8-bit AVR Microcontroller

# ATtiny102 / ATtiny104

DATASHEET COMPLETE

## Introduction

The Atmel® ATtiny102/ATtiny104 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 ATtiny102/ATtiny104 achieves throughputs close to 1 MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

## Feature

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

- Advanced RISC Architecture
  - 54 Powerful Instructions
  - Mostly Single Clock Cycle Execution
  - 16 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 12 MIPS Throughput at 12MHz
- Non-volatile Program and Data Memories
  - 1024 Bytes of In-system Programmable Flash Program Memory
  - 32 Bytes Internal SRAM
  - Flash Write/Erase Cycles: 10,000
  - Data Retention: 20 Years at 85°C / 100 Years at 25°C
  - Self-programming Flash on Full Operating Voltage Range (1.8 – 5.5V)
- Peripheral Features
  - One 16-bit Timer/Counter (TC) with Prescaler, Input Capture, Two Output Capture and Two PWM Channels
  - Programmable Watchdog Timer (WDT) with Separate On-chip Oscillator
  - Selectable Internal Voltage References: 1.1V, 2.2V and 4.3V
  - 10-bit ADC with 8-channels/14-pin and 5-channel/8-pin Package Options
  - On-chip Analog Comparator (AC)
  - Serial Communication Module: USART

- Special Microcontroller Features
  - In-system Programmable
    - External Programming (2.7 – 5.5V)
    - Self Programming (1.8 – 5.5V)
  - External and Internal Interrupt Sources
  - Low Power Idle, ADC Noise Reduction, and Power-pown Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Supply Voltage Level Monitor with Interrupt and Reset
  - Accurate Internal Calibrated Oscillator
  - Fast and Normal Start-up Time Options Available
  - Individual Serial Number to Represent a Unique ID.
- I/O and Packages
  - 12 Programmable I/O Lines for ATtiny104 and 6 Programmable I/O Lines for ATtiny102
  - 8-pin UDFN (ATtiny102)
  - 8-pin SOIC150 (ATtiny102)
  - 14-pin SOIC150 (ATtiny104)
- Operating Voltage
  - 1.8 - 5.5V
- Temperature Range
  - -40 to +125°C
- Speed Grades
  - 0 – 4MHz at 1.8 – 5.5V
  - 0 – 8MHz at 2.7 – 5.5V
  - 0 – 12MHz at 4.5 – 5.5V

# Table of Contents


Introduction....1

Feature....1

1. Description....8

2. Configuration Summary....9

3. Ordering Information....10

4. Block Diagram....11

5. Pin Configurations....12

   5.1. Pin Descriptions....12

6. I/O Multiplexing....14

7. General Information....15

   7.1. Resources....15
   7.2. Data Retention....15
   7.3. About Code Examples....15

8. AVR CPU Core....16

   8.1. Overview....16
   8.2. Features....16
   8.3. Block Diagram....16
   8.4. ALU – Arithmetic Logic Unit....18
   8.5. Status Register....18
   8.6. General Purpose Register File....18
   8.7. The X-register, Y-register, and Z-register....19
   8.8. Stack Pointer....19
   8.9. Accessing 16-bit Registers....20
   8.10. Instruction Execution Timing....21
   8.11. Reset and Interrupt Handling....21
   8.12. Register Description....22

9. AVR Memories....27

   9.1. Overview....27
   9.2. Features....27
   9.3. In-System Reprogrammable Flash Program Memory....27
   9.4. SRAM Data Memory....27
   9.5. I/O Memory....29

10. Clock System....30

   10.1. Overview....30
   10.2. Clock Distribution....30

10.3. Clock Subsystems....30
10.4. Clock Sources....31
10.5. System Clock Prescaler....32
10.6. Starting....33
10.7. Register Description....34

11. Power Management and Sleep Modes....39
11.1. Overview....39
11.2. Features....39
11.3. Sleep Modes....39
11.4. Power Reduction Register....40
11.5. Minimizing Power Consumption....41
11.6. Register Description....42

12. SCRST - System Control and Reset....45
12.1. Overview....45
12.2. Features....45
12.3. Resetting the AVR....45
12.4. Reset Sources....46
12.5. Watchdog Timer....48
12.6. Register Description....50

13. Interrupts....55
13.1. Overview....55
13.2. Interrupt Vectors....55
13.3. External Interrupts....56
13.4. Register Description....57

14. I/O-Ports....65
14.1. Overview....65
14.2. Features....65
14.3. I/O Pin Equivalent Schematic....65
14.4. Ports as General Digital I/O....66
14.5. Register Description....79

15. USART - Universal Synchronous Asynchronous Receiver Transceiver....89
15.1. Overview....89
15.2. Features....89
15.3. Block Diagram....89
15.4. Clock Generation....90
15.5. Frame Formats....93
15.6. USART Initialization....94
15.7. Data Transmission – The USART Transmitter....95
15.8. Data Reception – The USART Receiver....97
15.9. Asynchronous Data Reception....100
15.10. Multi-Processor Communication Mode....104
15.11. Examples of Baud Rate Setting....105
15.12. Register Description....108

16. USARTSPI - USART in SPI Mode..... 119
16.1. Overview..... 119
16.2. Features..... 119
16.3. Clock Generation..... 119
16.4. SPI Data Modes and Timing..... 120
16.5. Frame Formats..... 120
16.6. Data Transfer..... 122
16.7. AVR USART MSPIM vs. AVR SPI..... 123
16.8. Register Description..... 123

17. TC0 - 16-bit Timer/Counter0 with PWM..... 124
17.1. Overview..... 124
17.2. Features..... 124
17.3. Block Diagram..... 124
17.4. Definitions..... 125
17.5. Registers..... 126
17.6. Accessing 16-bit Timer/Counter Registers..... 127
17.7. Timer/Counter Clock Sources..... 129
17.8. Counter Unit..... 131
17.9. Input Capture Unit..... 132
17.10. Output Compare Units..... 134
17.11. Compare Match Output Unit..... 136
17.12. Modes of Operation..... 137
17.13. Timer/Counter Timing Diagrams..... 145
17.14. Register Description..... 146

18. AC - Analog Comparator..... 160
18.1. Overview..... 160
18.2. Features..... 160
18.3. Block Diagram..... 160
18.4. Register Description..... 161

19. ADC - Analog to Digital Converter..... 166
19.1. Overview..... 166
19.2. Features..... 166
19.3. Block Diagram..... 166
19.4. Operation..... 167
19.5. Starting a Conversion..... 168
19.6. Prescaling and Conversion Timing..... 169
19.7. Changing Channel or Reference Selection..... 172
19.8. ADC Input Channels..... 172
19.9. ADC Voltage Reference..... 172
19.10. ADC Noise Canceler..... 173
19.11. Analog Input Circuitry..... 173
19.12. Analog Noise Canceling Techniques..... 174
19.13. ADC Accuracy Definitions..... 174
19.14. ADC Conversion Result..... 176
19.15. Register Description..... 176

20. MEMPROG- Memory Programming.....185
20.1. Overview.....185
20.2. Features.....185
20.3. Non-Volatile Memories (NVM).....186
20.4. Accessing the NVM.....191
20.5. Self programming.....194
20.6. External Programming.....194
20.7. Register Description.....195

21. TPI-Tiny Programming Interface.....198
21.1. Overview.....198
21.2. Features.....198
21.3. Block Diagram.....198
21.4. Physical Layer of Tiny Programming Interface.....199
21.5. Instruction Set.....203
21.6. Accessing the Non-Volatile Memory Controller.....205
21.7. Control and Status Space Register Descriptions.....206

22. Electrical Characteristics.....210
22.1. Absolute Maximum Ratings*.....210
22.2. DC Characteristics.....210
22.3. Speed.....212
22.4. Clock Characteristics.....213
22.5. System and Reset Characteristics.....214
22.6. Analog Comparator Characteristics.....215
22.7. ADC Characteristics.....216
22.8. Serial Programming Characteristics.....217

23. Typical Characteristics.....218
23.1. Active Supply Current.....218
23.2. Idle Supply Current.....221
23.3. Supply Current of I/O Modules.....222
23.4. Power-down Supply Current.....223
23.5. Pin Driver Strength.....224
23.6. Pin Threshold and Hysteresis.....228
23.7. Analog Comparator Offset.....232
23.8. Pin Pull-up.....233
23.9. Internal Oscillator Speed.....236
23.10. VLM Thresholds.....238
23.11. Current Consumption of Peripheral Units.....240
23.12. Current Consumption in Reset and Reset Pulsewidth.....243

24. Register Summary.....244
24.1. Note.....245

25. Instruction Set Summary.....246

26. Packaging Information.....250
26.1. 8-pin UDFN.....250

26.2. 8-pin SOIC150....251
26.3. 14-pin SOIC150....252

27. Errata....253

27.1. ATtiny102....253
27.2. ATtiny104....253

28. Datasheet Revision History....254

28.1. Rev D - 10/2016....254
28.2. Rev C - 07/2016....254
28.3. Rev B - 06/2016....254
28.4. Rev A - 02/2016....254

# 1. Description

The Atmel®AVR® core combines a rich instruction set with 16 general purpose working registers. All the 16 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 device provides the following features: 1024 Bytes of In-System Programmable Flash with Read-While-Write capabilities, 32 Bytes SRAM, 6/12 general purpose I/O lines for ATtiny102/ATtiny104, 16 general purpose working registers, a 16-bit Timer/Counters (TC) with compare modes, internal and external interrupts, one serial programmable USART, a programmable Watchdog Timer with internal Oscillator and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, TC, USART, ADC, Analog Comparator (AC), and interrupt system to continue functioning. ADC Noise Reduction mode minimizes switching noise during ADC conversions by stopping the CPU and all I/O modules except the ADC. 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 device is manufactured using Atmel's high density Non-Volatile Memory (NVM) technology. The on-chip, in-system programmable Flash allows program memory to be re-programmed in-system by a conventional, NVM programmer.

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

## 2. Configuration Summary

| | ATtiny102 | ATtiny104 |
|---|---|---|
| Pin Count | 8 | 14 |
| Flash (Bytes) | 1024 | 1024 |
| SRAM (Bytes) | 32 | 32 |
| EEPROM (Bytes) | - | - |
| General Purpose I/O-pins (GPIOs) | 6 | 12 |
| USART | 1 | 1 |
| Analog-to-Digital Converter (ADC) / Channels | 10-bit ADC with 5-channel | 10-bit ADC with 8-channels |
| Analog Comparators (AC) Channels | 1 | 1 |
| AC Propagation Delay | 75-750ns | 75-750ns |
| 16-bit Timer Counter (TC) Instances | 1 | 1 |
| PWM Channels | 2 | 2 |
| RC Oscillator | +/-2 % | +/-2 % |
| Internal Voltage Reference | 1.1V/2.2V/4.3V | 1.1V/2.2V/4.3V |
| Operating Voltage | 1.8 - 5.5V | |
| Max Operating Frequency (MHz) | 12 | |
| Temperature Range | -40°C to +125°C | |
| Packages | 8-pin UDFN<br>8-pin SOIC150 | 14-pin SOIC150 |

# 3. Ordering Information

| Speed [MHz] | Power Supply [V] | Ordering Code | Package | Operational Range |
|---|---|---|---|---|
| 12 | 1.8 -5.5 | ATtiny102-M7R | 8 pad UDFN | Industrial (-40°C to +105°C) |
| | | ATtiny102F-M7R(1) | 8 pad UDFN | |
| | | ATtiny102-SSNR | 8 pin SOIC150 | |
| | | ATtiny102F-SSNR(1) | 8 pin SOIC150 | |
| | | ATtiny104-SSNR | 14 pin SOIC150 | |
| | | ATtiny104F-SSNR(1) | 14 pin SOIC150 | |
| | | ATtiny102-M8R | 8 pad UDFN | Industrial (-40°C to +125°C) |
| | | ATtiny102F-M8R(1) | 8 pad UDFN | |
| | | ATtiny102-SSFR | 8 pin SOIC150 | |
| | | ATtiny102F-SSFR(1) | 8 pin SOIC150 | |
| | | ATtiny104-SSFR | 14 pin SOIC150 | |
| | | ATtiny104F-SSFR(1) | 14 pin SOIC150 | |

**Note:**

1. ATtiny104F-xxx and ATtiny102F-xxx have the fast start-up time option.

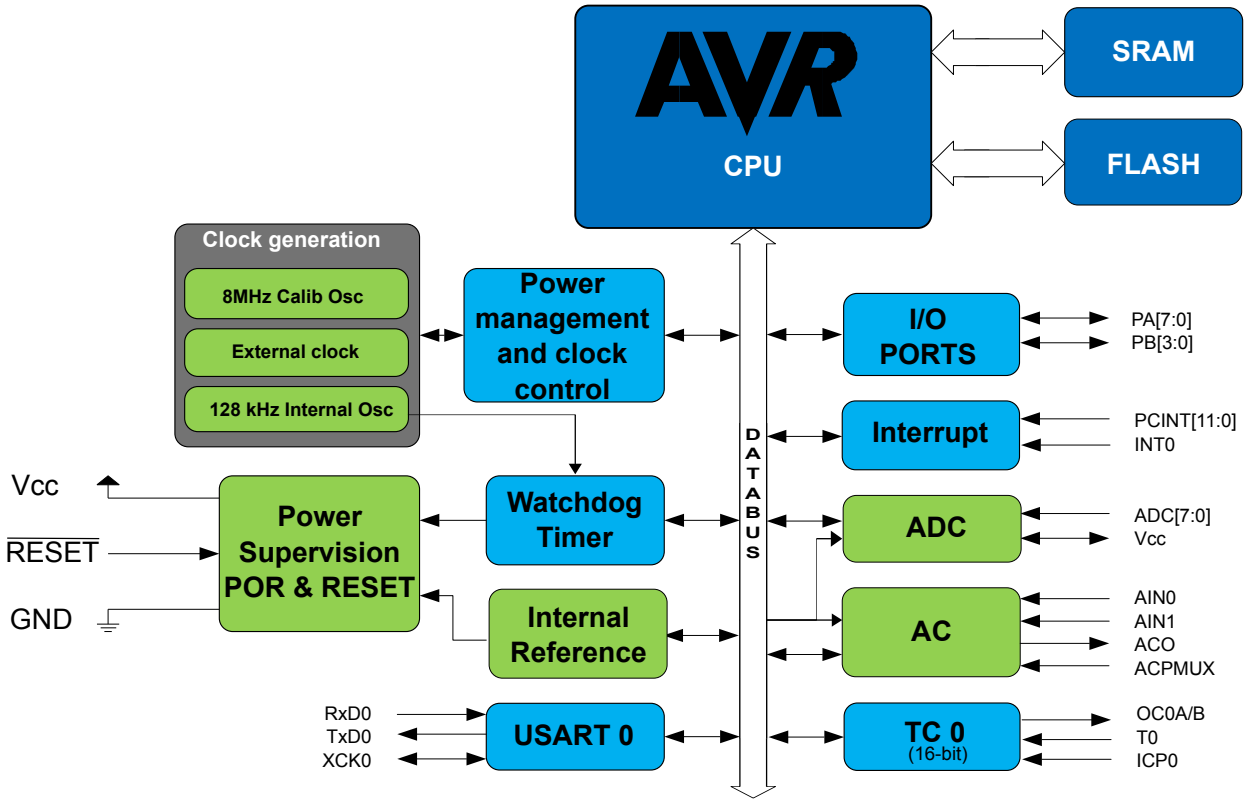
| Package Type | |
|---|---|
| 8 pad UDFN | 8-pad, 2 x 3 x 0.6mm Body, Thermally Enhanced Plastic Ultra Thin Dual Flat No-Lead Package (UDFN) |
| 8 pin SOIC150 | 8-lead, 0.150” Wide Body, Plastic Gull Wing Small Outline (JEDEC SOIC) |
| 14 pin SOIC150 | 14-lead, 1.27mm Pitch, 8.65 x 3.90 x 1.60mm Body Size, Plastic Small Outline Package (SOIC) |

**Related Links**

Starting from Reset on page 33

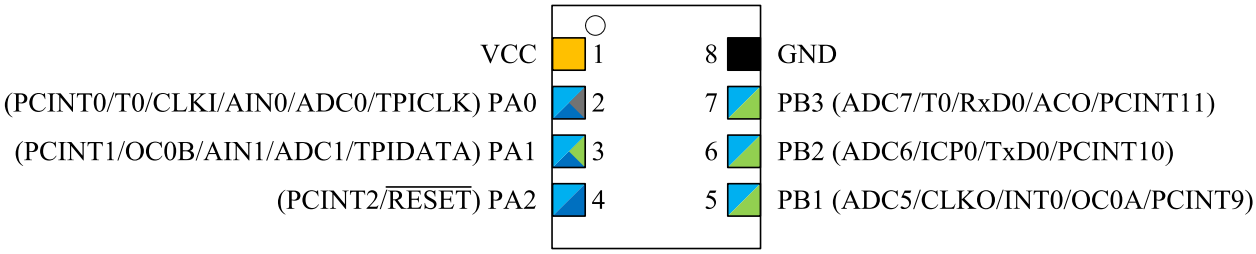
# 4. Block Diagram

**Figure 4-1. Block Diagram**



# 5. Pin Configurations

**Figure 5-1. Pin-Out of 8-Pin UDFN**




**Caution:** The thermal pad on the rear of the package should not be connected.

**Figure 5-2. Pin-Out of 8-Pin SOIC150**

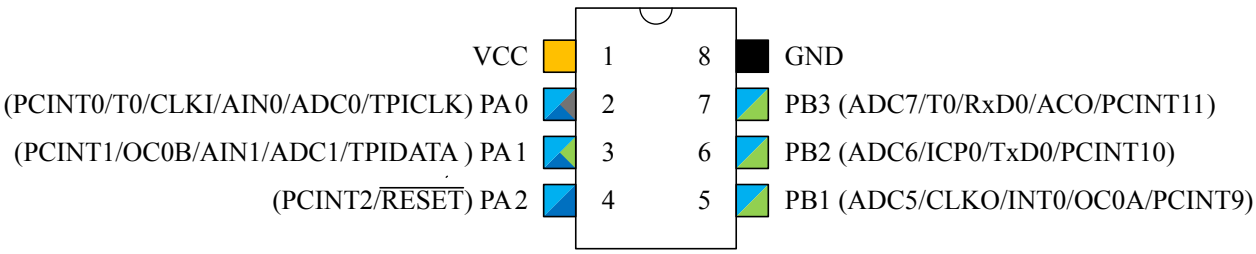
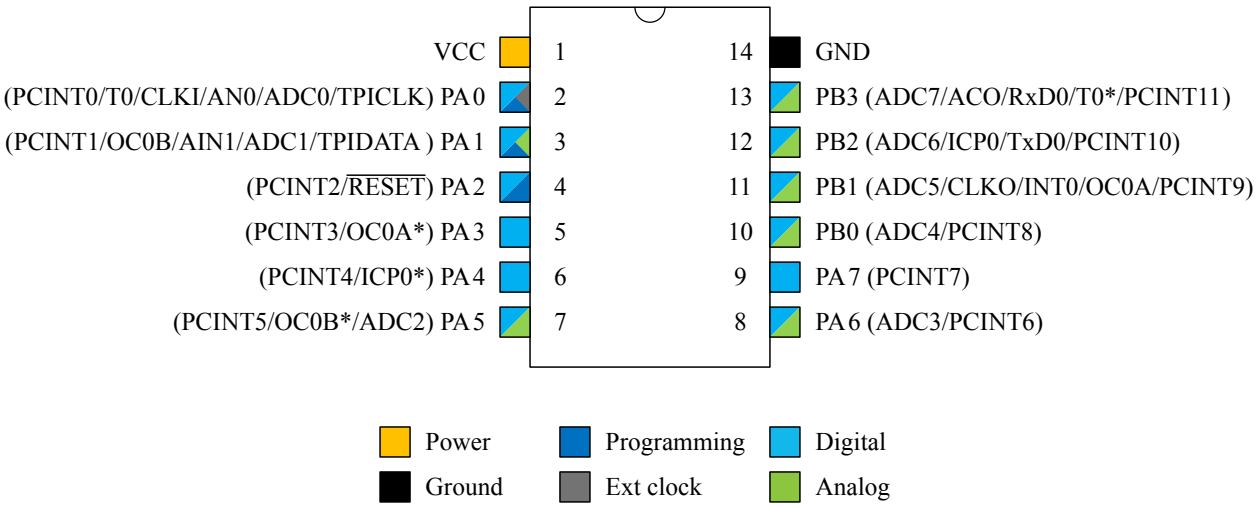



**Figure 5-3. Pin-Out of 14-Pin SOIC150**




## 5.1. Pin Descriptions

### 5.1.1. VCC

Digital supply voltage.

### 5.1.2. GND

Ground.

### 5.1.3. Port A (PA[7:0])

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

### 5.1.4. Port B (PB[3:0])

This is a 4-bit, bi-directional I/O port with internal pull-up resistors, individually selectable for each bit. The output buffers have symmetrical drive characteristics, with both high sink and source capability. As inputs, the port pins that are externally pulled low will source current if pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

### 5.1.5. $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running and provided the reset pin has not been disabled. The minimum pulse length is given in *System and Reset Characteristics of Electrical Characteristics*. Shorter pulses are not guaranteed to generate a reset.

The reset pin can also be used as a (weak) I/O pin.

**Related Links**

System and Reset Characteristics on page 214

# 6. I/O Multiplexing

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions. The following table describes the peripheral signals multiplexed to the PORT I/O pins.

**Table 6-1. PORT Function Multiplexing**

| 14-pin | 8-pin | Pin name | Special | INT(3) | ADC(3) | AC | USART | Timer | Programming(8) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1(1) | $V_{CC}$ | | | | | | | |
| 2 | 2 | PA[0](2) | CLKI | PCINT0 | ADC0 | AIN0 | | T0 | TPICLK |
| 3 | 3 | PA[1](5) | | PCINT1 | ADC1 | AIN1 | | OC0B | TPIDATA |
| 4 | 4 | PA[2] | RESET | PCINT2 | | | | | RESET |
| 5 | - | PA[3](9) | | PCINT3 | | | | OC0A | |
| 6 | - | PA[4](9) | | PCINT4 | | | | ICP0 | |
| 7 | - | PA[5](5)(9) | | PCINT5 | ADC2 | | | OC0B | |
| 8 | - | PA[6] | | PCINT6 | ADC3 | | | | |
| 9 | - | PA[7] | | PCINT7 | | | | | |
| 10 | - | PB[0] | | PCINT8 | ADC4 | | | | |
| 11 | 5 | PB[1](6) | CLKO | PCINT9/INT0 | ADC5 | | XCK0 | OC0A | |
| 12 | 6 | PB[2](7) | | PCINT10 | ADC6 | | TxD0 | ICP0 | |
| 13 | 7 | PB[3](4)(9) | | PCINT11 | ADC7 | ACO | RxD0 | T0 | |
| 14 | 8 | GND | | | | | | | |

**Note:**

1. On the 8-pin UDFN package, the thermal pad should not be connected as well.
2. Priority of CLKI is higher than ADC0. When EXT_CLK is enabled, ADC channel will not work and DIDR0 will not disable the digital input buffer.
3. When both PCINT and the corresponding ADC channel are enabled, the digital input buffer will not be disabled.
4. When ACO is enabled, ADC, TC and USART RX inputs are not disabled.
5. When OC0B is enabled, ADC and AC will continue to receive inputs on that channel if enabled.
6. When CLKO is enable in PB[1], OCA will get lower priority.
7. When USART is enabled, the users must ensure that ADC channel corresponding to the TxD0 pin is not used. Because DIDR0 register will only control the input buffer, not the output part.
8. During reset/external programming, all pins are treated as inputs and outputs are disabled.
9. Alternative location when enabling T/C Remap

# 7. General Information

## 7.1. Resources

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

## 7.2. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

## 7.3. About Code Examples

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

**Related Links**

Reset and Interrupt Handling on page 21
Code Examples on page 50

# 8. AVR CPU Core

## 8.1. Overview

The Atmel®AVR® core combines a rich instruction set with 16 general purpose working registers. All the 16 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.

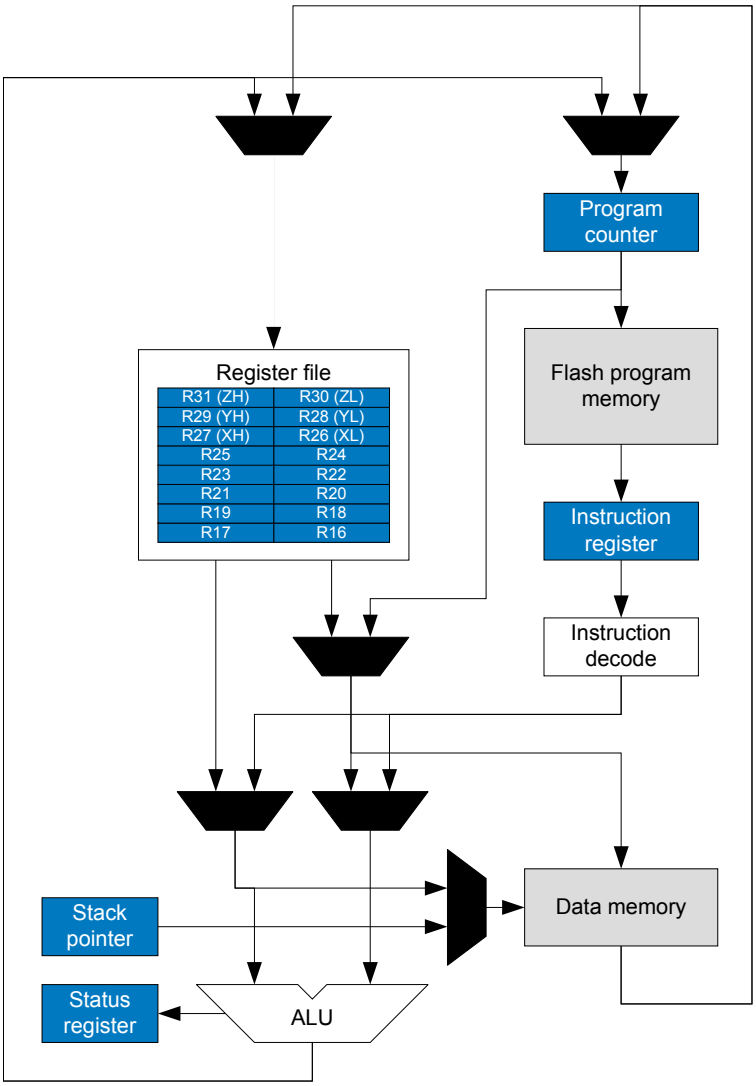
## 8.2. Features

- Advanced RISC Architecture
- 54 Powerful Instructions
- Mostly Single Clock Cycle Execution
- 16 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 12 MIPS Throughput at 12MHz

## 8.3. Block Diagram

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

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

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format but 32-bit wide instructions also exist. The actual instruction set varies, as some devices only implement a part of the instruction set.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the four 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 as the data space locations, 0x0000 - 0x003F. See *Instruction Set Summary* section for a detailed description.

## 8.4. ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 16 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See *Instruction Set Summary* section for a detailed description.

## 8.5. 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.

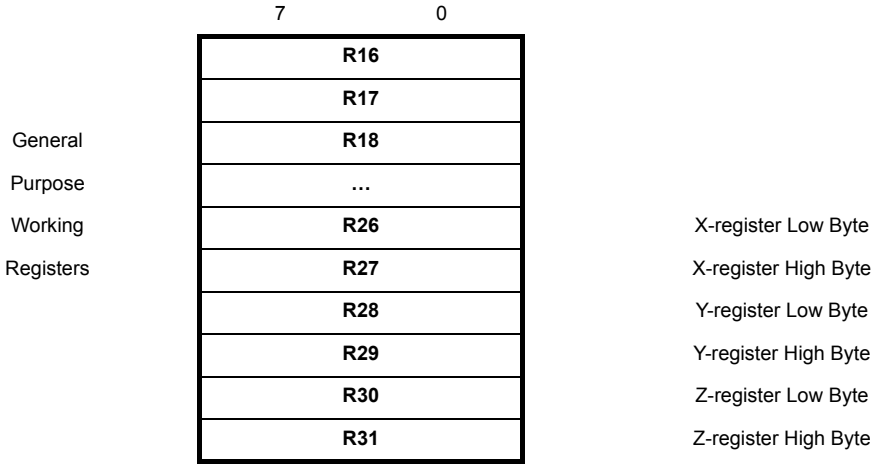
The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

## 8.6. 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
- One 16-bit output operand and one 16-bit result input

**Figure 8-2. AVR CPU General Purpose Working Registers**




**Note:** A typical implementation of the AVR register file includes 32 general purpose registers but ATtiny102/ATtiny104 implement only 16 registers. For reasons of compatibility the registers are numbered R16...R31, not R0...R15.

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

## 8.7. 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).

## 8.8. Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack, and the Stack Pointer must be set to point above 0x40.

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 |
| ICALL<br>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<br>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.

**Related Links**

SPL and SPH on page 24
SRAM Data Memory on page 27

## 8.9. Accessing 16-bit Registers

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

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

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

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

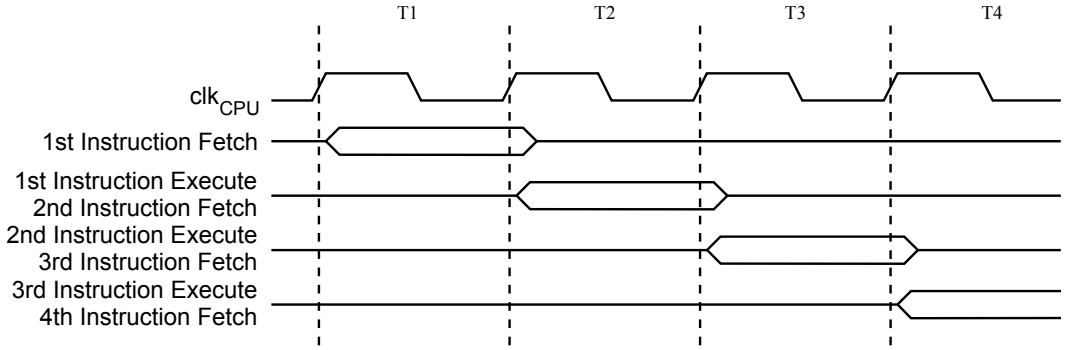
Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

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

## 8.10. Instruction Execution Timing

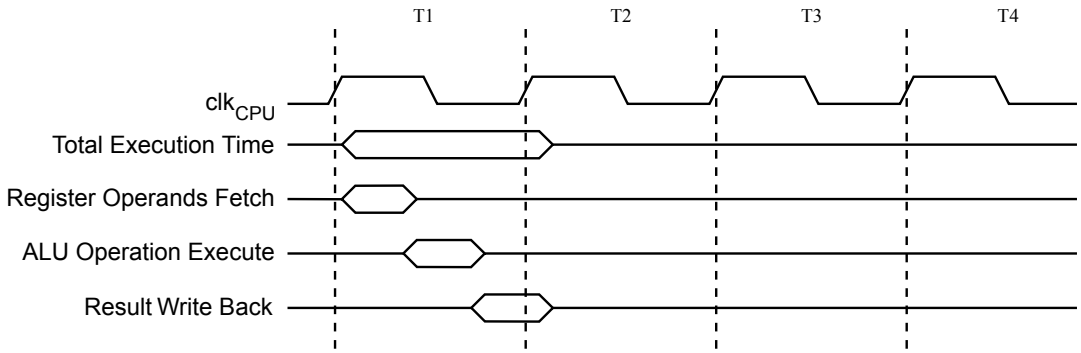
This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock $clk_{CPU}$, directly generated from the selected clock source for the chip. No internal clock division is used. The Figure below shows the parallel instruction fetches and instruction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




## 8.11. 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.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in *Interrupts*. 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.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then

interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts:

- The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.
- The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

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

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

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction.

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

**Assembly Code Example**

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

**Note:** See *Code Examples*

**Related Links**

Interrupts on page 55

### 8.11.1. Interrupt Response Time

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

## 8.12. Register Description

### 8.12.1. Configuration Change Protection Register

**Name:** CCP
**Offset:** 0x3C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCP[7:0] | | | | | | | |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – CCP[7:0]: Configuration Change Protection**
In order to change the contents of a protected I/O register, the CCP register must first be written with the correct signature. After CCP is written, the protected I/O registers may be written to during the next four CPU instruction cycles. All interrupts are ignored during these cycles. After these cycles interrupts are automatically handled again by the CPU, and any pending interrupts will be executed according to their priority.

When the protected I/O register signature is written, CCP[0] will read as one as long as the protected feature is enabled, while CCP[7:1] will always read as zero.

When the NVM self-programming signature is written, CCP[1] will read as one for four CPU instruction cycles , other bits will read as zero and CCP[1] will be cleared automatically after four cycles. The software should write data to flash high byte within this four clock cycles to execute self-programming.

**Table 8-2. Signatures Recognized by the Configuration Change Protection Register**

| Signature | Group | Description |
|---|---|---|
| 0xD8 | IOREG: CLKMSR, CLKPSR, WDTCSR | Protected I/O register |
| 0xE7 | SPM | NVM self-programming enable |

**Note:** Bit 0 and 1 have R/W access. The other bits only have W access.

### 8.12.2. Stack Pointer Register Low and High byte

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

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

**Name:** SPL and SPH
**Offset:** 0x3D
**Reset:** 0x5F
**Property:** -

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Access | R | R | R | R | R | R | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | | |

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

**Bits 0, 1, 2, 3, 4, 5, 6 – SPn: Stack Pointer Register**
SPL and SPH are combined into SP.

### 8.12.3. Status Register

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

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

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

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

**Bit 6 – T: 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.

# 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. All memory spaces are linear and regular.

## 9.2. Features

- Non-volatile Program and Data Memories
- 1024 Bytes of In-system Programmable Flash Program Memory
- 32 Bytes Internal SRAM
- Flash Write/Erase Cycles: 10,000
- Data Retention: 20 Years at 85°C / 100 Years at 25°C
- Self-programming Flash on Full Operating Voltage Range (1.8 – 5.5V)

## 9.3. In-System Reprogrammable Flash Program Memory

The ATtiny102/ATtiny104 contains 1024 Bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 512 x 16.

The device Program Counter (PC) is 9 bits wide, thus addressing the 512 program memory locations, starting at 0x000. *Memory Programming* contains a detailed description on Flash data serial downloading.

Constant tables can be allocated within the entire address space of program memory by using load/store instructions. Since program memory can not be accessed directly, it has been mapped to the data memory. The mapped program memory begins at byte address 0x4000 in data memory. Although programs are executed starting from address 0x000 in program memory it must be addressed starting from 0x4000 when accessed via the data memory.

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

**Related Links**

Instruction Execution Timing on page 21
MEMPROG- Memory Programming on page 185

## 9.4. SRAM Data Memory

Data memory locations include the I/O memory, the internal SRAM memory, the Non-Volatile Memory (NVM) Lock bits, and the Flash memory. The following figure shows how the ATtiny102/ATtiny104 SRAM Memory is organized.

The first 64 locations are reserved for I/O memory, while the following 32 data memory locations address the internal data SRAM.

The Non-Volatile Memory (NVM) Lock bits and all the Flash memory sections are mapped to the data memory space. These locations appear as read-only for device firmware.

The four different addressing modes for data memory are direct, indirect, indirect with pre-decrement, and indirect with post-increment. In the register file, registers R26 to R31 function as pointer registers for indirect addressing.

The IN and OUT instructions can access all 64 locations of I/O memory. Direct addressing using the LDS and STS instructions reaches the 128 locations between 0x0040 and 0x00BF.

The indirect addressing reaches the entire data memory space. When using indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

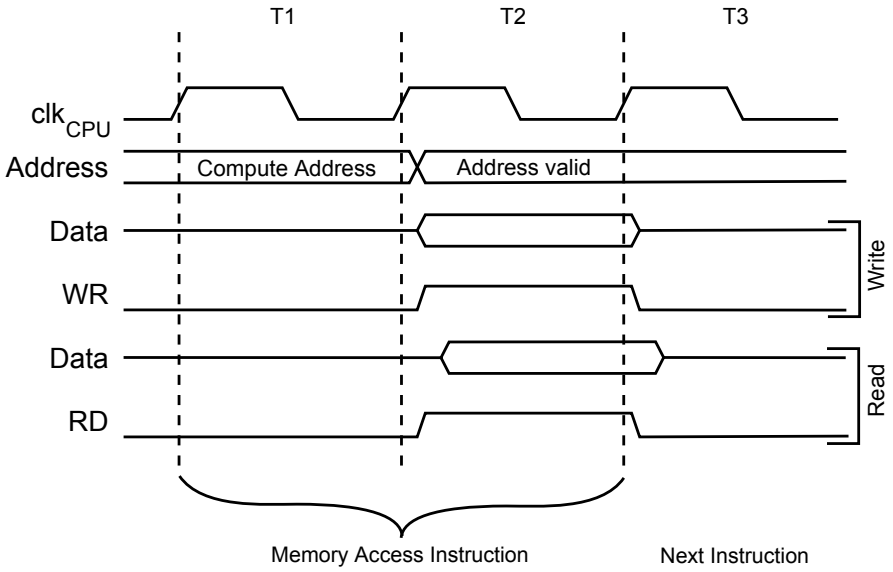
**Figure 9-1. Data Memory Map (Byte Addressing)**

| | |
|---|---|
| **I/O SPACE** | 0x0000 ... 0x003F |
| **SRAM DATA MEMORY** | 0x0040 ... 0x005F |
| **(reserved)** | 0x0060 ... 0x3EFF |
| **NVM LOCK BITS** | 0x3F00 ... 0x3F01 |
| **(reserved)** | 0x3F02 ... 0x3F3F |
| **CONFIGURATION BITS** | 0x3F40 ... 0x3F41 |
| **(reserved)** | 0x3F42 ... 0x3F7F |
| **CALIBRATION BITS** | 0x3F80 ... 0x3F81 |
| **(reserved)** | 0x3F82 ... 0x3FBF |
| **DEVICE ID BITS** | 0x3FC0 ... 0x3FC3 |
| **(reserved)** | 0x3FC4 ... 0x3FFF |
| **FLASH PROGRAM MEMORY** | 0x4000 ... 0x41FF/0x43FF |
| **(reserved)** | 0x4400 ... 0xFFFF |

### 9.4.1. Data Memory Access Times

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

**Figure 9-2. On-chip Data SRAM Access Cycles**




## 9.5. I/O Memory

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

All ATtiny102/ATtiny104 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD and ST instructions, transferring data between the 16 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, except USART registers. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the *Instruction Set Summary* section for more details.

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**

Register Summary on page 244

# 10. Clock System

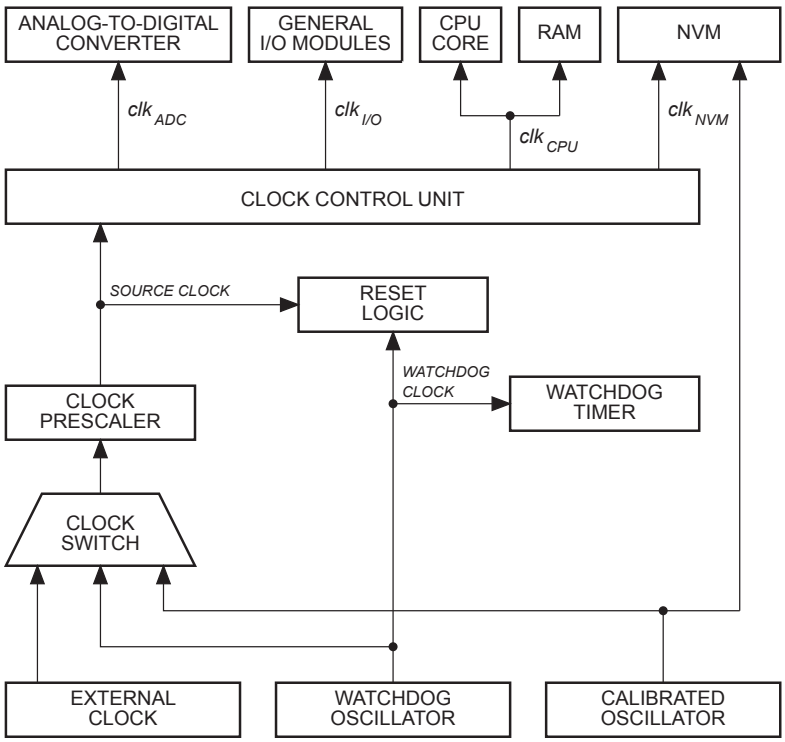
## 10.1. Overview

This chapter summarizes the clock distribution and terminology in the ATtiny102/ATtiny104 device.

## 10.2. Clock 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, as described in the section on Power Management and Sleep Modes. The clock systems are detailed below.

**Figure 10-1. Clock Distribution**




**Related Links**

Power Management and Sleep Modes on page 39

## 10.3. Clock Subsystems

### 10.3.1. 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 System Registers and the SRAM data memory. Halting the CPU clock inhibits the core from performing general operations and calculations.

### 10.3.2. I/O Clock – $clk_{I/O}$

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

### 10.3.3. NVM Clock – $clk_{NVM}$

The NVM clock controls operation of the Non-Volatile Memory Controller. The NVM clock is usually active simultaneously with the CPU clock.

### 10.3.4. 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.

## 10.4. Clock Sources

The device has the following clock source options, selectable by Clock Main Select Bits in Clock Main Settings Register (CLKMSR.CLKMS). All synchronous clock signals are derived from the main clock. The three alternative sources for the main clock are as follows:

- Calibrated Internal 8MHz Oscillator
- External Clock
- Internal 128kHz Oscillator.

Refer to description of Clock Main Select Bits in Clock Main Settings Register (CLKMSR.CLKMS) for how to select and change the active clock source.

### 10.4.1. Calibrated Internal 8MHz Oscillator

The calibrated internal oscillator provides an approximately 8MHz clock signal. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user.

During reset, hardware loads the calibration byte into the Oscillator Calibration Register (OSCCAL) register and thereby automatically calibrates the oscillator. The accuracy of this calibration is shown as *Factory calibration* in *Accuracy of Calibrated Internal Oscillator* of Electrical Characteristics chapter.

When this oscillator is used as the main clock, the watchdog oscillator will still be used for the watchdog timer and reset time-out. For more information on the pre-programmed calibration value, see section *Calibration Section*.

It is possible to reach higher accuracies than factory defaults, especially when the application allows temperature and voltage ranges to be narrowed. The firmware can reprogram the calibration data in OSCCAL either at start-up or during run-time. The continuous, run-time calibration method allows firmware to monitor voltage and temperature and compensate for any detected variations.

When this oscillator is used as the chip clock, it will still be used for the Watchdog Timer and for the Reset Time-out.

**Related Links**

Calibration Section on page 191
Accuracy of Calibrated Internal Oscillator on page 213
Internal Oscillator Speed on page 236

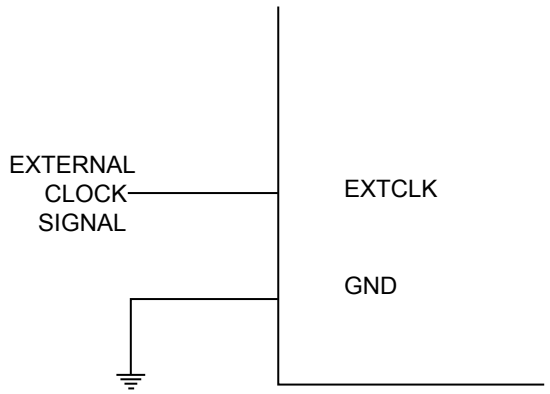
### 10.4.2. External Clock

To drive the device from an external clock source, CLKI should be driven as shown in the Figure below. To run the device on an external clock, the CLKMSR.CLKMS must be programmed to '0b10'.

**Table 10-1. External Clock Frequency**

| Frequency | CLKMSR.CLKMS |
|---|---|
| 0 - 12MHz | 0b10 |

**Figure 10-2. External Clock Drive Configuration**




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

### 10.4.3. Internal 128kHz Oscillator

The internal 128kHz oscillator is a low power oscillator providing a clock of 128kHz. The frequency depends on supply voltage, temperature and batch variations. This clock may be select as the main clock by setting the CLKMSR.CLKMS to 0b01.

### 10.4.4. Switching Clock Source

The main clock source can be switched at run-time using the CLKMSR – Clock Main Settings Register. When switching between any clock sources, the clock system ensures that no glitch occurs in the main clock.

### 10.4.5. Default Clock Source

The calibrated internal 8MHz oscillator is always selected as main clock when the device is powered up or has been reset. The synchronous system clock is the main clock divided by 8, controlled by the System Clock Prescaler. The Clock Prescaler Select Bits in Clock Prescale Register (CLKPSR.CLKPS) can be written later to change the system clock frequency. See section “System Clock Prescaler”.

## 10.5. System Clock Prescaler

The system clock is derived from the main clock via the System Clock Prescaler. The system clock can be divided by setting the “CLKPSR – Clock Prescale Register”. The system clock prescaler can be used to decrease power consumption at times when requirements for processing power is low or to bring the system clock within limits of maximum frequency. The prescaler can be used with all main clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals.

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

### 10.5.1. Switching Prescaler Setting

When switching between prescaler settings, the system clock prescaler ensures that no glitch occurs in the system clock and that no intermediate frequency is higher than neither the clock frequency corresponding 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 main 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, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

From the time the CLKPSR.CLKPS values are written, it takes between $T_1 + T_2$ and $T_1 + 2*T_2$ before the new clock frequency is active. In this interval, two active clock edges are produced. Here, $T_1$ is the previous clock period, and $T_2$ is the period corresponding to the new prescaler setting.

## 10.6. Starting

### 10.6.1. Starting from Reset

The internal reset is immediately asserted when a reset source goes active. The internal reset is kept asserted until the reset source is released and the start-up sequence is completed. The start-up sequence includes three steps, as follows.

1. The first step after the reset source has been released consists of the device counting the reset start-up time. The purpose of this reset start-up time is to ensure that supply voltage has reached sufficient levels. The reset start-up time is counted using the internal 128kHz oscillator.
   **Note:** The actual supply voltage is not monitored by the start-up logic. The device will count until the reset start-up time has elapsed even if the device has reached sufficient supply voltage levels earlier.
2. The second step is to count the oscillator start-up time, which ensures that the calibrated internal oscillator has reached a stable state before it is used by the other parts of the system. The calibrated internal oscillator needs to oscillate for a minimum number of cycles before it can be considered stable.
3. The last step before releasing the internal reset is to load the calibration and the configuration values from the Non-Volatile Memory to configure the device properly. The configuration time is listed in the next table.

There are two start-up time options which will be supported :

- Normal start-up time: 64ms
- Fast start-up time: 8ms

**Table 10-2. Start-up Times when Using the Internal Calibrated Oscillator with Normal start-up time**

| Reset | Oscillator | Configuration | Total start-up time |
|---|---|---|---|
| 64ms | 6cycles | 21cycles | 64ms + 6 oscillator cycles + 21 system clock cycles (1) |

**Table 10-3. Start-up Times when Using the Internal Calibrated Oscillator with shorter startup time**

| Reset | Oscillator | Configuration | Total start-up time |
|---|---|---|---|
| 8ms | 6cycles | 21cycles | 8 ms + 6 oscillator cycles + 21 system clock cycles (1) |

**Note:**

1. After powering up the device or after a reset the system clock is automatically set to calibrated internal 8MHz oscillator, divided by 8

### 10.6.2. Starting from Power-Down Mode

When waking up from Power-Down sleep mode, the supply voltage is assumed to be at a sufficient level and only the oscillator start-up time is counted to ensure the stable operation of the oscillator. The oscillator start-up time is counted on the selected main clock, and the start-up time depends on the clock selected.

**Table 10-4. Start-up Time from Power-Down Sleep Mode.**

| Oscillator start-up time | Total start-up time |
|---|---|
| 6 cycles | 6 oscillator cycles (1) |

**Note:** 1. The start-up time is measured in main clock oscillator cycles.

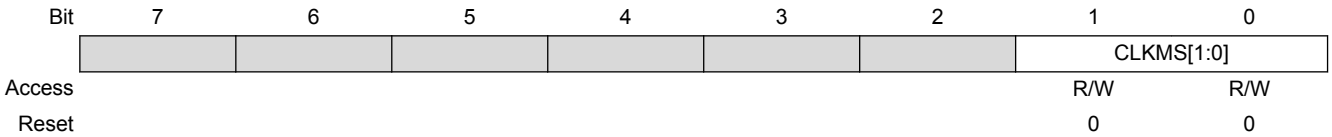
### 10.6.3. Starting from Idle / ADC Noise Reduction / Standby Mode

When waking up from Idle, ADC Noise Reduction or Standby Mode, the oscillator is already running and no oscillator start-up time is introduced.

## 10.7. Register Description

### 10.7.1. Clock Main Settings Register

**Name:** CLKMSR
**Offset:** 0x37
**Reset:** 0x00
**Property:** -




**Bits 1:0 – CLKMS[1:0]: Clock Main Select Bits**
These bits select the main clock source of the system. The bits can be written at run-time to switch the source of the main clock. The clock system ensures glitch free switching of the main clock source.

**Table 10-5. Selection of Main Clock**

| CLKM | Main Clock Source |
|---|---|
| 00 | Calibrated Internal 8MHz Oscillator |
| 01 | Internal 128kHz Oscillator (WDT Oscillator) |
| 10 | External clock |
| 11 | Reserved |

To avoid unintentional switching of main clock source, a protected change sequence must be followed to change the CLKMS bits, as follows:

1. Write the signature for change enable of protected I/O register to register CCP.
2. Within four instruction cycles, write the CLKMS bits with the desired value.

### 10.7.2. Oscillator Calibration Register

**Name:** OSCCAL
**Offset:** 0x39
**Reset:** xxxxxxxx
**Property:** -

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

**Bits 7:0 – CAL[7:0]: Oscillator Calibration Value**
The Oscillator Calibration Register is used to trim the Calibrated Internal RC Oscillator to remove process variations from the oscillator frequency. A pre-programmed calibration value is automatically written to this register during chip reset, giving the Factory calibrated frequency as specified in *Accuracy of Calibrated Internal Oscillator* section of *Electrical Characteristics* chapter.

The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in *Accuracy of Calibrated Internal Oscillator* section of *Electrical Characteristics* chapter. Calibration outside that range is not guaranteed.

The lowest oscillator frequency is reached by programming these bits to zero. Increasing the register value increases the oscillator frequency.

Note that this oscillator is used to time Flash write accesses, and write times will be affected accordingly. Do not calibrate to more than 8.8MHz if Flash is to be written. Otherwise, the Flash write may fail.

To ensure stable operation of the MCU the calibration value should be changed in small steps. A step change in frequency of more than 2% from one cycle to the next can lead to unpredictable behavior. Also, the difference between two consecutive register values should not exceed 0x20. If these limits are exceeded the MCU must be kept in reset during changes to clock frequency.

### 10.7.3. Clock Prescaler Register

**Name:** CLKPSR
**Offset:** 0x36
**Reset:** 0x00000011
**Property:** -

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

**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.

**Table 10-6. Clock Prescaler Select**

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

To avoid unintentional changes of clock frequency, a protected change sequence must be followed to change the CLKPS bits:

1. Write the signature for change enable of protected I/O register to register CCP
2. Within four instruction cycles, write the desired value to CLKPS bits

At start-up, CLKPS bits are reset to 0b0011 to select the clock division factor of 8. If the selected clock source has a frequency higher than the maximum allowed the application software must make sure a sufficient division factor is used. To make sure the write procedure is not interrupted, interrupts must be disabled when changing prescaler settings.

# 11. Power Management and Sleep Modes

## 11.1. Overview

The high performance and industry leading code efficiency makes the AVR microcontrollers an ideal choise for low power applications. In addition, sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

## 11.2. Features

- Minimizing Power Consumption
- Sleep modes:
  - Idle
  - ADC Noise Reduction Mode
  - Power-Down Mode
  - Standby Mode

## 11.3. Sleep Modes

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

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

| | Active Clock Domains | | | | Oscillators | Wake-up Sources | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Sleep Mode | $clk_{CPU}$ | $clk_{NVM}$ | $clk_{IO}$ | $clk_{ADC}$ | Main Clock Source Enabled | INT0 and Pin Change | ADC | Other I/O | Watchdog Interrupt | VLM Interrupt |
| Idle | | | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| ADC Noise Reduction | | | | Yes | Yes | Yes(1) | Yes | | Yes | Yes |
| Standby | | | | | Yes | Yes(1) | | | Yes | |
| Power-down | | | | | | Yes(1) | | | Yes | |

**Note:**

1. For INT0, only level interrupt.

To enter any of the four sleep modes (Idle, ADC Noise Reduction, Power-down or Standby), 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) select which sleep mode will be activated by the SLEEP instruction.

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.

**Note:** If a level triggered interrupt is used for wake-up the changed level must be held for some time to wake up the MCU (and for the MCU to enter the interrupt service routine). See *External Interrupts* for details.

**Related Links**

SMCR on page 43

Interrupts on page 55

### 11.3.1. Idle Mode

When the SMCR.SM is written to '0x000', the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the Analog Comparator, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts $clk_{CPU}$ and $clk_{NVM}$, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the timer overflow. 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.ACD). This will reduce power consumption in Idle mode. If the ADC is enabled , a conversion starts automatically when this mode is entered.

**Related Links**

ACSRA on page 162

### 11.3.2. ADC Noise Reduction Mode

When the SMCR.SM is written to '0x001', the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the external interrupts and the Watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, $clk_{CPU}$, and $clk_{NVM}$, 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.

This mode is available in all devices equipped with an ADC.

### 11.3.3. Power-Down Mode

When the SMCR.SM is written to '0x010', the SLEEP instruction makes the MCU enter Power-Down mode. In this mode, the external Oscillator is stopped, while the external interrupts and the Watchdog continue operating (if enabled).

Only an these events can wake up the MCU:

- Watchdog System Reset
- External level interrupt on INT0
- Pin change interrupt

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

### 11.3.4. Standby Mode

When the SMCR.SM is written to '0x100', the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-Down with the exception that the Oscillator is kept running. This reduces wake-up time, because the oscillator is already running and doesn't need to be started up.

## 11.4. Power Reduction Register

The Power Reduction Register (PRR) provides a method to stop the clock to individual peripherals to reduce power consumption. When the clock for a peripheral is stopped then:

- The current state of the peripheral is frozen.

- The associated registers can not be read or written.
- Resources used by the peripheral will remain occupied.

The peripheral should in most cases be disabled before stopping the clock. Clearing the PRR bit wakes up the peripheral and puts it in the same state as before shutdown.

Peripheral 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.

**Related Links**

PRR on page 44
Supply Current of I/O Modules on page 222

## 11.5. 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.

### 11.5.1. Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. In the power-down mode, the analog comparator is automatically disabled. See *Analog Comparator* for further details.

When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode.

**Related Links**

AC - Analog Comparator on page 160

### 11.5.2. Analog to Digital Converter

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

**Related Links**

ADC - Analog to Digital Converter on page 166

### 11.5.3. Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Analog Comparator or the 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.

### 11.5.4. 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**

Watchdog Timer on page 48

### 11.5.5. 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 ($clk_{I/O}$) is stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section *Digital Input Enable and Sleep Modes* 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 0 (DIDR0).

**Related Links**

Digital Input Enable and Sleep Modes on page 69

DIDR0 on page 165

## 11.6. Register Description

### 11.6.1. Sleep Mode Control Register

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

**Name:** SMCR
**Offset:** 0x3A
**Reset:** 0x00
**Property:**

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

**Bits 3:1 – SM[2:0]: Sleep Mode Select**

The SM[2:0] bits select between the five available sleep modes.

**Table 11-2. Sleep Mode Select**

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

**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.

### 11.6.2. Power Reduction Register

**Name:** PRR
**Offset:** 0x35
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PRUSART0 | PRADC | PRTIM0 |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bit 2 – PRUSART0: Power Reduction USART0**
Writing a logic one to this bit shuts down the USART by stopping the clock to the module. When waking up the USART again, the USART should be re initialized to ensure proper operation.

**Bit 1 – 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.

**Bit 0 – 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.

# 12. SCRST - System Control and Reset

## 12.1. Overview

The reset logic manages the reset of the microcontroller. It issues a microcontroller reset, sets the device to its initial state and allows the reset source to be identified by software.

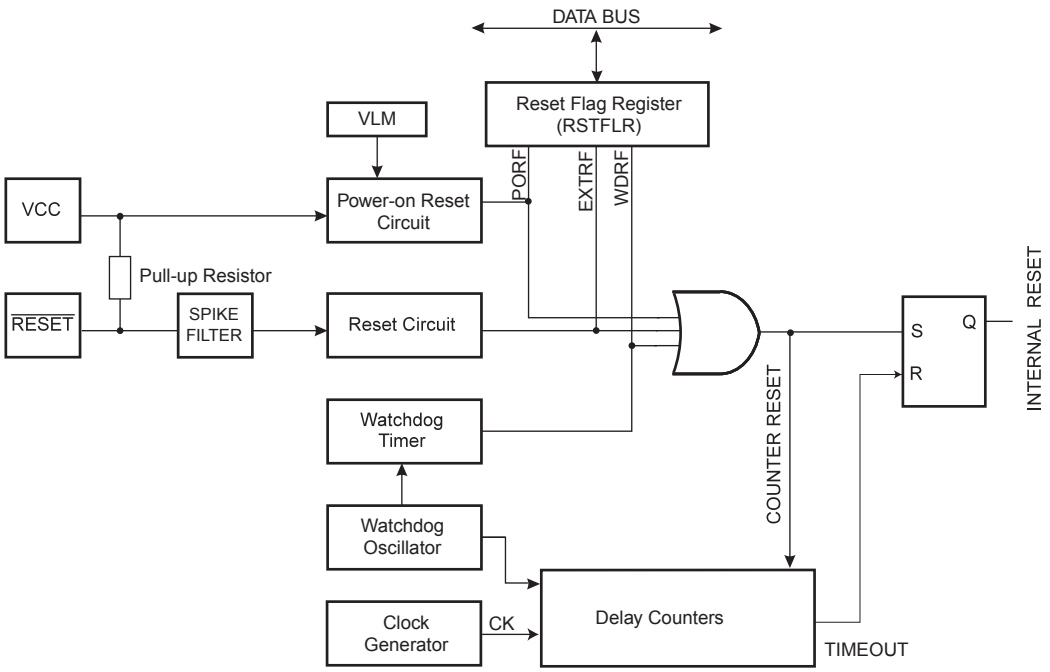
## 12.2. Features

- Reset The Microcontroller to Initial State.
- Multiple Reset Sources:
  - Power-on Reset
  - $V_{CC}$ Level Monitoring (VLM) Reset
  - External Reset
  - Watchdog System Reset

## 12.3. 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 Relative Jump instruction (RJMP) to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. The circuit diagram in the next shows the reset logic. Electrical parameters of the reset circuitry are defined in section *System and Reset Characteristics*.

**Figure 12-1. Reset Logic**




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

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts.

**Related Links**

Starting from Reset on page 33
System and Reset Characteristics on page 214

## 12.4. Reset Sources

The device has four sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is less than the Power-on Reset threshold ($V_{POT}$).
- VLM ($V_{CC}$ Level Monitoring) Reset. The MCU is reset when voltage on the $V_{CC}$ pin is below the selected trigger level.
- 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.

### 12.4.1. Power-on Reset

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

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

**Figure 12-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to $V_{CC}$**

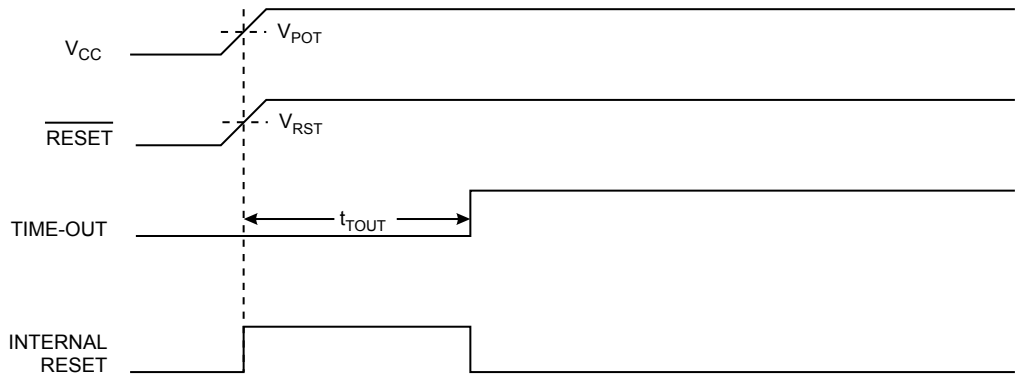
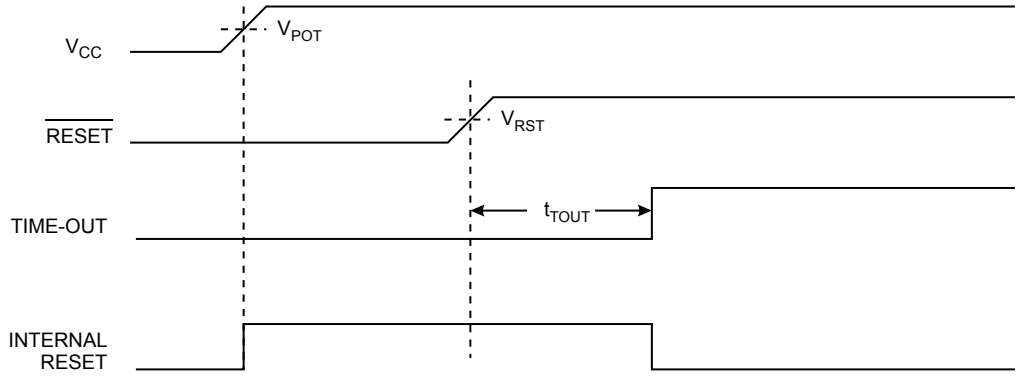


**Figure 12-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally**




**Related Links**

System and Reset Characteristics on page 214

### 12.4.2. $V_{CC}$ Level Monitoring

ATtiny102/ATtiny104 have a $V_{CC}$ Level Monitoring (VLM) circuit that compares the voltage level at the $V_{CC}$ pin against fixed trigger levels. The trigger levels are set with VLM[2:0] bits, see VLMCSR – $V_{CC}$ Level Monitoring Control and Status register.

The VLM circuit provides a status flag, VLMF, that indicates if voltage on the $V_{CC}$ pin is below the selected trigger level. The flag can be read from VLMCSR, but it is also possible to have an interrupt generated when the VLMF status flag is set. This interrupt is enabled by the VLMIE bit in the VLMCSR register. The flag can be cleared by changing the trigger level or by writing it to zero. The flag is automatically cleared when the voltage at $V_{CC}$ rises back above the selected trigger level.

The VLM can also be used to improve reset characteristics at falling supply. Without VLM, the Power-On Reset (POR) does not activate before supply voltage has dropped to a level where the MCU is not necessarily functional any more. With VLM, it is possible to generate a reset earlier.

When active, the VLM circuit consumes some power, as illustrated in the figure of $V_{CC}$ Level Monitor Current vs. $V_{CC}$ in *Typical Characteristics*. To save power the VLM circuit can be turned off completely, or it can be switched on and off at regular intervals. However, detection takes some time and it is therefore recommended to leave the circuitry on long enough for signals to settle. See *$V_{CC}$ Level Monitor*.

When VLM is active and voltage at $V_{CC}$ is above the selected trigger level operation will be as normal and the VLM can be shut down for a short period of time. If voltage at $V_{CC}$ drops below the selected threshold the VLM will either flag an interrupt or generate a reset, depending on the configuration.

When the VLM has been configured to generate a reset at low supply voltage it will keep the device in reset as long as $V_{CC}$ is below the reset level. If supply voltage rises above the reset level the condition is removed and the MCU will come out of reset, and initiate the power-up start-up sequence.

If supply voltage drops enough to trigger the POR then PORF is set after supply voltage has been restored.

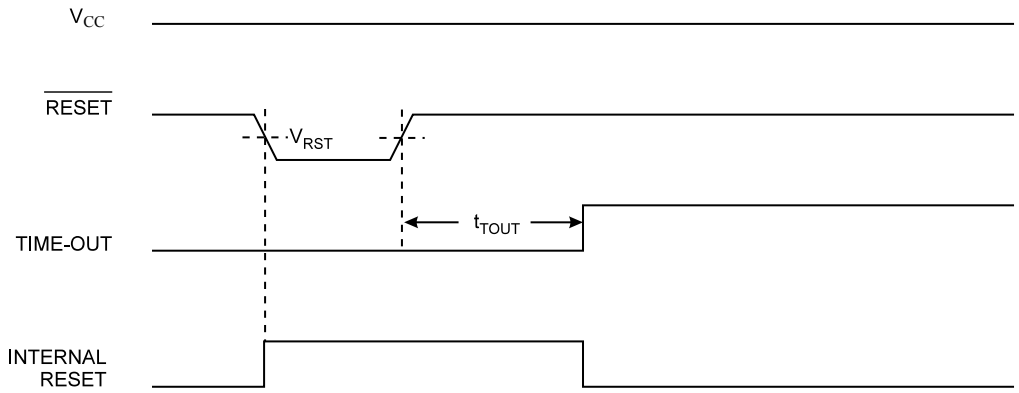
**Related Links**

VLMCSR on page 53
Electrical Characteristics on page 210
VCC Level Monitor on page 215
Typical Characteristics on page 218

### 12.4.3. 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**



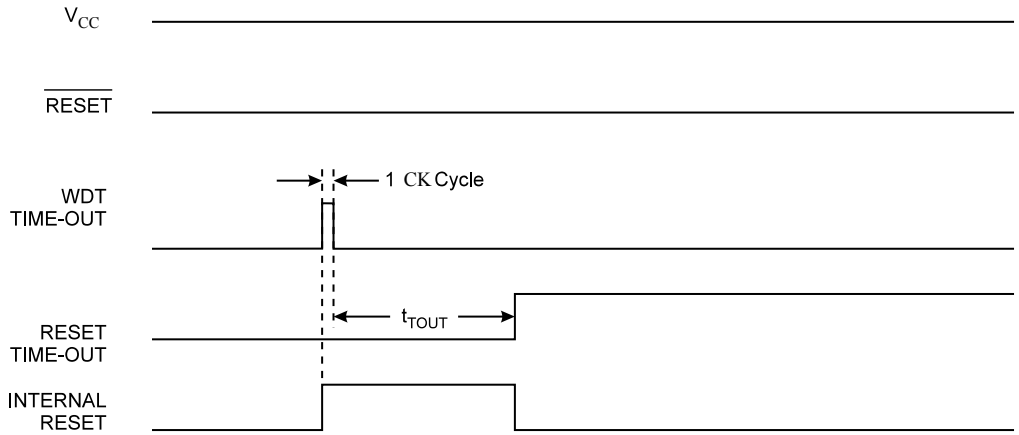

**Related Links**

System and Reset Characteristics on page 214

### 12.4.4. 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 12-5. Watchdog System Reset During Operation**




## 12.5. 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.

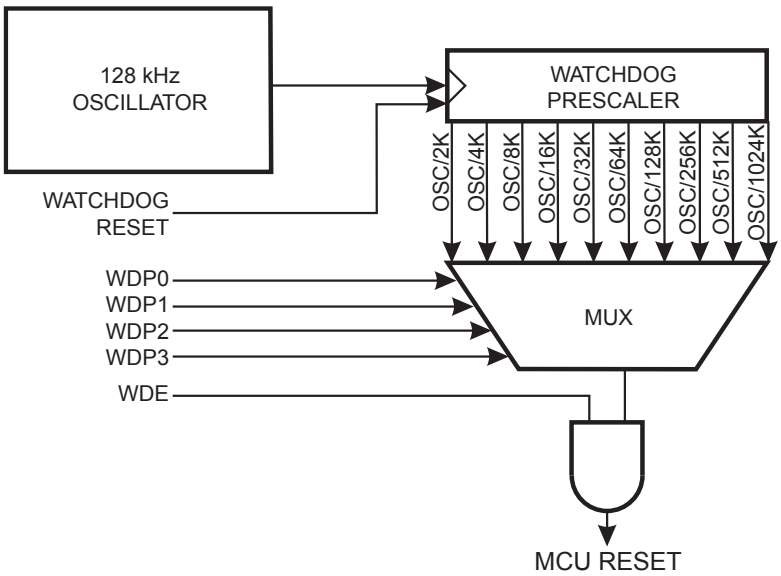
Refer to Watchdog System Reset for details on how to configure the watchdog timer.

### 12.5.1. Overview

The Watchdog Timer is clocked from an on-chip oscillator, which runs at 128kHz, as the next figure. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted. The Watchdog

Reset (WDR) instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a device reset occurs. Ten different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the device resets and executes from the Reset Vector.

**Figure 12-6. Watchdog Timer**




The Watchdog Timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the Watchdog to wake-up from Power-down.

To prevent unintentional disabling of the Watchdog or unintentional change of time-out period, two different safety levels are selected by the fuse WDTON. See Procedure for Changing the Watchdog Timer Configuration for details.

**Table 12-1. WDT Configuration as a Function of the Fuse Settings of WDTON**

| WDTON | Safety Level | WDT Initial State | How to Disable the WDT | How to Change Time-out |
|---|---|---|---|---|
| Unprogrammed | 1 | Disabled | Protected change sequence | No limitations |
| Programmed | 2 | Enabled | Always enabled | Protected change sequence |

### 12.5.2. Procedure for Changing the Watchdog Timer Configuration

The sequence for changing configuration differs between the two safety levels, as follows:

#### 12.5.2.1. Safety Level 1

In this mode, the Watchdog Timer is initially disabled, but can be enabled by writing the WDE bit to one without any restriction. A special sequence is needed when disabling an enabled Watchdog Timer. To disable an enabled Watchdog Timer, the following procedure must be followed:

1. Write the signature for change enable of protected I/O registers to register CCP
2. Within four instruction cycles, in the same operation, write WDE and WDP bits

#### 12.5.2.2. Safety Level 2

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

1. Write the signature for change enable of protected I/O registers to register CCP
2. Within four instruction cycles, write the WDP bit. The value written to WDE is irrelevant

### 12.5.3. Code Examples

The following code example shows how to turn off the WDT. The example assumes that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
WDT_off:
wdr
; Clear WDRF in RSTFLR
in r16, RSTFLR
andi r16, ~(1<<WDRF)
out RSTFLR, r16
; Write signature for change enable of protected I/O register
ldi r16, 0xD8
out CCP, r16
; Within four instruction cycles, turn off WDT
ldi r16, (0<<WDE)
out WDTCSR, r16
ret
```

**Note:** See *About Code Examples*.

## 12.6. Register Description

### 12.6.1. Watchdog Timer Control Register

**Name:** WDTCSR
**Offset:** 0x31
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDIF | WDIE | WDP3 | | WDE | WDP2 | WDP1 | WDP0 |
| Access | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | x | 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 logic one to the flag. When the I-bit in SREG and WDIE are set, the Watchdog Time-out Interrupt is executed.

**Bit 6 – WDIE: Watchdog Interrupt Enable**
When this bit is written to one 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 time-out in the Watchdog Timer occurs. If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode).

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

**Table 12-2. 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 3 – WDE: Watchdog System Reset Enable**
WDE is overridden by WDRF in RSTFLR. 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 5,2:0 – WDPn: Watchdog Timer Prescaler [n=3:0]**

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 table below.

**Table 12-3. Watchdog Timer Prescale Select**

| WDP[3:0] | Number of WDT Oscillator Cycles | Typical Time-out at $V_{CC}$ = 5.0V |
|---|---|---|
| 0000 | 2K (2048) cycles | 16ms |
| 0001 | 4K (4096) cycles | 32ms |
| 0010 | 8K (8192) cycles | 64ms |
| 0011 | 16K (16384) cycles | 0.125s |
| 0100 | 32K (32768) cycles | 0.25s |
| 0101 | 64K (65536) cycles | 0.5s |
| 0110 | 128K (131072) cycles | 1.0s |
| 0111 | 256K (262144) cycles | 2.0s |
| 1000 | 512K (524288) cycles | 4.0s |
| 1001 | 1024K (1048576) cycles | 8.0s |
| 1010 | Reserved | Reserved |
| 1011 | Reserved | Reserved |
| 1100 | Reserved | Reserved |
| 1101 | Reserved | Reserved |
| 1110 | Reserved | Reserved |
| 1111 | Reserved | Reserved |

### 12.6.2. VCC Level Monitoring Control and Status register

**Name:** VLMCSR
**Offset:** 0x34
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | VLMF | VLMIE | | | | VLM[2:0] | | |
| Access | R | R/W | | | | R/W | R/W | R/W |
| Reset | 0 | 0 | | | | 0 | 0 | 0 |

**Bit 7 – VLMF: VLM Flag**
This bit is set by the VLM circuit to indicate that a voltage level condition has been triggered. The bit is cleared when the trigger level selection is set to “Disabled”, or when voltage at $V_{CC}$ rises above the selected trigger level.

**Bit 6 – VLMIE: VLM Interrupt Enable**
When this bit is set the VLM interrupt is enabled. A VLM interrupt is generated every time the VLMF flag is set.

**Bits 2:0 – VLM[2:0]: Trigger Level of Voltage Level Monitor**
These bits set the trigger level for the voltage level monitor.

**Table 12-4. Setting the Trigger Level of Voltage Level Monitor**

| VLM[2:0] | Label | Description |
|---|---|---|
| 000 | VLM0 | Voltage Level Monitor disabled |
| 001 | VLM1L | Triggering generates a regular Power-On Reset (POR).<br>The VLM flag is not set |
| 010 | VLM1H | |
| 011 | VLM2 | Triggering sets the VLM Flag (VLMF) and generates a VLM interrupt, if enabled |
| 100 | VLM3 | |
| 101 | | Not allowed |
| 110 | | Not allowed |
| 111 | | Not allowed |

### 12.6.3. Reset Flag Register

**Name:** RSTFLR
**Offset:** 0x3B
**Reset:** N/A
**Property:** -

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

**Bit 3 – WDRF: Watchdog Reset Flag**
This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

**Bit 1 – EXTRF: External Reset Flag**
This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

**Bit 0 – PORF: Power-on Reset Flag**
This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag. To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

# 13. Interrupts

## 13.1. Overview

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*.

**Related Links**

Reset and Interrupt Handling on page 21

## 13.2. Interrupt Vectors

Interrupt vectors are described in the table below.

**Table 13-1. Reset and Interrupt Vectors**

| Vector No. | Program Address | Source | Interrupt Definition |
|---|---|---|---|
| 1 | 0x000 | RESET | External Pin, Power-on Reset, VLM Reset and Watchdog Reset |
| 2 | 0x001 | INT0 | External Interrupt Request 0 |
| 3 | 0x002 | PCINT0 | Pin Change Interrupt Request 0 |
| 4 | 0x003 | PCINT1 | Pin Change Interrupt Request 1 |
| 5 | 0x004 | TIM0_CAPT | Timer/Counter0 Capture |
| 6 | 0x005 | TIM0_OVF | Timer/Counter0 Overflow |
| 7 | 0x006 | TIM0_COMPA | Timer/Counter0 Compare Match A |
| 8 | 0x007 | TIM0_COMPB | Timer/Counter0 Compare Match B |
| 9 | 0x008 | ANA_COMP | Analog Comparator |
| 10 | 0x009 | WDT | Watchdog Time-out Interrupt |
| 11 | 0x00A | VLM | $V_{CC}$ Voltage Level Monitor |
| 12 | 0x00B | ADC | ADC Conversion Complete |
| 13 | 0x00C | USART0_RXS | USART0 Rx Start |
| 14 | 0x00D | USART0_RXC | USART0 Rx Complete |
| 15 | 0x00E | USART0_DRE | USART0 Data Register Empty |
| 16 | 0x00F | USART0_TXC | USART0 Tx Complete |

In case the program never enables an interrupt source, the Interrupt Vectors will not be used and, consequently, regular program code can be placed at these locations.

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

```
Address   Labels    Code                 Comments
0x000               rjmp RESET           ; Reset Handler
0x001               rjmp INT0            ; IRQ0 Handler
```

```
0x002              rjmp PCINT0            ; PCINT0 Handler
0x003              rjmp PCINT1            ; PCINT1 Handler
0x004              rjmp TIM0_CAPT         ; Timer/Counter0 Capture Handler
0x005              rjmp TIM0_OVF          ; Timer/Counter0 Overflow Handler
0x006              rjmp TIM0_COMPA        ; Timer/Counter0 Compare Match A Handler
0x007              rjmp TIM0_COMPB        ; Timer/Counter0 Compare Match B Handler
0x008              rjmp ANA_COMP          ; Analog Comparator Handler
0x009              rjmp WDT               ; Watchdog Timer Handler
0x00A              rjmp VLM               ; Voltage Level Monitor Handler
0x00B              rjmp ADC               ; ADC Conversion Handler
0x00C              rjmp USART0_RXS        ; USART0 Rx Start Handler
0x00D              rjmp USART0_RXC        ; USART0 Rx Complete Handler
0x00E              rjmp USART0_DRE        ; USART0 Data Register Empty Handler
0x00F              rjmp USART0_TXC        ; USART0 Tx Complete Handler

0x010     RESET:   ldi r16, high (RAMEND) ; Main program start
0x011              out SPH, r16           ; Set Stack Pointer
0x012              ldi r16, low(RAMEND)   ; to top of RAM
0x013              out SPL, r16
0x014              sei                    ; Enable interrupts
0x015              <instr>
...                ...
```

## 13.3. External Interrupts

The External Interrupts are triggered by the INT0 pins or any of the PCINT[11:0] pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT[11:0] pins are configured as outputs. This feature provides a way of generating a software interrupt. The pin change interrupt PCI0 will trigger if any enabled PCINT[11:0] pin toggles. The Pin Change Mask 0/1 Register (PCMSK 0/1) controls which pins contribute to the pin change interrupts. Pin change interrupts on PCINT[11:0] are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 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 INT0 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 INT0 requires the presence of an I/O clock, described in *Clock Systems and their Distribution* chapter.

**Related Links**

Clock System on page 30
EICRA on page 58
PCMSK0 on page 63
PCMSK1 on page 64

### 13.3.1. Low Level Interrupt

A low level interrupt on INT0 is detected asynchronously. This means that the interrupt source can be used for waking the part also from sleep modes other than Idle (the I/O clock is halted in all sleep modes except Idle).

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 as described in *Clock System*

If the low level on the interrupt pin is removed before the device has woken up then program execution will not be diverted to the interrupt service routine but continue from the instruction following the SLEEP command.

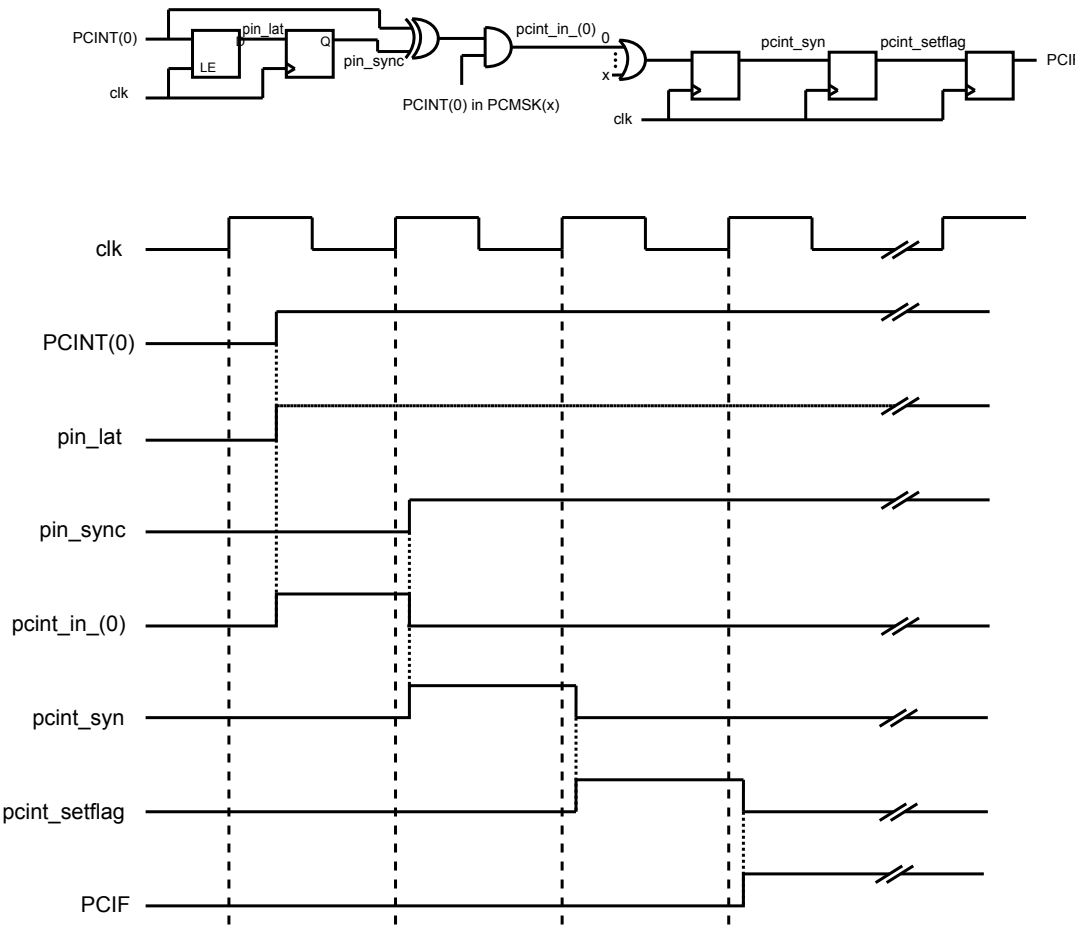
**Related Links**

Clock System on page 30

### 13.3.2. Pin Change Interrupt Timing

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

**Figure 13-1. Timing of pin change interrupts**




## 13.4. Register Description

### 13.4.1. External Interrupt Control Register A

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

**Name:** EICRA
**Offset:** 0x15
**Reset:** 0x00
**Property:** -

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

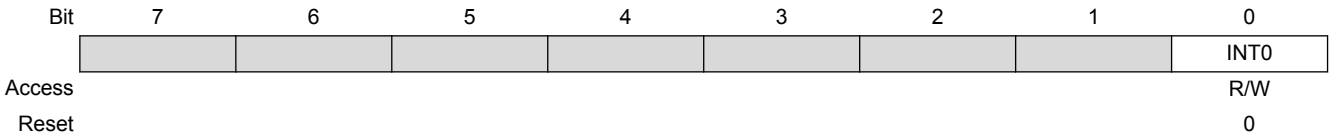
**Bits 1:0 – ISC0[1:0]: Interrupt Sense Control 0**

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. |

### 13.4.2. External Interrupt Mask Register

**Name:** EIMSK
**Offset:** 0x13
**Reset:** 0x00
**Property:** -



| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | INT0 |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – INT0: External Interrupt Request 0 Enable**
When the INT0 bit is set ('1') and the I-bit in the Status Register (SREG) is set ('1'), the external pin interrupt is enabled. The Interrupt Sense Control 0 bits in the External Interrupt Control Register A (EICRA.ISC0) 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.

### 13.4.3. External Interrupt Flag Register

**Name:** EIFR
**Offset:** 0x14
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | INTF0 |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – INTF0: External Interrupt Flag 0**
When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

### 13.4.4. Pin Change Interrupt Control Register

**Name:** PCICR
**Offset:** 0x12
**Reset:** 0x00
**Property:** -

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

**Bit 1 – PCIE1: Pin Change Interrupt Enable 1**
When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT[11:8] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT[11:8] pins are enabled individually by the PCMSK1 Register.

**Bit 0 – PCIE0: Pin Change Interrupt Enable 0**
When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), 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 PCMSK Register.

### 13.4.5. Pin Change Interrupt Flag Register

**Name:** PCIFR
**Offset:** 0x11
**Reset:** 0x00
**Property:** -

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

**Bit 1 – PCIF1: Pin Change Interrupt Flag 1**
When a logic change on any PCINT[11:8] pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

**Bit 0 – PCIF0: Pin Change Interrupt Flag 0**
When a logic change on any PCINT[7:0] pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

### 13.4.6. Pin Change Mask Register 0

**Name:** PCMSK0
**Offset:** 0x0F
**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 7:0 – PCINTn: Pin Change Enable Mask [n = 7:0]**
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.

### 13.4.7. Pin Change Mask Register 1

**Name:** PCMSK1
**Offset:** 0x10
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | PCINT11 | PCINT10 | PCINT9 | PCINT8 |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3 – PCINT8, PCINT9, PCINT10, PCINT11: Pin Change Enable Mask [11:8]**
Each PCINT[11:8]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[11:8] is set and the PCICR.PCIE1 is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[11:8] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

# 14. I/O-Ports

## 14.1. Overview

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability.

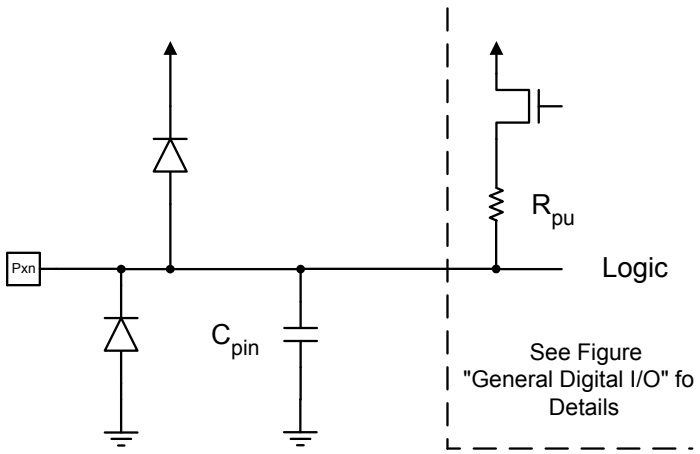
## 14.2. Features

- All AVR Ports Have True Read-Modify-Write Functionality.
- Flexible Pin configuration Through the Dedicated Registers.
- Each Output Buffer Has Symmetrical Drive Characteristics with both High Sink and Source Capability.

## 14.3. I/O Pin Equivalent Schematic

All I/O pins have protection diodes to both $V_{CC}$ and Ground as indicated in the following figure.

**Figure 14-1. I/O Pin Equivalent Schematic**




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

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.

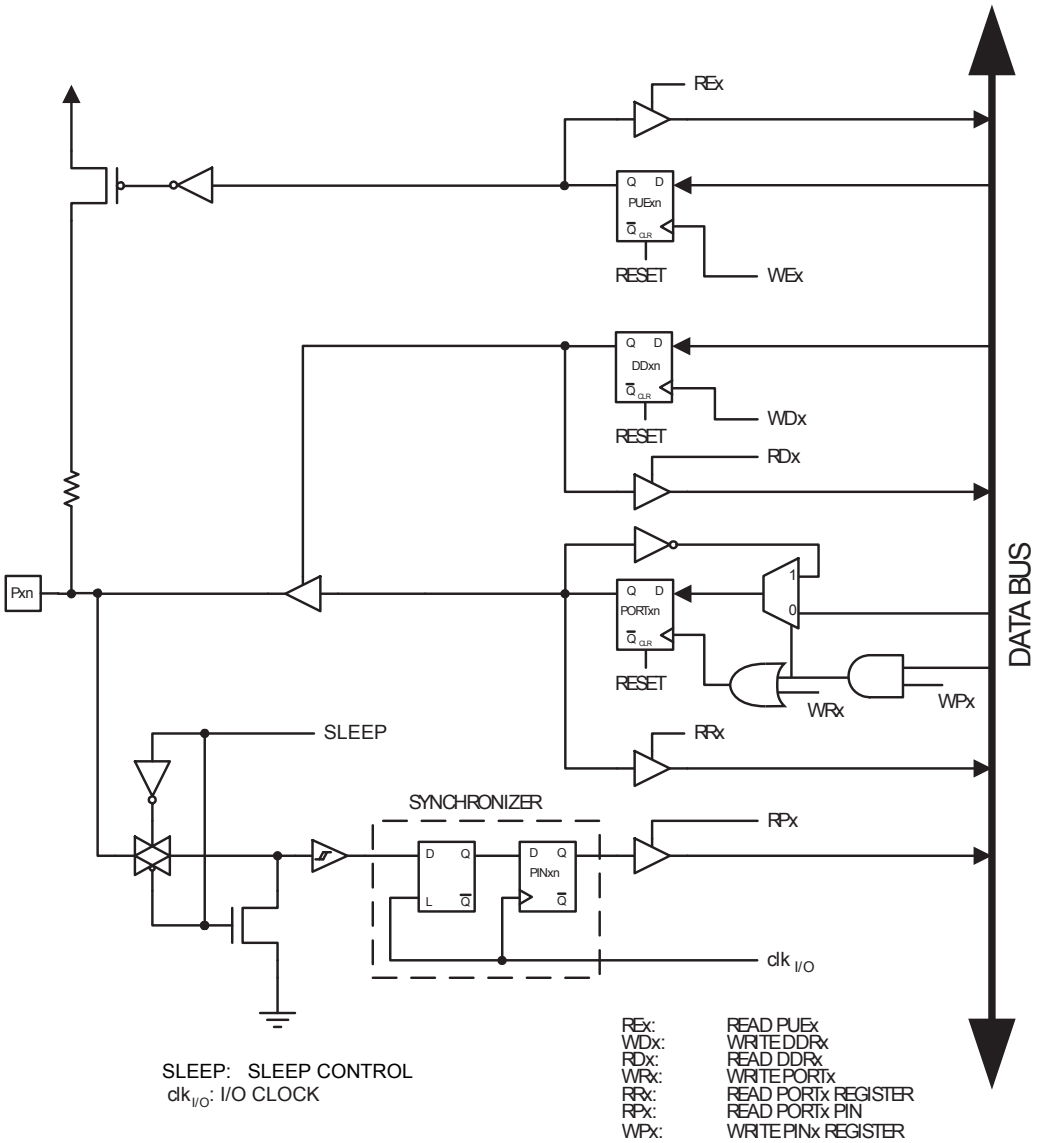
**Related Links**

Electrical Characteristics on page 210

## 14.4. Ports as General Digital I/O

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

**Figure 14-2. General Digital I/O**




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

### 14.4.1. Configuring the Pin

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

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

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

**Table 14-1. Port Pin Configurations**

| DDxn | PORTxn | PUExn | I/O | Pull-up | Comment |
|---|---|---|---|---|---|
| 0 | x | 0 | Input | No | Tri-state (hi-Z) |
| 0 | x | 1 | Input | Yes | Sources current if pulled low externally |
| 1 | 0 | 0 | Output | No | Output low (sink) |
| 1 | 0 | 1 | Output | Yes | NOT RECOMMENDED.<br>Output low (sink) and internal pull-up active. Sources current through the internal pull-up resistor and consumes power constantly |
| 1 | 1 | 0 | Output | No | Output high (source) |
| 1 | 1 | 1 | Output | Yes | Output high (source) and internal pull-up active |

Port pins are tri-stated when a reset condition becomes active, even when no clocks are running.

### 14.4.2. 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.

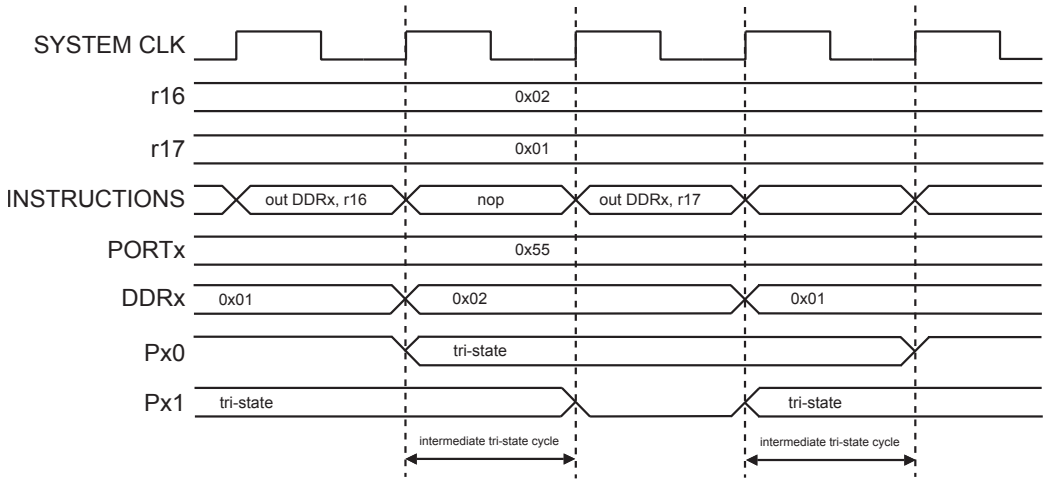
### 14.4.3. Break-Before-Make Switching

In Break-Before-Make mode, switching the DDRxn bit from input to output introduces an immediate tri-state period lasting one system clock cycle, as indicated in the figure below. For example, if the system clock is 4 MHz and the DDRxn is written to make an output, an immediate tri-state period of 250 ns is introduced before the value of PORTxn is seen on the port pin.

To avoid glitches it is recommended that the maximum DDRxn toggle frequency is two system clock cycles. The Break-Before-Make mode applies to the entire port and it is activated by the BBMx bit. For more details, see *PORTCR – Port Control Register*.

When switching the DDRxn bit from output to input no immediate tri-state period is introduced.

**Figure 14-3. Switching Between Input and Output in Break-Before-Make-Mode**



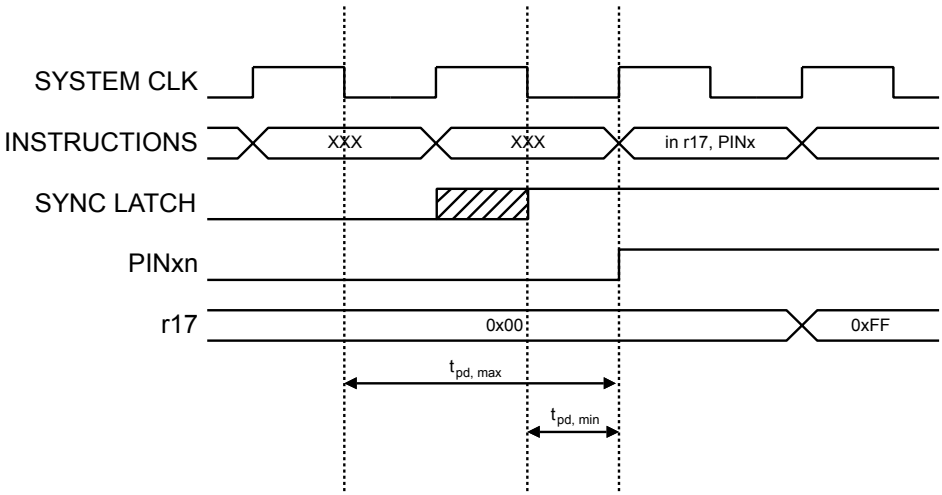

**Related Links**

PORTCR on page 88

### 14.4.4. 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 Figure 14-2, 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.

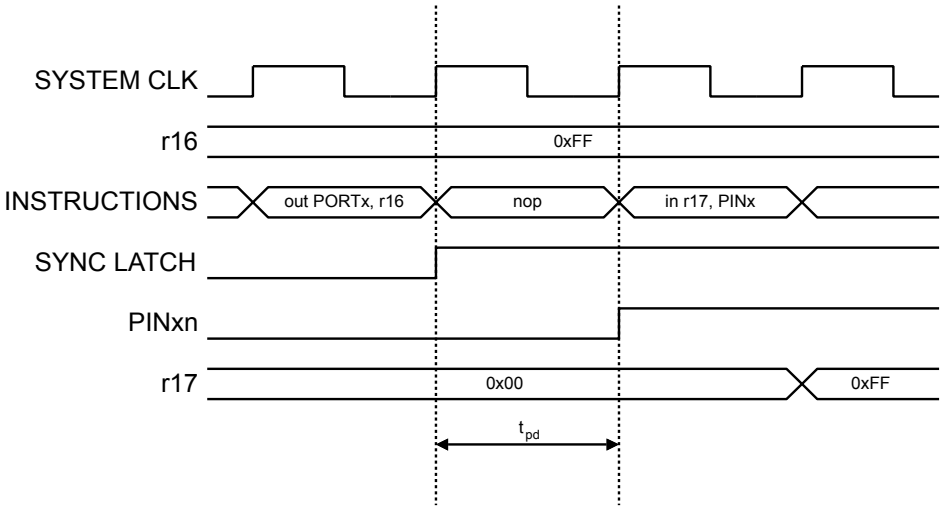
**Figure 14-4. Synchronization when Reading an Externally Applied Pin value**




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

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

**Figure 14-5. Synchronization when Reading a Software Assigned Pin Value**




### 14.4.5. Digital Input Enable and Sleep Modes

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

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

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

### 14.4.6. 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.

### 14.4.7. Program Example

The following code example shows how to set port B pin 0 high, pin 1 low, and define the port pins from 2 to 3 as input with a pull-up assigned to port pin 2. 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.

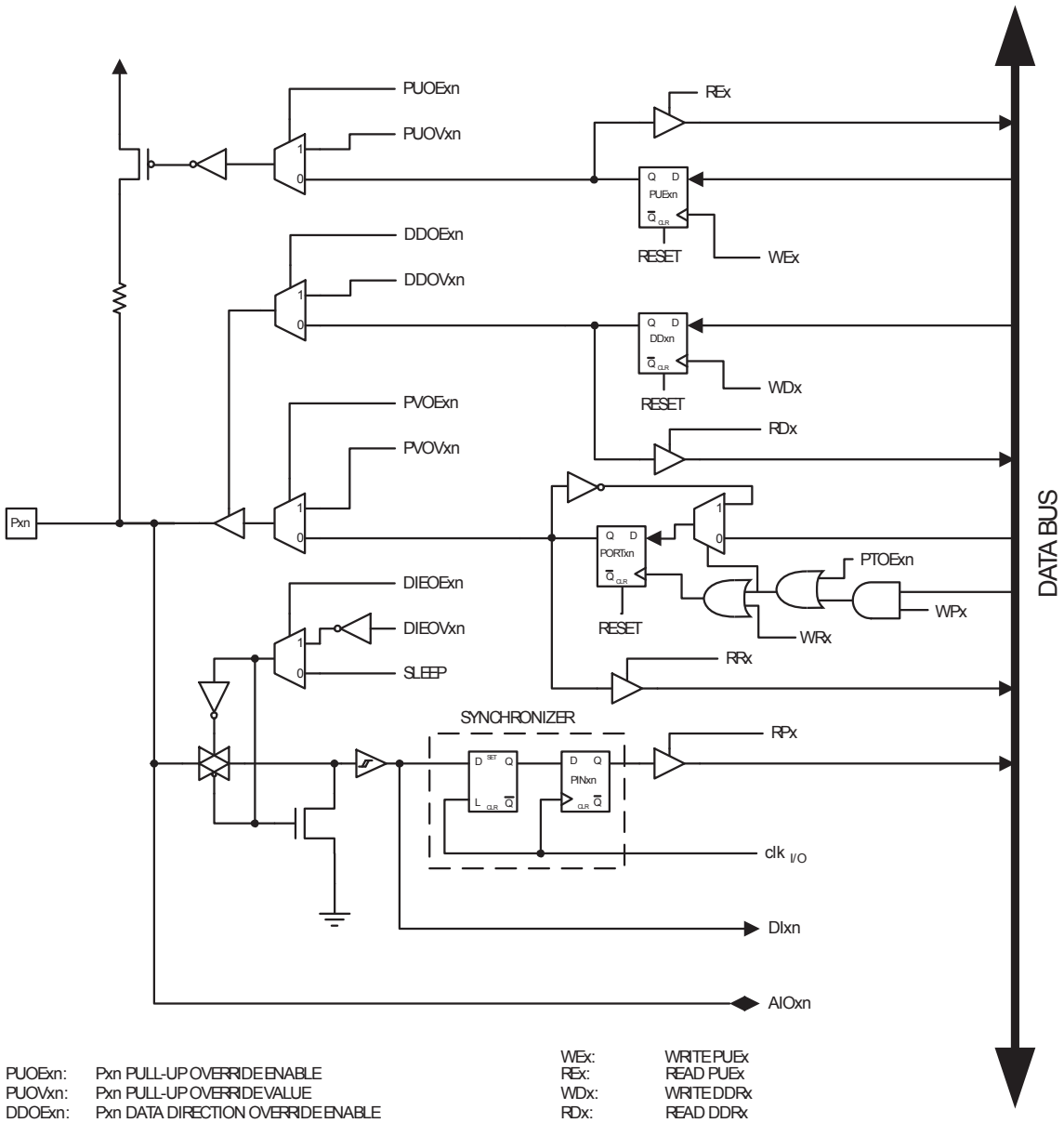
Assembly Code Example

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi r16,(1<<PUEB2)
ldi r17,(1<<PB0)
ldi r18,(1<<DDB1)|(1<<DDB0)
out PUEB,r16
out PORTB,r17
out DDRB,r18
; Insert nop for synchronization
nop
; Read port pins
in r16,PINB
...
```

### 14.4.8. 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 in the figure of Ports as General Digital I/O 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 14-6. Alternate Port Functions**




| | |
|---|---|
| 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 |
| PTOExn: | Pxn, PORT TOGGLE OVERRIDE ENABLE |

| | |
|---|---|
| WEx: | WRITE PUEx |
| REx: | READ PUEx |
| 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. WEx, WRx, WPx, WDx, REx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$ and SLEEP are common to all ports. All other signals are unique for each pin.

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

**Table 14-2. 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 PUExn = 0b1. |
| 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 PUExn Register bit. |
| 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. |
| PTOE | Port Toggle Override Enable | If PTOE is set, the PORTxn Register bit is inverted. |
| 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.

#### 14.4.8.1. Alternate Functions of Port A

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

**Table 14-3. Port A Pins Alternate Functions**

| Port Pin | Alternate Functions |
|---|---|
| PA[0] | ADC0: ADC Input Channel 0<br>AIN0: Analog Comparator, Positive Input<br>T0: Timer/Counter0 Clock Source (default location)<br>PCINT0: Pin Change Interrupt source 0<br>CLKI: External Clock<br>TPICLK: Serial Programming Clock |
| PA[1] | ADC1: ADC Input Channel 1<br>AIN1: Analog Comparator, Negative Input<br>OC0B: Timer/Counter0 Compare Match B Output (default location)<br>PCINT1: Pin Change Interrupt source 1<br>TPIDATA: Serial Programming Data |
| PA[2] | PCINT2: Pin Change Interrupt source 2<br>$\overline{\text{RESET}}$: Reset Pin |
| PA[3] | OC0A: Timer/Counter0 Compare Match A Output (alternative location)<br>PCINT3: Pin Change Interrupt source 3 |
| PA[4] | ICP0: Timer/Counter0 Input Capture Input (alternative location)<br>PCINT4: Pin Change Interrupt source 4 |
| PA[5] | ADC2: ADC Input Channel 2<br>OC0B: Timer/Counter0 Compare Match B Output (alternative location)<br>PCINT5: Pin Change Interrupt source 5 |
| PA[6] | ADC3: ADC Input Channel 3<br>PCINT6: Pin Change Interrupt source 6 |
| PA[7] | PCINT7: Pin Change Interrupt source 7 |

The alternate pin configuration is as follows:

- PA[0] – ADC0/AIN0/T0/PCINT0/CLKI/TPICLK
  - ADC0: Analog to Digital Converter, Channel 0.
  - AIN0: Analog Comparator Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - T0: Timer/Counter0 counter source.
  - PCINT0: Pin Change Interrupt source 0. The PA[0] pin can serve as an external interrupt source for pin change interrupt 0.
  - CLKI: External Clock.
  - TPICLK: Serial Programming Clock.
- PA[1] – ADC1/AIN1/OC0B/PCINT1/TPIDATA
  - ADC1: Analog to Digital Converter, Channel 1.
  - AIN1: Analog Comparator Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - OC0B: Output Compare Match B Output. The PA[1] pin can serve as an external output for the Timer/Counter0 Compare Match B. The PA[1] pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
  - PCINT1: Pin Change Interrupt source 1. The PA[1] pin can serve as an external interrupt source for pin change interrupt 0.
  - TPIDATA: Serial Programming Data.
- PA[2] – PCINT2/$\overline{\text{RESET}}$
  - PCINT2: Pin Change Interrupt source 2. The PA[2] pin can serve as an external interrupt source for pin change interrupt 0.
  - $\overline{\text{RESET}}$: Reset Pin.
- PA[3] – OC0A/PCINT3
  - OC0A: Output Compare Match A Output. The PA[3] pin can serve as an external output for the Timer/Counter0 Compare Match A. The pin has to be configured as an output (DDB0 set (one)) to serve this function. This is also the output pin for the PWM mode timer function.
  - PCINT3: Pin Change Interrupt source 3. The PA[3] pin can serve as an external interrupt source for pin change interrupt 0.
- PA[4] - ICP0/PCINT4
  - ICP0: Input Capture Pin. The PA[4] pin can act as an Input Capture pin for Timer/Counter0.
  - PCINT4: Pin Change Interrupt source 4. The PA4 pin can serve as an external interrupt source for pin change interrupt 0.
- PA[5] - ADC2/OC0B/PCINT5
  - ADC2: Analog to Digital Converter, Channel 2.
  - OC0B: Output Compare Match B Output: The PA1 pin can serve as an external output for the Timer/Counter0 Compare Match B. The PA[5] pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
  - PCINT5: Pin Change Interrupt source 5. The PA5 pin can serve as an external interrupt source for pin change interrupt 0.
- PA[6] - ADC3/PCINT6
  - ADC3: Analog to Digital Converter, Channel 3.

- PCINT6: Pin Change Interrupt source 6. The PA6 pin can serve as an external interrupt source for pin change interrupt 0.

- PA[7] - PCINT7
  - PCINT7: Pin Change Interrupt source 7. The PA7 pin can serve as an external interrupt source for pin change interrupt 0.

The following tables relate the alternate functions of Port B to the overriding signals shown in the figure of Alternate Port Functions.

**Table 14-4. Overriding Signals for Alternate Functions in PA[7:6]**

| Signal Name | PA7/PCINT7 | PA6/ADC3/PCINT6 |
|---|---|---|
| PUOE | 0 | 0 |
| PUOV | 0 | 0 |
| DDOE | 0 | 0 |
| DDOV | 0 | 0 |
| PVOE | 0 | 0 |
| PVOV | 0 | 0 |
| PTOE | 0 | 0 |
| DIEOE | (PCINT7 • PCIE0) | (PCINT6 • PCIE0) + ADC3D |
| DIEOV | PCINT7 • PCIE0 | PCINT6 • PCIE0 |
| DI | PCINT7 Input | PCINT6 input |
| AIO | - | ADC3 |

**Table 14-5. Overriding Signals for Alternate Functions in PA[5:4]**

| Signal Name | PA5/ADC2/OC0B/PCINT5 | PA4/ICP0/PCINT4 |
|---|---|---|
| PUOE | 0 | 0 |
| PUOV | 0 | 0 |
| DDOE | 0 | 0 |
| DDOV | 0 | 0 |
| PVOE | (OC0B Enable • REMAP) | 0 |
| PVOV | (OC0B • REMAP) | 0 |
| PTOE | 0 | 0 |
| DIEOE | (PCINT5 • PCIE0) + ADC2D | (PCINT4 • PCIE0) |
| DIEOV | PCINT5 • PCIE0 | (PCINT4 • PCIE0) |
| DI | PCINT5 Input | ICP0/PCINT4 Input |
| AIO | ADC2 | - |

**Table 14-6. Overriding Signals for Alternate Functions in PA[3:2]**

| Signal Name | PA3/OC0A/PCINT3 | PA4/ICP0/PCINT4 |
|---|---|---|
| PUOE | 0 | $\overline{\text{RSTDISBL}}$[1] |
| PUOV | 0 | 1 |
| DDOE | 0 | $\overline{\text{RSTDISBL}}$[1] |
| DDOV | 0 | 0 |
| PVOE | (OC0A Enable • REMAP) | 0 |
| PVOV | (OC0A • REMAP) | 0 |
| PTOE | 0 | 0 |
| DIEOE | (PCINT3 • PCIE0) | $\overline{\text{RSTDISBL}}$[1] + (PCINT2 • PCIE0) |
| DIEOV | PCINT3 • PCIE0 | RSTDISBL • PCINT2 • PCIE0 |
| DI | PCINT3 Input | PCINT2 input |
| AIO | - | - |

**Note:**

1. RSTDISBL is 1 when the configuration bit is “0” (Programmed).

**Table 14-7. Overriding Signals for Alternate Functions in PA[1:0]**

| Signal Name | PA1/ADC1/AIN1/OC0B/PCINT1 | PA0/ADC0/AIN0/CLKI/T0/PCINT0 |
|---|---|---|
| PUOE | 0 | EXT_CLOCK[1] |
| PUOV | 0 | 0 |
| DDOE | 0 | EXT_CLOCK[1] |
| DDOV | 0 | 0 |
| PVOE | (OC0B Enable • $\overline{\text{REMAP}}$) | EXT_CLOCK[1] |
| PVOV | (OC0B • $\overline{\text{REMAP}}$) | 0 |
| PTOE | 0 | 0 |
| DIEOE | (PCINT1 • PCIE0) + ADC1D | EXT_CLOCK[1] + (PCINT0 • PCIE0) + ADC0D |
| DIEOV | PCINT1 • PCIE0 | EXT_CLOCK[1]• $\overline{\text{PWR_DOWN}}$) + ($\overline{\text{EXT_CLOCK}}$[1]• PCINT0 • PCIE0) |
| DI | PCINT1 Input | CLKI/T0/PCINT0 Input |
| AIO | ADC1/Analog Comparator Negative Input | ADC0/Analog Comparator Positive Input |

**Note:**

1. EXT_CLOCK is 1 when external clock is selected as main clock.

#### 14.4.8.2. Alternate Functions of Port B

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

**Table 14-8. Port B Pins Alternate Functions**

| Port Pin | Alternate Functions |
|---|---|
| PB[0] | ADC4: ADC Input Channel 4<br>PCINT8: Pin Change Interrupt source 8 |
| PB[1] | ADC5: ADC Input Channel 5<br>OC0A: Timer/Counter0 Compare Match A Output (default location)<br>PCINT9: Pin Change Interrupt source 9<br>INT0: External Interrupt 0 Source<br>CLKO: System Clock Output |
| PB[2] | ADC6: ADC Input Channel 6<br>ICP0: Timer/Counter0 Input Capture Input (default location)<br>TxD0: USART Output<br>PCINT10: Pin Change Interrupt source 10 |
| PB[3] | ACO: AC Output<br>ADC7: ADC Input Channel 7<br>T0: Timer/Counter0 Clock Source (alternative location)<br>RxD0: USART Input<br>PCINT11: Pin Change Interrupt source 11 |

The alternate pin configuration is as follows:

- PB[0] – ADC4/PCINT8
  - ADC4: Analog to Digital Converter, Channel 4.
  - PCINT8: Pin Change Interrupt source 8. The PB[0] pin can serve as an external interrupt source for pin change interrupt 1.
- PB[1] – ADC5/OC0A/PCINT9/INT0/CLKO
  - ADC5: Analog to Digital Converter, Channel 5
  - OC0A: Output Compare Match output. The PB[1] pin can serve as an external output for the Timer/Counter0 Compare Match A. The PB[1] pin has to be configured as an output (DDB0 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.
  - PCINT9: Pin Change Interrupt source 9. The PB[1] pin can serve as an external interrupt source for pin change interrupt 1.
  - INT0: External Interrupt Request 0.

- CLKO: System Clock Output. The system clock can be output on pin PB[1]. The system clock will be output if CKOUT bit is programmed, regardless of the **PORTB2 and DDB2** settings.

- PB[2] – ADC6/ICP0/TxD0/PCINT10
  - ADC6: Analog to Digital Converter, Channel 6.
  - ICP0: Input Capture Pin. The PB[2] pin can act as an Input Capture pin for Timer/Counter0.
  - PCINT10: Pin Change Interrupt source 10. The PB[2] pin can serve as an external interrupt source for pin change interrupt 1.
  - TxD0: USART Output
- PB[3] – ACO/ADC7/T0/RXD0/PCINT3/RXD0
  - ACO: AC Output
  - ADC7: Analog to Digital Converter, Channel 7.
  - T0: Timer/Counter0 counter source.
  - PCINT11: Pin Change Interrupt source 11. The PB[3] pin can serve as an external interrupt source for pin change interrupt 1.
  - RXD0: USART input

The following tables relate the alternate functions of Port B to the overriding signals shown in the figure of Alternate Port Functions.

**Table 14-9. Overriding Signals for Alternate Functions in PB[3:2]**

| Signal Name | PB3/ADC7/ACO/RxD0/T0/PCINT11 | PB2/ADC6/TxD0/ICP0/PCINT10 |
|---|---|---|
| PUOE | ACOE | TxEN0 |
| PUOV | 0 | 0 |
| DDOE | RxEN0 + ($\overline{\text{RxEN0}}$ • ACOE) | TxEN0 |
| DDOV | ACOE | TxEN0 |
| PVOE | ACOE | TxEN0 |
| PVOV | ACO • ACOE | TxEN0• TXD0_OUT |
| PTOE | 0 | 0 |
| DIEOE | (PCINT11 • PCIE1) + ADC7D | (PCINT10 • PCIE1) + ADC6D |
| DIEOV | PCINT11• PCIE1 | PCINT10 • PCIE1 |
| DI | RxD0/T0/PCINT11 Input | ICP0/PCINT10 input |
| AIO | ADC7/ AC Output | ADC6 |

**Table 14-10. Overriding Signals for Alternate Functions in PB[1:0]**

| Signal Name | PB1/ADC5/INT0/XCK0/CLKO/OC0A/PCINT9 | PB0/ADC4/PCINT8 |
|---|---|---|
| PUOE | CKOUT(1) | 0 |
| PUOV | 0 | 0 |
| DDOE | CKOUT(1)+ (OC0A Enable • $\overline{\text{REMAP}}$) + XCK0_MASTER | 0 |

| Signal Name | PB1/ADC5/INT0/XCK0/CLKO/OC0A/PCINT9 | PB0/ADC4/PCINT8 |
|---|---|---|
| DDOV | CLKO + ($\overline{\text{CKOUT}}$ • OC0A Enable • $\overline{\text{REMAP}}$ • OC0A) + ($\overline{\text{CKOUT}}$ • ($\overline{\text{OC0A Enable}}$ + REMAP) • XCK0_MASTER • XCK0_OUT) | 0 |
| PVOE | CKOUT(1) | 0 |
| PVOV | (system clock) | 0 |
| PTOE | 0 | 0 |
| DIEOE | (PCINT9 • PCIE1) + ADC5D + INT0 | (PCINT8 • PCIE1) + ADC4D |
| DIEOV | (PCINT9 • PCIE1) + INT0 | (PCINT8 • PCIE1) |
| DI | INT0/PCINT1 Input | PCINT8 Input |
| AIO | ADC5 | ADC4 |

**Note:**

1. CKOUT is 1 when the configuration bit is “0” (Programmed).

## 14.5. Register Description

### 14.5.1. Port A Input Pins Address

**Name:** PINA
**Offset:** 0x00
**Reset:** N/A
**Property:**

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

**Bits 7:0 – PINAn: Port A Input Pins Address [n = 7:0]**

### 14.5.2. Port A Data Direction Register

**Name:** DDRA
**Offset:** 0x01
**Reset:** 0x00
**Property:**

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

**Bits 7:0 – DDRAn: Port A Input Pins Address [n = 7:0]**

### 14.5.3. Port A Data Register

**Name:** PORTA
**Offset:** 0x02
**Reset:** 0x00
**Property:**

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

**Bits 7:0 – PORTAn: Port A Data [n = 7:0]**

### 14.5.4. Port A Pull-up Enable Control Register

**Name:** PUEA
**Offset:** 0x03
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PUEA7 | PUEA6 | PUEA5 | PUEA4 | PUEA3 | PUEA2 | PUEA1 | PUEA0 |
| 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 – PUEAn: Port A Input Pins Address [n = 7:0]**

### 14.5.5. Port B Input Pins Address

**Name:** PINB
**Offset:** 0x04
**Reset:** N/A
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | PINB3 | PINB2 | PINB1 | PINB0 |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | x | x | x | x |

**Bits 3:0 – PINBn: Port B Input Pins Address [n = 3:0]**

### 14.5.6. Port B Data Direction Register

**Name:** DDRB
**Offset:** 0x05
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – DDRBn: Port B Input Pins Address [n = 3:0]**

### 14.5.7. Port B Data Register

**Name:** PORTB
**Offset:** 0x06
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | PORTB3 | PORTB2 | PORTB1 | PORTB0 |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – PORTBn: Port B Data [n = 3:0]**

### 14.5.8. Port B Pull-up Enable Control Register

**Name:** PUEB
**Offset:** 0x07
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | PUEB3 | PUEB2 | PUEB1 | PUEB0 |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – PUEBn: Port B Input Pins Address [n = 3:0]**

### 14.5.9. Port Control Register

**Name:** PORTCR
**Offset:** 0x16
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | BBMB | BBMA |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – BBMB: Break-Before-Make Mode Enable**
When this bit is set the Break-Before-Make mode is activated for the entire Port B. The intermediate tri-state cycle is then inserted when writing DDRxn to make an output.

**Bit 0 – BBMA: Break-Before-Make Mode Enable**
When this bit is set the Break-Before-Make mode is activated for the entire Port A. The intermediate tri-state cycle is then inserted when writing DDRxn to make an output.

# 15. USART - Universal Synchronous Asynchronous Receiver Transceiver

## 15.1. Overview

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

The USART can also be used in Master SPI mode. The Power Reduction USART bit in the Power Reduction Register (0.PRUSARTn) must be written to '0' in order to enable USARTn. USART 0 in 0.

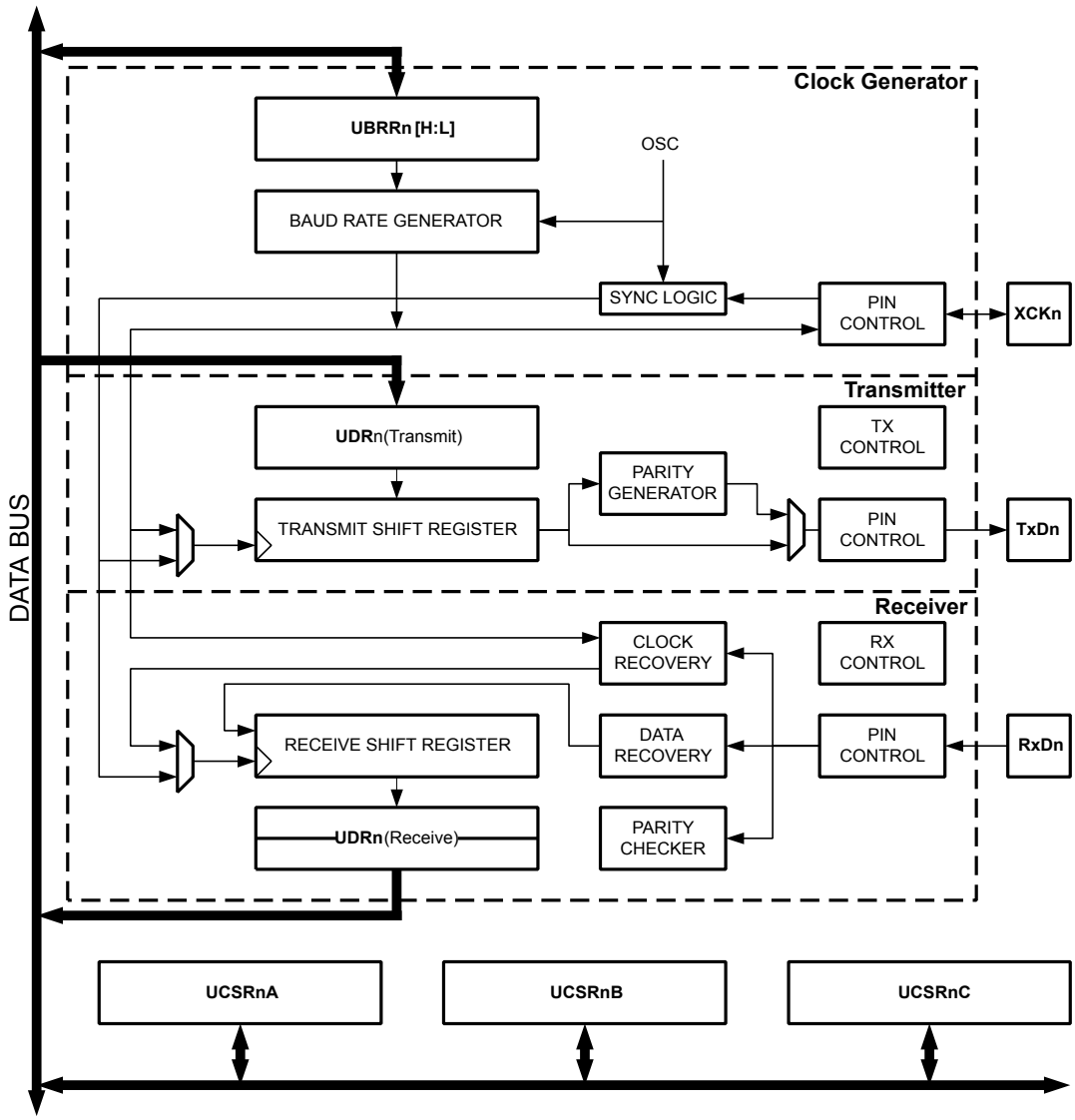
## 15.2. Features

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

## 15.3. Block Diagram

In the USART Block Diagram, the CPU accessible I/O Registers and I/O pins are shown in bold. The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter, and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCKn (Transfer Clock) pin is only used by synchronous transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator, and Control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a Parity Checker, Control logic, a Shift Register, and a two level receive buffer (UDRn). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun, and Parity Errors.

**Figure 15-1. USART Block Diagram**




**Note:** Refer to the *Pin Configurations* and the *I/O-Ports* description for USART pin placement.

**Related Links**

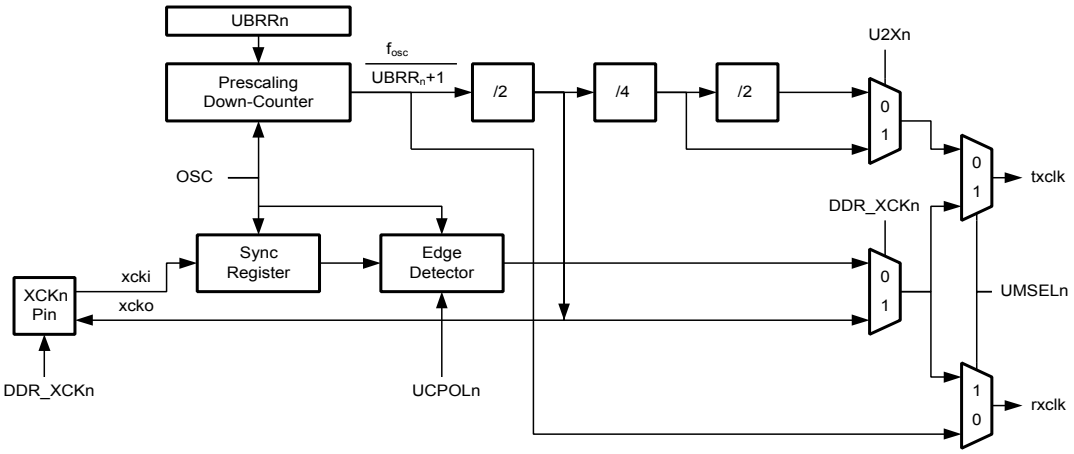
Pin Descriptions on page 12
I/O-Ports on page 65

## 15.4. Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal asynchronous, Double Speed asynchronous, Master synchronous and Slave synchronous mode. The USART Mode Select bit 0 in the USART Control and Status Register n C (UCSRnC.UMSELn0) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2X0 found in the UCSRnA Register. When using synchronous mode (UMSELn0=1), the Data Direction Register for the XCKn pin (DDR_XCKn) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCKn pin is only active when using synchronous mode.

Below is a block diagram of the clock generation logic.

**Figure 15-2. Clock Generation Logic, Block Diagram**




Signal description:

- txclk: Transmitter clock (internal signal).
- rxclk: Receiver base clock (internal signal).
- xcki: Input from XCKn pin (internal signal). Used for synchronous slave operation.
- xcko: Clock output to XCKn pin (internal signal). Used for synchronous master operation.
- $f_{osc}$: System clock frequency.

### 15.4.1. Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to the Clock Generation Logic block diagram in the previous section..

The USART Baud Rate Register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock ($f_{osc}$), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRnL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= $f_{osc}$/(UBRRn+1)). The Transmitter divides the baud rate generator clock output by 2, 8, or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8, or 16 states depending on mode set by the state of the UMSEL, U2X0 and DDR_XCK bits.

The table below contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

**Table 15-1. Equations for Calculating Baud Rate Register Setting**

| Operating Mode | Equation for Calculating Baud Rate(1) | Equation for Calculating UBRRn Value |
|---|---|---|
| Asynchronous Normal mode (U2X0 = 0) | $\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{16(\mathbf{UBRR}n+1)}$ | $\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{16\mathrm{BAUD}} - 1$ |
| Asynchronous Double Speed mode (U2X0 = 1) | $\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{8(\mathbf{UBRR}n+1)}$ | $\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{8\mathrm{BAUD}} - 1$ |
| Synchronous Master mode | $\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}n+1)}$ | $\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$ |

**Note:** 1. The baud rate is defined to be the transfer rate in bits per second (bps)

| | |
|---|---|
| **BAUD** | Baud rate (in bits per second, bps) |
| **$f_{OSC}$** | System oscillator clock frequency |
| **UBRRn** | Contents of the UBRRnH and UBRRnL Registers, (0-4095).<br>Some examples of UBRRn values for some system clock frequencies are found in Examples of Baud Rate Settings. |

### 15.4.2. Double Speed Operation (U2X0)

The transfer rate can be doubled by setting the U2X0 bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. However, in this case, the Receiver will only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used.

For the Transmitter, there are no downsides.

### 15.4.3. External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to the Clock Generation Logic block diagram in the previous section.

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

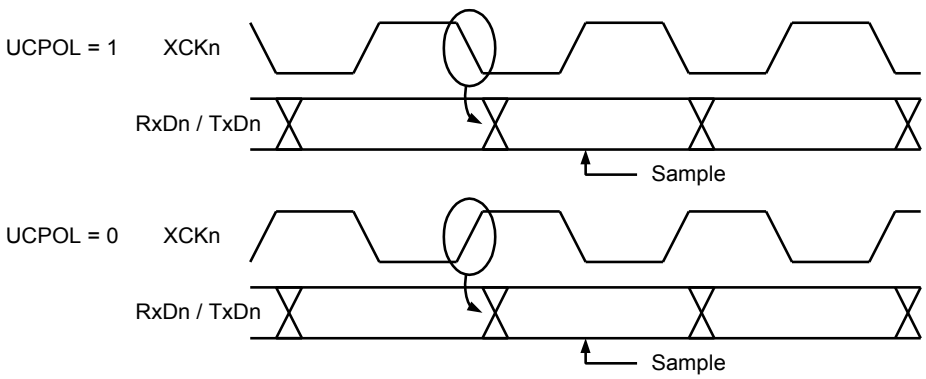
$$f_{XCKn} < \frac{f_{OSC}}{4}$$

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

### 15.4.4. Synchronous Clock Operation

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

**Figure 15-3. Synchronous Mode XCKn Timing**




The UCPOL bit UCRSC selects which XCKn clock edge is used for data sampling and which is used for data change. As the above timing diagram shows, when UCPOL is zero, the data will be changed at

rising XCKn edge and sampled at falling XCKn edge. If UCPOL is set, the data will be changed at falling XCKn edge and sampled at rising XCKn edge.

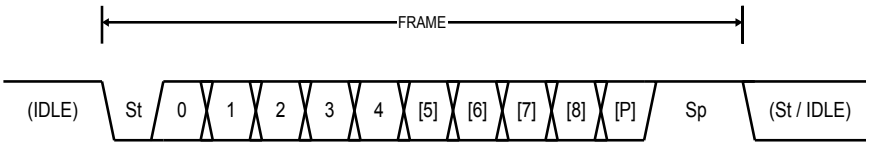
## 15.5. Frame Formats

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

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

A frame starts with the start bit, followed by the data bits (from five up to nine data bits in total): first the least significant data bit, then the next data bits ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the one or two stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. the figure below illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

**Figure 15-4. Frame Formats**




| | |
|---|---|
| St | Start bit, always low. |
| (n) | Data bits (0 to 8). |
| P | Parity bit. Can be odd or even. |
| Sp | Stop bit, always high. |
| IDLE | No transfers on the communication line (RxDn or TxDn). An IDLE line must be high. |

The frame format used by the USART is set by:

- Character Size bits (UCSRnC.UCSZn[2:0]) select the number of data bits in the frame.
- Parity Mode bits (UCSRnC.UPMn[1:0]) enable and set the type of parity bit.
- Stop Bit Select bit (UCSRnC.USBSn) select the number of stop bits. The Receiver ignores the second stop bit.

The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter. An FE (Frame Error) will only be detected in cases where the first stop bit is zero.

### 15.5.1. Parity Bit Calculation

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

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

$P_{even}$ Parity bit using even parity

$P_{odd}$ Parity bit using odd parity

$d_n$ Data bit n of the character

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

## 15.6. USART Initialization

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

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

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17, r16 Registers.

**Assembly Code Example**

```
USART_Init:
   ; Set baud rate to UBRR0
   out    UBRR0H, r17
   out    UBRR0L, r16
   ; Enable receiver and transmitter
   ldi    r16, (1<<RXEN0)|(1<<TXEN0)
   out    UCSR0B,r16
   ; Set frame format: 8data, 2stop bit
   ldi    r16, (1<<USBS0)|(3<<UCSZ00)
   out    UCSR0C,r16
   ret
```

**C Code Example**

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
   ...
   USART_Init(MYUBRR)
   ...
}
void USART_Init( unsigned int ubrr)
{
   /*Set baud rate */
   UBRR0H = (unsigned char)(ubrr>>8);
   UBRR0L = (unsigned char)ubrr;
   Enable receiver and transmitter */
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   /* Set frame format: 8data, 2stop bit */
   UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

More advanced initialization routines can be written to include frame format as parameters, disable interrupts, and so on. However, many applications use a fixed setting

of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

## 15.7. Data Transmission – The USART Transmitter

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

### 15.7.1. Sending Frames with 5 to 8 Data Bits

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDRn I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register, U2X0 bit or by XCKn depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the Data Register Empty (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR0 are ignored. The USART 0 has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R17.

**Assembly Code Example**

```
USART_Transmit:
   ; Wait for empty transmit buffer
   in      r17, UCSR0A
   sbrs    r17, UDRE
   rjmp    USART_Transmit
   ; Put data (r16) into buffer, sends the data
   out    UDR0,r16
   ret
```

**C Code Example**

```
void USART_Transmit( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSR0A & (1<<UDRE)) )
      ;
   /* Put data into buffer, sends the data */
   UDR0 = data;
}
```

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

### 15.7.2. Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZn = 7), the ninth bit must be written to the TXB8 bit in UCSRnB before the low byte of the character is written to UDRn.

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

The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

**Assembly Code Example**

```
USART_Transmit:
   ; Wait for empty transmit buffer
   in      r18, UCSR0A
   sbrs    r18, UDRE
   rjmp    USART_Transmit
   ; Copy 9th bit from r17 to TXB8
   cbi     UCSR0B,TXB8
   sbrc    r17,0
   sbi     UCSR0B,TXB8
   ; Put LSB data (r16) into buffer, sends the data
   out    UDR0,r16
   ret
```

**C Code Example**

```
void USART_Transmit( unsigned int data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSR0A & (1<<UDRE)) )
      ;
   /* Copy 9th bit to TXB8 */
   UCSR0B &= ~(1<<TXB8);
   if ( data & 0x0100 )
      UCSR0B |= (1<<TXB8);
   /* Put data into buffer, sends the data */
   UDR0 = data;
}
```

**Note:** These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB Register is used after initialization.

### 15.7.3. Transmitter Flags and Interrupts

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

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

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRnB is written to '1', the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDRn. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDRn in order to clear UDRE or disable the Data Register Empty interrupt - otherwise, a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC) Flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The TXC Flag bit is either automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a '1' to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like the RS-485

standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRnB is written to '1', the USART Transmit Complete Interrupt will be executed when the TXC Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC Flag, this is done automatically when the interrupt is executed.

### 15.7.4. Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UCSRnC.UPM[1]=1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

### 15.7.5. Disabling the Transmitter

When writing the TX Enable bit in the USART Control and Status Register n B (UCSRnB.TXEN) to zero, the disabling of the Transmitter will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn pin.

## 15.8. Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRnB Register to '1'. When the Receiver is enabled, the normal pin operation of the RxDn pin is overridden by the USART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCKn pin will be used as transfer clock.

### 15.8.1. Receiving Frames with 5 to 8 Data Bits

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

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR0 will be masked to zero. The USART 0 has to be initialized before the function can be used. For the assembly code, the received data will be stored in R16 after the code completes.

**Assembly Code Example**

```
USART_Receive:
   ; Wait for data to be received
   in    r17, UCSR0A
   sbrs  r17, RXC
   rjmp  USART_Receive
   ; Get and return received data from buffer
   in    r16, UDR0
   ret
```

**C Code Example**

```
unsigned char USART_Receive( void )
{
```

```
   /* Wait for data to be received */
   while ( !(UCSR0A & (1<<RXC)) )
      ;
   /* Get and return received data from buffer */
   return UDR0;
}
```

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

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

### 15.8.2. Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZn=7) the ninth bit must be read from the RXB8 bit in UCSRnB before reading the low bits from the UDRn. This rule applies to the FE, DOR and UPE Status Flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and UPE bits, which all are stored in the FIFO, will change.

The following code example shows a simple receive function for USART0 that handles both nine bit characters and the status bits. For the assembly code, the received data will be stored in R17:R16 after the code completes.

**Assembly Code Example**

```
USART_Receive:
   ; Wait for data to be received
   in     r16, UCSR0A
   sbrs   r16, RXC
   rjmp   USART_Receive
   ; Get status and 9th bit, then data from buffer
   in     r18, UCSR0A
   in     r17, UCSR0B
   in     r16, UDR0
   ; If error, return -1
   andi   r18,(1<<FE)|(1<<DOR)|(1<<UPE)
   breq   USART_ReceiveNoError
   ldi    r17, HIGH(-1)
   ldi    r16, LOW(-1)
USART_ReceiveNoError:
   ; Filter the 9th bit, then return
   lsr    r17
   andi   r17, 0x01
   ret
```

**C Code Example**

```
unsigned int USART_Receive( void )
{
   unsigned char status, resh, resl;
   /* Wait for data to be received */
   while ( !(UCSR0A & (1<<RXC)) )
      ;
   /* Get status and 9th bit, then data */
   /* from buffer */
   status = UCSR0A;
   resh = UCSR0B;
   resl = UDR0;
   /* If error, return -1 */
   if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
      return -1;
   /* Filter the 9th bit, then return */
```

```
    resh = (resh >> 1) & 0x01;
    return ((resh << 8) | resl);
}
```

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

### 15.8.3. Receive Compete Flag and Interrupt

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

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

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

### 15.8.4. Receiver Error Flags

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

The Frame Error (FE) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE Flag is zero when the stop bit was correctly read as '1', and the FE Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE Flag is not affected by the setting of the USBS bit in UCSRnC since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRnA.

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

The Parity Error (UPE) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPE bit will always read '0'. For compatibility with future devices, always set this bit to zero when writing to UCSRnA. For more details see Parity Bit Calculation and 'Parity Checker' below.

### 15.8.5. Parity Checker

The Parity Checker is active when the high USART Parity Mode bit 1 in the USART Control and Status Register n C (UCSRnC.UPM[1]) is written to '1'. The type of Parity Check to be performed (odd or even)

is selected by the UCSRnC.UPM[0] bit. When enabled, the Parity Checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The USART Parity Error Flag in the USART Control and Status Register n A (UCSRnA.UPE) can then be read by software to check if the frame had a Parity Error.

The UPEn bit is set if the next character that can be read from the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM[1] = 1). This bit is valid until the receive buffer (UDRn) is read.

### 15.8.6. Disabling the Receiver

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

### 15.8.7. Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDRn I/O location until the RXCn Flag is cleared.

The following code shows how to flush the receive buffer of USART0.

**Assembly Code Example**

```
USART_Flush:
   in      r16, UCSR0A
   sbrs    r16, RXC
   ret
   in      r16, UDR0
   rjmp    USART_Flush
```

**C Code Example**

```
void USART_Flush( void )
{
   unsigned char dummy;
   while ( UCSR0A & (1<<RXC) ) dummy = UDR0;
}
```

## 15.9. Asynchronous Data Reception

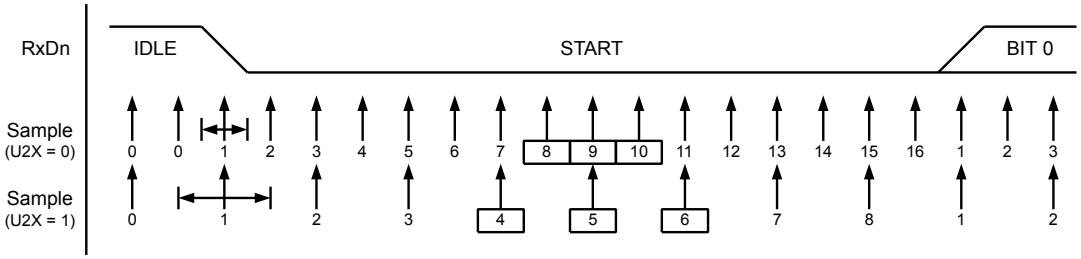
The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxDn pin. The data recovery logic samples and low pass filters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

### 15.9.1. Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. The figure below illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16-times the baud rate for Normal mode, and 8 times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when

using the Double Speed mode (UCSRnA.U2X0=1) of operation. Samples denoted '0' are samples taken while the RxDn line is idle (i.e., no communication activity).

**Figure 15-5. Start Bit Sampling**

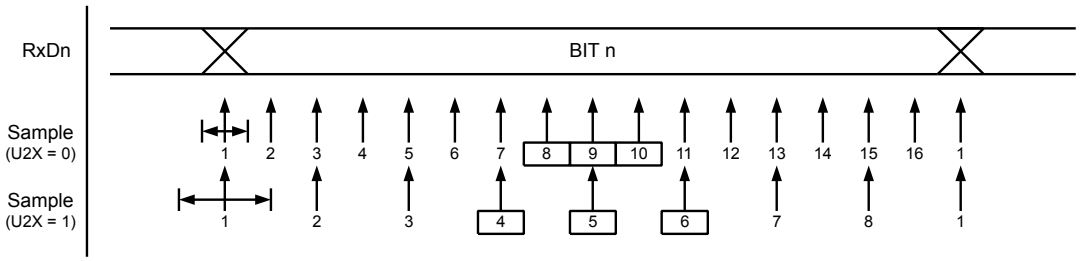



When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and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

### 15.9.2. Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. The figure below shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

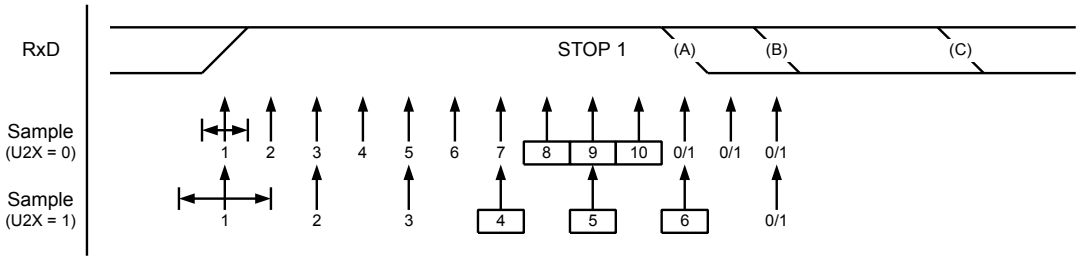
**Figure 15-6. Sampling of Data and Parity Bit**




The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit: If two or all three center samples (those marked by their sample number inside boxes) have high levels, the received bit is registered to be a logic '1'. If two or all three samples have low levels, the received bit is registered to be a logic '0'. This majority voting process acts as a low pass filter for the incoming signal on the RxDn pin. The recovery process is then repeated until a complete frame is received, including the first stop bit. The Receiver only uses the first stop bit of a frame.

The following figure shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

**Figure 15-7. Stop Bit Sampling and Next Start Bit Sampling**




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

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be taken at point marked (A) in the figure above. For Double Speed mode, the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

### 15.9.3. Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar base frequency (see recommendations below), the Receiver will not be able to synchronize the frames to the start bit.

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

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

- $D$: Sum of character size and parity size (D = 5 to 10 bit)
- $S$: Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
- $S_F$: First sample number used for majority voting. $S_F$ = 8 for normal speed and $S_F$ = 4 for Double Speed mode.
- $S_M$: Middle sample number used for majority voting. $S_M$ = 9 for normal speed and $S_M$ = 5 for Double Speed mode.
- $R_{slow}$ : is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. $R_{fast}$ is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

The following tables list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

**Table 15-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X0 = 0)**

| D # (Data+Parity Bit) | $R_{slow}$ [%] | $R_{fast}$ [%] | Max. Total Error [%] | Recommended Max. Receiver Error [%] |
|---|---|---|---|---|
| 5 | 93.20 | 106.67 | +6.67/-6.8 | ±3.0 |
| 6 | 94.12 | 105.79 | +5.79/-5.88 | ±2.5 |
| 7 | 94.81 | 105.11 | +5.11/-5.19 | ±2.0 |
| 8 | 95.36 | 104.58 | +4.58/-4.54 | ±2.0 |

| D # (Data+Parity Bit) | $R_{slow}$ [%] | $R_{fast}$ [%] | Max. Total Error [%] | Recommended Max. Receiver Error [%] |
|---|---|---|---|---|
| 9 | 95.81 | 104.14 | +4.14/-4.19 | ±1.5 |
| 10 | 96.17 | 103.78 | +3.78/-3.83 | ±1.5 |

**Table 15-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X0 = 1)**

| D # (Data+Parity Bit) | $R_{slow}$ [%] | $R_{fast}$ [%] | Max Total Error [%] | Recommended Max Receiver Error [%] |
|---|---|---|---|---|
| 5 | 94.12 | 105.66 | +5.66/-5.88 | ±2.5 |
| 6 | 94.92 | 104.92 | +4.92/-5.08 | ±2.0 |
| 7 | 95.52 | 104,35 | +4.35/-4.48 | ±1.5 |
| 8 | 96.00 | 103.90 | +3.90/-4.00 | ±1.5 |
| 9 | 96.39 | 103.53 | +3.53/-3.61 | ±1.5 |
| 10 | 96.70 | 103.23 | +3.23/-3.30 | ±1.0 |

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

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

### 15.9.4. Start Frame Detection

The USART start frame detector can wake up the MCU from Power-down and Standby sleep mode when it detects a start bit.

When a high-to-low transition is detected on RxDn, the internal 8MHz oscillator is powered up and the USART clock is enabled. After start-up the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the internal 8MHz oscillator start-up time. Start-up time of the internal 8MHz oscillator varies with supply voltage and temperature.

The USART start frame detection works both in asynchronous and synchronous modes. It is enabled by writing the Start Frame Detection Enable bit (SFDE). If the USART Start Interrupt Enable (RXSIE) bit is set, the USART Receive Start Interrupt is generated immediately when start is detected.

When using the feature without start interrupt, the start detection logic activates the internal 8MHz oscillator and the USART clock while the frame is being received, only. Other clocks remain stopped until the Receive Complete Interrupt wakes up the MCU.

The maximum baud rate in synchronous mode depends on the sleep mode the device is woken up from, as follows:

- Idle sleep mode: system clock frequency divided by four
- Standby or Power-down: 500kbps

The maximum baud rate in asynchronous mode depends on the sleep mode the device is woken up from, as follows:

- Idle sleep mode: the same as in active mode

**Table 15-4. Maximum Total Baudrate Error in Normal Speed Mode**

| Baudrate | Frame Size | | | | | |
|---|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 - 28.8kbps | +6.67<br>-5.88 | +5.79<br>-5.08 | +5.11<br>-4.48 | +4.58<br>-4.00 | +4.14<br>-3.61 | +3.78<br>-3.30 |
| 38.4kbps | +6.63<br>-5.88 | +5.75<br>-5.08 | +5.08<br>-4.48 | +4.55<br>-4.00 | +4.12<br>-3.61 | +3.76<br>-3.30 |
| 57.6kbps | +6.10<br>-5.88 | +5.30<br>-5.08 | +4.69<br>-4.48 | +4.20<br>-4.00 | +3.80<br>-3.61 | +3.47<br>-3.30 |
| 76.8kbps | +5.59<br>-5.88 | +4.85<br>-5.08 | +4.29<br>-4.48 | +3.85<br>-4.00 | +3.48<br>-3.61 | +3.18<br>-3.30 |
| 115.2kbps | +4.57<br>-5.88 | +3.97<br>-5.08 | +3.51<br>-4.48 | +3.15<br>-4.00 | +2.86<br>-3.61 | +2.61<br>-3.30 |

**Table 15-5. Maximum Total Baudrate Error in Double Speed Mode**

| Baudrate | Frame Size | | | | | |
|---|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 - 57.6kbps | +5.66<br>-4.00 | +4.92<br>-3.45 | +4.35<br>-3.03 | +3.90<br>-2.70 | +3.53<br>-2.44 | +3.23<br>-2.22 |
| 76.8kbps | +5.59<br>-4.00 | +4.85<br>-3.45 | +4.29<br>-3.03 | +3.85<br>-2.70 | +3.48<br>-2.44 | +3.18<br>-2.22 |
| 115.2kbps | +4.57<br>-4.00 | +3.97<br>-3.45 | +3.51<br>-3.03 | +3.15<br>-2.70 | +2.86<br>-2.44 | +2.61<br>-2.22 |

**Related Links**

UCSR0D on page 117

## 15.10. Multi-Processor Communication Mode

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

If the Receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the Receiver is set up for frames with 9 data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is '1', the frame contains an address. When the frame type bit is '0', the frame is a data frame.

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

### 15.10.1. Using MPCMn

For an MCU to act as a master MCU, it can use a 9-bit character frame format (UCSZ1=7). The ninth bit (TXB8) must be set when an address frame (TXB8=1) or cleared when a data frame (TXB=0) is being transmitted. The slave MCUs must in this case be set to use a 9-bit character frame format.

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

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

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

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

## 15.11. Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRRn settings as listed in the table below.

UBRRn values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see also section *Asynchronous Operational Range*). The error values are calculated using the following equation:

$$Error\left[\%\right] = \left(\frac{\mathrm{BaudRate}_{\mathrm{Closest\ Match}}}{\mathrm{BaudRate}} - 1\right)^2 \ 100\,\%$$

**Table 15-6. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies**

| Baud Rate [bps] | $f_{osc}$ = 1.0000MHz | | | | $f_{osc}$ = 1.8432MHz | | | | $f_{osc}$ = 2.0000MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | |
| | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error |
| 2400 | 25 | 0.2% | 51 | 0.2% | 47 | 0.0% | 95 | 0.0% | 51 | 0.2% | 103 | 0.2% |
| 4800 | 12 | 0.2% | 25 | 0.2% | 23 | 0.0% | 47 | 0.0% | 25 | 0.2% | 51 | 0.2% |
| 9600 | 6 | -7.0% | 12 | 0.2% | 11 | 0.0% | 23 | 0.0% | 12 | 0.2% | 25 | 0.2% |
| 14.4k | 3 | 8.5% | 8 | -3.5% | 7 | 0.0% | 15 | 0.0% | 8 | -3.5% | 16 | 2.1% |
| 19.2k | 2 | 8.5% | 6 | -7.0% | 5 | 0.0% | 11 | 0.0% | 6 | -7.0% | 12 | 0.2% |
| 28.8k | 1 | 8.5% | 3 | 8.5% | 3 | 0.0% | 7 | 0.0% | 3 | 8.5% | 8 | -3.5% |
| 38.4k | 1 | -18.6% | 2 | 8.5% | 2 | 0.0% | 5 | 0.0% | 2 | 8.5% | 6 | -7.0% |
| 57.6k | 0 | 8.5% | 1 | 8.5% | 1 | 0.0% | 3 | 0.0% | 1 | 8.5% | 3 | 8.5% |
| 76.8k | – | – | 1 | -18.6% | 1 | -25.0% | 2 | 0.0% | 1 | -18.6% | 2 | 8.5% |
| 115.2k | – | – | 0 | 8.5% | 0 | 0.0% | 1 | 0.0% | 0 | 8.5% | 1 | 8.5% |
| 230.4k | – | – | – | – | – | – | 0 | 0.0% | – | – | – | – |
| 250k | – | – | – | – | – | – | – | – | – | – | 0 | 0.0% |
| Max.(1) | 62.5kbps | | 125kbps | | 115.2kbps | | 230.4kbps | | 125kbps | | 250kbps | |

Note: 1. UBRRn = 0, Error = 0.0%

**Table 15-7. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies**

| Baud Rate [bps] | $f_{osc}$ = 3.6864MHz | | | | $f_{osc}$ = 4.0000MHz | | | | $f_{osc}$ = 7.3728MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | |
| | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error |
| 2400 | 95 | 0.0% | 191 | 0.0% | 103 | 0.2% | 207 | 0.2% | 191 | 0.0% | 383 | 0.0% |
| 4800 | 47 | 0.0% | 95 | 0.0% | 51 | 0.2% | 103 | 0.2% | 95 | 0.0% | 191 | 0.0% |
| 9600 | 23 | 0.0% | 47 | 0.0% | 25 | 0.2% | 51 | 0.2% | 47 | 0.0% | 95 | 0.0% |
| 14.4k | 15 | 0.0% | 31 | 0.0% | 16 | 2.1% | 34 | -0.8% | 31 | 0.0% | 63 | 0.0% |
| 19.2k | 11 | 0.0% | 23 | 0.0% | 12 | 0.2% | 25 | 0.2% | 23 | 0.0% | 47 | 0.0% |
| 28.8k | 7 | 0.0% | 15 | 0.0% | 8 | -3.5% | 16 | 2.1% | 15 | 0.0% | 31 | 0.0% |
| 38.4k | 5 | 0.0% | 11 | 0.0% | 6 | -7.0% | 12 | 0.2% | 11 | 0.0% | 23 | 0.0% |
| 57.6k | 3 | 0.0% | 7 | 0.0% | 3 | 8.5% | 8 | -3.5% | 7 | 0.0% | 15 | 0.0% |
| 76.8k | 2 | 0.0% | 5 | 0.0% | 2 | 8.5% | 6 | -7.0% | 5 | 0.0% | 11 | 0.0% |
| 115.2k | 1 | 0.0% | 3 | 0.0% | 1 | 8.5% | 3 | 8.5% | 3 | 0.0% | 7 | 0.0% |
| 230.4k | 0 | 0.0% | 1 | 0.0% | 0 | 8.5% | 1 | 8.5% | 1 | 0.0% | 3 | 0.0% |

| Baud Rate [bps] | $f_{osc}$ = 3.6864MHz | | | | $f_{osc}$ = 4.0000MHz | | | | $f_{osc}$ = 7.3728MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | |
| | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error |
| 250k | 0 | -7.8% | 1 | -7.8% | 0 | 0.0% | 1 | 0.0% | 1 | -7.8% | 3 | -7.8% |
| 0.5M | – | – | 0 | -7.8% | – | – | 0 | 0.0% | 0 | -7.8% | 1 | -7.8% |
| 1M | – | – | – | – | – | – | – | – | – | – | 0 | -7.8% |
| Max.(1) | 230.4kbps | | 460.8kbps | | 250kbps | | 0.5Mbps | | 460.8kbps | | 921.6kbps | |

(1) UBRRn = 0, Error = 0.0%

**Table 15-8. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies**

| Baud Rate [bps] | $f_{osc}$ = 8.0000MHz | | | | $f_{osc}$ = 11.0592MHz | | | | $f_{osc}$ = 14.7456MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | |
| | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error |
| 2400 | 207 | 0.2% | 416 | -0.1% | 287 | 0.0% | 575 | 0.0% | 383 | 0.0% | 767 | 0.0% |
| 4800 | 103 | 0.2% | 207 | 0.2% | 143 | 0.0% | 287 | 0.0% | 191 | 0.0% | 383 | 0.0% |
| 9600 | 51 | 0.2% | 103 | 0.2% | 71 | 0.0% | 143 | 0.0% | 95 | 0.0% | 191 | 0.0% |
| 14.4k | 34 | -0.8% | 68 | 0.6% | 47 | 0.0% | 95 | 0.0% | 63 | 0.0% | 127 | 0.0% |
| 19.2k | 25 | 0.2% | 51 | 0.2% | 35 | 0.0% | 71 | 0.0% | 47 | 0.0% | 95 | 0.0% |
| 28.8k | 16 | 2.1% | 34 | -0.8% | 23 | 0.0% | 47 | 0.0% | 31 | 0.0% | 63 | 0.0% |
| 38.4k | 12 | 0.2% | 25 | 0.2% | 17 | 0.0% | 35 | 0.0% | 23 | 0.0% | 47 | 0.0% |
| 57.6k | 8 | -3.5% | 16 | 2.1% | 11 | 0.0% | 23 | 0.0% | 15 | 0.0% | 31 | 0.0% |
| 76.8k | 6 | -7.0% | 12 | 0.2% | 8 | 0.0% | 17 | 0.0% | 11 | 0.0% | 23 | 0.0% |
| 115.2k | 3 | 8.5% | 8 | -3.5% | 5 | 0.0% | 11 | 0.0% | 7 | 0.0% | 15 | 0.0% |
| 230.4k | 1 | 8.5% | 3 | 8.5% | 2 | 0.0% | 5 | 0.0% | 3 | 0.0% | 7 | 0.0% |
| 250k | 1 | 0.0% | 3 | 0.0% | 2 | -7.8% | 5 | -7.8% | 3 | -7.8% | 6 | 5.3% |
| 0.5M | 0 | 0.0% | 1 | 0.0% | – | – | 2 | -7.8% | 1 | -7.8% | 3 | -7.8% |
| 1M | – | – | 0 | 0.0% | – | – | – | – | 0 | -7.8% | 1 | -7.8% |
| Max.(1) | 0.5Mbps | | 1Mbps | | 691.2kbps | | 1.3824Mbps | | 921.6kbps | | 1.8432Mbps | |

(1) UBRRn = 0, Error = 0.0%

**Table 15-9. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies**

| Baud Rate [bps] | $f_{osc}$ = 16.0000MHz | | | | $f_{osc}$ = 18.4320MHz | | | | $f_{osc}$ = 20.0000MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | | U2X0 = 0 | | U2X0 = 1 | |
| | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error |
| 2400 | 416 | -0.1% | 832 | 0.0% | 479 | 0.0% | 959 | 0.0% | 520 | 0.0% | 1041 | 0.0% |
| 4800 | 207 | 0.2% | 416 | -0.1% | 239 | 0.0% | 479 | 0.0% | 259 | 0.2% | 520 | 0.0% |
| 9600 | 103 | 0.2% | 207 | 0.2% | 119 | 0.0% | 239 | 0.0% | 129 | 0.2% | 259 | 0.2% |
| 14.4k | 68 | 0.6% | 138 | -0.1% | 79 | 0.0% | 159 | 0.0% | 86 | -0.2% | 173 | -0.2% |
| 19.2k | 51 | 0.2% | 103 | 0.2% | 59 | 0.0% | 119 | 0.0% | 64 | 0.2% | 129 | 0.2% |
| 28.8k | 34 | -0.8% | 68 | 0.6% | 39 | 0.0% | 79 | 0.0% | 42 | 0.9% | 86 | -0.2% |
| 38.4k | 25 | 0.2% | 51 | 0.2% | 29 | 0.0% | 59 | 0.0% | 32 | -1.4% | 64 | 0.2% |
| 57.6k | 16 | 2.1% | 34 | -0.8% | 19 | 0.0% | 39 | 0.0% | 21 | -1.4% | 42 | 0.9% |
| 76.8k | 12 | 0.2% | 25 | 0.2% | 14 | 0.0% | 29 | 0.0% | 15 | 1.7% | 32 | -1.4% |
| 115.2k | 8 | -3.5% | 16 | 2.1% | 9 | 0.0% | 19 | 0.0% | 10 | -1.4% | 21 | -1.4% |
| 230.4k | 3 | 8.5% | 8 | -3.5% | 4 | 0.0% | 9 | 0.0% | 4 | 8.5% | 10 | -1.4% |
| 250k | 3 | 0.0% | 7 | 0.0% | 4 | -7.8% | 8 | 2.4% | 4 | 0.0% | 9 | 0.0% |
| 0.5M | 1 | 0.0% | 3 | 0.0% | – | – | 4 | -7.8% | – | – | 4 | 0.0% |
| 1M | 0 | 0.0% | 1 | 0.0% | – | – | – | – | – | – | – | – |
| Max.(1) | 1Mbps | | 2Mbps | | 1.152Mbps | | 2.304Mbps | | 1.25Mbps | | 2.5Mbps | |

(1) UBRRn = 0, Error = 0.0%

## 15.12. Register Description

All of USART registers are NOT accessible using SBI and CBI instructions.

### 15.12.1. USART I/O Data Register 0

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR0. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR0 Register location. Reading the UDR0 Register location will return the contents of the Receive Data Buffer Register (RXB).

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

The transmit buffer can only be written when the UDRE0 Flag in the UCSR0A Register is set. Data written to UDR0 when the UDRE0 Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD0 pin.

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

**Name:** UDR0
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXB / RXB[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TXB / RXB[7:0]: USART Transmit / Receive Data Buffer**

### 15.12.2. USART Control and Status Register 0 A

**Name:** UCSR0A
**Offset:** 0x0E
**Reset:** 0x20
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXC0 | TXC0 | UDRE0 | FE0 | DOR0 | UPE0 | U2X0 | MPCM0 |
| Access | R | R/W | R | R | R | R | R/W | R/W |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – RXC0: USART Receive Complete**
This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

**Bit 6 – TXC0: USART Transmit Complete**
This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

**Bit 5 – UDRE0: USART Data Register Empty**
The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

**Bit 4 – FE0: Frame Error**
This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR0) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 3 – DOR0: Data OverRun**
The Data OverRun (DORn) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), a new character is waiting in the Receive Shift Register, and a new start bit is detected.

If this bit is set, one or more serial frames were lost between the last frame read from UDRn, and the next frame read from UDRn. For compatibility with future devices, always write this bit to zero when writing to UCSRnA. This bit is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 2 – UPE0: USART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM0 = 1). This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 1 – U2X0: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

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

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 0 – MPCM0: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCMn bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM0 setting.

This bit is reserved in Master SPI Mode (MSPIM).

### 15.12.3. USART Control and Status Register 0 B

**Name:** UCSR0B
**Offset:** 0x0D
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCIE0 | TXCIE0 | UDRIE0 | RXEN0 | TXEN0 | UCSZ02 | RXB80 | TXB80 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – RXCIE0: RX Complete Interrupt Enable 0**
Writing this bit to one enables interrupt on the RXC0 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

**Bit 6 – TXCIE0: TX Complete Interrupt Enable 0**
Writing this bit to one enables interrupt on the TXC0 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

**Bit 5 – UDRIE0: USART Data Register Empty Interrupt Enable 0**
Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

**Bit 4 – RXEN0: Receiver Enable 0**
Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxDn pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE0, DOR0, and UPE0 Flags.

**Bit 3 – TXEN0: Transmitter Enable 0**
Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD0 pin when enabled. The disabling of the Transmitter (writing TXEN0 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 port.

**Bit 2 – UCSZ02: Character Size 0**
The UCSZ02 bits combined with the UCSZ0[1:0] bit in UCSR0C sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 1 – RXB80: Receive Data Bit 8 0**
RXB80 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR0.

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 0 – TXB80: Transmit Data Bit 8 0**

TXB80 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR0.

This bit is reserved in Master SPI Mode (MSPIM).

### 15.12.4. USART Control and Status Register 0 C

**Name:** UCSR0C
**Offset:** 0x0C
**Reset:** 0x06
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | UMSEL01 | UMSEL00 | UPM01 | UPM00 | USBS0 | UCSZ01 / UDORD0 | UCSZ00 / UCPHA0 | UCPOL0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

**Bits 7:6 – UMSEL0n: USART Mode Select 0 n [n = 1:0]**
These bits select the mode of operation of the USART0

**Table 15-10. USART Mode Selection**

| UMSEL0[1:0] | Mode |
|---|---|
| 00 | Asynchronous USART |
| 01 | Synchronous USART |
| 10 | Reserved |
| 11 | Master SPI (MSPIM)(1) |

**Note:**
1. The UDORD0, UCPHA0, and UCPOL0 can be set in the same write operation where the MSPIM is enabled.

**Bits 5:4 – UPM0n: USART Parity Mode 0 n [n = 1:0]**
These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the UPE0 Flag in UCSR0A will be set.

**Table 15-11. USART Mode Selection**

| UPM0[1:0] | ParityMode |
|---|---|
| 00 | Disabled |
| 01 | Reserved |
| 10 | Enabled, Even Parity |
| 11 | Enabled, Odd Parity |

These bits are reserved in Master SPI Mode (MSPIM).

**Bit 3 – USBS0: USART Stop Bit Select 0**
This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

**Table 15-12. Stop Bit Settings**

| USBS0 | Stop Bit(s) |
|---|---|
| 0 | 1-bit |
| 1 | 2-bit |

This bit is reserved in Master SPI Mode (MSPIM).

**Bit 2 – UCSZ01 / UDORD0: USART Character Size / Data Order**
**UCSZ0[1:0]: USART Modes:** The UCSZ0[1:0] bits combined with the UCSZ02 bit in UCSR0B sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

**Table 15-13. Character Size Settings**

| UCSZ0[2:0] | Character Size |
|---|---|
| 000 | 5-bit |
| 001 | 6-bit |
| 010 | 7-bit |
| 011 | 8-bit |
| 100 | Reserved |
| 101 | Reserved |
| 110 | Reserved |
| 111 | 9-bit |

**UDPRD0: Master SPI Mode:** When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to the *USART in SPI Mode - Frame Formats* for details.

**Bit 1 – UCSZ00 / UCPHA0: USART Character Size / Clock Phase**
**UCSZ00: USART Modes:** Refer to UCSZ01.

**UCPHA0: Master SPI Mode:** The UCPHA0 bit setting determine if data is sampled on the leasing edge (first) or tailing (last) edge of XCK0. Refer to the *SPI Data Modes and Timing* for details.

**Bit 0 – UCPOL0: Clock Polarity 0**
**USART0 Modes:** This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL0 bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK0).

**Table 15-14. USART Clock Polarity Settings**

| UCPOL0 | Transmitted Data Changed (Output of TxD0 Pin) | Received Data Sampled (Input on RxD0 Pin) |
|---|---|---|
| 0 | Rising XCK0 Edge | Falling XCK0 Edge |
| 1 | Falling XCK0 Edge | Rising XCK0 Edge |

**Master SPI Mode:** The UCPOL0 bit sets the polarity of the XCK0 clock. The combination of the UCPOL0 and UCPHA0 bit settings determine the timing of the data transfer. Refer to the *SPI Data Modes and Timing* for details.

### 15.12.5. USART Control and Status Register 0 D

This register is not used in Master SPI Mode (`UMSEL0[1:0] = 11`)

**Name:** UCSR0D
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXSIE | RXS | SFDE | | | | | |
| Access | R/W | R/W | R/W | | | | | |
| Reset | 0 | 0 | 0 | | | | | |

**Bit 7 – RXSIE: USART RX Start Interrupt Enable**
Writing this bit to one enables the interrupt on the RXS flag. In sleep modes this bit enables start frame detector that can wake up the MCU when a start condition is detected on the RxD line. The USART RX Start Interrupt is generated only, if the RXSIE bit, the Global Interrupt flag, and RXS are set.

**Bit 6 – RXS: USART RX Start**
The RXS flag is set when a start condition is detected on the RxD line. If the RXSIE bit and the Global Interrupt Enable flag are set, an RX Start Interrupt will be generated when the flag is set. The flag can only be cleared by writing a logical one on the RXS bit location.

If the start frame detector is enabled (RXSIE = 1) and the Global Interrupt Enable flag is set, the RX Start Interrupt will wake up the MCU from all sleep modes.

**Bit 5 – SFDE: Start Frame Detection Enable**
Writing this bit to one enables the USART Start Frame Detection mode. The start frame detector is able to wake up the MCU from sleep mode when a start condition, i.e. a high (IDLE) to low (START) transition, is detected on the RxD line.

**Table 15-15. USART Start Frame Detection Modes**

| UCSR0D.SFDE | UCSR0D.RXSIE | UCSR0B.RXCIE | Description |
|---|---|---|---|
| 0 | X | X | Start frame detector disabled |
| 1 | 0 | 0 | Reserved |
| 1 | 0 | 1 | Start frame detector enabled. RXC flag wakes up MCU from all sleep modes |
| 1 | 1 | 0 | Start frame detector enabled. RXS flag wakes up MCU from all sleep modes |
| 1 | 1 | 1 | Start frame detector enabled. Both RXC and RXS wake up the MCU from all sleep modes |

### 15.12.6. USART Baud Rate 0 Register Low and High byte

The UBRR0L and UBRR0H register pair represents the 16-bit value, UBRR0.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

**Name:** UBRR0L and UBRR0H
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | UBRR0[11:8] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | UBRR0[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – UBRR0[11:0]: USART Baud Rate**

This is a 12-bit register which contains the USART baud rate. The UBRR0H contains the four most significant bits and the UBRR0L contains the eight least significant bits of the USART 0 baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR0L will trigger an immediate update of the baud rate prescaler.

# 16. USARTSPI - USART in SPI Mode

## 16.1. Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation.

Setting both UMSELn[1:0] bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

## 16.2. Features

- Full Duplex, Three-wire Synchronous Data Transfer
- Master Operation
- Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)
- LSB First or MSB First Data Transfer (Configurable Data Order)
- Queued Operation (Double Buffered)
- High Resolution Baud Rate Generator
- High Speed Operation ($f_{XCKmax} = f_{CK}/2$)
- Flexible Interrupt Generation

## 16.3. Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The table below contains the equations for calculating the baud rate or UBRRn setting for Synchronous Master Mode.

**Table 16-1. Equations for Calculating Baud Rate Register Setting**

| Operating Mode | Equation for Calculating Baud Rate[1] | Equation for Calculating UBRRn Value |
|---|---|---|
| Synchronous Master mode | $\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}n + 1)}$ | $\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$ |

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

| | |
|---|---|
| BAUD | Baud rate (in bits per second, bps) |
| $f_{OSC}$ | System Oscillator clock frequency |
| UBRRn | Contents of the UBRRnH and UBRRnL Registers, (0-4095) |

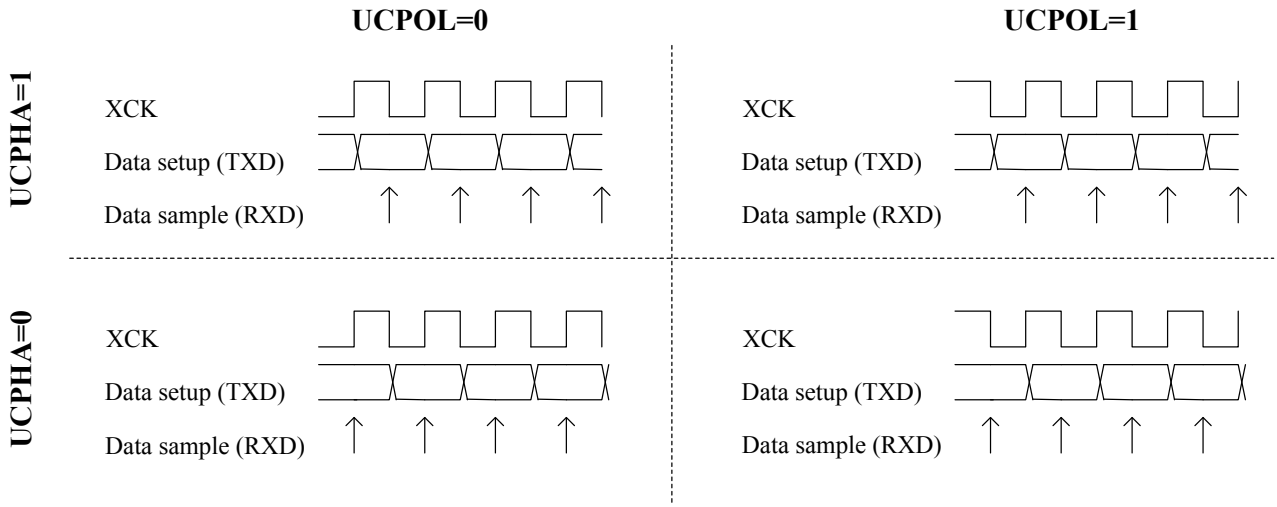
## 16.4. SPI Data Modes and Timing

There are four combinations of XCKn (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHAn and UCPOLn. The data transfer timing diagrams are shown in the following figure. Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAn functionality is summarized in the following table. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

**Table 16-2. UCPOLn and UCPHAn Functionality**

| UCPOLn | UCPHAn | SPI Mode | Leading Edge | Trailing Edge |
|---|---|---|---|---|
| 0 | 0 | 0 | Sample (Rising) | Setup (Falling) |
| 0 | 1 | 1 | Setup (Rising) | Sample (Falling) |
| 1 | 0 | 2 | Sample (Falling) | Setup (Rising) |
| 1 | 1 | 3 | Setup (Falling) | Sample (Rising) |

**Figure 16-1. UCPHAn and UCPOLn data transfer timing diagrams.**




## 16.5. Frame Formats

A serial frame for the MSPIM is defined to be one character of eight data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. Then the next data bits, up to a total of eight, are succeeding, ending with the most or least significant bit accordingly. When a complete frame is transmitted, a new frame can directly follow it, or the communication line can be set to an idle (high) state.

The UDORDn bit in UCSRnC sets the frame format used by the USART in MSPIM mode. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

### 16.5.1. USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

**Note:** To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

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

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

**Assembly Code Example**

```
clr r18
out UBRRnH,r18
out UBRRnL,r18
; Setting the XCKn port pin as output, enables master mode.
sbi XCKn_DDR, XCKn
; Set MSPI mode of operation and SPI data mode 0.
ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
out UCSRnC,r18
; Enable receiver and transmitter.
ldi r18, (1<<RXENn)|(1<<TXENn)
out UCSRnB,r18
; Set baud rate.
; IMPORTANT: The Baud Rate must be set after the transmitter is enabled!
out UBRRnH, r17
out UBRRnL, r18
ret
```

**C Code Example**

```
{
 UBRRn = 0;
 /* Setting the XCKn port pin as output, enables master mode. */
 XCKn_DDR |= (1<<XCKn);
 /* Set MSPI mode of operation and SPI data mode 0. */
 UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
 /* Enable receiver and transmitter. */
 UCSRnB = (1<<RXENn)|(1<<TXENn);
 /* Set baud rate. */
```

```
 /* IMPORTANT: The Baud Rate must be set after the transmitter is enabled */
 UBRRn = baud;
}
```

## 16.6. Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, i.e. the TXENn bit in the UCSRnB register is set to one. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the Receiver's serial input. The XCKn will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDRn I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

**Note:** To keep the input buffer in sync with the number of data bytes transmitted, the UDRn register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDREn) Flag and the Receive Complete (RXCn) Flag. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16 and the data received will be available in the same register (R16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDREn Flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

**Assembly Code Example**

```
USART_MSPIM_Transfer:
 ; Wait for empty transmit buffer
 in r16, UCSRnA
 sbrs r16, UDREn
 rjmp USART_MSPIM_Transfer
 ; Put data (r16) into buffer, sends the data
 out UDRn,r16
 ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
 in r16, UCSRnA
 sbrs r16, RXCn
 rjmp USART_MSPIM_Wait_RXCn
 ; Get and return received data from buffer
 in r16, UDRn
 ret
```

**C Code Example**

```
{
 /* Wait for empty transmit buffer */
 while ( !( UCSRnA & (1<<UDREn)) );
 /* Put data into buffer, sends the data */
 UDRn = data;
```

```
  /* Wait for data to be received */
  while ( !(UCSRnA & (1<<RXCn)) );
  /* Get and return received data from buffer */
  return UDRn;
}
```

### 16.6.1. Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREn flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and is always read as zero.

### 16.6.2. Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

## 16.7. AVR USART MSPIM vs. AVR SPI

The USART in MSPIM mode is fully compatible with the AVR SPI regarding:

- Master mode timing diagram
- The UCPOLn bit functionality is identical to the SPI CPOL bit
- The UCPHAn bit functionality is identical to the SPI CPHA bit
- The UDORDn bit functionality is identical to the SPI DORD bit

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer
- The USART in MSPIM mode receiver includes an additional buffer level
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting UBRRn accordingly
- Interrupt timing is not compatible
- Pin control differs due to the master only operation of the USART in MSPIM mode

A comparison of the USART in MSPIM mode and the SPI pins is shown in the table below.

**Table 16-3. Comparison of USART in MSPIM mode and SPI pins**

| USART_MSPIM | SPI | Comments |
|---|---|---|
| TxDn | MOSI | Master Out only |
| RxDn | MISO | Master In only |
| XCKn | SCK | (Functionally identical) |
| (N/A) | $\overline{SS}$ | Not supported by USART in MSPIM |

## 16.8. Register Description

Refer to the USART Register Description.

# 17. TC0 - 16-bit Timer/Counter0 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**

Pin Configurations on page 12

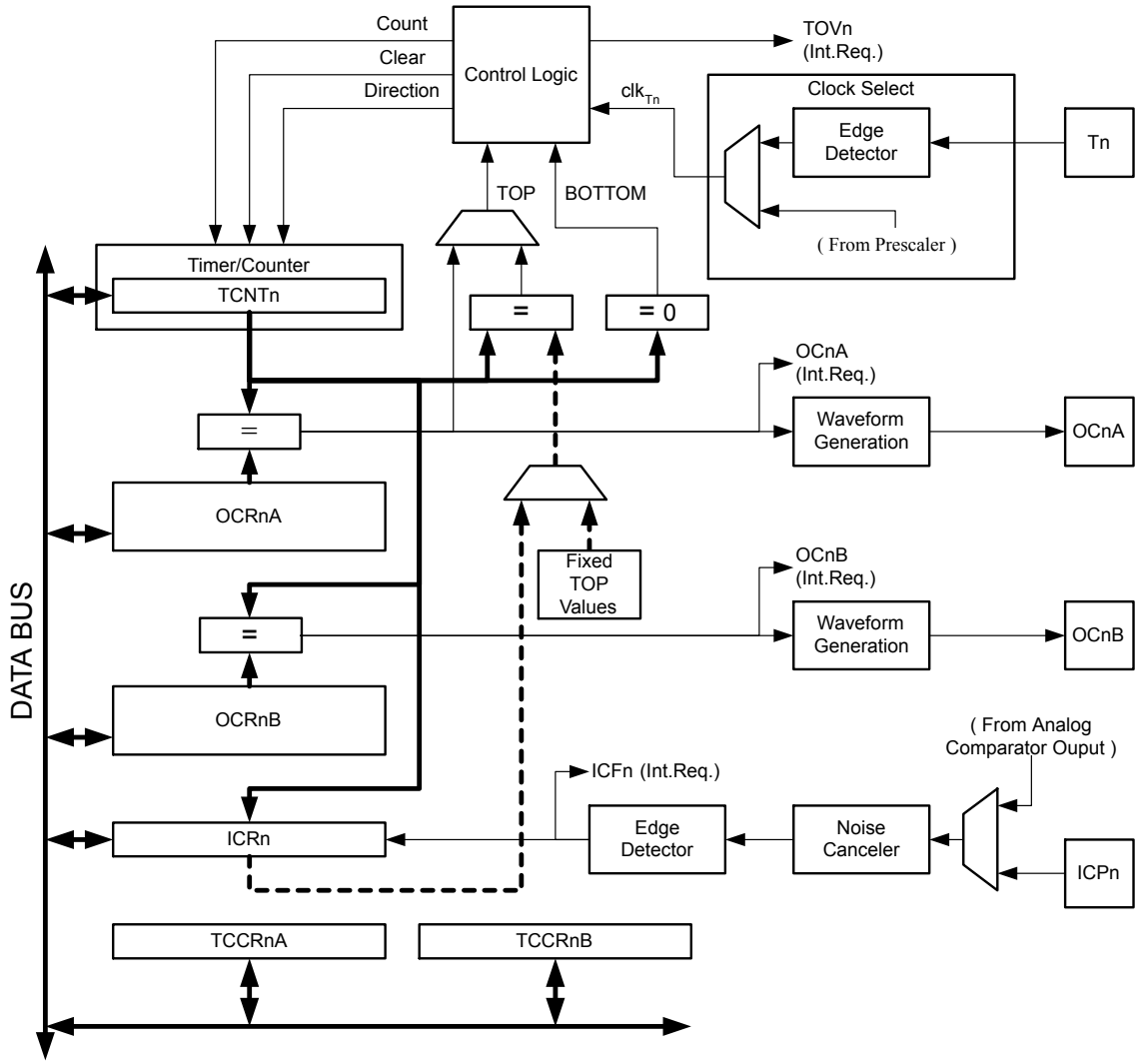
## 17.2. Features

- 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
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Independent interrupt Sources (TOV, OCFA, OCFB, and ICF)

## 17.3. Block Diagram

The Power Reduction TC0 bit in the Power Reduction Register (PRR0.PRTIM0) must be written to zero to enable the TC0 module.

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

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

The following definitions are used throughout the section:

**Table 17-1. Definitions**

| Constant | Description |
|---|---|
| BOTTOM | The counter reaches the BOTTOM when it becomes zero (0x00 for 8-bit counters, or 0x0000 for 16-bit counters). |
| MAX | The counter reaches its Maximum when it becomes 0xF (decimal 15, for 8-bit counters) or 0xFF (decimal 255, for 16-bit counters). |
| TOP | The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value MAX or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation. |

## 17.5. Registers

The Timer/Counter (TCNT0), Output Compare Registers (OCRA/B), and Input Capture Register (ICR0) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in section Accessing 16-bit Registers.

The Timer/Counter Control Registers (TCCR0A/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 (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

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

The double buffered Output Compare Registers (OCR0A/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 (OC0A/B). See Output Compare Units. The compare match event will also set the Compare Match Flag (OCF0A/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 (ICP0) 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 OCR0A Register, the ICR0 Register, or by a set of fixed values. When using OCR0A as TOP value in a PWM mode, the OCR0A Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR0 Register can be used as an alternative, freeing the OCR0A to be used as PWM output.

**Related Links**

ICR0L and ICR0H on page 156
OCR0AL and OCR0AH on page 154
OCR0BL and OCR0BH on page 155
TCNT0L and TCNT0H on page 153
TCCR0A on page 147
TCCR0B on page 150

TIFR0 on page 158
TIMSK0 on page 157
AC - Analog Comparator on page 160

## 17.6. Accessing 16-bit Timer/Counter Registers

The TCNT0, OCR0A/B, and ICR0 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be accessed byte-wise, using two read or write operations. Each 16-bit timer has a single 8-bit TEMP register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer.

Accessing the low byte triggers the 16-bit read or write operation: When the low byte of a 16-bit register is written by the CPU, the high byte that is currently stored in TEMP and the low byte being written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the TEMP register in the same clock cycle as the low byte is read, and must be read subsequently.

**Note:** To perform a 16-bit write operation, the low byte must be written before the high byte. For a 16-bit read, the low byte must be read before the high byte.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR0A/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 OCR0A/B and ICR0 Registers. Note that when using C, the compiler handles the 16-bit access.

**Assembly Code Example(1)**

```
    ...
; Set TCNT0 to 0x01FF
ldi     r17,0x01
ldi     r16,0xFF
out     TCNT0H,r17
out     TCNT0L,r16
; Read TCNT0 into r17:r16
in      r16,TCNT0L
in      r17,TCNT0H
    ...
```

The assembly code example returns the TCNT**0** value in the r17:r16 register pair.

**C Code Example(1)**

```
unsigned int i;
    ...
/* Set TCNT0 to 0x01FF */
TCNT0 = 0x1FF;
/* Read TCNT0 into i */
i = TCNT0;
    ...
```

**Note:**

1. The example code assumes that the part specific header file is included. For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

### Atomic Read

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to perform an atomic read of the TCNT0 Register contents. The OCR0A/B or ICR0 Registers can be ready by using the same principle.

**Assembly Code Example(1)**

```
TIM16_ReadTCNT0:
   ; Save global interrupt flag
   in    r18,SREG
   ; Disable interrupts
   cli
   ; Read TCNT0 into r17:r16
   in    r16,TCNT0L
   in    r17,TCNT0H
   ; Restore global interrupt flag
   out   SREG,r18
   ret
```

The assembly code example returns the TCNT0 value in the r17:r16 register pair.

**C Code Example(1)**

```
unsigned int TIM16_ReadTCNT0( void )
{
   unsigned char sreg;
   unsigned int i;
   /* Save global interrupt flag */
   sreg = SREG;
   /* Disable interrupts */
   _CLI();
   /* Read TCNT0 into i */
   i = TCNT0;
   /* 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 TCNT0 Register contents. Writing any of the OCR0A/B or ICR0 Registers can be done by using the same principle.

**Assembly Code Example(1)**

```
TIM16_WriteTCNT0:
   ; Save global interrupt flag
   in     r18,SREG
   ; Disable interrupts
   cli
   ; Set TCNT0 to r17:r16
   out    TCNT0H,r17
   out    TCNT0L,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 TCNT0.

**C Code Example(1)**

```
void TIM16_WriteTCNT0( unsigned int i )
{
   unsigned char sreg;
   unsigned int i;
   /* Save global interrupt flag */
   sreg = SREG;
   /* Disable interrupts */
   _CLI();
   /* Set TCNT0 to i */
   TCNT0 = 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.

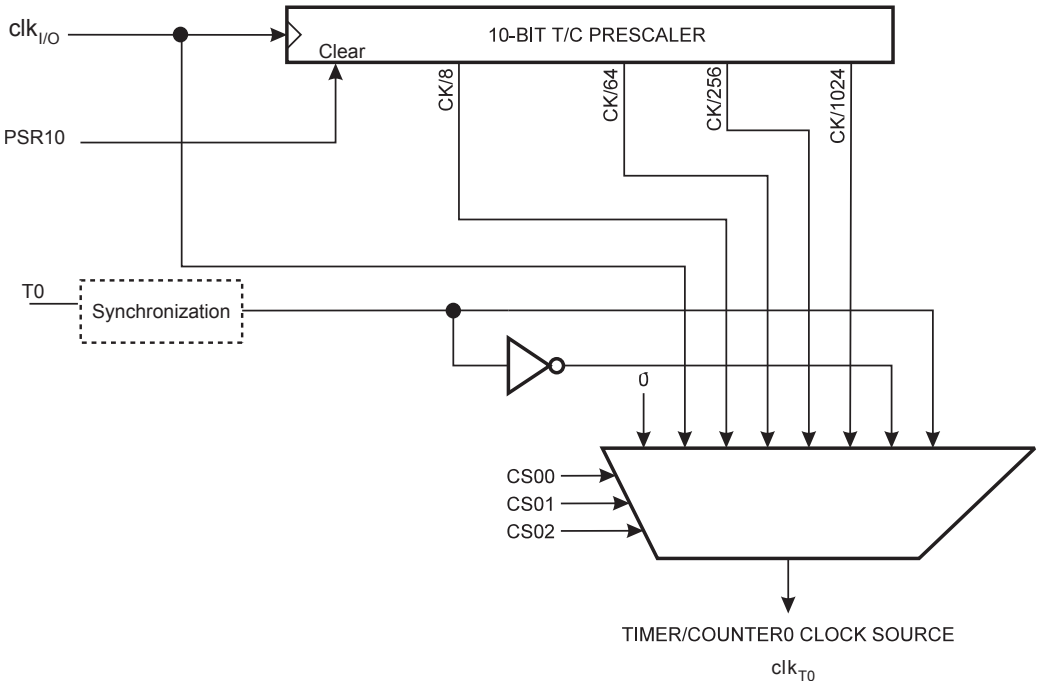
## 17.7. Timer/Counter Clock Sources

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 (TCCR0B.CS[2:0]).

### 17.7.1. Internal Clock Source - Prescaler

The Timer/Counter can be clocked directly by the system clock (by setting the TCCR0B.CS0[2:0]=0x1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$.

**Figure 17-2. Prescaler for Timer/Counter0**




### 17.7.2. Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by Timer/Counter 0 (T0). 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 (TCCR0B.CS0[2:0] = 2, 3, 4, or 5). 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).

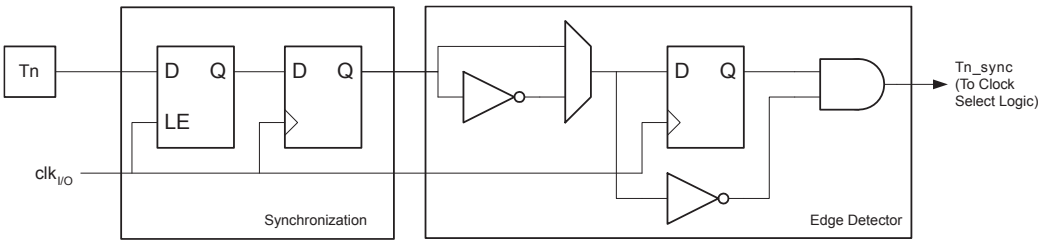
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

### 17.7.3. External Clock Source

An external clock source applied to the T0 pin can be used as Timer/Counter clock ($clk_{T0}$). The T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. See also the block diagram of the 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 $clk_{T0}$ pulse for each positive (CS0[2:0]=0x7) or negative (CS0[2:0]=0x6) edge it detects.

**Figure 17-3. T0 Pin Sampling**




**Note:** The “n” indicates the device number (n = 0 for Timer/Counter 0)

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 T0 pin to the counter is updated.

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

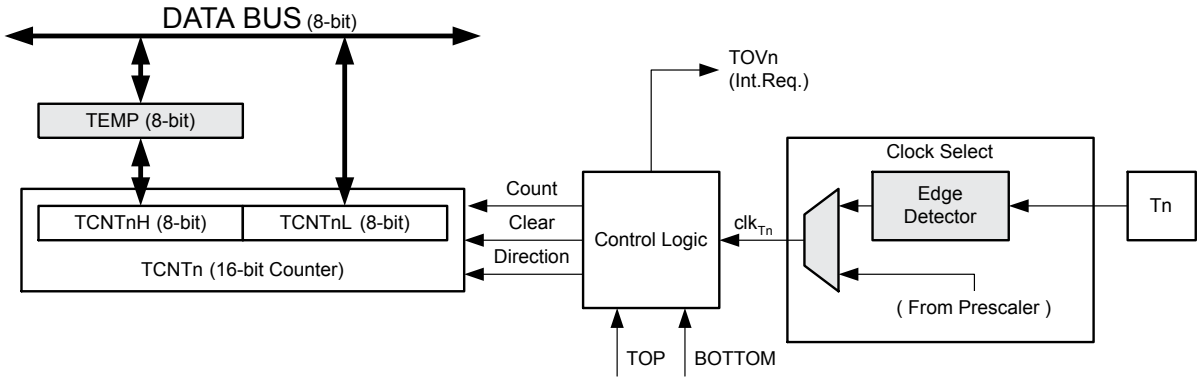
Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk_I/O}/2$) given a 50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by the tolerances of the oscillator source (crystal, resonator, and capacitors), it is recommended that maximum frequency of an external clock source is less than $f_{clk_I/O}/2.5$.

An external clock source can not be prescaled.

## 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 17-4. Counter Unit Block Diagram**




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

**Table 17-2. Signal description (internal signals)**

| Signal Name | Description |
|---|---|
| Count | Increment or decrement TCNT0 by 1. |
| Direction | Select between increment and decrement. |
| Clear | Clear TCNT0 (set all bits to zero). |

| Signal Name | Description |
|---|---|
| $clk_{T0}$ | Timer/Counter clock. |
| TOP | Signalize that TCNT0 has reached maximum value. |
| BOTTOM | Signalize that TCNT0 has reached minimum value (zero). |

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT0H) containing the upper eight bits of the counter, and Counter Low (TCNT0L) containing the lower eight bits. The TCNT0H Register can only be accessed indirectly by the CPU. When the CPU does an access to the TCNT0H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT0H value when the TCNT0L is read, and TCNT0H is updated with the temporary register value when TCNT0L 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 TCNT0 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 ($clk_{T0}$). The clock $clk_{T0}$ can be generated from an external or internal clock source, as selected by the Clock Select bits in the Timer/Counter0 Control Register B (TCCR0B.CS[2:0]). When no clock source is selected (CS[2:0]=0x0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, independent of whether $clk_{T0}$ 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 (TCCR0B.WGM0[3:2] and TCCR0A.WGM0[1:0]). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0x. For more details about advanced counting sequences and waveform generation, see Modes of Operation.

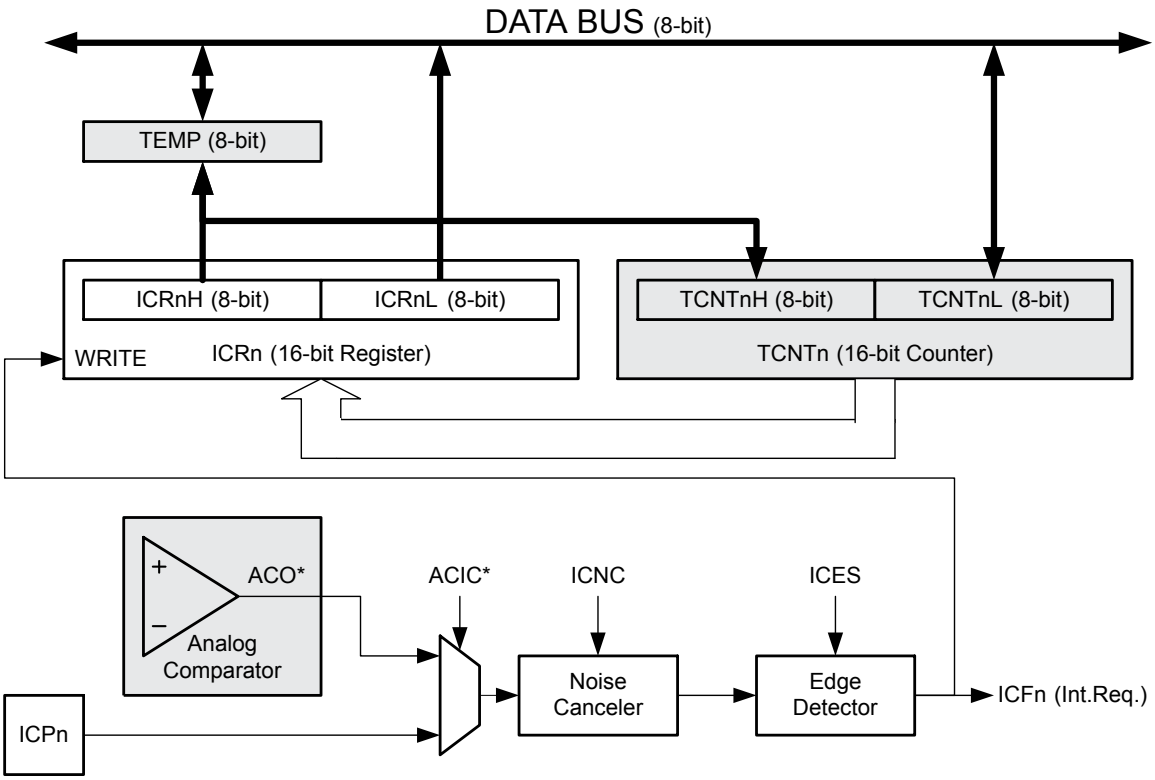
The Timer/Counter Overflow Flag in the TC0 Interrupt Flag Register (TIFR0.TOV) is set according to the mode of operation selected by the WGM0[3:0] bits. TOV can be used for generating a CPU interrupt.

## 17.9. Input Capture Unit

The Timer/Counter0 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 ICP0 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 17-5. Input Capture Unit Block Diagram for TC0**




**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).

When a change of the logic level (an event) occurs on the Input Capture pin (ICP0), 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 (TCNT0) is written to the Input Capture Register (ICR0). The Input Capture Flag (ICF) is set at the same system clock cycle as the TCNT0 value is copied into the ICR0 Register. If enabled (TIMSK0.ICIE=1), the Input Capture Flag generates an Input Capture interrupt. The ICF0 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF Flag can be cleared by software by writing '1' to its I/O bit location.

Reading the 16-bit value in the Input Capture Register (ICR0) is done by first reading the low byte (ICR0L) and then the high byte (ICR0H). When the low byte is read form ICR0L, the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR0H I/O location it will access the TEMP Register.

The ICR0 Register can only be written when using a Waveform Generation mode that utilizes the ICR0 Register for defining the counter’s TOP value. In these cases the Waveform Generation mode bits (WGM0[3:0]) must be set before the TOP value can be written to the ICR0 Register. When writing the ICR0 Register, the high byte must be written to the ICR0H I/O location before the low byte is written to ICR0L.

See also Accessing 16-bit Timer/Counter Registers.

### 17.9.1. Input Capture Trigger Source

The main trigger source for the Input Capture unit is the Input Capture pin (ICP0). Timer/Counter0 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the Analog Comparator Input Capture (ACIC) bit in

the Analog Comparator Control and Status Register (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the Input Capture pin (ICP0) and the Analog Comparator output (ACO) inputs are sampled using the same technique as for the T0 pin. The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. The input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR0 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP0 pin.

**Related Links**

ACSRA on page 162

### 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 (TCCR0B.ICNC). When enabled, the noise canceler introduces an additional delay of four system clock cycles between a change applied to the input and the update of the ICR0 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 ICR0 Register before the next event occurs, the ICR0 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 ICR0 Register should be read as early in the interrupt handler routine as possible. Even though the Input Capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

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

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR0 Register has been read. After a change of the edge, the Input Capture Flag (ICF) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF Flag is not required (if an interrupt handler is used).

## 17.10. Output Compare Units

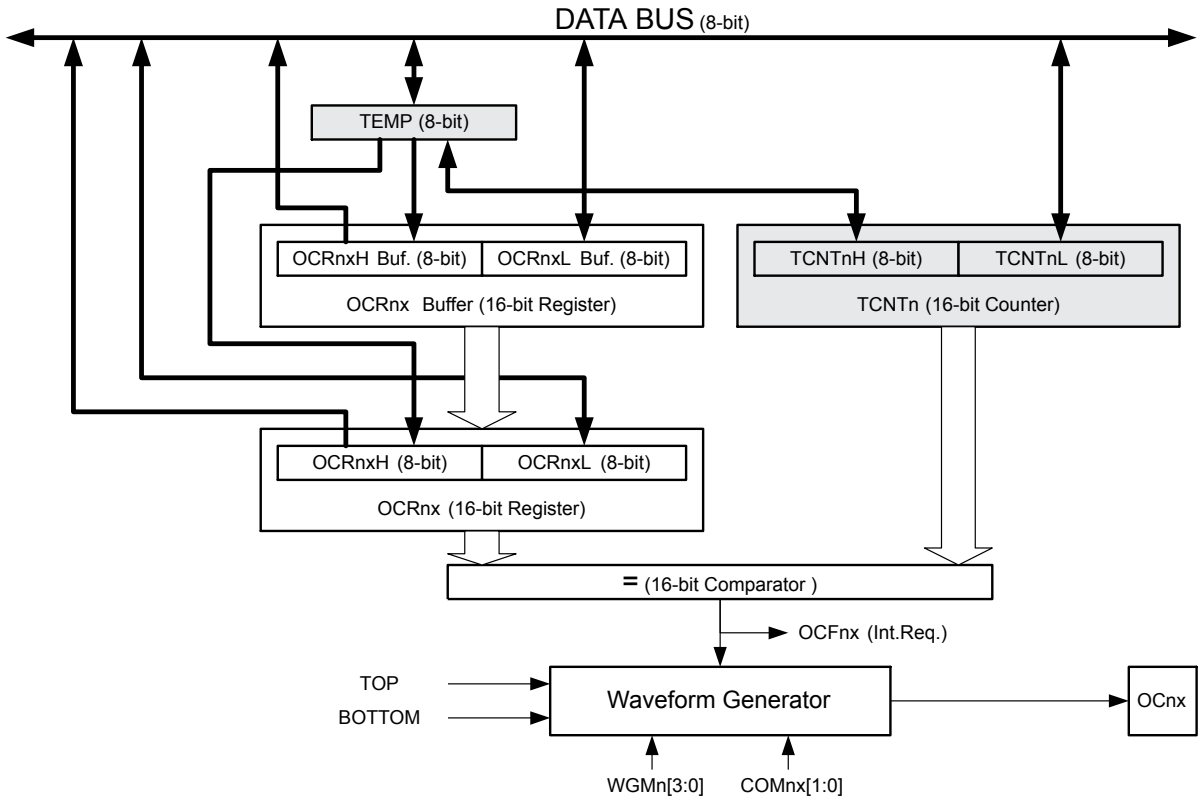
The 16-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0x). If TCNT equals OCR0x the comparator signals a match. A match will set the Output Compare Flag (TIFR0.OCFx) at the next timer clock cycle. If enabled (TIMSK0.OCIEx = 1), the Output Compare Flag generates an Output Compare interrupt. The OCFx Flag is automatically cleared when the interrupt is executed. Alternatively the OCFx Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the Waveform Generation mode (WGM0[3:0]) bits and Compare Output mode (COM0x[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 17-6. 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 Register is double buffered when using any of the twelve Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

When double buffering is enabled, the CPU has access to the OCR0x Buffer Register. When double buffering is disabled, the CPU will access the OCR0x directly.

The content of the OCR0x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT0 and ICR0 Register). Therefore OCR0x 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 OCR0x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR0xH) 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 (OCR0xL) is written to the lower eight bits, the high byte will be

copied into the upper 8-bits of either the OCR0x buffer or OCR0x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to Accessing 16-bit Timer/Counter Registers.

### 17.10.1. 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.FOC0x) bit. Forcing compare match will not set the OCF0x Flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the TCCR0A.COM0x[1:0] bits define whether the OC0x pin is set, cleared or toggled).

### 17.10.2. Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

### 17.10.3. Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down counting.

The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (FOC0x) strobe bits in Normal mode. The OC0x Registers keep their values even when changing between Waveform Generation modes.

Be aware that the TCCR0A.COM0x[1:0] bits are not double buffered together with the compare value. Changing the TCCR0A.COM0x[1:0] bits will take effect immediately.

## 17.11. Compare Match Output Unit

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

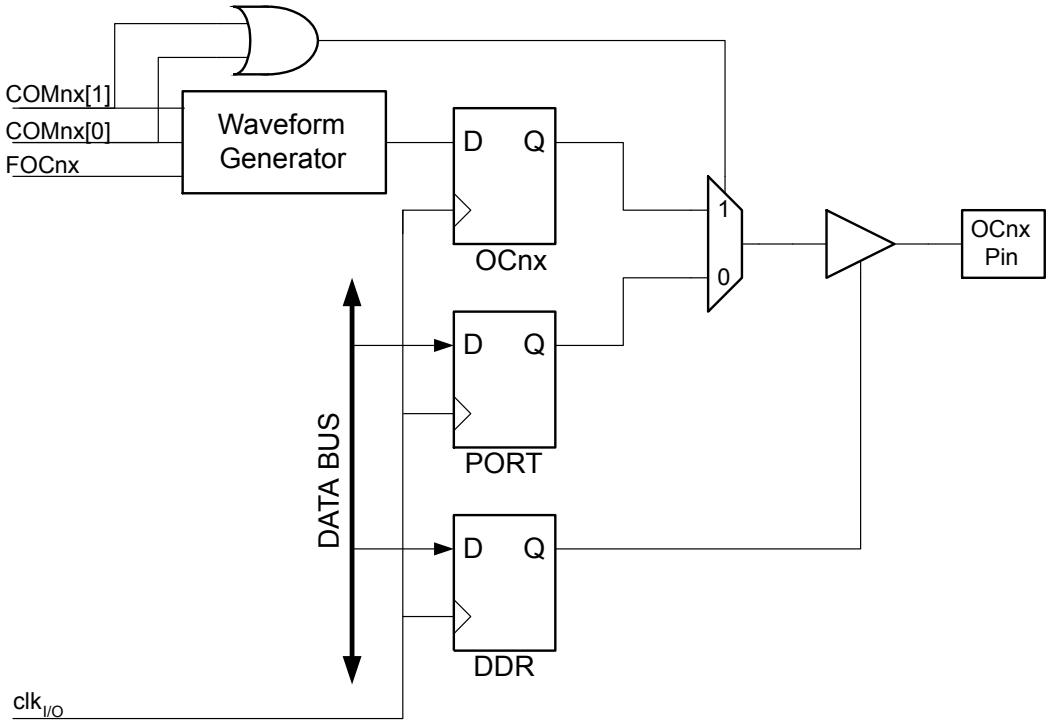
- The Waveform Generator uses the COM0x bits for defining the Output Compare (OC0x) register state at the next compare match.
- The COM0x bits control the OC0x pin output source

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

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

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

**Figure 17-7. Compare Match Output Unit, Schematic**




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

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

### 17.11.1. Compare Output Mode and Waveform Generation

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

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

## 17.12. 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 (WGM0[3:0]) and Compare Output mode (TCCR0A.COM0x[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The TCCR0A.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

TCCR0A.COM0x[1:0] bits control whether the output should be set, cleared, or toggle at a compare match.

### 17.12.1. Normal Mode

The simplest mode of operation is the Normal mode (TCCR0A.WGM0[3:0]=0x0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX=0xFFFF) and then restarts from BOTTOM=0x0000. In normal operation the Timer/Counter Overflow Flag (TIFR0.TOV) will be set in the same timer clock cycle as the TCNT0 becomes zero. In this case, the TOV Flag in behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

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

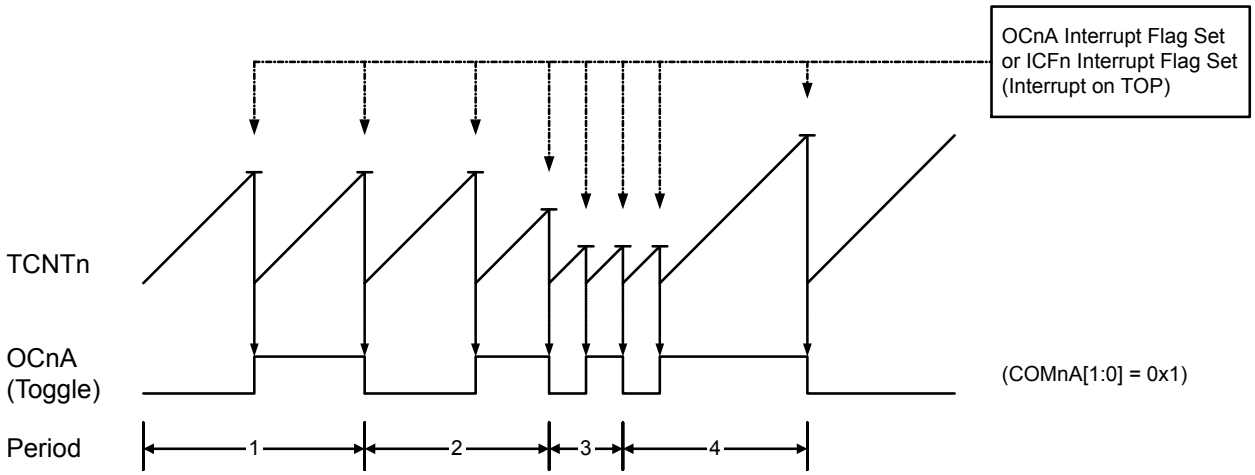
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

### 17.12.2. Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC modes (mode 4 or 12, WGM0[3:0]=0x4 or 0xC), the OCR0A or ICR0 registers are used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNT0) matches either the OCR0A (if WGM0[3:0]=0x4) or the ICR0 (WGM0[3:0]=0xC). The OCR0A or ICR0 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown below. The counter value (TCNT0) increases until a compare match occurs with either OCR0A or ICR0, and then TCNT0 is cleared.

**Figure 17-8. CTC 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).

An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF0A or ICF0 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 OCR0A is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then count to its maximum value (0xFF for a 8-bit counter, 0xFFFF for a 16-bit counter) and wrap around starting at 0x00 before the compare match will occur.
In many cases this feature is not desirable. An alternative will then be to use the Fast PWM mode using OCR0A for defining TOP (WGM0[3:0]=0xF), since the OCR0A then will be double buffered.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM0A[1:0]=0x1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC0A=1). The waveform generated will have a maximum frequency of $f_{OC0A} = f_{clk_I/O}/2$ when OCR0A 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 = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).
- *N* represents the prescaler factor (1, 8, 64, 256, or 1024).

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

### 17.12.3. Fast PWM Mode

The Fast Pulse Width Modulation or Fast PWM modes (modes 5, 6, 7, 14,and 15, WGM0[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.

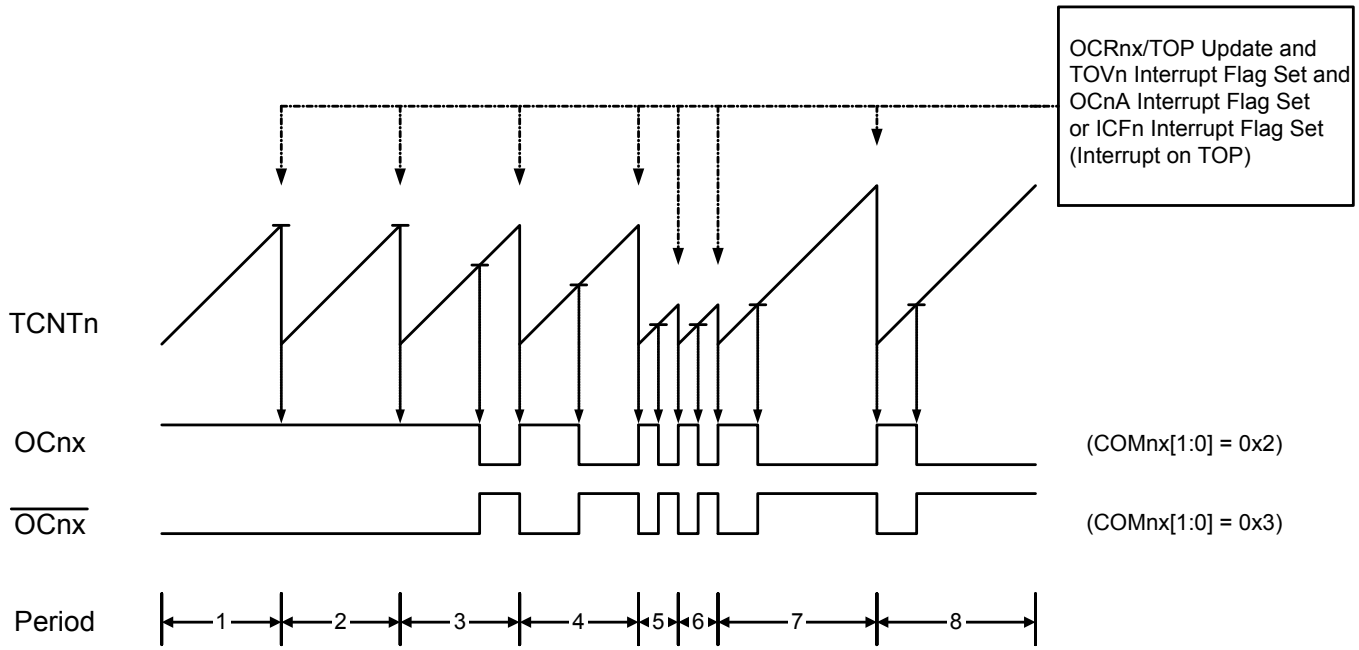
In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the Fast PWM 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 ICR0 or OCR0A. The minimum resolution allowed is 2-bit (ICR0 or OCR0A register set to 0x0003), and the maximum resolution is 16-bit (ICR0 or OCR0A 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 (WGM0[3:0] = 0x5, 0x6, or 0x7), the value in ICR0 (WGM0[3:0]=0xE), or the value in OCR0A (WGM0[3:0]=0xF). The counter is then cleared at the following timer clock cycle. The timing diagram for the Fast PWM mode using OCR0A or ICR0 to define TOP 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. The OC0x Interrupt Flag will be set when a compare match occurs.

**Figure 17-9. Fast 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 TOP. In addition, when either OCR0A or ICR0 is used for defining the TOP value, the OC0A or ICF0 Flag is set at the same timer clock cycle TOV0 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 TCNT0 and the OCR0x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR0x Registers are written.

The procedure for updating ICR0 differs from updating OCR0A when used for defining the TOP value. The ICR0 Register is not double buffered. This means that if ICR0 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 ICR0 value written is lower than the current value of TCNT0. 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 OCR0A Register however, is double buffered. This feature allows the OCR0A I/O location to be written anytime. When the OCR0A I/O location is written the value written will be put into the OCR0A Buffer Register. The OCR0A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT0 matches TOP. The update is done at the same timer clock cycle as the TCNT0 is cleared and the TOV0 Flag is set.

Using the ICR0 Register for defining TOP works well when using fixed TOP values. By using ICR0, the OCR0A Register is free to be used for generating a PWM output on OC0A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR0A 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 OC0x pins. Writing the COM0x[1:0] bits to 0x2 will produce an inverted PWM and a non-inverted PWM output can be generated by writing the COM0x[1:0] to 0x3. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC0x). 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:

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

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

The extreme values for the OCR0x registers represents special cases when generating a PWM waveform output in the Fast PWM mode. If the OCR0x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR0x equal to TOP will result in a constant high or low output (depending on the polarity of the output which is controlled by COM0x[1:0]).

A frequency waveform output with 50% duty cycle can be achieved in Fast PWM mode by selecting OC0A to toggle its logical level on each compare match (COM0A[1:0]=0x1). This applies only if OCR0A is used to define the TOP value (WGM0[3:0]=0xF). The waveform generated will have a maximum frequency of $f_{OC0A} = f_{clk_I/O}/2$ when OCR0A is set to zero (0x0000). This feature is similar to the OC0A 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

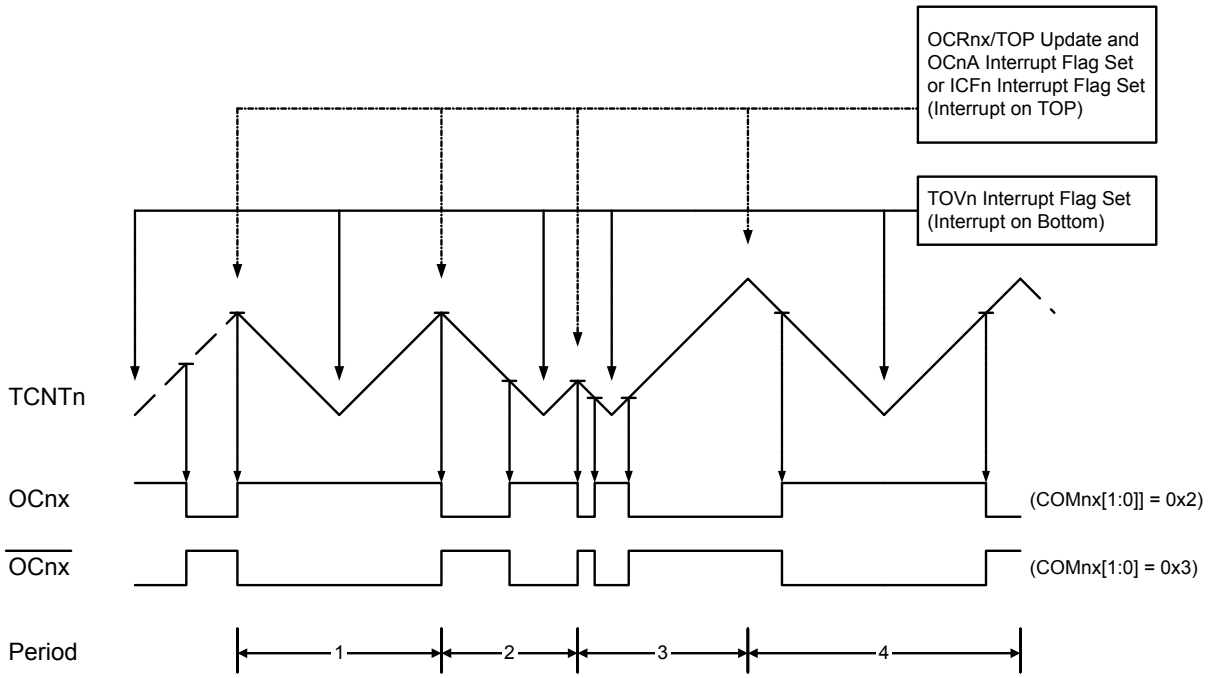
The Phase Correct Pulse Width Modulation or Phase Correct PWM modes (WGM0[3:0]= 0x1, 0x2, 0x3, 0xA, and 0xB) provide a high resolution, phase correct PWM waveform generation option. The Phase Correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x 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 ICR0 or OCR0A. The minimum resolution allowed is 2-bit (ICR0 or OCR0A set to 0x0003), and the maximum resolution is 16-bit (ICR0 or OCR0A 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 (WGM0[3:0]= 0x1, 0x2, or 0x3), the value in ICR0 (WGM0[3:0]=0xA), or the value in OCR0A (WGM0[3:0]=0xB). The counter has then reached the TOP and 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, using OCR0A or ICR0 to define TOP. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT0 slopes mark compare matches between OCR0x and TCNT0. The OC0x Interrupt Flag will be set when a compare match occurs.

**Figure 17-10. 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. When either OCR0A or ICR0 is used for defining the TOP value, the OC0A or ICF0 Flag is set accordingly at the same timer clock cycle as the OCR0x 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 TCNT0 and the OCR0x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR0x 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 OCR0x Register. Since the OCR0x 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 OC0x pins. Writing COM0x[1:0] bits to 0x2 will produce a non-inverted PWM. An inverted PWM output can be generated by writing the COM0x[1:0] to 0x3. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC0x). The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and clearing (or setting) 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 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 OCR0x Register represent special cases when generating a PWM waveform output in the Phase Correct PWM mode. If the OCR0x 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 OCR0A is used to define the TOP value (WGM0[3:0]=0xB) and COM0A[1:0]=0x1, the OC0A 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 (WGM0[3:0] = 0x8 or 0x9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x 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.

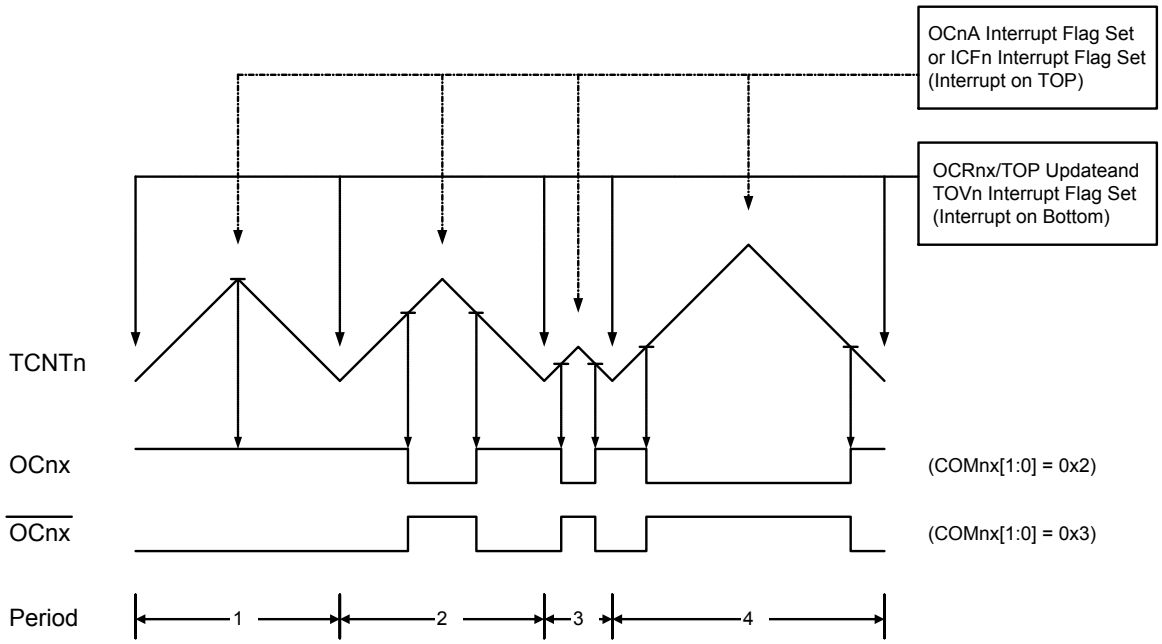
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR0x Register is updated by the OCR0x Buffer Register, (see Figure 17-10 and the Timing Diagram below).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR0 or OCR0A. The minimum resolution allowed is 2-bit (ICR0 or OCR0A set to 0x0003), and the maximum resolution is 16-bit (ICR0 or OCR0A 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 ICR0 (WGM0[3:0]=0x8), or the value in OCR0A (WGM0[3:0]=0x9). The counter has then reached the TOP and changes the count direction. The TCNT0 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 OCR0A or ICR0 is used to define TOP. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0. The OC0x Interrupt Flag will be set when a compare match occurs.

**Figure 17-11. Phase and Frequency 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 at the same timer clock cycle as the OCR0x Registers are updated with the double buffer value (at BOTTOM). When either OCR0A or ICR0 is used for defining the TOP value, the OC0A or ICF0 Flag set when TCNT0 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 TCNT0 and the OCR0x.

As shown in the timing diagram above, the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR0x 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 ICR0 Register for defining TOP works well when using fixed TOP values. By using ICR0, the OCR0A Register is free to be used for generating a PWM output on OC0A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR0A 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 OC0x pins. Setting the COM0x[1:0] bits to 0x2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x[1:0] to 0x3 (See description of TCCRA.COM0x). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC0x). The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and clearing (or setting) the OC0x Register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{\mathrm{OCnxPFCPWM}} = \frac{f_{\mathrm{clk_I/O}}}{2 \cdot N \cdot \mathrm{TOP}}$$

**Note:**

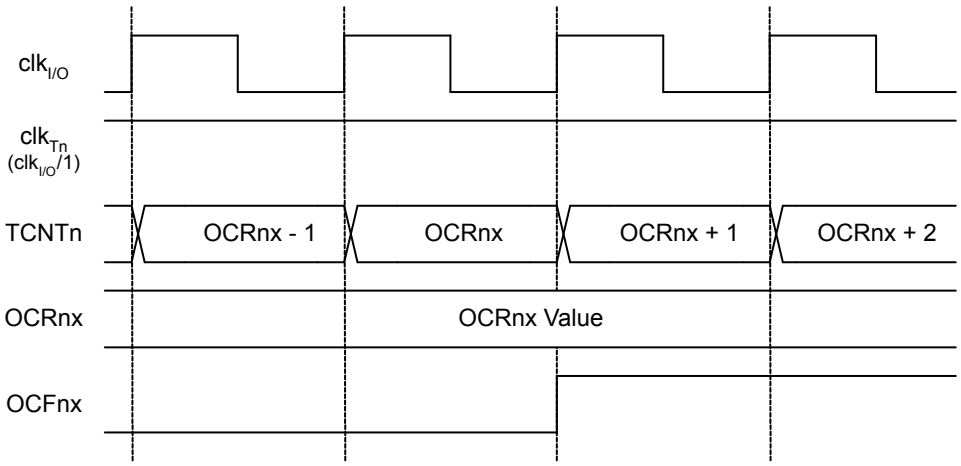
- The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).
- *N* represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR0x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0x 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 OCR0A is used to define the TOP value (WGM0[3:0]=0x9) and COM0A[1:0]=0x1, the OC0A output will toggle with a 50% duty cycle.

## 17.13. Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock ($clk_{T0}$) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCR0x Register is updated with the OCR0x buffer value (only for modes utilizing double buffering). The first figure shows a timing diagram for the setting of OCF0x.

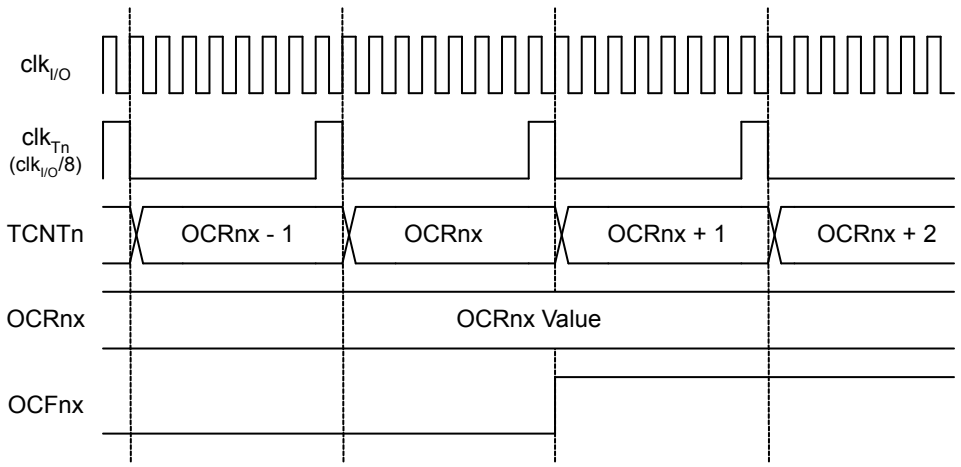
**Figure 17-12. Timer/Counter Timing Diagram, Setting of OCF0x, no Prescaling**




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

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

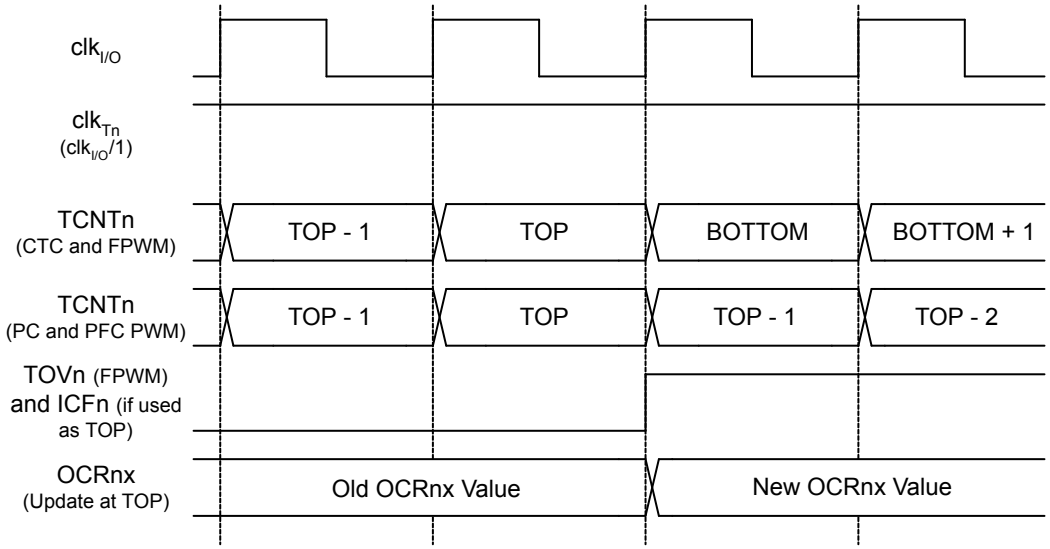
**Figure 17-13. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk_I/O}/8$)**



**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 count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR0x Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV0 Flag at BOTTOM.

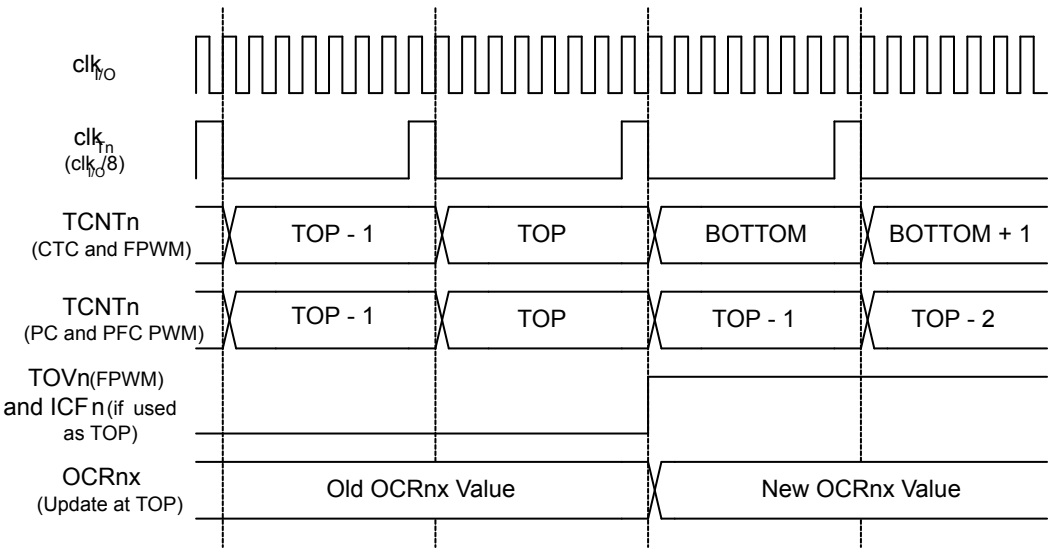
**Figure 17-14. Timer/Counter Timing Diagram, no Prescaling.**




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

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

**Figure 17-15. 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).

## 17.14. Register Description

### 17.14.1. Timer/Counter0 Control Register A

**Name:** TCCR0A
**Offset:** 0x2E
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | COM0A1 | COM0A0 | COM0B1 | COM0B0 | | | WGM01 | WGM00 |
| Access | R/W | R/W | R/W | R/W | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

**Bits 7:6 – COM0An: Compare Output Mode for Channel A [n = 1:0]**

**Bits 5:4 – COM0Bn: Compare Output Mode for Channel B [n = 1:0]**
The COM0A[1:0] and COM0B[1:0] control the Output Compare pins (OC0A and OC0B respectively) behavior. If one or both of the COM0A[1:0] bits are written to one, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM0B[1:0] bit are written to one, 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 OC0A or OC0B pin must be set in order to enable the output driver.

When the OC0A or OC0B is connected to the pin, the function of the COM0x[1:0] bits is dependent of the WGM0[3:0] bits setting. The table below shows the COM0x[1:0] bit functionality when the WGM0[3:0] bits are set to a Normal or a CTC mode (non-PWM).

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

| COM0A1/COM0B1 | COM0A0/COM0B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A/OC0B disconnected. |
| 0 | 1 | Toggle OC0A/OC0B on Compare Match. |
| 1 | 0 | Clear OC0A/OC0B on Compare Match (Set output to low level). |
| 1 | 1 | Set OC0A/OC0B on Compare Match (Set output to high level). |

The table below shows the COM0x[1:0] bit functionality when the WGM0[3:0] bits are set to the fast PWM mode.

**Table 17-4. Compare Output Mode, Fast PWM**

| COM0A1/ COM0B1 | COM0A0/ COM0B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A/OC0B disconnected. |
| 0 | 1 | WGM0[3:0]=0: Normal port operation, OC0A/OC0B disconnected<br>WGM0[3:0]=1: Toggle OC0A on compare match, OC0B reserved |

| COM0A1/ COM0B1 | COM0A0/ COM0B0 | Description |
|---|---|---|
| 1[1] | 0 | Clear OC0A/OC0B on Compare Match, set OC0A/OC0B at BOTTOM (non-inverting mode) |
| 1[1] | 1 | Set OC0A/OC0B on Compare Match, clear OC0A/OC0B at BOTTOM (inverting mode) |

**Note:**

1. A special case occurs when OCR0A/OCR0B equals TOP and COM0A1/COM0B1 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 COM0x[1:0] bit functionality when the WGM0[3:0] bits 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**

| COM0A1/ COM0B1 | COM0A0/ COM0B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A/OC0B disconnected. |
| 0 | 1 | WGM0[3:0]=0: Normal port operation, OC0A/OC0B disconnected<br>WGM0[3:0]=1: Toggle OC0A on compare match, OC0B reserved |
| 1[1] | 0 | Clear OC0A/OC0B on Compare Match when up-counting. Set OC0A/OC0B on Compare Match when down-counting. |
| 1[1] | 1 | Set OC0A/OC0B on Compare Match when up-counting. Clear OC0A/OC0B on Compare Match when down-counting. |

**Note:**

1. A special case occurs when OCR0A/OCR0B equals TOP and COM0A1/COM0B1 is set. Refer to Phase Correct PWM Mode for details.

**Bits 1:0 – WGM0n: Waveform Generation Mode [n = 1:0]**

Combined with the WGM0[3:2] bits 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 three types of Pulse Width Modulation (PWM) modes. (See Modes of Operation).

**Table 17-6. Waveform Generation Mode Bit Description**

| Mode | WGM0[3:0] | Timer/Counter Mode of Operation | TOP | Update of OCR0x at | TOV0 Flag Set on |
|---|---|---|---|---|---|
| 0 | 0000 | Normal | 0xFFFF | Immediate | MAX |
| 1 | 0001 | PWM, Phase Correct, 8-bit | 0x00FF | TOP | BOTTOM |
| 2 | 0010 | PWM, Phase Correct, 9-bit | 0x01FF | TOP | BOTTOM |
| 3 | 0011 | PWM, Phase Correct, 10-bit | 0x03FF | TOP | BOTTOM |
| 4 | 0100 | CTC (Clear Timer on Compare) | OCR0A | Immediate | MAX |
| 5 | 0101 | Fast PWM, 8-bit | 0x00FF | TOP | TOP |
| 6 | 0110 | Fast PWM, 9-bit | 0x01FF | TOP | TOP |

| Mode | WGM0[3:0] | Timer/Counter Mode of Operation | TOP | Update of OCR0x at | TOV0 Flag Set on |
|---|---|---|---|---|---|
| 7 | 0111 | Fast PWM, 10-bit | 0x03FF | TOP | TOP |
| 8 | 1000 | PWM, Phase and Frequency Correct | ICR0 | BOTTOM | BOTTOM |
| 9 | 1001 | PWM, Phase and Frequency Correct | OCR0A | BOTTOM | BOTTOM |
| 10 | 1010 | PWM, Phase Correct | ICR0 | TOP | BOTTOM |
| 11 | 1011 | PWM, Phase Correct | OCR0A | TOP | BOTTOM |
| 12 | 1100 | CTC (Clear Timer on Compare) | ICR0 | Immediate | MAX |
| 13 | 1101 | Reserved | - | - | - |
| 14 | 1110 | Fast PWM | ICR0 | TOP | TOP |
| 15 | 1111 | Fast PWM | OCR0A | TOP | TOP |

### 17.14.2. Timer/Counter0 Control Register B

**Name:** TCCR0B
**Offset:** 0x2D
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICNC0 | ICES0 | | WGM03 | WGM02 | CS0[2:0] | | |
| Access | R/W | R/W | | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ICNC0: Input Capture Noise Canceler**
Setting this bit (to one) activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture pin (ICP0) is filtered. The filter function requires four successive equal valued samples of the ICP0 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

**Bit 6 – ICES0: Input Capture Edge Select**
This bit selects which edge on the Input Capture pin (ICP0) that is used to trigger a capture event. When the ICES0 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES0 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES0 setting, the counter value is copied into the Input Capture Register (ICR0). The event will also set the Input Capture Flag (ICF0), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR0 is used as TOP value (see description of the WGM0[3:0] bits located in the TCCR0A and the TCCR0B Register), the ICP0 is disconnected and consequently the Input Capture function is disabled.

**Bit 4 – WGM03: Waveform Generation Mode**
Refer to TCCR0A.

**Bit 3 – WGM02: Waveform Generation Mode**
Refer to TCCR0A.

**Bits 2:0 – CS0[2:0]: Clock Select**
The three Clock Select bits select the clock source to be used by the Timer/Counter. Refer to Figure 17-12 and Figure 17-13.

**Table 17-7. Clock Select Bit Description**

| CA0[2] | CA0[1] | CS0[0] | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer/Counter stopped). |
| 0 | 0 | 1 | $clk_{I/O}$/1 (No prescaling) |
| 0 | 1 | 0 | $clk_{I/O}$/8 (From prescaler) |
| 0 | 1 | 1 | $clk_{I/O}$/64 (From prescaler) |
| 1 | 0 | 0 | clkI/O/256 (From prescaler) |

| CA0[2] | CA0[1] | CS0[0] | Description |
|---|---|---|---|
| 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/Counter 0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

### 17.14.3. Timer/Counter0 Control Register C

**Name:** TCCR0C
**Offset:** 0x2C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FOC0A | FOC0B | | | | | | |
| Access | R/W | R/W | | | | | | |
| Reset | 0 | 0 | | | | | | |

**Bit 7 – FOC0A: Force Output Compare for Channel A**

**Bit 6 – FOC0B: Force Output Compare for Channel B**
The FOC0A/FOC0B bits are only active when the WGM0[3:0] bits specifies a non-PWM mode. When writing a logical one to the FOC0A/FOC0B bit, an immediate compare match is forced on the Waveform Generation unit. The OC0A/OC0B output is changed according to its COM0x[1:0] bits setting. Note that the FOC0A/FOC0B bits are implemented as strobes. Therefore it is the value present in the COM0x[1:0] bits that determine the effect of the forced compare.

A FOC0A/FOC0B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR0A as TOP. The FOC0A/FOC0B bits are always read as zero.

### 17.14.4. Timer/Counter 0 Low and High byte

The TCNT0L and TCNT0H register pair represents the 16-bit value, TCNT0.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

**Name:** TCNT0L and TCNT0H
**Offset:** 0x28
**Reset:** 0x00
**Property:** -

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TCNT0[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 |
|---|---|---|---|---|---|---|---|---|
| | 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 15:0 – TCNT0[15:0]: Timer/Counter 0**
TCNT0H and TCNT0L are combined into TCNT0.

Modifying the counter (TCNT0) while the counter is running introduces a risk of missing a compare match between TCNT0 and one of the OCR0x Registers.

Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock for all compare units.

### 17.14.5. Output Comparte Register A 0 Low and High byte

The OCR0AL and OCR0AH register pair represents the 16-bit value, OCR0A.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

**Name:** OCR0AL and OCR0AH
**Offset:** 0x26
**Reset:** 0x00
**Property:** -

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | OCR0A[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 |
|---|---|---|---|---|---|---|---|---|
| | 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 15:0 – OCR0A[15:0]: Output Compare 0 A**
OCR0AH and OCR0AL are combined into OCR0A.

The Output Compare Registers contain a 16-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 OC0x pin.

### 17.14.6. Output Comparte Register B 0 Low and High byte

The OCR0BL and OCR0BH register pair represents the 16-bit value, OCR0B.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

**Name:** OCR0BL and OCR0BH
**Offset:** 0x24
**Reset:** 0x00
**Property:** -

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | OCR0B[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 |
|---|---|---|---|---|---|---|---|---|
| | 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 15:0 – OCR0B[15:0]: Output Compare 0 B**
OCR0BH and OCR0BL are combined into OCR0B.

The Output Compare Registers contain a 16-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 OC0x pin.

### 17.14.7. Input Capture Register 0 Low and High byte

The ICR0L and ICR0H register pair represents the 16-bit value, ICR0.The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

**Name:** ICR0L and ICR0H
**Offset:** 0x22
**Reset:** 0x00
**Property:** -

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ICR0[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 |
|---|---|---|---|---|---|---|---|---|
| | ICR0[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 – ICR0[15:0]: Input Capture 0**

ICR0H and ICR0L are combined into ICR0.

The Input Capture is updated with the counter (TCNT0) value each time an event occurs on the ICP0 pin (or optionally on the Analog Comparator output for Timer/Counter0). The Input Capture can be used for defining the counter TOP value.

### 17.14.8. Timer/Counter0 Interrupt Mask Register

**Name:** TIMSK0
**Offset:** 0x2B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | ICIE0 | | | OCIE0B | OCIE0A | TOIE0 |
| Access | | | R/W | | | R/W | R/W | R/W |
| Reset | | | 0 | | | 0 | 0 | 0 |

**Bit 5 – ICIE0: Timer/Counter0, Input Capture Interrupt Enable**
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF0 Flag, located in TIFR0, is set.

**Bit 2 – OCIE0B: Timer/Counter0, Output Compare B Match Interrupt Enable**
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter 0 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF0B Flag, located in TIFR0, is set.

**Bit 1 – OCIE0A: Timer/Counter0, Output Compare A Match Interrupt Enable**
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF0A Flag, located in TIFR0, is set.

**Bit 0 – TOIE0: Timer/Counter0, Overflow Interrupt Enable**
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter0 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV0 Flag, located in TIFR0, is set.

### 17.14.9. Timer/Counter0 Interrupt Flag Register

**Name:** TIFR0
**Offset:** 0x2A
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | ICF0 | | | OCF0B | OCF0A | TOV0 |
| Access | | | R/W | | | R/W | R/W | R/W |
| Reset | | | 0 | | | 0 | 0 | 0 |

**Bit 5 – ICF0: Timer/Counter0, Input Capture Flag**
This flag is set when a capture event occurs on the ICP0 pin. When the Input Capture Register (ICR0) is set by the WGM0[3:0] to be used as the TOP value, the ICF0 Flag is set when the counter reaches the TOP value.

ICF0 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF0 can be cleared by writing a logic one to its bit location.

**Bit 2 – OCF0B: Timer/Counter0, Output Compare B Match Flag**
This flag is set in the timer clock cycle after the counter (TCNT0) value matches the Output Compare Register B (OCR0B).

Note that a Forced Output Compare (FOC0B) strobe will not set the OCF0B Flag.

OCF0B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF0B can be cleared by writing a logic one to its bit location.

**Bit 1 – OCF0A: Timer/Counter0, Output Compare A Match Flag**
This flag is set in the timer clock cycle after the counter (TCNT0) value matches the Output Compare Register A (OCR0A).

Note that a Forced Output Compare (FOC0A) strobe will not set the OCF0A Flag.

OCF0A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF0A can be cleared by writing a logic one to its bit location.

**Bit 0 – TOV0: Timer/Counter0, Overflow Flag**
The setting of this flag is dependent of the WGM0[3:0] bits setting. In Normal and CTC modes, the TOV0 Flag is set when the timer overflows. Refer to Table 17-6 for the TOV0 Flag behavior when using another WGM0[3:0] bit setting.

TOV0 is automatically cleared when the Timer/Counter0 Overflow Interrupt Vector is executed. Alternatively, TOV0 can be cleared by writing a logic one to its bit location.

### 17.14.10. General Timer/Counter Control Register

**Name:** GTCCR
**Offset:** 0x2F
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TSM | | | | | | REMAP | PSR |
| Access | R/W | | | | | | R/W | R/W |
| Reset | 0 | | | | | | 0 | 0 |

**Bit 7 – TSM: Timer/Counter Synchronization Mode**
Writing the TSM bit to '1' activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR bit is kept, hence keeping the Prescaler Reset signal asserted. This ensures that the Timer/Counter is halted and can be configured without the risk of advancing during configuration. When the TSM bit is written to '0', the PSR bit is cleared by hardware, and the Timer/Counter start counting.

**Bit 1 – REMAP**
This bit controls how the TIMER pins are mapped to pins as shown in the table:

| REMAP | TO_CLK | OC0B | OC0A | ICP0 | NOTE |
|---|---|---|---|---|---|
| **0** | PA0 | PA1 | PB1 | PB2 | DEFAULT |
| **1** | PB3 | PA5 | PA3 | PA4 | REMAPPED |

**Bit 0 – PSR: Prescaler 0 Reset Timer/Counter 0**
When this bit is one, the Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

# 18. AC - Analog Comparator

## 18.1. Overview

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO is set.

## 18.2. Features

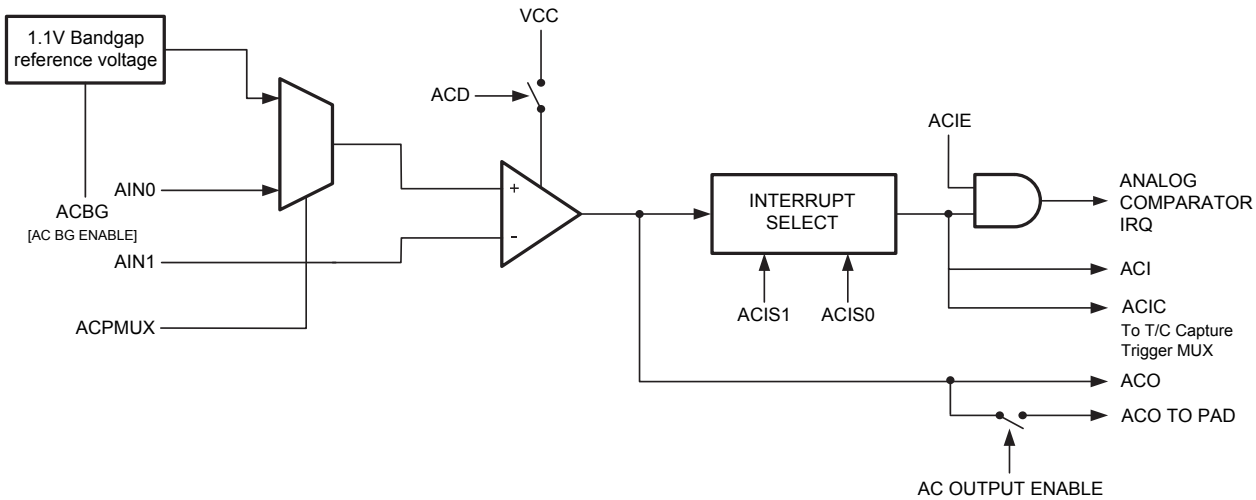
- Flexible Input Selection:
  - Internal Voltage Reference
  - Two Pins Selectable for Positive or Negative Inputs
- Interrupt Generation on:
  - Output Toggle
  - Falling Output Edge
  - Rising Output Edge

## 18.3. Block Diagram

Only one Internal reference (1.1V – Bandgap) will be connected to the positive input of the AC. For using Bandgap reference voltage as positive input to AC , it is advisable that Bandgap reference is first enabled by writing '1' to ACSRA.ACBG and then selected by writing '1' to ACSRB.ACPMUX . The output of comparator output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown below.

The Power Reduction ADC bit in the Power Reduction Register (PRR.PRADC) must be written to '0' in order to be able to use the ADC input MUX.

**Figure 18-1. Analog Comparator Block Diagram**




**Note:** Refer to the *Pin Configuration* and the I/O Ports description for Analog Comparator pin placement.

**Related Links**

Pin Configurations on page 12

## 18.4. Register Description

### 18.4.1. Analog Comparator Control and Status Register

**Name:** ACSRA
**Offset:** 0x1F
**Reset:** 0
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACD | ACBG | ACO | ACI | ACIE | ACIC | ACIS1 | ACIS0 |
| Access | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ACD: Analog Comparator Disable**
When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

**Bit 6 – ACBG: Analog Comparator Bandgap ENABLE**
When this bit is set, 1.1V bandgap reference voltage is enabled. When ACPMUX bit is also set, bandgap reference voltage is applied to the positive input of analog comparator. It is advised that bandgap reference is first enabled by writing one to ACBG bit and then selected by writing one to ACPMUX bit to allow the stabilization of voltage.

**Bit 5 – ACO: Analog Comparator Output**
The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

**Bit 4 – ACI: Analog Comparator Interrupt Flag**
This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

**Bit 3 – ACIE: Analog Comparator Interrupt Enable**
When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

**Bit 2 – ACIC: Analog Comparator Input Capture Enable**
When written logic one, this bit enables the input capture function in Timer/Counter0 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter0 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter0 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK0) must be set.

**Bits 1:0 – ACISn: Analog Comparator Interrupt Mode Select [n = 1:0]**
These bits determine which comparator events that trigger the Analog Comparator interrupt.

**Table 18-1. ACIS[1:0] Settings**

| ACIS1 | ACIS0 | Interrupt Mode |
|---|---|---|
| 0 | 0 | Comparator Interrupt on Output Toggle. |
| 0 | 1 | Reserved |
| 1 | 0 | Comparator Interrupt on Falling Output Edge. |
| 1 | 1 | Comparator Interrupt on Rising Output Edge. |

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

### 18.4.2. Analog Comparator Control and Status Register 0

**Name:** ACSRB
**Offset:** 0x1E
**Reset:** 0x00
**Property:**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | ACOE | ACPMUX |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – ACOE: Analog Comparator Output Enable**
When this bit is set, the analog comparator output is connected to the ACO pin.

**Bit 0 – ACPMUX: Analog Comparator Positive Input Multiplexer**
When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. When the bandgap reference is used as input to the Analog Comparator, it will take a certain time for the voltage to stabilize. If not stabilized, the first conversion may give a wrong value.

### 18.4.3. Digital Input Disable Register 0

When the respective bits are written to logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC[7:0] pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

**Name:** DIDR0
**Offset:** 0x17
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADC7D | ADC6D | ADC5D | ADC4D | ADC3D | ADC2D | ADC1D | ADC0D |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ADC0D, ADC1D, ADC2D, ADC3D, ADC4D, ADC5D, ADC6D, ADC7D: ADC Digital Input Disable**

- ADC:
  - When ADC0D or ADC1D is set to '1', 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 ADC[7:0] 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.
- AC:
  - When ADC0D or ADC1D is set to '1', the digital input buffer on pin AIN1 (ADC1) / AIN0 (ADC0) is disabled and the corresponding PIN register bit will read as zero. When used as an analog input but not required as a digital input the power consumption in the digital input buffer can be reduced by writing this bit to logic one.
  - DIDR0[7:2] : these bits are not applicable for AC.

# 19. ADC - Analog to Digital Converter

## 19.1. Overview

ATtiny102/ATtiny104 feature an 10-bit, successive approximation ADC. The ADC is connected to a 8/5-channel analog multiplexer for 14/8-pin device which allows 8/5 single-ended voltage inputs constructed from the pins of port A and B, internal voltage reference, analog ground, or supply voltage. The single-ended voltage inputs refer to 0V (GND).

## 19.2. Features

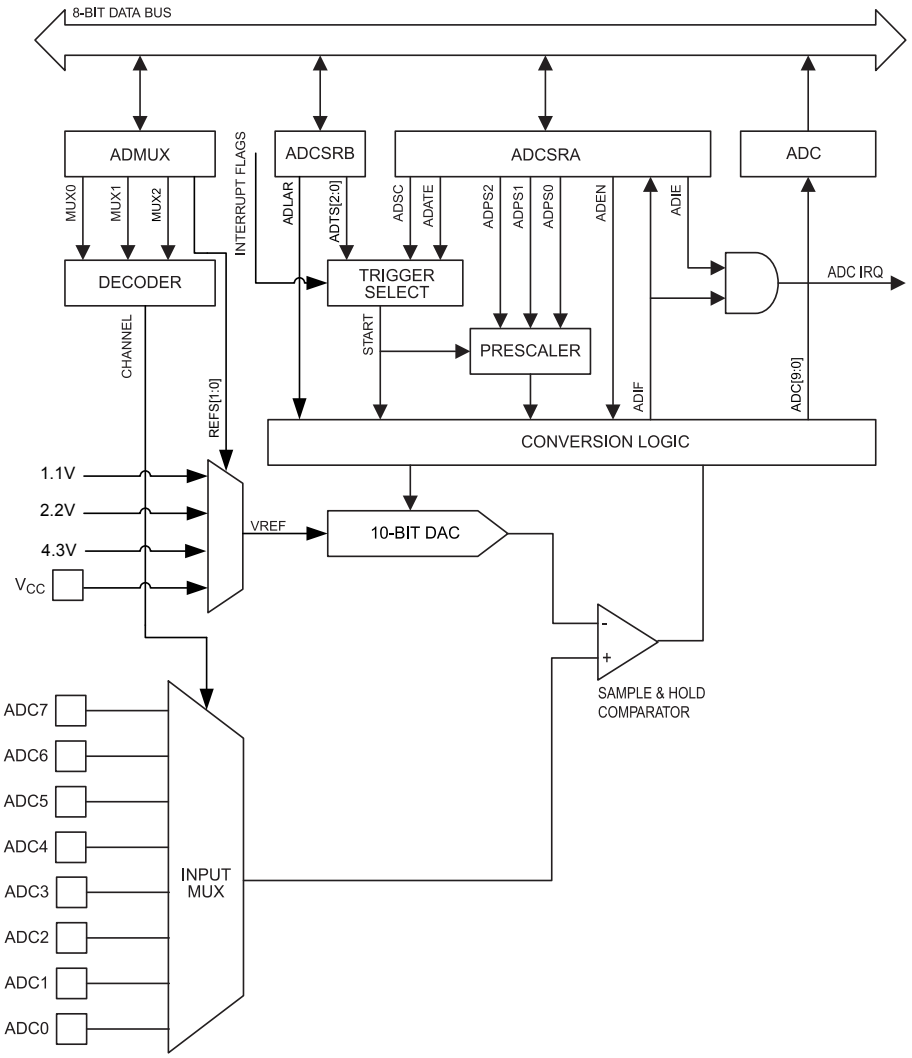
- 10-bit Resolution
- 1 LSB Integral Non-Linearity
- ±2 LSB Absolute Accuracy
- 15µs Conversion Time
- 15ksps at Maximum Resolution
- 8/5 Multiplexed Single Ended Input Channels on 14-pin/8-pin
- Optional Left Adjustment for ADC Result Readout
- Input Voltage Range: 0 - $V_{CC}$
- ADC Reference Voltages: 1.1V, 2.2V, and 4.3V
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

## 19.3. Block Diagram

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.

Internal reference voltage of nominally 1.1V, 2.2V, and 4.3V is provided on-chip. Alternatively, $V_{CC}$ can be used as reference voltage for single ended channels.

**Figure 19-1. Analog to Digital Converter Block Schematic Operation**




## 19.4. Operation

The Power Reduction ADC bit in the Power Reduction Register (PRR.PRADC) must be written to '0' in order to be enable the ADC.

The ADC is enabled by writing a '1' to the ADC Enable bit in the ADC Control and Status Register A (ADCSRA.ADEN). Voltage reference and input channel selections will not take effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC converts an analog input voltage to an 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the $V_{REF}$ voltage. The ADC voltage reference is selected by writing the ADMUX.REFS[1:0] register.

The analog input channel is selected by writing to the MUX bits in the ADC Multiplexer Selection register (ADMUX.MUX). Any of the ADC input pins can be selected as single ended inputs to the ADC.

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 ADCSRB.ADLAR.

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

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

**Related Links**

PRR on page 44
ADMUX on page 177
ADCSRA on page 178
ADCSRB on page 180
ADCL and ADCH on page 182
ADCL and ADCH on page 183

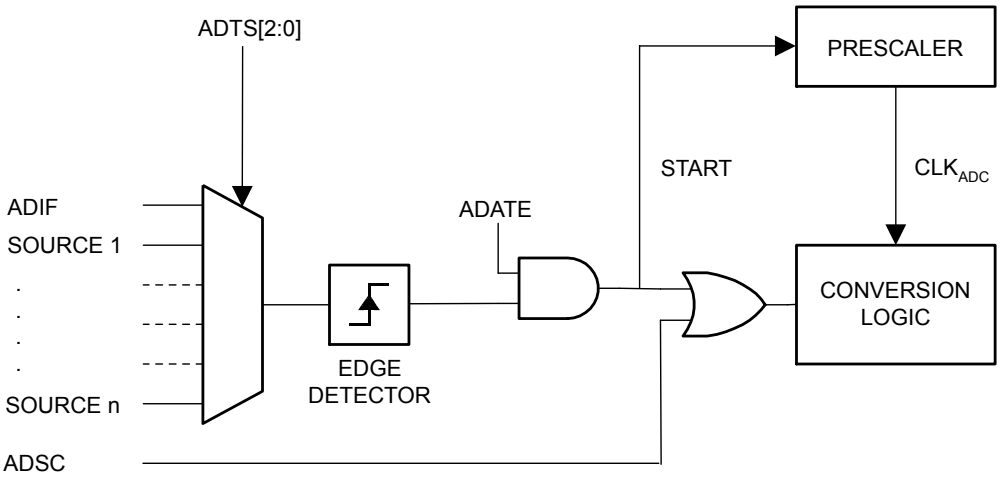
## 19.5. Starting a Conversion

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 19-2. ADC Auto Trigger Logic**




Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a '1' to ADCSRA.ADSC. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag (ADIF) is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADCSRA.ADSC to '1'. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as '1' during a conversion, independently of how the conversion was started.

**Related Links**

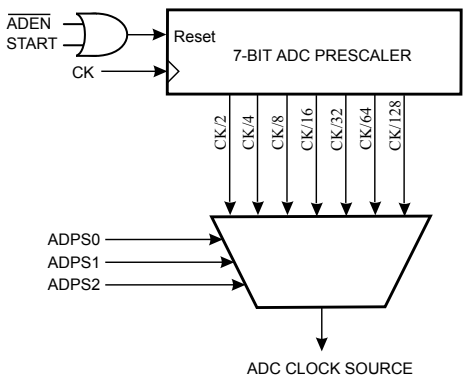
PRR on page 44

## 19.6. Prescaling and Conversion Timing

By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. 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.

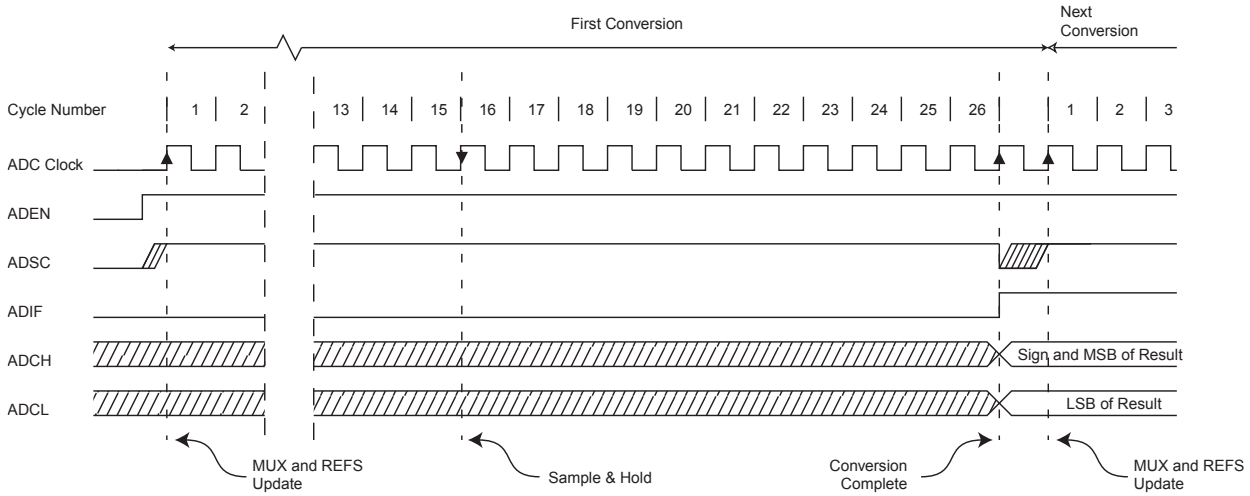
**Figure 19-3. ADC Prescaler**



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.

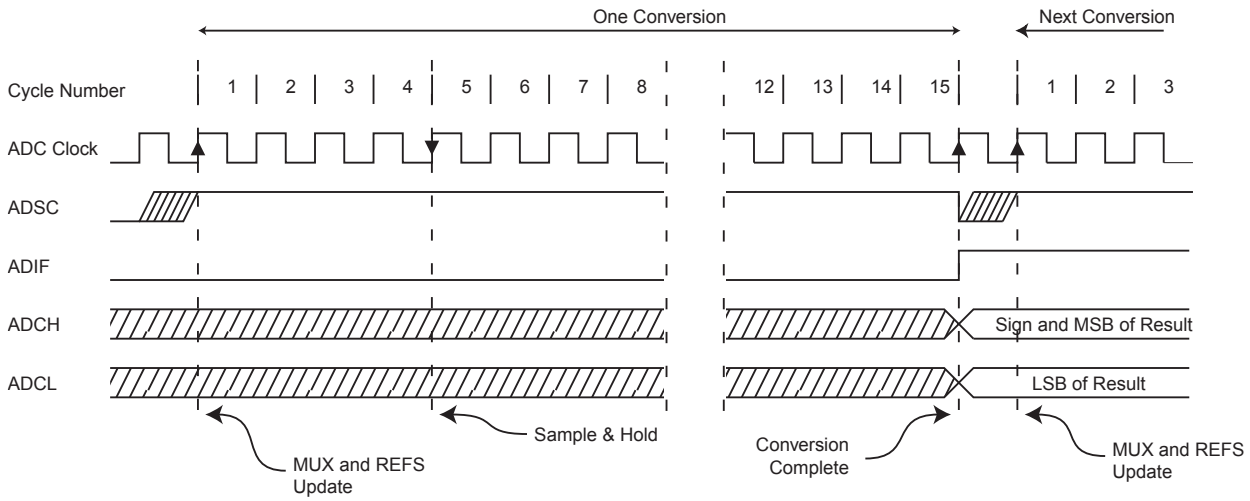
A normal conversion takes 15 ADC clock cycles. The first conversion after the ADC is switched on (i.e., ADCSRA.ADEN is written to '1') takes 26 ADC clock cycles in order to initialize the analog circuitry, as the figure below.

**Figure 19-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)**



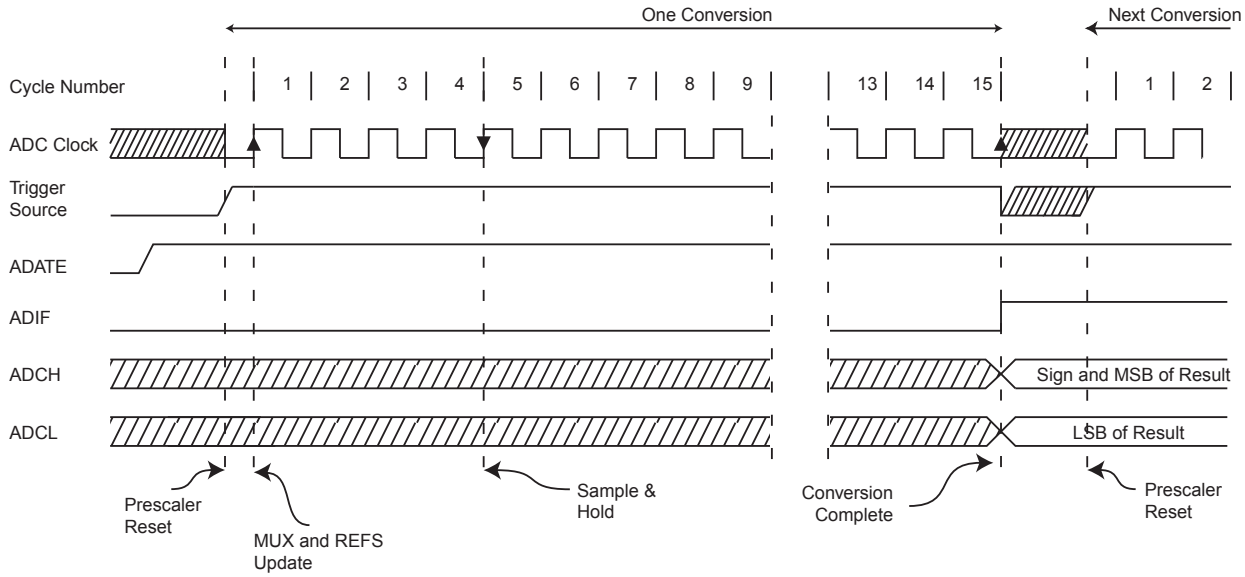

The actual sample-and-hold takes place 4 ADC clock cycles after the start of a normal conversion and 15 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC Data Registers (ADCL), and the ADC Interrupt Flag (ADCSRA.ADIF) is set. In Single 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.

**Figure 19-5. ADC Timing Diagram, Single Conversion**




When Auto Triggering is used, the prescaler is reset when the trigger event occurs, as the next figure. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place 4.5 ADC clock cycles after the rising edge on the trigger source signal. Two additional CPU clock cycles are used for synchronization logic.

**Figure 19-6. ADC Timing Diagram, Auto Triggered Conversion**




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 19-7. ADC Timing Diagram, Free Running Conversion**

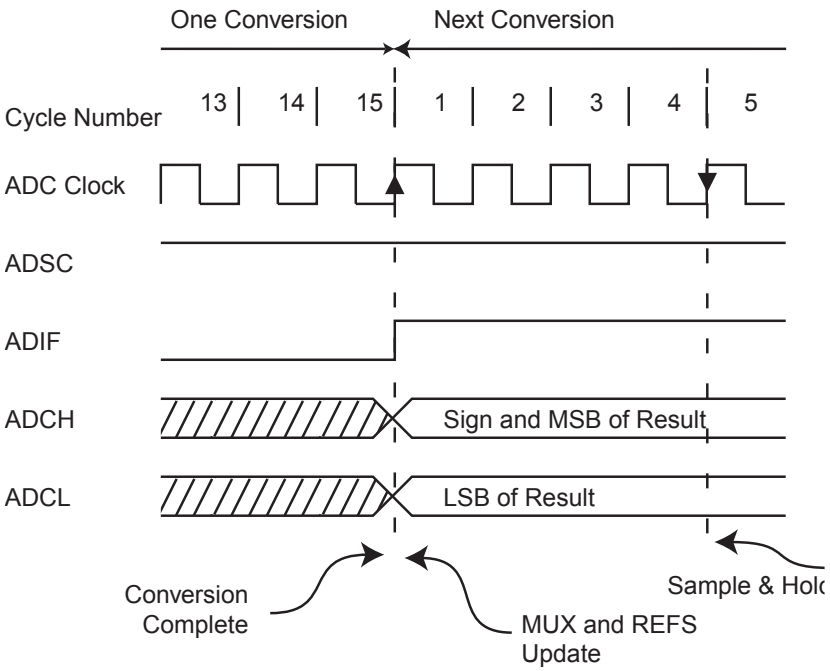



**Table 19-1. ADC Conversion Time**

| Condition | Sample & Hold (Cycles from Start of Conversion) | Conversion Time (Cycles) |
|---|---|---|
| First conversion(1) | 15 | 26 |
| Normal conversions | 4 | 15 |
| Auto Triggered conversions | 4.5 | 15.5 |
| Free Running conversion | 4 | 15 |

**Note:**

1. When gain amplifier is active, also includes the first conversion after a change in channel, reference or gain setting.

## 19.7. Changing Channel or Reference Selection

The Analog Channel Selection bits (MUX) in the ADC Multiplexer Selection Register (ADCMUX.MUX) 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 selection is continuously updated until a conversion is started. Once the conversion starts, the channel 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 (ADCRSA.ADATE) and ADC Enable bits (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.

## 19.8. ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

- In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.
- In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

The user is advised not to write new channel or reference selection values during Free Running mode.

## 19.9. ADC Voltage Reference

The reference voltage for the ADC ($V_{REF}$) indicates the conversion range, which in this case is limited to 0V ($V_{GND}$) and $V_{REF} = V_{cc}$. Single ended channels that exceed $V_{REF}$ will result in codes close to 0x3FF.

$V_{REF}$ can be selected from $V_{CC}$ or internal reference. The internal voltage reference can be set to 1.1, 2.2, or 4.3V and is generated from the internal bandgap reference ($V_{BG}$) through an internal amplifier. The first

ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

## 19.10. 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.

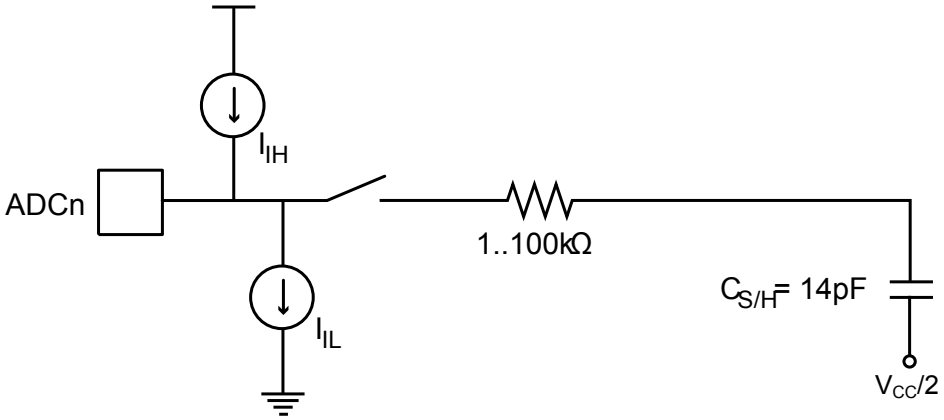
## 19.11. Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated below. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

**Figure 19-8. Analog Input Circuitry**




## 19.12. 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:

- Keep analog signal paths as short as possible.
- Make sure analog tracks run over the analog ground plane
- Keep analog tracks well away from high-speed switching digital tracks.
- If any port pin is used as a digital output, it mustn't switch while a conversion is in progress.
- Place bypass capacitors as close to $V_{CC}$ and GND pins as possible.

When high ADC accuracy is required it is recommended to use ADC Noise Reduction Mode. A good system design with properly placed, external bypass capacitors does reduce the need for using ADC Noise Reduction Mode.

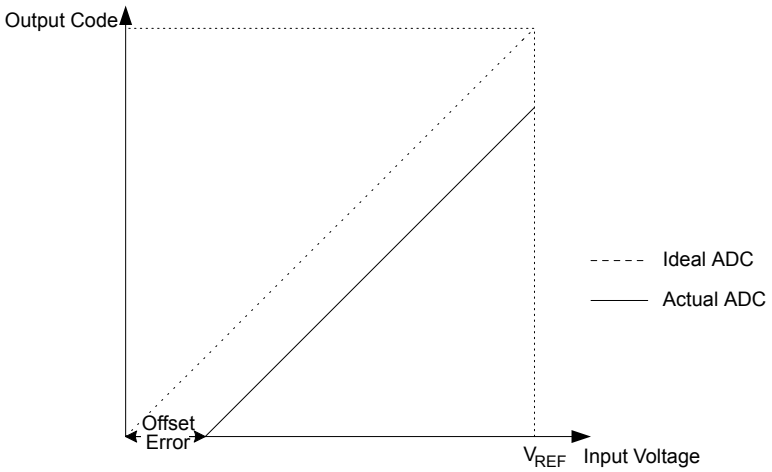
## 19.13. ADC Accuracy Definitions

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:

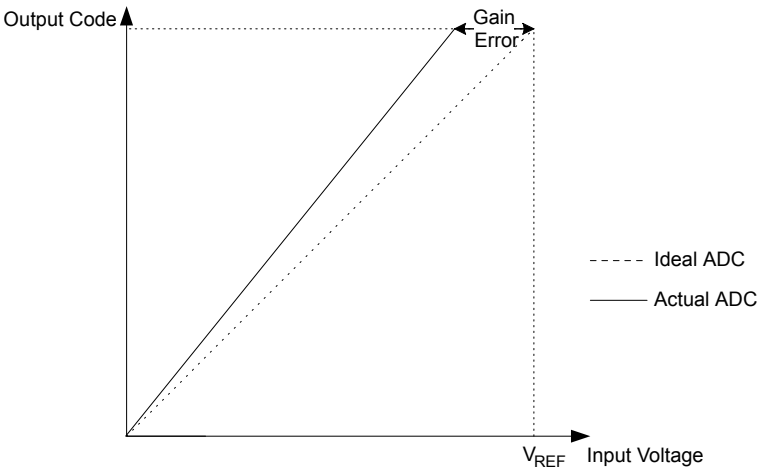
- Offset: The deviation of the first transition (0x00 to 0x01) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

**Figure 19-9. Offset Error**



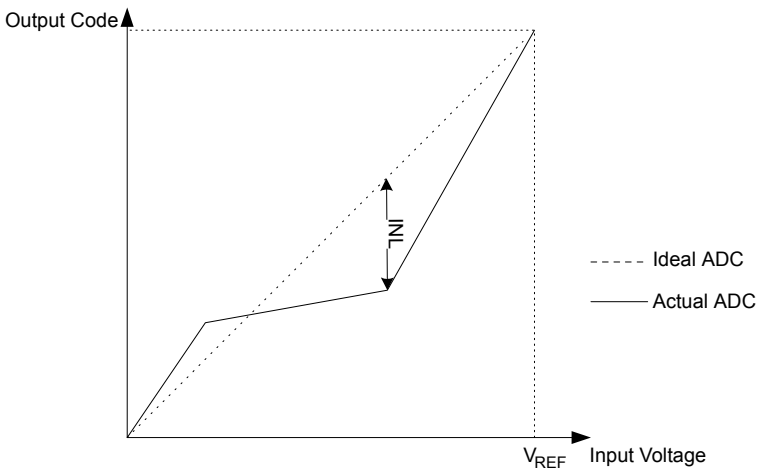

- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0xFE to 0xFF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

**Figure 19-10. Gain Error**



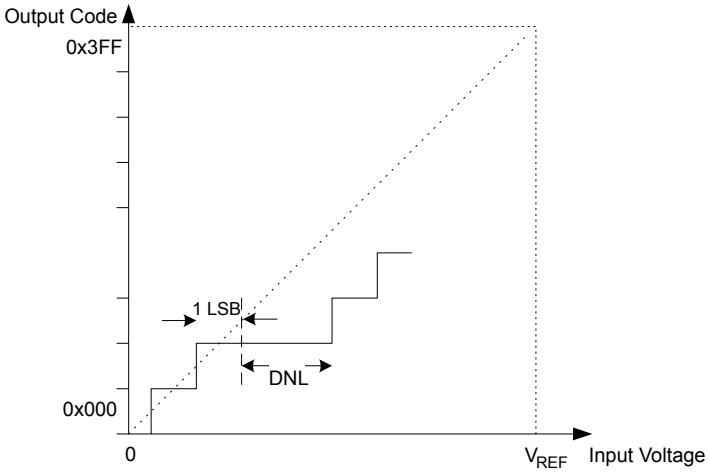

- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

**Figure 19-11. 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 19-12. 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.

## 19.14. ADC Conversion Result

After the conversion is complete (ADCSRA.ADIF is set), the conversion result can be found in the ADC Result Registers (ADCL and ADCH).

ADCL must be read first in order to avoid locking the data registers from getting updated by the next conversion result. The form of the conversion result depends on the type of conversion.

### 19.14.1. Single-Ended Conversion

For single-ended conversion, the result is as follows

$$\mathrm{ADC} = \frac{V_{\mathrm{IN}} \cdot 1024}{V_{REF}}$$

where $V_{IN}$ is the voltage on the selected input pin, and $V_{REF}$ the selected voltage reference (see also descriptions of ADMUX.MUX). 0x00 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

**Note:** 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.

## 19.15. Register Description

### 19.15.1. ADC Multiplexer Selection Register

**Name:** ADMUX
**Offset:** 0x1B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | REFS1 | REFS0 | | | | MUX2 | MUX1 | MUX0 |
| Access | R/W | R/W | | | | R/W | R/W | R/W |
| Reset | 0 | 0 | | | | 0 | 0 | 0 |

**Bits 7:6 – REFSn: Reference Selection [n = 1:0]**
These bits select the voltage reference for the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

**Table 19-3. ADC Voltage Reference Selection**

| REFS[1:0] | Voltage Reference Selection |
|---|---|
| 00 | $V_{cc}$ |
| 01 | Internal 1.1V reference |
| 10 | Internal 2.2V reference |
| 11 | Internal 4.3V reference |

**Bits 2:0 – MUXn: Analog Channel Selection [n = 2:0]**
The value of these bits selects which analog inputs are connected to the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADCSRA.ADIF is set).

**Table 19-2. Input Channel Selection**

| MUX[2:0] | Single Ended Input | Pin name |
|---|---|---|
| 000 | ADC0 | PA[0] |
| 001 | ADC1 | PA[1] |
| 010 | ADC2 | PA[5] |
| 011 | ADC3 | PA[6] |
| 100 | ADC4 | PB[0] |
| 101 | ADC5 | PB[1] |
| 110 | ADC6 | PB[2] |
| 111 | ADC7 | PB[3] |

### 19.15.2. ADC Control and Status Register A

**Name:** ADCSRA
**Offset:** 0x1D
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ADEN: ADC Enable**
Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

**Bit 6 – ADSC: ADC Start Conversion**
In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

**Bit 5 – ADATE: ADC Auto Trigger Enable**
When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

**Bit 4 – ADIF: ADC Interrupt Flag**
This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

**Bit 3 – ADIE: ADC Interrupt Enable**
When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

**Bits 2:0 – ADPSn: ADC Prescaler Select [n = 2:0]**
These bits determine the division factor between the system clock frequency and the input clock to the ADC.

**Table 19-4. 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 |

### 19.15.3. ADC Control and Status Register B

**Name:** ADCSRB
**Offset:** 0x1C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADLAR | | | | | ADTS2 | ADTS1 | ADTS0 |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – ADLAR: Left Adjustment for ADC Result Readout**
The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions.

ADLAR = 0

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1A | – | – | – | – | – | – | ADC9 | ADC8 | ADCH |
| 0x19 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADC1 | ADC0 | ADCL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

ADLAR = 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1A | ADC9 | ADC8 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADCH |
| 0x19 | ADC1 | ADC0 | – | – | – | – | – | – | ADCL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**Bits 2:0 – ADTSn: ADC Auto Trigger Source [n = 2:0]**
If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS[2:0] settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

**Table 19-5. ADC Auto Trigger Source Selection**

| ADTS[2:0] | Trigger Source |
|---|---|
| 000 | Free Running mode |
| 001 | Analog Comparator |
| 010 | External Interrupt Request 0 |

| ADTS[2:0] | Trigger Source |
|---|---|
| 011 | Timer/Counter 0 Compare Match A |
| 100 | Timer/Counter 0 Overflow |
| 101 | Timer/Counter 0 Compare Match B |
| 110 | Pin Change Interrupt 0 Request |
| 111 | Timer/Counter 0 Capture Event |

### 19.15.4. ADC Data Register Low and High Byte (ADLAR=0)

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

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

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

The ADLAR bit and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set (ADLAR=1), the result is left adjusted. If ADLAR is cleared (ADLAR=0 which is the default value), the result is right adjusted.

**Name:** ADCL and ADCH
**Offset:** 0x19
**Reset:** 0x00
**Property:** ADLAR = 0

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | ADC9 | ADC8 |
| Access | | | | | | | R | R |
| Reset | | | | | | | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADC1 | ADC0 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – ADCn: ADC Conversion Result**
These bits represent the result from the conversion. Refer to ADC Conversion Result for details.

### 19.15.5. ADC Data Register Low and High Byte (ADLAR=1)

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

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

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

The ADLAR bit and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set (ADLAR=1), the result is left adjusted. If ADLAR is cleared (ADLAR=0 which is the default value), the result is right adjusted.

**Name:** ADCL and ADCH
**Offset:** 0x19
**Reset:** 0x00
**Property:** ADLAR = 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ADC9 | ADC8 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 |
| 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 |
|---|---|---|---|---|---|---|---|---|
| | ADC1 | ADC0 | | | | | | |
| Access | R | R | | | | | | |
| Reset | 0 | 0 | | | | | | |

**Bits 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 – ADCn: ADC Conversion Result**
These bits represent the result from the conversion. Refer to ADC Conversion Result for details.

### 19.15.6. Digital Input Disable Register 0

When the respective bits are written to logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC[7:0] pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

**Name:** DIDR0
**Offset:** 0x17
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADC7D | ADC6D | ADC5D | ADC4D | ADC3D | ADC2D | ADC1D | ADC0D |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ADC0D, ADC1D, ADC2D, ADC3D, ADC4D, ADC5D, ADC6D, ADC7D: ADC Digital Input Disable**

- ADC:
  - When ADC0D or ADC1D is set to '1', 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 ADC[7:0] 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.
- AC:
  - When ADC0D or ADC1D is set to '1', the digital input buffer on pin AIN1 (ADC1) / AIN0 (ADC0) is disabled and the corresponding PIN register bit will read as zero. When used as an analog input but not required as a digital input the power consumption in the digital input buffer can be reduced by writing this bit to logic one.
  - DIDR0[7:2] : these bits are not applicable for AC.

# 20. MEMPROG- Memory Programming

## 20.1. Overview

The Non-Volatile Memory (NVM) Controller manages all access to the Non-Volatile Memories. The NVM Controller controls all NVM timing and access privileges, and holds the status of the NVM.

During normal execution the CPU will execute code from the code section of the Flash memory (program memory). When entering sleep and no programming operations are active, the Flash memory is disabled to minimize power consumption.

All NVM are mapped to the data memory. Application software can read the NVM from the mapped locations of data memory using load instruction with indirect addressing.

The NVM has only one read port and, therefore, the next instruction and the data can not be read simultaneously. When the application reads data from NVM locations mapped to the data space, the data is read first before the next instruction is fetched. The CPU execution is here delayed by one system clock cycle.

Internal programming (self-programming) operations to NVM have been disabled and the NVM therefore appears to the application software as read-only. Internal write or erase operations of the NVM will not be successful.

The method used by the external programmer for writing the Non-Volatile Memories is referred to as external programming. External programming can be done both in-system or in mass production. The external programmer can read and program the NVM via the Tiny Programming Interface (TPI).

In the external programming mode all NVM can be read and programmed, except the signature and the calibration sections which are read-only.

NVM can be programmed between 1.8-5.5V.

## 20.2. Features

- Two Embedded Non-Volatile Memories:
  - Non-Volatile Memory Lock bits (NVM Lock bits)
  - Flash Memory
- Four Separate Sections Inside Flash Memory:
  - Code Section (Program Memory)
  - Signature Section
  - Configuration Section
  - Calibration Section
- Read Access to All Non-Volatile Memories from Application Software
- Read and Write Access to Non-Volatile Memories from External programmer:
  - Read Access to All Non-Volatile Memories
  - Write Access to NVM Lock Bits, Flash Code Section and Flash Configuration Section
- External Programming:
  - Support for In-System and Mass Production Programming
  - Programming Through the Tiny Programming Interface (TPI)
- High Security with NVM Lock Bits

- Self-Programming Flash on Full Operating Voltage Range (1.8 – 5.5V)

## 20.3. Non-Volatile Memories (NVM)

The device has the following, embedded NVM:

- Non-Volatile Memory Lock Bits
- Flash memory with four separate sections
- 1KB Flash Memory
  - CPU execution will be halted while doing external programming
- Extra rows
  - Flash - Unique ID needs to be added

### 20.3.1. Non-Volatile Memory Lock Bits

The device provides two Lock Bits.

**Table 20-1. Lock Bit Byte**

| Lock Bit Byte | Bit No. | Description | Default Value |
|---|---|---|---|
| | 7 | | 1 (unprogrammed) |
| | 6 | | 1 (unprogrammed) |
| | 5 | | 1 (unprogrammed) |
| | 4 | | 1 (unprogrammed) |
| | 3 | | 1 (unprogrammed) |
| | 2 | | 1 (unprogrammed) |
| NVLB2 | 1 | Non-Volatile Lock Bit | 1 (unprogrammed) |
| NVLB1 | 0 | Non-Volatile Lock Bit | 1 (unprogrammed) |

The Lock Bits can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional security. Lock Bits can be erased to "1" with the Chip Erase command, only.

**Table 20-2. Lock Bit Protection Modes**

| Memory Lock Bits(1) | | | Protection Type |
|---|---|---|---|
| LB Mode | NVLB2(2) | NVLB1(2) | |
| 1 | 1 | 1 | No memory lock features enabled. |
| 2 | 1 | 0 | Further Programming of the Flash memory is disabled in the external programming mode. The configuration section bits are locked in the external programming mode |
| 3 | 0 | 0 | Further programming and verification of the flash is disabled in the external programming mode. The configuration section bits are locked in the external programming mode |

**Note:**

1. Program the configuration section bits before programming NVLB1 and NVLB2.
2. "1" means unprogrammed, "0" means programmed

### 20.3.2. Flash Memory

The embedded Flash memory has four separate sections.

**Table 20-3. Number of Words and Pages in the Flash**

| Section | Size (Bytes) | Page Size (Words) | Pages | WADDR | PADDR |
|---|---|---|---|---|---|
| Code (program memory) | 1024 | 8 | 64 | [3:1] | [9:4] |
| Configuration | 8 | 8 | 1 | [3:1] | - |
| Signature (1) | 16 | 8 | 2 | [3:1] | [4:4] |
| Calibration(1) | 8 | 8 | 1 | [3:1] | - |

**Note:**

1. This section is read-only.

### 20.3.3. Configuration Section

ATtiny102/ATtiny104 have one configuration byte, which resides in the configuration section.

**Table 20-4. Configuration bytes**

| Configuration byte | Offset Address | Configuration word data |
|---|---|---|
| CONFW0 | 0x04 | Configuration word (fuse values- RSTDISBL, WDTON, CKOUT, SELFPROGEN) |

The next table briefly describes the functionality of all configuration bits and how they are mapped into the configuration byte.

**Table 20-5. Configuration Byte 0**

| Bit | Fuse values | Description | Default Value |
|---|---|---|---|
| 7:4 | – | Reserved | 1 (unprogrammed) |
| 3 | SELFPROGEN | Self-Programming | 1 (unprogrammed) |
| 2 | CKOUT | System Clock Output | 1 (unprogrammed) |
| 1 | WDTON | Watchdog Timer always on | 1 (unprogrammed) |
| 0 | RSTDISBL | External Reset disable | 1 (unprogrammed) |

Configuration bits are not affected by a chip erase but they can be cleared using the configuration section erase command (see *Erasing the Configuration Section* in this chapter). Note that configuration bits are locked if Non- Volatile Lock Bit 1 (NVLB1) is programmed.

#### 20.3.3.1. Latching of Configuration Bits

All configuration bits are latched either when the device is reset or when the device exits the external programming mode. Changes to configuration bit values have no effect until the device leaves the external programming mode.

### 20.3.4. Signature Section

The signature section is a dedicated memory area used for storing miscellaneous device information, such as the device signature. Most of this memory section is reserved for internal use.

ATtiny102/ATtiny104 have a three-byte signature code, which can be used to identify the device. The three bytes reside in the signature section, as shown in the above table. The signature data for ATtiny102/ATtiny104 is given in the next table.

**Table 20-6. Signature bytes**

| Signature word address | Configuration word data | |
|---|---|---|
| | High byte | Low byte |
| 0x00 | Device ID 1 | Manufacturer ID |
| 0x01 | Reserved for internal use | Device ID 2 |
| 0x02 | Reserved for internal use | |
| 0x03 ... 0x07 | Serial number | |

**Table 20-7. Signature codes**

| Part | Signature Bytes | | |
|---|---|---|---|
| | Manufacturer ID | Device ID 1 | Device ID 2 |
| ATtiny102 | 0x1E | 0x90 | 0x0C |
| ATtiny104 | 0x1E | 0x90 | 0x0B |

### 20.3.4.1. Signature Row Summary

| Offset | Name | Bit Pos. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | SIGROW_DEVICEID0 | 7:0 | DEVICEID[7:0] | | | | | | | | |
| 0x01 | SIGROW_DEVICEID1 | 7:0 | DEVICEID[7:0] | | | | | | | | |
| 0x02 | SIGROW_DEVICEID2 | 7:0 | DEVICEID[7:0] | | | | | | | | |
| 0x03<br>...<br>0x05 | Reserved | | | | | | | | | | |
| 0x06 | SIGROW_SERNUM0 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x07 | SIGROW_SERNUM1 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x08 | SIGROW_SERNUM2 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x09 | SIGROW_SERNUM3 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x0A | SIGROW_SERNUM4 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x0B | SIGROW_SERNUM5 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x0C | SIGROW_SERNUM6 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x0D | SIGROW_SERNUM7 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x0E | SIGROW_SERNUM8 | 7:0 | SERNUM[7:0] | | | | | | | | |
| 0x0F | SIGROW_SERNUM9 | 7:0 | SERNUM[7:0] | | | | | | | | |

**Device ID n**

**Name:** SIGROW_DEVICEIDn
**Offset:** 0x00 + n*0x01 [n=0..2]
**Reset:** [Device ID]
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DEVICEID[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – DEVICEID[7:0]: Byte n of the Device ID**

**Serial Number Byte n**

**Name:** SIGROW_SERNUMn
**Offset:** 0x06 + n*0x01 [n=0..9]
**Reset:** [device serial number]
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SERNUM[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – SERNUM[7:0]: Serial Number n [n=0..9]**
Each device has an individual serial number, representing a unique ID. This can be used to identify a specific device in the field. The serial number consists of ten bytes..

### 20.3.5. Calibration Section

ATtiny102/ATtiny104 have one calibration byte. The calibration byte contains the calibration data for the internal oscillator and resides in the calibration section. During reset, the calibration byte is automatically written into the OSCCAL register to ensure correct frequency of the calibrated internal oscillator.

**Table 20-8. Calibration byte**

| Calibration byte | Offset Address | Configuration word data | |
|---|---|---|---|
| | | High byte [BYTE1] | Low byte [BYTE0] |
| OSCCAL | 0x00 | Reserved | Oscillator calibration value |
| Reserved | 0x01 ... 0x07 | Reserved | Reserved |

## 20.4. Accessing the NVM

NVM lock bits, and all Flash memory sections are mapped to the data space as shown in *Data Memory*. The NVM can be accessed for read and programming via the locations mapped in the data space.

The NVM Controller recognizes a set of commands that can be used to instruct the controller what type of programming task to perform on the NVM. Commands to the NVM Controller are issued via the NVM Command Register. See *NVMCMD - Non-Volatile Memory Command Register*. After the selected command has been loaded, the operation is started by writing data to the NVM locations mapped to the data space.

When the NVM Controller is busy performing an operation it will signal this via the NVM Busy Flag in the NVM Control and Status Register. See *NVMCSR - Non-Volatile Memory Control and Status Register*. The NVM Command Register is blocked for write access as long as the busy flag is active. This is to ensure that the current command is fully executed before a new command can start.

Programming any part of the NVM will automatically inhibit the following operations:

- All programming to any other part of the NVM
- All reading from any NVM location

ATtiny102/ATtiny104 supports external programming and internal programming (self-programming).

**Related Links**

SRAM Data Memory on page 27
NVMCSR on page 196
NVMCMD on page 197

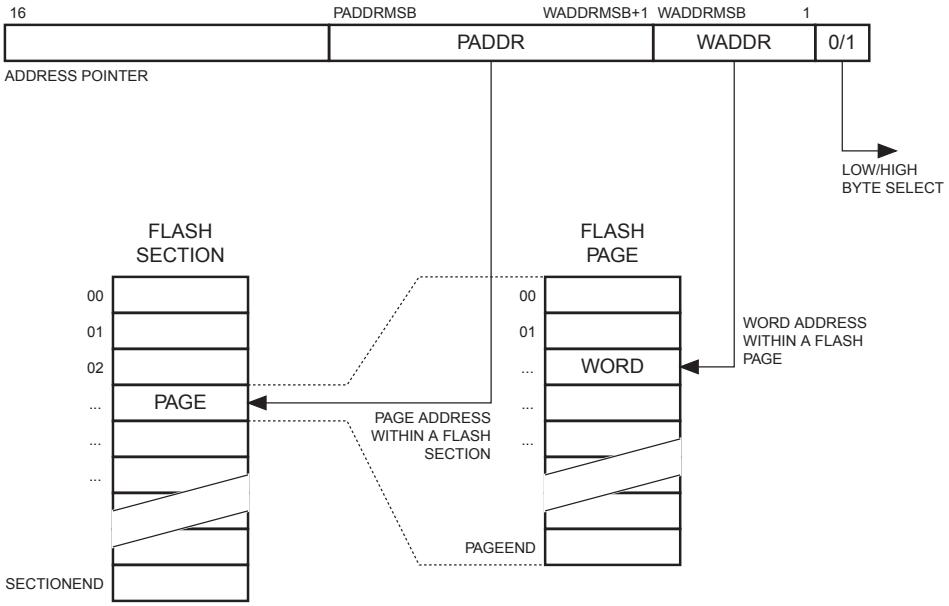
### 20.4.1. Addressing the Flash

The data space uses byte accessing but since the Flash sections are accessed as words and organized in pages, the byte-address of the data space must be converted to the word-address of the Flash section.

The most significant bits of the data space address select the NVM Lock bits or the Flash section mapped to the data memory. The word address within a page (WADDR) is held by bits [WADDRMSB:1], and the page address (PADDR) by bits [PADDRMSB:WADDRMSB+1]. Together, PADDR and WADDR form the absolute address of a word in the Flash section.

The least significant bit of the Flash section address is used to select the low or high byte of the word.

**Figure 20-1. Addressing the Flash Memory**




### 20.4.2. Reading the Flash

The Flash can be read from the data memory mapped locations one byte at a time. For read operations, the least significant bit (bit 0) is used to select the low or high byte in the word address. If this bit is zero, the low byte is read, and if it is one, the high byte is read.

### 20.4.3. Programming the Flash

The Flash can be written word-by-word. Before writing a Flash word, the Flash target location must be erased. Writing to an un-erased Flash word will corrupt its content.

The Flash is word-accessed for writing, and the data space uses byte-addressing to access Flash that has been mapped to data memory. It is therefore important to write the word in the correct order to the Flash, namely low bytes before high bytes. First, the low byte is written to the temporary buffer. Then, writing the high byte latches both the high byte and the low byte into the Flash word buffer, starting the write operation to Flash.

The Flash erase operations can only performed for the entire Flash sections.

The Flash programming sequence is as follows:

1. Perform a Flash section erase or perform a Chip erase

2. Write the Flash section word by word

#### 20.4.3.1. Chip Erase

The Chip Erase command will erase the entire code section of the Flash memory and the NVM Lock Bits. For security reasons, the NVM Lock Bits are not reset before the code section has been completely erased. Configuration, Signature and Calibration sections are not changed.

Before starting the Chip erase, the NVMCMD register must be loaded with the CHIP_ERASE command. To start the erase operation a dummy byte must be written into the high byte of a word location that resides inside the Flash code section. The NVMBSY bit remains set until erasing has been completed. While the Flash is being erased neither Flash buffer loading nor Flash reading can be performed.

The Chip Erase can be carried out as follows:

1. Write the 0x10 (CHIP_ERASE) to the NVMCMD register
2. Start the erase operation by writing a dummy byte to the high byte of any word location inside the code section
3. Wait until the NVMBSY bit has been cleared

#### 20.4.3.2. Erasing the Code Section

The algorithm for erasing all pages of the Flash code section is as follows:

1. Write the 0x14 (SECTION_ERASE) to the NVMCMD register
2. Start the erase operation by writing a dummy byte to the high byte of any word location inside the code section
3. Wait until the NVMBSY bit has been cleared

#### 20.4.3.3. Writing a Code Word

The algorithm for writing a word to the code section is as follows:

1. Write the 0x1D (WORD_WRITE) to the NVMCMD register
2. Write the low byte of the data into the low byte of a word location
3. Write the high byte of the data into the high byte of the same word location. This will start the Flash write operation
4. Wait until the NVMBSY bit has been cleared

#### 20.4.3.4. Erasing the Configuration Section

The algorithm for erasing the Configuration section is as follows:

1. Write the 0x14 (SECTION_ERASE) to the NVMCMD register
2. Start the erase operation by writing a dummy byte to the high byte of any word location inside the configuration section
3. Wait until the NVMBSY bit has been cleared

#### 20.4.3.5. Writing a Configuration Word

The algorithm for writing a Configuration word is as follows:

1. Write the 0x1D (WORD_WRITE) to the NVMCMD register
2. Write the low byte of the data to the low byte of a configuration word location
3. Write the high byte of the data to the high byte of the same configuration word location. This will start the Flash write operation.
4. Wait until the NVMBSY bit has been cleared

### 20.4.4. Reading NVM Lock Bits

The Non-Volatile Memory Lock Byte can be read from the mapped location in data memory.

### 20.4.5. Writing NVM Lock Bits

The algorithm for writing the Lock bits is as follows:

1. Write the WORD_WRITE command to the NVMCMD register.
2. Write the lock bits value to the Non-Volatile Memory Lock Byte location. This is the low byte of the Non- Volatile Memory Lock Word.
3. Start the NVM Lock Bit write operation by writing a dummy byte to the high byte of the NVM Lock Word location.
4. Wait until the NVMBSY bit has been cleared.

## 20.5. Self programming

The Flash in Tiny104 does not support Read-While-Write, and cannot be read during an erase or write operation. Therefore, the CPU will halt execution.

The device provides a Self-Programming mechanism for downloading and uploading program code by the MCU itself. Only WORD_WRITE and PAGE_ERASE commands are supported in self-programming. The CPU can execute Page Erase and Word Write in the NVM code memory section to perform programming operations.

**Note:** The user needs to add two NOP operations after the ST operation that triggers the self-programming to ensure correct CPU operation.

**Table 20-9. Example code for self-programming:**

| Assembly Code Example |
|---|
| The sequence for entering self-programming mode is given below (r19 can be any register): |

```
;  set CCP
ldi             r19, 0xe7
out           CCP, r19
```

The software then has to perform the desired self-programming operation within 4 clock cycles. Example of the complete code to perform page erase:

```
;  erase page
;  set the page address pointer
ldi             ZL, 0xE1
ldi             ZH, 0x43
;  set NVMCMD to page erase
ldi             temp, 0b011000
out           NVMCMD, temp
;  set CCP to enter program mode
ldi             r19, 0xe7
out           CCP, r19
; trigger the erase operation (within four clock cycles)
ldi             temp, 0x00
st             Z+, temp
; required for proper CPU halting
nop
nop
```

## 20.6. External Programming

The method for programming the Non-Volatile Memories by means of an external programmer is referred to as external programming. External programming can be done both in-system or in mass production.

The Non-Volatile Memories can be externally programmed via the Tiny Programming Interface (TPI). For details on the TPI, see *Programming interface*. Using the TPI, the external programmer can access the NVM control and status registers mapped to I/O space and the NVM memory mapped to data memory space.

**Related Links**

TPI-Tiny Programming Interface on page 198

### 20.6.1. Entering External Programming Mode

The TPI must be enabled before external programming mode can be entered. The following procedure describes, how to enter the external programming mode after the TPI has been enabled:

1. Make a request for enabling NVM programming by sending the NVM memory access key with the SKEY instruction.
2. Poll the status of the NVMEN bit in TPISR until it has been set.

Refer to the *Programming Interface* description for more detailed information of enabling the TPI and programming the NVM.

**Related Links**

TPI-Tiny Programming Interface on page 198

### 20.6.2. Exiting External Programming Mode

Clear the NVM enable bit to disable NVM programming, then release the $\overline{\text{RESET}}$ pin.

See NVMEN bit in *TPISR – Tiny Programming Interface Status Register*.

**Related Links**

TPISR on page 209

## 20.7. Register Description

### 20.7.1. Non-Volatile Memory Control and Status Register

**Name:** NVMCSR
**Offset:** 0x32
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMBSY | | | | | | | |
| Access | R/W | | | | | | | |
| Reset | 0 | | | | | | | |

**Bit 7 – NVMBSY: Non-Volatile Memory Busy**
This bit indicates the NVM memory (Flash memory and Lock Bits) is busy, being programmed. This bit is set when a program operation is started, and it remains set until the operation has been completed.

### 20.7.2. Non-Volatile Memory Command Register

**Name:** NVMCMD
**Offset:** 0x33
**Reset:** 0x00
**Property:** -

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

**Bits 5:0 – NVMCMD[5:0]: Non-Volatile Memory Command**
These bits define the programming commands for the flash.

**Table 20-10. NVM Programming commands**

| Operation Type | NVMCMD | | Mnemonic | Description |
|---|---|---|---|---|
| | Binary | Hex | | |
| General | 0b000000 | 0x00 | NO_OPERATION | No operation |
| | 0b010000 | 0x10 | CHIP_ERASE | Chip erase(1) |
| | 0b010001 | 0x11 | CHIP_WRITE | Write chip(2) |
| Section | 0b010100 | 0x14 | SECTION_ERASE | Section erase |
| Word | 0b011101 | 0x1D | WORD_WRITE | Word write |
| Page | 0b011000 | 0x18 | PAGE_ERASE | Erase page |

**Note:**
1. Erase the Code section and the Non-Volatile Memory lock bits.
2. Write the Code section, but doesn't affect the Non-Volatile Memory lock bits. Self-programming supports NO_OPERATION, WORD_WRITE, and PAGE_ERASE

# 21. TPI-Tiny Programming Interface

## 21.1. Overview

The Tiny Programming Interface (TPI) supports external programming of all Non-Volatile Memories (NVM). Memory programming is done via the NVM Controller, by executing NVM controller commands as described in *Memory Programming*.

**Related Links**

MEMPROG- Memory Programming on page 185

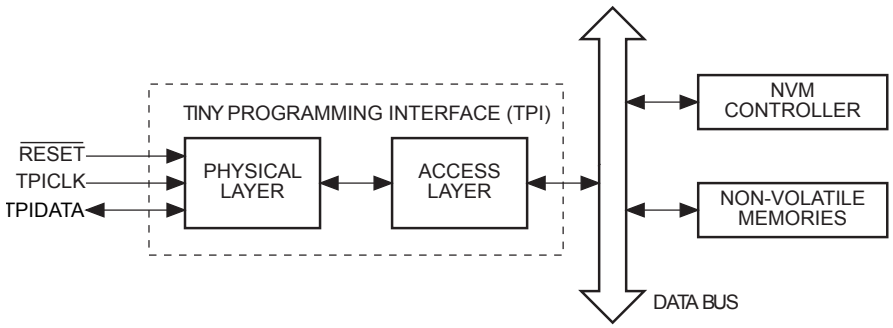
## 21.2. Features

- Physical Layer:
  - Synchronous Data Transfer
  - Bi-directional, Half-duplex Receiver And Transmitter
  - Fixed Frame Format With One Start Bit, 8 Data Bits, One Parity Bit And 2 Stop Bits
  - Parity Error Detection, Frame Error Detection And Break Character Detection
  - Parity Generation And Collision Detection – Automatic Guard Time Insertion Between Data Reception And Transmission
- Access Layer:
  - Communication Based On Messages
  - Automatic Exception Handling Mechanism
  - Compact Instruction Set
  - NVM Programming Access Control
  - Tiny Programming Interface Control And Status Space Access Control
  - Data Space Access Control

## 21.3. Block Diagram

The Tiny Programming Interface (TPI) provides access to the programming facilities. The interface consists of two layers: the access layer and the physical layer.

**Figure 21-1. The Tiny Programming Interface and Related Internal Interfaces**




Programming is done via the physical interface. This is a 3-pin interface, which uses the $\overline{\text{RESET}}$ pin as enable, the TPICLK pin as the clock input, and the TPIDATA pin as data input and output. NVM can be programmed between 1.8-5.5V.

## 21.4. Physical Layer of Tiny Programming Interface

The TPI physical layer handles the basic low-level serial communication. The TPI physical layer uses a bi-directional, half-duplex serial receiver and transmitter. The physical layer includes serial-to-parallel and parallel-to-serial data conversion, start-of-frame detection, frame error detection, parity error detection, parity generation and collision detection.

The TPI is accessed via three pins, as follows:

- $\overline{\text{RESET}}$: Tiny Programming Interface enable input
- TPICLK: Tiny Programming Interface clock input
- TPIDATA: Tiny Programming Interface data input/output

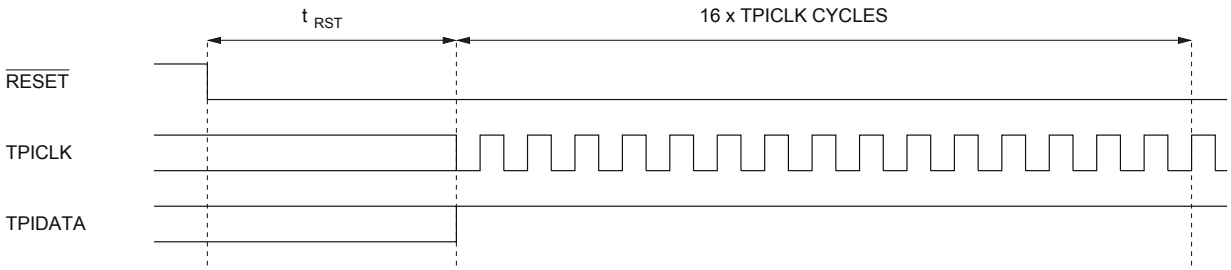
In addition, the $V_{CC}$ and GND pins must be connected between the external programmer and the device.

### 21.4.1. Enabling

The following sequence enables the Tiny Programming Interface:

- Apply 5V between $V_{CC}$ and GND
- Depending on the method of reset to be used:
  - Either: wait $t_{TOUT}$ (see *System and Reset Characteristics*) and then set the $\overline{\text{RESET}}$ pin low. This will reset the device and enable the TPI physical layer. The $\overline{\text{RESET}}$ pin must then be kept low for the entire programming session
  - Or: if the RSTDISBL configuration bit has been programmed, apply 12V to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin must be kept at 12V for the entire programming session
- Wait $t_{RST}$ (see *System and Reset Characteristics* )
- Keep the TPIDATA pin high for 16 TPICLK cycles

**Figure 21-2. Sequence for enabling the Tiny Programming Interface**




**Related Links**

System and Reset Characteristics on page 214

### 21.4.2. Disabling

Provided that the NVM enable bit has been cleared, the TPI is automatically disabled if the $\overline{\text{RESET}}$ pin is released to inactive high state or, alternatively, if VHV is no longer applied to the $\overline{\text{RESET}}$ pin.

If the NVM enable bit is not cleared a power down is required to exit TPI programming mode. See NVMEN bit in *TPISR – Tiny Programming Interface Status Register*.

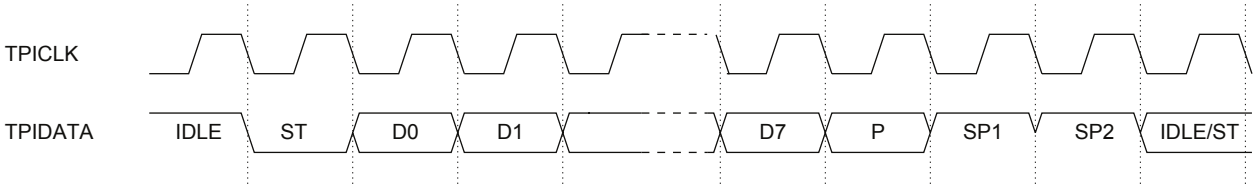
**Related Links**

TPISR on page 209

### 21.4.3. Frame Format

The TPI physical layer supports a fixed frame format. A frame consists of one character, eight bits in length, and one start bit, a parity bit and two stop bits. Data is transferred with the least significant bit first.

**Figure 21-3. Serial frame format.**




Symbols used in the above figure:

- ST: Start bit (always low)
- D0-D7: Data bits (least significant bit sent first)
- P: Parity bit (using even parity)
- SP1: Stop bit 1 (always high)
- SP2: Stop bit 2 (always high)

### 21.4.4. Parity Bit Calculation

The parity bit is always calculated using even parity. The value of the bit is calculated by doing an exclusive-or of all the data bits, as follows:

$$P = D0 \otimes D1 \otimes D2 \otimes D3 \otimes D4 \otimes D5 \otimes D6 \otimes D7 \otimes 0$$

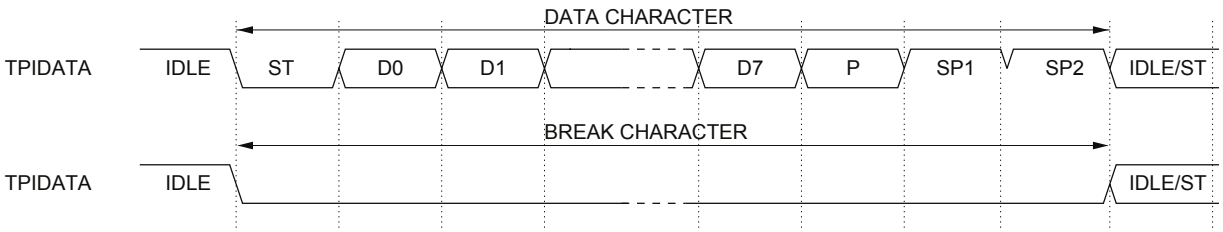
where:

- P: Parity bit using even parity
- D0-D7: Data bits of the character

### 21.4.5. Supported Characters

The BREAK character is equal to a 12 bit long low level. It can be extended beyond a bit-length of 12.

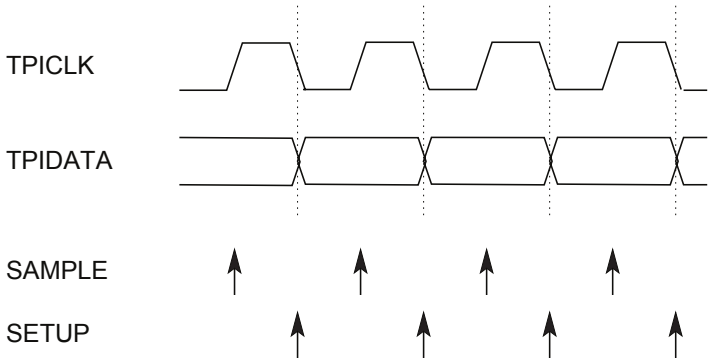
**Figure 21-4. Supported characters.**




### 21.4.6. Operation

The TPI physical layer operates synchronously on the TPICLK provided by the external programmer. The dependency between the clock edges and data sampling or data change is shown in the figure below. Data is changed at falling edges and sampled at rising edges.

**Figure 21-5. Data changing and Data sampling.**




The TPI physical layer supports two modes of operation: Transmit and Receive. By default, the layer is in Receive mode, waiting for a start bit. The mode of operation is controlled by the access layer.

### 21.4.7. Serial Data Reception

When the TPI physical layer is in receive mode, data reception is started as soon as a start bit has been detected. Each bit that follows the start bit will be sampled at the rising edge of the TPICLK and shifted into the shift register until the second stop bit has been received. When the complete frame is present in the shift register the received data will be available for the TPI access layer.

There are three possible exceptions in the receive mode: frame error, parity error and break detection. All these exceptions are signalized to the TPI access layer, which then enters the error state and puts the TPI physical layer into receive mode, waiting for a BREAK character.

- Frame Error Exception. The frame error exception indicates the state of the stop bit. The frame error exception is set if the stop bit was read as zero.
- Parity Error Exception. The parity of the data bits is calculated during the frame reception. After the frame is received completely, the result is compared with the parity bit of the frame. If the comparison fails the parity error exception is signalized.
- Break Detection Exception. The Break detection exception is given when a complete frame of all zeros has been received.

### 21.4.8. Serial Data Transmission

When the TPI physical layer is ready to send a new frame it initiates data transmission by loading the shift register with the data to be transmitted. When the shift register has been loaded with new data, the transmitter shifts one complete frame out on the TPIDATA line at the transfer rate given by TPICLK.

If a collision is detected during transmission, the output driver is disabled. The TPI access layer enters the error state and the TPI physical layer is put into receive mode, waiting for a BREAK character.

### 21.4.9. Collision Detection Exception

The TPI physical layer uses one bi-directional data line for both data reception and transmission. A possible drive contention may occur, if the external programmer and the TPI physical layer drive the TPIDATA line simultaneously. In order to reduce the effect of the drive contention, a collision detection mechanism is supported. The collision detection is based on the way the TPI physical layer drives the TPIDATA line.

The TPIDATA line is driven by a tri-state, push-pull driver with internal pull-up. The output driver is always enabled when a logical zero is sent. When sending successive logical ones, the output is only driven actively during the first clock cycle. After this, the output driver is automatically tri-stated and the TPIDATA line is kept high by the internal pull-up. The output is re-enabled, when the next logical zero is sent.

The collision detection is enabled in transmit mode, when the output driver has been disabled. The data line should now be kept high by the internal pull-up and it is monitored to see, if it is driven low by the external programmer. If the output is read low, a collision has been detected.

There are some potential pit-falls related to the way collision detection is performed. For example, collisions cannot be detected when the TPI physical layer transmits a bit-stream of successive logical zeros, or bit-stream of alternating logical ones and zeros. This is because the output driver is active all the time, preventing polling of the TPIDATA line. However, within a single frame the two stop bits should always be transmitted as logical ones, enabling collision detection at least once per frame (as long as the frame format is not violated regarding the stop bits).

The TPI physical layer will cease transmission when it detects a collision on the TPIDATA line. The collision is signalized to the TPI access layer, which immediately changes the physical layer to receive mode and goes to the error state. The TPI access layer can be recovered from the error state only by sending a BREAK character.

### 21.4.10. Direction Change

In order to ensure correct timing of the half-duplex operation, a simple guard time mechanism has been added to the physical layer. When the TPI physical layer changes from receive to transmit mode, a configurable number of additional IDLE bits are inserted before the start bit is transmitted. The minimum transition time between receive and transmit mode is two IDLE bits.

The total IDLE time is the specified guard time plus two IDLE bits. The guard time is configured by dedicated bits in the TPIPCR register. The default guard time value after the physical layer is initialized is 128 bits.

The external programmer looses control of the TPIDATA line when the TPI target changes from receive mode to transmit. The guard time feature relaxes this critical phase of the communication. When the external programmer changes from receive mode to transmit, a minimum of one IDLE bit should be inserted before the start bit is transmitted.

### 21.4.11. Access Layer of Tiny Programming Interface

The TPI access layer is responsible for handling the communication with the external programmer. The communication is based on message format, where each message comprises an instruction followed by one or more byte-sized operands. The instruction is always sent by the external programmer but operands are sent either by the external programmer or by the TPI access layer, depending on the type of instruction issued.

The TPI access layer controls the character transfer direction on the TPI physical layer. It also handles the recovery from the error state after exception.

The Control and Status Space (CSS) of the Tiny Programming Interface is allocated for control and status registers in the TPI access Layer. The CSS consist of registers directly involved in the operation of the TPI itself. These register are accessible using the SLDCS and SSTCS instructions.

The access layer can also access the data space, either directly or indirectly using the Pointer Register (PR) as the address pointer. The data space is accessible using the SLD, SST, SIN and SOUT instructions. The address pointer can be stored in the Pointer Register using the SLDPR instruction.

#### 21.4.11.1. Message format

Each message comprises an instruction followed by one or more byte operands. The instruction is always sent by the external programmer. Depending on the instruction all the following operands are sent either by the external programmer or by the TPI.

The messages can be categorized in two types based on the instruction, as follows:

- Write messages. A write message is a request to write data. The write message is sent entirely by the external programmer. This message type is used with the SSTCS, SST, STPR, SOUT and SKEY instructions.
- Read messages. A read message is a request to read data. The TPI reacts to the request by sending the byte operands. This message type is used with the SLDCS, SLD and SIN instructions.

All the instructions except the SKEY instruction require the instruction to be followed by one byte operand. The SKEY instruction requires 8 byte operands. For more information, see the TPI instruction set.

#### 21.4.11.2. Exception Handling and Synchronisation

Several situations are considered exceptions from normal operation of the TPI. When the TPI physical layer is in receive mode, these exceptions are:

- The TPI physical layer detects a parity error.
- The TPI physical layer detects a frame error.
- The TPI physical layer recognizes a BREAK character.

When the TPI physical layer is in transmit mode, the possible exceptions are:

- The TPI physical layer detects a data collision.

All these exceptions are signalized to the TPI access layer. The access layer responds to an exception by aborting any on-going operation and enters the error state. The access layer will stay in the error state until a BREAK character has been received, after which it is taken back to its default state. As a consequence, the external programmer can always synchronize the protocol by simply transmitting two successive BREAK characters.

## 21.5. Instruction Set

The TPI has a compact instruction set that is used to access the TPI Control and Status Space (CSS) and the data space. The instructions allow the external programmer to access the TPI, the NVM Controller and the NVM memories. All instructions except SKEY require one byte operand following the instruction. The SKEY instruction is followed by 8 data bytes. All instructions are byte-sized.

**Table 21-1. Instruction Set Summary**

| Mnemonic | Operand | Description | Operation |
|---|---|---|---|
| SLD | data, PR | Serial LoaD from data space using indirect addressing | data ← DS[PR] |
| SLD | data, PR+ | Serial LoaD from data space using indirect addressing and post-increment | data ← DS[PR]<br>PR ← PR+1 |
| SST | PR, data | Serial STore to data space using indirect addressing | DS[PR] ← data |
| SST | PR+, data | Serial STore to data space using indirect addressing and post-increment | DS[PR] ← data<br>PR ← PR+1 |
| SSTPR | PR, a | Serial STore to Pointer Register using direct addressing | PR[a] ← data |
| SIN | data, a | Serial IN from data space | data ← I/O[a] |
| SOUT | a, data | Serial OUT to data space | I/O[a] ← data |

| Mnemonic | Operand | Description | Operation |
|---|---|---|---|
| SLDCS | data, a | Serial LoaD from Control and Status space using direct addressing | data ← CSS[a] |
| SSTCS | a, data | Serial STore to Control and Status space using direct addressing | CSS[a] ← data |
| SKEY | Key, {8{data}} | Serial KEY | Key ← {8{data}} |

### 21.5.1. SLD - Serial LoaD from data space using indirect addressing

The SLD instruction uses indirect addressing to load data from the data space to the TPI physical layer shift-register for serial read-out. The data space location is pointed by the Pointer Register (PR), where the address must have been stored before data is accessed. The Pointer Register is either left unchanged by the operation, or post-incremented.

**Table 21-2. The Serial Load from Data Space (SLD) Instruction**

| Operation | Opcode | Remarks | Register |
|---|---|---|---|
| data ← DS[PR] | 0010 0000 | PR ← PR | Unchanged |
| data ← DS[PR] | 0010 0100 | PR ← PR + 1 | Post increment |

### 21.5.2. SST - Serial STore to data space using indirect addressing

The SST instruction uses indirect addressing to store into data space the byte that is shifted into the physical layer shift register. The data space location is pointed by the Pointer Register (PR), where the address must have been stored before the operation. The Pointer Register can be either left unchanged by the operation, or it can be post-incremented.

**Table 21-3. The Serial Store to Data Space (SST) Instruction**

| Operation | Opcode | Remarks | Register |
|---|---|---|---|
| DS[PR] ← data | 0110 0000 | PR ← PR | Unchanged |
| DS[PR] ← data | 0110 0000 | PR ← PR + 1 | Post increment |

### 21.5.3. SSTPR - Serial STore to Pointer Register

The SSTPR instruction stores the data byte that is shifted into the physical layer shift register to the Pointer Register (PR). The address bit of the instruction specifies which byte of the Pointer Register is accessed.

**Table 21-4. The Serial Store to Pointer Register (SSTPR) Instruction**

| Operation | Opcode | Remarks |
|---|---|---|
| PR[a] ← data | 0110 100a | Bit ‘a’ addresses Pointer Register byte |

### 21.5.4. SIN - Serial IN from i/o space using direct addressing

The SIN instruction loads data byte from the I/O space to the shift register of the physical layer for serial read-out. The instuction uses direct addressing, the address consisting of the 6 address bits of the instruction.

**Table 21-5. The Serial IN from i/o space (SIN) Instruction**

| Operation | Opcode | Remarks |
|---|---|---|
| data ← I/O[a] | 0aa1 aaaa | Bits marked ‘a’ form the direct, 6-bit address |

### 21.5.5. SOUT - Serial OUT to i/o space using direct addressing

The SOUT instruction stores the data byte that is shifted into the physical layer shift register to the I/O space. The instruction uses direct addressing, the address consisting of the 6 address bits of the instruction.

**Table 21-6. The Serial OUT to i/o space (SOUT) Instruction**

| Operation | Opcode | Remarks |
|---|---|---|
| I/O[a] ← data | 1aa1 aaaa | Bits marked ‘a’ form the direct, 6-bit address |

### 21.5.6. SLDCS - Serial LoaD data from Control and Status space using direct addressing

The SLDCS instruction loads data byte from the TPI Control and Status Space to the TPI physical layer shift register for serial read-out. The SLDCS instruction uses direct addressing, the direct address consisting of the 4 address bits of the instruction.

**Table 21-7. The Serial Load Data from Control and Status space (SLDCS) Instruction**

| Operation | Opcode | Remarks |
|---|---|---|
| data ← CSS[a] | 1000 aaaa | Bits marked ‘a’ form the direct, 4-bit address |

### 21.5.7. SSTCS - Serial STore data to Control and Status space using direct addressing

The SSTCS instruction stores the data byte that is shifted into the TPI physical layer shift register to the TPI Control and Status Space. The SSTCS instruction uses direct addressing, the direct address consisting of the 4 address bits of the instruction.

**Table 21-8. The Serial STore data to Control and Status space (SSTCS) Instruction**

| Operation | Opcode | Remarks |
|---|---|---|
| CSS[a] ← data | 1100 aaaa | Bits marked ‘a’ form the direct, 4-bit address |

### 21.5.8. SKEY - Serial KEY signaling

The SKEY instruction is used to signal the activation key that enables NVM programming. The SKEY instruction is followed by the 8 data bytes that includes the activation key.

**Table 21-9. The Serial KEY signaling (SKEY) Instruction**

| Operation | Opcode | Remarks |
|---|---|---|
| Key ← {8[data]} | 1110 0000 | Data bytes follow after instruction |

## 21.6. Accessing the Non-Volatile Memory Controller

By default, NVM programming is not enabled. In order to access the NVM Controller and be able to program the non-volatile memories, a unique key must be sent using the SKEY instruction.

**Table 21-10. Enable Key for Non-Volatile Memory Programming**

| Key | Value |
|---|---|
| NVM Program Enable | 0x1289AB45CDD888FF |

After the key has been given, the Non-Volatile Memory Enable (NVMEN) bit in the TPI Status Register (TPISR) must be polled until the Non-Volatile memory has been enabled.

NVM programming is disabled by writing a logical zero to the NVMEN bit in TPISR.

## 21.7. Control and Status Space Register Descriptions

The control and status registers of the Tiny Programming Interface are mapped in the Control and Status Space (CSS) of the interface. These registers are not part of the I/O register map and are accessible via SLDCS and SSTCS instructions, only. The control and status registers are directly involved in configuration and status monitoring of the TPI.

**Table 21-11. Summary of Control and Status Registers**

| Offset | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0F | TPIIR | Tiny Programming Interface Identification Code | | | | | | | |
| 0x0E<br>...<br>0x03 | Reserved | - | - | - | - | - | - | - | - |
| 0x02 | TPIPCR | - | - | - | - | - | GT2 | GT1 | GT0 |
| 0x01 | Reserved | - | - | - | - | - | - | - | - |
| 0x00 | TPISR | - | - | - | - | - | - | NVMEN | - |

### 21.7.1. Tiny Programming Interface Identification Register

**Name:** TPIIR
**Offset:**
**Reset:** 0x00
**Property:** CSS: 0x0F

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TPIIC[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 – TPIIC[7:0]: Tiny Programming Interface Identification Code**
These bits give the identification code for the Tiny Programming Interface. The code can be used be the external programmer to identify the TPI.

**Table 21-12. Identification Code for Tiny Programming Interface**

| Code | Value |
|---|---|
| Interface Identification | 0x80 |

### 21.7.2. Tiny Programming Interface Physical Layer Control Register

**Name:** TPIPCR
**Offset:**
**Reset:** 0x00
**Property:** CSS: 0x02

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | GT2 | GT1 | GT0 |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – GTn: Guard Time [n=2:0]**
These bits specify the number of additional IDLE bits that are inserted to the idle time when changing from reception mode to transmission mode. Additional delays are not inserted when changing from transmission mode to reception.

The total idle time when changing from reception to transmission mode is Guard Time plus two IDLE bits.

**Table 21-13. Identification Code for Tiny Programming Interface**

| GT2 | GT1 | GT0 | Guard Time (Number of IDLE bits) |
|---|---|---|---|
| 0 | 0 | 0 | +128 (default value) |
| 0 | 0 | 1 | +64 |
| 0 | 1 | 0 | +32 |
| 0 | 1 | 1 | +16 |
| 1 | 0 | 0 | +8 |
| 1 | 0 | 1 | +4 |
| 1 | 1 | 0 | +2 |
| 1 | 1 | 1 | +0 |

The default Guard Time is 128 IDLE bits. To speed up the communication, the Guard Time should be set to the shortest safe value.

### 21.7.3. Tiny Programming Interface Status Register

**Name:** TPISR
**Offset:**
**Reset:** 0x00
**Property:** CSS: 0x00

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | NVMEN | |
| Access | | | | | | | R/W | |
| Reset | | | | | | | 0 | |

**Bit 1 – NVMEN: Non-Volatile Memory Programming Enabled**
NVM programming is enabled when this bit is set. The external programmer can poll this bit to verify the interface has been successfully enabled.

NVM programming is disabled by writing this bit to zero.

# 22. Electrical Characteristics

## 22.1. Absolute Maximum Ratings*

| | |
|---|---|
| Operating Temperature | -55°C to +125°C |
| Storage Temperature | -65°C to +150°C |
| Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground | -0.5V to $V_{CC}$+0.5V |
| Voltage on $\overline{\text{RESET}}$ with respect to Ground | -0.5V to +13.0V |
| Maximum Operating Voltage | 6.0V |
| DC Current per I/O Pin | 40.0 mA |
| DC Current $V_{CC}$ and GND Pins | 200.0 mA |

**Note:** Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## 22.2. DC Characteristics

**Table 22-1. Common DC characteristics $T_A$ = -40°C to 125°C, $V_{CC}$ = 1.8V to 5.5V (unless otherwise noted)**

| Symbol | Parameter | Condition | Min. | Typ.(1) | Max. | Units |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage, except XTAL1 and RESET pin | $V_{CC}$ = 1.8V - 2.4V | -0.5 | | $0.2V_{CC}$(2) | V |
| | | $V_{CC}$ = 2.4V - 5.5V | -0.5 | | $0.3V_{CC}$(2) | |
| $V_{IH}$ | Input High Voltage, except XTAL1 and RESET pins | $V_{CC}$ = 1.8V - 2.4V | $0.7V_{CC}$(3) | | $V_{CC}$ + 0.5 | V |
| | | $V_{CC}$ = 2.4V - 5.5V | $0.6V_{CC}$(3) | | $V_{CC}$ + 0.5 | |
| $V_{IL1}$ | Input Low Voltage, CLKI pin | $V_{CC}$ = 1.8V - 5.5V | -0.5 | | $0.1V_{CC}$(2) | V |
| $V_{IH1}$ | Input High Voltage,CLKI pin | $V_{CC}$ = 1.8V - 2.4V | $0.8V_{CC}$(3) | | $V_{CC}$ + 0.5 | V |
| | | $V_{CC}$ = 2.4V - 5.5V | $0.7V_{CC}$(3) | | $V_{CC}$ + 0.5 | |
| $V_{IL2}$ | Input Low Voltage, RESET pin | $V_{CC}$ = 1.8V - 5.5V | -0.5 | | $0.1V_{CC}$(2) | V |

| Symbol | Parameter | Condition | Min. | Typ.(1) | Max. | Units |
|---|---|---|---|---|---|---|
| $V_{IH2}$ | Input High Voltage, RESET pin | $V_{CC}$ = 1.8V - 5.5V | $0.9V_{CC}$(3) | | $V_{CC}$ + 0.5 | V |
| $V_{IL3}$ | Input Low Voltage, RESET pin as I/O | $V_{CC}$ = 1.8V - 2.4V | -0.5 | | $0.2V_{CC}$(2) | V |
| | | $V_{CC}$ = 2.4V - 5.5V | -0.5 | | $0.3V_{CC}$(2) | |
| $V_{IH3}$ | Input High Voltage, RESET pin as I/O | $V_{CC}$ = 1.8V - 2.4V | $0.7V_{CC}$(3) | | $V_{CC}$ + 0.5 | V |
| | | $V_{CC}$ = 2.4V - 5.5V | $0.6V_{CC}$(3) | | $V_{CC}$ + 0.5 | |
| $V_{OL}$ | Output Low Voltage(4) except RESET pin(6) | $I_{OL}$ = 10mA, $V_{CC}$ = 5V | | | 0.6 | V |
| | | $I_{OL}$ = 5mA, $V_{CC}$ = 3V | | | 0.5 | |
| $V_{OH}$ | Output High Voltage(5) except Reset pin(6) | $I_{OH}$ = 10mA, $V_{CC}$ = 5V | 4.3 | | | V |
| | | $I_{OH}$ = 5mA, $V_{CC}$ = 3V | 2.5 | | | |
| $I_{IL}$ | Input Leakage Current I/O Pin | $V_{CC}$ = 5.5V, pin low (absolute value) | | <0.05 | 1 | µA |
| $I_{IH}$ | Input Leakage Current I/O Pin | $V_{CC}$ = 5.5V, pin high (absolute value) | | <0.05 | 1 | µA |
| $R_{RST}$ | Reset Pull-up Resistor | $V_{CC}$ = 5.5V, input low | 30 | | 60 | kΩ |
| $R_{PU}$ | I/O Pin Pull-up Resistor | $V_{CC}$ = 5.5V, input low | 20 | | 50 | kΩ |
| $I_{ACLK}$ | Analog Comparator Input Leakage Current | $V_{CC}$=5V , $V_{in}$ = $V_{CC}$/2 | -50 | | 50 | nA |

| Symbol | Parameter | Condition | | Min. | Typ.(1) | Max. | Units |
|---|---|---|---|---|---|---|---|
| $I_{CC}$ | Power Supply Current(7) | Active 1MHz, VCC = 2V | | | 0.2 | 0.5 | mA |
| | | Active 4MHz, VCC = 3V | | | 1.1 | 1.2 | mA |
| | | Active 8MHz, VCC = 5V | | | 3.2 | 4 | mA |
| | | Idle 1MHz, VCC = 2V | | | 0.03 | 0.2 | mA |
| | | Idle 4MHz, VCC = 3V | | | 0.2 | 0.5 | mA |
| | | Idle 8MHz, VCC = 5V | | | 0.9 | 1.5 | mA |
| | Power-down mode(8) | WDT enabled, Vcc =3V | $T_A$=105° C | | 5.5 | 10 | µA |
| | | | $T_A$=125° C | | 5.5 | 16 | µA |
| | | WDT disabled, Vcc =3V | $T_A$=105° C | | 0.11 | 2 | µA |
| | | | $T_A$=125° C | | 0.11 | 8 | µA |

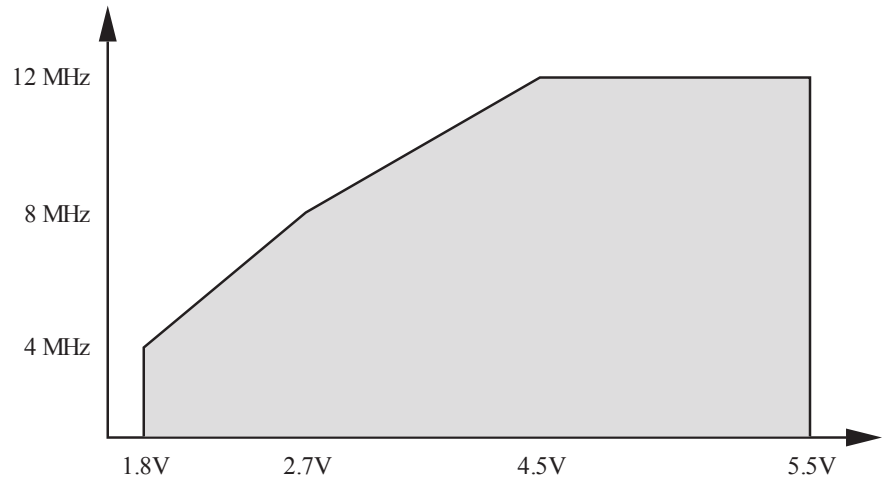
**Note:**

1. Typical values at 25°C, maximum values unless otherwise noted.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. “Min” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (10 mA at $V_{CC}$ = 5V, 5 mA at $V_{CC}$ = 3V) under steady state conditions (non-transient), the sum of all $I_{OL}$ (for all ports) should not exceed 60 mA. If $I_{OL}$ exceeds the test conditions, $V_{OL}$ may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10 mA at $V_{CC}$ = 5V, 5 mA at $V_{CC}$ = 3V) under steady state conditions (non-transient), the sum of all $I_{OH}$ (for all ports) should not exceed 60 mA. 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.
6. The $\overline{\text{RESET}}$ pin must tolerate high voltages when entering and operating in programming modes and, as a consequence, has a weak drive strength as compared to regular I/O pins.
7. Values are with external clock using methods described in *Minimizing Power Consumption*. Power Reduction is enabled (PRR = 0xFF) and there is no I/O drive.
8. BOD Disabled.

## 22.3. Speed

The maximum operating frequency of the device depends on $V_{CC}$ . The relationship between maximum frequency vs. $V_{CC}$ is linear between 1.8V < $V_{CC}$ < 4.5V.

**Figure 22-1. Maximum Frequency vs. $V_{CC}$**




## 22.4. Clock Characteristics

### 22.4.1. Accuracy of Calibrated Internal Oscillator

It is possible to manually calibrate the internal oscillator to be more accurate than default factory calibration. Note that the oscillator frequency depends on temperature and voltage. .

**Table 22-2. Calibration Accuracy of Internal RC Oscillator**

| Calibration Method | Target Frequency | $V_{CC}$ | Temperature | Accuracy at given Voltage & Temperature |
|---|---|---|---|---|
| Factory Calibration | 8.0MHz | 2.7 - 4.0V | 0°C - 85°C | ±2% |
| User Calibration | Fixed frequency within: 7.3 - 8.1MHz | Fixed voltage within: 1.7 - 5.5V | Fixed temp. within: -40°C - 85°C | ±1%(1) |

**Note:**

1. Accuracy of oscillator frequency at calibration point (fixed temperature and fixed voltage).

### 22.4.2. External Clock Drive

**Figure 22-2. External Clock Drive Waveform**

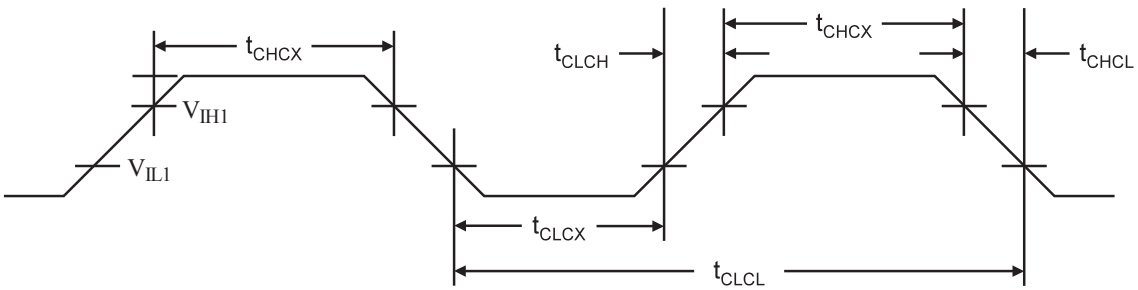


**Table 22-3. External Clock Drive Characteristics**

| Symbol | Parameter | $V_{CC}$ = 1.8 - 5.5V | | $V_{CC}$ = 2.7 - 5.5V | | $V_{CC}$ = 4.5 - 5.5V | | Units |
|---|---|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | Min. | Max. | |
| $1/t_{CLCL}$ | Clock Frequency | 0 | 4 | 0 | 8 | 0 | 12 | MHz |
| $t_{CLCL}$ | Clock Period | 250 | | 125 | | 83 | | ns |
| $t_{CHCX}$ | High Time | 100 | | 50 | | 33 | | ns |
| $t_{CLCX}$ | Low Time | 100 | | 50 | | 33 | | ns |
| $t_{CLCH}$ | Rise Time | | 2.0 | | 1 | | 0.6 | µs |
| $t_{CHCL}$ | Fall Time | | 2.0 | | 1 | | 0.6 | µs |
| $\Delta t_{CLCL}$ | Change in period from one clock cycle to the next | | 2 | | 2 | | 2 | % |

## 22.5. System and Reset Characteristics

**Table 22-4. Reset, VLM, and Internal Voltage Characteristics**

| Symbol | Parameter | Condition | Min(1) | Typ(1) | Max(1) | Units |
|---|---|---|---|---|---|---|
| $V_{RST}$ | $\overline{\text{RESET}}$ Pin Threshold Voltage | | 0.2 $V_{CC}$ | | 0.9$V_{CC}$ | V |
| $V_{BG}$ | Internal bandgap voltage | $V_{CC}$ = 1.8V to 5.5V | 1.0 | 1.1 | 1.2 | V |
| $t_{RST}$ | Minimum pulse width on $\overline{\text{RESET}}$ Pin | $V_{CC}$ = 1.8V | | 2100 | | ns |
| | | $V_{CC}$ = 3V | | 700 | | ns |
| | | $V_{CC}$ = 5V | | 400 | | ns |
| $t_{TOUT}$ | Time-out after reset | | | 64 | 128 | ms |

**Note:**

1. Values are guidelines, only

### 22.5.1. Power-On Reset

**Table 22-5. Characteristics of Enhanced Power-On Reset. $T_A$ = -40 to 125°C**

| Symbol | Parameter | Min(1) | Typ(1) | Max(1) | Units |
|---|---|---|---|---|---|
| $V_{POR}$ | Release threshold of power-on reset (2) | 1.1 | 1.4 | 1.6 | V |
| $V_{POA}$ | Activation threshold of power-on reset (3) | 0.6 | 1.2 | 1.6 | V |
| $SR_{ON}$ | Power-On Slope Rate | 0.01 | | | V/ms |

**Note:**

1. Values are guidelines, only
2. Threshold where device is released from reset when voltage is rising

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

### 22.5.2. $V_{CC}$ Level Monitor

**Table 22-6. Voltage Level Monitor Thresholds**

| Parameter | Min | Typ (1) | Max | Units |
|---|---|---|---|---|
| Trigger level VLM1L | 1.1 | 1.4 | 1.6 | V |
| Trigger level VLM1H | 1.4 | 1.6 | 1.8 | |
| Trigger level VLM2 | 2.0 | 2.5 | 2.7 | |
| Trigger level VLM3 | 3.2 | 3.7 | 4.5 | |
| Settling time VMLM2,VLM3 (VLM1H,VLM1L) | | 5 (50) | | µs |

**Note:**

1. Typical values at room temperature

## 22.6. Analog Comparator Characteristics

**Table 22-7. Analog Comparator Characteristics, $T_A$ = -40°C to 125°C**

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{AIO}$ | Input Offset Voltage | $V_{CC}$ = 5V, $V_{IN}$ = $V_{CC}$ / 2 | | 10 | 40 | mV |
| $I_{LAC}$ | Input Leakage Current | $V_{CC}$ = 5V, $V_{IN}$ = $V_{CC}$ / 2 | -50 | | 50 | nA |
| $t_{APD}$* | Analog Propagation Delay (from saturation to 100mV overdrive) | $V_{CC}$ = 2.7V | | 150 | | ns |
| | | $V_{CC}$ = 4.0V | | 185 | | |
| | Analog Propagation Delay (from step change of 100mV) | $V_{CC}$ = 2.7V | | 135 | | |
| | | $V_{CC}$ = 4.0V | | 160 | | |
| $t_{DPD}$ | Digital Propagation Delay | $V_{CC}$ = 1.8V - 5.5V | | 1 | 2 | CLK |

**Note:** *: 15ns delay IP to pad removed

## 22.7. ADC Characteristics

**Table 22-8. ADC Characteristics. T = -40°C to 125°C $V_{CC}$ = 1.8V - 5.5V**

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Resolution | | | | 10 | Bits |
| | Absolute accuracy (Including INL, DNL, and Quantization, Gain and Offset Errors) | $V_{REF}$ = $V_{CC}$ = 4.3V, ADC clock = 200 kHz | | 2 | | LSB |
| | | $V_{REF}$ = $V_{CC}$ = 4.3V, ADC clock = 1 MHz | | 3 | | |
| | | $V_{REF}$ = $V_{CC}$ = 4.3V, ADC clock = 200 kHz<br>Noise Reduction Mode | | 1.5 | | LSB |
| | | $V_{REF}$ = $V_{CC}$ = 4.3V, ADC clock = 1 MHz<br>Noise Reduction Mode | | 2.5 | | |
| INL | Integral Non-Linearity (Accuracy after Offset and Gain Calibration) | $V_{REF}$ = $V_{CC}$ = 4.0V, ADC clock = 200 kHz | 0.51 | 0.68 | 0.88 | LSB |
| | | $V_{REF}$ = $V_{CC}$ = 4.0V, ADC clock = 1 MHz | 0.39 | 0.62 | 0.92 | |
| DNL | Differential Non-linearity | $V_{REF}$ = $V_{CC}$ = 4.0V, ADC clock = 200 kHz | 0.42 | 0.49 | 0.73 | LSB |
| | | $V_{REF}$ = $V_{CC}$ = 4.0V, ADC clock = 1 MHz | 0.22 | 0.48 | 0.55 | |
| | Gain Error | $V_{REF}$ = $V_{CC}$ = 4.0V, ADC clock = 200 kHz | -7.2 | -4 | -1 | LSB |
| | | Int $V_{REF}$ = 1.1V, $V_{CC}$ = 4.0V, ADC clock = 200 kHz | -41.3 | -13.3 | -1.9 | |
| | | Int $V_{REF}$ = 2.2V, $V_{CC}$ = 4.0V, ADC clock = 200 kHz | -38.3 | -8.7 | 3.1 | |
| | | Int $V_{REF}$ = 4.3V, $V_{CC}$ = 4.0V, ADC clock = 200 kHz | -80.4 | -3.2 | 9.9 | |
| | Offset Error | $V_{REF}$ = $V_{CC}$ = 4.0V, ADC clock = 200 kHz | 3.0 | 5.1 | 8 | LSB |
| | | Int $V_{REF}$ = 1.1V, $V_{CC}$ = 4.0V, ADC clock = 200 kHz | -54 | 9.1 | 2 | |
| | | Int $V_{REF}$ = 2.2V, $V_{CC}$ = 4.0V, ADC clock = 200 kHz | 2 | 5.4 | 11 | |
| | | Int $V_{REF}$ = 4.3V, $V_{CC}$ = 4.0V, ADC clock = 200 kHz | 1 | 3.4 | 5.4 | |

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Conversion Time | Single Conversion | 13 | | 260 | µs |
| | Clock Frequency | | 50 | | 1000 | kHz |
| $V_{IN}$ | Input Voltage | | GND | | $V_{REF}$ | V |
| | Input Bandwidth | | | 38.5 | | kHz |
| $R_{AIN}$ | Analog Input Resistance | | | 100 | | MΩ |
| | ADC Conversion Output | | 0 | | 1023 | LSB |

## 22.8. Serial Programming Characteristics

**Figure 22-3. Serial Programming Timing**

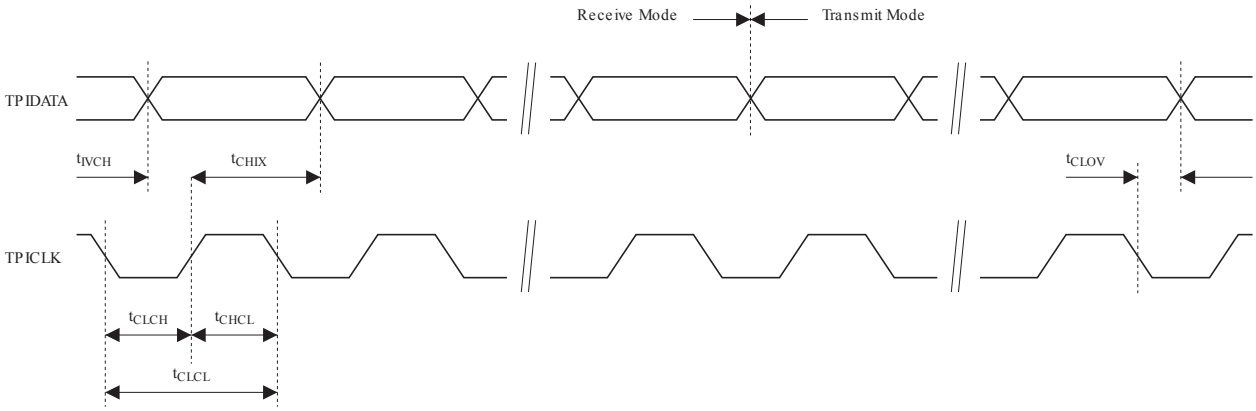



**Table 22-9. Serial Programming Characteristics, $T_A$ = -40°C to 125°C, $V_{CC}$ = 5V ± 5% (Unless Otherwise Noted)**

| Symbol | Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $1/t_{CLCL}$ | Clock Frequency | | | 2 | MHz |
| $t_{CLCL}$ | Clock Period | 500 | | | ns |
| $t_{CLCH}$ | Clock Low Pulse Width | 200 | | | ns |
| $t_{CHCH}$ | Clock High Pulse Width | 200 | | | ns |
| $t_{IVCH}$ | Data Input to Clock High Setup Time | 50 | | | ns |
| $t_{CHIX}$ | Data Input Hold Time After Clock High | 100 | | | ns |
| $t_{CLOV}$ | Data Output Valid After Clock Low Time | | | 200 | ns |

# 23. Typical Characteristics

The data contained in this section is largely based on simulations and characterization of similar devices in the same process and design methods. Thus, the data should be treated as indications of how the part will behave.

The following charts show typical behavior. These figures are not tested during manufacturing. During characterisation devices are operated at frequencies higher than test limits but they are not guaranteed to function properly at frequencies higher than the ordering code indicates.

All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. Current consumption is a function of several factors such as operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

A sine wave generator with rail-to-rail output is used as clock source but current consumption in Power-Down mode is independent of clock selection. 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.

The current drawn from pins with a capacitive load may be estimated (for one pin) as follows:

$$I_{CP} \simeq V_{CC} \times C_L \times f_{SW}$$

where $V_{CC}$ = operating voltage, $C_L$ = load capacitance and $f_{SW}$ = average switching frequency of I/O pin.

## 23.1. Active Supply Current

**Figure 23-1. Active Supply Current vs. Low Frequency (0.1 - 1.0 MHz)**

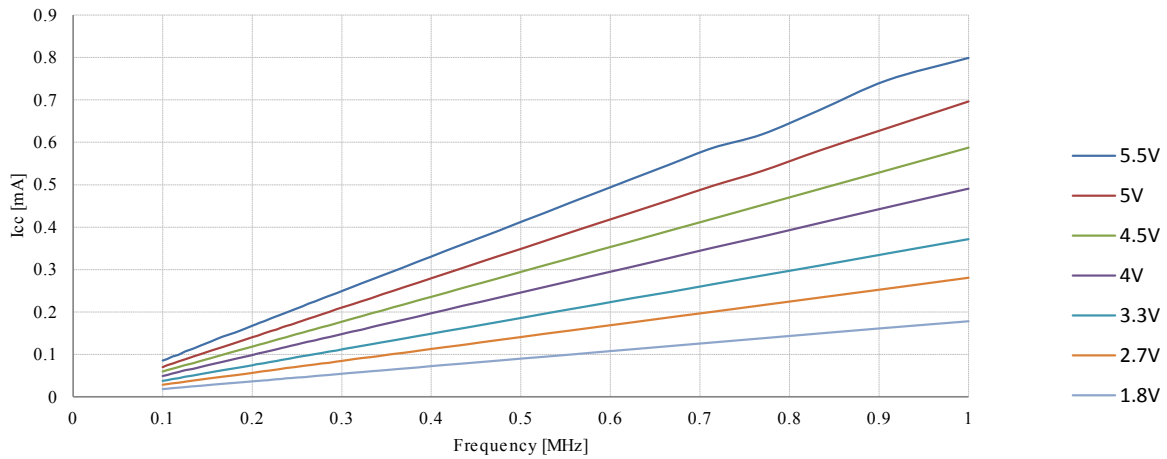


**Figure 23-2. Active Supply Current vs. frequency (1 - 12 MHz)**

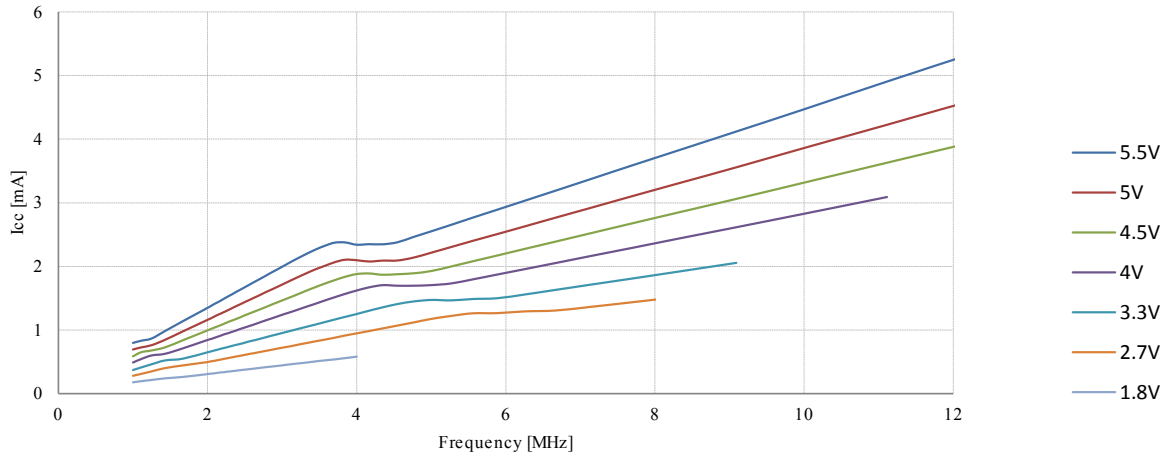



**Figure 23-3. Active Supply Current vs. $V_{CC}$ (Internal Oscillator, 8 MHz)**

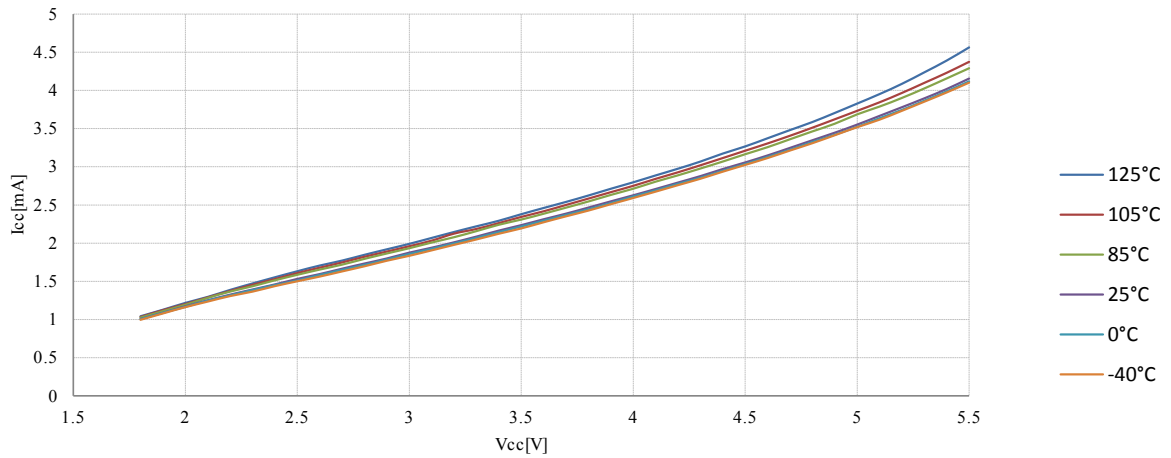


**Figure 23-4. Active Supply Current vs. $V_{CC}$ (Internal Oscillator, 1 MHz)**

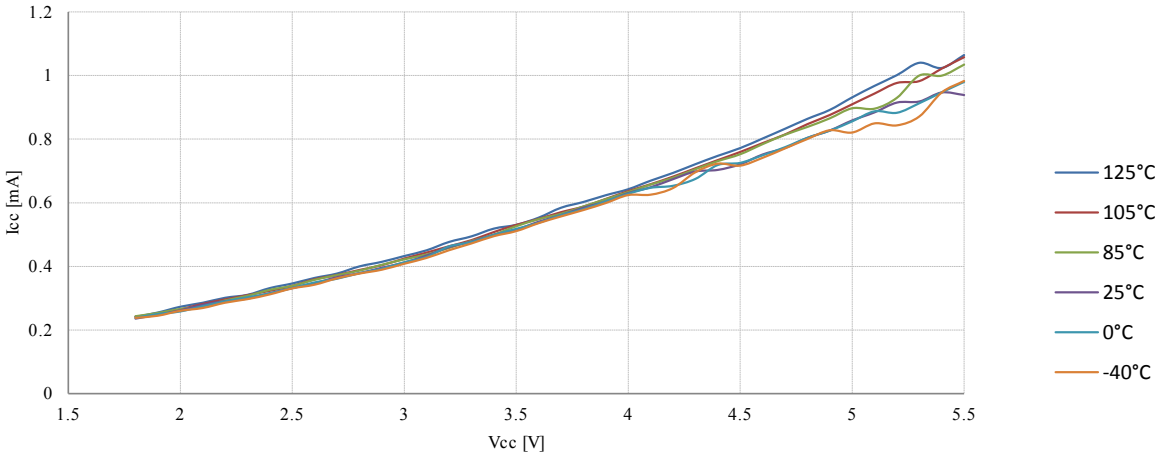
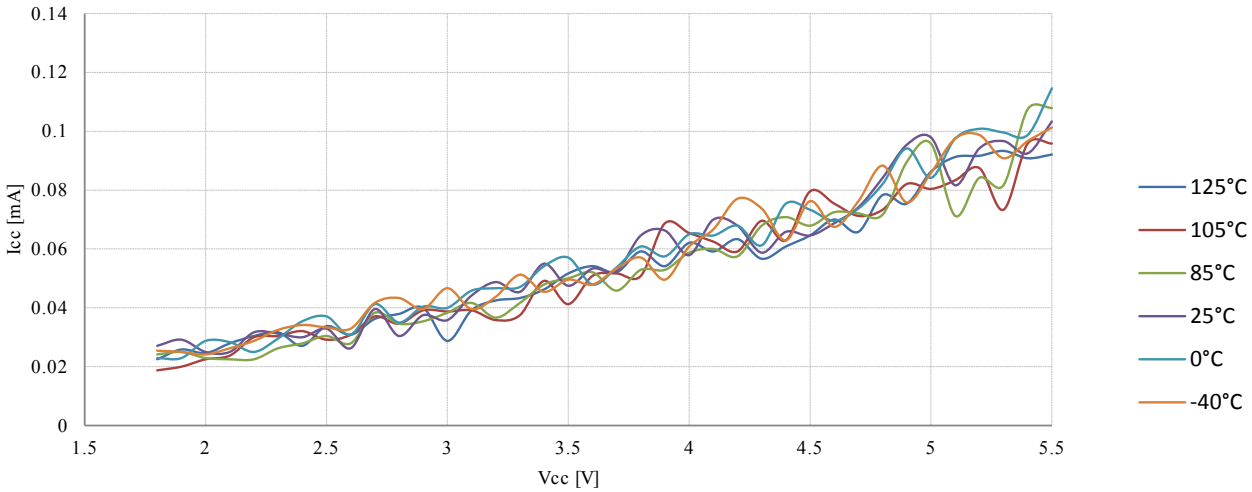



**Figure 23-5. Active Supply Current vs. $V_{CC}$ (Internal Oscillator, 128 kHz)**



## 23.2. Idle Supply Current

**Figure 23-6. Idle Supply Current vs. Low Frequency (0.1 - 1.0 MHz)**

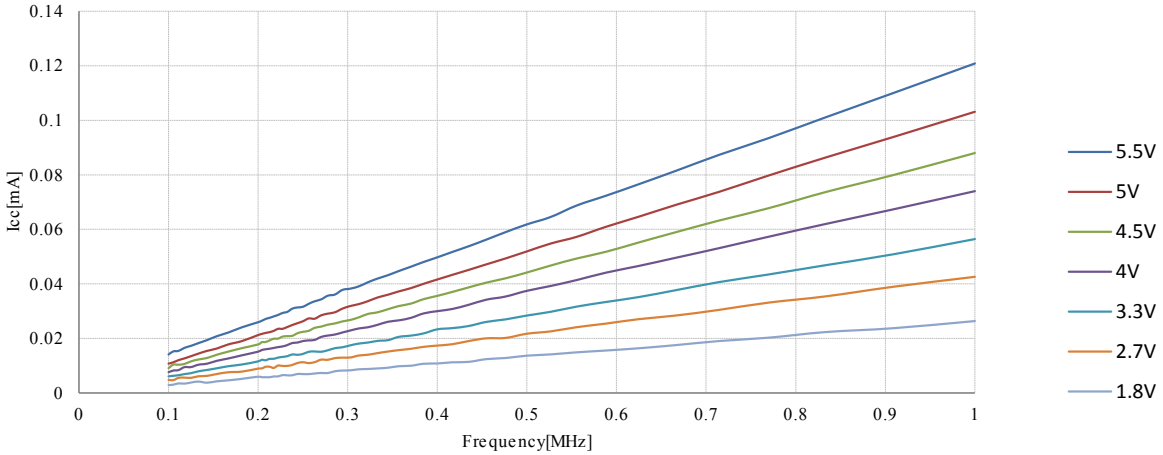



**Figure 23-7. Idle Supply Current vs. Frequency (1 - 12 MHz)**

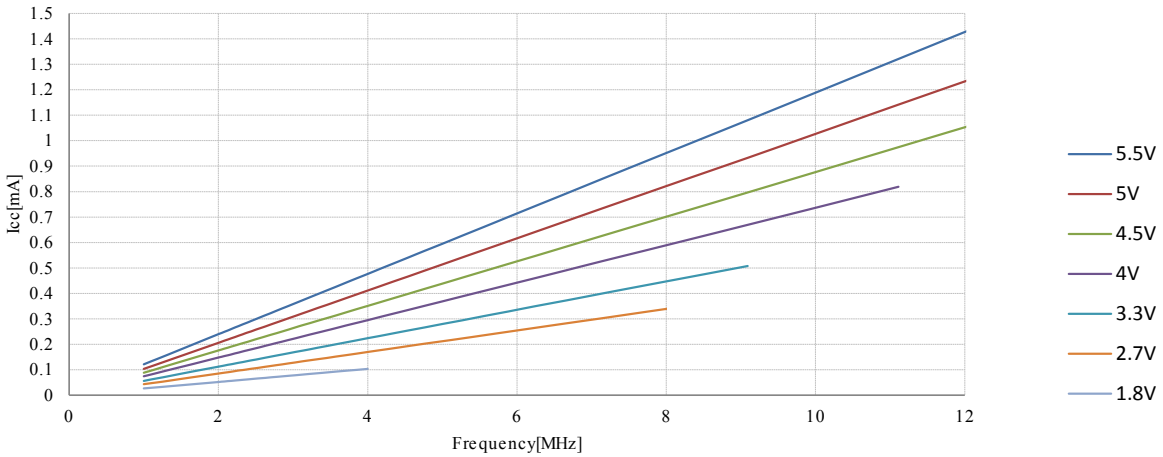


**Figure 23-8. Idle Supply Current vs. $V_{CC}$ (Internal Oscillator, 8 MHz)**

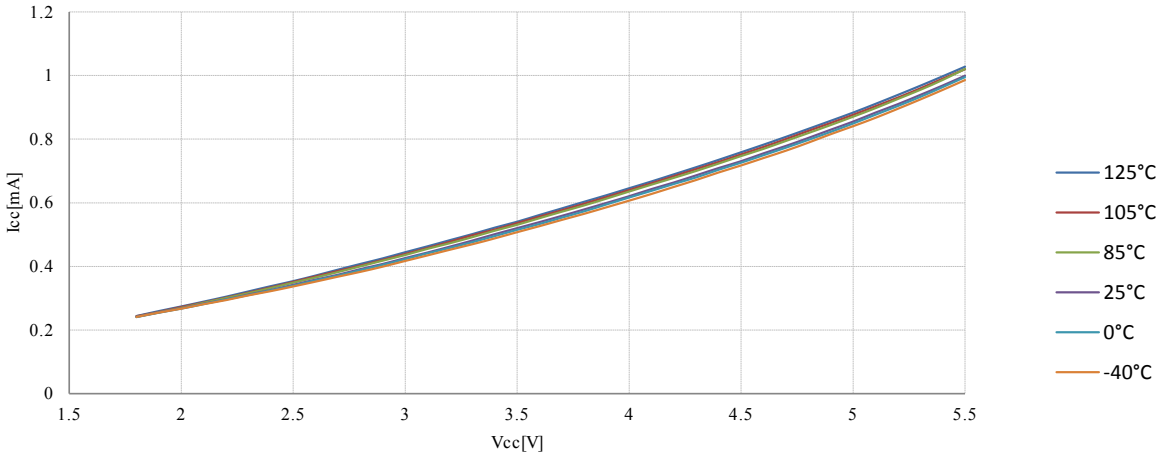
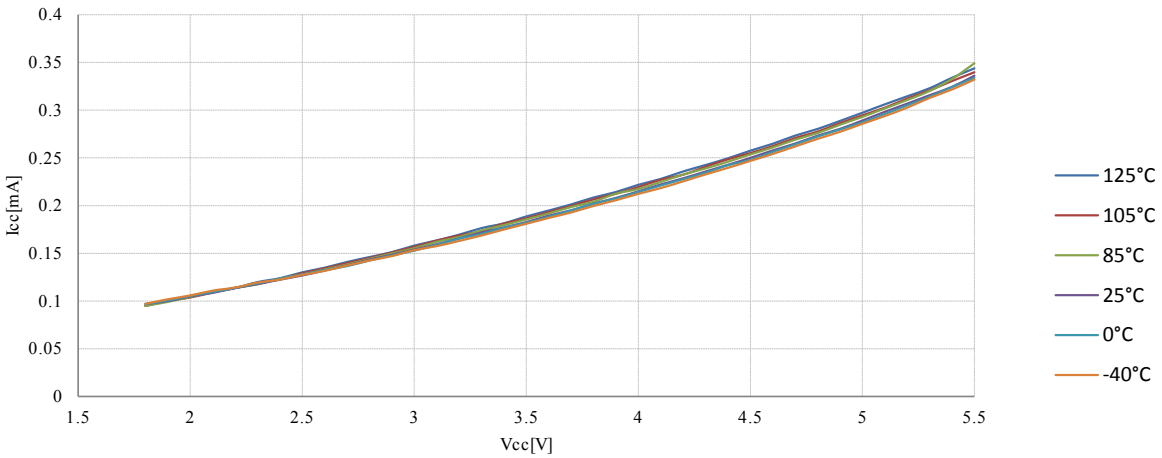



**Figure 23-9. Idle Supply Current vs. $V_{CC}$ (Internal Oscillator, 1 MHz)**




## 23.3. Supply Current of I/O Modules

Tables and formulas below can be used to calculate additional current consumption for the different I/O modules in Active and Idle mode. Enabling and disabling of I/O modules is controlled by the Power Reduction Register. See *Power Reduction Register* for details.

**Table 23-1. Additional Current Consumption for the different I/O modules (absolute values)**

| PRR bit | Typical numbers | | |
|---|---|---|---|
| | $V_{CC}$ = 2V, f = 1MHz | $V_{CC}$ = 3V, f = 4MHz | $V_{CC}$ = 5V, f = 8MHz |
| PRTIM0 | 6.6 µA | 40.0 µA | 153.0 µA |
| PRADC | 29.6 µA | 88.3 µA | 333.3 µA |

The table below can be used for calculating typical current consumption for other supply voltages and frequencies than those mentioned in the table above.

**Table 23-2. Additional Current Consumption (percentage) in Active and Idle mode**

| PRR bit | Current consumption additional to active mode with external clock | Current consumption additional to idle mode with external clock |
|---|---|---|
| PRTIM0 | 2.3 % | 10.4 % |
| PRADC | 6.7 % | 28.8 % |

## 23.4. Power-down Supply Current

**Figure 23-10. Power-down Supply Current vs. $V_{CC}$ (Watchdog Timer Disabled)**

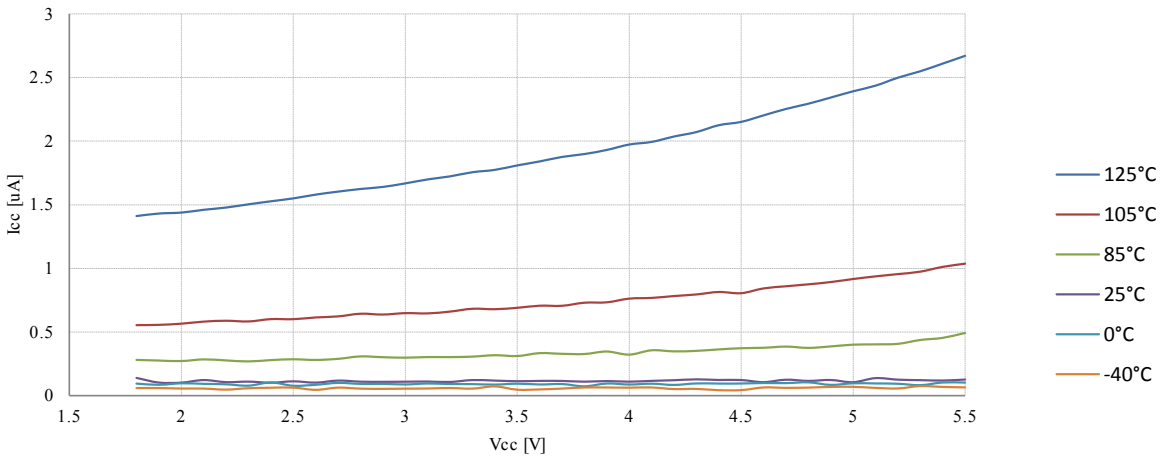
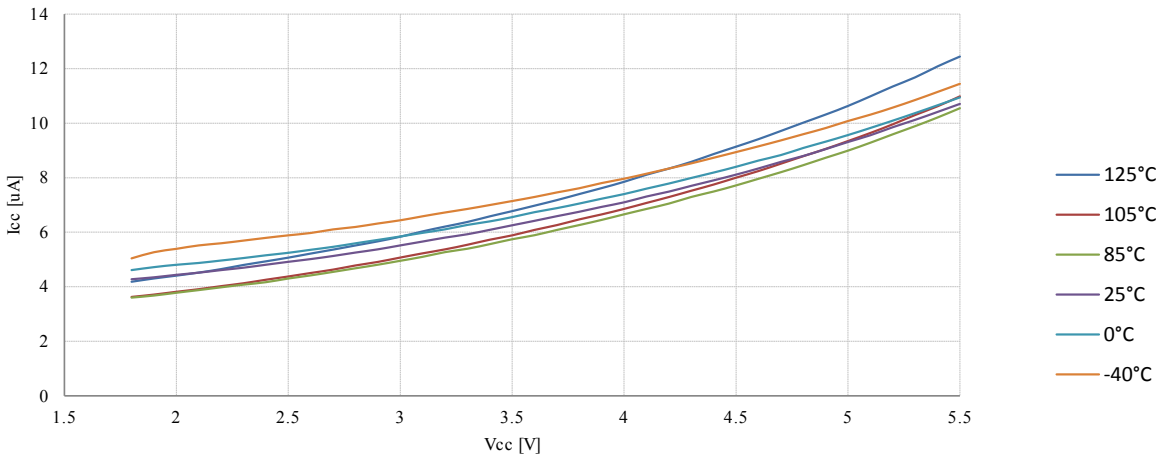



**Figure 23-11. Power-down Supply Current vs. $V_{CC}$ (Watchdog Timer Enabled)**



## 23.5. Pin Driver Strength

**Figure 23-12. I/O Pin Output Voltage vs. Sink Current ($V_{CC}$ = 1.8V)**

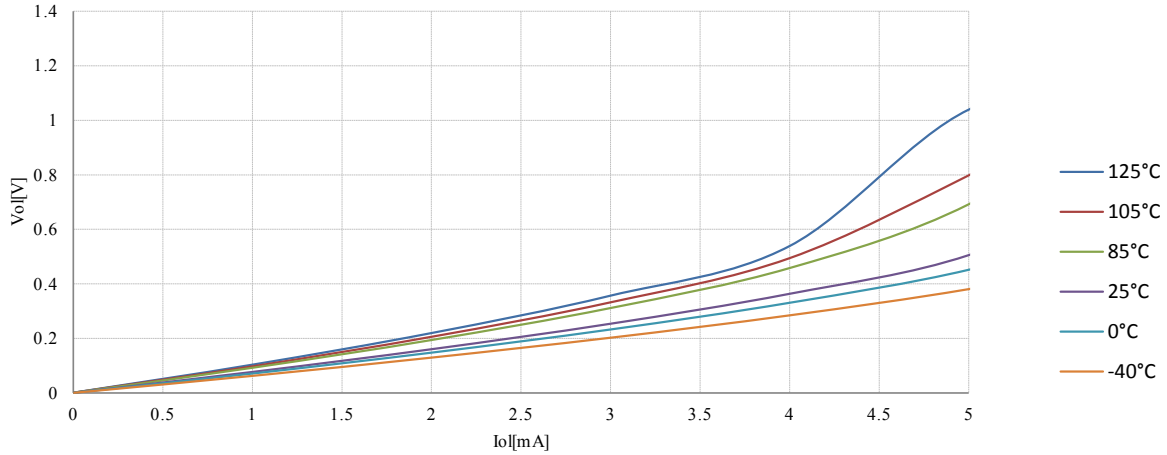



**Figure 23-13. I/O Pin Output Voltage vs. Sink Current ($V_{CC}$ = 3V)**

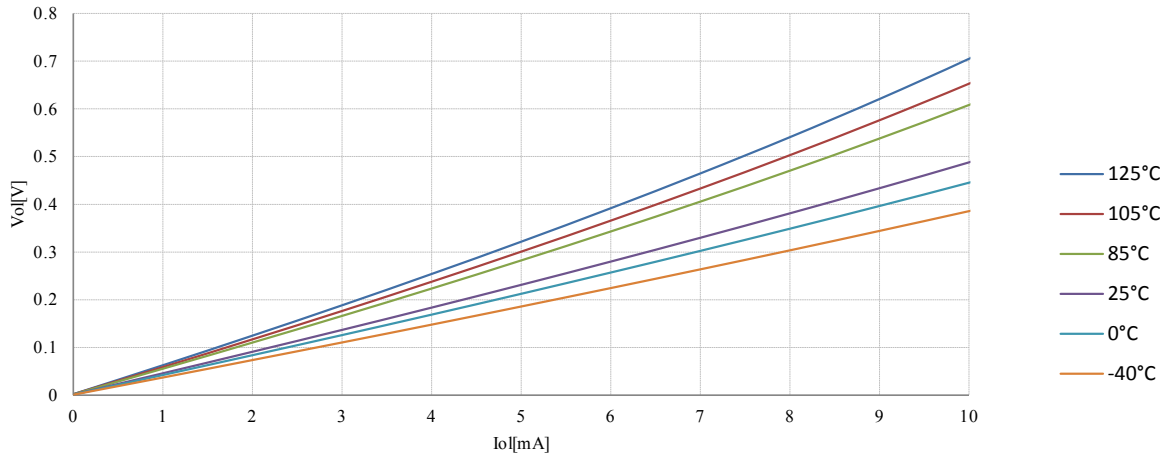


**Figure 23-14. I/O pin Output Voltage vs. Sink Current ($V_{CC}$ = 5V)**

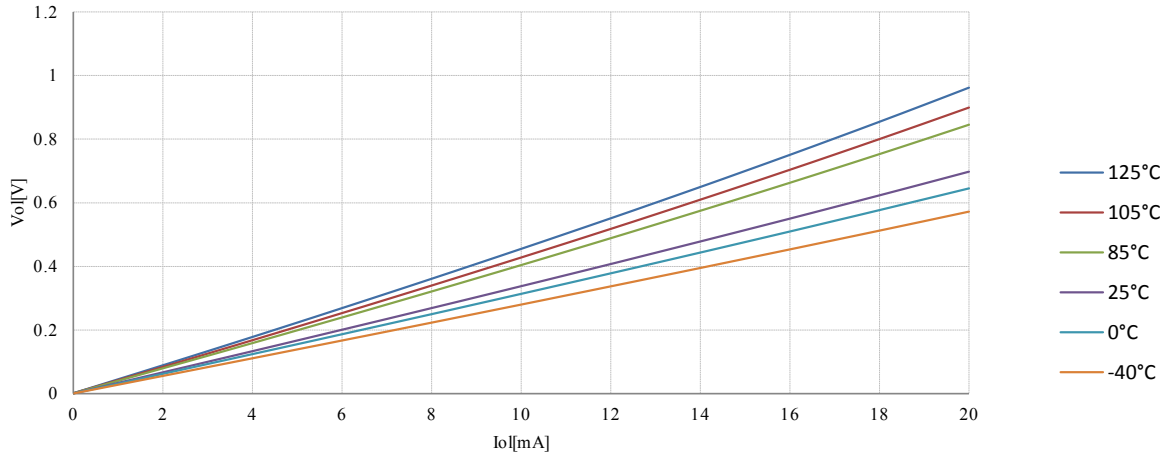



**Figure 23-15. I/O Pin Output Voltage vs. Source Current ($V_{CC}$ = 1.8V)**

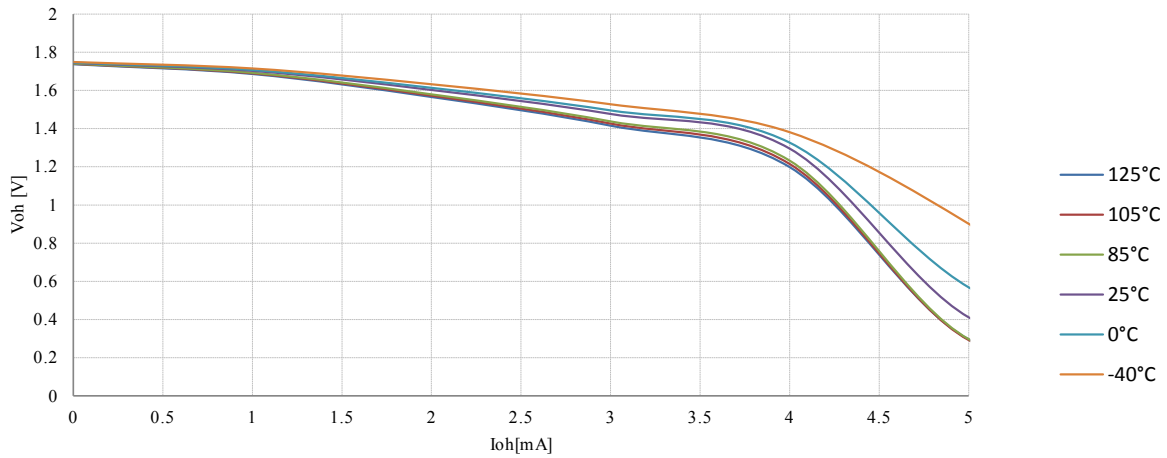


**Figure 23-16. I/O Pin Output Voltage vs. Source Current ($V_{CC}$ = 3V)**

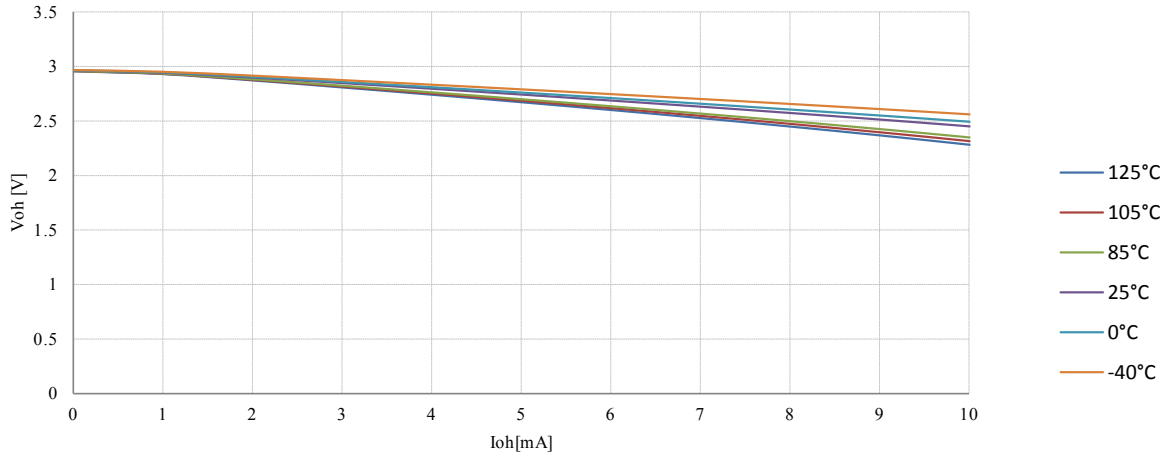



**Figure 23-17. I/O Pin output Voltage vs. Source Current ($V_{CC}$ = 5V)**

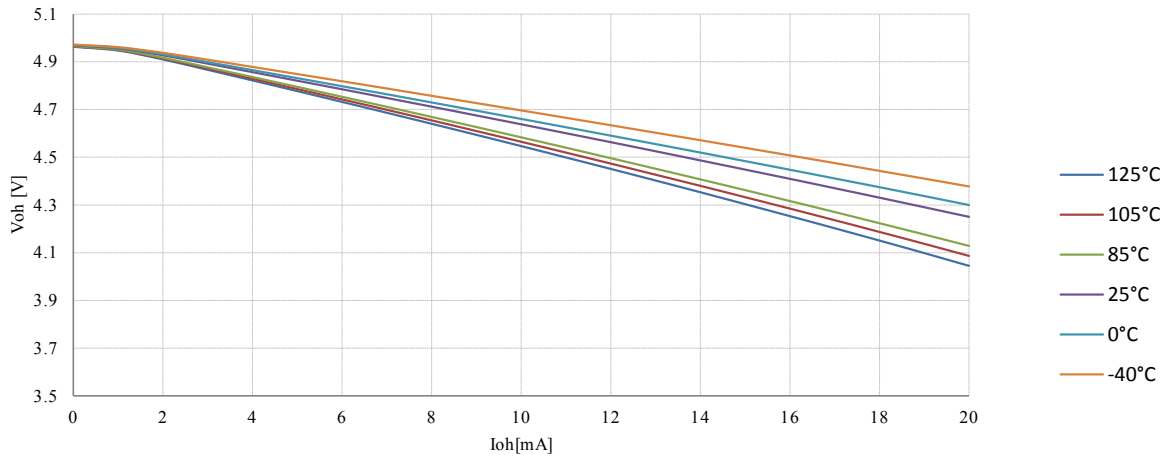


**Figure 23-18. Reset Pin as I/O, Output Voltage vs. Sink Current**

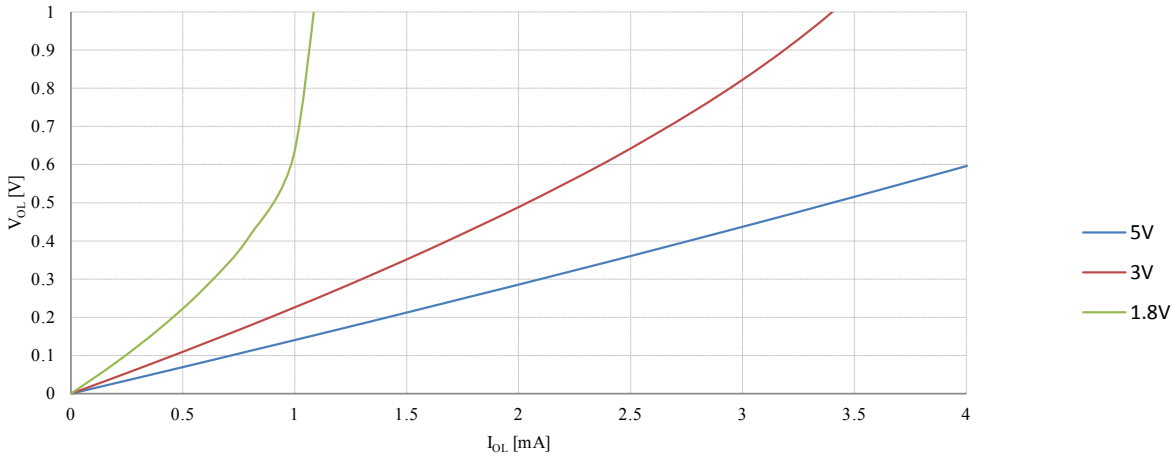
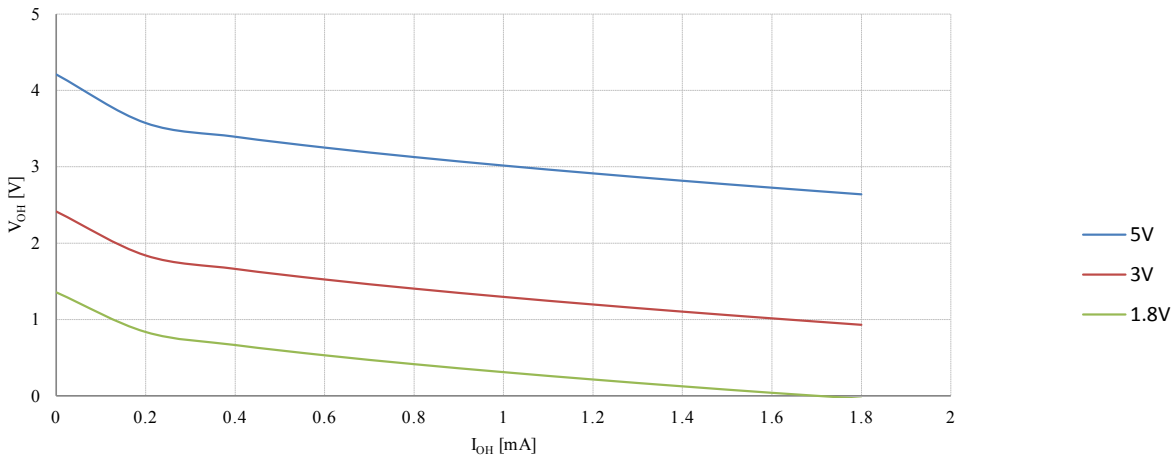



**Figure 23-19. Reset Pin as I/O, Output Voltage vs. Source Current**



## 23.6. Pin Threshold and Hysteresis

**Figure 23-20. I/O Pin Input Threshold Voltage vs. $V_{CC}$ ($V_{IH}$, IO Pin Read as '1')**

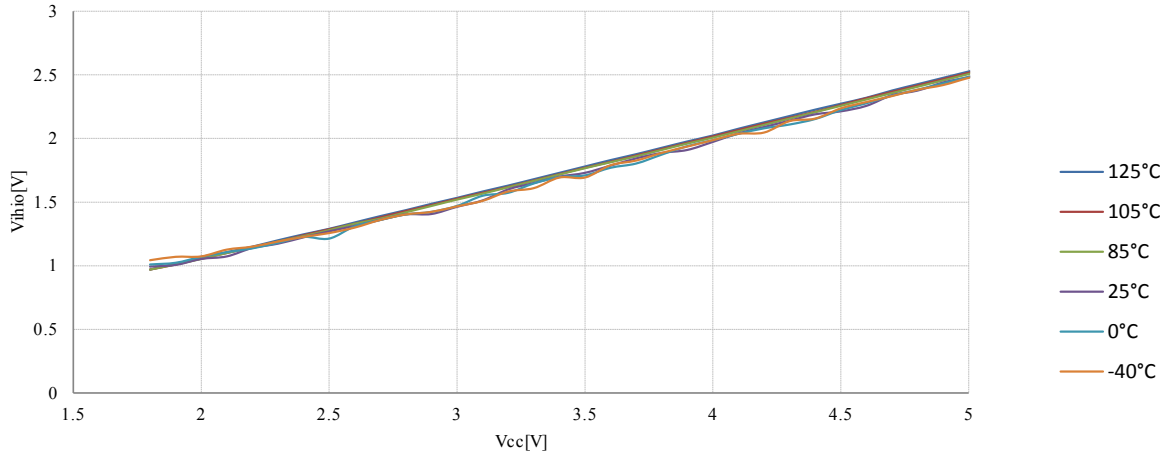



**Figure 23-21. I/O Pin Input threshold Voltage vs. $V_{CC}$ ($V_{IL}$, IO Pin Read as '0')**

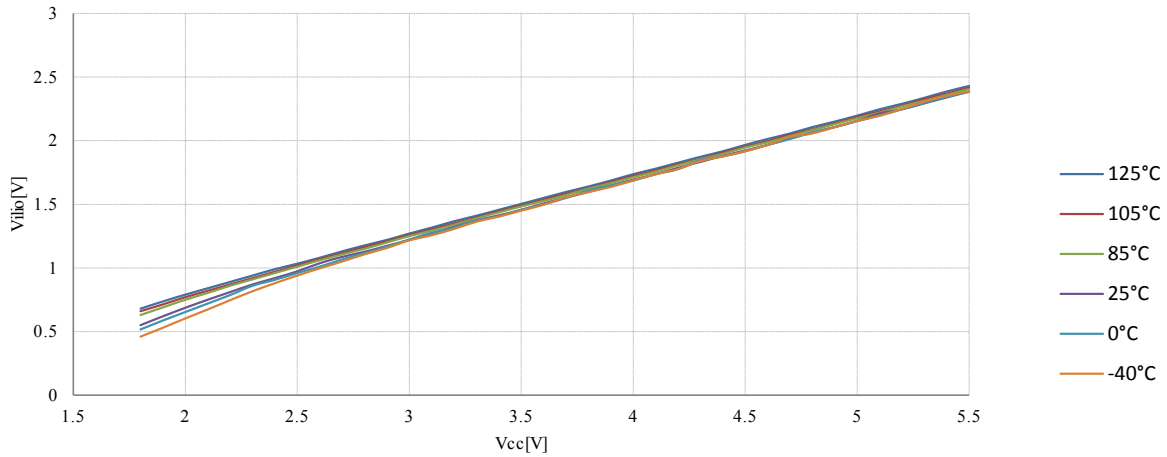


**Figure 23-22. I/O Pin Input Hysteresis vs. $V_{CC}$**

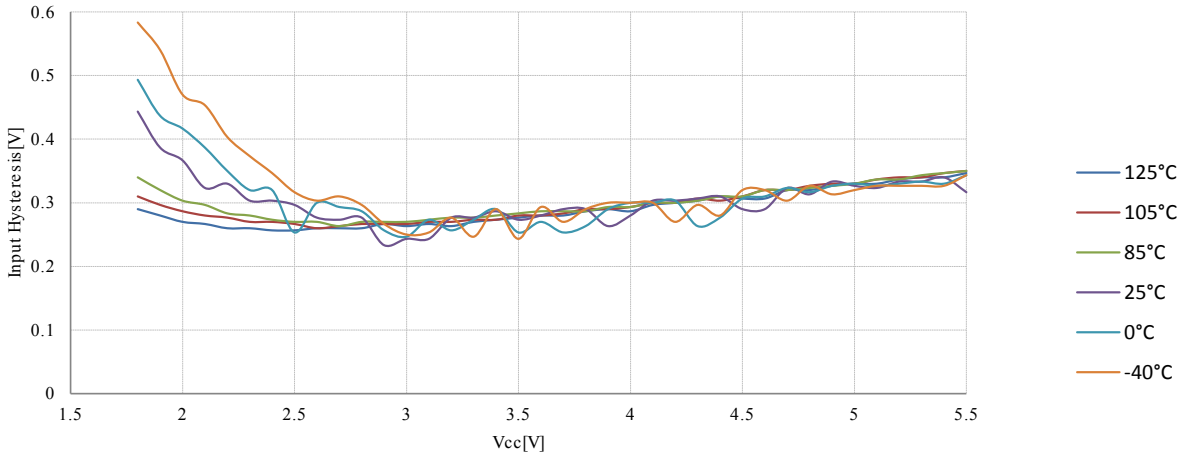



**Figure 23-23. Reset Pin as I/O, Input Threshold Voltage vs. $V_{CC}$ ($V_{IH}$, I/O Pin Read as '1')**

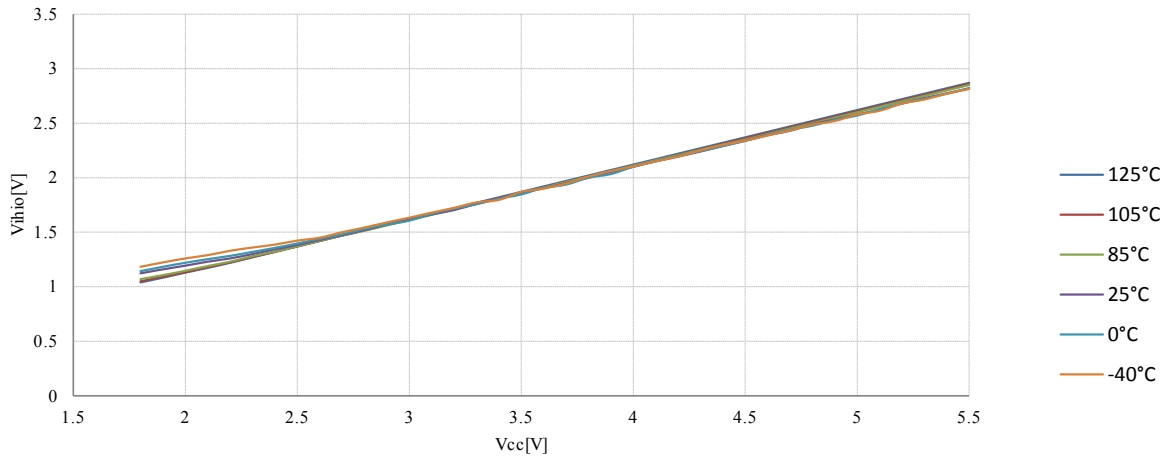


**Figure 23-24. Reset Pin as I/O, Input Threshold Voltage vs. $V_{CC}$ ($V_{IL}$, I/O pin Read as '0')**

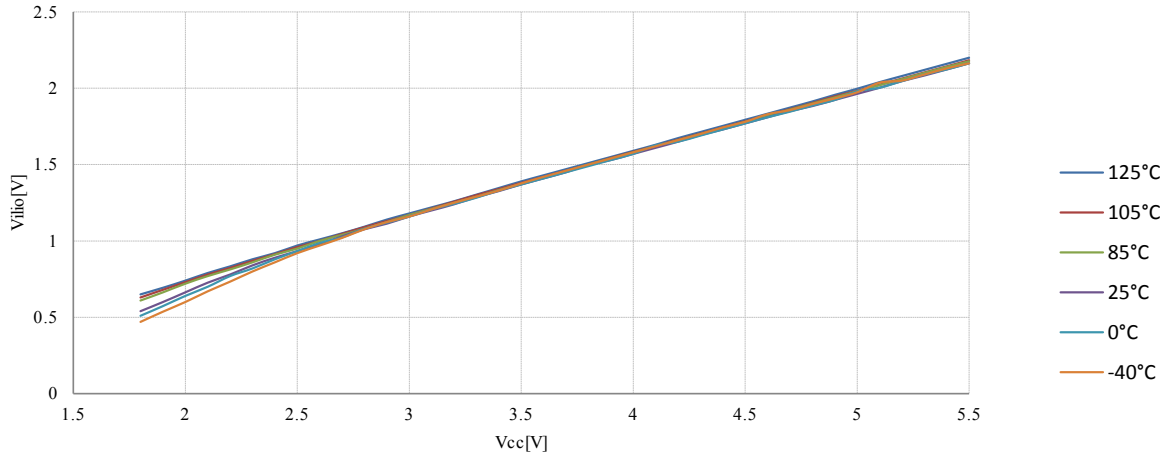



**Figure 23-25. Reset Input Hysteresis vs. $V_{CC}$ (Reset Pin Used as I/O)**

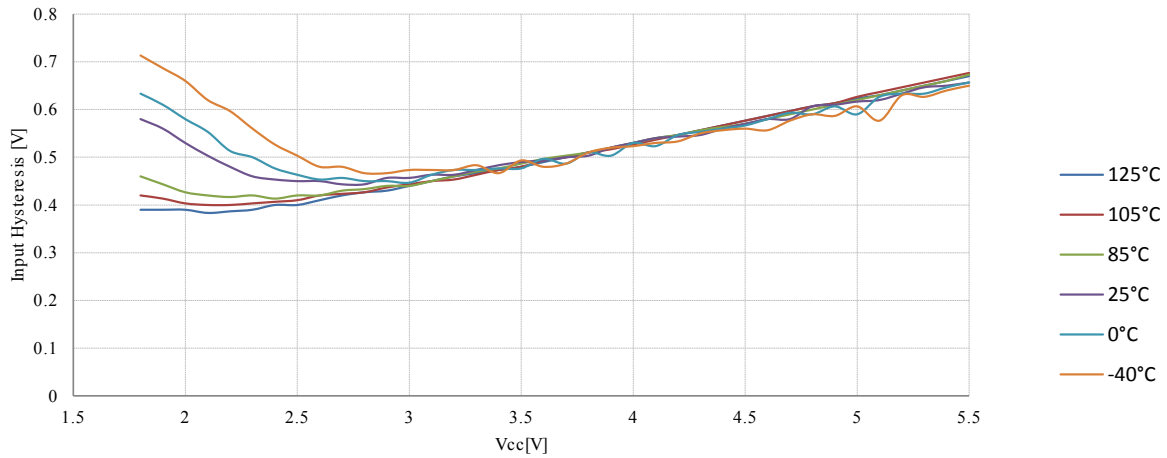


**Figure 23-26. Reset Input Threshold Voltage vs. $V_{CC}$ ($V_{IH}$, I/O Pin Read as '1')**

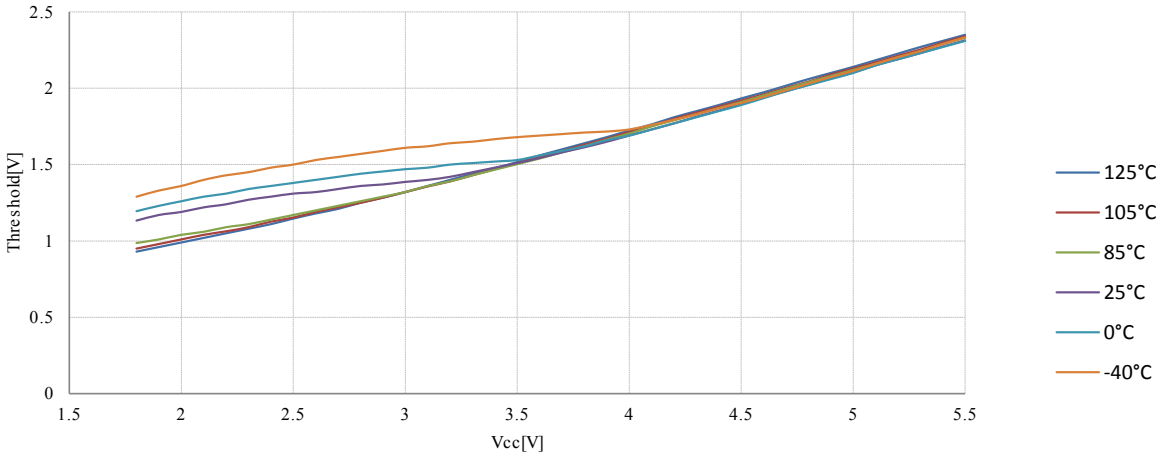



**Figure 23-27. Reset Input Threshold Voltage vs. $V_{CC}$ ($V_{IL}$, I/O pin Read as '0')**

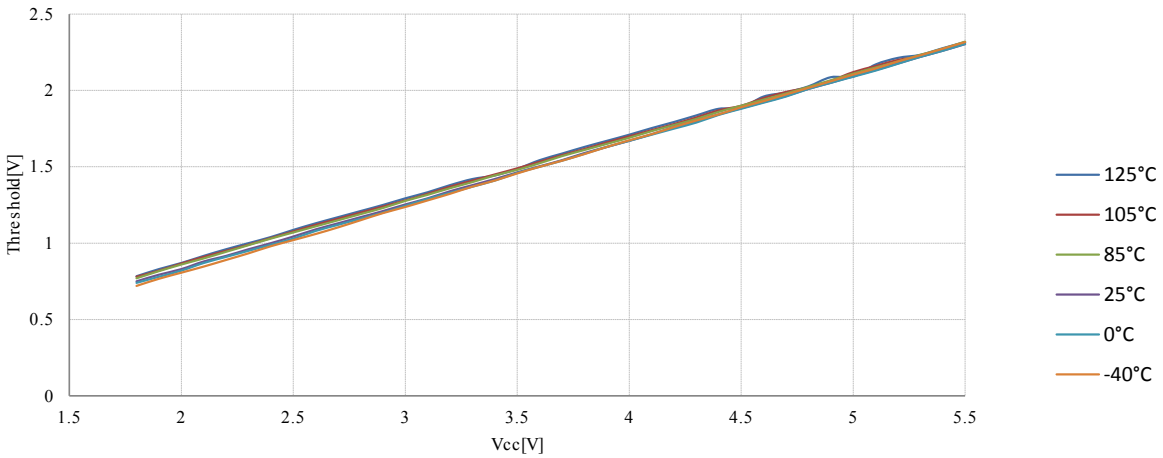
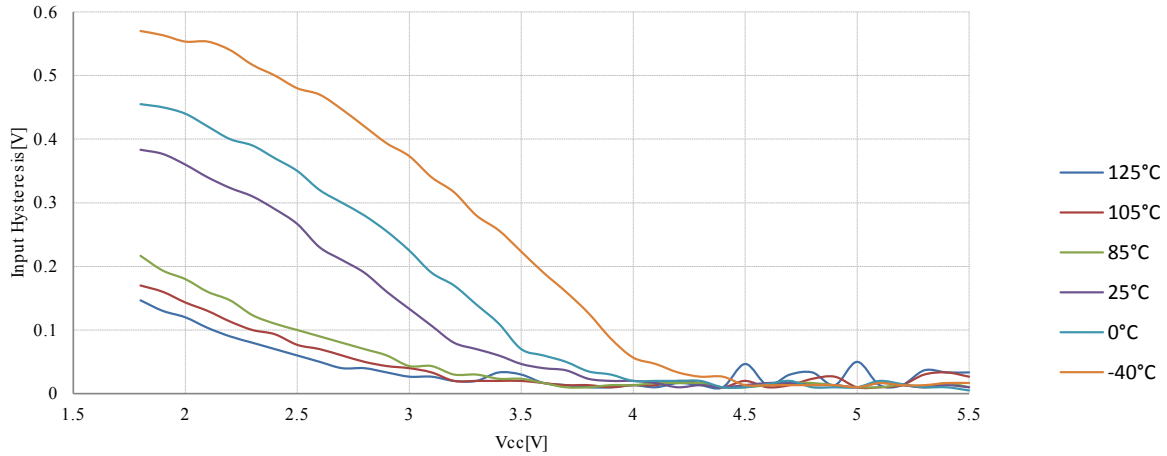
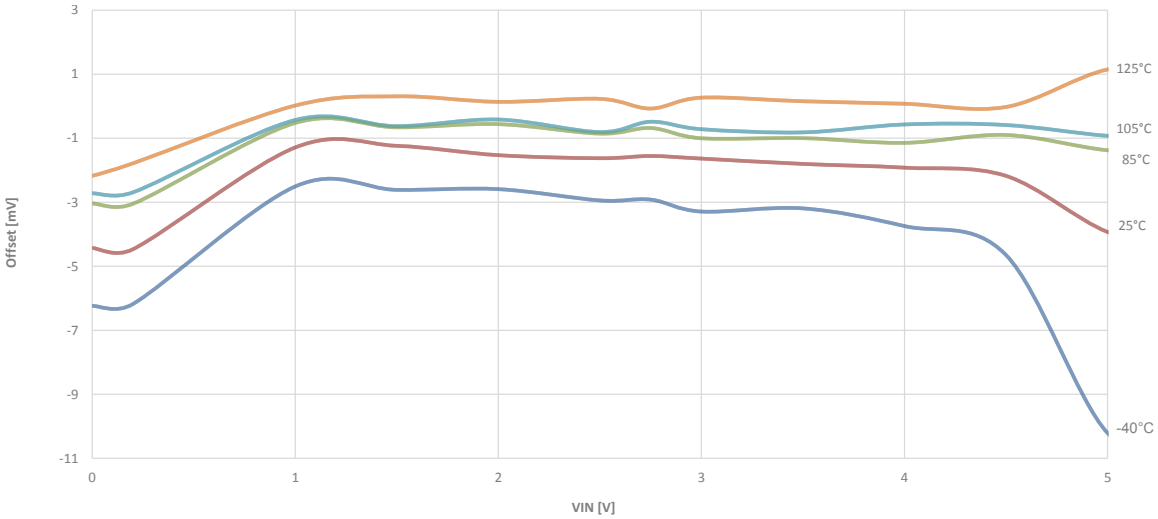


**Figure 23-28. Reset Pin, Input Hysteresis vs. $V_{CC}$**




## 23.7. Analog Comparator Offset

**Figure 23-29. Analog Comparator Offset**



## 23.8. Pin Pull-up

**Figure 23-30. I/O pin Pull-up Resistor Current vs. Input Voltage ($V_{CC}$ = 1.8V)**

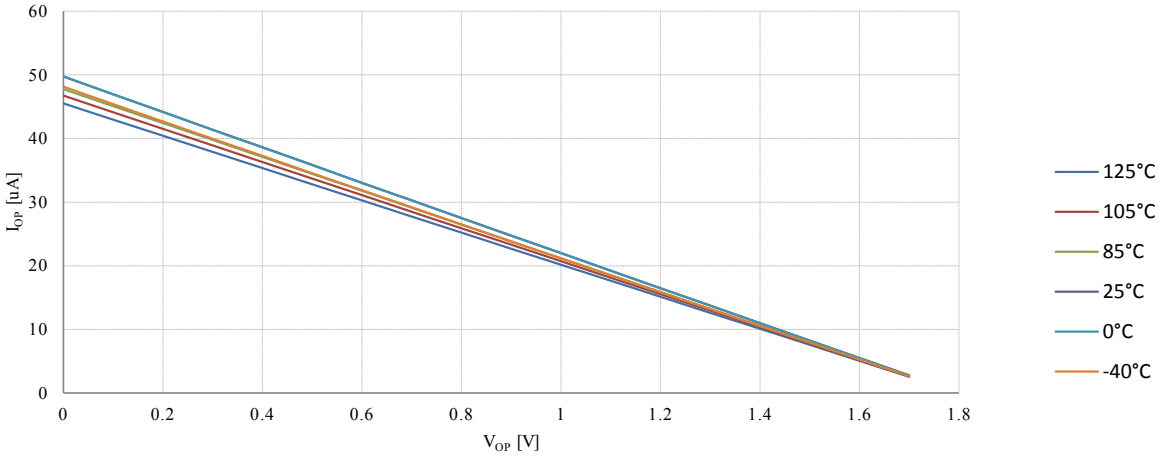



**Figure 23-31. I/O Pin Pull-up Resistor Current vs. input Voltage ($V_{CC}$ = 2.7V)**

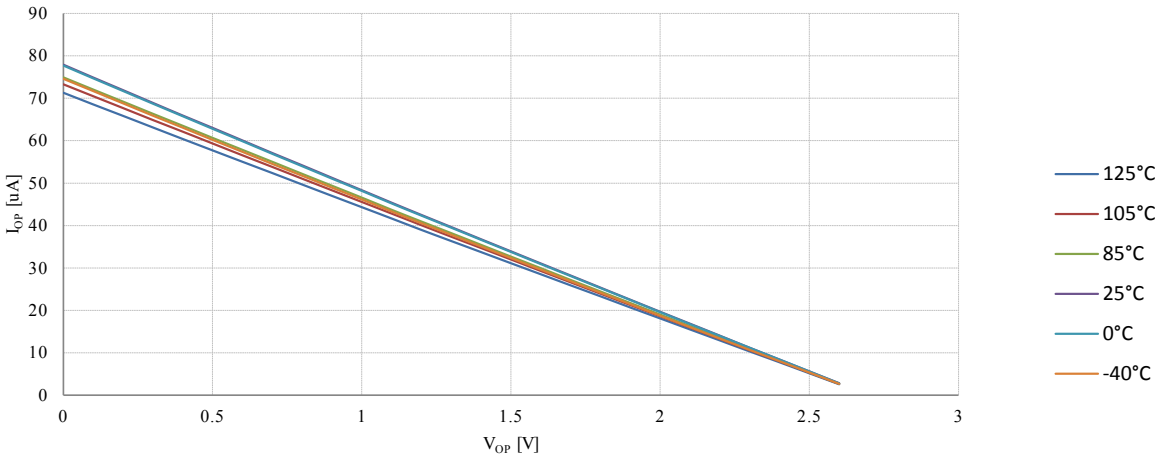


**Figure 23-32. I/O pin Pull-up Resistor Current vs. Input Voltage ($V_{CC}$ = 5V)**

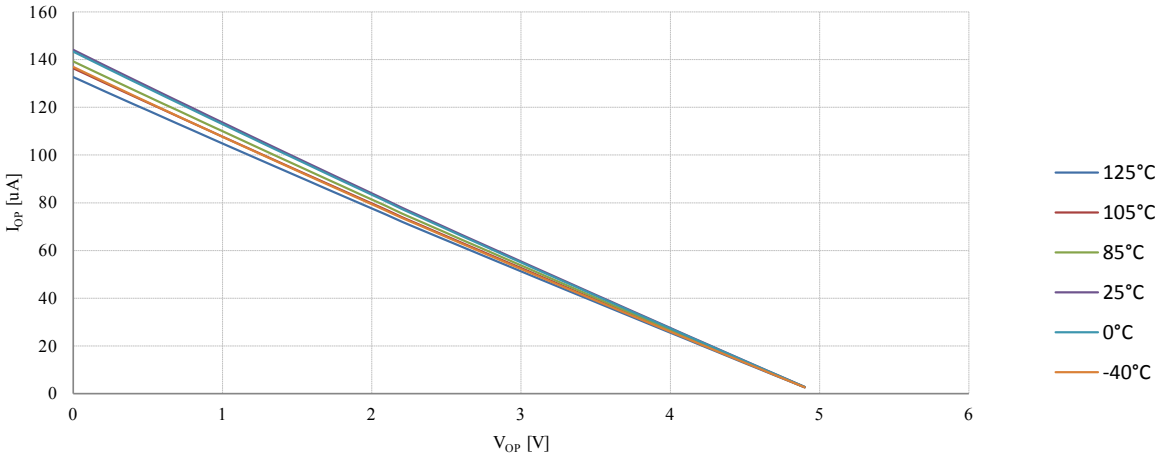



**Figure 23-33. Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC}$ = 1.8V)**

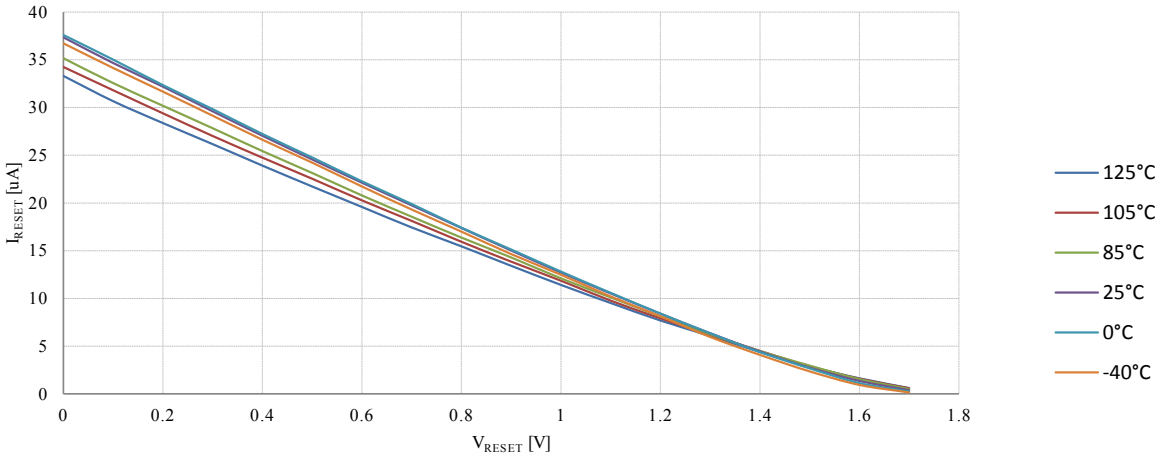


**Figure 23-34. Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC}$ = 2.7V)**

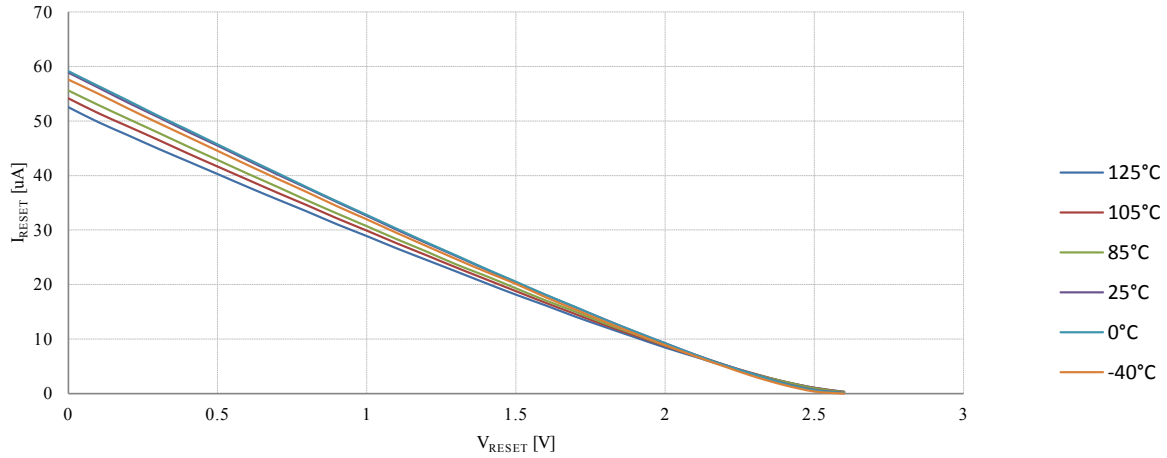
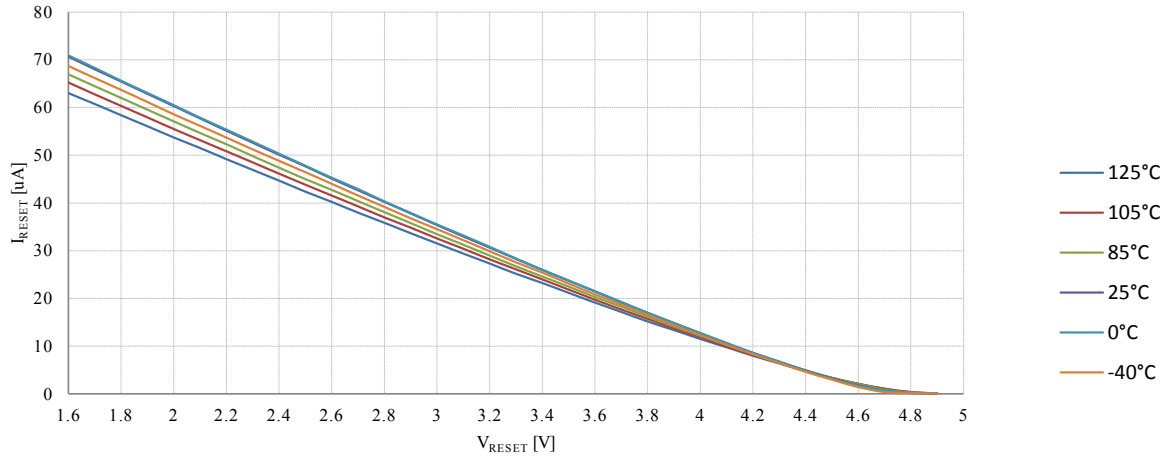



**Figure 23-35. Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC}$ = 5V)**



## 23.9. Internal Oscillator Speed

**Figure 23-36. Watchdog Oscillator Frequency vs. $V_{CC}$**

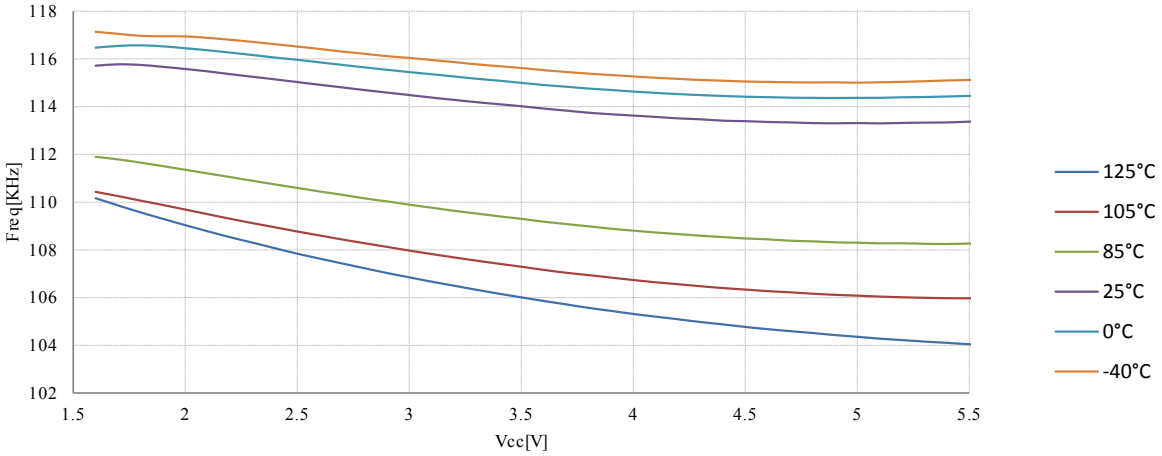



**Figure 23-37. Watchdog Oscillator Frequency vs. Temperature**

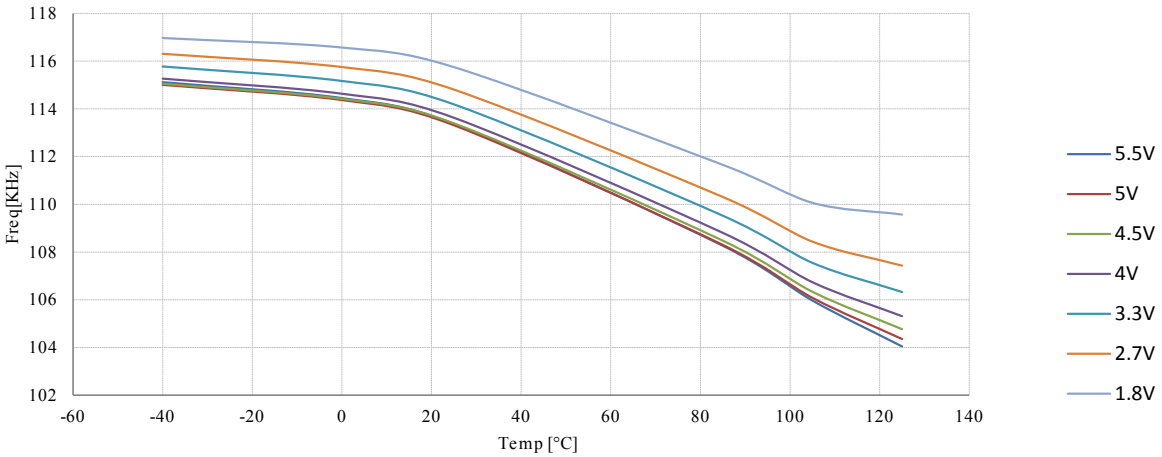


**Figure 23-38. Calibrated Oscillator Frequency vs. $V_{CC}$**

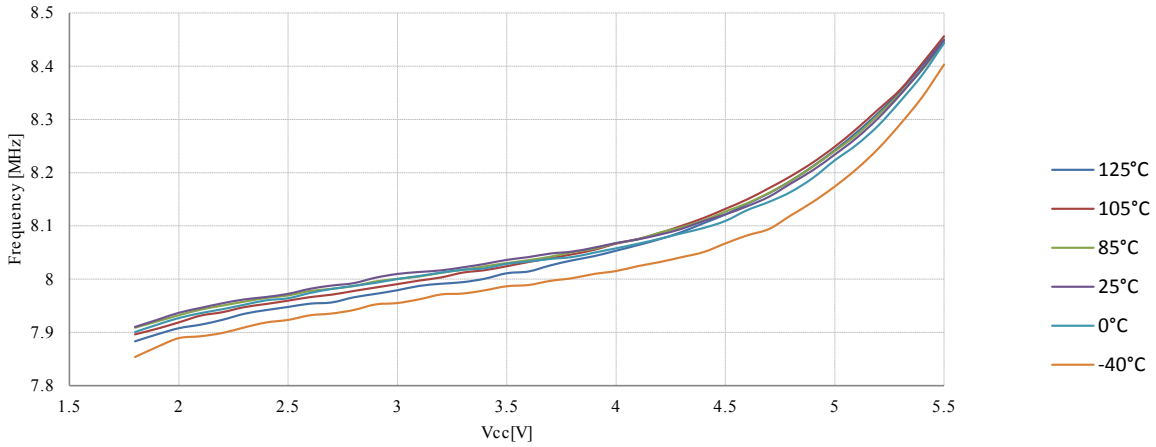



**Figure 23-39. Calibrated Oscillator Frequency vs. Temperature**

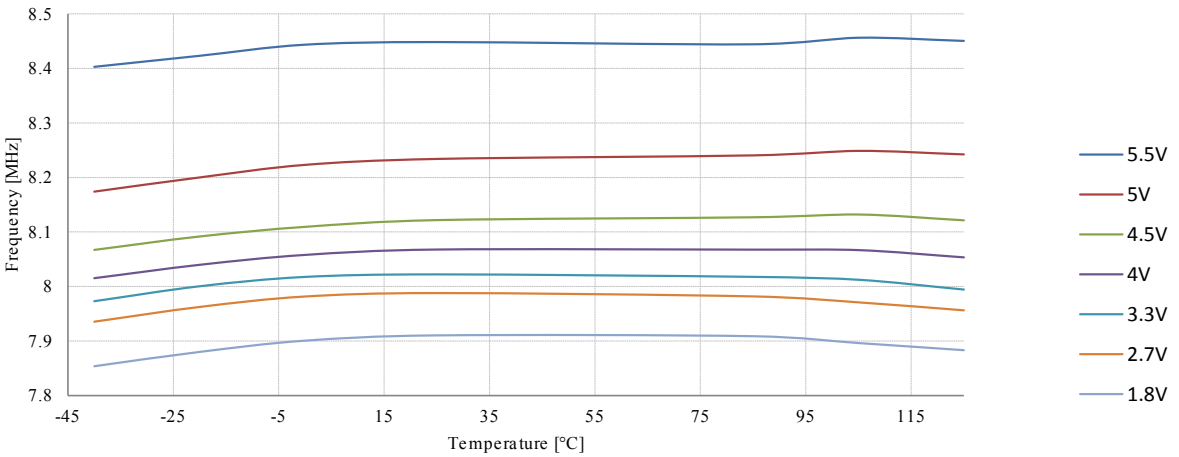
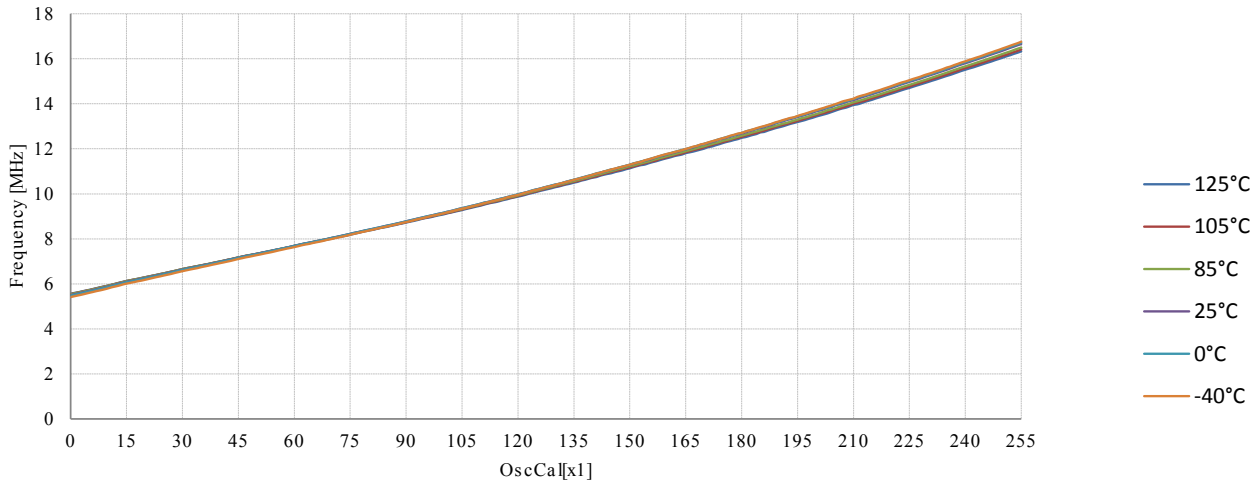


**Figure 23-40. Calibrated Oscillator Frequency vs. OSCCAL Value**




## 23.10. VLM Thresholds

**Figure 23-41. VLM1L Threshold of $V_{CC}$ Level Monitor**

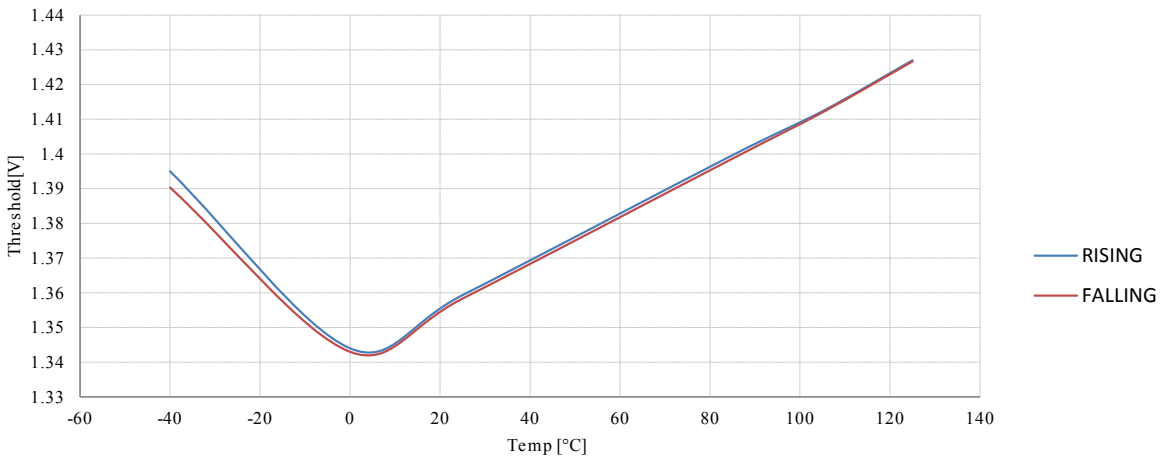


**Figure 23-42. VLM1H Threshold of $V_{CC}$ Level Monitor**

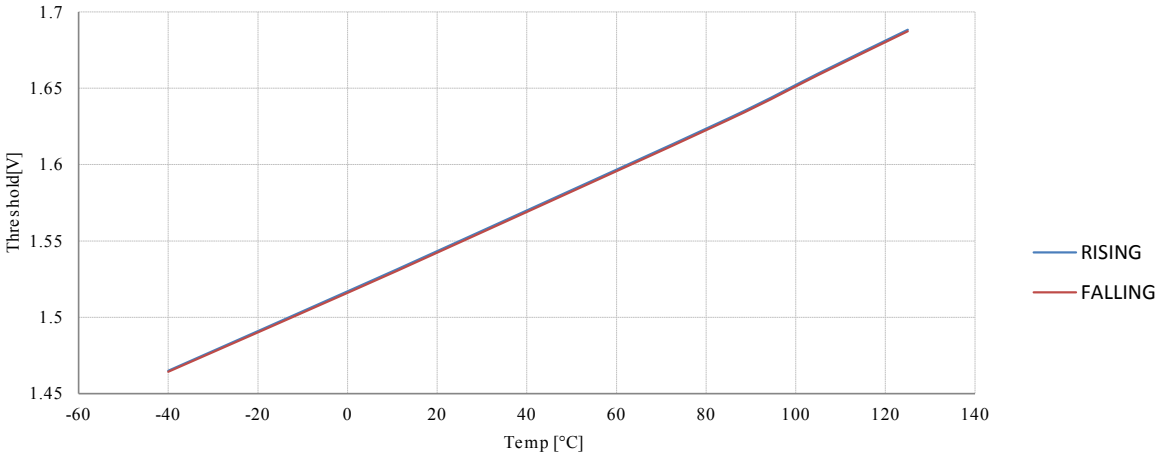



**Figure 23-43. VLM2 Threshold of $V_{CC}$ Level Monitor**

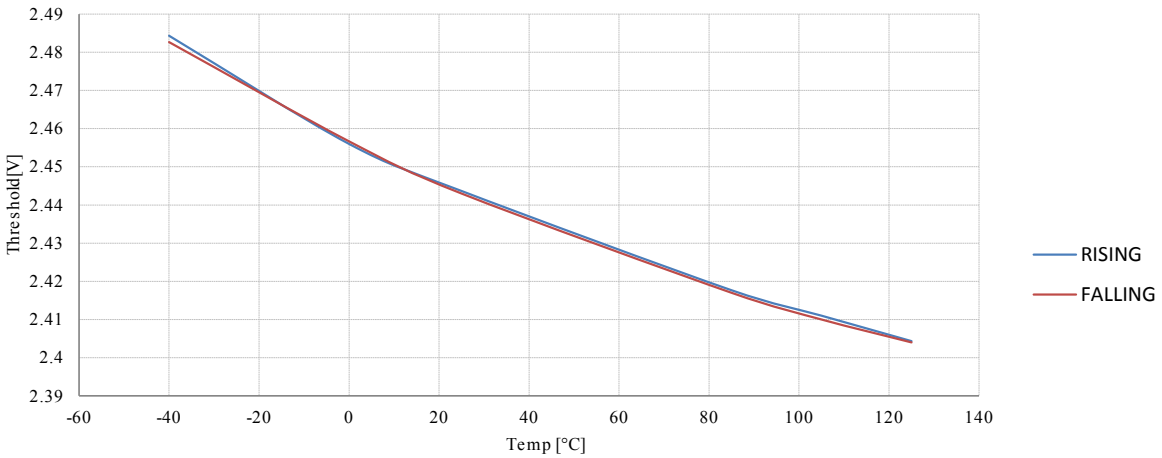
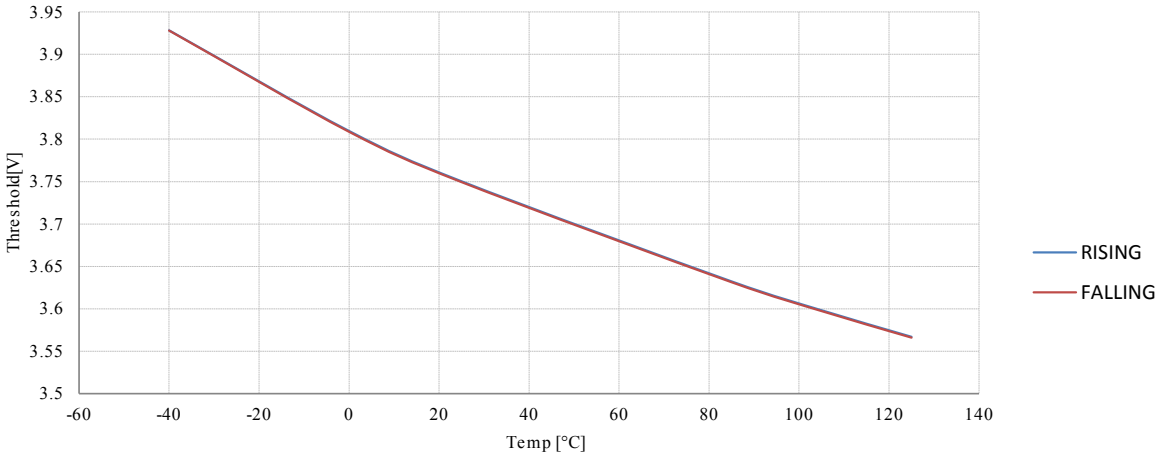


**Figure 23-44. VLM3 Threshold of $V_{CC}$ Level Monitor**




## 23.11. Current Consumption of Peripheral Units

**Figure 23-45. ADC Current vs. $V_{CC}$**

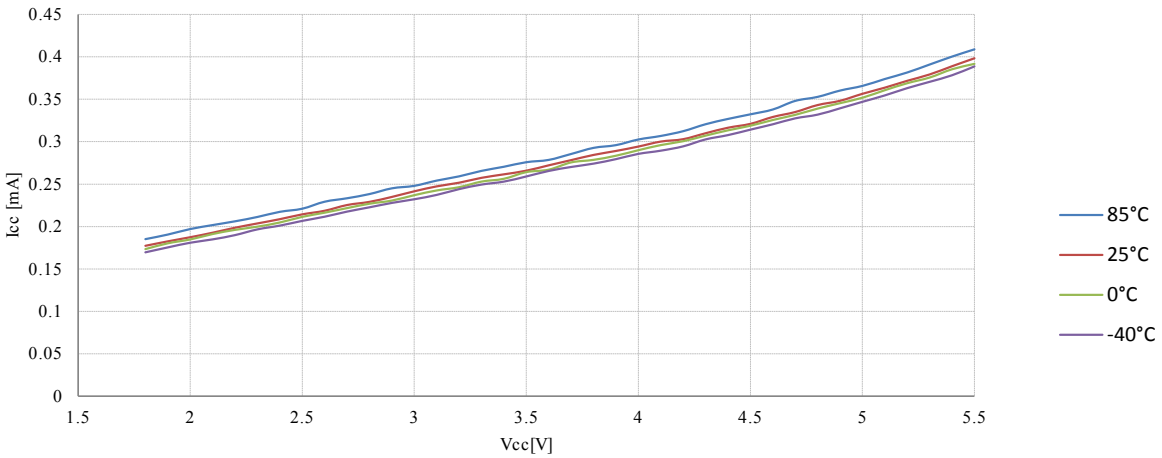


**Figure 23-46. Analog Comparator (AC) Current vs. $V_{CC}$**

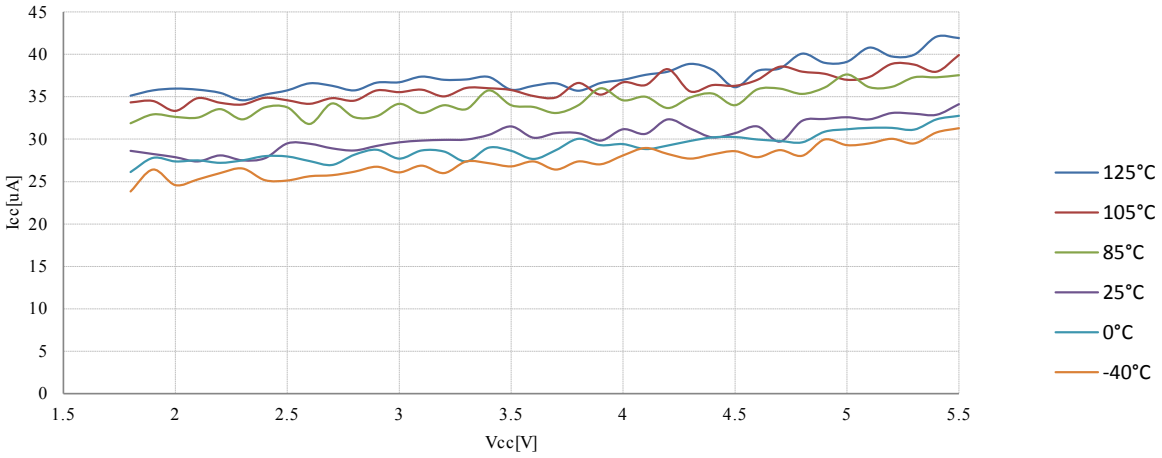



**Figure 23-47. $V_{CC}$ Level Monitor Current vs. $V_{CC}$**

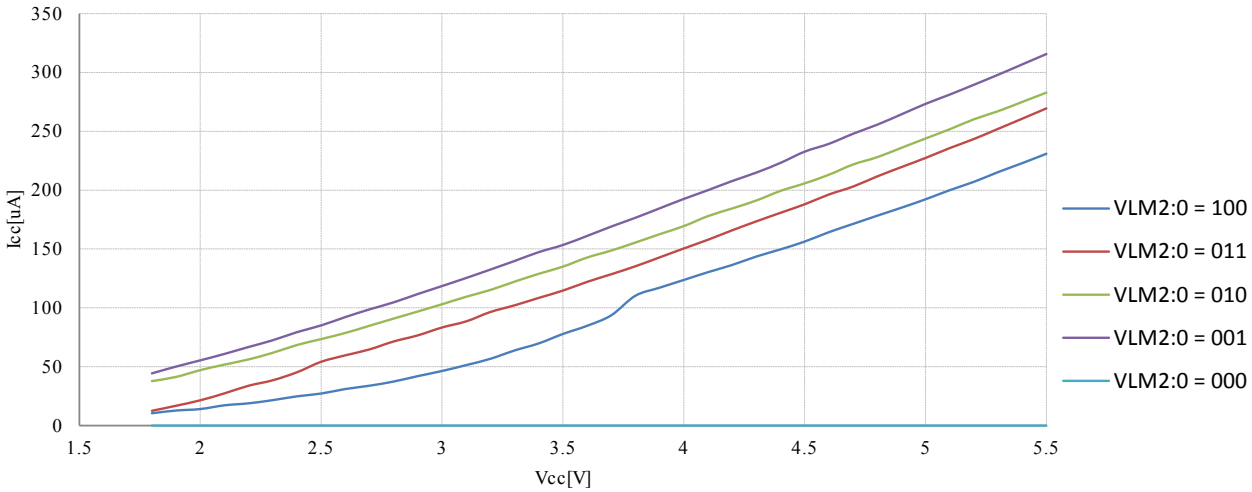


**Figure 23-48. Temperature Dependence of VLM Current vs. $V_{CC}$**

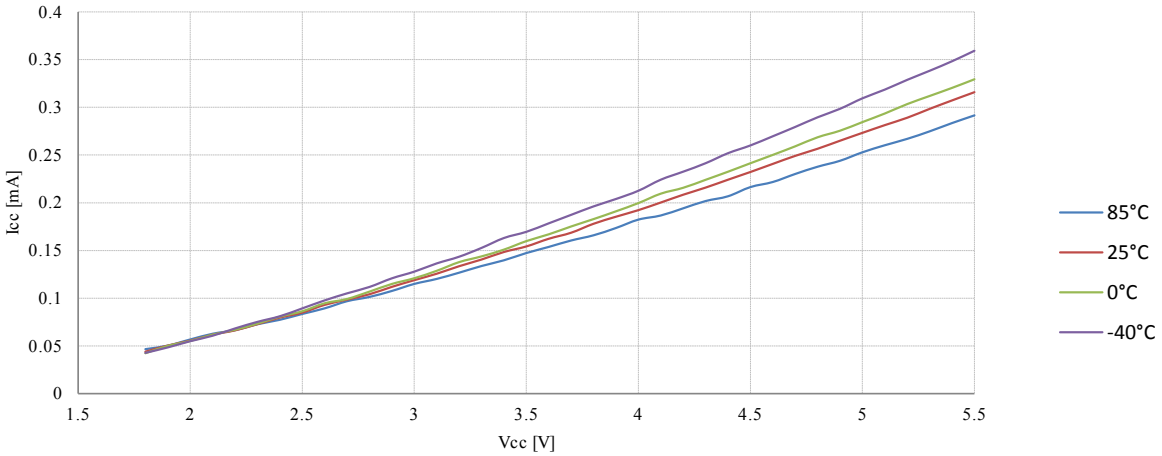
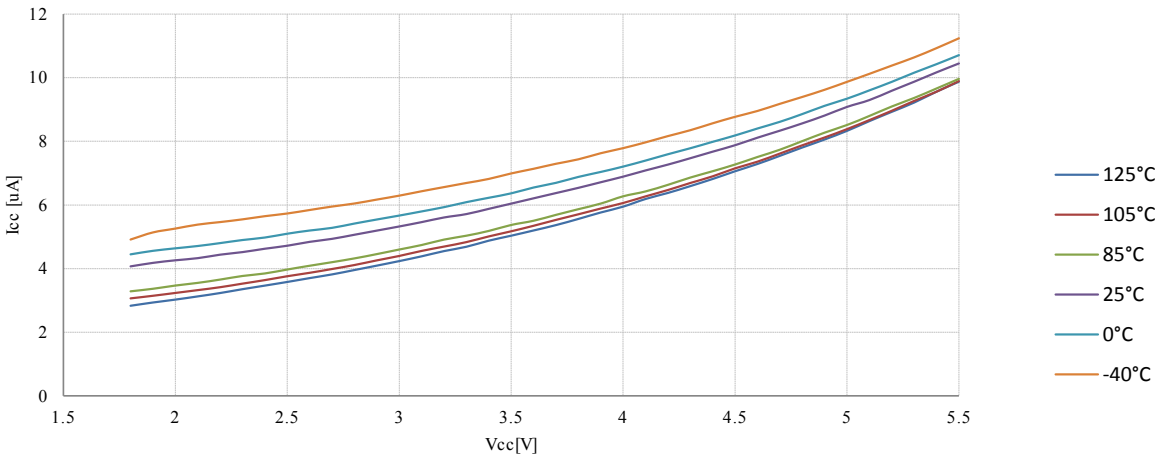
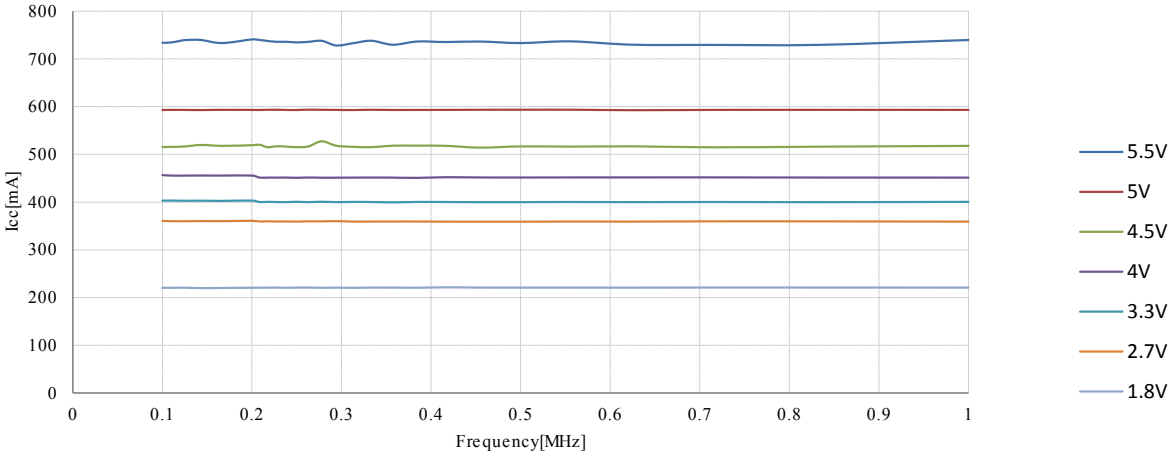



**Figure 23-49. Watchdog Timer Current vs. $V_{CC}$**



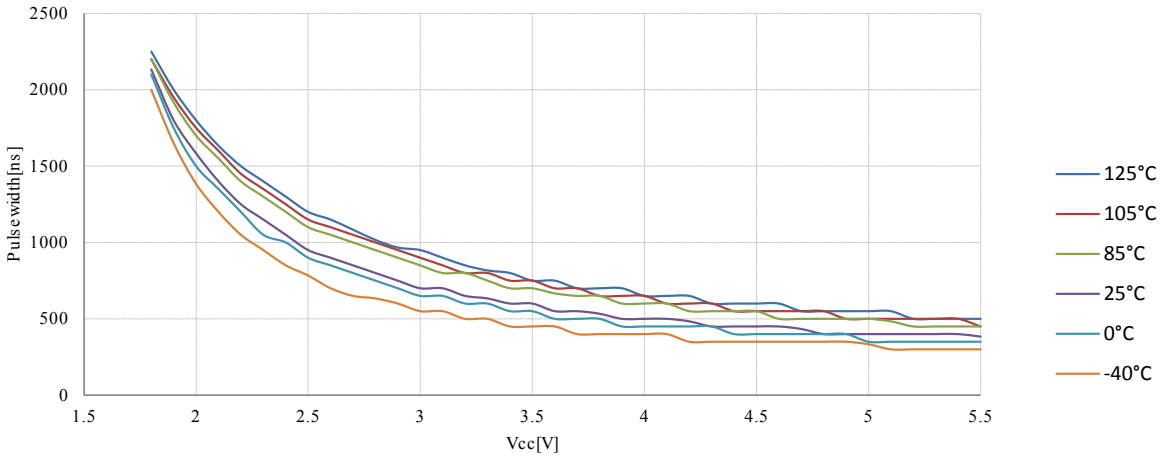
## 23.12. Current Consumption in Reset and Reset Pulsewidth

**Figure 23-50. Reset Supply Current vs. $V_{CC}$ (0.1 - 1.0 MHz, excluding Current Through the Reset Pull-up)**




**Note:** The default clock source for the device is always the internal 8 MHz oscillator. Hence, current consumption in reset remains unaffected by external clock signals.

**Figure 23-51. Minimum Reset Pulse Width vs. $V_{CC}$**



# 24. Register Summary

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | PINA | 7:0 | PINA7 | PINA6 | PINA5 | PINA4 | PINA3 | PINA2 | PINA1 | PINA0 |
| 0x01 | DDRA | 7:0 | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| 0x02 | PORTA | 7:0 | PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |
| 0x03 | PUEA | 7:0 | PUEA7 | PUEA6 | PUEA5 | PUEA4 | PUEA3 | PUEA2 | PUEA1 | PUEA0 |
| 0x04 | PINB | 7:0 | | | | | PINB3 | PINB2 | PINB1 | PINB0 |
| 0x05 | DDRB | 7:0 | | | | | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| 0x06 | PORTB | 7:0 | | | | | PORTB3 | PORTB2 | PORTB1 | PORTB0 |
| 0x07 | PUEB | 7:0 | | | | | PUEB3 | PUEB2 | PUEB1 | PUEB0 |
| 0x08 | UDR0 | 7:0 | TXB / RXB[7:0] | | | | | | | |
| 0x09 | UBRR0L and | 7:0 | UBRR0[7:0] | | | | | | | |
| 0x0A | UBRR0H | 15:8 | | | | | UBRR0[11:8] | | | |
| 0x0B | UCSR0D | 7:0 | RXSIE | RXS | SFDE | | | | | |
| 0x0C | UCSR0C | 7:0 | UMSEL01 | UMSEL00 | UPM01 | UPM00 | USBS0 | UCSZ01 / UDORD0 | UCSZ00 / UCPHA0 | UCPOL0 |
| 0x0D | UCSR0B | 7:0 | RXCIE0 | TXCIE0 | UDRIE0 | RXEN0 | TXEN0 | UCSZ02 | RXB80 | TXB80 |
| 0x0E | UCSR0A | 7:0 | RXC0 | TXC0 | UDRE0 | FE0 | DOR0 | UPE0 | U2X0 | MPCM0 |
| 0x0F | PCMSK0 | 7:0 | PCINT7 | PCINT6 | PCINT5 | PCINT4 | PCINT3 | PCINT2 | PCINT1 | PCINT0 |
| 0x10 | PCMSK1 | 7:0 | | | | | PCINT11 | PCINT10 | PCINT9 | PCINT8 |
| 0x11 | PCIFR | 7:0 | | | | | | | PCIF1 | PCIF0 |
| 0x12 | PCICR | 7:0 | | | | | | | PCIE1 | PCIE0 |
| 0x13 | EIMSK | 7:0 | | | | | | | | INT0 |
| 0x14 | EIFR | 7:0 | | | | | | | | INTF0 |
| 0x15 | EICRA | 7:0 | | | | | | | ISC0[1:0] | |
| 0x16 | PORTCR | 7:0 | | | | | | | BBMB | BBMA |
| 0x17 | DIDR0 | 7:0 | ADC7D | ADC6D | ADC5D | ADC4D | ADC3D | ADC2D | ADC1D | ADC0D |
| 0x18 | Reserved | | | | | | | | | |
| 0x19 | ADCL and ADCH | 7:0 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADC1 | ADC0 |
| 0x1A | | 15:8 | | | | | | | ADC9 | ADC8 |
| 0x1B | ADMUX | 7:0 | REFS1 | REFS0 | | | | MUX2 | MUX1 | MUX0 |
| 0x1C | ADCSRB | 7:0 | ADLAR | | | | | ADTS2 | ADTS1 | ADTS0 |
| 0x1D | ADCSRA | 7:0 | ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 |
| 0x1E | ACSRB | 7:0 | | | | | | | ACOE | ACPMUX |
| 0x1F | ACSRA | 7:0 | ACD | ACBG | ACO | ACI | ACIE | ACIC | ACIS1 | ACIS0 |
| 0x20 ... 0x21 | Reserved | | | | | | | | | |
| 0x22 | ICR0L and ICR0H | 7:0 | ICR0[7:0] | | | | | | | |
| 0x23 | | 15:8 | ICR0[15:8] | | | | | | | |
| 0x24 | OCR0BL and | 7:0 | OCR0B[7:0] | | | | | | | |
| 0x25 | OCR0BH | 15:8 | OCR0B[15:8] | | | | | | | |
| 0x26 | OCR0AL and | 7:0 | OCR0A[7:0] | | | | | | | |
| 0x27 | OCR0AH | 15:8 | OCR0A[15:8] | | | | | | | |
| 0x28 | TCNT0L and | 7:0 | TCNT0[7:0] | | | | | | | |
| 0x29 | TCNT0H | 15:8 | TCNT0[15:8] | | | | | | | |

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x2A | TIFR0 | 7:0 | | | ICF0 | | | OCF0B | OCF0A | TOV0 |
| 0x2B | TIMSK0 | 7:0 | | | ICIE0 | | | OCIE0B | OCIE0A | TOIE0 |
| 0x2C | TCCR0C | 7:0 | FOC0A | FOC0B | | | | | | |
| 0x2D | TCCR0B | 7:0 | ICNC0 | ICES0 | | WGM03 | WGM02 | CS0[2:0] | | |
| 0x2E | TCCR0A | 7:0 | COM0A1 | COM0A0 | COM0B1 | COM0B0 | | | WGM01 | WGM00 |
| 0x2F | GTCCR | 7:0 | TSM | | | | | | REMAP | PSR |
| 0x30 | Reserved | | | | | | | | | |
| 0x31 | WDTCSR | 7:0 | WDIF | WDIE | WDP3 | | WDE | WDP2 | WDP1 | WDP0 |
| 0x32 | NVMCSR | 7:0 | NVMBSY | | | | | | | |
| 0x33 | NVMCMD | 7:0 | | | NVMCMD[5:0] | | | | | |
| 0x34 | VLMCSR | 7:0 | VLMF | VLMIE | | | | VLM[2:0] | | |
| 0x35 | PRR | 7:0 | | | | | | PRUSART0 | PRADC | PRTIM0 |
| 0x36 | CLKPSR | 7:0 | | | | | CLKPS[3:0] | | | |
| 0x37 | CLKMSR | 7:0 | | | | | | | CLKMS[1:0] | |
| 0x38 | Reserved | | | | | | | | | |
| 0x39 | OSCCAL | 7:0 | CAL[7:0] | | | | | | | |
| 0x3A | SMCR | 7:0 | | | | | SM[2:0] | | | SE |
| 0x3B | RSTFLR | 7:0 | | | | | WDRF | | EXTRF | PORF |
| 0x3C | CCP | 7:0 | CCP[7:0] | | | | | | | |
| 0x3D | SPL and SPH | 7:0 | | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |
| 0x3E | | 15:8 | | | | | | | | |
| 0x3F ... 0x5E | Reserved | | | | | | | | | |
| 0x5F | SREG | 7:0 | I | T | H | S | V | N | Z | C |

## 24.1. Note

- USART registers 0x08-0x0E are NOT bit accessible using SBI/CBI instructions

# 25. Instruction Set Summary

**Table 24-1. Arithmetic and Logic Instructions**

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| 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 | ← | Rd + 1:Rd + K | Z,C,N,V,S | N/A |
| 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 | N/A |
| 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 |

**Table 24-2. Branch Instructions**

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| RJMP | k | Relative Jump | PC | ← | PC + k + 1 | None | 2 |
| IJMP | | Indirect Jump to (Z) | PC(15:0)<br>PC(21:16) | ←<br>← | Z<br>0 | None | 2 |
| RCALL | k | Relative Call Subroutine | PC | ← | PC + k + 1 | None | 3(1) |
| ICALL | | Indirect Call to (Z) | PC(15:0)<br>PC(21:16) | ←<br>← | Z<br>0 | None | 3(1) |
| RET | | Subroutine Return | PC | ← | STACK | None | 6(1) |
| RETI | | Interrupt Return | PC | ← | STACK | I | 6(1) |
| CPSE | Rd,Rr | Compare, Skip if Equal | if (Rd = Rr) PC | ← | PC + 2 or 3 | None | 1 / 2 |
| 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 |

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| SBRC | Rr, b | Skip if Bit in Register Cleared | if (Rr(b) = 0) PC | ← | PC + 2 or 3 | None | 1 / 2 |
| SBRS | Rr, b | Skip if Bit in Register Set | if (Rr(b) = 1) PC | ← | PC + 2 or 3 | None | 1 / 2 |
| 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 |
| 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 |
| 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 |
| 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 24-3. Data Transfer Instructions**

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| MOV | Rd, Rr | Copy Register | Rd | ← | Rr | None | 1 |
| LDI | Rd, K | Load Immediate | Rd | ← | K | None | 1 |
| LDS | Rd, k | Load Direct from data space | Rd | ← | (k) | None | 2 |
| LD | Rd, X | Load Indirect | Rd | ← | (X) | None | 1 / 2 |
| LD | Rd, X+ | Load Indirect and Post-Increment | Rd<br>X | ←<br>← | (X)<br>X + 1 | None | 2 / 3 |
| LD | Rd, -X | Load Indirect and Pre-Decrement | X<br>Rd | ←<br>← | X - 1<br>(X) | None | 2 / 3 |
| LD | Rd, Y | Load Indirect | Rd | ← | (Y) | None | 1 / 2 |
| LD | Rd, Y+ | Load Indirect and Post-Increment | Rd<br>Y | ←<br>← | (Y)<br>Y + 1 | None | 2 / 3 |
| LD | Rd, -Y | Load Indirect and Pre-Decrement | Y<br>Rd | ←<br>← | Y - 1<br>(Y) | None | 2 / 3 |

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| LDD | Rd, Y+q | Load Indirect with Displacement | Rd | ← | (Y + q) | None | N/A |
| LD | Rd, Z | Load Indirect | Rd | ← | (Z) | None | 1 / 2 |
| LD | Rd, Z+ | Load Indirect and Post-Increment | Rd<br>Z | ←<br>← | (Z)<br>Z+1 | None | 2 / 3 |
| LD | Rd, -Z | Load Indirect and Pre-Decrement | Z<br>Rd | ←<br>← | Z - 1<br>(Z) | None | 2 / 3 |
| LDD | Rd, Z+q | Load Indirect with Displacement | Rd | ← | (Z + q) | None | N/A |
| STS | k, Rr | Store Direct to Data Space | (k) | ← | Rd | None | 1 |
| ST | X, Rr | Store Indirect | (X) | ← | Rr | None | 1 |
| ST | X+, Rr | Store Indirect and Post-Increment | (X)<br>X | ←<br>← | Rr<br>X + 1 | None | 1 |
| ST | -X, Rr | Store Indirect and Pre-Decrement | X<br>(X) | ←<br>← | X - 1<br>Rr | None | 2 |
| ST | Y, Rr | Store Indirect | (Y) | ← | Rr | None | 1 |
| ST | Y+, Rr | Store Indirect and Post-Increment | (Y)<br>Y | ←<br>← | Rr<br>Y + 1 | None | 1 |
| ST | -Y, Rr | Store Indirect and Pre-Decrement | Y<br>(Y) | ←<br>← | Y - 1<br>Rr | None | 2 |
| STD | Y+q, Rr | Store Indirect with Displacement | (Y + q) | ← | Rr | None | N/A |
| ST | Z, Rr | Store Indirect | (Z) | ← | Rr | None | 1 |
| ST | Z+, Rr | Store Indirect and Post-Increment | (Z)<br>Z | ←<br>← | Rr<br>Z + 1 | None | 1 |
| ST | -Z, Rr | Store Indirect and Pre-Decrement | Z | ← | Z - 1 | None | 2 |
| STD | Z+q,Rr | Store Indirect with Displacement | (Z + q) | ← | Rr | None | N/A |
| LPM | | Load Program Memory | R0 | ← | (Z) | None | N/A |
| 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 | 1(1) |
| POP | Rd | Pop Register from Stack | Rd | ← | STACK | None | 3(1) |

**Table 24-4. Bit and Bit-test Instructions**

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| LSL | Rd | Logical Shift Left | Rd(n+1)<br>Rd(0)<br>C | ←<br>←<br>← | Rd(n)<br>0<br>Rd(7) | Z,C,N,V,H | 1 |
| LSR | Rd | Logical Shift Right | Rd(n)<br>Rd(7)<br>C | ←<br>←<br>← | Rd(n+1)<br>0<br>Rd(0) | Z,C,N,V | 1 |
| ROL | Rd | Rotate Left Through Carry | Rd(0)<br>Rd(n+1)<br>C | ←<br>←<br>← | C<br>Rd(n)<br>Rd(7) | Z,C,N,V,H | 1 |

| Mnemonic | Operands | Description | | Op | | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|---|---|
| ROR | Rd | Rotate Right Through Carry | Rd(7)<br>Rd(n)<br>C | ←<br>←<br>← | C<br>Rd(n+1)<br>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 | 1 |
| CBI | A, b | Clear Bit in I/O Register | I/O(A, b) | ← | 0 | None | 1 |
| BST | Rr, b | Bit Store from Register to T | T | ← | Rr(b) | T | 1 |
| BLD | Rd, b | Bit load from T to Register | Rd(b) | ← | T | None | 1 |
| BSET | s | Flag Set | SREG(s) | ← | 1 | SREG(s) | 1 |
| 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 24-5. MCU Control Instructions**

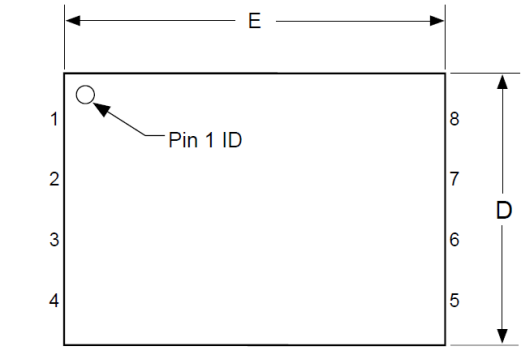
| Mnemonic | Operands | Description | Operation | Flags | #Clocks AVRrc |
|---|---|---|---|---|---|
| BREAK | | Break | (See also in Debug interface description) | None | 1 |
| NOP | | No Operation | | None | 1 |
| SLEEP | | Sleep | (see also power management and sleep description) | None | 1 |
| WDR | | Watchdog Reset | (see also Watchdog Controller description) | None | 1 |

**Note:**

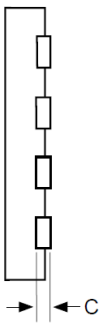
1. Cycle time for data memory accesses assume internal RAM access, and are not valid for accesses through the NVM controller. A minimum of one extra cycle must be added when accessing memory through the NVM controller (such as Flash and EEPROM), but depending on simultaneous accesses by other masters or the NVM controller state, there may be more than one extra cycle.

# 26. Packaging Information

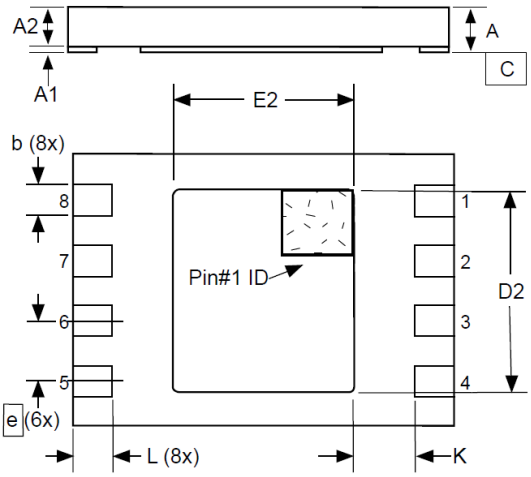
## 26.1. 8-pin UDFN




TOP VIEW




SIDE VIEW




BOTTOM VIEW

Notes:
1. This drawing is for general information only. Refer to Drawing MO-229, for proper dimensions, tolerances, datums, etc.
2. The Pin #1 ID is a laser-marked feature on Top View.
3. Dimensions b applies to metallized terminal and is measured between 0.15 mm and 0.30 mm from the terminal tip. If the terminal has the optional radius on the other end of the terminal, the dimension should not be measured in that radius area.
4. The Pin #1 ID on the Bottom View is an orientation feature on the thermal pad.

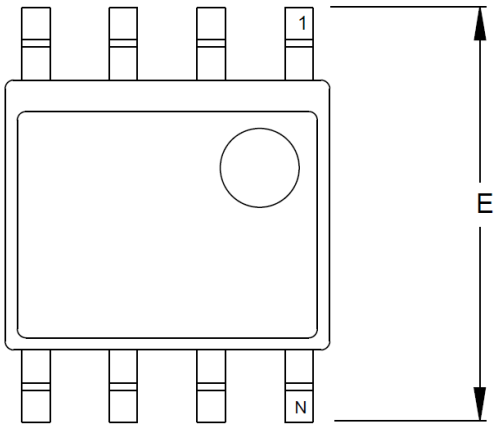
COMMON DIMENSIONS
(Unit of Measure = mm)

| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | 0.50 | 0.55 | 0.60 | |
| A1 | 0.0 | 0.02 | 0.05 | |
| A2 | - | - | 0.55 | |
| D | 1.90 | 2.00 | 2.10 | |
| D2 | 1.40 | 1.50 | 1.60 | |
| E | 2.90 | 3.00 | 3.10 | |
| E2 | 1.20 | 1.30 | 1.40 | |
| b | 0.18 | 0.25 | 0.30 | 3 |
| C | 0.152 REF | | | |
| L | 0.35 | 0.40 | 0.45 | |
| e | 0.50 BSC | | | |
| K | 0.20 | - | - | |

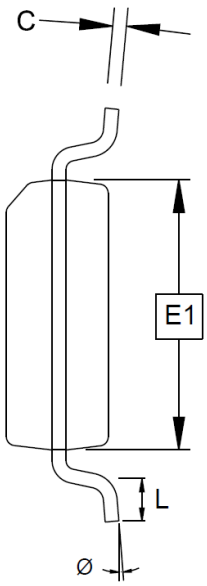
11/2/15

| Atmel<br>Package Drawing Contact:<br>packagedrawings@atmel.com | TITLE<br>8MA2, 8-pad 2 x 3 x 0.6mm Body, Thermally Enhanced Plastic Ultra Thin Dual Flat No-Lead Package (UDFN) | GPC<br>YNZ | DRAWING NO.<br>8MA2 | REV.<br>H |
|---|---|---|---|---|

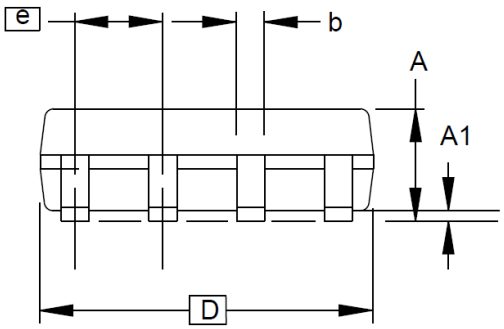
## 26.2. 8-pin SOIC150




TOP VIEW




END VIEW




SIDE VIEW

Notes: This drawing is for general information only.
Refer to JEDEC Drawing MS-012, Variation AA
for proper dimensions, tolerances, datums, etc.

COMMON DIMENSIONS
(Unit of Measure = mm)

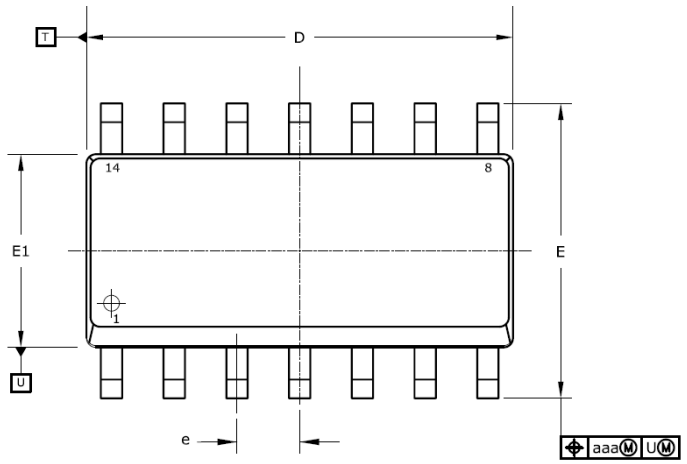
| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | 1.35 | – | 1.75 | |
| A1 | 0.10 | – | 0.25 | |
| b | 0.31 | – | 0.51 | |
| C | 0.17 | – | 0.25 | |
| D | 4.80 | – | 5.05 | |
| E1 | 3.81 | – | 3.99 | |
| E | 5.79 | – | 6.20 | |
| e | 1.27 BSC | | | |
| L | 0.40 | – | 1.27 | |
| Ø | 0° | – | 8° | |

6/22/11

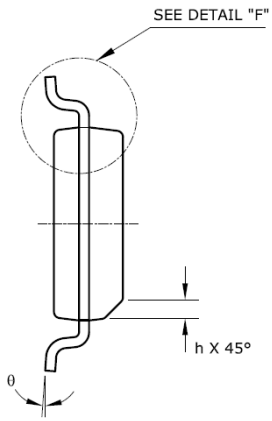
| Atmel<br>Package Drawing Contact:<br>packagedrawings@atmel.com | **TITLE**<br>8S1, 8-lead (0.150” Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) | **GPC**<br>SWB | **DRAWING NO.**<br>8S1 | **REV.**<br>G |
|---|---|---|---|---|

## 26.3. 14-pin SOIC150

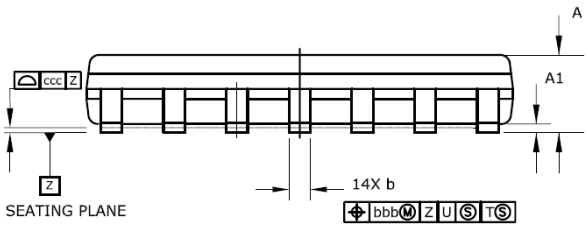
DRAWINGS NOT SCALED




TOP VIEW

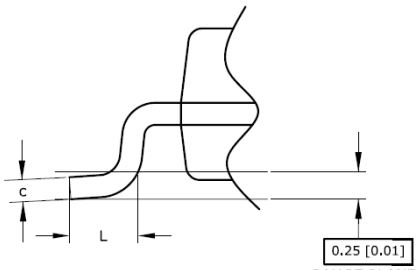



END VIEW




SEATING PLANE

SIDE VIEW




GAUGE PLANE

DETAIL 'F'

COMMON DIMENSIONS

(Unit of Measure = mm)

| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | 1.35 | 1.60 | 1.75 | |
| A1 | 0.10 | ---- | 0.25 | |
| D | 8.55 | 8.65 | 8.75 | 2 |
| E | 5.80 | 6.00 | 6.20 | |
| E1 | 3.80 | 3.90 | 4.00 | 3 |
| L | 0.40 | ---- | 1.27 | |
| b | 0.33 | ---- | 0.51 | 4 |
| c | 0.19 | ----- | 0.25 | |
| e | 1.27 BSC | | | |
| h | 0.25 | ----- | 0.50 | |
| θ | 0° | ----- | 8° | |
| TOLERANCES OF FORM AND POSITION | | | | |
| aaa | 0.25 | | | |
| bbb | 0.25 | | | |
| ccc | 0.10 | | | |
| n | 14 | | | |

Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MS-012, Variation AB.
2. Dimensions D does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shal not exceed 0.15mm (0.006 in) per side.
3. Dimensions E1 does not include inter-lead Flash or protrusions. Inter-lead Flash and protrusions shall not exceed 0.25mm (0.010 in) per side.
4. Dimensions b does not include Dambar protrusion. Allowable Dambar protrusion shall be 0.10mm total in excess of the b dimension at maximum material condition. Dambar cannot be located on the lower radius of the foot.

01/31/2014

| Atmel | TITLE | GPC | DRAWING NO. | REV. |
|---|---|---|---|---|
| **Package Drawing Contact:** **packagedrawings@atmel.com** | TU, 14 Lds - 1.27mm Pitch, 8.65X3.90x1.60mm Body Size Plastic Small Outline Package (SOIC) | SVQ | TU | D |

# 27. Errata

## 27.1. ATtiny102

### 27.1.1. Rev.A

No known Errata.

## 27.2. ATtiny104

### 27.2.1. Rev.A

No known Errata.

# 28. Datasheet Revision History

## 28.1. Rev D - 10/2016

- AVR CPU Core:
  - New section added, Accessing 16-bit Registers.
  - Stack Pointer (SPL and SPH) register updated.
- USART - Universal Synchronous Asynchronous Receiver Transceiver:
  - Added UCSR0D.SFDE bit and corrected RXIE bit name to RXSIE.
  - Updated presentation of the UBRR0L and UBRR0H registers.
- TC0 - 16-bit Timer/Counter0 with PWM:
  - Updated presentation of theTCNT0L and TCNT0H, OCR0AL and OCR0AH, and ICR0L and ICR0H registers.
- ADC - Analog to Digital Converter:
  - Updated the ADCL and ADCH registers (for both ADLAR=0 and 1).

## 28.2. Rev C - 07/2016

- Thermal pad of 8-pin UDFN package should not be connected.

## 28.3. Rev B - 06/2016

- Update Electrical Characteristics
- Update Typical Characteristics

## 28.4. Rev A - 02/2016

Initial revision.

Atmel | Enabling Unlimited Possibilities®

**Atmel Corporation** 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

© 2016 Atmel Corporation. / Rev.: Atmel-42505D-ATtiny102-ATtiny104_Datasheet_Complete-10/2016

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, AVR® and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.