' and Lock bits are unprogrammed ('1'), the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 27-1. The debugWIRE Setup



The debugWIRE Setup shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10 kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

27.4 Software Breakpoints

debugWIRE supports the breakpoint functions in program memory by the AVR BREAK instruction. Setting a breakpoint in Atmel Studio will insert a BREAK instruction in the program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time when a breakpoint is changed. This is automatically handled by Atmel Studio through the debugWIRE interface. The use of breakpoints will, therefore, reduce the Flash data retention. Devices used for debugging purposes should not be shipped to end customers.

27.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as external Reset (RESET). An external Reset source is therefore not supported when the debugWIRE is enabled.

A programmed DWEN fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN fuse should be disabled when debugWire is not used.

27.6 Register Description

The following section describes the registers used with the debugWire.

27.6.1 debugWire Data Register

Name: DWDR
Offset: 0x51
Reset: 0x00
Property: When addressing as I/O register: address offset is 0x31

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.

Bit	7	6	5	4	3	2	1	0
	DWDR[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 – DWDR[7:0] debugWire Data
The DWDR register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

28. Boot Loader Support – Read-While-Write Self-programming (BTLDR)

28.1 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. Number of words in a page and number of pages in the Flash in *Page Size*) used during programming. The page organization does not affect normal operation.

28.2 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.

28.3 Application and Boot Loader Flash Sections

The Flash memory is organized into 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.

28.3.1 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.

28.3.2 Boot Loader Section (BLS)

While the application section is used for storing the application code, the boot loader software must be located in the Boot Loader Section (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 BLS can be selected by the Boot Loader Lock bits (Boot Lock bits 1).

28.4 Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write (RWW) 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 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 28-2. The main differences between the two sections are:

- 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

ATmegaS64M1 Boot Loader Parameters

28.4.1 Read-While-Write (RWW) 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 ongoing programming, the software must ensure that the RWW section is never 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. Refer to SPMCSR – Store Program Memory Control and Status Register in this chapter for details on how to clear RWWSB.

28.4.2 No Read-While-Write (NRWW) 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 28-1. 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 28-1. Read-While-Write vs. No Read-While-Write

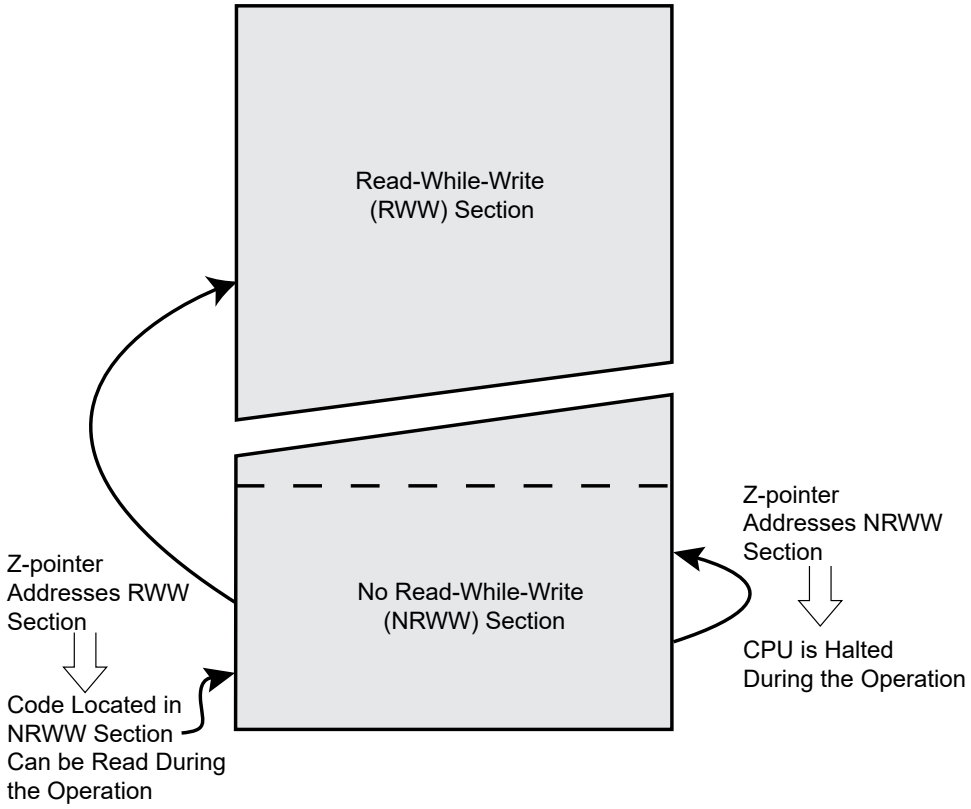
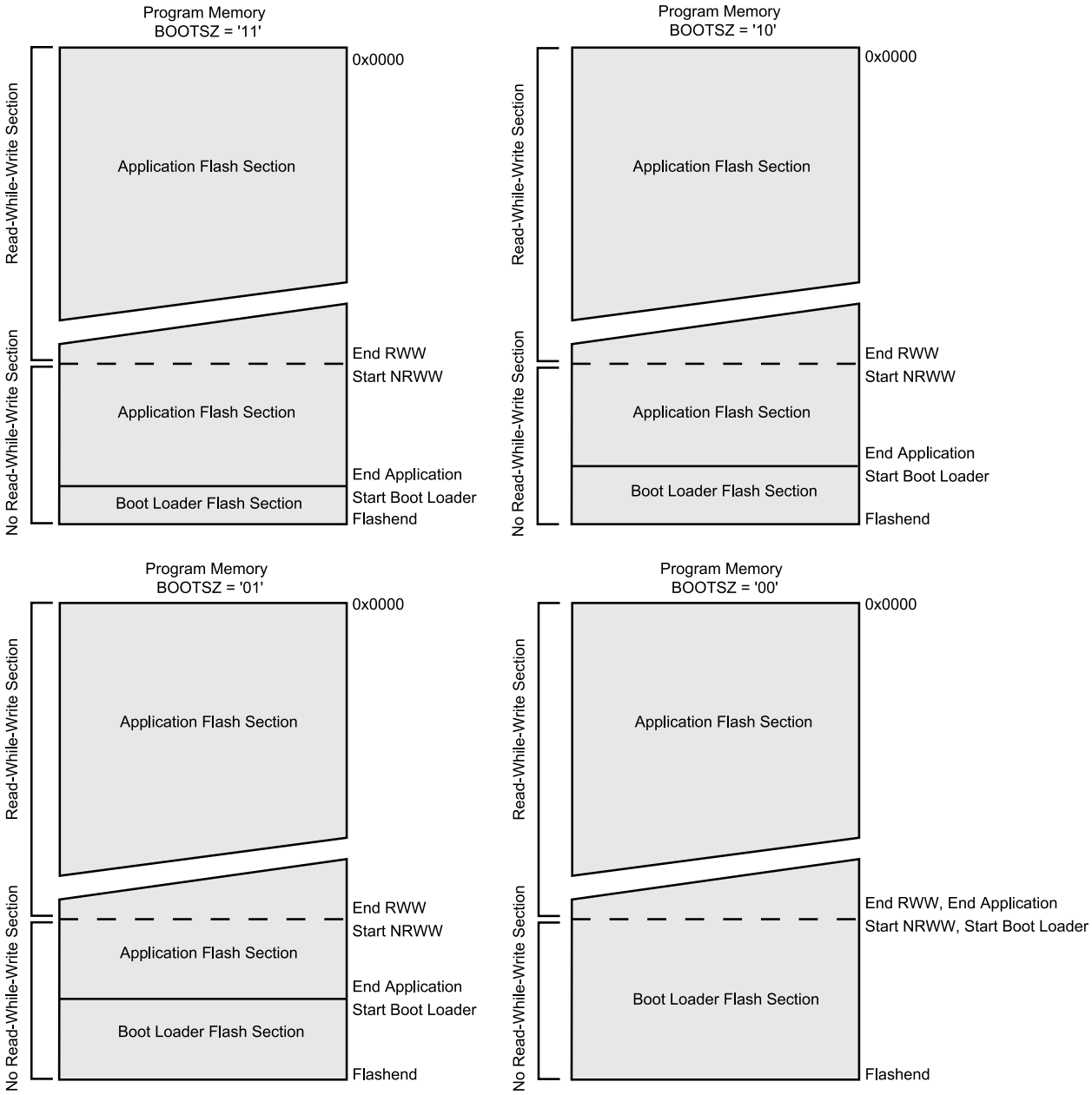


Figure 28-2. Memory Sections



Related Links

ATmegaS64M1 Boot Loader Parameters

28.5 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 28-2. 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 28-3. Boot Lock Bit1 Protection Modes (Boot Loader Section)

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

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

28.6 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 28-4. 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.

28.7 Addressing the Flash During Self-Programming

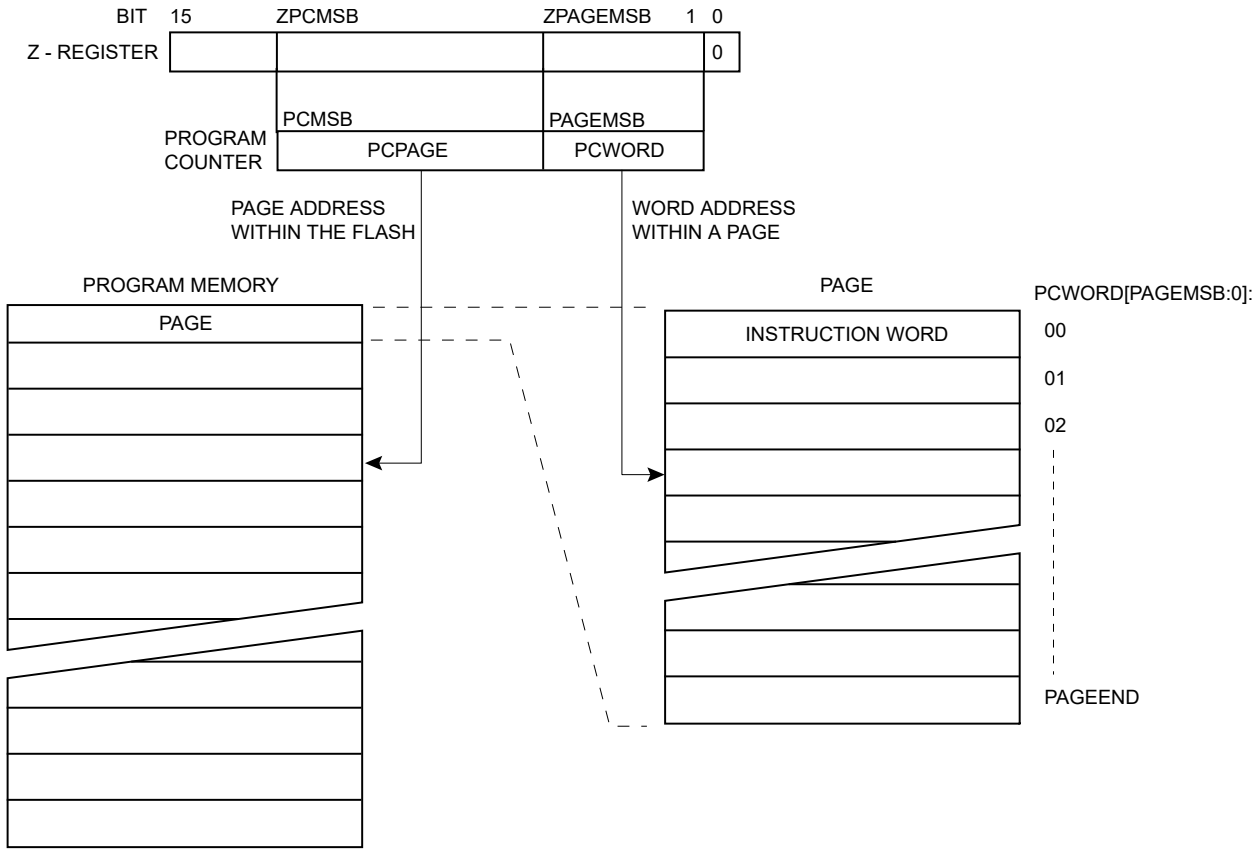
The Z-pointer is used to address the SPM commands. The Z-pointer consists of the Z-registers ZL and ZH in the register file, and RAMPZ in the I/O space. The number of bits actually used is implementation dependent. Note that the RAMPZ register is only implemented when the program space is larger than 64 KB.

Bit	23 15	22 14	21 13	20 12	19 11	18 10	17 9	16 8
	RAMPZ7	RAMPZ6	RAMPZ5	RAMPZ4	RAMPZ3	RAMPZ2	RAMPZ1	RAMPZ0
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 28-3. Addressing the Flash During SPM



Note: The different variables used in this figure are listed in the Related Links.

28.8 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 operations are addressing the same page. Refer to Simple Assembly Code Example for a Boot Loader.

28.8.1 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.

28.8.2 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.

28.8.3 Performing a Page Write

To execute page write, setup 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

28.8.4 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

Interrupt Vectors in ATmegaS64M1

28.8.5 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.

28.8.6 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. Refer to Simple Assembly Code Example for a Boot Loader for an example.

Related Links

Watchdog System Reset

28.8.7 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). For future compatibility, it is recommended to load the Z-pointer with 0x0001 (same as used for reading the I/O_{ck} bits). 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.

28.8.8 EEPROM Write Prevents Writing to SPMCSR

An EEPROM write operation will block all software programming to Flash. Reading the fuses and Lock bits from the software will be prevented during the EEPROM write operation. It is recommended to check the status bit (EEPE) in the EECR Register (EECR.EEPE) and verify that the bit is cleared before writing to the SPMCSR register.

28.8.9 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 into 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 into 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 into the destination register as shown below.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	-	-	EFB3	EFB2	EFB1	EFB0

Fuse and Lock bits that are programmed read as '0'. Fuse and Lock bits that are unprogrammed, will read as '1'.

28.8.10 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.

28.8.11 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 it 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.

28.8.12 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 28-5. SPM Programming Time

Symbol	Min. Programming Time	Max. Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.2 ms	3.4 ms

Note: Minimum and maximum programming time is per individual operation.

28.8.13 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
```

28.8.14 ATmegaS64M1 Boot Loader Parameters

The following tables are the parameters used in the description of the self programming are given.

Table 28-6. Boot Size Configuration, ATmegaS64M1

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	512 words	4	0x0000 - 0x7DFF	0x7E00 - 0x7FFF	0x7DFF	0x7E00
1	0	1024 words	8	0x0000 - 0x7BFF	0x7C00 - 0x7FFF	0x7BFF	0x7C00
0	1	2048 words	16	0x0000 - 0x77FF	0x7800 - 0x7FFF	0x77FF	0x7800
0	0	4096 words	32	0x0000 - 0x6FFF	0x7000 - 0x7FFF	0x6FFF	0x7000

Note: The different BOOTSZ Fuse configurations are shown in Figure 28-2.

Table 28-7. Read-While-Write Limit, ATmegaS64M1

Section	Pages	Address
Read-While-Write section (RWW)	224	0x0000 - 0x6FFF
No Read-While-Write section (NRWW)	32	0x7000 - 0x7FFF

For details about these two sections, refer to No Read-While-Write (NRWW) Section and Read-While-Write (RWW) Section.

Table 28-8. Variables Used in Figure 28-3 and Z-pointer Mapping

Variable		Corresponding Z-value(1)	Description
PCMSB	14	–	Most significant bit in the Program Counter. (The Program Counter is 15 bits PC[14:0])
PAGEMSB	7	–	Most significant bit which is used to address the words within one page (64 words in a page require 7 bits PC [5:0]).
ZPCMSB	–	Z15	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB	–	Z8	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[14:7]	Z15:Z8	Program counter page address: Page select, for page erase and page write
PCWORD	PC[6:0]	Z7:Z1	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z13: always ignored

Z0: should be zero for all SPM commands, byte select for the LPM instruction.

Refer to Addressing the Flash During Self-Programming for details about the use of Z-pointer during Self-Programming.

28.9 Register Description

28.9.1 Store Program Memory Control and Status Register (SPMCSR)

Name: SPMCSR
Offset: 0x57
Reset: 0x00
Property: When addressing as I/O register: address offset is 0x37

The Store Program Memory Control and Status Register (SPMCSR) 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.

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. 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. 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 Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

Bit 1 – PGERS Page Erase
If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

Bit 0 – SPMEN Store Program Memory
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 the 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.

29. Memory Programming (MEMPROG)

29.1 Program And Data Memory Lock Bits

The device provides six Lock bits. These can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in the table *Lock Bit Protection Modes* below. The Lock bits can only be erased to '1' with the Chip Erase command.

Table 29-1. Lock Bit Byte(1)

Lock Bit Byte	Bit No.	Description	Default Value
N/A	7	–	0 (programmed)
N/A	6	–	0 (programmed)
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 29-2. 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 modes. The Fuse bits are locked in both Serial and Parallel Programming modes.(1)
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming modes. The Boot Lock bits and Fuse bits are locked in both Serial and Parallel Programming modes.(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 29-3. Lock Bit Protection - BLB0 Mode(1)(2)

BLB0 Mode	BLB02	BLB01	
1	1	1	No restrictions for Store Program Memory (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 29-4. 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.

29.2 Fuse Bits

The device has three Fuse bytes. The following table describes 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 29-5. Extended Fuse Byte

Extended Fuse Byte	Bit No.	Description	Default Value
N/A	7	–	1 (unprogrammed)
N/A	6	–	1 (unprogrammed)
PSCRB	5	PSC reset behavior	1 (unprogrammed)
PSCRVA	4	PSCOUTnA reset value	1 (unprogrammed)
PSCRVB	3	PSCOUTnB reset value	1 (unprogrammed)
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. See Table 30-5. BODLEVEL Fuse Coding on page 337 for BODLEVEL fuse decoding.

29.3 PSC Output Behavior During Reset

For external component safety reasons, the state of PSC outputs during Reset can be programmed by fuses PSCRB, PSCARV and PSCBRV.

These fuses are located in the Extended Fuse byte.

If PSCRB fuse equals 1 (unprogrammed), all PSC outputs maintain a standard port behavior.

If PSC0RB fuse equals 0 (programmed), all PSC outputs are forced at reset to low level or high level according to PSCARV and PSCBRV fuse bits. In this case, the PSC outputs keep the forced state until POC register is written.

PSCARV (PSCOUTnA Reset Value) gives the state low or high which will be forced on PSCOUT0A, PSCOUT1A, and PSCOUT2A outputs when PSCRB is programmed.

- If PSCARV fuse equals 0 (programmed), the PSCOUT0A, PSCOUT1A, and PSCOUT2A outputs will be forced to high state.
- If PSCRV fuse equals 1 (unprogrammed), the PSCOUT0A, PSCOUT1A, and PSCOUT2A outputs will be forced to low state.

PSCBRV (PSCOUTnB Reset Value) gives the state low or high which will be forced on PSCOUT0B, PSCOUT1B, and PSCOUT2B outputs when PSCRB is programmed.

- If PSCBRV fuse equals 0 (programmed), the PSCOUT0B, PSCOUT1B and PSCOUT2B outputs will be forced to high state.
- If PSCRV fuse equals 1 (unprogrammed), the PSCOUT0B, PSCOUT1B, and PSCOUT2B outputs will be forced to low state.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit 1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Table 29-6. PSC Output Behavior during and after Reset until POC Register is Written

PSCRB	PSCARV	PSCBRV	PSCOUTnA	PSCOUTnB
unprogrammed	X	X	normal port	normal port
programmed	unprogrammed	unprogrammed	forced low	forced low
programmed	unprogrammed	programmed	forced low	forced high
programmed	programmed	unprogrammed	forced high	forced low
programmed	programmed	programmed	forced high	forced high
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)

Table 29-7. Fuse High Byte

High Fuse Byte	Bit Number	Description	Default Value
RSTDISBL (1)	7	External reset disable	1 (unprogrammed)
DWEN	6	debugWIRE enable	1 (unprogrammed)
SPIEN (2)	5	Enable Serial Program and Data Downloading	0 (programmed, SPI programming 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	2	Select Boot Size	1 (unprogrammed)
BOOTSZ0	1	Select Boot Size	1 (unprogrammed)
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Note:

1. See Alternate Functions of Port C for a description of RSTDISBL Fuse.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. See Table 12-1 for details.

Table 29-8. Fuse Low Byte

Low Fuse Byte	Bit Number	Description	Default Value
CKDIV8 (4)	7	Divide clock by 8	1 (unprogrammed)
CKOUT (3)	6	Clock output	1 (unprogrammed)

Low Fuse Byte	Bit Number	Description	Default Value
SUT1	5	Select start-up time	1 (unprogrammed) (1)
SUT0	4	Select start-up time	0 (programmed) (1)
CKSEL3	3	Select Clock source	0 (programmed) (2)
CKSEL2	2	Select Clock source	0 (programmed) (2)
CKSEL1	1	Select Clock source	1 (unprogrammed) (2)
CKSEL0	0	Select Clock source	0 (programmed) (2)

Note:

1. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 10-4 for details.
2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 8MHz. See Device Clocking Options Selector for details.
3. The CKOUT Fuse allows the system clock to be output on PORTB0. See Clock output buffer for details.
4. See System clock prescaler for details.

29.3.1 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.

29.4 Signature Bytes

The device has a three-byte signature code which identifies the device. This code can be read in both serial and parallel modes, as well as when the device is locked. The three bytes reside in a separate address space. Signature bytes are given in the following table.

Table 29-9. Device ID

Part	Signature Byte Addresses		
	0x000	0x001	0x002
ATmegaS64M1	0x1E	0x96	0x84

29.5 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.

29.6 Page Size

Table 29-10. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmegaS64M1	32K words (64 KB)	128 words	PC[6:0]	256	PC[14:7]	14

Table 29-11. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEARMSB
ATmegaS64M1	2 KB	8 bytes	EEAR[2:0]	256	EEAR[10:2]	10

29.7 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 250 ns unless otherwise noted.

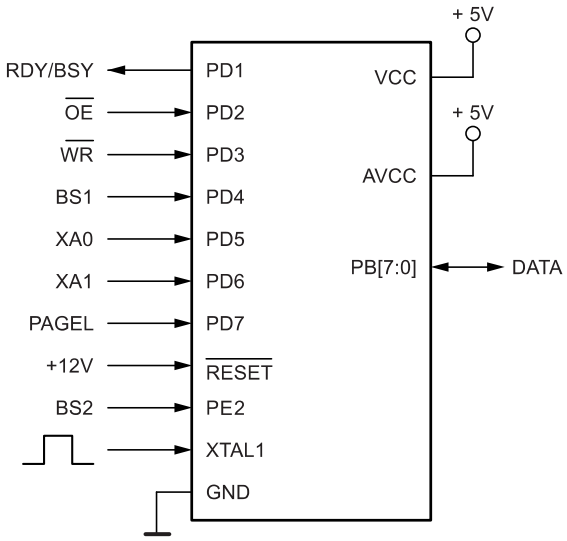
29.7.1 Signal Names

In this section, some pins of this device are referenced by signal names describing their functionality during parallel programming. Refer to figure *Parallel Programming* and table *Pin Name Mapping* below. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in table *XA1 and XA0 Coding* below.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different commands are shown in table *Command Byte Bit Coding* below.

Figure 29-1. Parallel Programming



Note: V_{CC} - 0.3V < AV_{CC} < V_{CC} + 0.3V; however, AV_{CC} should always be within 4.5V to 5.5V.

Table 29-12. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active low)
$\overline{WR}$	PD3	I	Write Pulse (Active low)
BS1	PD4	I	Byte Select 1 (“0” selects Low byte, “1” selects High byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
PAGEL	PD7	I	Program memory and EEPROM Data Page Load
BS2	PE2	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 $\overline{OE}$ is low)

Table 29-13. 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 29-14. 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 29-15. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM

Command Byte	Command Executed
0000 1000	Read Signature Bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

29.8 Parallel Programming

29.8.1 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 V_{CC} and GND.
   Ensure that V_{CC} reaches at least 1.8V within the next 20 µs.
3. Wait for 20–60 µs, and apply 11.5–12.5V to $\overline{\text{RESET}}$.
4. Keep the Prog_enable pins unchanged for at least 10 µs after the high voltage has been applied to ensure the Prog_enable signature has been latched.
5. Wait at least 300 µs 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 V_{CC} 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 10 µs 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.

29.8.2 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 256 byte EEPROM. This consideration also applies to Signature bytes reading.

29.8.3 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 $\overline{WR}$ a negative pulse. This starts the chip erase. RDY/$\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

29.8.4 Programming the Flash

The Flash is organized in pages as a 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. (Refer to 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 (page size < 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

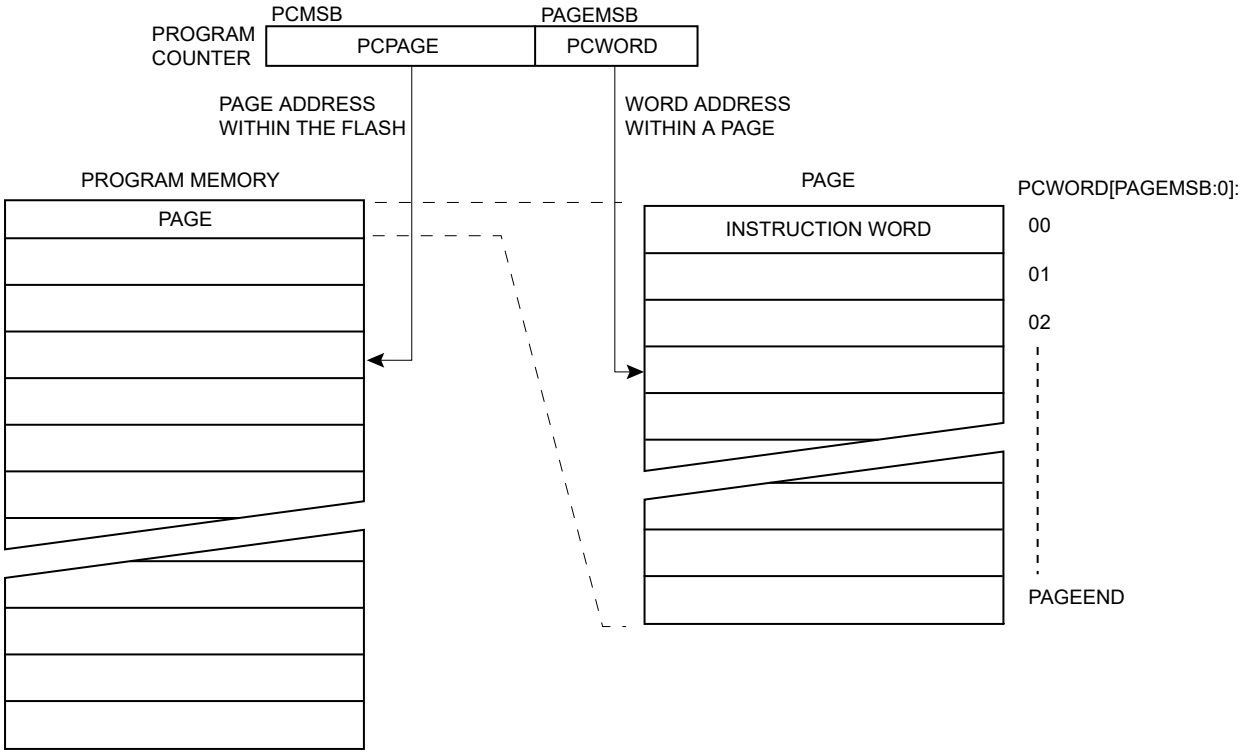
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high. (Refer to the figure, Programming the Flash Waveforms, in this section for signal waveforms.)

Step I. Repeat B Through H Until the Entire Flash is Programmed or Until All Data Has Been Programmed

Step J. End Page Programming

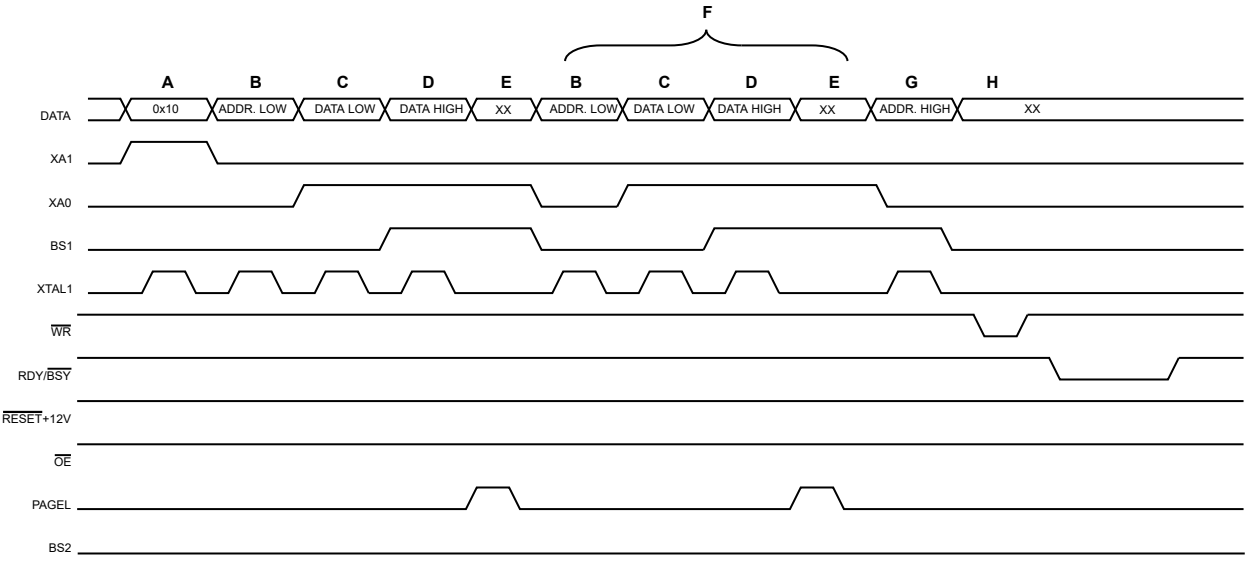
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for no operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are Reset.

Figure 29-2. Addressing the Flash Which is Organized in Pages



Note: PCPAGE and PCWORD are listed in table *No. of Words in a Page and No. of Pages* in the Flash in *Page Size* section.

Figure 29-3. Programming the Flash Waveforms



Note: "XX" is don't care. The letters refer to the programming description above.

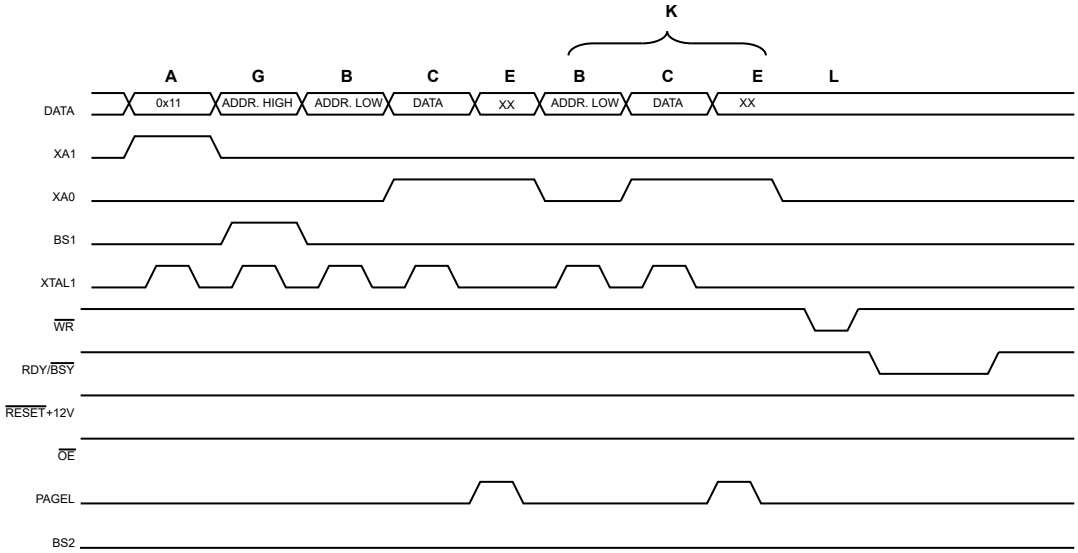
29.8.5 Programming the EEPROM

The EEPROM is organized in pages, refer to the table, number of words in a page and number 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, 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 (refer to the following figure for signal waveforms).

Figure 29-4. Programming the EEPROM Waveforms



29.8.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (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”.

29.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (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 $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

29.8.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (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 $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

29.8.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to Programming the Flash for details on Command and Data loading):

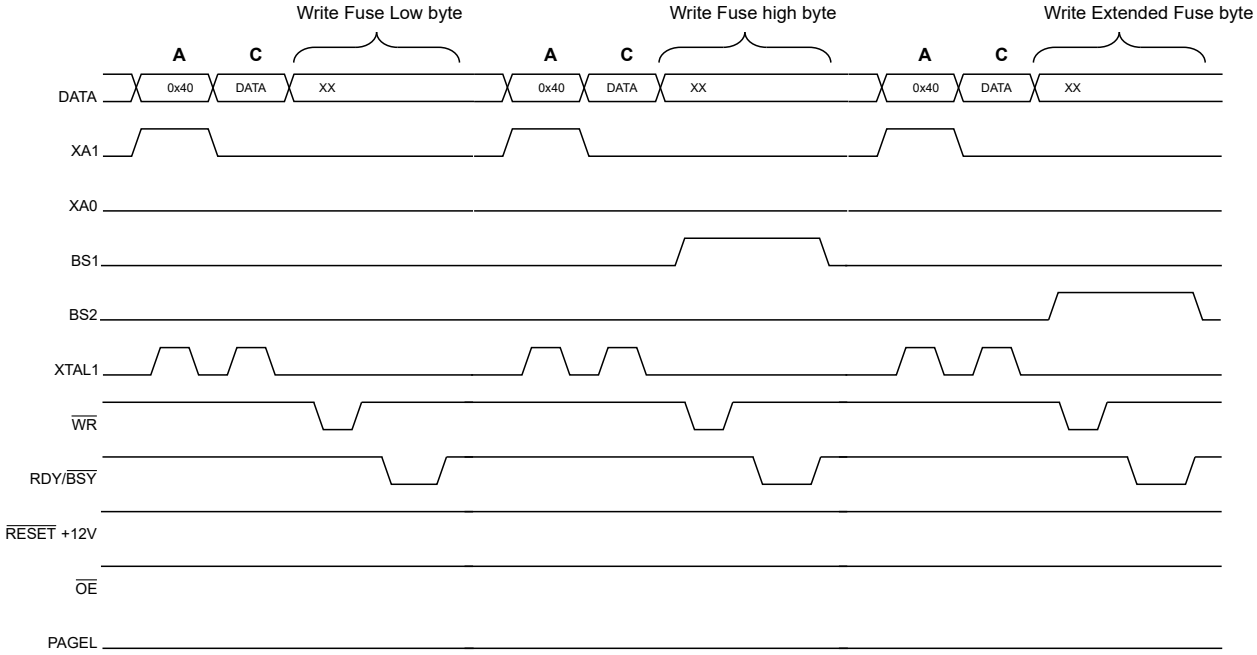
1. Step A: Load Command “0100 0000”.
2. Step C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give WR a negative pulse and wait for RDY/BSY to go high.
5. Set BS1 to “0”. This selects low data byte.

29.8.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (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 $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2 to “0”. This selects low data byte.

Figure 29-5. Programming the FUSES Waveforms



29.8.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (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 $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

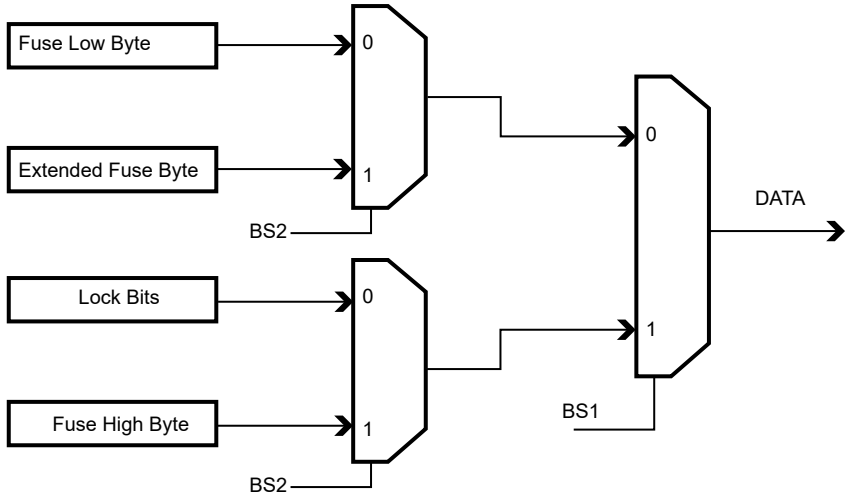
The Lock bits can only be cleared by executing chip erase.

29.8.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to Programming the Flash for details on Command loading):

1. Step A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set $\overline{OE}$ to “0”, BS2 to “1”, and BS1 to “0”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 29-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



29.8.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to Programming the Flash for details on command and address loading):

1. Step A: Load Command “0000 1000”.
2. Step B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

29.8.14 Reading the Calibration Byte

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

1. Step A: Load Command “0000 1000”.
2. Step B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

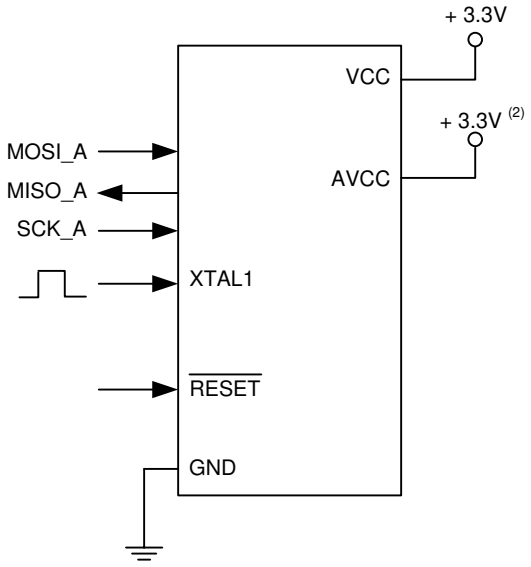
29.8.15 Parallel Programming Characteristics

For characteristics of the Parallel Programming, refer to *Parallel Programming Characteristics*.

29.9 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the programming enable instruction needs to be executed first before the program/erase operations can be executed.

Figure 29-7. Serial Programming and Verify, V_{CC} = 3.3V(See Note 1)



Note:

1. If the device is clocked by the internal oscillator, it is not necessary 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 the specified voltage range (V_{CC}) for the device.

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} ≤ 8 MHz
- High: > 2 CPU clock cycles for f_{ck} ≤ 8 MHz

29.9.1 Serial Programming Pin Mapping

Table 29-16. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI_A	PD3	I	Serial data in
MISO_A	PD2	O	Serial data out
SCK_A	PD4	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.

29.9.2 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. 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 the table Serial Programming Instruction Set (Hexadecimal values) in section Serial Programming Instruction Set):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer cannot guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the programming enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in 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 are altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. In a chip erased device, no 0xFF in the data file(s) needs to be programmed.
6. Any memory location can be verified by using the read instruction, which returns the content at the selected address at 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”, then turn V_{CC} power off.

Table 29-17. Typical Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Typical Wait Delay
t_{WD_FLASH}	2.6 ms
t_{WD_EEPROM}	3.6 ms
t_{WD_ERASE}	10.5 ms
t_{WD_FUSE}	2.6 ms

29.9.3 Data Polling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed gives the value 0xFF. When the device is ready for a new page, the programmed value reads correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value 0xFF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF can be skipped. See the table "Typical Wait Delay Before Writing the Next Flash or EEPROM Location" for t_{WD_FLASH} value.

29.9.4 Data Polling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed gives the value 0xFF. When the device is ready for a new byte, the programmed value reads correctly. This is used to determine when the next byte can be written. This does not work for the value 0xFF, but the user should keep in mind that as a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF can be skipped. This does not apply if the EEPROM is re-programmed without chip erasing the device. In this case, data polling cannot be used for the value 0xFF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See "Typical Wait Delay Before Writing the Next Flash or EEPROM Location" for t_{WD_EEPROM} value.

29.9.5 Serial Programming Instruction Set

Table 29-18. Serial Programming Instruction Set (Hexadecimal Values)

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming Enable	0xAC	0x53	0xXX	0xXX	Enable serial programming after $\overline{\text{RESET}}$ goes low.
Chip Erase	0xAC	100x xxxx	0xXX	0xXX	Chip erase EEPROM and Flash
Poll RDY/$\overline{\text{BSY}}$	0xF0	0x00	xxxx xxxx	xxxx xxx**o**	If **o** = “1”, a programming operation is still busy. Wait until this bit returns to “0” before applying another command.
Load Instructions					
Load Program Memory Page	0100 **H**000	000x xxxx	**bbbb bbbb**(2)	**data byte in**	Write **H** (high or low) data to program memory page at word address **b**. Data low byte must be loaded before Data high byte is

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
					applied within the same address.
Load EEPROM Memory Page (page access)	0xC1	0x00	0000 00**bb**(2)	data byte in	Load data to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Read Instructions(4)					
Read Program Memory	0010 **H**000	000**a aaaa**(2)	**bbbb bbbb**(2)	**data byte out**	Read **H** (high or low) data from program memory at word address **a:b**.
Read EEPROM Memory	0xA0	000x xx**aa**(2)	**bbbb bbbb**(2)	data byte out	Read data **out** from EEPROM memory at address **a:b**.
Read Lock bits	0x58	0x00	0xXX	xx**oo oooo**	Read lock bits. “0” = programmed, “1” = unprogrammed. See Table 29-1 for details.
Read Signature Byte	0x30	000x xxxx	xxxx xx**bb**(2)	data byte out	Read signature byte at address **b**.
Read Fuse bits	0x50	0x00	0xXX	data byte out	Read Fuse bits. “0” = programmed, “1” = unprogrammed.
Read Fuse High bits	0x58	0x08	0xXX	data byte out	Read fuse high bits. “0” = programmed, “1” = unprogrammed. See Table 29-7 for details.
Read Extended Fuse Bits	0x50	0x08	0xXX	data byte out	Read extended fuse bits. “0” = programmed, “1” = unprogrammed. See Table 29-5 for details.
Read Calibration Byte	0x38	000x xxxx	0x00	data byte out	Read calibration byte
Write Instructions(4)					

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Write Program Memory Page[5]	0x4C	**aaaa aaaa**[2]	**bb**xx xxxx[2]	0xXX	Write program memory page at address a:b.
Write EEPROM Memory	0xC0	000x xx**aa**[2]	**bbbb bbbb**[2]	data byte in	Write data to EEPROM memory at address **a:b**.
Write EEPROM Memory Page (page access)	0xC2	00xx xx**aa**[2]	**bbbb bb00**[2]	0xXX	Write EEPROM page at address **a:b**.
Write Lock bits[3]	0xAC	111x xxxx	0xXX	11**ii iiii**[2]	Write lock bits. Set bits = “0” to program lock bits. SeeTable 29-1 for details.
Write Fuse bits[3]	0xAC	0xA0	0xXX	data byte in	Set bits = “0” to program, “1” to unprogram.
Write Fuse High bits[3]	0xAC	0xA8	0xXX	data byte in	Set bits = “0” to program, “1” to unprogram. See Table 29-7 for details.
Write Extended Fuse Bits[3]	0xAC	0xA4	0xXX	11**ii iiii**[2]	Set bits = “0” to program, “1” to unprogram. See Table 29-5 for details.

Note:

1. Not all instructions are applicable for all parts.
2. **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
3. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed ('1').
4. Refer to the corresponding section for Fuse and Lock bits, Calibration and Signature bytes and page size.
5. Instructions accessing program memory use a word address. This address may be random within the page range.

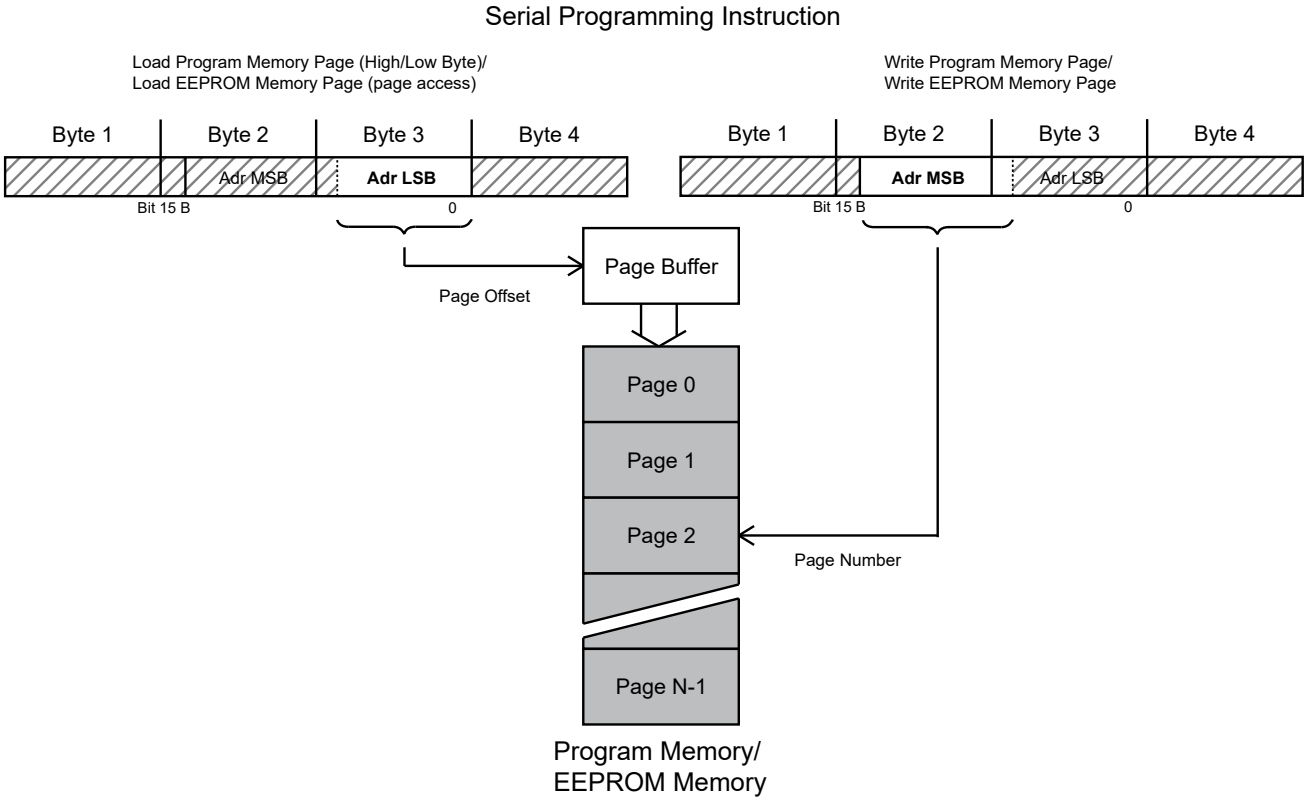
Note: See http://www.microchip.com/design-centers/8-bit/microchip-avr-mcus for application notes regarding programming and programmers.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded into the page buffer, program the EEPROM page. Refer to the following figure.

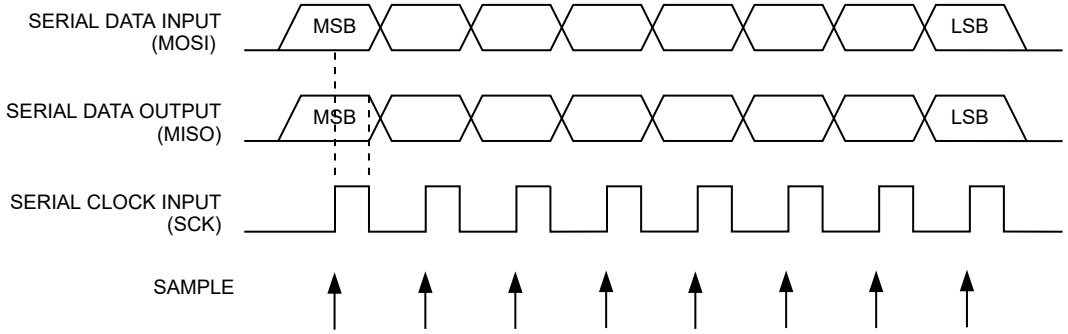
Within the same moisture group, the user should not configure all the sensors to the single multi-touch group.

Figure 29-8. Serial Programming Instruction Example



29.9.6 SPI Serial Programming Characteristics

Figure 29-9. Serial Programming Waveforms



30. Electrical Characteristics

30.1 Absolute Maximum Ratings*

Operating temperature	-55°C to +125°C	*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
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	200.0mA	

30.2 DC Characteristics

Table 30-1. T_A = -55°C to +125°C, V_{CC} = 3.0V to 3.6V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	Port B, C & D and XTAL1, XTAL2 pins as I/O	–	–	$0.2V_{CC}$ (1)	V
V_{IH}	Input High Voltage	Port B, C & D and XTAL1, XTAL2 pins as I/O	$0.6V_{CC}$ (2)	–	V_{CC}+0.5	
V_{IL1}	Input Low Voltage	XTAL1 pin, external clock selected	-0.5	–	$0.1V_{CC}$ (1)	
V_{IH1}	Input High Voltage	XTAL1 pin, external clock selected	$0.8V_{CC}$ (2)	–	V_{CC}+0.5	
V_{IL2}	Input Low Voltage	$\overline{\text{RESET}}$ pin	-0.5	–	$0.2V_{CC}$ (1)	
V_{IH2}	Input High Voltage	$\overline{\text{RESET}}$ pin	$0.9V_{CC}$ (2)	–	V_{CC}+0.5	
V_{IL3}	Input Low Voltage	$\overline{\text{RESET}}$ pin as I/O	-0.5	–	$0.2V_{CC}$ (1)	
V_{IH3}	Input High Voltage	RESET pin as I/O	$0.8V_{CC}$ (2)		V_{CC}+0.5	
V_{OL}	Output Low Voltage (3) (Port B, C & D and XTAL1, XTAL2 pins as I/O)	IOL = 6mA, V_{CC}=3V	–	–	0.5	
V_{OH}	Output High Voltage (4) (Port B, C & D and	IOH = -8mA, V_{CC}=3V	2.2	–	–	

Symbol	Parameter	Condition	Min	Typ	Max	Units
	XTAL1, XTAL2 pins as I/O)					
V_{OL3}	Output Low Voltage (3) ($\overline{RESET}$ pin as I/O)	IOL = 0.8mA, V_{CC}=3V	–	–	0.7	
V_{OH3}	Output High Voltage (4) ($\overline{RESET}$ pin as I/O)	IOH = -0.2mA, V_{CC}=3V	1.7	–	–	
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 3.6V, pin low (absolute value)	–	–	1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 3.6V, pin high (absolute value)	–	–	1	
R_{RST}	Reset Pull-up Resistor	–	30	–	200	kΩ
R_{PU}	I/O Pin Pull-up Resistor	–	20	–	50	
I_{CC}	Power Supply Current	Active 8MHz, V_{CC} max, RC osc, PRR = 0xFF	–	3.8	29	mA
		Idle 8MHz, V_{CC} max, RC Osc	–	1.5	29	
	Power-down mode (5)	WDT enabled, V_{CC} max, 125°C	–	9	80	µA
		WDT disabled, V_{CC} max, 125°C	–	5	80	
V_{HYST}	Analog Comparator Hysteresis Voltage	V_{CC} = 3.3V, Vin = 3V Rising Edge Falling Edge	-100	 25 -35	TBD	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 3.3V, Vin = V_{CC}/2	-50	–	TBD	nA
t_{ACID}	Analog Comparator Propagation Delay	V_{CC} = 2.7V	–	(6)	–	ns

Note:

1. “Max” means the highest value where the pin is guaranteed to be read as low.
2. “Min” means the lowest value where the pin is guaranteed to be read as high.
3. Although each I/O port can sink more than the test conditions (6mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all I_{OL} for ports B0–B1, C2–C3, D4, E1–E2 should not exceed 70mA
   - The sum of all I_{OL} for ports B6–B7, C0–C1, D0–D3, E0 should not exceed 70mA
   - The sum of all I_{OL} for ports B2–B5, C4–C7, D5–D7 should not exceed 70mA

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.

4. Although each I/O port can source more than the test conditions (8mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all I_{OH} for ports B0–B1 C2–C3, D4, E1–E2 should not exceed 100mA
   - The sum of all I_{OH} for ports B6–B7 C0–C1, D0–D3, E0 should not exceed 100mA
   - The sum of all I_{OH} for ports B2–B5 C4–C7, D5–D7 should not exceed 100mA

   If I_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum V_{CC} for power-down is 2.5V.
6. The Analog Comparator Propagation Delay equals one comparator clock plus 30ns. See the section *AC - Analog Comparator* for comparator clock definition.

30.3 Clock Characteristics

30.3.1 Calibrated Internal RC Oscillator Accuracy

Table 30-2. Factory Calibration

Frequency	V_{CC}	Temperature	Calibration Accuracy
8.0MHz	3.3V	25°C	±3%

Table 30-3. User Calibration

User Calibration (max %)			
V_{CC}	-55°C	25°C	125°C
3.0V	3.32%	3.2%	8.83%
3.3V	5.94%	1.4%	7.05%
3.6V	8.15%	3.32%	6.59%

Figure 30-1. User Oscillator at 25°C versus V_{CC}

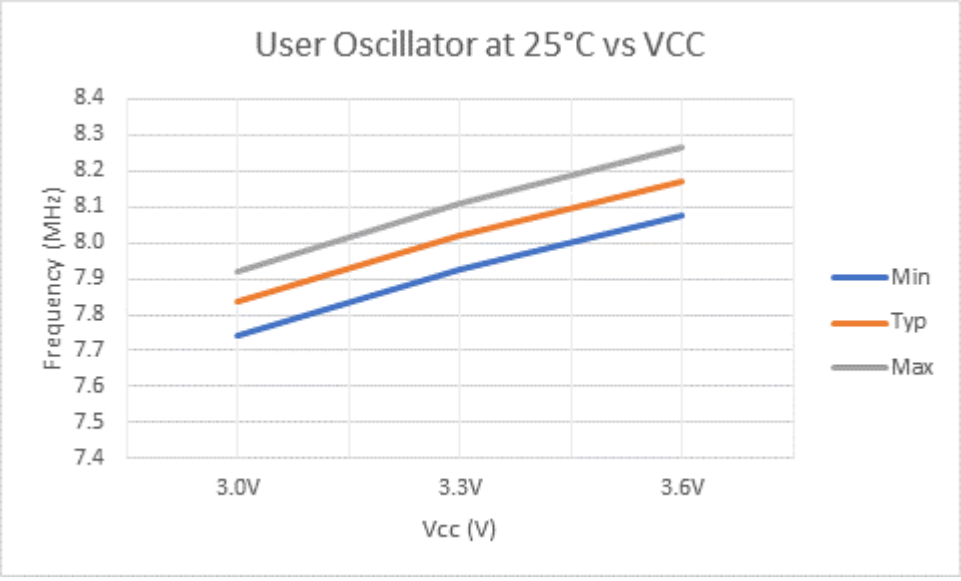


Figure 30-2. User Oscillator at 125°C versus V_{CC}

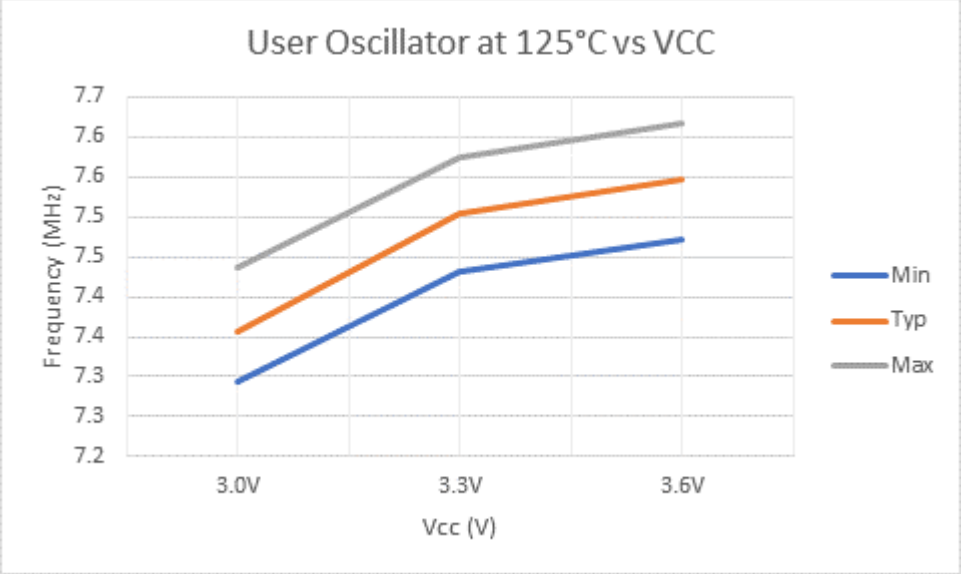
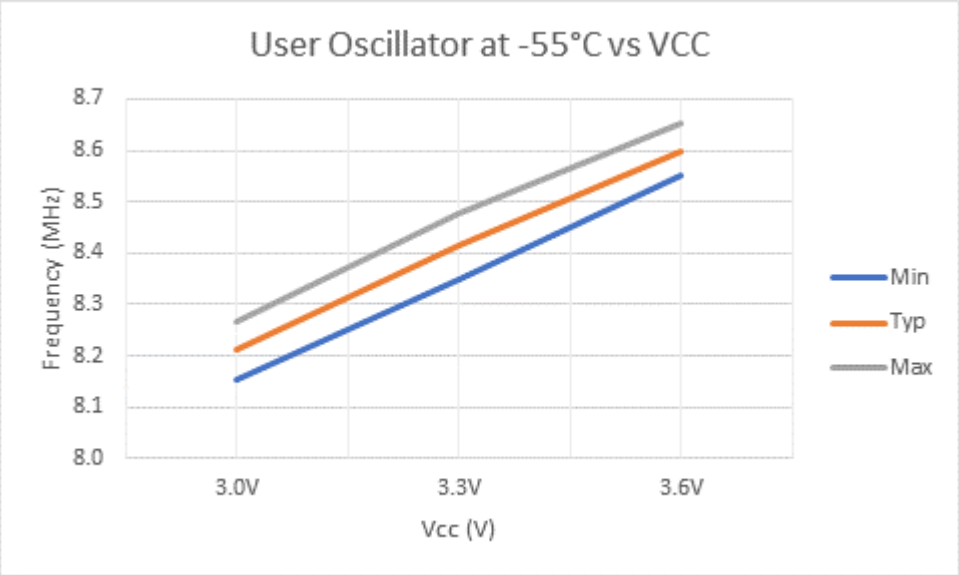


Figure 30-3. User Oscillator at -55°C versus V_{CC}



30.4 External Clock Drive Characteristics

Figure 30-4. External Clock Drive Waveforms

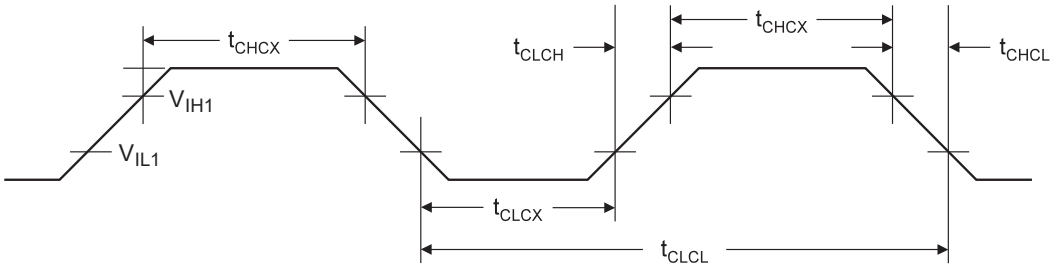


Table 30-4. External Clock Drive

Symbol	Parameter	V_{CC} = 3.0V–3.6V		Units
		Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	8	MHz
t_{CLCL}	Clock Period	125	–	ns
t_{CHCX}	High Time	50	–	
t_{CLCX}	Low Time	50	–	
t_{CLCH}	Rise Time	–	1.6	µs
t_{CHCL}	Fall Time	–	1.6	
Δt_{CLCL}	Change in period from one clock cycle to the next	–	2	%

30.5 System and Reset Characteristics

Table 30-5. Reset, Brown-out[1] and Internal Voltage [1] Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-on reset threshold voltage (rising)	1.1	1.4	1.7	V
	Power-on reset threshold voltage (falling)[2]	0.8	0.9	1.6	
V_{PORMAX}	V_{CC} maximum start voltage to ensure internal power-on reset signal	–	–	0.4	
V_{PORMIN}	V_{CC} minimum start voltage to ensure internal power-on reset signal	-0.1	–	–	
V_{CCRR}	V_{CC} rise rate to ensure power-on reset	0.01	–	–	V/ms
V_{RST}	$\overline{\text{RESET}}$ pin threshold voltage	$0.1V_{CC}$	–	$0.9V_{CC}$	V
V_{HYST}	Brown-out detector hysteresis	–	50	–	mV
t_{BOD}	Min pulse width on brown-out reset	–	2	–	µs
V_{BG}	Bandgap reference voltage	–	1.1	–	V
t_{BG}	Bandgap reference start-up time	–	40	–	µs
I_{BG}	Bandgap reference current consumption	–	15	–	µA

Note:

1. Values are guidelines only.
2. Before rising, the supply has to be between V_{PORMIN} and V_{PORMAX} to ensure a reset.

Table 30-6. BODLEVEL Fuse Coding [1][2]

BODLEVEL[2..0] Fuses	Typ V_{BOT}	Units
111	Disabled	–
110	Reserved	V
011	Reserved	
100	Reserved	
010	Reserved	
001	2.8	
101	2.7	
000	2.6	

Note:

1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a brown-out reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 101.

2. Values are guidelines only.

30.6 PLL Characteristics

Table 30-7. PLL Characteristics (V_{CC} = 3.0V–3.6V unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
PLLIF	Input frequency	0.5	1	2	MHz
PLLF	PLL factor	–	64	–	–
PLLLT	Lock-in time	–	–	64	µS

Note: While connected to external clock or external oscillator, PLL input frequency must be selected to provide outputs with frequency in accordance with driven parts of the circuit (CPU core, PSC, etc.).

30.7 SPI Timing Characteristics

Table 30-8. SPI Timing Parameters

	Description	Mode	Min	Typ	Max	Units
1	SCK period	Master	–	See the SPI Control register	–	ns
2	SCK high/low	Master	–	50% duty cycle	–	
3	Rise/Fall time	Master	–	3.6	–	
4	Setup	Master	–	10	–	
5	Hold	Master	–	10	–	
6	Out to SCK	Master	–	0.5 × t_{SCK}	–	
7	SCK to out	Master	–	10	–	
8	SCK to out high	Master	–	10	–	
9	SS low to out	Slave	–	15	–	
10	SCK period	Slave	4 × t_{CK}	–	–	
11	SCK high/low(1)	Slave	2 × t_{CK}	–	–	
12	Rise/Fall time	Slave	–	–	1600	
13	Setup	Slave	10	–	–	
14	Hold	Slave	t_{CK}	–	–	
15	SCK to out	Slave	–	15	–	
16	SCK to SS high	Slave	20	–	–	
17	SS high to tri-state	Slave	–	10	–	
18	SS low to SCK	Slave	20	–	–	

1. In SPI Programming mode, the minimum SCK high/low period is (-2 t_{CLCL}) for $f_{CK} \leq 8MHz$

.

Figure 30-5. SPI Interface Timing Requirements (Master Mode)

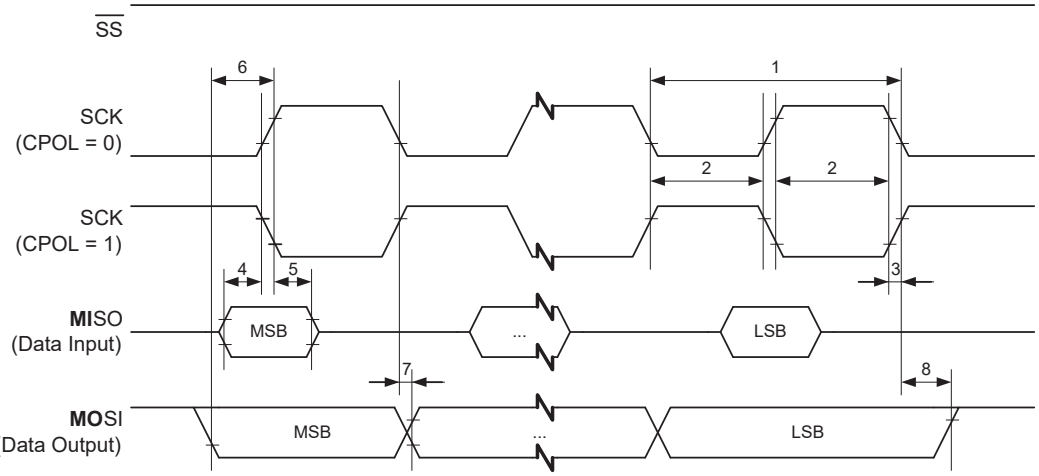
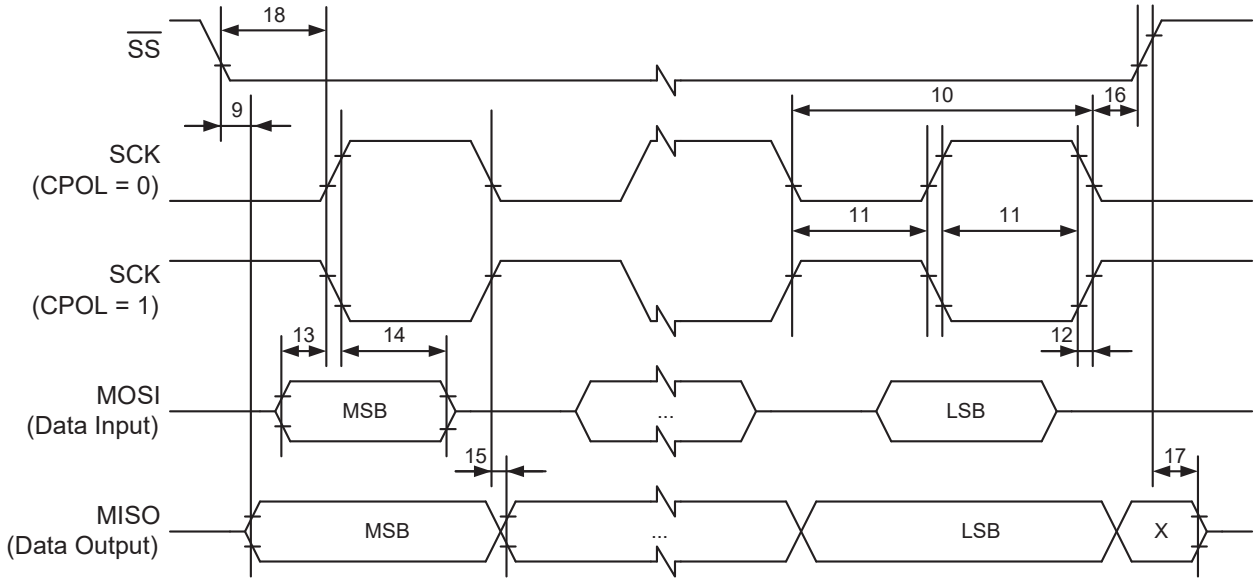


Figure 30-6. SPI Interface Timing Requirements (Slave Mode)



30.8 ADC Characteristics

Table 30-9. ADC Characteristics for Single-ended Conversion (T_A = -55°C to +125°C, V_{CC} = 3.0V–3.6V unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
–	Resolution	Single-ended conversion	–	10	–	bits
TUE	Absolute accuracy	V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 1MHz	–	3.2	TBD	LSB
		V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	3.2	TBD	
INL	Integral non-linearity	V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 1MHz	–	0.7	TBD	

Symbol	Parameter	Condition	Min	Typ	Max	Units
		V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.8	TBD	
DNL	Differential non-linearity	V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 1MHz	–	0.5	TBD	
		V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.6	TBD	
–	Gain error	V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 1MHz	TBD	-5.0	TBD	
		V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	TBD	-5.0	TBD	
–	Offset error	V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 1MHz	TBD	2.5	TBD	
		V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	TBD	2.5	TBD	
V_{REF}	Reference voltage	–	2.56	–	AV_{CC}	V

Table 30-10. ADC Characteristics for Differential Conversion (T_A = -55°C to +125°C, V_{CC} = 3.0V–3.6V unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
–	Resolution	Differential conversion	–	8	–	bits
TUE	Absolute accuracy	Gain = 5×, 10×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	1.5	TBD	LSB
		Gain = 20×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	1.5	TBD	
		Gain = 40×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	1.5	TBD	
INL	Integral non-linearity	Gain = 5×, 10×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.1	TBD	
		Gain = 20×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.2	TBD	
		Gain = 40×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 1MHz	–	0.3	TBD	
		Gain = 40×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.7	TBD	
DNL	Differential non-linearity	Gain = 5×, 10×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.1	TBD	
		Gain = 20×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.2	TBD	
		Gain = 40×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	–	0.3	TBD	
–	Gain error	Gain = 5×, 10×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	TBD	–	TBD	
		Gain = 20×, 40×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	TBD	–	TBD	

Symbol	Parameter	Condition	Min	Typ	Max	Units
–	Offset error	Gain = 5×, 10×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	TBD	–	TBD	
		Gain = 20×, 40×, V_{CC} = 3.6V, V_{REF} = 2.56V, ADC clock = 2MHz	TBD	–	TBD	
V_{REF}	Reference voltage	–	2.56	–	AV_{CC}-0.5	V

30.9 Parallel Programming Characteristics

Figure 30-7. Parallel programming timing, including some general timing requirements.

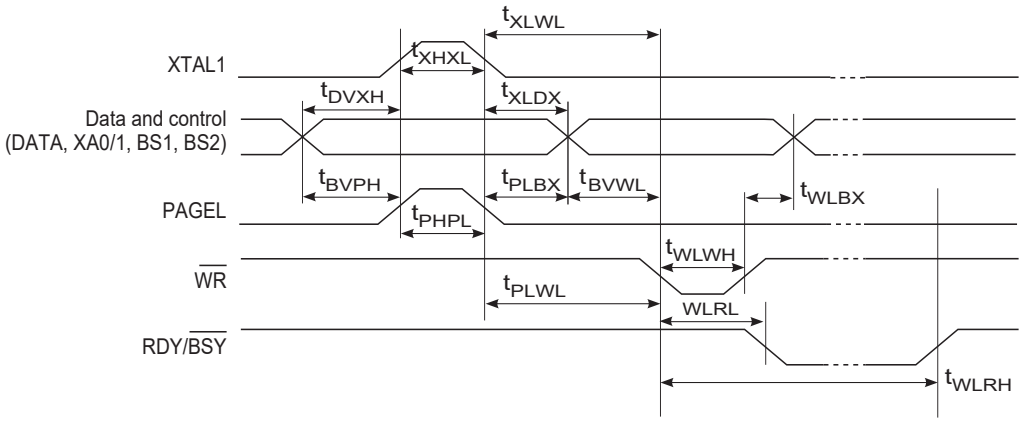
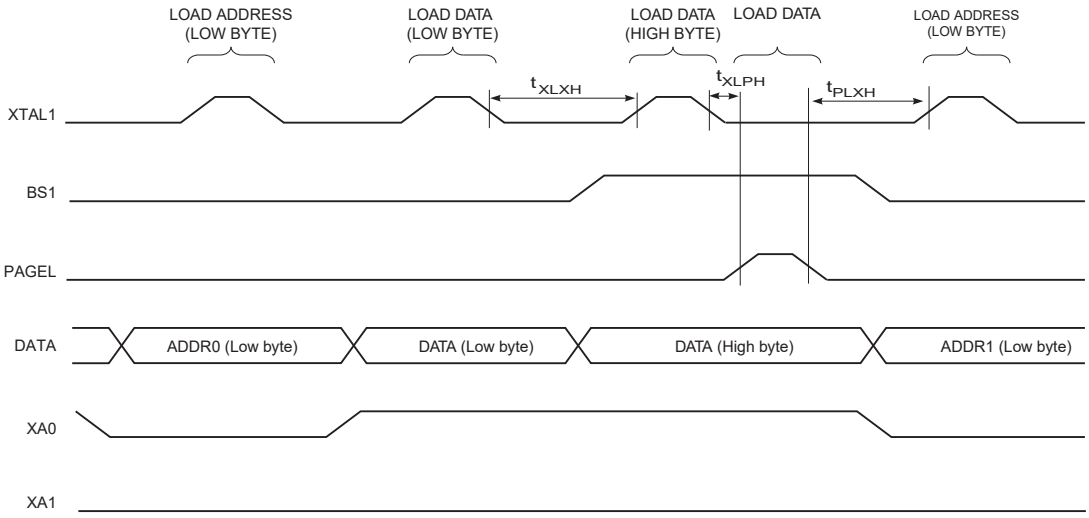
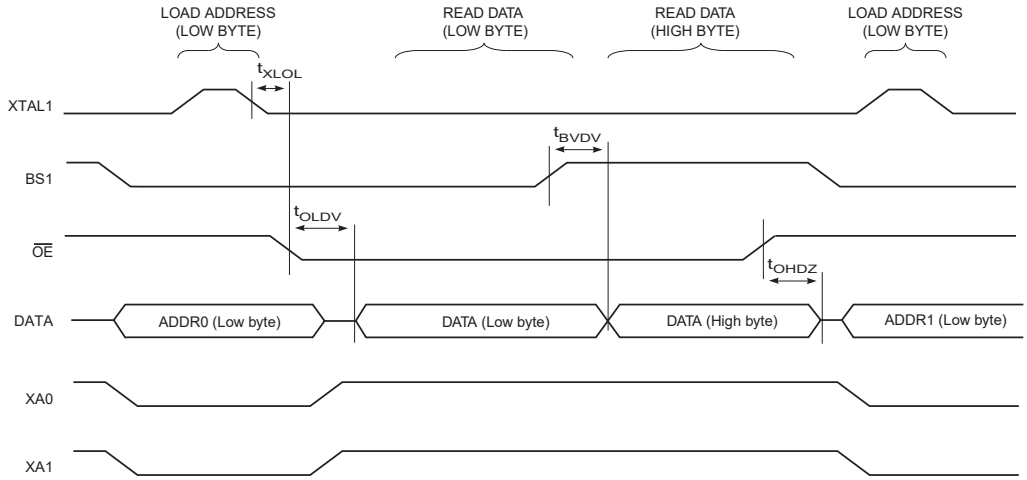


Figure 30-8. Parallel programming timing, loading sequence with timing requirements.



Note: The timing requirements shown in the figures above (that is, tDVXH, tXHXL, and tXLDX) also apply to loading operation.

Figure 30-9. Parallel programming timing, reading sequence (within the same page) with timing requirements.



Note: The timing requirements shown in the figures above (that is, tDVXH, tXHXL, and tXLDX) also apply to reading operation.

Table 30-11. Parallel programming characteristics, V_{CC} = 5V ±10%.

Symbol	Parameter	Min.	Typ.	Max.	Units
VPP	Programming Enable Voltage	11.5		12.5	V
IPP	Programming Enable Current			250	µA
tDVXH	Data and Control Valid before XTAL1 High	67			ns
tXLXH	XTAL1 Low to XTAL1 High	200			
tXHXL	XTAL1 Pulse Width High	150			
tXLDX	Data and Control Hold after XTAL1 Low	67			
tXLWL	XTAL1 Low to WR Low	0			
tXLPH	XTAL1 Low to PAGEL high	0			
tPLXH	PAGEL low to XTAL1 high	150			
tBVPH	BS1 Valid before PAGEL High	67			
tPHPL	PAGEL Pulse Width High	150			
tPLBX	BS1 Hold after PAGEL Low	67			
tWLBX	BS2/1 Hold after WR Low	67			
tPLWL	PAGEL Low to WR Low	67			
tBVWL	BS1 Valid to WR Low	67			
tWLWH	WR Pulse Width Low	150			
tWLRL	WR Low to RDY/BSY Low	0		1	µs
tWLRH	WR Low to RDY/BSY High (1)	3.7		5	ms
tWLRH_CE	WR Low to RDY/BSY High for Chip Erase (2)	7.5		10	

Symbol	Parameter	Min.	Typ.	Max.	Units
tXLOL	XTAL1 Low to OE Low	0			ns
tBVDV	BS1 Valid to DATA valid	0		250	
tOLDV	OE Low to DATA Valid			250	
tOHDZ	OE High to DATA Tri-stated			250	

1. **Note:** tWLRH is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.
2. **Note:** tWLRH_CE is valid for the Chip Erase command.

30.10 Typical Characteristics

All DC characteristics contained in this datasheet are based on characterization data. 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 register set and thus, the corresponding I/O modules are turned off. In addition, 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 determinant factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L x V_{CC} x 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.

30.10.1 Pin Pull-Up

Figure 30-10. I/O Pin Pull-up Resistor Current versus Input Voltage – V_{CC} = 3.0V

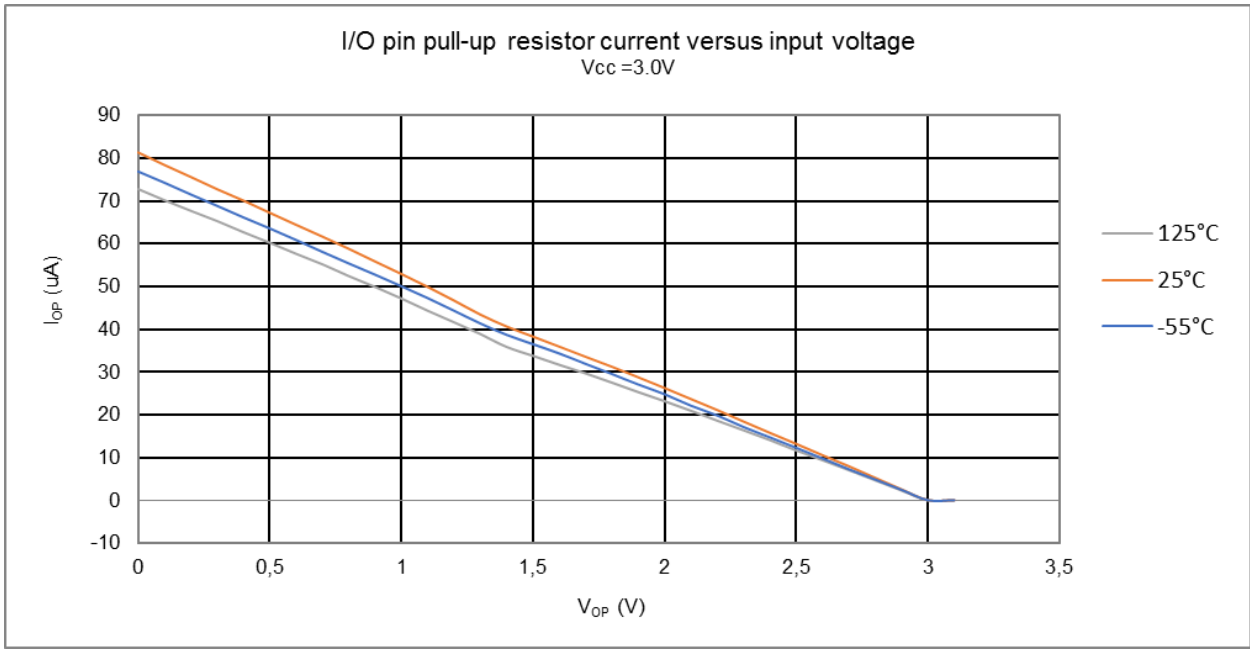
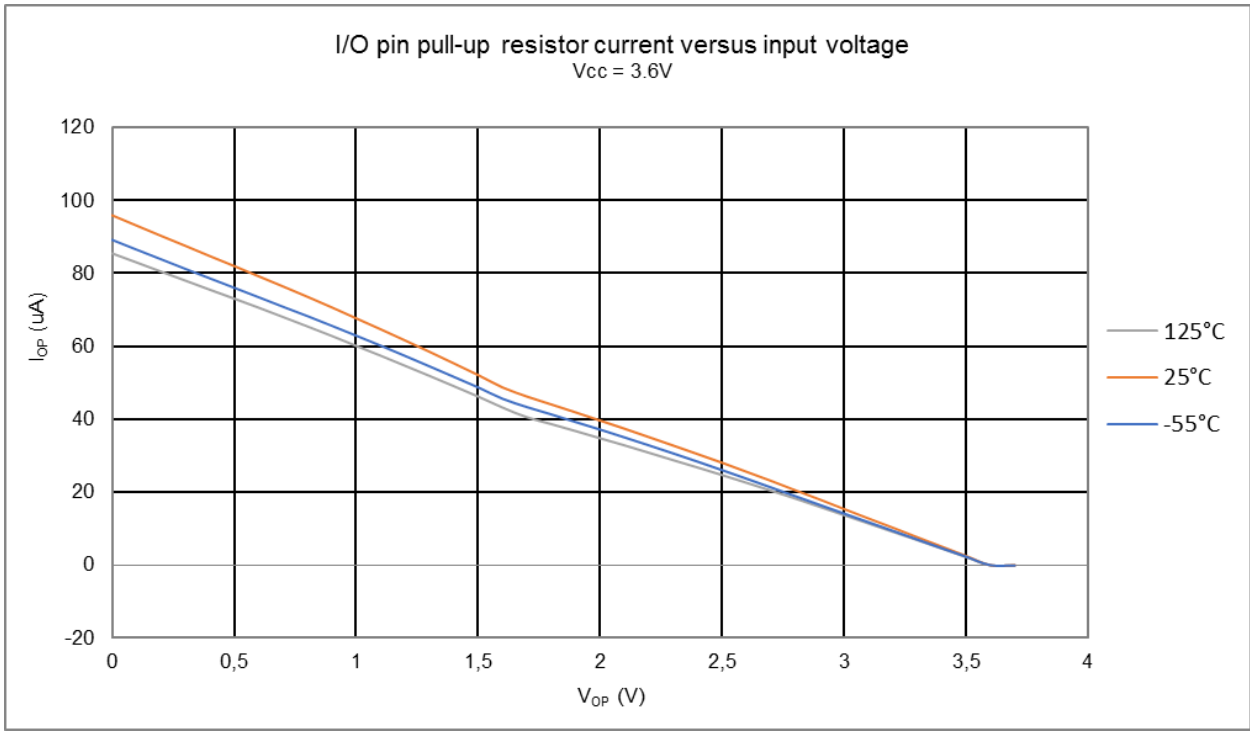


Figure 30-11. I/O Pin Pull-up Resistor Current versus Input Voltage – V_{CC} = 3.6V



30.10.2 Pin Driver Strength

Figure 30-12. I/O Pin Low Output Voltage versus Sink Current – V_{CC} = 3.0V

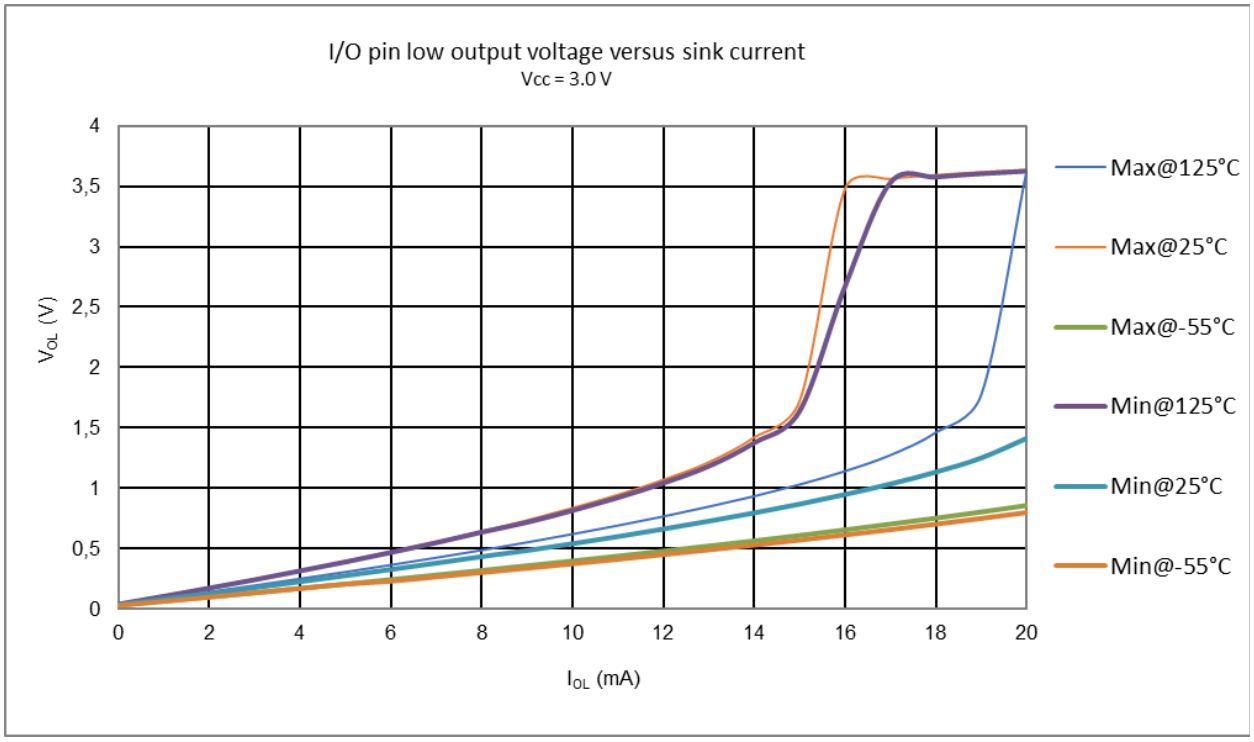
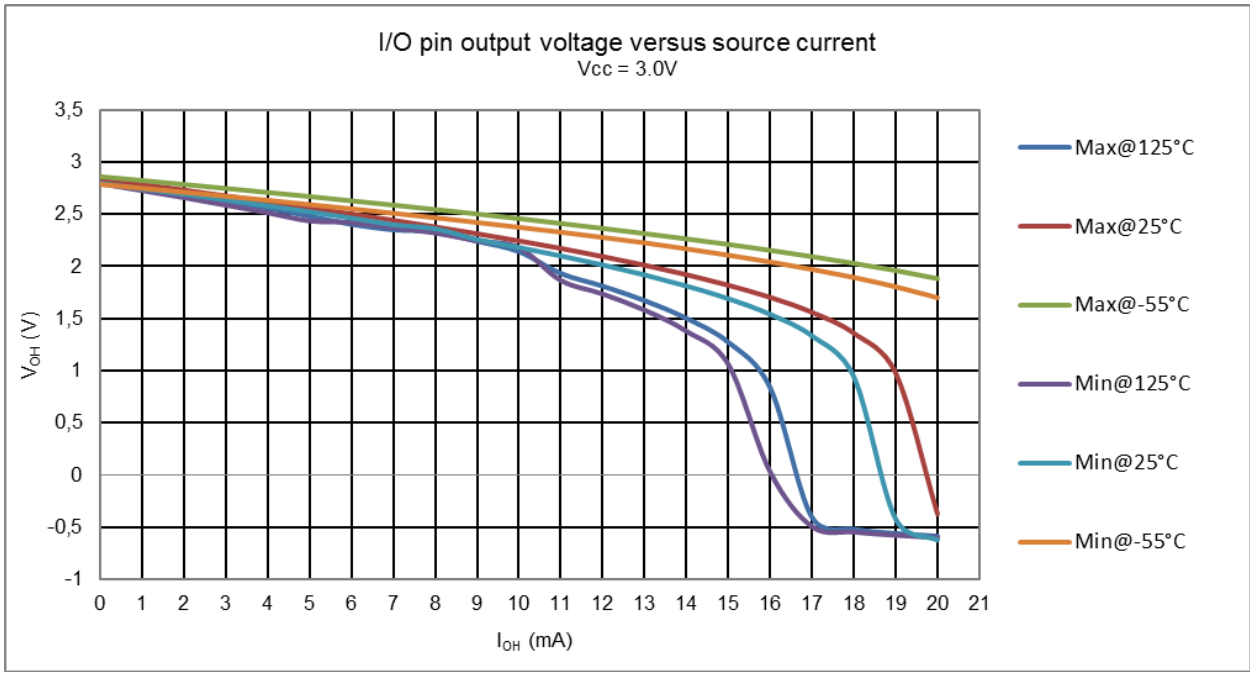


Figure 30-13. I/O Pin Low Output Voltage versus Source Current – V_{CC} = 3.0V



30.10.3 Pin Thresholds and Hysteresis

Figure 30-14. I/O Pin Input Threshold Voltage versus V_{CC} - VIH, I/O Pin Read as '1'

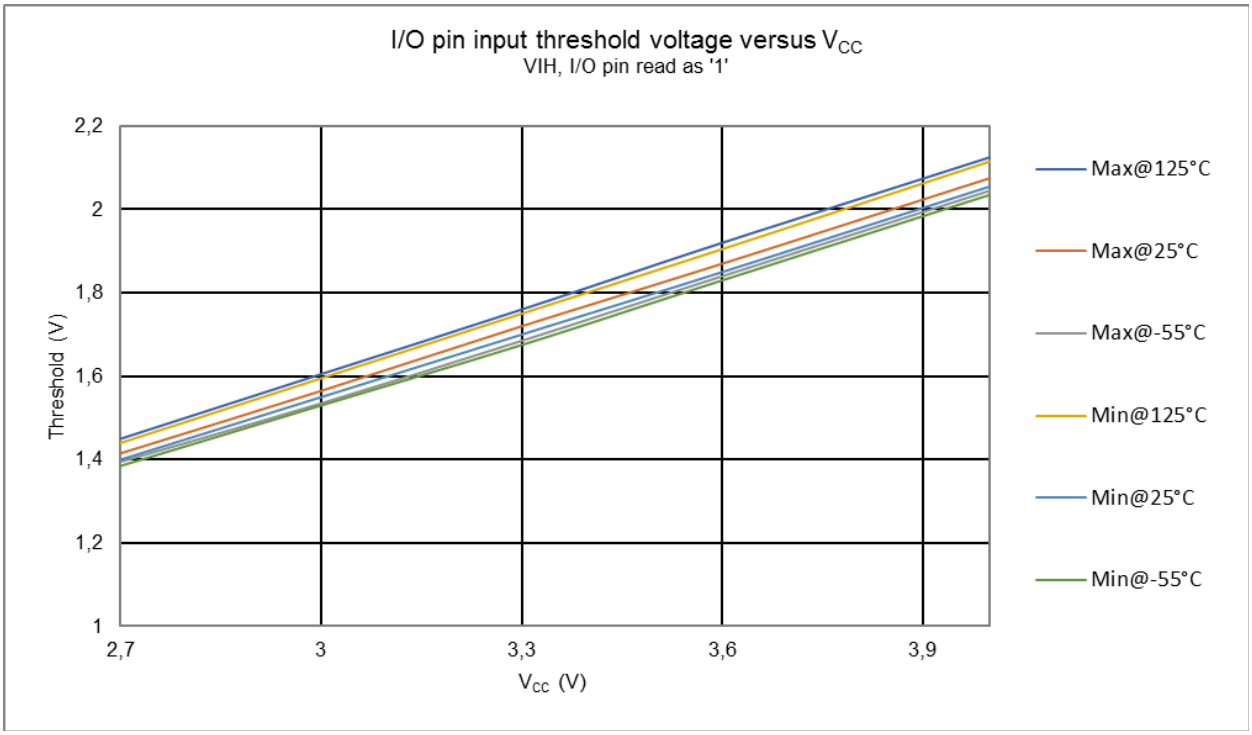


Figure 30-15. I/O Pin Input Threshold Voltage versus V_{CC} - VIH, I/O Pin Read as '0'

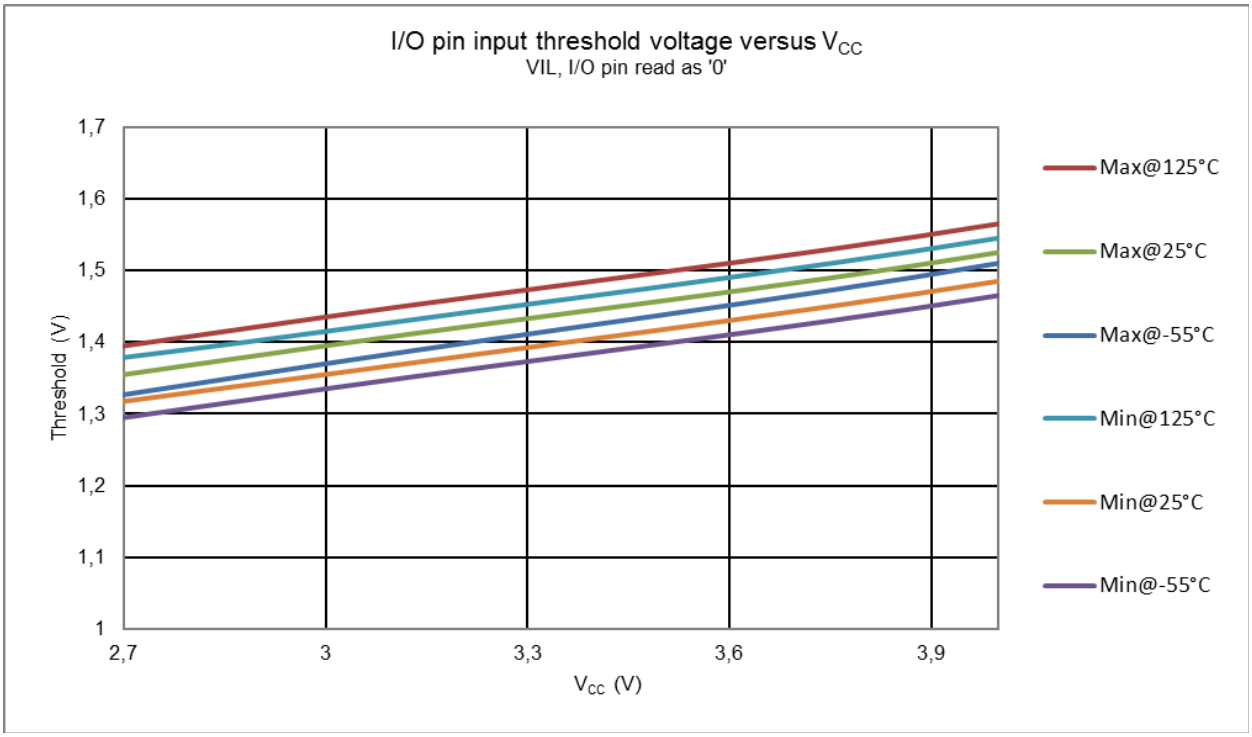


Figure 30-16. I/O Pin Input Hysteresis versus V_{CC}

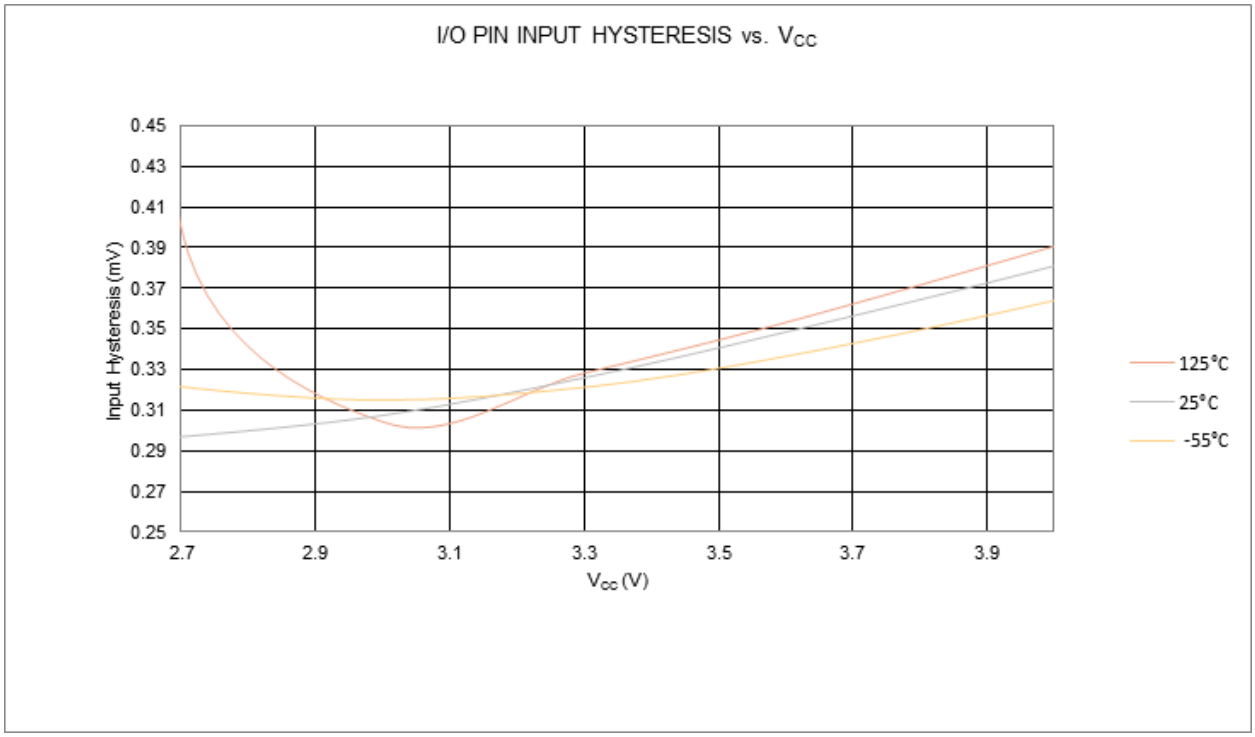


Figure 30-17. Reset Input Threshold Voltage versus V_{CC} - VIH, I/O Pin Read as '1'

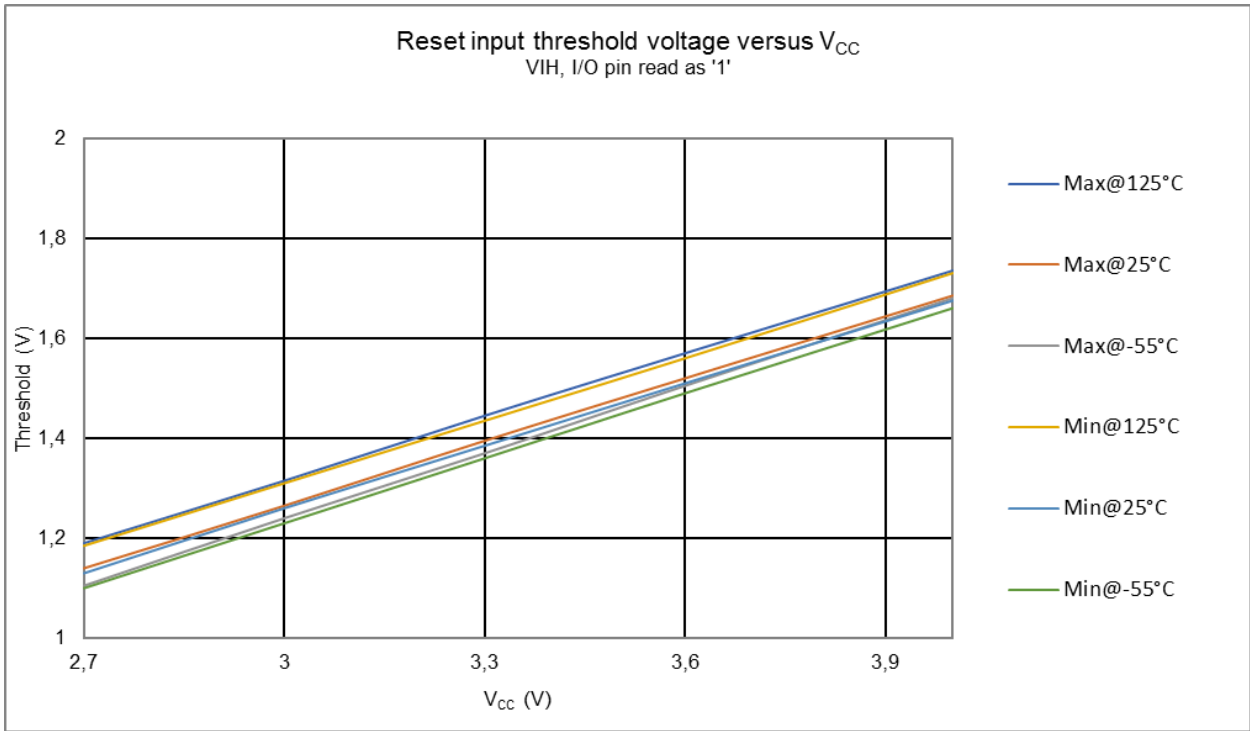


Figure 30-18. Reset Input Threshold Voltage versus V_{CC} - VIL, I/O Pin Read as '0'

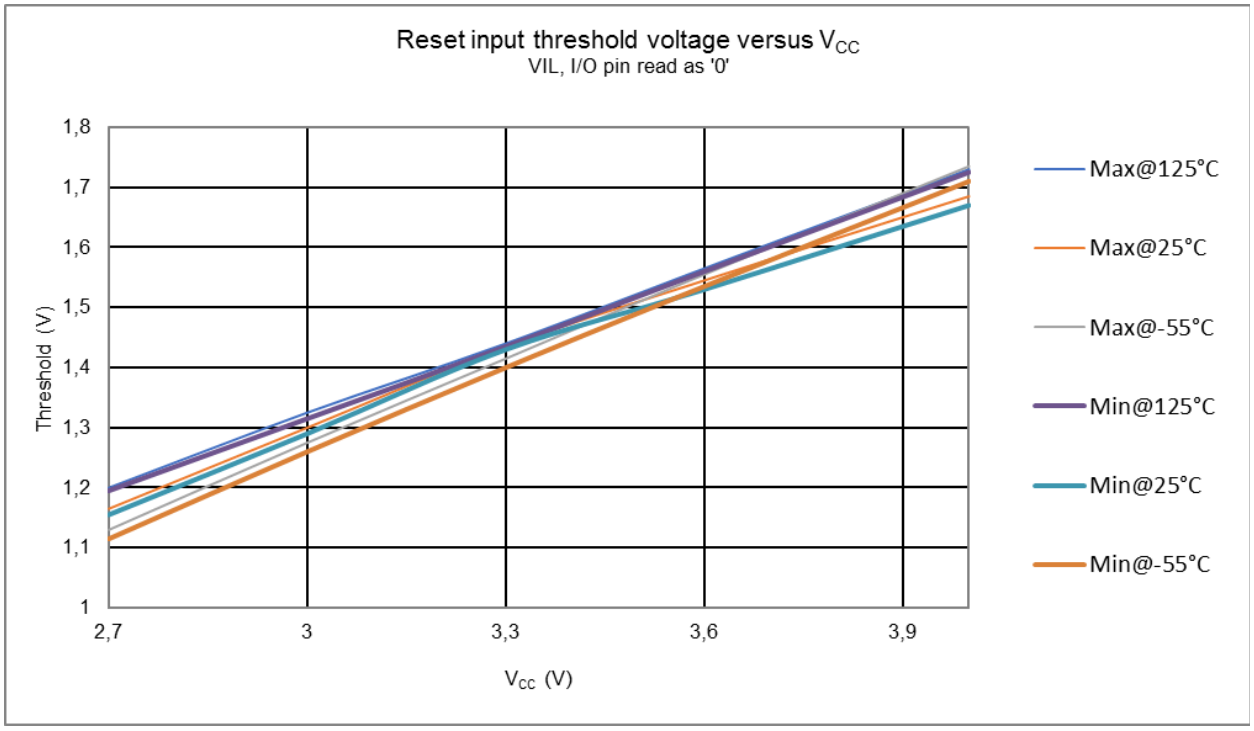


Figure 30-19. Reset Pull-up Resistor Current versus Reset Pin Voltage – V_{CC} = 3.0V

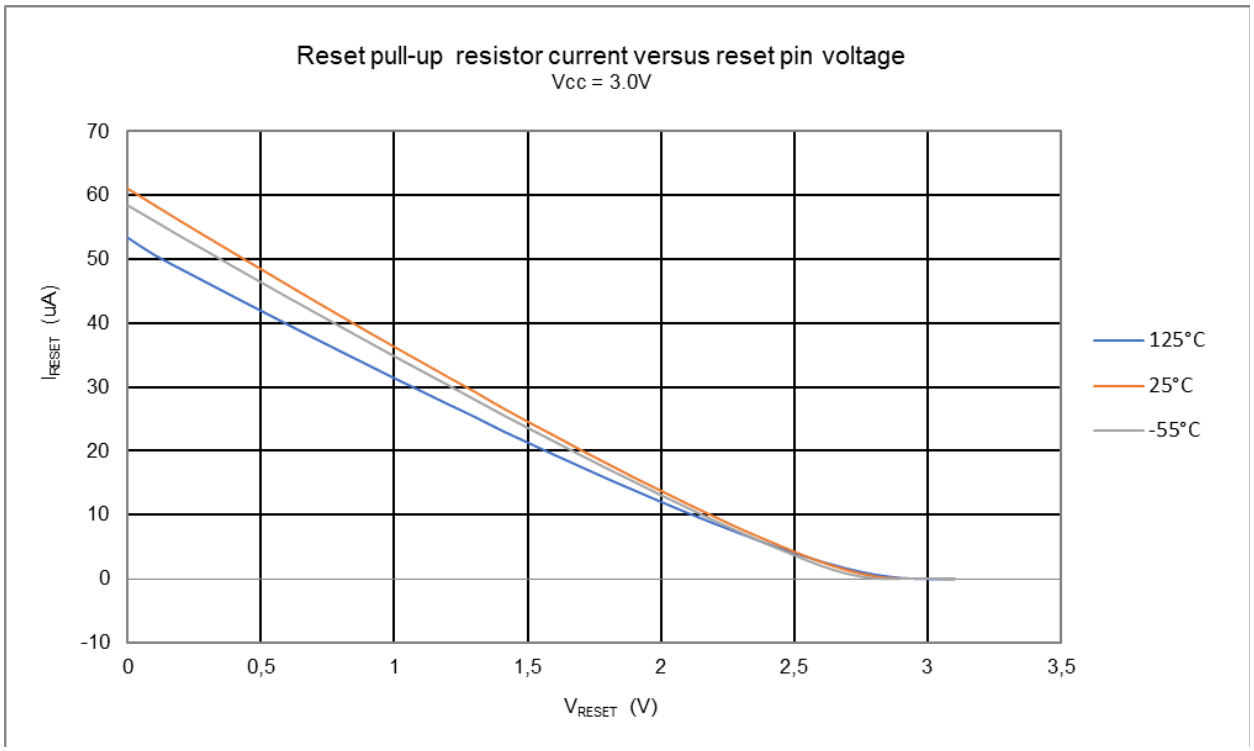


Figure 30-20. Reset Pull-up Resistor Current versus Reset Pin Voltage – V_{CC} = 3.6V

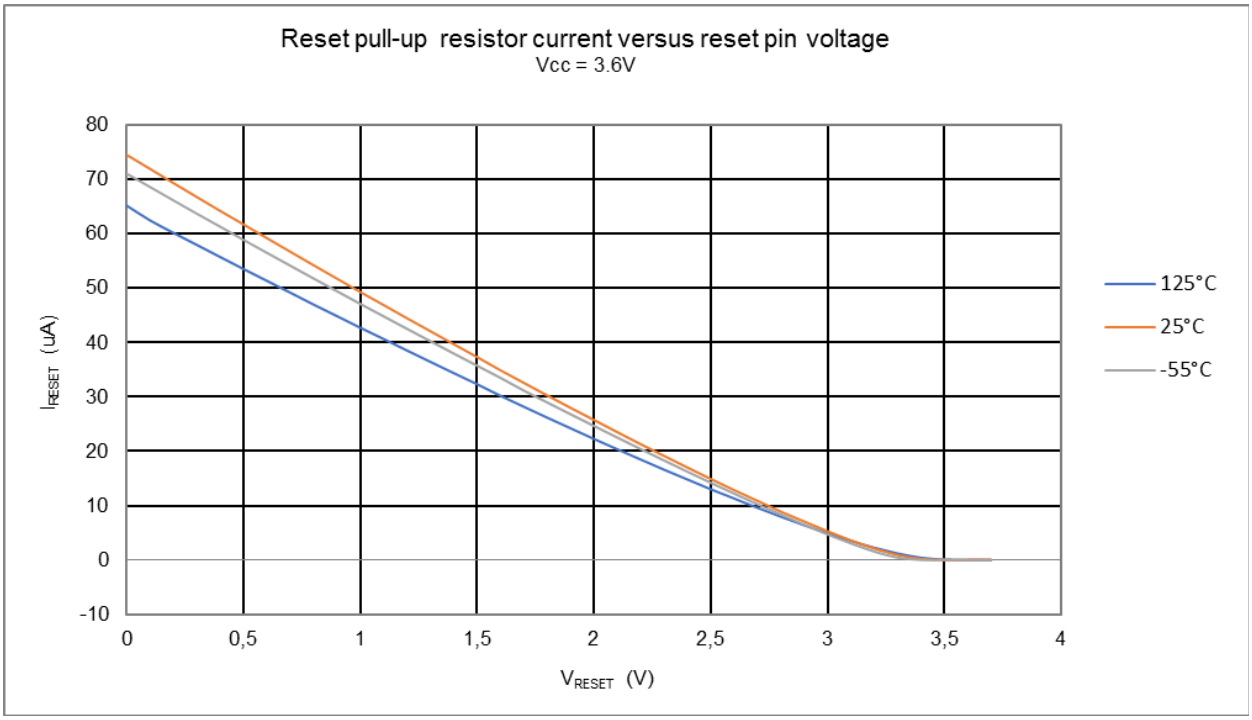


Figure 30-21. XTAL1 Input Threshold Voltage versus V_{CC} – VIL, XTAL1 Pin Read as '0'

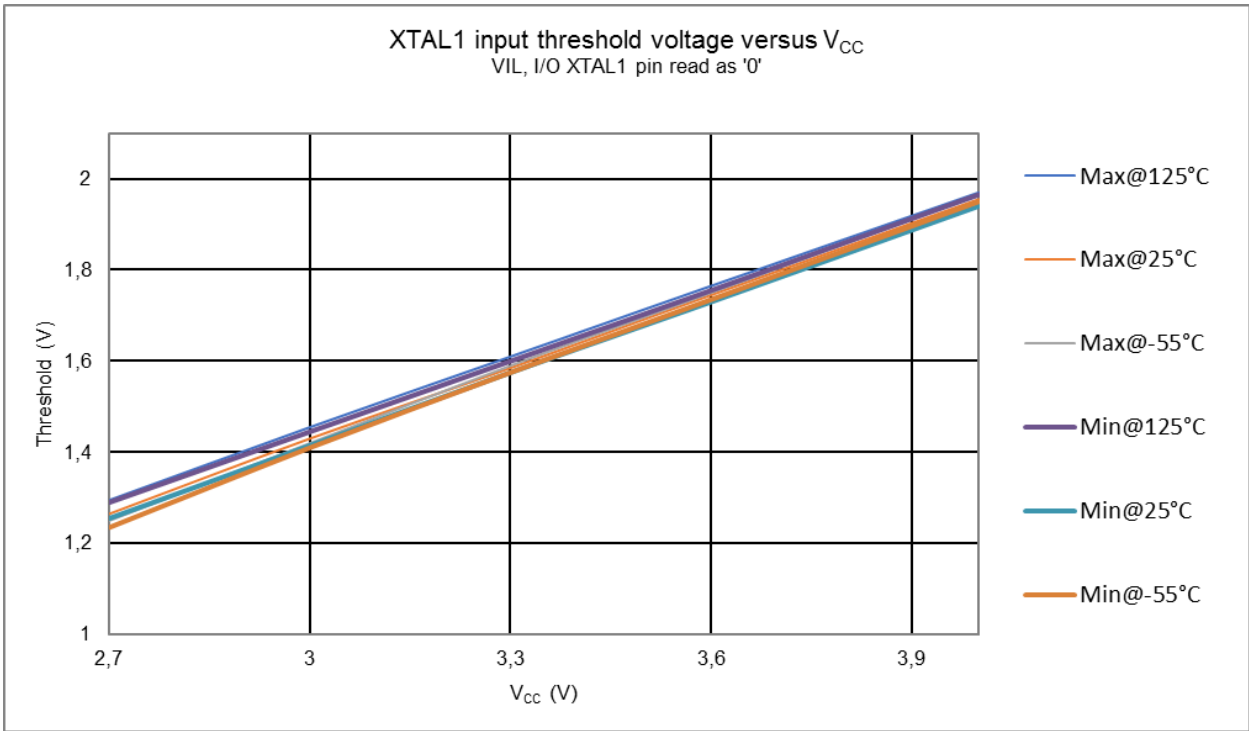
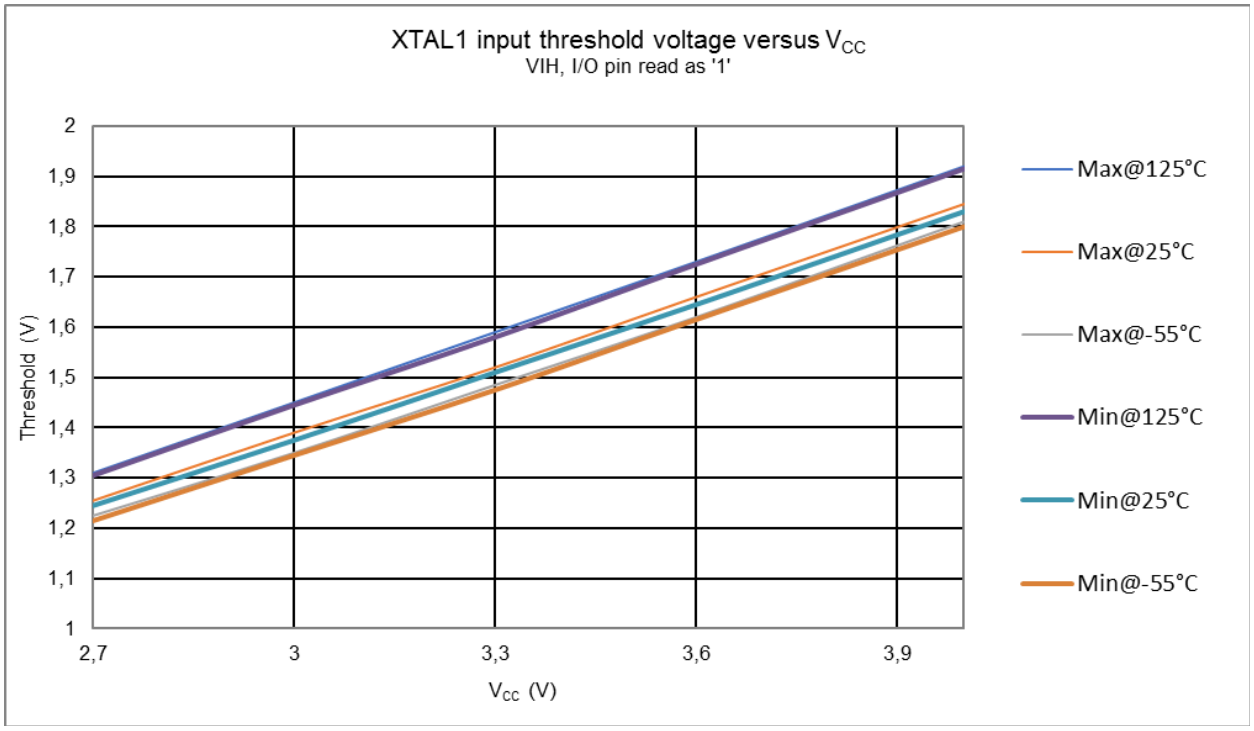
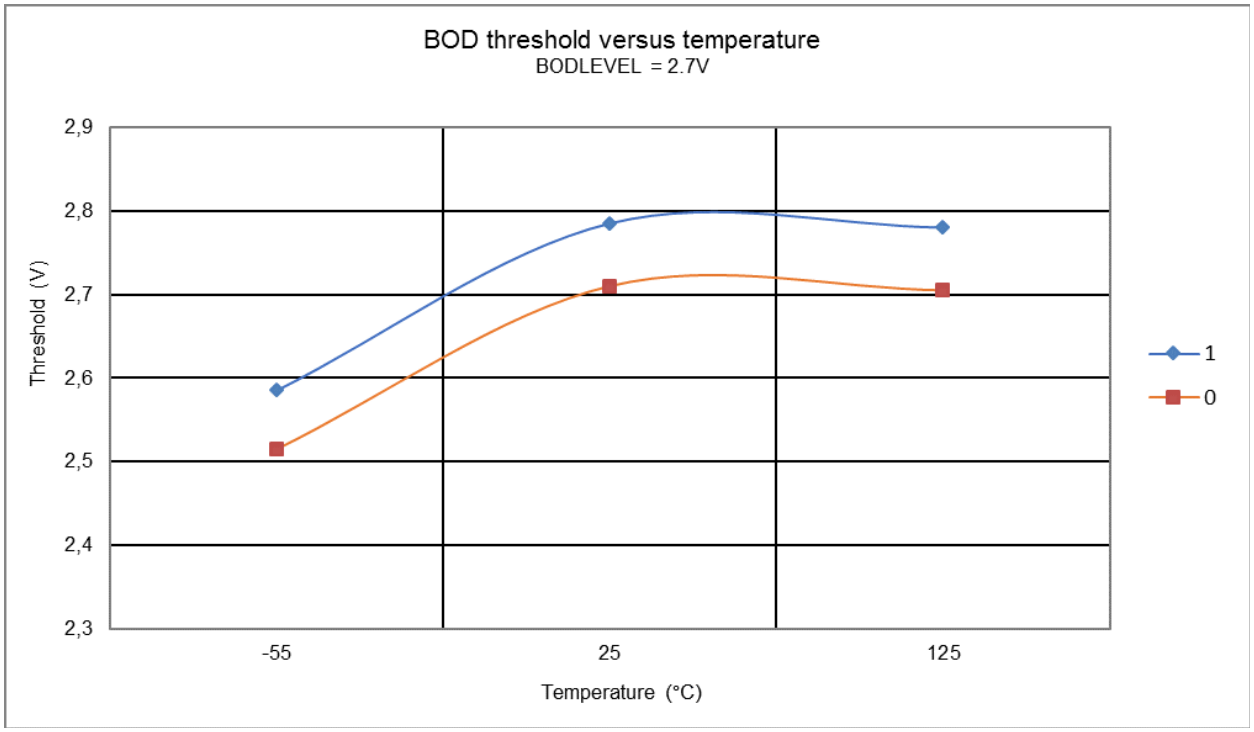


Figure 30-22. XTAL1 Input Threshold Voltage versus V_{CC} – VIL, XTAL1 Pin Read as '1'



30.10.4 BOD Thresholds and Analog Comparator Hysteresis

Figure 30-23. BOD Threshold versus Temperature – BODLEVEL = 2.7V



30.10.5 Internal Oscillator Speed

Figure 30-24. Watchdog Oscillator Frequency versus V_{CC}

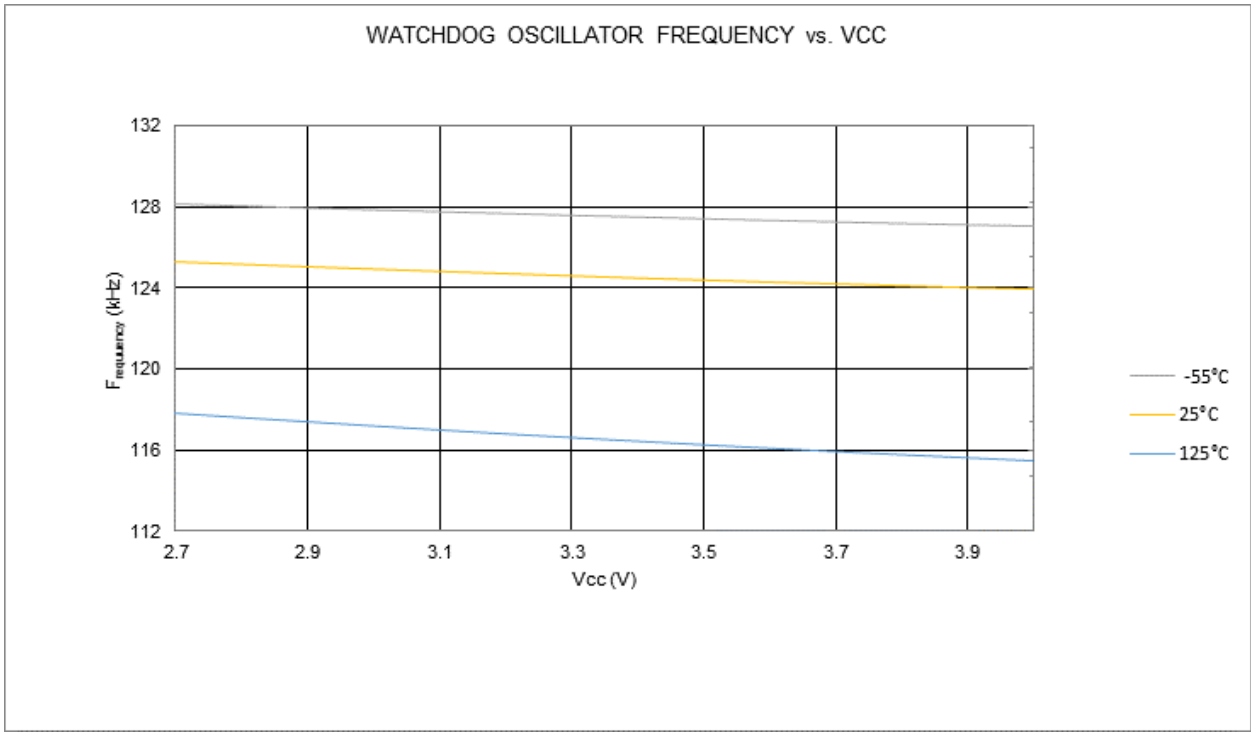


Figure 30-25. Watchdog Oscillator Frequency versus Temperature

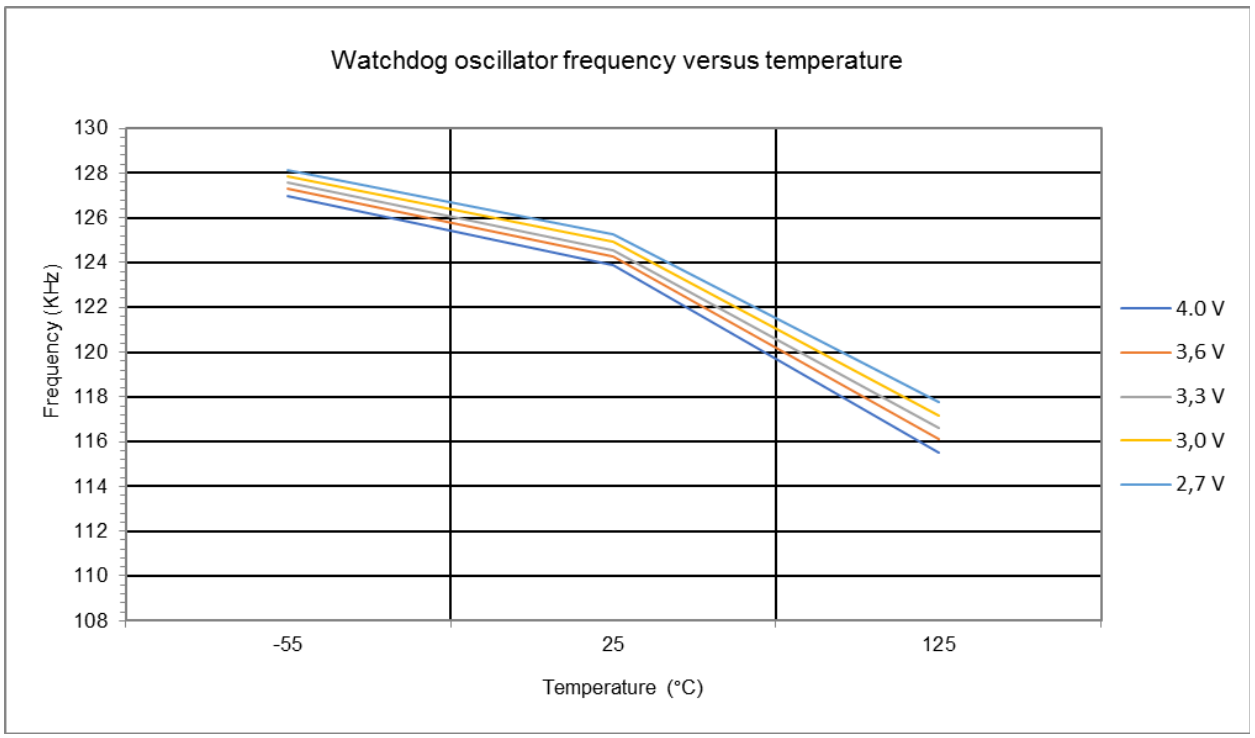


Figure 30-26. Calibrated 8MHz Oscillator Frequency versus V_{CC} – max

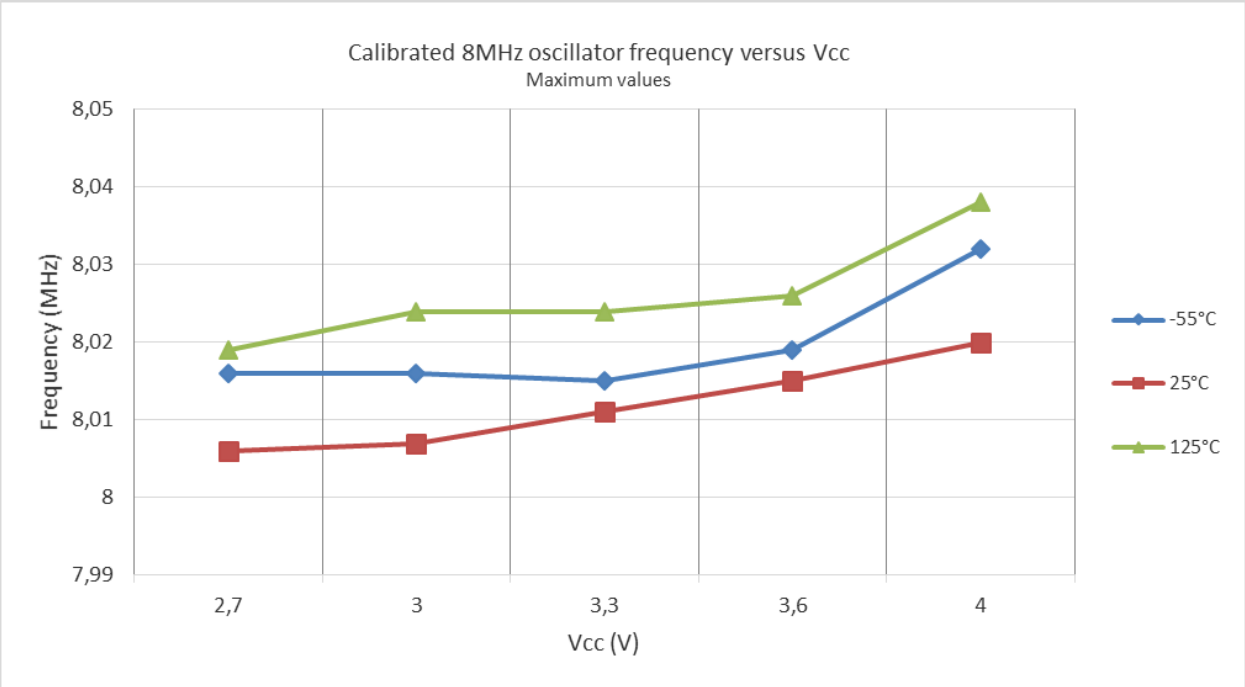


Figure 30-27. Calibrated 8MHz Oscillator Frequency versus V_{CC} – average

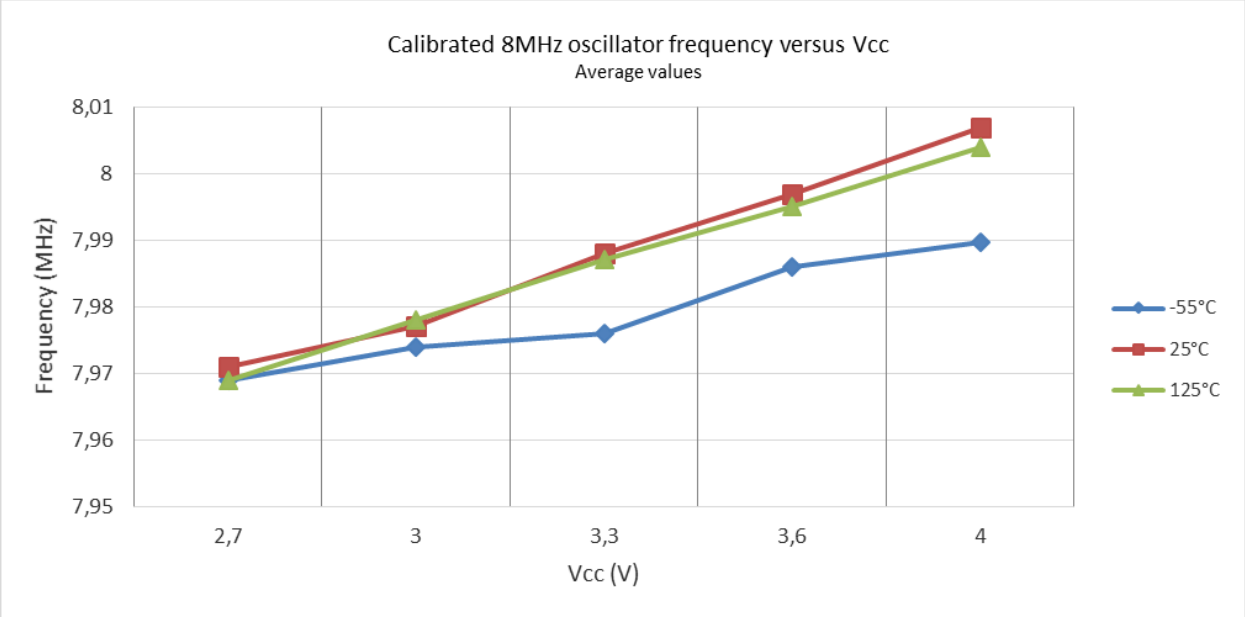


Figure 30-28. Calibrated 8MHz Oscillator Frequency versus V_{CC} – min

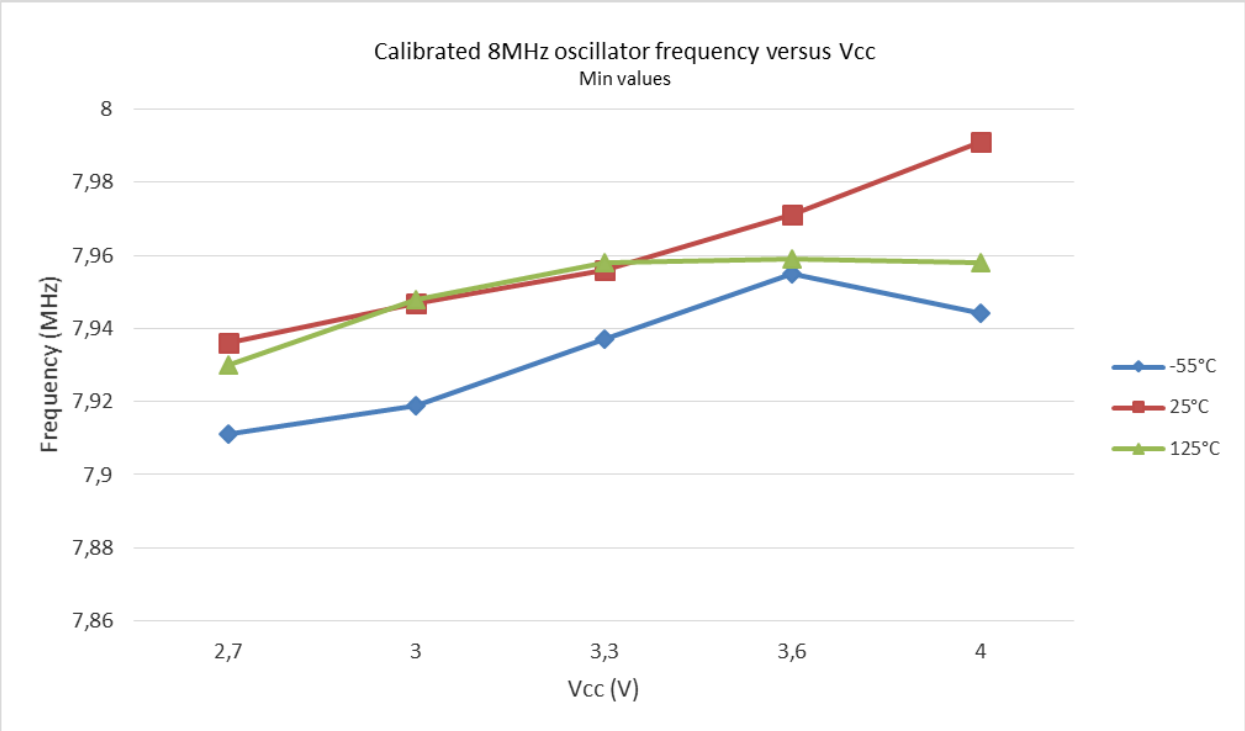
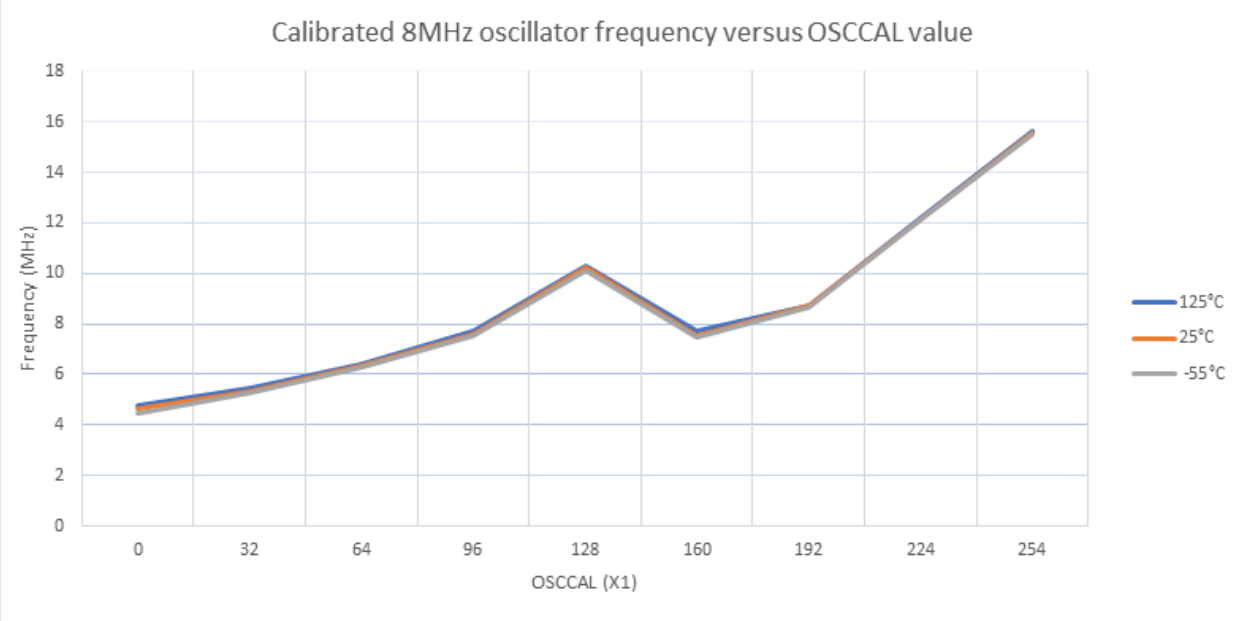


Figure 30-29. Calibrated 8MHz Oscillator Frequency versus OSCCAL Value



31. Typical Characteristics

All DC characteristics contained in this datasheet are based on characterization data. 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 register set and thus, the corresponding I/O modules are turned off. In addition, 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 determinant factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L x V_{CC} x 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.

31.1 Pin Pull-Up

Figure 31-1. I/O Pin Pull-up Resistor Current versus Input Voltage – V_{CC} = 3.0V

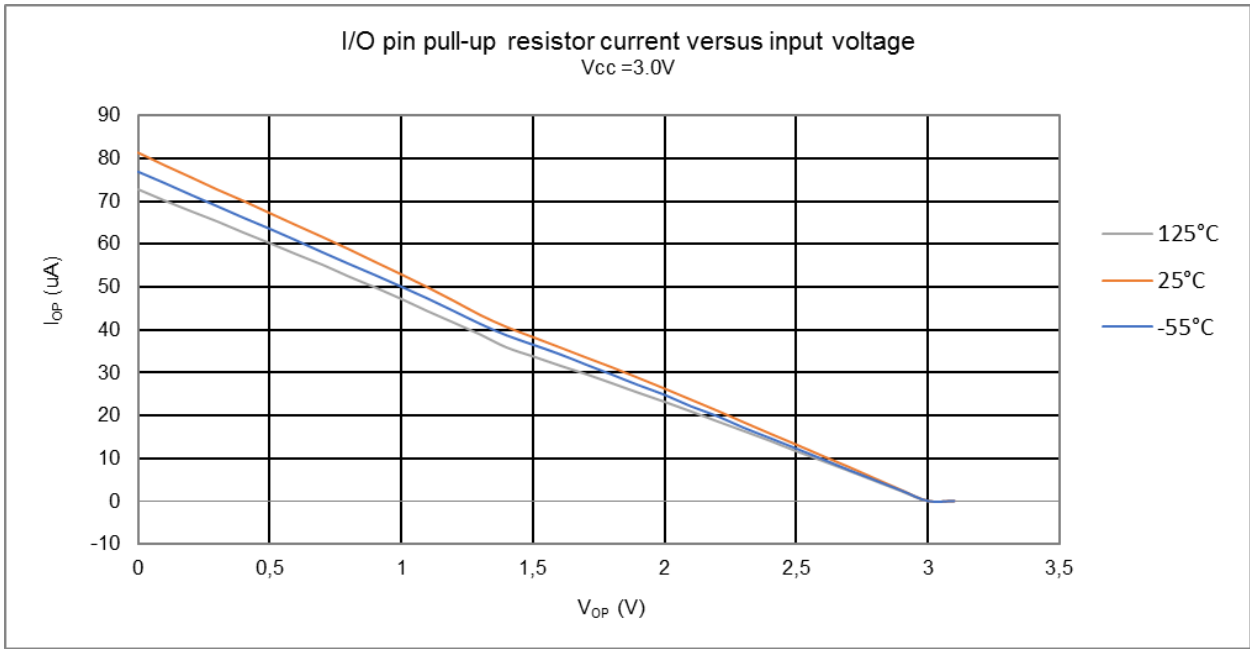
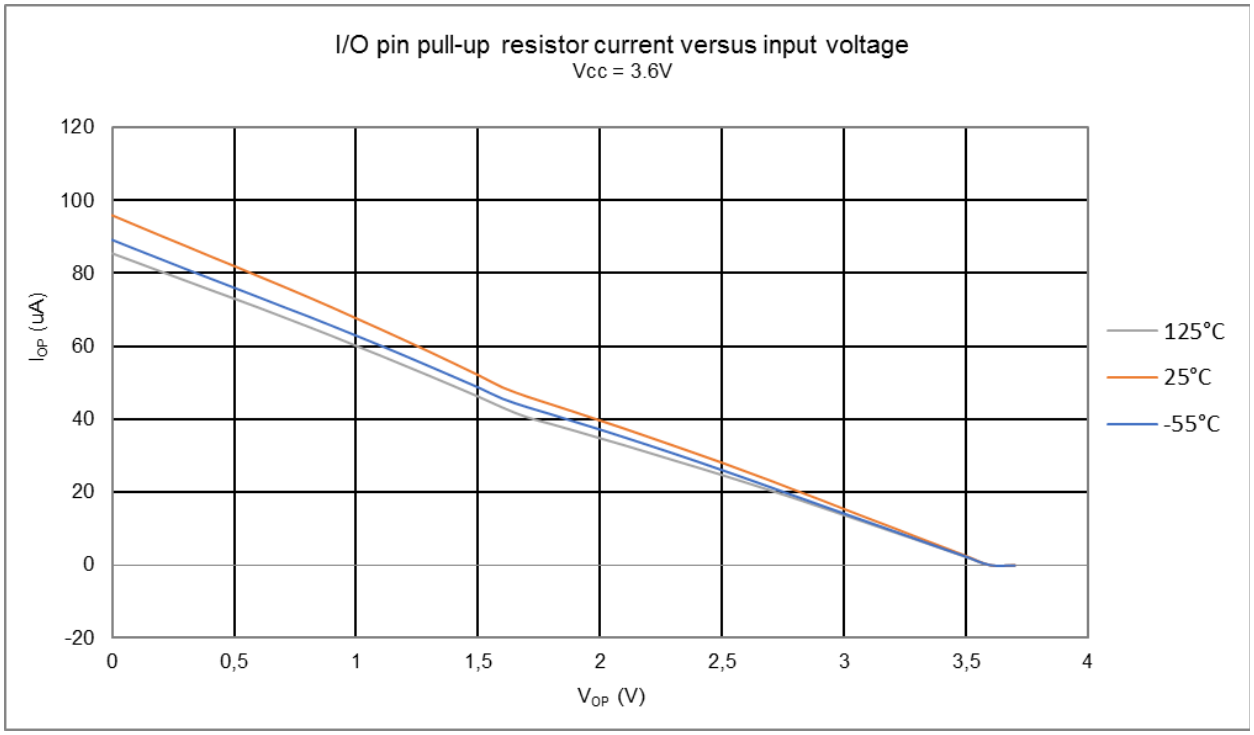


Figure 31-2. I/O Pin Pull-up Resistor Current versus Input Voltage – V_{CC} = 3.6V



31.2 Pin Driver Strength

Figure 31-3. I/O Pin Low Output Voltage versus Sink Current – V_{CC} = 3.0V

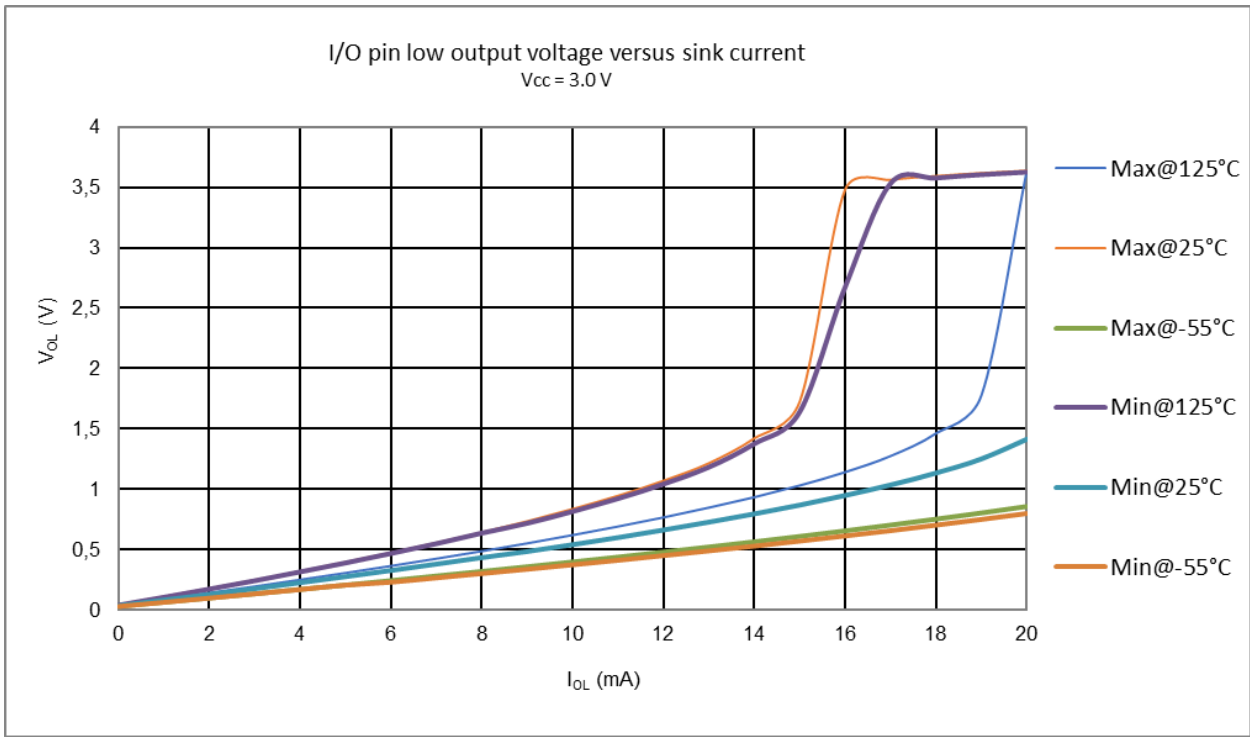
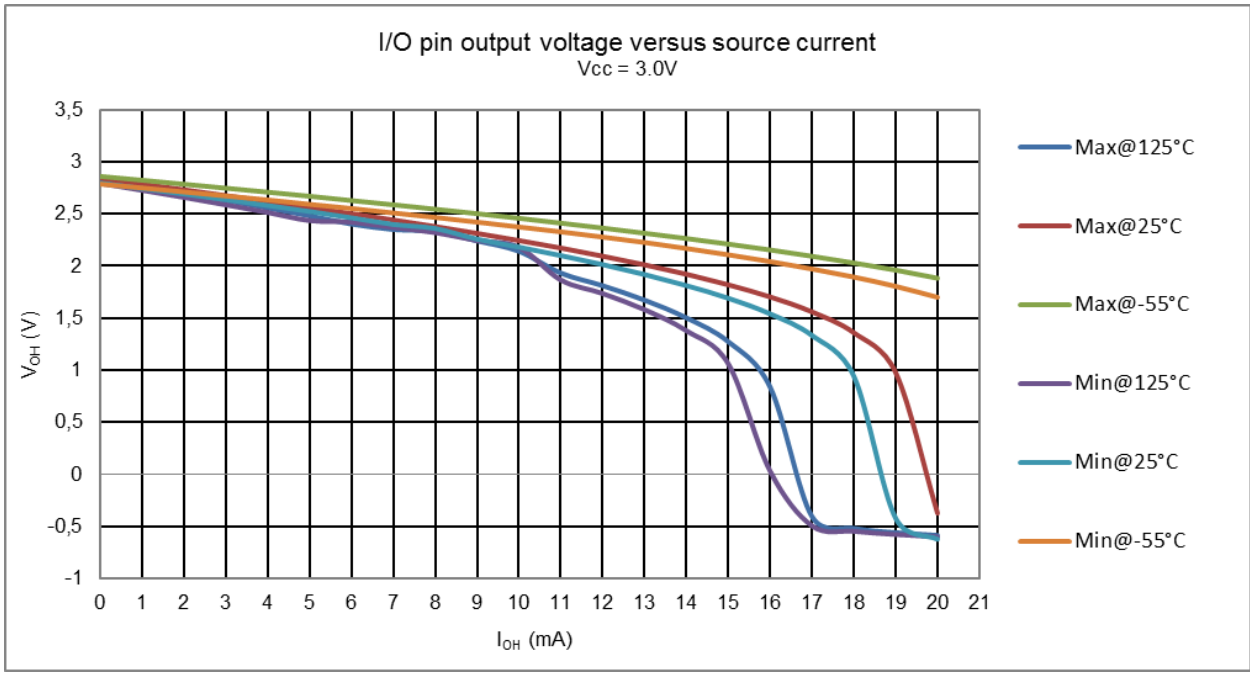


Figure 31-4. I/O Pin Low Output Voltage versus Source Current – V_{CC} = 3.0V



31.3 Pin Thresholds and Hysteresis

Figure 31-5. I/O Pin Input Threshold Voltage versus V_{CC} - VIH, I/O Pin Read as '1'

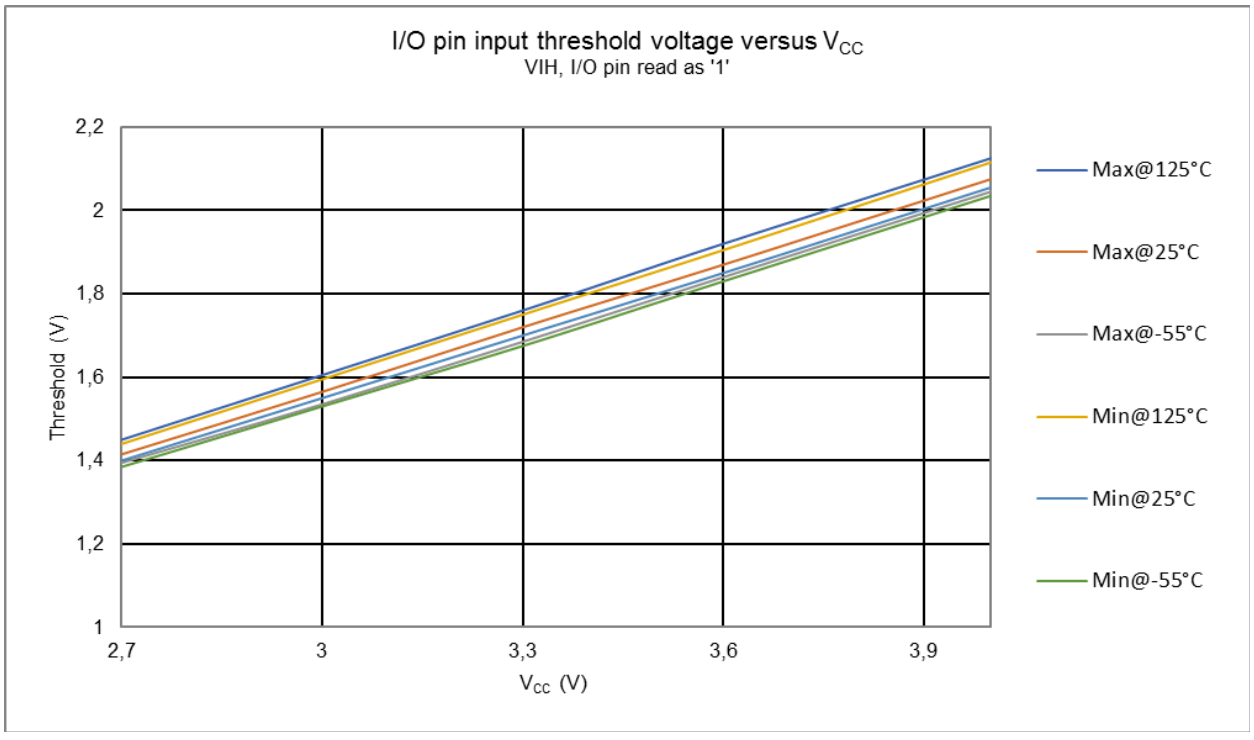


Figure 31-6. I/O Pin Input Threshold Voltage versus V_{CC} - VIH, I/O Pin Read as '0'

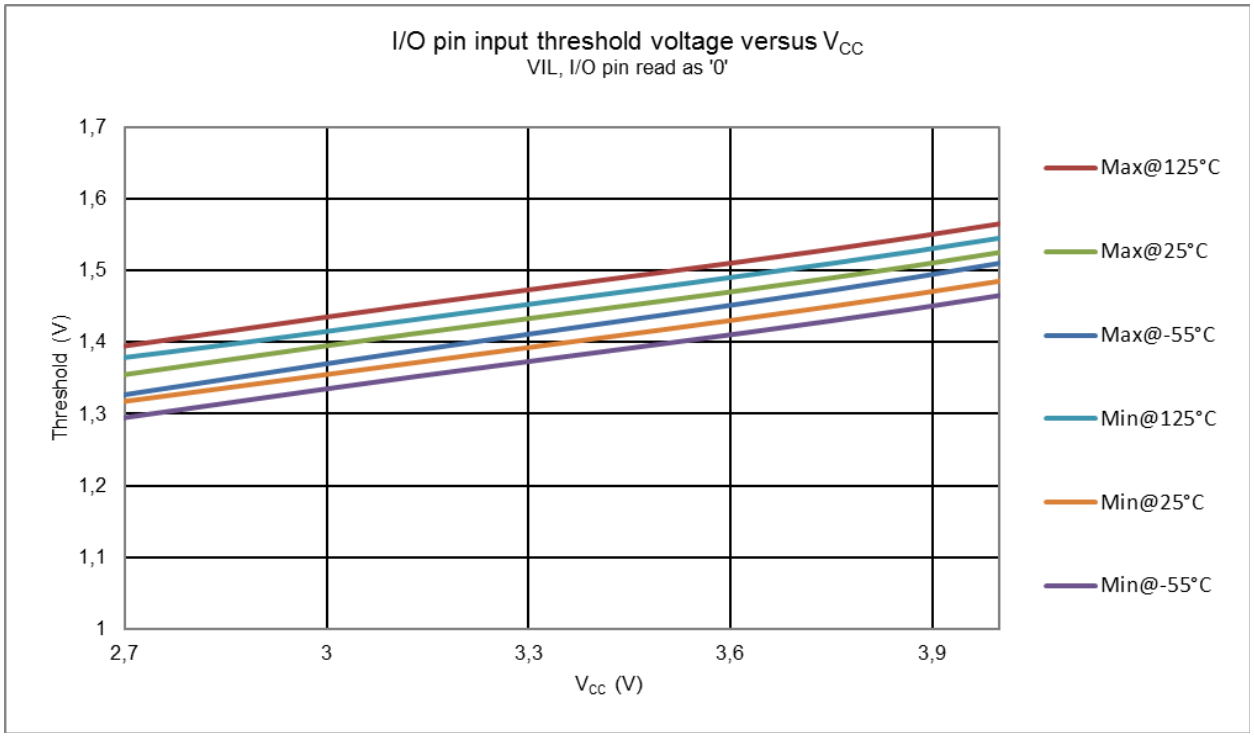


Figure 31-7. I/O Pin Input Hysteresis versus V_{CC}

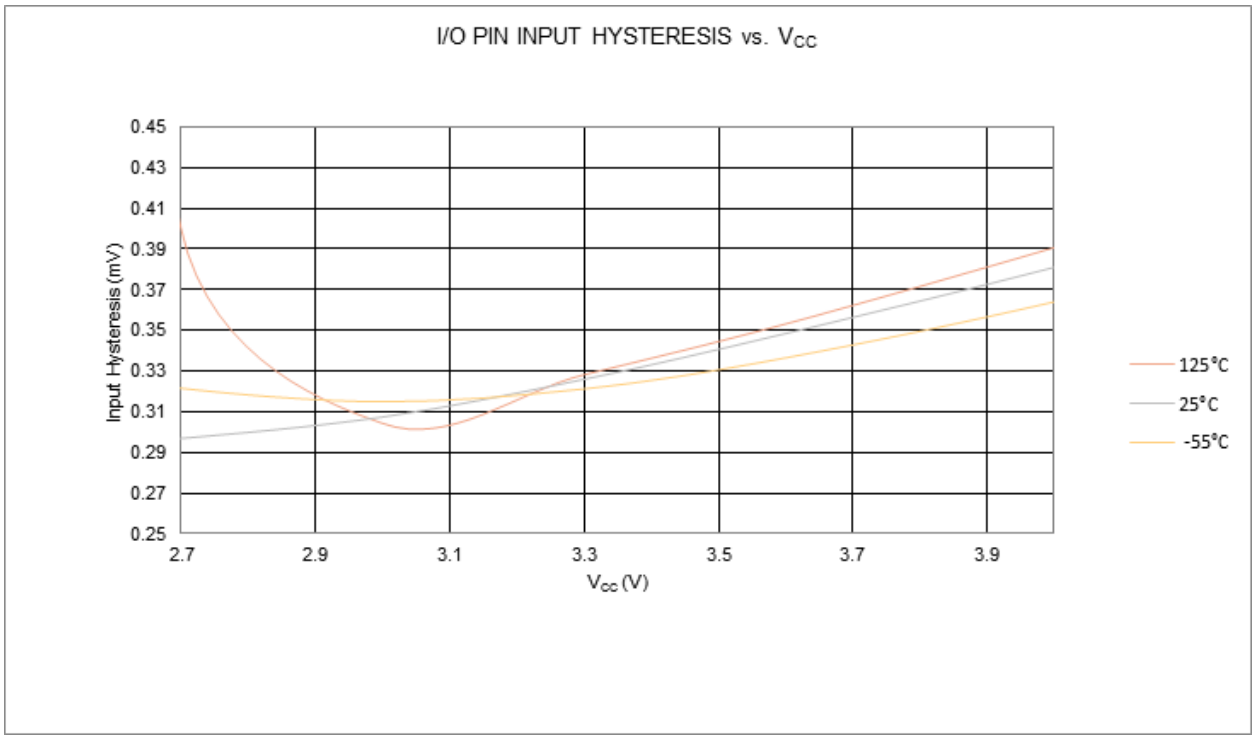


Figure 31-8. Reset Input Threshold Voltage versus V_{CC} - VIH, I/O Pin Read as '1'

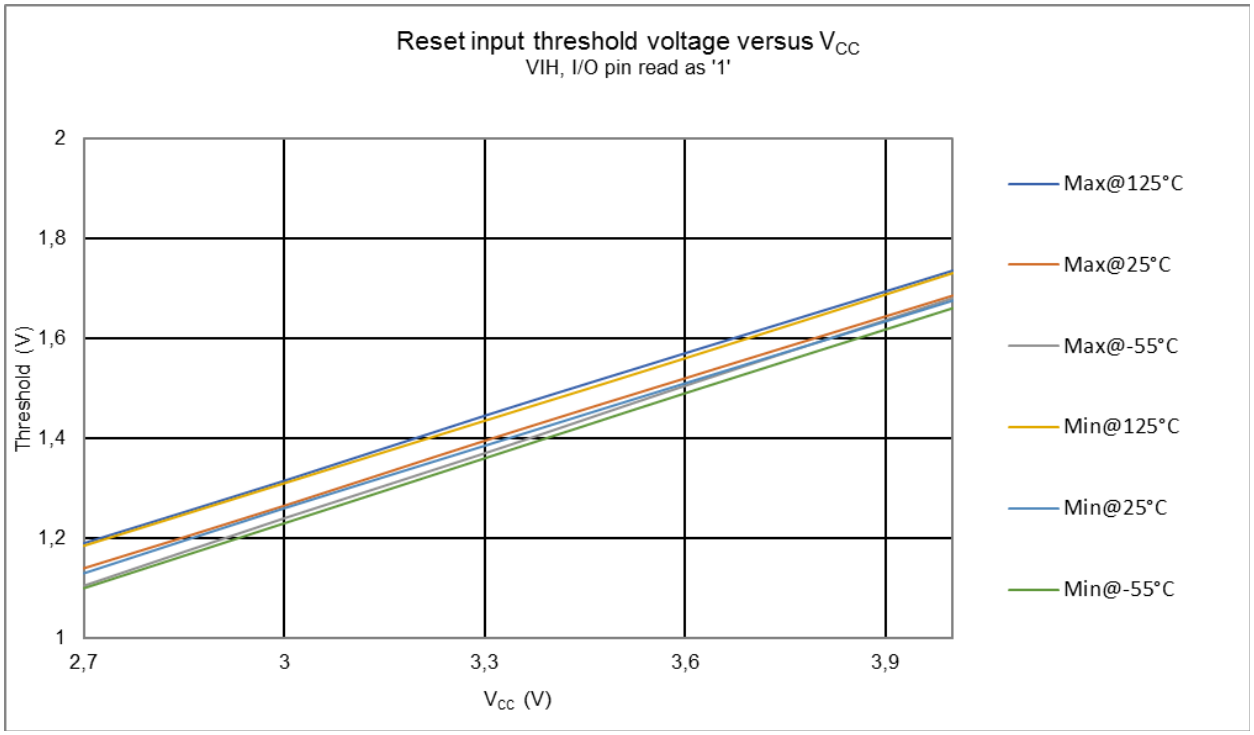


Figure 31-9. Reset Input Threshold Voltage versus V_{CC} - VIL, I/O Pin Read as '0'

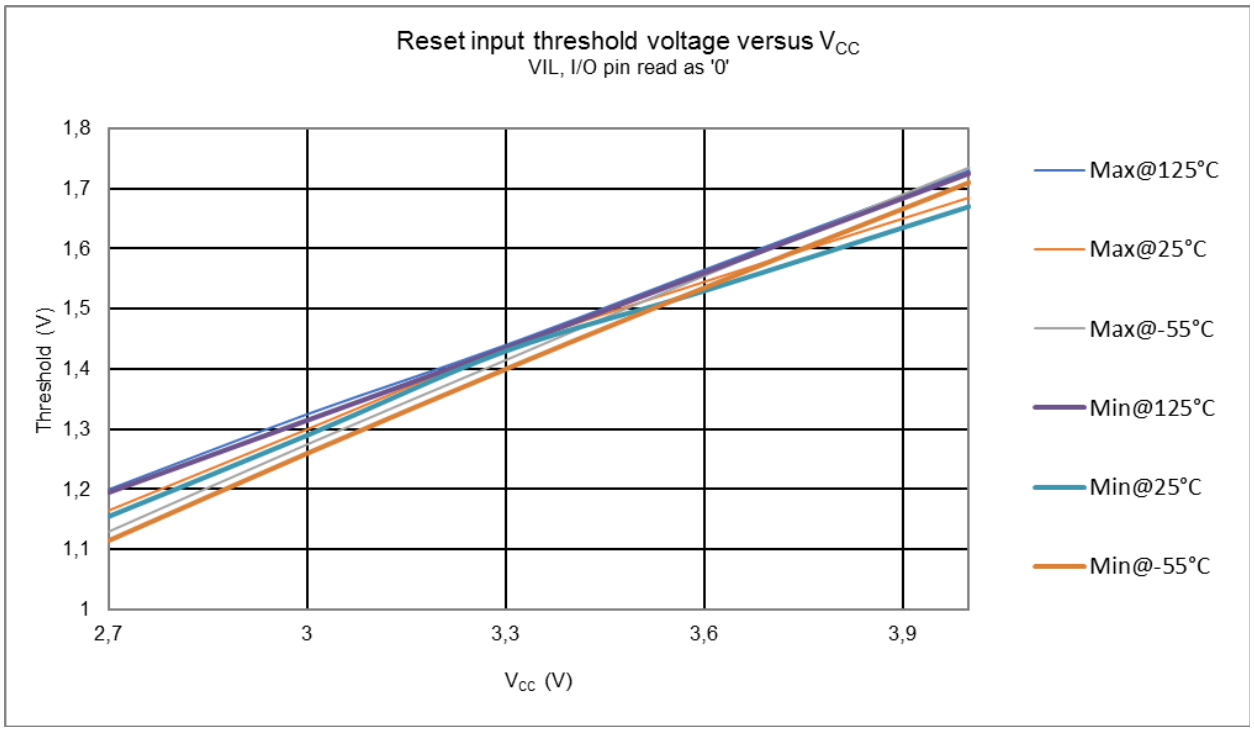


Figure 31-10. Reset Pull-up Resistor Current versus Reset Pin Voltage – V_{CC} = 3.0V

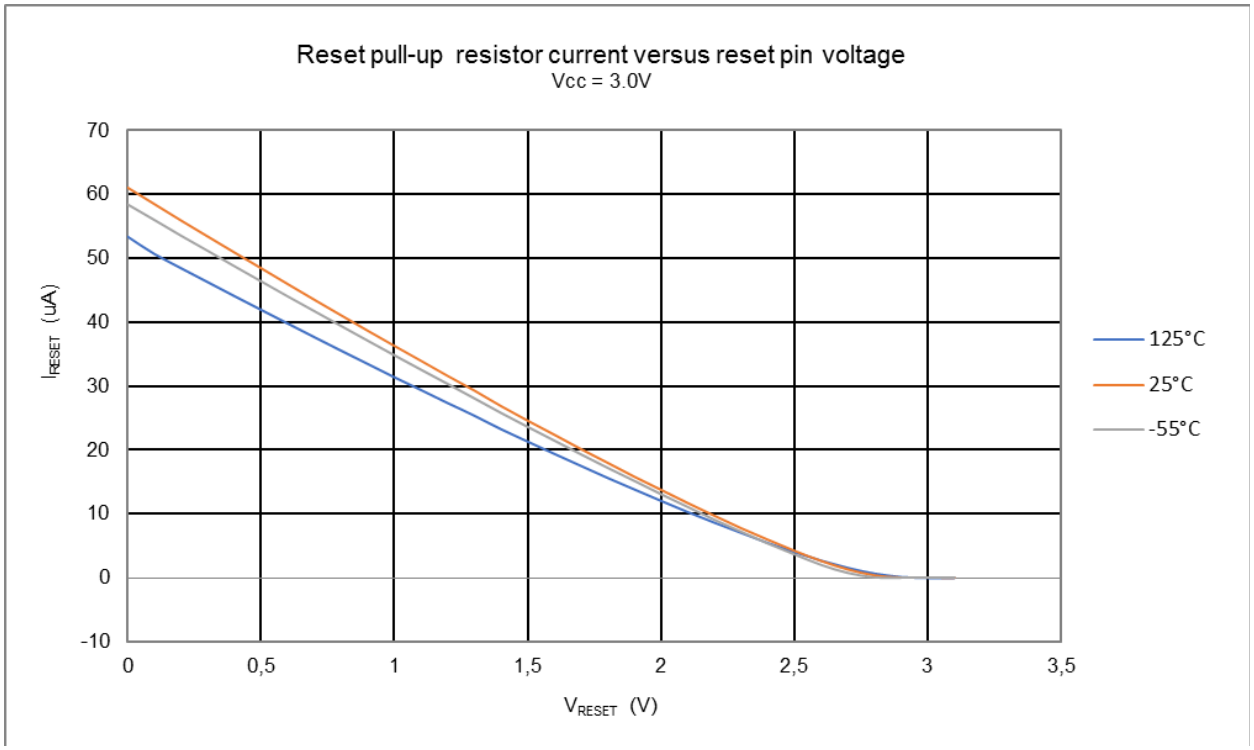


Figure 31-11. Reset Pull-up Resistor Current versus Reset Pin Voltage – V_{CC} = 3.6V

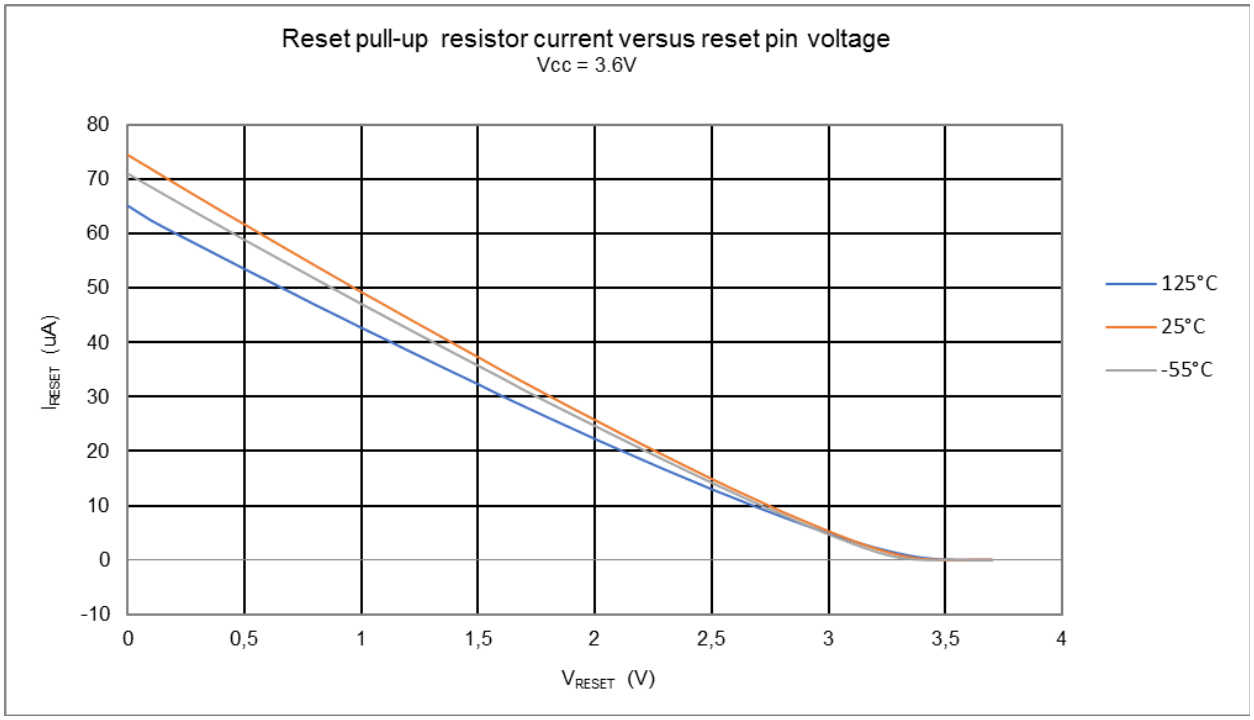


Figure 31-12. XTAL1 Input Threshold Voltage versus V_{CC} – VIL, XTAL1 Pin Read as '0'

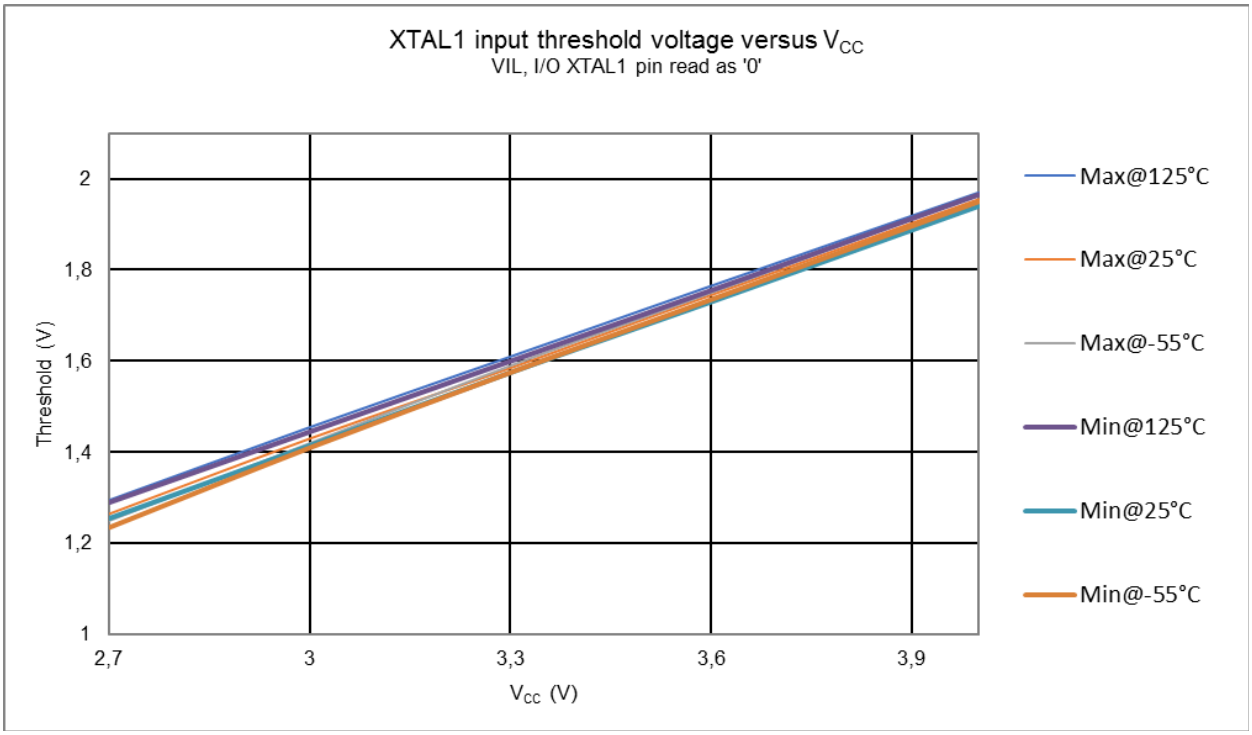
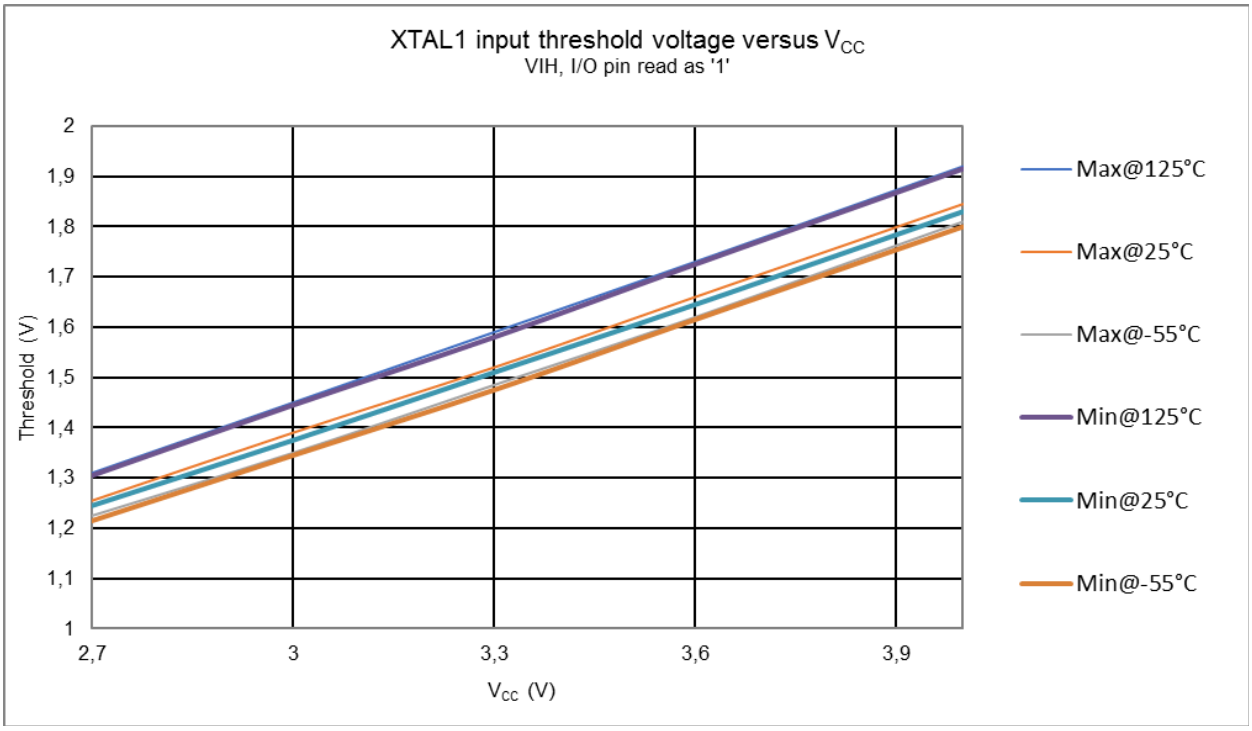
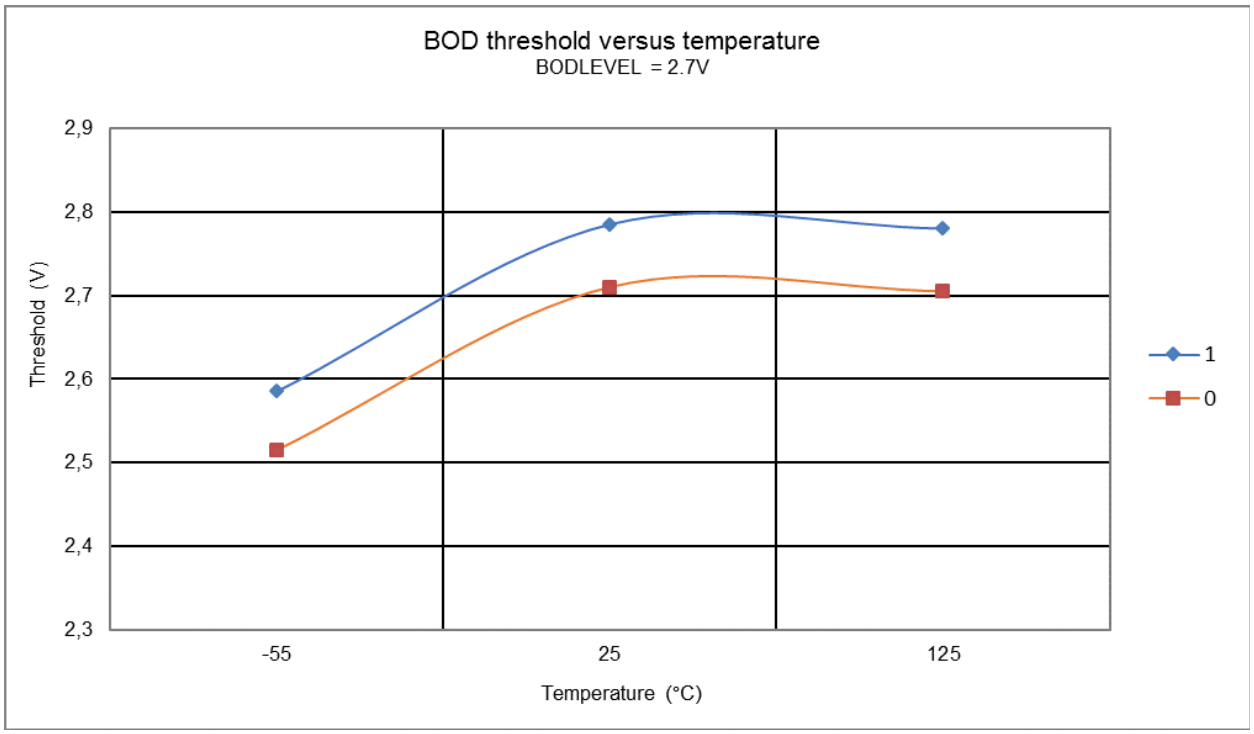


Figure 31-13. XTAL1 Input Threshold Voltage versus V_{CC} – VIL, XTAL1 Pin Read as '1'



31.4 BOD Thresholds and Analog Comparator Hysteresis

Figure 31-14. BOD Threshold versus Temperature – BODLEVEL = 2.7V



31.5 Internal Oscillator Speed

Figure 31-15. Watchdog Oscillator Frequency versus V_{CC}

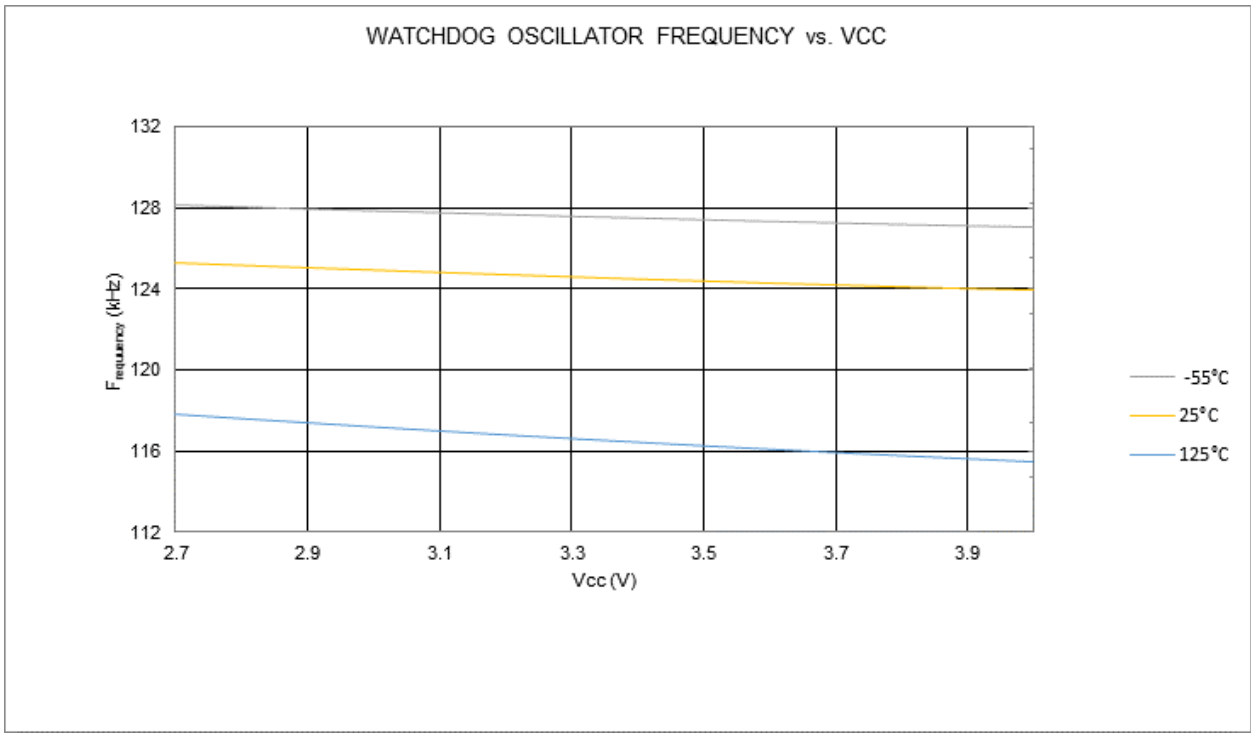


Figure 31-16. Watchdog Oscillator Frequency versus Temperature

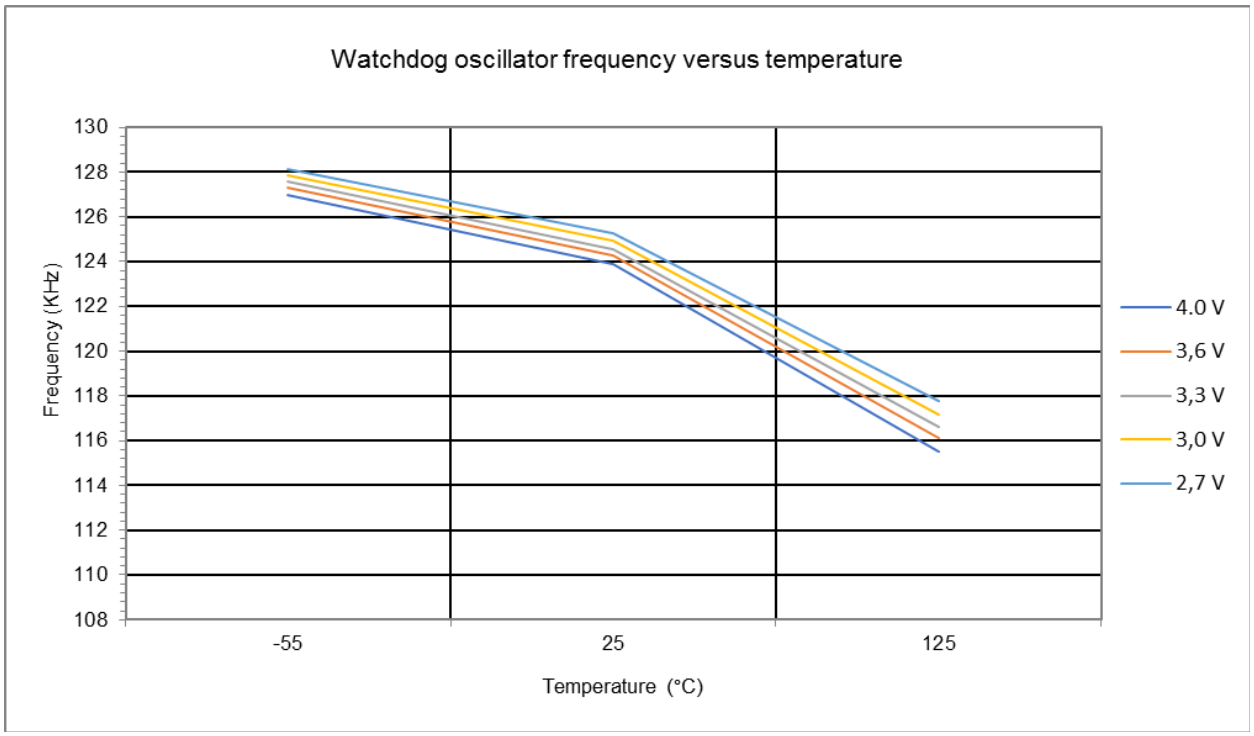


Figure 31-17. Calibrated 8MHz Oscillator Frequency versus V_{CC} – max

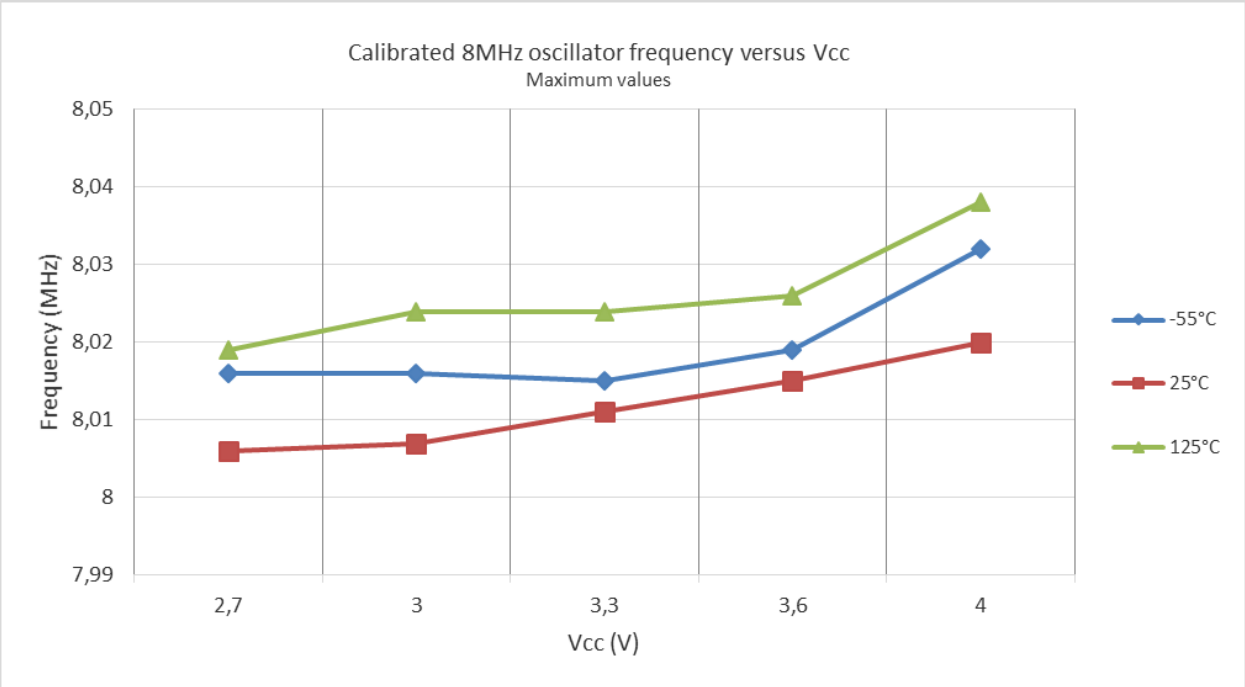


Figure 31-18. Calibrated 8MHz Oscillator Frequency versus V_{CC} – average

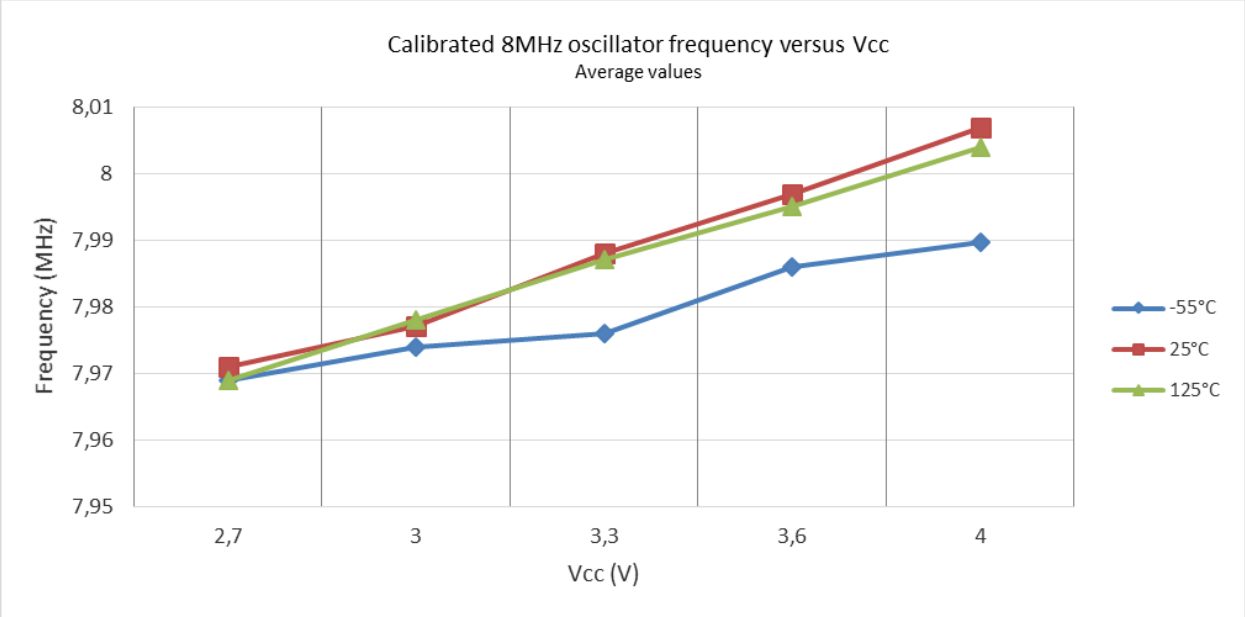


Figure 31-19. Calibrated 8MHz Oscillator Frequency versus V_{CC} – min

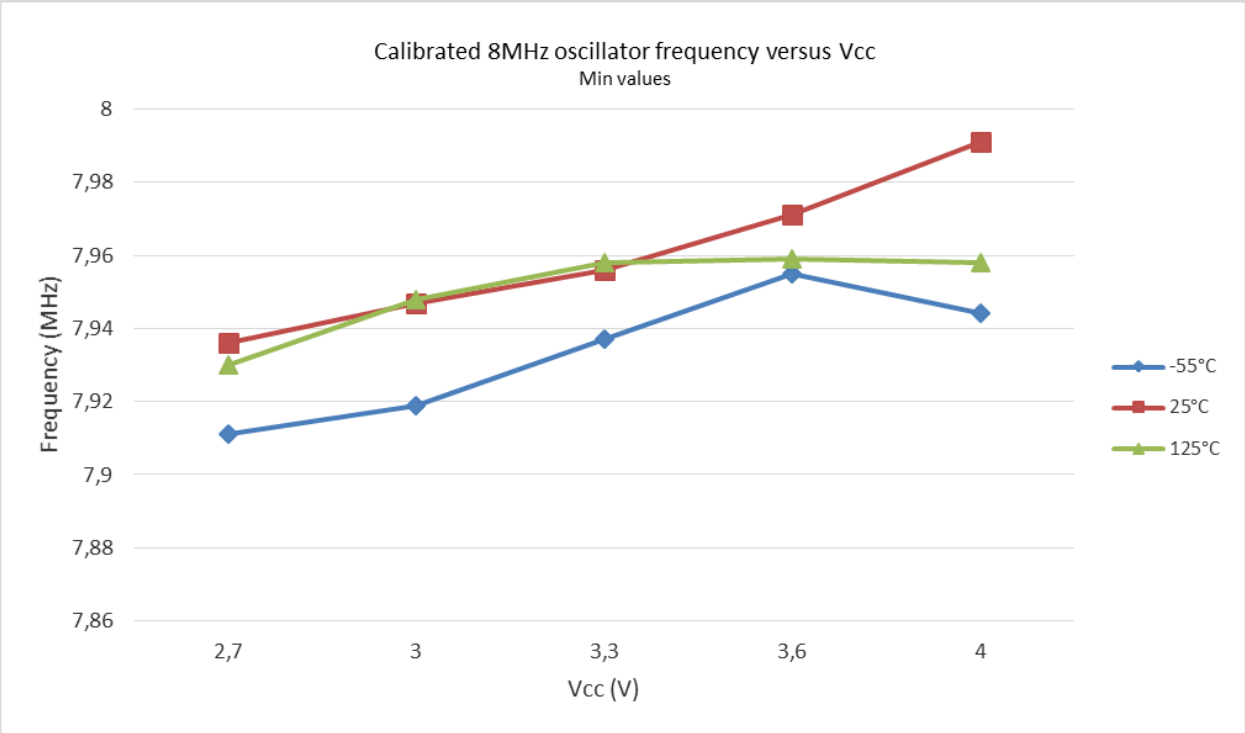
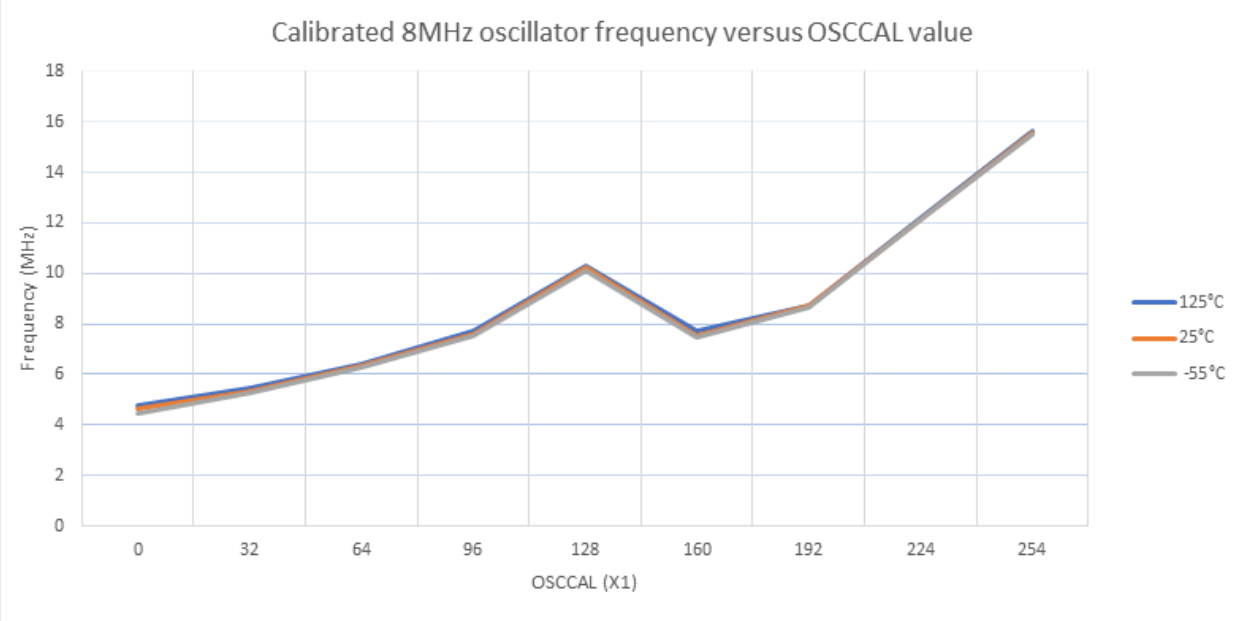


Figure 31-20. Calibrated 8MHz Oscillator Frequency versus OSCCAL Value



32. Register Summary

Offset	Name	Bit Pos.								
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						PINE2	PINE1	PINE0
0x2D	DDRE	7:0						DDRE2	DDRE1	DDRE0
0x2E	PORTE	7:0						PORTE2	PORTE1	PORTE0
0x2F ... 0x34	Reserved									
0x35	TIFR0	7:0						OCF0B	OCF0A	TOV0
0x36	TIFR1	7:0			ICF1			OCF1B	OCF1A	TOV1
0x37 ... 0x38	Reserved									
0x39	GPIOR1	7:0	GPIOR1[7:0]							
0x3A	GPIOR2	7:0	GPIOR2[7:0]							
0x3B	PCIFR	7:0					PCIF3	PCIF2	PCIF1	PCIF0
0x3C	EIFR	7:0					INTF3	INTF2	INTF1	INTF0
0x3D	EIMSK	7:0					INT3	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]							
		15:8						EEAR[10:8]		
0x43	GTCCR	7:0	TSM	ICPSEL1						PSRSYNC
0x44	TCCR0A	7:0	COM0A[1:0]		COM0B[1:0]				WGM0[1:0]	
0x45	TCCR0B	7:0	FOC0A	FOC0B			WGM02	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	PLLCSR	7:0						PLLF	PLLE	PLOCK
0x4A ... 0x4B	Reserved									
0x4C	SPCR	7:0	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR[1:0]	
0x4D	SPSR	7:0	SPIF	WCOL						SPI2X
0x4E	SPDR	7:0	SPID[7:0]							

Offset	Name	Bit Pos.								
0x4F ... 0x50	Reserved									
0x51	DWDR	7:0	DWDR[7:0]							
0x52	Reserved									
0x53	SMCR	7:0					SM2	SM1	SM0	SE
0x54	MCUSR	7:0					WDRF	BORF	EXTRF	PORF
0x55	MCUCR	7:0	SPIPS			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	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
		15:8	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
0x5F	SREG	7:0	I	T	H	S	V	N	Z	C
0x60	WDTCSR	7:0	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
0x61	CLKPR	7:0	CLKPCE				CLKPS[3:0]			
0x62 ... 0x63	Reserved									
0x64	PRR	7:0		PRCAN	PRPSC	PRTIM1	PRTIM0	PRSPI	PRLIN	PRADC
0x65	Reserved									
0x66	OSCCAL	7:0	CAL[7:0]							
0x67	Reserved									
0x68	PCICR	7:0					PCIE3	PCIE2	PCIE1	PCIE0
0x69	EICRA	7:0	ISC3[1:0]		ISC2[1:0]		ISC1[1:0]		ISC0[1:0]	
0x6A	PCMSK0	7:0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
0x6B	PCMSK1	7:0	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6C	PCMSK2	7:0	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x6D	PCMSK3	7:0						PCINT26	PCINT25	PCINT24
0x6E	TIMSK0	7:0						OCIE0B	OCIE0A	TOIE0
0x6F	TIMSK1	7:0			ICIE1			OCIE1B	OCIE1A	TOIE1
0x70 ... 0x74	Reserved									
0x75	AMP0CSR	7:0	AMP0EN	AMP0IS	AMP0G[1:0]		AMPCMP0	AMP0TS[2:0]		
0x76	AMP1CSR	7:0	AMP1EN	AMP1IS	AMP1G[1:0]		AMPCMP1	AMP1TS[2:0]		
0x77	AMP2CSR	7:0	AMP2EN	AMP2IS	AMP2G[1:0]		AMPCMP2	AMP2TS[2:0]		
0x78	ADCL and ADCH (ADLAR = 0)	7:0	ADC[7:0]							
		15:8							ADC[9:8]	
0x78	ADCL and ADCH (ADLAR = 1)	7:0	ADC[1:0]							
		15:8	ADC[9:2]							
0x7A	ADCSRA	7:0	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS[2:0]		
0x7B	ADCSRB	7:0	ADHSM	ISRCEN	AREFEN		ADTS[3:0]			
0x7C	ADMUX	7:0	REFS[1:0]		ADLAR	MUX[4:0]				

Offset	Name	Bit Pos.								
0x7D	Reserved									
0x7E	DIDR0	7:0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
0x7F	DIDR1	7:0						ADC10D	ADC9D	ADC8D
0x80	TCCR1A	7:0	COM1A[1:0]		COM1B[1:0]				WGM1[1:0]	
0x81	TCCR1B	7:0	ICNC1	ICES1	RTGEN	WGM13	WGM12	CS1[2:0]		
0x82	TCCR1C	7:0	FOC1A	FOC1B						
0x83	Reserved									
0x84	TCNT1L and TCNT1H	7:0	TCNT1[7:0]							
		15:8	TCNT1[15:8]							
0x86	ICR1L and ICR1H	7:0	ICR1[7:0]							
		15:8	ICR1[15:8]							
0x88	OCR1AL and OCR1AH	7:0	OCR1A[7:0]							
		15:8	OCR1A[15:8]							
0x8A	OCR1BL and OCR1BH	7:0	OCR1B[7:0]							
		15:8	OCR1B[15:8]							
0x8C ... 0xD7	Reserved									
0xD8	CANGCON	7:0	ABRQ	OVRQ	TTC	SYNTTC	LISTEN	TEST	$\overline{\text{ENA/STB}}$	SWRES
0xD9	CANGSTA	7:0		OVRG		TXBSY	RXBSY	ENFG	BOFF	ERRP
0xDA	CANGIT	7:0	CANIT	BOFFIT	OVRTIM	BXOK	SERG	CERG	FERG	AERG
0xDB	CANGIE	7:0	ENIT	ENBOFF	ENRX	ENTX	ENERR	ENBX	ENERG	ENOVRT
0xDC	CANEN	7:0			ENMOB5	ENMOB4	ENMOB3	ENMOB2	ENMOB1	ENMOB0
		15:8								
0xDE	CANIE	7:0			IEMOB5	IEMOB4	IEMOB3	IEMOB2	IEMOB1	IEMOB0
		15:8								
0xE0	CANSIT	7:0			SIT5	SIT4	SIT3	SIT2	SIT1	SIT0
		15:8								
0xE2	CANBT1	7:0		BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	
0xE3	CANBT2	7:0		SJW1	SJW0		PRS2	PRS1	PRS0	
0xE4	CANBT3	7:0		PHS22	PHS21	PHS20	PHS12	PHS11	PHS10	SMP
0xE5	CANTCON	7:0	TPRSC7	TPRSC6	TPRSC5	TPRSC4	TPRSC3	TPRSC2	TPRSC1	TPRSC0
0xE6	CANTIM	7:0	CANTIM7	CANTIM6	CANTIM5	CANTIM4	CANTIM3	CANTIM2	CANTIM1	CANTIM0
		15:8	CANTIM15	CANTIM14	CANTIM13	CANTIM12	CANTIM11	CANTIM10	CANTIM9	CANTIM8
0xE8	CANTTC	7:0	TIMTTC7	TIMTTC6	TIMTTC5	TIMTTC4	TIMTTC3	TIMTTC2	TIMTTC1	TIMTTC0
		15:8	TIMTTC15	TIMTTC14	TIMTTC13	TIMTTC12	TIMTTC11	TIMTTC10	TIMTTC9	TIMTTC8
0xEA	CANTEC	7:0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
0xEB	CANREC	7:0	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
0xEC	CANHPMOB	7:0	HPMOB3	HPMOB2	HPMOB1	HPMOB0	CGP3	CGP2	CGP1	CGP0
0xED	CANPAGE	7:0	MOBNB3	MOBNB2	MOBNB1	MOBNB0	$\overline{\text{AINC}}$	INDX2	INDX1	INDX0
0xEE	CANSTMOB	7:0	DLCW	TXOK	RXOK	BERR	SERR	CERR	FERR	AERR
0xEF	CANCDMOB	7:0	CONMOB1	CONMOB0	RPLV	IDE	DLC3	DLC2	DLC1	DLC0
0xF0	CANIDT	7:0						RTRTAG		RB0TAG
		15:8								
		23:16	IDT2	IDT1	IDT0					
		31:24	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5	IDT4	IDT3

Offset	Name	Bit Pos.								
0xF0	CANIDT	7:0	IDT4	IDT3	IDT2	IDT1	IDT0	RTRTAG	RB1TAG	RB0TAG
		15:8	IDT12	IDT11	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5
		23:16	IDT20	IDT19	IDT18	IDT17	IDT16	IDT15	IDT14	IDT13
		31:24	IDT28	IDT27	IDT26	IDT25	IDT24	IDT23	IDT22	IDT21
0xF4	CANIDM	7:0						RTRMSK		IDEMSK
		15:8								
		23:16	IDMSK2	IDMSK1	IDMSK0					
		31:24	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5	IDMSK4	IDMSK3
0xF4	CANIDM	7:0	IDMSK4	IDMSK3	IDMSK2	IDMSK1	IDMSK0	RTRMSK		IDEMSK
		15:8	IDMSK12	IDMSK11	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5
		23:16	IDMSK20	IDMSK19	IDMSK18	IDMSK17	IDMSK16	IDMSK15	IDMSK14	IDMSK13
		31:24	IDMSK28	IDMSK27	IDMSK26	IDMSK25	IDMSK24	IDMSK23	IDMSK22	IDMSK21
0xF8	CANSTM	7:0	TIMSTM7	TIMSTM6	TIMSTM5	TIMSTM4	TIMSTM3	TIMSTM2	TIMSTM1	TIMSTM0
		15:8	TIMSTM15	TIMSTM14	TIMSTM13	TIMSTM12	TIMSTM11	TIMSTM10	TIMSTM9	TIMSTM8
0xFA	CANMSG	7:0	MSG7	MSG6	MSG5	MSG4	MSG3	MSG2	MSG1	MSG0

33. Instruction Set Summary

Table 33-1. Arithmetic and Logic Instructions

Mnemonic	Operands	Description		Op		Flags	#Clocks
ADD	Rd, Rr	Add without Carry	Rd	←	Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd	←	Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	Rd + 1:Rd	←	Rd + 1:Rd + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract without Carry	Rd	←	Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd	←	Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd	←	Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd	←	Rd - K - C	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd + 1:Rd	←	Rd + 1:Rd - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	Rd	←	Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd	←	Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd	←	Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd	←	Rd v K	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd	←	Rd ⊕ Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd	←	$FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd	←	$00 - Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd	←	Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd	←	Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd	←	Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd	←	Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd	←	Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd	←	Rd ⊕ Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd	←	$FF	None	1
MUL	Rd,Rr	Multiply Unsigned	R1:R0	←	Rd x Rr (UU)	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0	←	Rd x Rr (SS)	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0	←	Rd x Rr (SU)	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0	←	Rd x Rr<<1 (UU)	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0	←	Rd x Rr<<1 (SS)	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0	←	Rd x Rr<<1 (SU)	Z,C	2

Table 33-2. Branch Instructions

Mnemonic	Operands	Description		Op		Flags	#Clocks
RJMP	k	Relative Jump	PC	←	PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) PC(21:16)	← ←	Z 0	None	2
JMP	k	Jump	PC	←	k	None	3
RCALL	k	Relative Call Subroutine	PC	←	PC + k + 1	None	3 / 4(1)
ICALL		Indirect Call to (Z)	PC(15:0)	←	Z	None	3 / 4(1)

Mnemonic	Operands	Description		Op		Flags	#Clocks
			PC(21:16)	←	0		
CALL	k	Call Subroutine	PC	←	k	None	4 / 5(1)
RET		Subroutine Return	PC	←	STACK	None	4 / 5(1)
RETI		Interrupt Return	PC	←	STACK	I	4 / 5(1)
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,C,N,V,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C			Z,C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K			Z,C,N,V,S,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 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) = 0) PC	←	PC + 2 or 3	None	1 / 2 / 3
SBIS	A, b	Skip if Bit in I/O Register 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, 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

Mnemonic	Operands	Description		Op		Flags	#Clocks
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

Table 33-3. Data Transfer Instructions

Mnemonic	Operands	Description		Op		Flags	#Clocks
MOV	Rd, Rr	Copy Register	Rd	←	Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd	←	Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd	←	K	None	1
LDS	Rd, k	Load Direct from data space	Rd	←	(k)	None	2(1)
LD	Rd, X	Load Indirect	Rd	←	(X)	None	2(1)
LD	Rd, X+	Load Indirect and Post-Increment	Rd X	← ←	(X) X + 1	None	2(1)
LD	Rd, -X	Load Indirect and Pre-Decrement	X Rd	← ←	X - 1 (X)	None	2(1)
LD	Rd, Y	Load Indirect	Rd	←	(Y)	None	2(1)
LD	Rd, Y+	Load Indirect and Post-Increment	Rd Y	← ←	(Y) Y + 1	None	2(1)
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y Rd	← ←	Y - 1 (Y)	None	2(1)
LDD	Rd, Y+q	Load Indirect with Displacement	Rd	←	(Y + q)	None	2(1)
LD	Rd, Z	Load Indirect	Rd	←	(Z)	None	2(1)
LD	Rd, Z+	Load Indirect and Post-Increment	Rd Z	← ←	(Z) Z+1	None	2(1)
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z Rd	← ←	Z - 1 (Z)	None	2(1)
LDD	Rd, Z+q	Load Indirect with Displacement	Rd	←	(Z + q)	None	2(1)
STS	k, Rr	Store Direct to Data Space	(k)	←	Rd	None	2(1)
ST	X, Rr	Store Indirect	(X)	←	Rr	None	1(1)
ST	X+, Rr	Store Indirect and Post-Increment	(X) X	← ←	Rr X + 1	None	1(1)
ST	-X, Rr	Store Indirect and Pre-Decrement	X (X)	← ←	X - 1 Rr	None	2(1)
ST	Y, Rr	Store Indirect	(Y)	←	Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) Y	← ←	Rr Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y (Y)	← ←	Y - 1 Rr	None	2(1)
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q)	←	Rr	None	2(1)

Mnemonic	Operands	Description		Op		Flags	#Clocks
ST	Z, Rr	Store Indirect	(Z)	←	Rr	None	2(1)
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) Z	← ←	Rr Z + 1	None	2(1)
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z	←	Z - 1	None	2(1)
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q)	←	Rr	None	2(1)
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-Increment	Rd Z	← ←	(Z) Z + 1	None	3
ELPM		Extended Load Program Memory	R0	←	(RAMPZ:Z)	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd	←	(RAMPZ:Z)	None	3
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd (RAMPZ:Z)	← ←	(RAMPZ:Z) (RAMPZ:Z) + 1	None	3
SPM		Store Program Memory	(RAMPZ:Z)	←	R1:R0	None	
SPM	Z+	Store Program Memory and Post-Increment by 2	(RAMPZ:Z) Z	← ←	R1:R0 Z + 2	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

Table 33-4. Bit and Bit-Test Instructions

Mnemonic	Operands	Description		Op		Flags	#Clocks
LSL	Rd	Logical Shift Left	Rd(n+1) Rd(0) C	← ← ←	Rd(n) 0 Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n) Rd(7) C	← ← ←	Rd(n+1) 0 Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) Rd(n+1) C	← ← ←	C Rd(n) Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7) Rd(n) C	← ← ←	C Rd(n+1) 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)	None	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b)	←	1	None	2
CBI	A, b	Clear Bit in I/O Register	I/O(A, b)	←	0	None	2
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
BSET	s	Flag Set	SREG(s)	←	1	SREG(s)	1

Mnemonic	Operands	Description		Op		Flags	#Clocks
BCLR	s	Flag Clear	SREG(s)	←	0	SREG(s)	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

Table 33-5. MCU Control Instructions

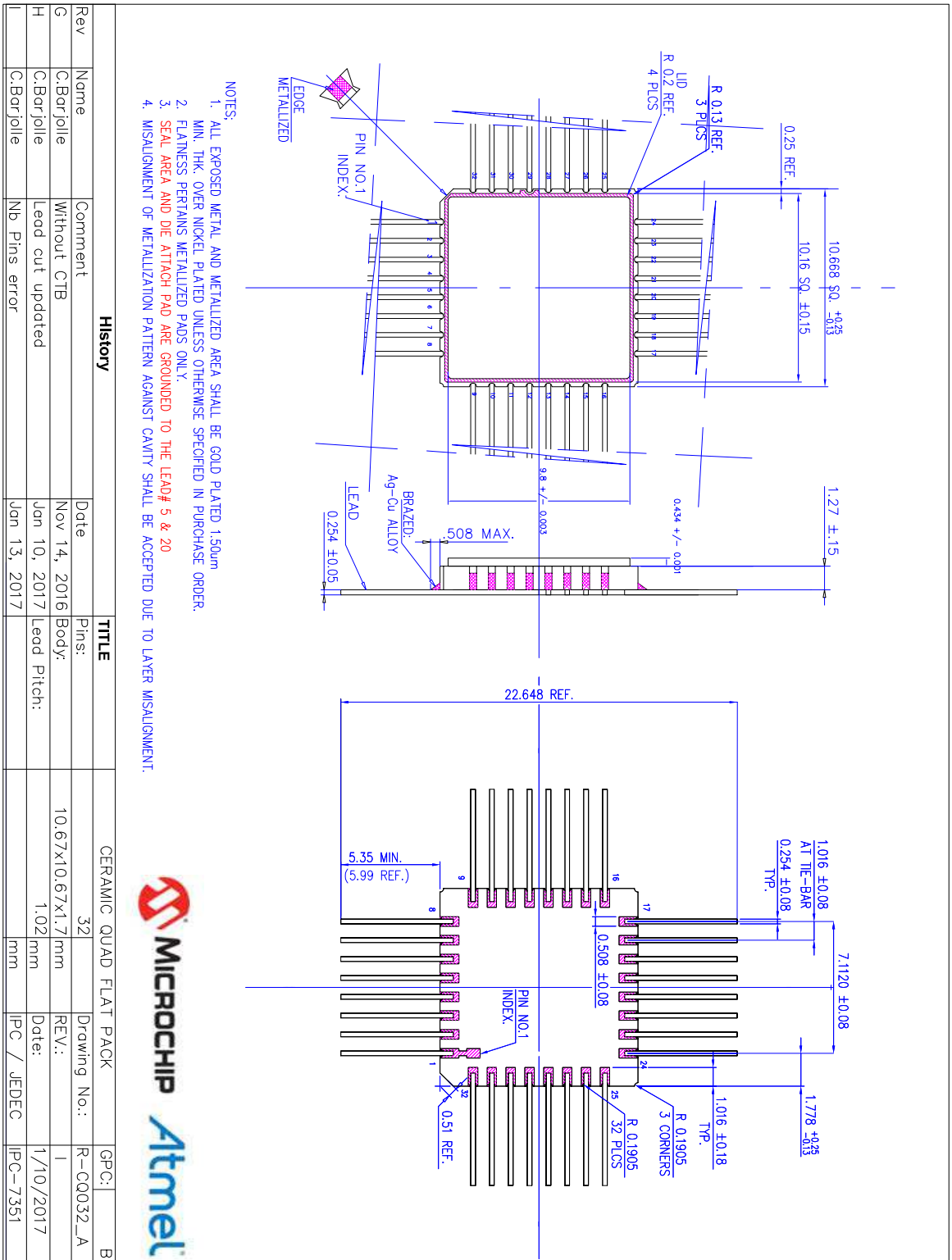
Mnemonic	Operands	Description	Operation	Flags	#Clocks
BREAK		Break	(See also in Debug interface description)	None	1
NOP		No Operation		None	1
SLEEP		Sleep	(see also power management and sleep description)	None	1
WDR		Watchdog Reset	(see also Watchdog Controller description)	None	1

Note:

1. Cycle time for data memory accesses assume internal RAM access and are not valid for accesses through the NVM controller. A minimum of one extra cycle must be added when accessing memory through the NVM controller (such as Flash and EEPROM), but depending on simultaneous accesses by other masters or the NVM controller state, there may be more than one extra cycle.

34. Packaging Information

34.1 CQFP32

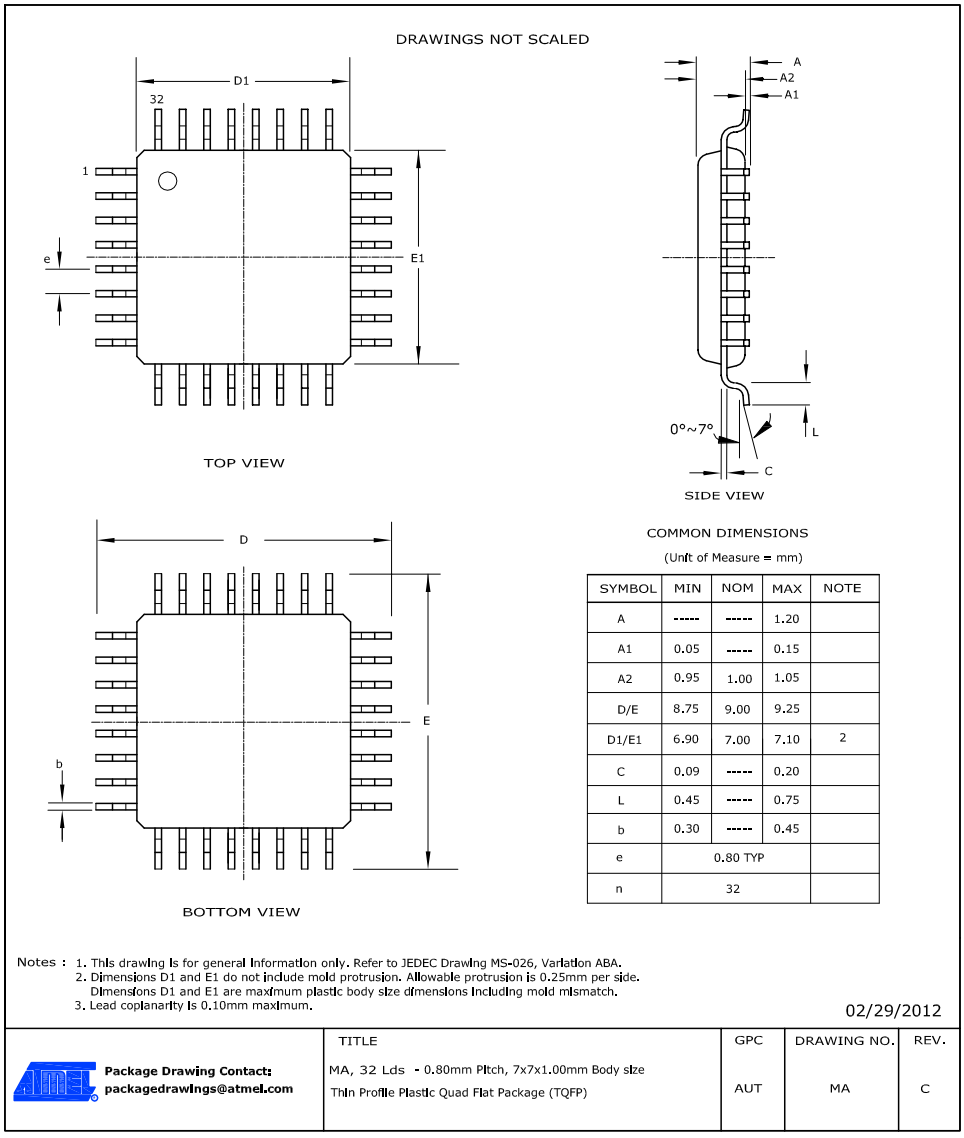


History			
Rev	Name	Comment	Date
G	C.Barjolle	Without CTB	Nov 14, 2016
H	C.Barjolle	Lead cut updated	Jan 10, 2017
I	C.Barjolle	Nb Pins error	Jan 13, 2017

TITLE			
Pins:	32	Drawing No.:	R-CQ032_A
Body:	10.67x10.67x1.7 mm	REV.:	I
Lead Pitch:	1.02 mm	Date:	1/10/2017
	mm	IPC / JEDEC	IPC-7351

CERAMIC QUAD FLAT PACK | GPC: BNH

34.2 TQFP32



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/E	8.75	9.00	9.25	
D1/E1	6.90	7.00	7.10	2
C	0.09	-----	0.20	
L	0.45	-----	0.75	
b	0.30	-----	0.45	
e	0.80 TYP			
n	32			

Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MS-026, Variation ABA.
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.

02/29/2012

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	MA, 32 Lds - 0.80mm Pitch, 7x7x1.00mm Body size Thin Profile Plastic Quad Flat Package (TQFP)	AUT	MA	C

35. Revision History

35.1 Rev. A - 06/2017

Initial release based on Atmel-8209F-ATmega16M1/32M1/64M1_Datasheet_Complete_10/2016

35.2 Rev. B - 03/2018

Section	Changes
All	Editorial changes.
Electrical Characteristics	Updated DC Characteristics. Updated Clock Characteristics. Added Typical Characteristics.

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

Trademarks

ISBN: 978-1-5224-2782-7

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	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
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	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**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 **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