ATtiny87/167

tinyAVR® Data Sheet

Introduction

The ATtiny87/167 is a low power, CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. The ATtiny87/167 is a 20/32-pins device with 8/16 KB Flash, 512 bytes SRAM and 512 bytes EEPROM. By executing instructions in a single clock cycle, the devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

Features

- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 123 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- Non-volatile Program and Data Memories
  - 8K/16K Bytes of In-System Programmable Flash Program Memory
    - Endurance: 10,000 Write/Erase Cycles
  - 512 Bytes of In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 512 Bytes of Internal SRAM
  - Data retention: 20 Years at 85°C / 100 Years at 25°C
  - In-System Programmable via SPI Port
  - Low size LIN/UART Software In-System Programmable
  - Programming Lock for Software Security
- Peripheral Features
  - LIN 2.1 and 1.3 Controller or 8-bit UART
  - One 8-bit Asynchronous Timer/Counter with Prescaler
    - Output Compare or 8-bit PWM Channel
  - One 16-bit Synchronous Timer/Counter with Prescaler
    - External Event Counter
    - 2 Output Compare Units or PWM Channels each Driving up to 4 Output Pins
  - Master/Slave SPI Serial Interface
  - Universal Serial Interface with Start Condition Detector
  - 10-bit ADC

    - 11 Single Ended Channels
    - 8 Differential ADC Channel Pairs with Programmable Gain (8x or 20x)
  - On-chip Analog Comparator with Selectable Voltage Reference
  - 100 µA ±10% Current Source for LIN Node Identification
  - On-chip Temperature Sensor
  - Programmable Watchdog Timer with Separate On-chip Oscillator
- Special Microcontroller Features
  - Software Controlled Clock Switching for Power Control, EMC Reduction
  - debugWIRE On-chip Debug System
  - External and Internal Interrupt Sources
  - Low Power Idle, ADC Noise Reduction, and Power-down Modes
  - Power-on Reset and Programmable Brown-out Detection
  - Internal 8MHz Calibrated Oscillator
  - 4-16 MHz and 32 KHz Crystal/Ceramic Resonator Oscillators
- I/O and Packages
  - 16 Programmable I/O Lines
  - 20-pin SOIC, 32-pad VQFN and 20-pin TSSOP
- Operating Voltage:
  - 1.8 – 5.5V for ATtiny87/167
- Speed Grade:
  - 0 – 4 MHz @ 1.8 – 5.5V
  - 0 – 8 MHz @ 2.7 – 5.5V
  - 0 – 16 MHz @ 4.5 – 5.5V
- Industrial Temperature Range

Table of Contents

1 Introduction 1

2 Features 1

3 Description 9

3.1 Comparison Between ATtiny87 and ATtiny167 9

3.2 Part Description 9

3.3 Block Diagram 10

3.4 Pin Configuration 11

3.5 Pin Description 13

3.6 Resources 14

3.7 About Code Examples 14

3.8 Data Retention 14

3.9 Disclaimer 14

4 AVR CPU Core 15

4.1 Overview 15

4.2 ALU – Arithmetic Logic Unit 16

4.3 Status Register 16

4.4 General Purpose Register File 18

4.5 Stack Pointer 19

4.6 Instruction Execution Timing 20

4.7 Reset and Interrupt Handling 20

5 AVR Memories 23

5.1 In-System Re-programmable Flash Program Memory 23

5.2 SRAM Data Memory 24

5.3 EEPROM Data Memory 25

5.4 I/O Memory 29

5.5 Register Description 29

6 System Clock and Clock Options 32

6.1 Clock Systems and their Distribution 32

6.2 Clock Sources 33

6.3 Dynamic Clock Switch 39

6.4 System Clock Prescaler 45

6.5 Register Description 45

7 Power Management and Sleep Modes 50
7.1 Sleep Modes 50
7.2 BOD Disable 51
7.3 Idle Mode 51
7.4 ADC Noise Reduction Mode 51
7.5 Power-down Mode 52
7.6 Power-save Mode 52
7.7 Power Reduction Register 52
7.8 Minimizing Power Consumption 53
7.9 Register Description 54
8 System Control and Reset 57
8.1 Reset 57
8.2 Internal Voltage Reference 61
8.3 Watchdog Timer 61
9 Interrupts 67
9.1 Interrupt Vectors in ATtiny87/167 67
9.2 Program Setup in ATtiny87 68
9.3 Program Setup in ATtiny167 69
10 External Interrupts 70
10.1 Overview 70
10.2 Pin Change Interrupt Timing 70
10.3 Register Description 71
11 I/O-Ports 75
11.1 Introduction 75
11.2 Ports as General Digital I/O 76
11.3 Alternate Port Functions 80
11.4 Register Description 93
12 8-bit Timer/Counter0 and Asynchronous Operation 94
12.1 Features 94
12.2 Overview 94
12.3 Timer/Counter Clock Sources 96
12.4 Counter Unit 96
12.5 Output Compare Unit 97
12.6 Compare Match Output Unit 98

12.7 Modes of Operation 99
12.8 Timer/Counter Timing Diagrams 104
12.9 Asynchronous Operation of Timer/Counter0 105
12.10 Timer/Counter0 Prescaler 107
12.11 Register Description 107

13 Timer/Counter1 Prescaler 114

13.1 Overview 114
13.2 Register Description 116

14 16-bit Timer/Counter1 117

14.1 Features 117
14.2 Overview 117
14.3 Accessing 16-bit Registers 119
14.4 Timer/Counter Clock Sources 122
14.5 Counter Unit 123
14.6 Input Capture Unit 124
14.7 Output Compare Units 126
14.8 Compare Match Output Unit 127
14.9 Modes of Operation 129
14.10 Timer/Counter Timing Diagrams 137
14.11 Register Description 139

15 SPI - Serial Peripheral Interface 147

15.1 Features 147
15.2 $\overline{SS}$ Pin Functionality 152
15.3 Data Modes 155

16 USI – Universal Serial Interface 156

16.1 Features 156
16.2 Overview 156
16.3 Functional Descriptions 157
16.4 Alternative USI Usage 163
16.5 Register Description 163

17 LIN / UART - Local Interconnect Network Controller or UART 169

17.1 LIN Features 169
17.2 UART Features 169
17.3 LIN Protocol 170

17.4 LIN / UART Controller 171
17.5 LIN / UART Description 177
17.6 Register Description 187

18 ISRC – Current Source 195

18.1 Features 195
18.2 Typical applications 195
18.3 Control Register 197

19 ADC – Analog to Digital Converter 198

19.1 Features 198
19.2 Overview 198
19.3 Operation 200
19.4 Starting a Conversion 201
19.5 Prescaling and Conversion Timing 202
19.6 Changing Channel or Reference Selection 204
19.7 ADC Noise Canceler 205
19.8 ADC Conversion Result 209
19.9 Temperature Measurement 210
19.10 Internal Voltage Reference Output 210
19.11 Register Description 212

20 AnaComp - Analog Comparator 218

20.1 Analog Comparator Inputs 218
20.2 Register Description 220

21 DebugWIRE On-chip Debug System 222

21.1 Features 222
21.2 Overview 222
21.3 Physical Interface 222
21.4 Software Break Points 223
21.5 Limitations of DebugWIRE 223
21.6 DebugWIRE Related Register in I/O Memory 223

22 Flash Programming 224

22.1 Self-Programming the Flash 224
22.2 Addressing the Flash During Self-Programming 225

23 Memory Programming 232

23.1 Program and Data Memory Lock Bits 232

23.2 Fuse Bits ... 232
23.3 Signature Bytes ... 234
23.4 Calibration Byte ... 235
23.5 Page Size ... 235
23.6 Parallel Programming Parameters, Pin Mapping, and Commands ... 235
23.7 Parallel Programming ... 238
23.8 Serial Downloading ... 245
23.9 Serial Programming Characteristics ... 249

24 Electrical Characteristics ... 250

24.1 Absolute Maximum Ratings* ... 250
24.2 DC Characteristics ... 250
24.3 Speed ... 252
24.4 Clock Characteristics ... 252
24.5 RESET Characteristics ... 253
24.6 Internal Voltage Characteristics ... 254
24.7 Current Source Characteristics ... 255
24.8 ADC Characteristics ... 255
24.9 Parallel Programming Characteristics ... 257
24.10 SPI Timing Characteristics ... 260

25 Typical Characteristics ... 262

25.1 Current Consumption in Active Mode ... 262
25.2 Current Consumption in Idle Mode ... 264
25.3 Supply Current of I/O modules ... 266
25.4 Current Consumption in Power-down Mode ... 266
25.5 Current Consumption in Reset ... 267
25.6 Pull-up Resistors ... 268
25.7 Output Driver Strength ... 270
25.8 Input Thresholds and Hysteresis (for I/O Ports) ... 272
25.9 BOD, Bandgap and Reset ... 273
25.10 Internal Oscillator Speed ... 276

26 Register Summary ... 278

27 Instruction Set Summary ... 282

28 Ordering Information ... 284

28.1 ATtiny87 ... 284

28.2 ATtiny167 285

29 Packaging Information 286

29.1 32PN 286

29.2 20S2 289

29.3 20X 292

29.4 TSB 295

30 Errata 298

31 Datasheet Revision History 299

31.1 Rev. A – 01/2020 299

31.2 Rev. 8265D – 01/2014 299

31.3 Rev. 8265C – 03/12 299

31.4 Rev. 8265B – 09/10 299

31.5 Rev. 8265A – 08/10 299

1. Description

1.1 Comparison Between ATtiny87 and ATtiny167

ATtiny87 and ATtiny167 are hardware and software compatible. They differ only in memory sizes as shown in Table 1-1.

Table 1-1. Memory Size Summary

Device	Flash	EEPROM	SRAM	Interrupt Vector size
ATtiny167	16K Bytes	512 Bytes	512 Bytes	2-instruction-words / vector
ATtiny87	8K Bytes	512 Bytes	512 Bytes	2-instruction-words / vector

1.2 Part Description

The ATtiny87/167 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 ATtiny87/167 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

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

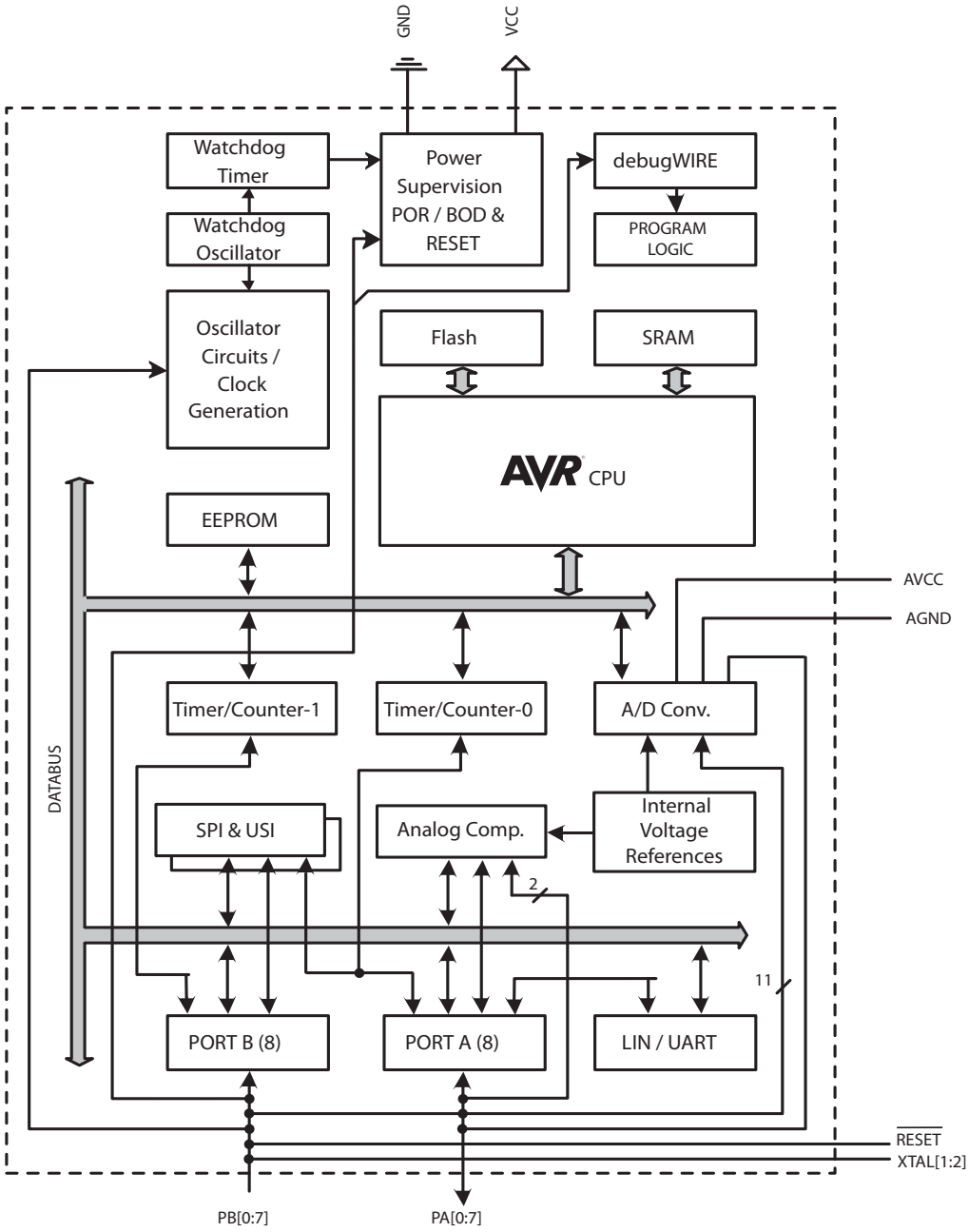
The ATtiny87/167 provides the following features: 8K/16K byte of In-System Programmable Flash, 512 bytes EEPROM, 512 bytes SRAM, 16 general purpose I/O lines, 32 general purpose working registers, one 8-bit Timer/Counter with compare modes, one 8-bit high speed Timer/Counter, Universal Serial Interface, a LIN controller, Internal and External Interrupts, a 11-channel, 10-bit ADC, a programmable Watchdog Timer with internal Oscillator, and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, Analog Comparator, and Interrupt system to continue functioning. The Power-down mode saves the register contents, disabling all chip functions until the next Interrupt or Hardware Reset. The ADC Noise Reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions.

The device is manufactured using Microchip's high density non-volatile memory technology. The On-chip ISP Flash allows the Program memory to be re-programmed In-System through an SPI serial interface, by a conventional non-volatile memory programmer or by an On-chip boot code running on the AVR core. The Boot program can use any interface to download the application program in the Flash memory. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Microchip ATtiny87/167 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATtiny87/167 AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

1.3 Block Diagram

Figure 1-1. Block Diagram



1.4 Pin Configuration

Figure 1-2. Pinout ATtiny87/167 - SOIC20 & TSSOP20

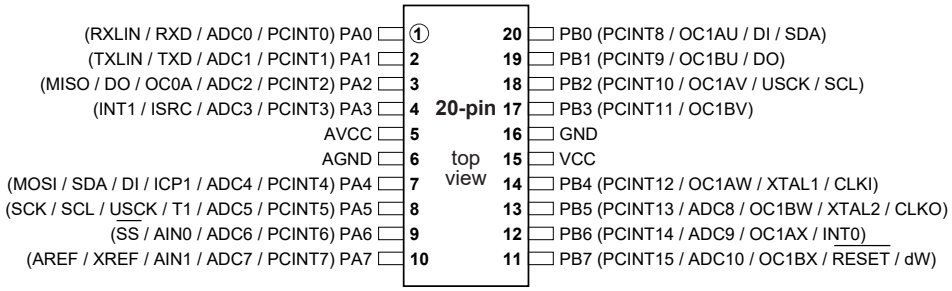


Figure 1-3. Pinout ATtiny87/167 - VQFN32

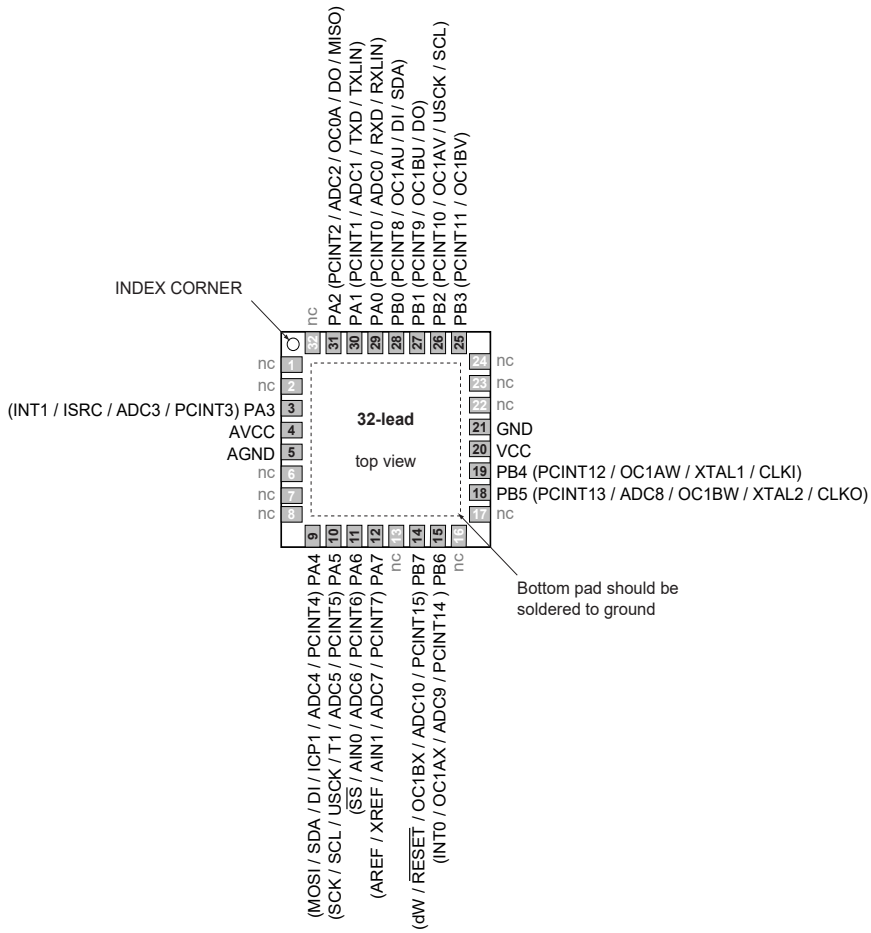
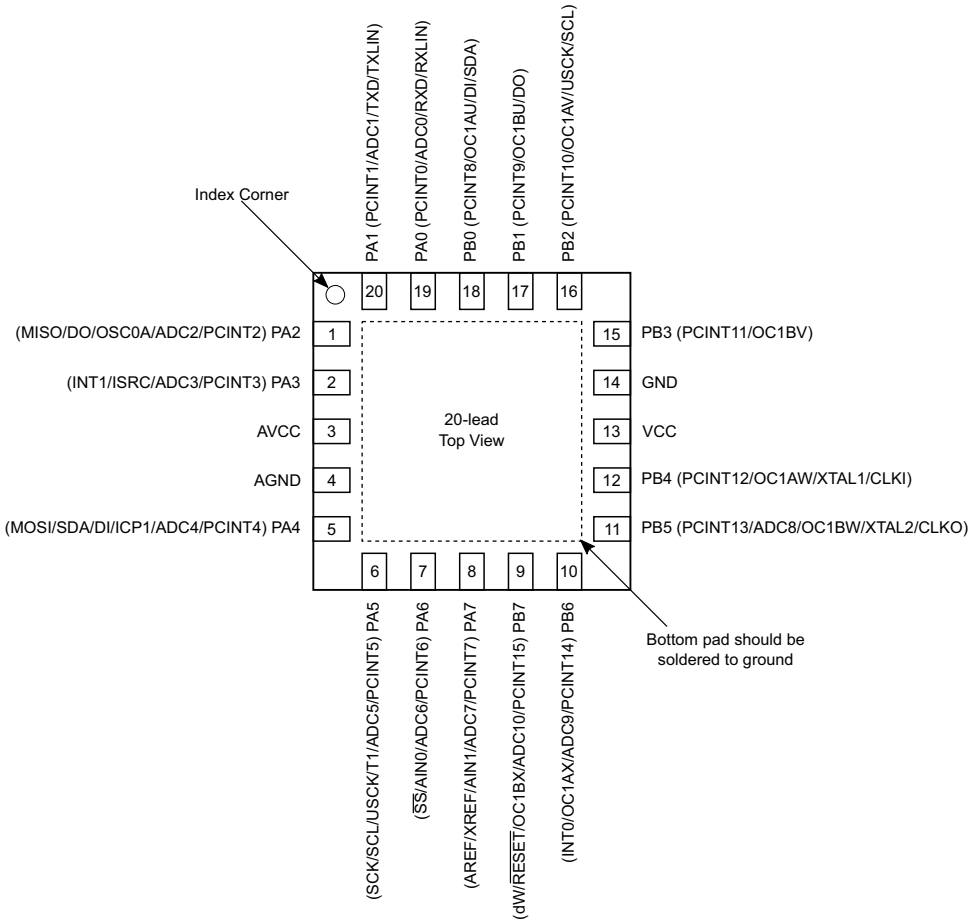


Figure 1-4. Pinout ATtiny167 - WQFN20



1.5 Pin Description

1.5.1 VCC

Supply voltage.

1.5.2 GND

Ground.

1.5.3 AVCC

Analog supply voltage.

1.5.4 AGND

Analog ground.

1.5.5 Port A (PA7:PA0)

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

Port A also serves the functions of various special features of the ATtiny87/167 as listed on Section 9.3.3 “Alternate Functions of Port A” on page 84.

1.5.6 Port B (PB7:PB0)

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

Port B also serves the functions of various special features of the ATtiny87/167 as listed on Section 9.3.4 “Alternate Functions of Port B” on page 89.

1.6 Resources

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

1.7 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. Please confirm with the C compiler documentation for more details.

1.8 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.

1.9 Disclaimer

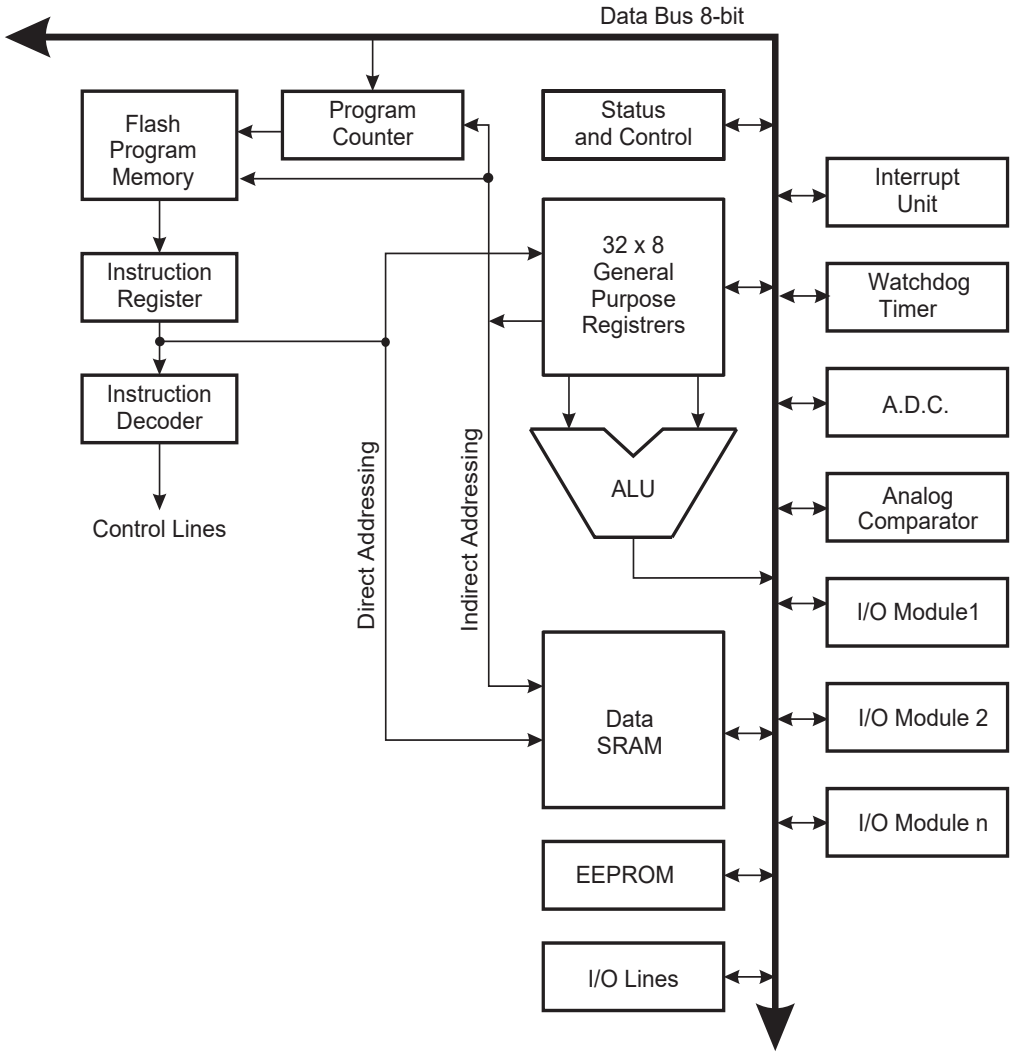
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device has been characterized.

2. AVR CPU Core

2.1 Overview

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

Figure 2-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the Program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the Program memory. This concept enables instructions to be executed in every clock cycle. The Program memory is In-System Reprogrammable Flash memory. The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

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

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

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

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

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

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

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F.

2.2 ALU – Arithmetic Logic Unit

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

2.3 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that 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.

2.3.1 SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set 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: Bit 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 is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, 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 arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

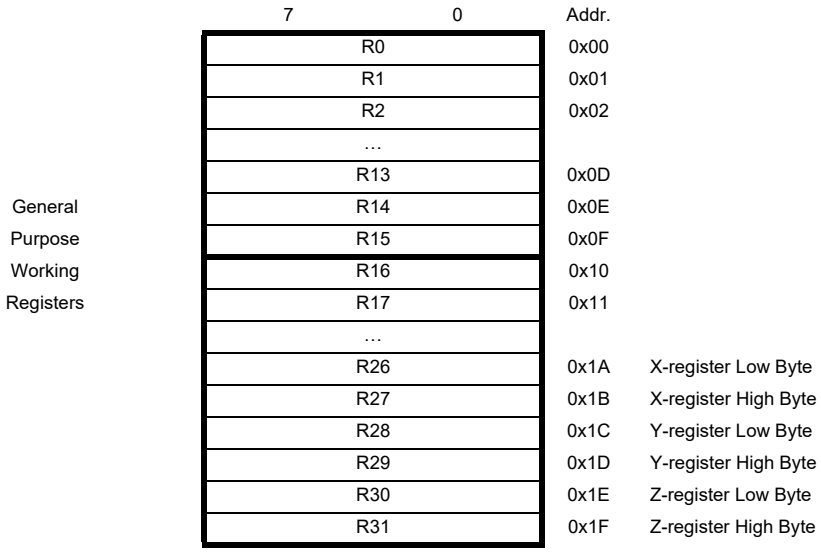
2.4 General Purpose Register File

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

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

Figure 2-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 2-2. AVR CPU General Purpose Working Registers



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

As shown in Figure 2-2, each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

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

The registers R26:R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 2-3 on page 19.

Figure 2-3. 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).

2.5 Stack Pointer

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

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

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

2.5.1 SPH and SPL – Stack Pointer Register

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	ISRAM end (See Table 3-1 on page 23)								

2.6 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.

Figure 2-4 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 2-4. The Parallel Instruction Fetches and Instruction Executions

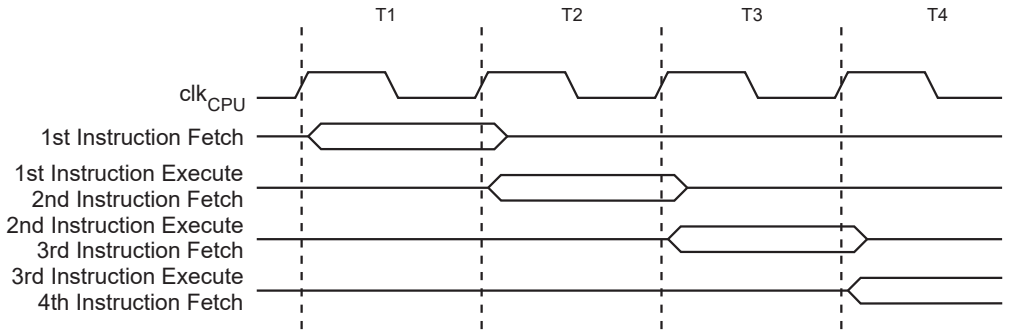
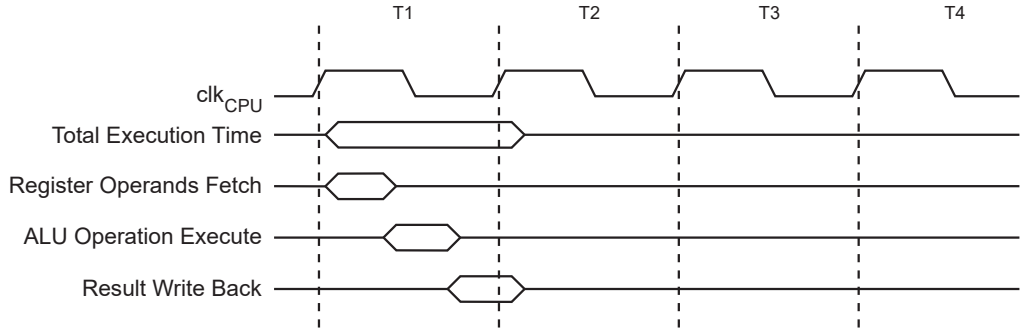


Figure 2-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 2-5. Single Cycle ALU Operation



2.7 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 Section 7. “Interrupts” on page 67. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0.

2.7.1 Interrupt behavior

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.

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

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

Assembly Code Example

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

C Code Example

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

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)
```

C Code Example

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

2.7.2 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.

3. AVR Memories

This section describes the different memories in the ATtiny87/167. The AVR architecture has two main memory spaces, the Data memory and the Program memory space. In addition, the ATtiny87/167 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

Table 3-1. Memory Mapping.

Memory		Mnemonic	ATtiny87	ATtiny167
Flash	Size	Flash size	8 K bytes	16 K bytes
	Start Address	-	0x0000	
	End Address	Flash end	0x1FFF(1) 0x0FFF(2)	0x3FFF(1) 0x1FFF(2)
32 Registers	Size	-	32 bytes	
	Start Address	-	0x0000	
	End Address	-	0x001F	
I/O Registers	Size	-	64 bytes	
	Start Address	-	0x0020	
	End Address	-	0x005F	
Ext I/O Registers	Size	-	160 bytes	
	Start Address	-	0x0060	
	End Address	-	0x00FF	
Internal SRAM	Size	ISRAM size	512 bytes	
	Start Address	ISRAM start	0x0100	
	End Address	ISRAM end	0x02FF	
EEPROM	Size	E2 size	512 bytes	
	Start Address	-	0x0000	
	End Address	E2 end	0x01FF	

Notes: 1. Byte address.
2. Word (16-bit) address.

3.1 In-System Re-programmable Flash Program Memory

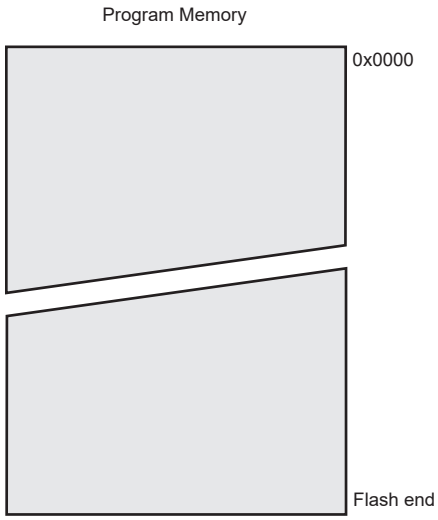
The ATtiny87/167 contains On-chip In-System Reprogrammable Flash memory for program storage (see “Flash size” in Table 3-1 on page 23). Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 16 bits wide. ATtiny87/167 does not have separate Boot Loader and Application Program sections, and the SPM instruction can be executed from the entire Flash. See SELFPRGEN description in Section 20.2.1 “SPMCSR – Store Program Memory Control and Status Register” on page 226 for more details.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATtiny87/167 Program Counter (PC) address the program memory locations. Section 21. “Memory Programming” on page 232 contains a detailed description on Flash data serial downloading using the SPI pins.

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

Timing diagrams for instruction fetch and execution are presented in Section 2.6 “Instruction Execution Timing” on page 20.

Figure 3-1. Program Memory Map



3.2 SRAM Data Memory

Figure 3-2 shows how the ATtiny87/167 SRAM Memory is organized.

The ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next locations address the internal data SRAM (see “ISRAM size” in Table 3-1 on page 23).

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

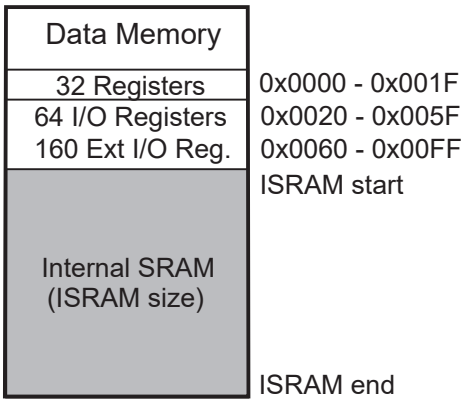
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers and the internal data SRAM in the ATtiny87/167 are all accessible through all these addressing modes. The Register File is described in “General Purpose Register File” on page 18.

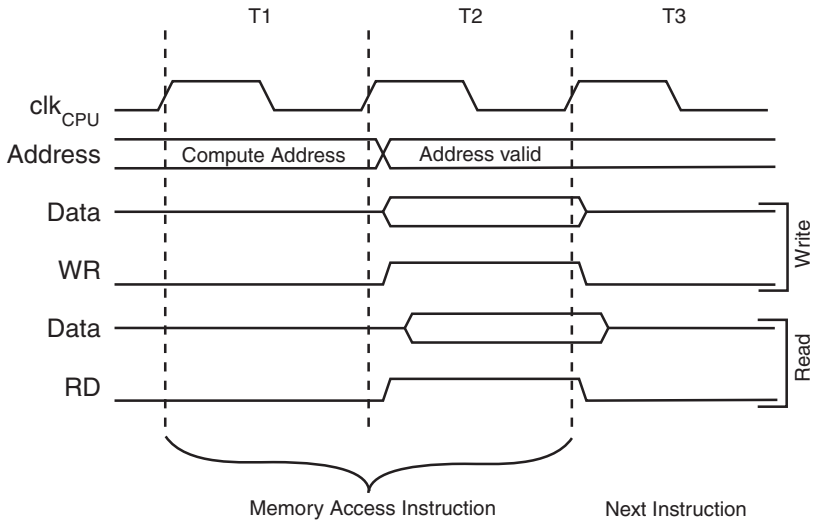
Figure 3-2. Data Memory Map



3.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 3-3.

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



3.3 EEPROM Data Memory

The ATtiny87/167 contains EEPROM memory (see “E2 size” in Table 3-1 on page 23). It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register and the EEPROM Control Register.

Section 21. “Memory Programming” on page 232 contains a detailed description on EEPROM programming in SPI or Parallel Programming mode.

3.3.1 EEPROM Read/Write Access

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

The write access times for the EEPROM are given in Table 3-2. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See “Preventing EEPROM Corruption” on page 28 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to “Atomic Byte Programming” on page 26 and “Split Byte Programming” on page 26 for details on this.

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

3.3.2 Atomic Byte Programming

Using Atomic Byte Programming is the simplest mode. When writing a byte to the EEPROM, the user must write the address into the EEARL Register and data into EEDR Register. If the EEPMn bits are zero, writing EEPE (within four cycles after EEMPE is written) will trigger the erase/write operation. Both the erase and write cycle are done in one operation and the total programming time is given in Table 1. The EEPE bit remains set until the erase and write operations are completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

3.3.3 Split Byte Programming

It is possible to split the erase and write cycle in two different operations. This may be useful if the system requires short access time for some limited period of time (typically if the power supply voltage falls). In order to take advantage of this method, it is required that the locations to be written have been erased before the write operation. But since the erase and write operations are split, it is possible to do the erase operations when the system allows doing time-critical operations (typically after Power-up).

3.3.4 Erase

To erase a byte, the address must be written to EEAR. If the EEPMn bits are 0b01, writing the EEPE (within four cycles after EEMPE is written) will trigger the erase operation only (programming time is given in Table 1). The EEPE bit remains set until the erase operation completes. While the device is busy programming, it is not possible to do any other EEPROM operations.

3.3.5 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEPMn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in Table 1). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated Oscillator is used to time the EEPROM accesses. Make sure the Oscillator frequency is within the requirements described in “OSCCAL – Oscillator Calibration Register” on page 45.

The following code examples show one assembly and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic   EECR,EEPE
  rjmp   EEPROM_write
  ; Set Programming mode
  ldi    r16, (0<<EEPM1)|(0<<EEPM0)
  out    EECR, r16
  ; Set up address (r18:r17) in address register
  out    EEARH, r18
  out    EEARL, r17
  ; Write data (r16) to data register
  out    EEDR, r16
  ; Write logical one to EEMPE
  sbi    EECR,EEMPE
  ; Start eeprom write by setting EEPE
  sbi    EECR,EEPE
  ret
```

C Code Example

```
void EEPROM_write(unsigned char ucAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set Programming mode */
  EECR = (0<<EEPM1)|(0<<EEPM0);
  /* Set up address and data registers */
  EEAR = ucAddress;
  EEDR = ucData;
  /* Write logical one to EEMPE */
  EECR |= (1<<EEMPE);
  /* Start eeprom write by setting EEPE */
  EECR |= (1<<EEPE);
}
```

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

Assembly Code Example

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

C Code Example

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

3.3.6 Preventing EEPROM Corruption

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

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

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

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

3.4 I/O Memory

The I/O space definition of the ATtiny87/167 is shown in Section 24. “Register Summary” on page 278.

All ATtiny87/167 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. 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 to 0x1F only.

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

3.4.1 General Purpose I/O Registers

The ATtiny87/167 contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags.

The General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

3.5 Register Description

3.5.1 EEARH and EEARL – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x22 (0x42)	–	–	–	–	–	–	–	EEAR8	EEARH
0x21 (0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
Initial Value	X	X	X	X	X	X	X	X	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved for future use and will always read as 0 in ATtiny87/167.

- **Bits 8:0 – EEAR[8:0]: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL – specifies the high EEPROM address in the EEPROM space (see “E2 size” in Table 3-1 on page 23). The EEPROM data bytes are

addressed linearly between 0 and “E2 size”. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

3.5.2 EEDR – EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – EEDR[7:0]: EEPROM Data**

For the EEPROM write operation the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

3.5.3 EECR – EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved for future use and will always read as 0 in ATtiny87/167. After reading, mask out these bits. For compatibility with future AVR devices, always write these bits to zero.

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

The EEPROM Programming mode bits setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in Table 3-2. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 3-2. EEPROM Mode Bits

EEPM1	EEPM0	Typical Programming Time	Operation
0	0	3.4 ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8 ms	Erase Only
1	0	1.8 ms	Write Only
1	1	–	Reserved for future use

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

Writing EERIE to one enables the EEPROM Ready Interrupt if the I-bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready Interrupt generates a constant interrupt when Non-volatile memory is ready for programming.

- **Bit 2 – EEMPE: EEPROM Master Program Enable**

The EEMPE bit determines whether writing EEPE to one will have effect or not.

When EEMPE is set, setting EEPE within four clock cycles will program the EEPROM at the selected address. If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles.

- **Bit 1 – EEPE: EEPROM Program Enable**

The EEPROM Program Enable Signal EEPE is the programming enable signal to the EEPROM. When EEPE is written, the EEPROM will be programmed according to the EEPMn bits setting. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. When the write access time has elapsed, the EEPE bit is cleared by hardware. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

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

The EEPROM Read Enable Signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

3.5.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	GPIOR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3.5.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x2A (0x4A)	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	GPIOR1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3.5.6 GPIOR0 – General Purpose I/O Register 0

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

4. System Clock and Clock Options

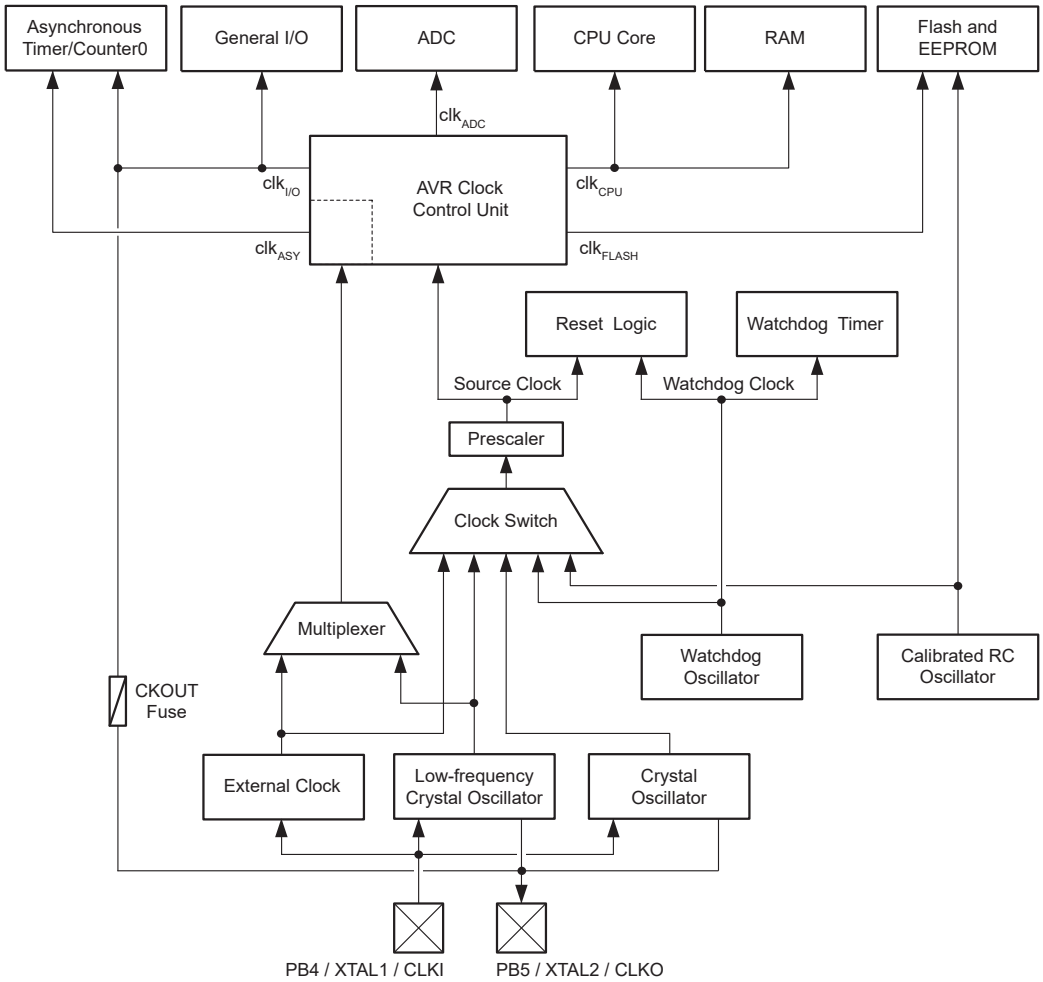
The ATtiny87/167 provides a large number of clock sources. They can be divided into two categories: internal and external. Some external clock sources can be shared with the asynchronous timer. After reset, the clock source is determined by the CKSEL Fuses. Once the device is running, software clock switching is possible to any other clock sources.

Hardware controls are provided for clock switching management but some specific procedures must be observed. Clock switching should be performed with caution as some settings could result in the device having an incorrect configuration.

4.1 Clock Systems and their Distribution

Figure 4-1 presents the principal clock systems in the AVR and their distribution. All of the clocks may not need to be active at any given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes or by using features of the dynamic clock switch circuit (See “Power Management and Sleep Modes” on page 50 and “Dynamic Clock Switch” on page 39). The clock systems are detailed below.

Figure 4-1. Clock Distribution



4.1.1 CPU Clock – clk_{CPU}

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

4.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like synchronous 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.

4.1.3 Flash Clock – clk_{FLASH}

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

4.1.4 Asynchronous Timer Clock – clk_{ASY}

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

4.1.5 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.

4.2 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits (default) or by the CLKSELR register (dynamic clock switch circuit) as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 4-1. Device Clocking Options Select(1) vs. PB4 and PB5 Functionality

Device Clocking Option	CKSEL[3:0] (2) CSEL[3:0] (3)	PB4	PB5
External Clock	0000_b	CLKI	CLKO - I/O
Calibrated Internal RC Oscillator 8.0 MHz	0010_b	I/O	CLKO - I/O
Watchdog Oscillator 128 kHz	0011_b	I/O	CLKO - I/O
External Low-frequency Oscillator	$01xx_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (0.4 - 0.9 MHz)	$100x_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (0.9 - 3.0 MHz)	$101x_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (3.0 - 8.0 MHz)	$110x_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (8.0 - 16.0 MHz)	$111x_b$	XTAL1	XTAL2

Notes:
1. For all fuses “1” means unprogrammed while “0” means programmed.
2. Flash Fuse bits.
3. CLKSELR register bits.

The various choices for each clocking option are given in the following sections.

When the CPU wakes up from Power-down or Power-save, or when a new clock source is enabled by the dynamic clock switch circuit, the selected clock source is used to time the start-up, ensuring stable oscillator operation before instruction execution starts.
When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up sequence. The number of WDT Oscillator cycles used for each time-out is shown in Table 4-2.

Table 4-2. Number of Watchdog Oscillator Cycles

Typ. Time-out (V_{CC} = 5.0V)	Typ. Time-out (V_{CC} = 5.0V)	Number of Cycles
4.1 ms	4.3 ms	512
65 ms	69 ms	8K (8,192)

4.2.1 Default Clock Source

At reset, the CKSEL and SUT fuse settings are copied into the CLKSELR register. The device will then use the clock source and the start-up timings defined by the CLKSELR bits (CSEL[3:0] and CSUT[1:0]).

The device is shipped with CKSEL Fuses = 0010_b, SUT Fuses = 10_b, and CKDIV8 Fuse programmed. The default clock source setting is therefore the Internal RC Oscillator running at 8 MHz with the longest start-up time and an initial system clock divided by 8. This default setting ensures that all users can make their desired clock source setting using an In-System or High-voltage Programmer. This set-up must be taken into account when using ISP tools.

4.2.2 Calibrated Internal RC Oscillator

By default, the Internal RC Oscillator provides an approximate 8.0 MHz clock. Though voltage and temperature dependent, this clock can be accurately calibrated by the user. See Table 22-1 on page 252 and Section 23.10 “Internal Oscillator Speed” on page 276 for more details.

If selected, it can operate without external components. At reset, hardware loads the pre-programmed calibration value into the OSCCAL Register and thereby automatically configuring the RC Oscillator. The accuracy of this calibration is shown as Factory calibration in Table 22-1 on page 252.

By adjusting the OSCCAL register in software, see “OSCCAL – Oscillator Calibration Register” on page 45, it is possible to get a higher calibration accuracy than by using the factory calibration. The accuracy of this calibration is shown as User calibration in Table 22-1 on page 252.

The Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out even when this Oscillator is used as the device clock. For more information on the pre-programmed calibration value, see the section “Calibration Byte” on page 235.

Table 4-3. Internal Calibrated RC Oscillator Operating Modes[1]

Frequency Range[2] (MHz)	CKSEL[3:0][3][4] CSEL[3:0][5]
7.6 - 8.4	0010

Notes:
1. If 8 MHz frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 fuse can be programmed in order to divide the internal frequency by 8.
2. The frequency ranges are guideline values.
3. The device is shipped with this CKSEL = “0010”.
4. Flash Fuse bits.
5. CLKSELR register bits.

When this Oscillator is selected, start-up times are determined by the SUT Fuses or by CSUT field as shown in Table 4-4.

Table 4-4. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT[1:0][1] CSUT[1:0][2]	Start-up Time from Power-down/save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00[3]	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4.1 ms	Fast rising power
10[4]	6 CK	14CK + 65 ms	Slowly rising power
11	Reserved		

Notes:
1. Flash Fuse bits
2. CLKSELR register bits
3. This setting is only available if RSTDISBL fuse is not set
4. The device is shipped with this option selected.

4.2.3 128 KHz Internal Oscillator

The 128 KHz internal Oscillator is a low power Oscillator providing a clock of 128 KHz. The frequency is nominal at 3V and 25°C. This clock may be selected as the system clock by programming CKSEL Fuses or CSEL field as shown in Table 4-1 on page 33.

When this clock source is selected, start-up times are determined by the SUT Fuses or by CSUT field as shown in Table 4-5.

Table 4-5. Start-up Times for the 128 kHz Internal Oscillator

SUT[1:0][1] CSUT[1:0][2]	Start-up Time from Power-down/save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00[3]	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4.1 ms	Fast rising power
10	6 CK	14CK + 65 ms	Slowly rising power
11	Reserved		

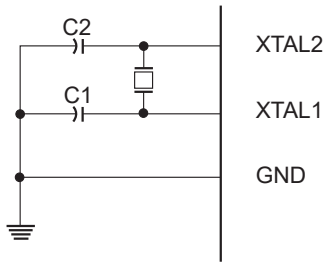
Notes:
1. Flash Fuse bits
2. CLKSELR register bits
3. This setting is only available if RSTDISBL fuse is not set

4.2.4 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 4-2. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 4-6. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 4-2. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by CKSEL[3:1] fuses or by CSEL[3:1] field as shown in Table 4-6.

Table 4-6. Crystal Oscillator Operating Modes

CKSEL[3:1](1) CSEL[3:1](2)	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100(3)	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 - 16.0	12 - 22

Notes:
1. Flash Fuse bits.
2. CLKSELR register bits.
3. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT[1:0] Fuses or CSEL0 together with CSUT[1:0] field select the start-up times as shown in Table 4-7.

Table 4-7. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0[1] CSEL0[2]	SUT[1:0][1] CSUT[1:0][2]	Start-up Time from Power-down/save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[3]	14CK + 4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK[3]	14CK + 65 ms	Ceramic resonator, slowly rising power
0	10[5]	1K (1024) CK[4]	14CK	Ceramic resonator, BOD enabled
0	11	1K (1024)CK[4]	14CK + 4.1 ms	Ceramic resonator, fast rising power
1	00	1K (1024)CK[4]	14CK + 65 ms	Ceramic resonator, slowly rising power
1	01[5]	16K (16384) CK	14CK	Crystal Oscillator, BOD enabled
1	10	16K (16384) CK	14CK + 4.1 ms	Crystal Oscillator, fast rising power
1	11	16K (16384) CK	14CK + 65 ms	Crystal Oscillator, slowly rising power

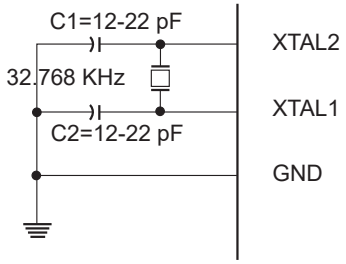
Notes:
1. Flash Fuse bits.
2. CLKSELR register bits.
3. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
4. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.
5. This setting is only available if RSTDISBL fuse is not set.

4.2.5 Low-frequency Crystal Oscillator

To use a 32.768 kHz watch crystal as the clock source for the device, the low-frequency crystal oscillator must be selected by setting CKSEL fuses or CSEL field as shown in Table 4-1 on page 33. The crystal should be connected as shown in Figure 4-3. Refer to the 32.768 kHz Crystal Oscillator Application Note for details on oscillator operation and how to choose appropriate values for C1 and C2.

The 32.768 kHz watch crystal oscillator can be used by the asynchronous timer if the (high-frequency) Crystal Oscillator is not running or if the External Clock is not enabed (See ”Enable/Disable Clock Source” on page 40.). The asynchronous timer is then able to start itself this low-frequency crystal oscillator.

Figure 4-3. Low-frequency Crystal Oscillator Connections



12-22 pF capacitors may be necessary if parasitic impedance (pads, wires & PCB) is very low.

When this oscillator is selected, start-up times are determined by the SUT fuses or by CSUT field as shown in Table 4-8.

Table 4-8. Start-up Times for the Low Frequency Crystal Oscillator Clock Selection

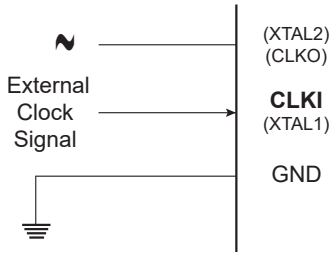
SUT[1:0](1) CSUT[1:0](2)	Start-up Time from Power-down/save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended usage
00	1K (1024) CK(3)	4.1 ms	Fast rising power or BOD enabled
01	1K (1024) CK(3)	65 ms	Slowly rising power
10	32K (32768) CK	65 ms	Stable frequency at start-up
11	Reserved		

Notes:
1. Flash Fuse bits.
2. CLKSELR register bits.
3. These options should only be used if frequency stability at start-up is not important for the application.

4.2.6 External Clock

To drive the device from this external clock source, CLKI should be driven as shown in Figure 4-4. To run the device on an external clock, the CKSEL Fuses or CSEL field must be programmed as shown in Table 4-1 on page 33.

Figure 4-4. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses or CSUT field as shown in Table 4-9.

This external clock can be used by the asynchronous timer if the high or low frequency Crystal Oscillator is not running (See "Enable/Disable Clock Source" on page 40.). The asynchronous timer is then able to enable this input.

Table 4-9. Start-up Times for the External Clock Selection

SUT[1:0][1] CSUT[1:0][2]	Start-up Time from Power-down/save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	14CK (+ 4.1 ms[3])	BOD enabled
01	6 CK	14CK + 4.1 ms	Fast rising power
10	6 CK	14CK + 65 ms	Slowly rising power
11	Reserved		

Notes: 1. Flash Fuse bits.
2. CLKSELR register bits.
3. Additional delay (+ 4ms) available if RSTDISBL fuse is set.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to “System Clock Prescaler” on page 45 for details.

4.2.7 Clock Output Buffer

If not using a crystal oscillator, the device can output the system clock on the CLKO pin. To enable the output, the CKOUT Fuse or COUT bit of CLKSELR register has to be programmed. This option is useful when the device clock is needed to drive other circuits on the system. Note that the clock will not be output during reset and the normal operation of I/O pin will be overridden when the fuses are programmed. If the System Clock Prescaler is used, it is the divided system clock that is output.

4.3 Dynamic Clock Switch

4.3.1 Features

The ATtiny87/167 provides a powerful dynamic clock switch circuit that allows users to turn on and off clocks of the device on the fly. The built-in de-glitching circuitry allows clocks to be enabled or disabled asynchronously. This enables efficient power management schemes to be implemented easily and quickly. In a safety application, the dynamic clock switch circuit allows continuous monitoring of the external clock permitting a fallback scheme in case of clock failure.

The control of the dynamic clock switch circuit must be supervised by software. This operation is facilitated by the following features:

- **Safe commands**, to avoid unintentional commands, a special write procedure must be followed to change the CLKCSR register bits (See “CLKPR – Clock Prescaler Register” on page 46.):
- **Exclusive action**, the actions are controlled by a decoding table (commands) written to the CLKCSR register. This ensures that only one command operation can be launched at any time. The main actions of the decoding table are:
  - ‘*Disable Clock Source*’,
  - ‘*Enable Clock Source*’,
  - ‘*Request Clock Availability*’,
  - ‘*Clock Source Switching*’,
  - ‘*Recover System Clock Source*’,

– ‘*Enable Watchdog in Automatic Reload Mode*’.

- **Command status return.** The ‘*Request Clock Availability* ’ command returns status via the CLKRDY bit in the CLKCSR register. The ‘*Recover System Clock Source* ’ command returns a code of the current clock source in the CLKSELR register. This information is used in the supervisory software routines as shown in Section 4.3.7 on page 41.

4.3.2 CLKSELR Register

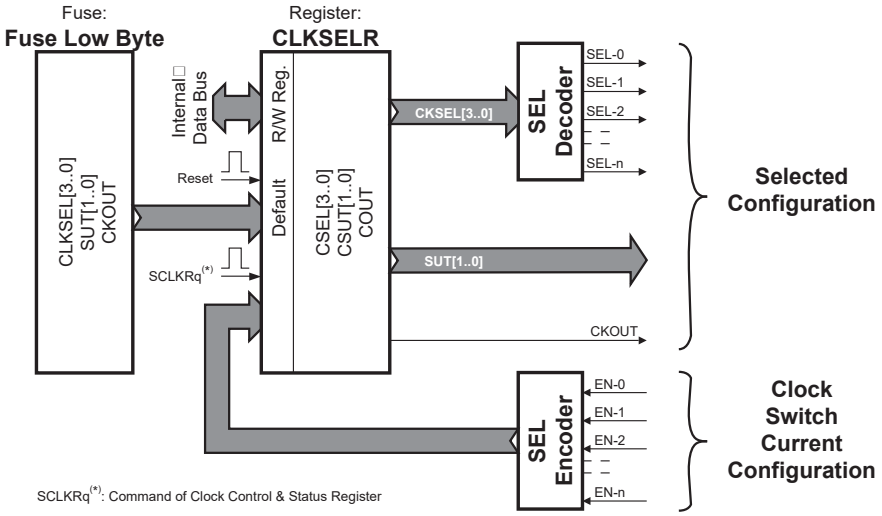
4.3.2.1 Fuses Substitution

At reset, bits of the Low Fuse Byte are copied into the CLKSELR register. The content of this register can subsequently be user modified to overwrite the default values from the Low Fuse Byte. CKSEL[3:0], SUT[1:0] and CKOUT fuses correspond respectively to CSEL[3:0], CSUT[1:0] and ~(COUT) bits of the CLKSELR register as shown in Figure 4-5 on page 40.

4.3.2.2 Source Selection

The available codes of clock source are given in Table 4-1 on page 33.

Figure 4-5. Fuses substitution and Clock Source Selection



The CLKSELR register contains the CSEL, CSUT and COUT values which will be used by the ‘Enable/Disable Clock Source’, ‘Request for Clock Availability’ or ‘Clock Source Switching’ commands.

4.3.2.3 Source Recovering

The ‘*Recover System Clock Source*’ command updates the CKSEL field of CLKSELR register (See “System Clock Source Recovering” on page 41.).

4.3.3 Enable/Disable Clock Source

The ‘*Enable Clock Source*’ command selects and enables a clock source configured by the settings in the CLKSELR register. CSEL[3:0] will select the clock source and CSUT[1:0] will select the start-up time (just as CKSEL and SUT fuse bits do). To be sure that a clock source is operating, the ‘*Request for Clock Availability* ’ command must be executed after the ‘*Enable Clock Source*’ command. This will indicate via the CLKRDY bit in the CLKCSR register that a valid clock source is available and operational.

The '*Disable Clock Source*' command disables the clock source indicated by the settings of CLKSELR register (only CSEL[3:0]). If the clock source indicated is currently the one that is used to drive the system clock, the command is not executed.

Because the selected configuration is latched at clock source level, it is possible to enable many clock sources at a given time (ex: the internal RC oscillator for system clock + an oscillator with external crystal). The user (code) is responsible of this management.

4.3.4 COUT Command

The '*CKOUT* ' command allows to drive the CLKO pin. Refer to Section 4.2.7 "Clock Output Buffer" on page 39 for using.

4.3.5 Clock Availability

'Request for Clock Availability' command enables a hardware oscillation cycle counter driven by the selected source clock, CSEL[3:0]. The count limit value is determined by the settings of CSUT[1:0]. The clock is declared ready (CLKRDY = 1) when the count limit value is reached. The CLKRDY flag is reset when the count starts. Once set, this flag remains unchanged until a new count is commanded. To perform this checking, the CKSEL and CSUT fields should not be changed while the operation is running.

Note that once the new clock source is selected ('*Enable Clock Source*' command), the count procedure is automatically started. The user (code) should wait for the setting of the CLKRDY flag in CLKSCR register before using a newly selected clock.

At any time, the user (code) can ask for the availability of a clock source. The user (code) can request it by writing the *'Request for Clock Availability '* command in the CLKSCR register. A full polling of the status of clock sources can thus be done.

4.3.6 System Clock Source Recovering

The '*Recover System Clock Source*' command returns the current clock source used to drive the system clock as per Table 4-1 on page 33. The CKSEL field of CLKSELR register is then updated with this returned value. There is no information on the SUT used or status on CKOUT.

4.3.7 Clock Switching

To drive the system clock, the user can switch from the current clock source to any other of the following ones (one of them being the current clock source):

1. Calibrated internal RC oscillator 8.0 MHz,
2. Internal watchdog oscillator 128 kHz,
3. External clock,
4. External low-frequency oscillator,
5. External Crystal/Ceramic Resonator.

The clock switching is performed by a sequence of commands. First, the user (code) must make sure that the new clock source is operating. Then the '*Clock Source Switching*' command can be issued. Once this command has been successfully completed using the '*Recover System Clock Source*' command, the user (code) may stop the previous clock source.

It is strongly recommended to run this sequence only once the interrupts have been disabled. The user (code) is responsible for the correct implementation of the clock switching sequence.

Here is a “light” C-code that describes such a sequence of commands.

C Code Example

```
void ClockSwiching (unsigned char clk_number, unsigned char sut) {

#define CLOCK_RECOVER   0x05
#define CLOCK_ENABLE    0x02
#define CLOCK_SWITCH    0x04
#define CLOCK_DISABLE   0x01

unsigned char previous_clk, temp;

  // Disable interrupts
    temp = SREG; asm ("cli");
  // Save the current system clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_RECOVER;
    previous_clk = CLKSELR & 0x0F;
  // Enable the new clock source
    CLKSELR = ((sut << 4 ) & 0x30) | (clk_number & 0x0F);
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_ENABLE;
  // Wait for clock validity
    while ((CLKCSR & (1 << CLKRDY)) == 0);
  // Switch clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_SWITCH;
  // Wait for effective switching
    while (1){
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK_RECOVER;
        if ((CLKSELR & 0x0F) == (clk_number & 0x0F)) break;
    }
  // Shut down unneeded clock source
    if (previous_clk != (clk_number & 0x0F)) {
        CLKSELR = previous_clk;
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK_DISABLE;
    }
  // Re-enable interrupts
    SREG = temp;
}
```

Warning:

In the ATtiny87/167, only one among the three external clock sources can be enabled at a given time. Moreover, the enables of the external clock and of the external low-frequency oscillator are shared with the asynchronous timer.

4.3.8 Clock Monitoring

A safe system needs to monitor its clock sources. Two domains need to be monitored:
- Clock sources for peripherals,
- Clocks sources for system clock generation.

In the first domain, the user (code) can easily check the validity of the clock(s) (See “COUT Command” on page 41.).

In the second domain, the lack of a clock results in the code not running. Thus, the presence of the system clock needs to be monitored by hardware.

Using the on-chip watchdog allows this monitoring. Normally, the watchdog reloading is performed only if the code reaches some specific software labels, reaching these labels proves that the system clock is running. Otherwise the watchdog reset is enabled. This behavior can be considered as a clock monitoring.

If the standard watchdog functionality is not desired, the ATtiny87/167 watchdog permits the system clock to be monitored without having to resort to the complexity of a full software watchdog handler. The solution proposed in the ATtiny87/167 is to automate the watchdog reloading with only one command, at the beginning of the session.

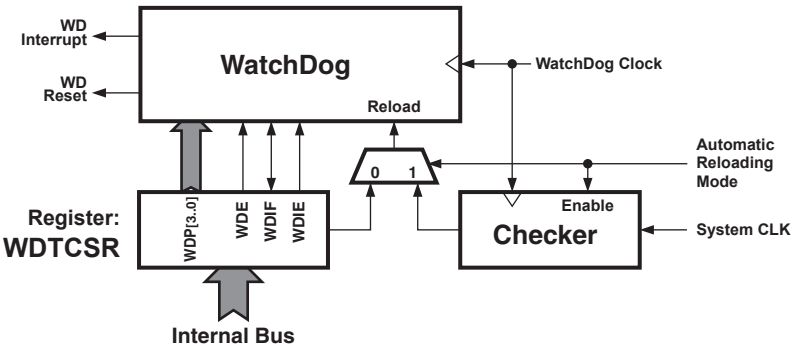
So, to monitor the system clock, the user will have two options:

1. Using the standard watchdog features (software reload),
2. Or using the automatic reloading (hardware reload).

The two options are exclusive.

Note: Warning:
These two options make sense ONLY if the clock source at RESET is an INTERNAL SOURCE. The fuse settings determine this operation.

Figure 4-6. Watchdog Timer with Automatic Reloading.



The '*Enable Watchdog in Automatic Reload Mode*' command has priority over the standard watchdog enabling. In this mode, only the reset function of the watchdog is enabled (no more watchdog interrupt). The WDP[3:0] bits of the WDTCSR register always determine the watchdog timer prescaling.

As the watchdog will not be active before executing the '*Enable Watchdog in Automatic Reload Mode*' command, it is recommended to activate this command before switching to an external clock source (See note on page 43).

Notes:
1. ONLY the reset (watchdog reset included) disables this function. The Watchdog System Reset Flag (WDRF bit of MCUSR register) can be used to monitor the reset cause.
2. ONLY clock frequencies greater than or equal to (4 * WatchDog Clock Frequency) can be monitored.

Here is a “light” C-code of a clock switching function using automatic clock monitoring.

C Code Example

```
void ClockSwiching (unsigned char clk_number, unsigned char sut) {

#define CLOCK_RECOVER  0x05
#define CLOCK_ENABLE   0x02
#define CLOCK_SWITCH   0x04
#define CLOCK_DISABLE  0x01
#define WD_ARL_ENABLE  0x06

#define WD_2048CYCLES  0x07

unsigned char previous_clk, temp;

  // Disable interrupts
    temp = SREG; asm ("cli");
  // Save the current system clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_RECOVER;
    previous_clk = CLKSELR & 0x0F;
  // Enable the new clock source
    CLKSELR = ((sut << 4 ) & 0x30) | (clk_number & 0x0F);
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_ENABLE;
  // Wait for clock validity
    while ((CLKCSR & (1 << CLKRDY)) == 0);

  // Enable the watchdog in automatic reload mode
    WDTCSR = (1 << WDCE) | (1 << WDE);
    WDTCSR = (1 << WDE ) | WD_2048CYCLES;
    CLKCSR = 1 << CLKCCE;
    CLKCSR = WD_ARL_ENABLE;

  // Switch clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_SWITCH;
  // Wait for effective switching
    while (1){
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK_RECOVER;
        if ((CLKSELR & 0x0F) == (clk_number & 0x0F)) break;
    }
  // Shut down unneeded clock source
    if (previous_clk != (clk_number & 0x0F)) {
    CLKSELR = previous_clk;
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_DISABLE;
}
  // Re-enable interrupts
    SREG = temp;
}
```

4.4 System Clock Prescaler

4.4.1 Features

The ATtiny87/167 system clock can be divided by setting the Clock Prescaler Register – CLKPR. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 4-10 on page 47.

4.4.2 Switching Time

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler – even if it were readable, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2*T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

4.5 Register Description

4.5.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
(0x66)	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **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 Table 22-1 on page 252. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 22-1 on page 252. Calibration outside that range is not guaranteed.

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

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

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

range. Incrementing CAL[6:0] by 1 will give a frequency increment of less than 2% in the frequency range 7.3 - 8.1 MHz.

4.5.2 CLKPR – Clock Prescaler Register

Bit	7	6	5	4	3	2	1	0	
(0x61)	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

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

- **Bits 6:4 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **Bits 3:0 – CLKPS[3:0]: Clock Prescaler Select Bits 3 - 0**

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 Table 4-10.

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

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting in order not to disturb the procedure.

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

Table 4-10. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

4.5.3 CLKCSR – Clock Control & Status Register

Bit	7	6	5	4	3	2	1	0	
(0x62)	CLKCCE	–	–	CLKRDY	CLKC3	CLKC2	CLKC1	CLKC0	CLKCSR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – CLKCCE: Clock Control Change Enable**

The CLKCCE bit must be written to logic one to enable change of the CLKCSR bits. The CLKCCE bit is only updated when the other bits in CLKCSR are simultaneously written to zero. CLKCCE is cleared by hardware four cycles after it is written or when the CLKCSR bits are written. Rewriting the CLKCCE bit within this time-out period does neither extend the time-out period, nor clear the CLKCCE bit.

- **Bits 6:5 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **Bit 4 – CLKRDY: Clock Ready Flag**

This flag is the output of the '*Clock Availability* ' logic.
This flag is cleared by the '*Request for Clock Availability*' command or '*Enable Clock Source*' command being entered.
It is set when '*Clock Availability*' logic confirms that the (selected) clock is running and is stable. The delay from the request and the flag setting is not fixed, it depends on the clock start-up time,

the clock frequency and, of course, if the clock is alive. The user's code has to differentiate between '***no_clock_signal***' and '***clock_signal_not_yet_available***' condition.

- **Bits 3:0 – CLKC[3:0]: Clock Control Bits 3 - 0**

These bits define the command to provide to the '*Clock Switch'* module. The special write procedure must be followed to change the CLKC[3:0] bits (See "Bit 7 – CLKCCE: Clock Control Change Enable" on page 47.).

1. Write the Clock Control Change Enable (CLKCCE) bit to one and all other bits in CLKCSR to zero.
2. Within 4 cycles, write the desired value to CLKCSR register while clearing CLKCCE bit.

Interrupts should be disabled when setting CLKCSR register in order not to disturb the procedure.

Table 4-11. Clock command list.

Clock Command	CLKC[3:0]
No command	0000_b
Disable clock source	0001_b
Enable clock source	0010_b
Request for clock availability	0011_b
Clock source switch	0100_b
Recover system clock source code	0101_b
Enable watchdog in automatic reload mode	0110_b
CKOUT command	0111_b
No command	$1xxx_b$

4.5.4 CLKSELR – Clock Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x63)	–	COUT	CSUT1	CSUT0	CSEL3	CSEL2	CSEL1	CSEL0	CLKSELR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	~ (CKOUT) fuse	SUT[1:0] fuses			CKSEL[3:0] fuses			

- **Bit 7 – Res: Reserved Bit**

This bit is a reserved bit in the ATtiny87/167 and will always read as zero.

- **Bit 6 – COUT: Clock Out**

The COUT bit is initialized with ~(CKOUT) Fuse bit.
The COUT bit is only used in case of '*CKOUT*' command. Refer to Section 4.2.7 "Clock Output Buffer" on page 39 for using.
In case of '*Recover System Clock Source*' command, COUT it is not affected (no recovering of this setting).

- **Bits 5:4 – CSUT[1:0]: Clock Start-up Time**

CSUT bits are initialized with the values of SUT Fuse bits.
In case of '*Enable/Disable Clock Source*' command, CSUT field provides the code of the clock start-up time. Refer to subdivisions of Section 4.2 "Clock Sources" on page 33 for code of clock start-up times.
In case of '*Recover System Clock Source*' command, CSUT field is not affected (no recovering of SUT code).

- **Bits 3:0 – CSEL[3:0]: Clock Source Select**

CSEL bits are initialized with the values of CKSEL Fuse bits.
In case of '*Enable/Disable Clock Source*', '*Request for Clock Availability*' or '*Clock Source Switch*' command, CSEL field provides the code of the clock source. Refer to Table 4-1 on page 33 and subdivisions of Section 4.2 "Clock Sources" on page 33 for clock source codes.
In case of '*Recover System Clock Source*' command, CSEL field contains the code of the clock source used to drive the Clock Control Unit as described in Figure 4-1 on page 32.

5. Power Management and Sleep Modes

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.

When enabled, the Brown-out Detector (BOD) actively monitors the power supply voltage during the sleep periods. To further save power, it is possible to disable the BOD in some sleep modes. See "BOD Disable" on page 51 for more details.

5.1 Sleep Modes

Figure 4-1 on page 32 presents the different clock systems in the ATtiny87/167, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 5-1 shows the different sleep modes, their wake up sources and BOD disable ability.

Table 5-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer0 Osc. Enable	INT1, INT0 and Pin Change	SPM/EEPROM Ready	ADC	WDT	USI Start Condition	Timer0	Other I/O	Software BOD Disable
Idle			X	X	X	X	X	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X	X(1)	X	X	X	X	X		
Power-down								X(1)			X	X			X
Power-Save					X		X	X(1)			X	X	X		X

Note: 1. For INT1 and INT0, only level interrupt.

To enter any of the four sleep modes, the SE bit in SMCR must be written to logic one and a SLEEP instruction must be executed. The SM1, and SM0 bits in the SMCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, or Power-save) will be activated by the SLEEP instruction. See Table 5-2 on page 55 for a summary.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles 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.

5.2 BOD Disable

When the Brown-out Detector (BOD) is enabled by BODLEVEL fuses, Table 21-3 on page 233, the BOD is actively monitoring the power supply voltage during a sleep period. To save power, it is possible to disable the BOD by software for some of the sleep modes, see Table 5-1. The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses. If BOD is disabled in software, the BOD function is turned off immediately after entering the sleep mode. Upon wake-up from sleep, BOD is automatically enabled again. This ensures safe operation in case the V_{CC} level has dropped during the sleep period.

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

BOD disable is controlled by BODS bit (BOD Sleep) in the control register MCUCR, see "MCUCR – MCU Control Register" on page 55. Setting it to one turns off the BOD in relevant sleep modes, while a zero in this bit keeps BOD active. Default setting keeps BOD active, i.e. BODS is cleared to zero.

Writing to the BODS bit is controlled by a timed sequence and an enable bit, see "MCUCR – MCU Control Register" on page 55.

5.3 Idle Mode

When the SM[1:0] bits are written to 00, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the SPI, Analog Comparator, ADC, USI start condition, Asynchronous Timer/Counter, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

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

5.4 ADC Noise Reduction Mode

When the SM[1:0] bits are written to 01, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the external interrupts, the USI start condition, the asynchronous Timer/Counter and the Watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC Conversion Complete interrupt, only an External Reset, a Watchdog System Reset, a Watchdog Interrupt, a Brown-out Reset, a USI start condition interrupt, an asynchronous Timer/Counter interrupt, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or INT1 or a pin change interrupt can wake up the MCU from ADC Noise Reduction mode.

5.5 Power-down Mode

When the SM[1:0] bits are written to 10, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the USI start condition, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog System Reset, a Watchdog Interrupt, a Brown-out Reset, the USI start condition interrupt, an external level interrupt on INT0 or INT1, or a pin change interrupt can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 8. “External Interrupts” on page 70 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period, as described in Section 4.2 “Clock Sources” on page 33.

5.6 Power-save Mode

When the SM[1:0] bits are written to 11, the SLEEP instruction makes the MCU enter Power-save mode. This mode is identical to Power-down, with one exception:

If Timer/Counter0 is clocked asynchronously, i.e., the AS0 bit in ASSR is set, Timer/Counter0 will run during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter0 if the corresponding Timer/Counter0 interrupt enable bits are set in TIMSK0, and the global interrupt enable bit in SREG is set.

If the Asynchronous Timer is **NOT** clocked asynchronously, Power-down mode is recommended instead of Power-save mode because the contents of the registers in the asynchronous timer should be considered undefined after wake-up in Power-save mode if AS0 is 0.

This sleep mode basically halts all clocks except clk_{ASY}, allowing operation only of asynchronous modules, including Timer/Counter0 if clocked asynchronously.

5.7 Power Reduction Register

The Power Reduction Register (PRR), see “PRR – Power Reduction Register” on page 55, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

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

5.8 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.

5.8.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to Section 17. "ADC – Analog to Digital Converter" on page 198 for details on ADC operation.

5.8.2 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to Section 18. "AnaComp - Analog Comparator" on page 218 for details on how to configure the Analog Comparator.

5.8.3 Brown-out Detector

If the Brown-out Detector is not needed by the application, this module should be turned off. If the Brown-out Detector is enabled by the BODLEVEL Fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to Section 6.1.5 "Brown-out Detection" on page 59 for details on how to configure the Brown-out Detector.

5.8.4 Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detection, 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. Refer to Section 6.2 "Internal Voltage Reference" on page 61 for details on the start-up time.

Output the internal voltage reference is not needed in the deeper sleep modes. This module should be turned off to reduce significantly to the total current consumption. Refer to "AMISCR – Analog Miscellaneous Control Register" on page 197 for details on how to disable the internal voltage reference output.

5.8.5 Internal Current Source

The Internal Current Source is not needed in the deeper sleep modes. This module should be turned off to reduce significantly to the total current consumption. Refer to "AMISCR – Analog Miscellaneous Control Register" on page 197 for details on how to disable the Internal Current Source.

5.8.6 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 Section 6.3 “Watchdog Timer” on page 61 for details on how to configure the Watchdog Timer.

5.8.7 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section Section 9.2.6 “Digital Input Enable and Sleep Modes” on page 80 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 (DIDR1 and DIDR0). Refer to Section 17.11.6 “DIDR1 – Digital Input Disable Register 1” on page 217 and Section 17.11.5 “DIDR0 – Digital Input Disable Register 0” on page 216 for details.

5.8.8 On-chip Debug System

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

5.9 Register Description

5.9.1 SMCR – Sleep Mode Control Register

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

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	–	–	–	–	–	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

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

These bits select between the four available sleep modes as shown in Table 5-2.

Table 5-2. Sleep Mode Select

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

- **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.

5.9.2 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	–	–	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – BODS: BOD Sleep**

The BODS bit must be written to logic one in order to turn off BOD during sleep, see Table 5-1 on page 50. Writing to the BODS bit is controlled by a timed sequence and an enable bit, BODSE in MCUCR. To disable BOD in relevant sleep modes, both BODS and BODSE must first be set to one. Then, to set the BODS bit, BODS must be set to one and BODSE must be set to zero within four clock cycles.

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

- **Bit 5 – BODSE: BOD Sleep Enable**

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

5.9.3 PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
(0x64)	–	–	PRLIN	PRSPI	PRTIM1	PRTIM0	PRUSI	PRADC	PRR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **Bit 5 – PRLIN: Power Reduction LIN / UART controller**

Writing a logic one to this bit shuts down the LIN by stopping the clock to the module. When waking up the LIN again, the LIN should be re initialized to ensure proper operation.

- **Bit 4 – PRSPI: Power Reduction Serial Peripheral Interface**

If using debugWIRE On-chip Debug System, this bit should not be written to one.

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

- **Bit 3 – PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2 – PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module in synchronous mode (AS0 is 0). When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

- **Bit 1 – PRUSI: Power Reduction USI**

Writing a logic one to this bit shuts down the USI by stopping the clock to the module. When waking up the USI again, the USI should be re-initialized to ensure proper operation.

- **Bit 0 – PRADC: Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

6. System Control and Reset

6.1 Reset

6.1.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be an RJMP – Relative Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 6-1 shows the reset circuit. Tables in Section 22.5 “RESET Characteristics” on page 253 defines the electrical parameters of the reset circuitry.

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

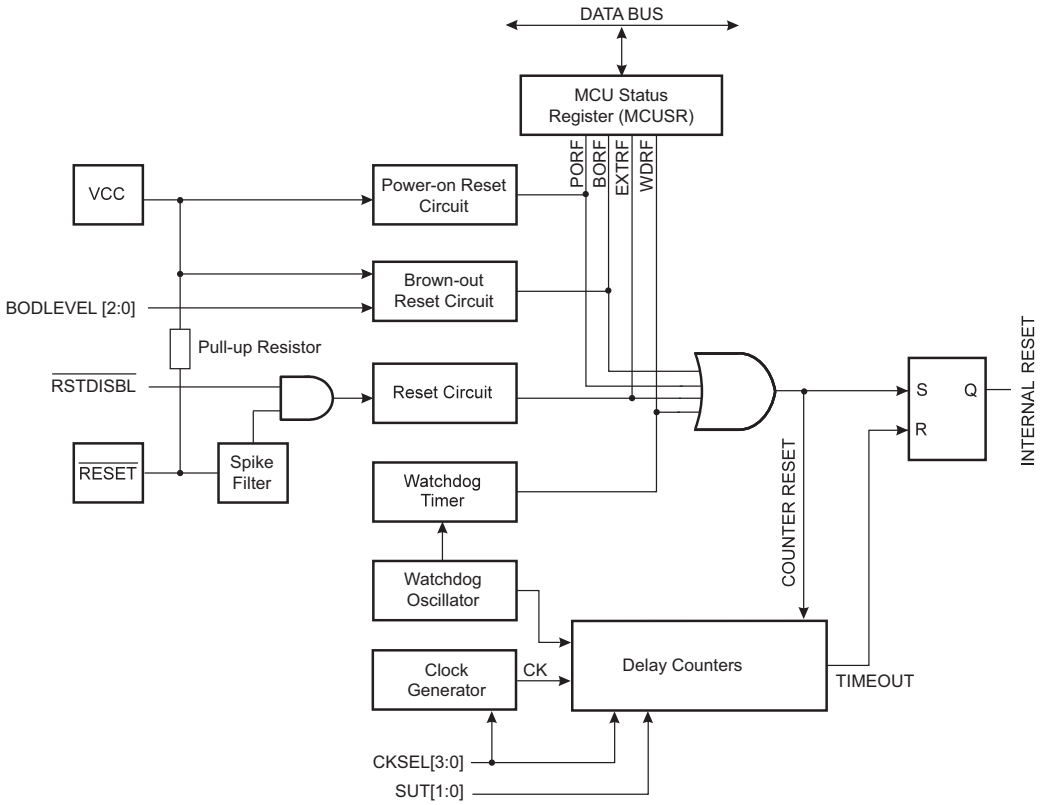
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL Fuses. The different selections for the delay period are presented in Section 4.2 “Clock Sources” on page 33.

6.1.2 Reset Sources

The ATtiny87/167 has four sources of reset:

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

Figure 6-1. Reset Circuit



6.1.3 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 22-4 on page 253. 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 6-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

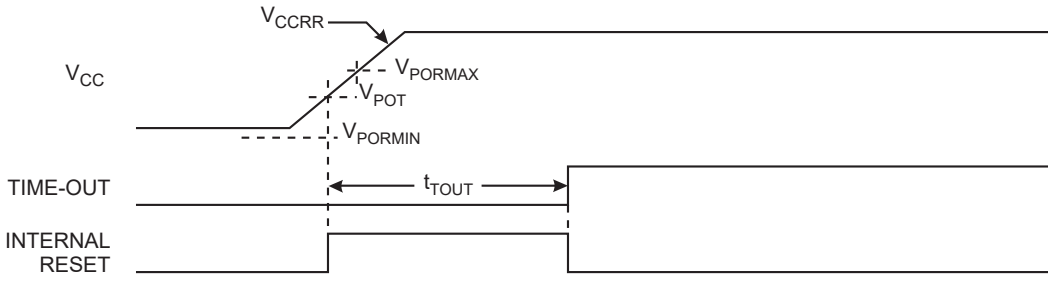
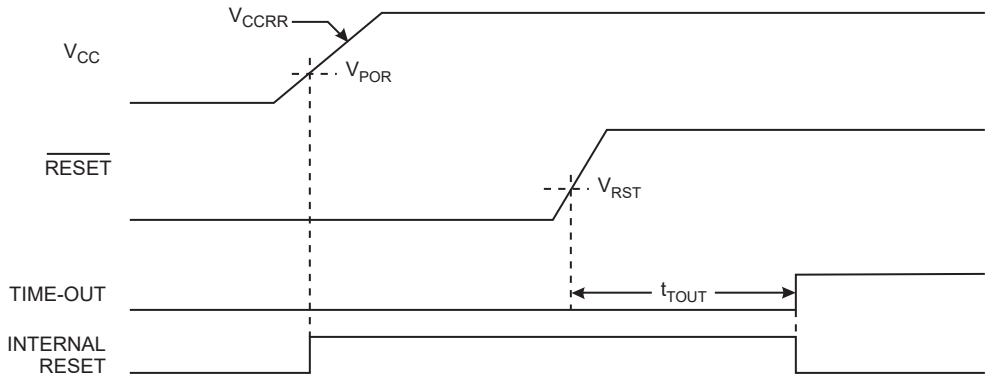


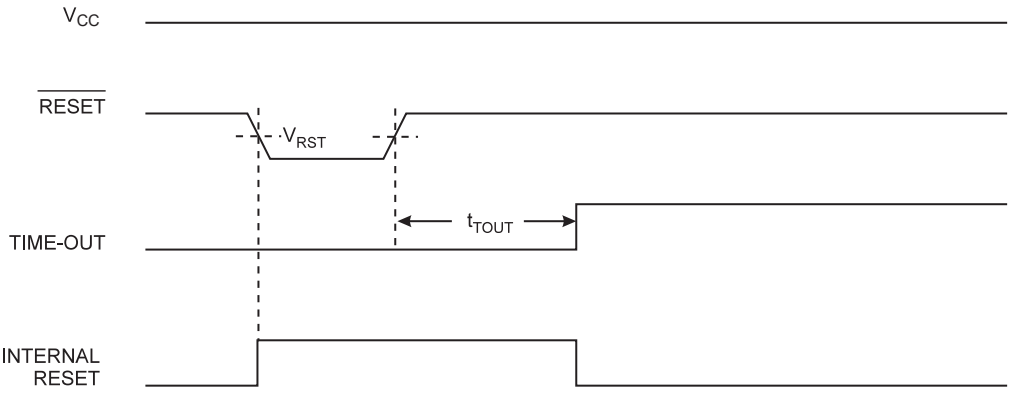
Figure 6-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



6.1.4 External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 22-3 on page 253) 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, see Table 21-4 on page 233.

Figure 6-4. External Reset During Operation



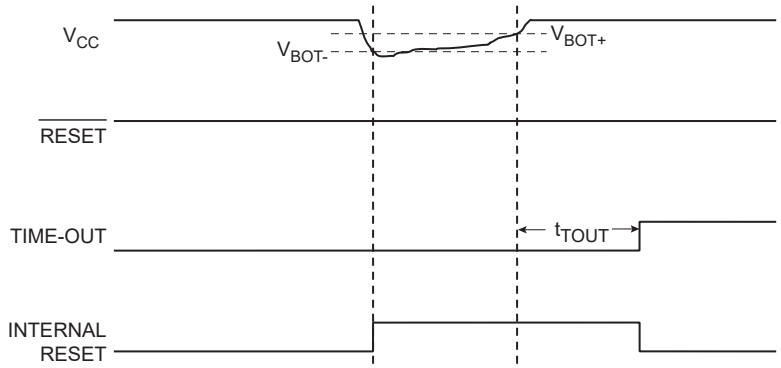
6.1.5 Brown-out Detection

ATtiny87/167 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses (See "BODLEVEL Fuse Coding" on page 254.). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST} / 2$ and $V_{BOT-} = V_{BOT} - V_{HYST} / 2$.

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 6-5), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 6-5), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 22-6 on page 254.

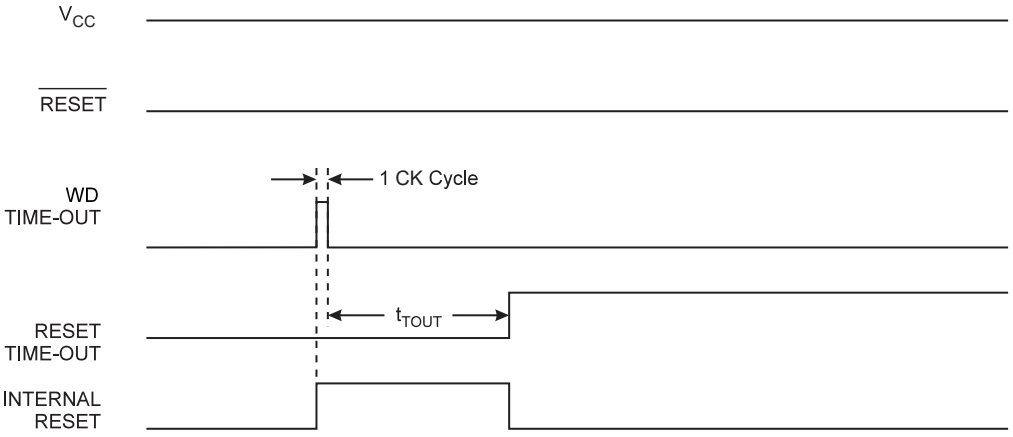
Figure 6-5. Brown-out Reset During Operation



6.1.6 Watchdog System Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 61 for details on operation of the Watchdog Timer.

Figure 6-6. Watchdog System Reset During Operation



6.1.7 MCUSR – MCU Status Register

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

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

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

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

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

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

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

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

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

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

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

6.2 Internal Voltage Reference

ATtiny87/167 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC.

6.2.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 22-7 on page 254. To save power, the reference is not always turned on. The reference is on during the following situations:

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

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

6.3 Watchdog Timer

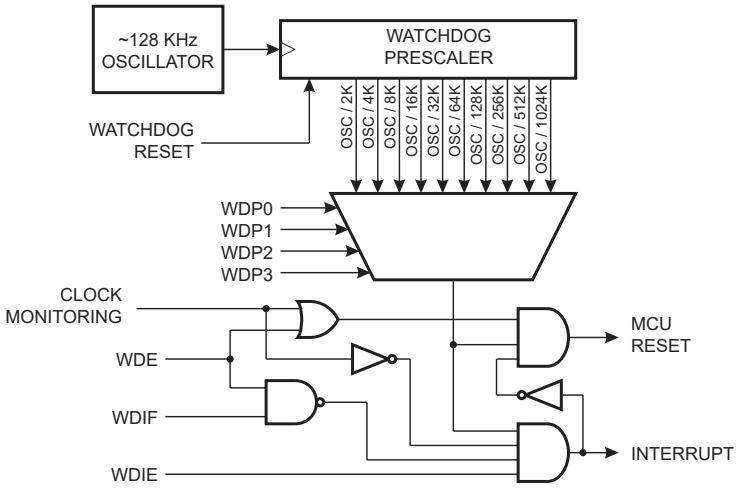
ATtiny87/167 has an Enhanced Watchdog Timer (WDT). The main features are:

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

6.3.1 Watchdog Timer Behavior

The Watchdog Timer (WDT) is a timer counting cycles of a separate on-chip 128 KHz oscillator.

Figure 6-7. Watchdog Timer



The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

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

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

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

The following code example shows one assembly and one C function for turning off the Watchdog Timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example[1]

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

C Code Example[1]

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

Note: 1. See "About Code Examples" on page 14.

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

The following code example shows one assembly and one C function for changing the time-out value of the Watchdog Timer.

Assembly Code Example[1]

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

C Code Example[1]

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

Notes:
1. See "About Code Examples" on page 14.
2. The Watchdog Timer should be reset before any change of the WDP bits, since a change in the WDP bits can result in a time-out when switching to a shorter time-out period.

6.3.2 Clock monitoring

The Watchdog Timer can be used to detect a loss of system clock. This configuration is driven by the dynamic clock switch circuit. Please refer to Section 4.3.8 "Clock Monitoring" on page 42 for more information.

6.3.3 WDTCR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	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.

If the Watchdog Timer is used as clock monitor (c.f. Section • “Bits 3:0 – CLKC[3:0]: Clock Control Bits 3 - 0” on page 48), the System Reset Mode is enabled and the Interrupt Mode is automatically disabled.

Table 6-1. Watchdog Timer Configuration

Clock Monitor	WDTON	WDE	WDIE	Mode	Action on Time-out
x	0	0	0	Stopped	None
On	y(1)	y(1)	y(1)	System Reset Mode	Reset
Off	0	0	1	Interrupt Mode	Interrupt
	0	1	0	System Reset Mode	Reset
	0	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
	1	x	x	System Reset Mode	Reset

Note: 1. At least one of these three enables (WDTON, WDE & WDIE) equal to 1.

- **Bit 4 – WDCE: Watchdog Change Enable**

This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set.

Once written to one, hardware will clear WDCE after four clock cycles.

- **Bit 3 – WDE: Watchdog System Reset Enable**

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

- **Bits 5, 2:0 – WDP[3:0]: Watchdog Timer Prescaler 3 - 0**

These 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 Table 6-2 on page 66.

Table 6-2. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

7. Interrupts

This section describes the specifics of the interrupt handling as performed in ATtiny87/167. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 20.

7.1 Interrupt Vectors in ATtiny87/167

Table 7-1. Reset and Interrupt Vectors in ATtiny87/167

Vector Nb.	Program Address		Source	Interrupt Definition
	ATtiny87	ATtiny167		
1	0x0000	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0001	0x0002	INT0	External Interrupt Request 0
3	0x0002	0x0004	INT1	External Interrupt Request 1
4	0x0003	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0004	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x0005	0x000A	WDT	Watchdog Time-out Interrupt
7	0x0006	0x000C	TIMER1_CAPT	Timer/Counter1 Capture Event
8	0x0007	0x000E	TIMER1_COMPA	Timer/Counter1 Compare Match A
9	0x0008	0x0010	TIMER1_COMPB	Timer/Counter1 Compare Match B
10	0x0009	0x0012	TIMER1_OVF	Timer/Counter1 Overflow
11	0x000A	0x0014	TIMER0_COMPA	Timer/Counter0 Compare Match A
12	0x000B	0x0016	TIMER0_OVF	Timer/Counter0 Overflow
13	0x000C	0x0018	LIN_TC	LIN/UART Transfer Complete
14	0x000D	0x001A	LIN_ERR	LIN/UART Error
15	0x000E	0x001C	SPI_STC	SPI Serial Transfer Complete
16	0x000F	0x001E	ADC	ADC Conversion Complete
17	0x0010	0x0020	EE_RDY	EEPROM Ready
18	0x0011	0x0022	ANA_COMP	Analog Comparator
19	0x0012	0x0024	USI_START	USI Start Condition Detection
20	0x0013	0x0026	USI_OVF	USI Counter Overflow

7.2 Program Setup in ATtiny87

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATtiny87 is (2-byte step - using “rjmp” instruction):

```
Address(Note:)Label  Code                 Comments
0x0000           rjmp   RESET           ; Reset Handler
0x0001           rjmp   INT0addr        ; IRQ0 Handler
0x0002           rjmp   INT1addr        ; IRQ1 Handler
0x0003           rjmp   PCINT0addr      ; PCINT0 Handler
0x0004           rjmp   PCINT1addr      ; PCINT1 Handler
0x0005           rjmp   WDTaddr         ; Watchdog Timer Handler
0x0006           rjmp   ICP1addr        ; Timer1 Capture Handler
0x0007           rjmp   OC1Aaddr        ; Timer1 Compare A Handler
0x0008           rjmp   OC1Baddr        ; Timer1 Compare B Handler
0x0009           rjmp   OVF1addr        ; Timer1 Overflow Handler
0x000A           rjmp   OC0Aaddr        ; Timer0 Compare A Handler
0x000B           rjmp   OVF0addr        ; Timer0 Overflow Handler
0x000C           rjmp   LINTCaddr       ; LIN Transfer Complete Handler
0x000D           rjmp   LINERRaddr      ; LIN Error Handler
0x000E           rjmp   SPIaddr         ; SPI Transfer Complete Handler
0x000F           rjmp   ADCCaddr        ; ADC Conversion Complete Handler
0x0010           rjmp   ERDYaddr        ; EEPROM Ready Handler
0x0011           rjmp   ACIaddr         ; Analog Comparator Handler
0x0012           rjmp   USISTARTaddr    ; USI Start Condition Handler
0x0013           rjmp   USIOVFaddr      ; USI Overflow Handler

0x0014 RESET:    ldi    r16, high(RAMEND); Main program start
0x0015           out    SPH,r16          ; Set Stack Pointer to top of RAM
0x0016           ldi    r16, low(RAMEND)
0x0017           out    SPL,r16
0x0018           sei                     ; Enable interrupts
0x0019           <instr>  xxx
  ...     ...    ...    ...
```

Note: 16-bit address

7.3 Program Setup in ATtiny167

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATtiny167 is (4-byte step - using “jmp” instruction):

```
Address(Note:) Label  Code                 Comments
0x0000         jmp    RESET          ; Reset Handler
0x0002         jmp    INT0addr       ; IRQ0 Handler
0x0004         jmp    INT1addr       ; IRQ1 Handler
0x0006         jmp    PCINT0addr     ; PCINT0 Handler
0x0008         jmp    PCINT1addr     ; PCINT1 Handler
0x000A         jmp    WDTaddr        ; Watchdog Timer Handler
0x000C         jmp    ICP1addr       ; Timer1 Capture Handler
0x000E         jmp    OC1Aaddr       ; Timer1 Compare A Handler
0x0010         jmp    OC1Baddr       ; Timer1 Compare B Handler
0x0012         jmp    OVF1addr       ; Timer1 Overflow Handler
0x0014         jmp    OC0Aaddr       ; Timer0 Compare A Handler
0x0016         jmp    OVF0addr       ; Timer0 Overflow Handler
0x0018         jmp    LINTCaddr      ; LIN Transfer Complete Handler
0x001A         jmp    LINERRaddr     ; LIN Error Handler
0x001C         jmp    SPIaddr        ; SPI Transfer Complete Handler
0x001E         jmp    ADCCaddr       ; ADC Conversion Complete Handler
0x0020         jmp    ERDYaddr       ; EEPROM Ready Handler
0x0022         jmp    ACIaddr        ; Analog Comparator Handler
0x0024         jmp    USISTARTaddr   ; USI Start Condition Handler
0x0026         jmp    USIOVFaddr     ; USI Overflow Handler

0x0028 RESET:  ldi    r16, high(RAMEND); Main program start
0x0029         out    SPH,r16          ; Set Stack Pointer to top of RAM
0x002A         ldi    r16, low(RAMEND)
0x002B         out    SPL,r16
0x002C         sei                     ; Enable interrupts
0x002D         <instr>  xxx
  ...     ...    ...    ...
```

Note: 16-bit address

8. External Interrupts

8.1 Overview

The External Interrupts are triggered by the INT0 or INT1 pin or any of the PCINT[15:0] pins. Observe that, if enabled, the interrupts will trigger even if the INT0, INT1 or PCINT[15:0] pins are configured as outputs. This feature provides a way of generating a software interrupt.

The pin change interrupt PCINT1 will trigger if any enabled PCINT[15:8] pin toggles. The pin change interrupt PCINT0 will trigger if any enabled PCINT[7:0] pin toggles. The PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT[15:0] are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

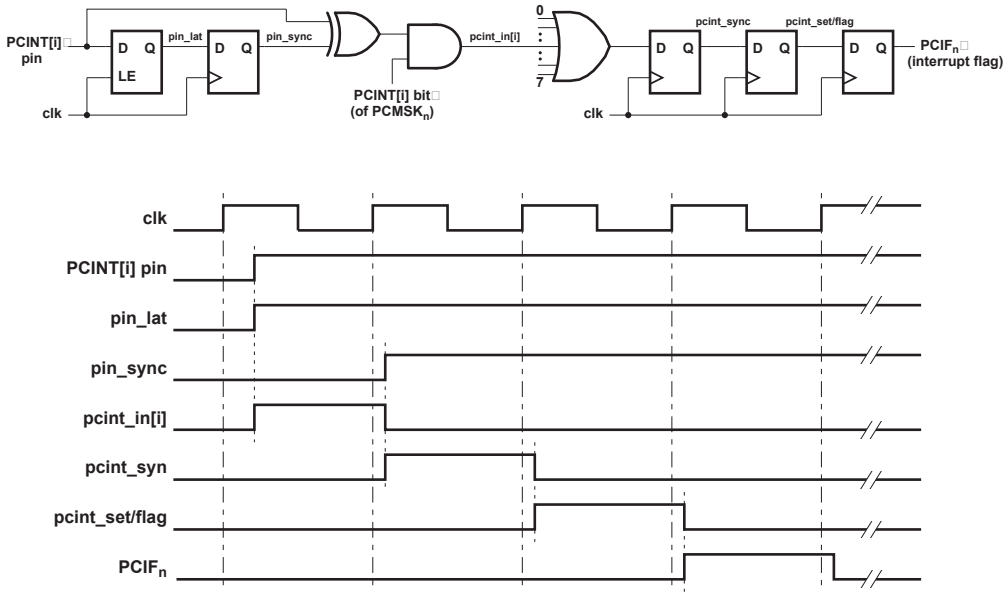
The INT0 or INT1 interrupt can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A – EICRA. When the INT0 or INT1 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. The recognition of falling or rising edge interrupts on INT0 or INT1 requires the presence of an I/O clock, described in “Clock Systems and their Distribution” on page 32. Low level interrupts and the edge interrupt on INT0 or INT1 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down or Power-save, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses as described in “Clock Systems and their Distribution” on page 32.

8.2 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 8-1.

Figure 8-1. Timing of pin change interrupts



8.3 Register Description

8.3.1 EICRA – External Interrupt Control Register A

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

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **Bits 3:2 – ISC1[1:0]: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 8-1. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

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

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 8-1. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 8-1. Interrupt Sense Control

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request.
1	0	The falling edge of INTn generates an interrupt request.
1	1	The rising edge of INTn generates an interrupt request.

8.3.2 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

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

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

- **Bit 0 – INT0: External Interrupt Request 0 Enable**

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

8.3.3 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **Bit 1 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 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 INT1 is configured as a level interrupt.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 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.

8.3.4 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	–	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **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[15:8] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT[15:8] 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 PCMSK0 Register.

8.3.5 PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	–	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

- **Bit 1 – PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT[15:8] 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.

8.3.6 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
(0x6C)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – PCINT[15:8]: Pin Change Enable Mask 15:8**

Each PCINT[15:8] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[15:8] is set and the PCIE1 bit in EIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[15:8] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

8.3.7 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:0 – PCINT[7:0]: Pin Change Enable Mask 7:0**

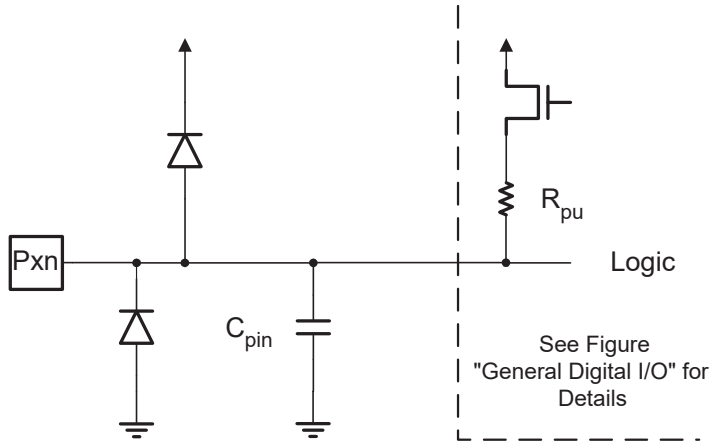
Each PCINT[7:0] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is set and the PCIE0 bit in EIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

9. I/O-Ports

9.1 Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 9-1. Refer to "Electrical Characteristics" on page 250 for a complete list of parameters.

Figure 9-1. 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. The physical I/O Registers and bit locations are listed in "Register Description" on page 93.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one 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 or PUDx in PORTCR disables the pull-up function for all pins in all ports when set.

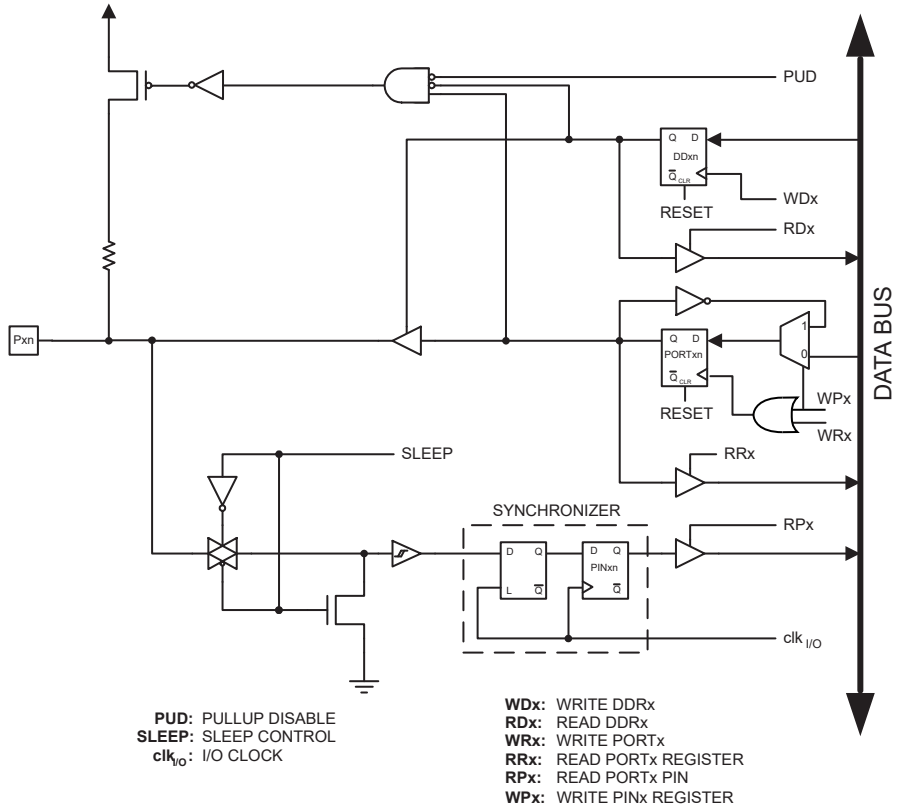
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 76. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 80. Refer to the individual module sections for a full description of the alternate functions.

Note that 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.

9.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 9-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 9-2. General Digital I/O(Note:)



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

9.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description" on page 93, 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 logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one 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 logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

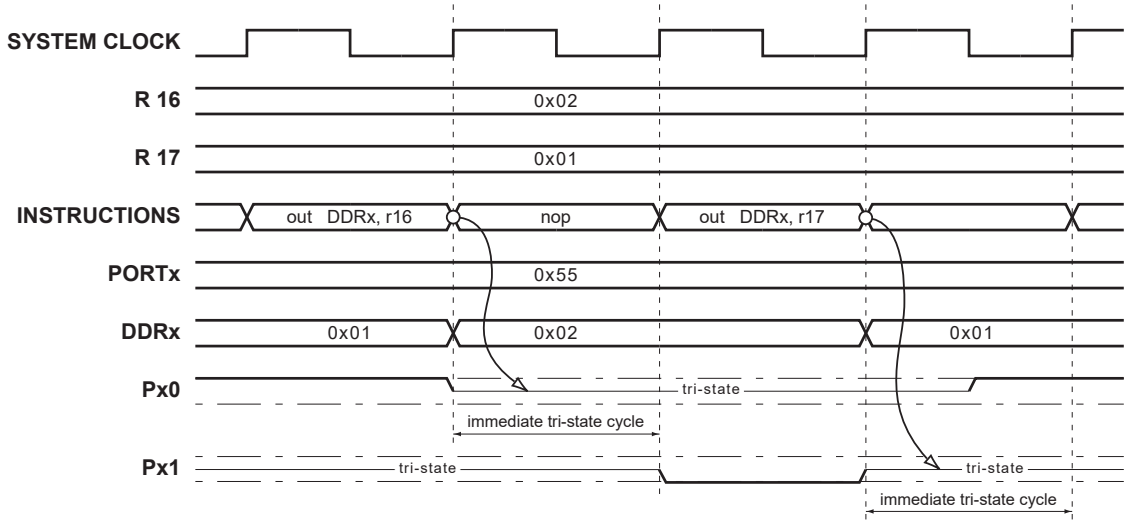
9.2.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI assembler instruction can be used to toggle one single bit in a port.

9.2.3 Break-Before-Make Switching

In the Break-Before-Make mode when switching the DDRxn bit from input to output an immediate tri-state period lasting one system clock cycle is introduced as indicated in Figure 9-3. For example, if the system clock is 4 MHz and the DDRxn is written to make an output, the 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 is a port-wise mode and it is activated by the port-wise BBMx enable bits. For further information about the BBMx bits, see “PORTCR – Port Control Register” on page 83. When switching the DDRxn bit from output to input there is no immediate tri-state period introduced.

Figure 9-3. Break Before Make, switching between input and output



9.2.4 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0, 0) and output high ({DDxn, PORTxn} = 1, 1), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0, 1) or output low ({DDxn, PORTxn} = 1, 0) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register or the PUDx bit in PORTCR Register can be set to disable all pull-ups in the port.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0, 0) or the output high state ({DDxn, PORTxn} = 1, 1) as an intermediate step.

Table 9-1 summarizes the control signals for the pin value.

Table 9-1. Port Pin Configurations

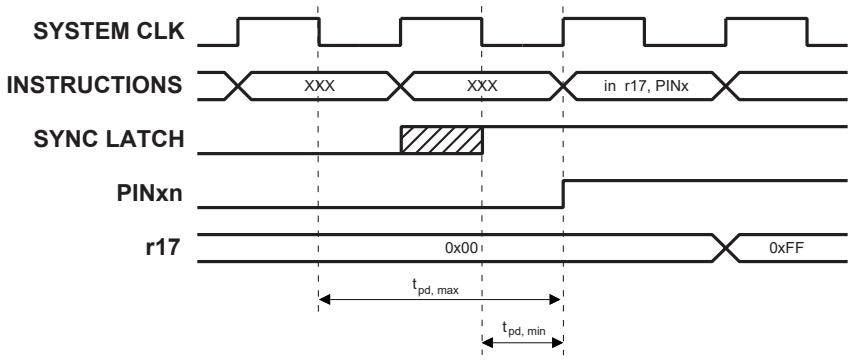
DDxn	PORTxn	PUD (in MCUCR) [1]	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Note: 1. Or port-wise PUDx bit in PORTCR register.

9.2.5 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 9-2, 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. Figure 9-4 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.

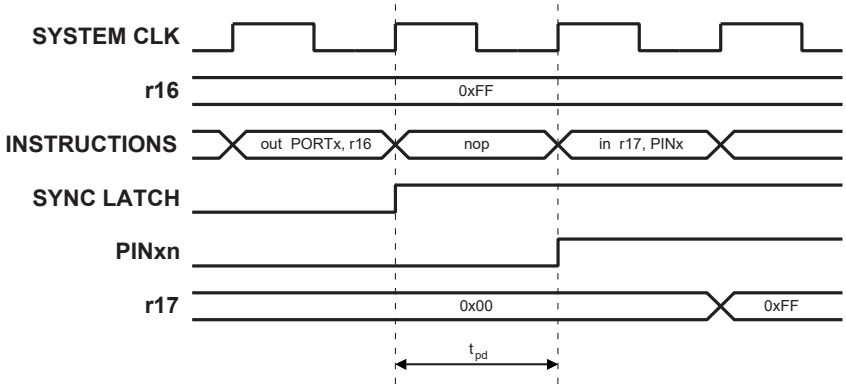
Figure 9-4. Synchronization when Reading an Externally Applied Pin value



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 Figure 9-5. 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 9-5. Synchronization when Reading a Software Assigned Pin Value



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

Assembly Code Example[1]

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

C Code Example

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

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

9.2.6 Digital Input Enable and Sleep Modes

As shown in Figure 9-2, 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 or Power-save 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” on page 80.

If a logic high level (“one”) 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.

9.2.7 Unconnected Pins

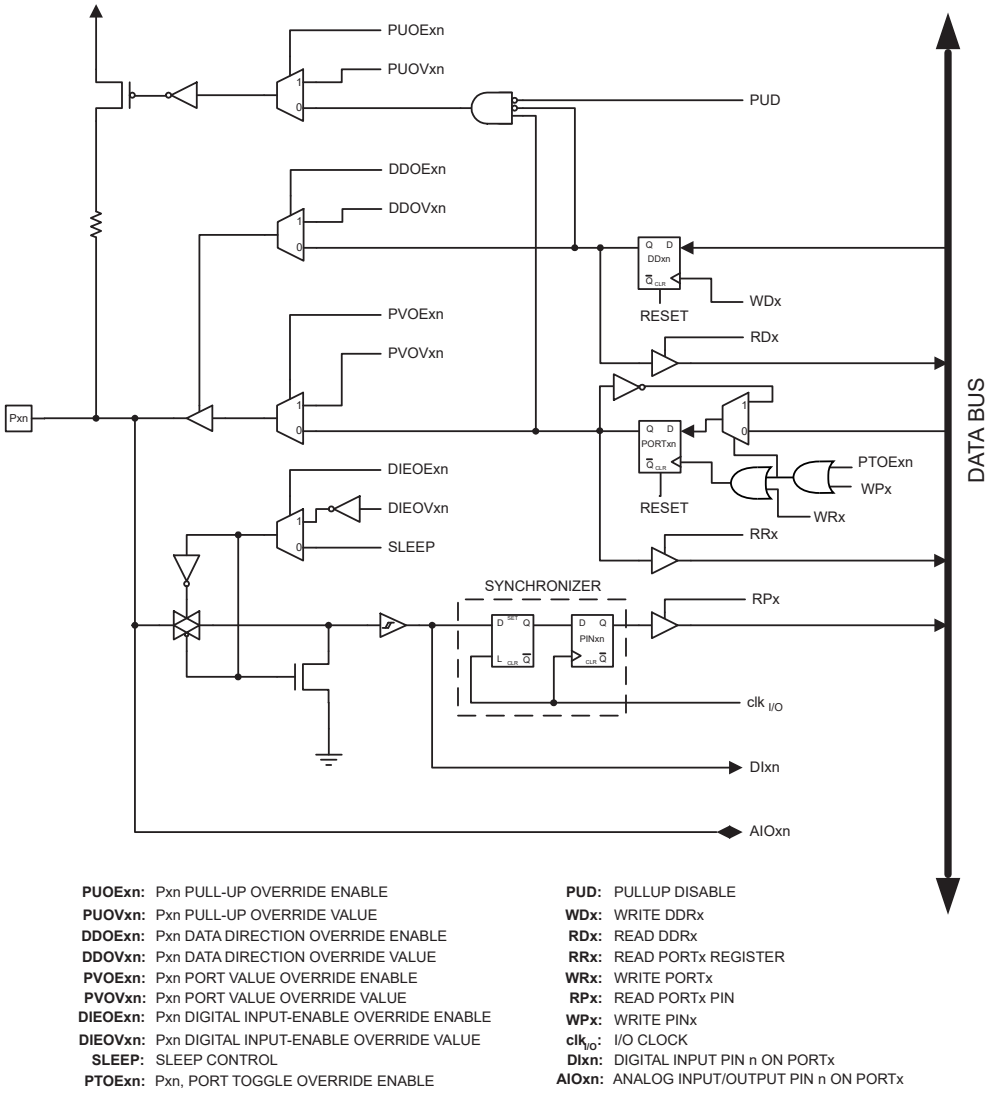
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-6 shows how the port pin control signals from the simplified Figure 9-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 9-6. Alternate Port Functions[1]



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

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

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

Table 9-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 9-6 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 9-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, (PUD or PDUx)} = 0, 1, 0.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, PUD and PUDx Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
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.

9.3.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	–	–	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0, 1). See “Configuring the Pin” on page 76 for more details about this feature.

9.3.2 PORTCR – Port Control Register

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	–	–	BBMB	BBMA	–	–	PUDB	PUDA	PORTCR
Read/Write	R	R	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 5:4 – BBMx: Break-Before-Make Mode Enable**

When these bits are written to one, the port-wise Break-Before-Make mode is activated. The intermediate tri-state cycle is then inserted when writing DDRxn to make an output. For further information, see “Break-Before-Make Switching” on page 77.

- **Bits 1:0 – PUDx: Port-Wise Pull-up Disable**

When these bits are written to one, the port-wise pull-ups in the defined I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0, 1). The Port-Wise Pull-up Disable bits are ORed with the global Pull-up Disable bit (PUD) from the MCUCR register. See “Configuring the Pin” on page 76 for more details about this feature.

9.3.3 Alternate Functions of Port A

The Port A pins with alternate functions are shown in Table 9-3.

Table 9-3. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	PCINT7 (Pin Change Interrupt 7) ADC7 (ADC Input Channel 7) AIN1 (Analog Comparator Positive Input) XREF (Internal Voltage Reference Output) AREF (External Voltage Reference Input)
PA6	PCINT6 (Pin Change Interrupt 6) ADC6 (ADC Input Channel 6) AIN0 (Analog Comparator Negative Input) $\overline{SS}$ (SPI Slave Select Input)
PA5	PCINT5 (Pin Change Interrupt 5) ADC5 (ADC Input Channel 5) T1 (Timer/Counter1 Clock Input) USCK (Three-wire Mode USI *Alternate* Clock Input) SCL (Two-wire Mode USI *Alternate* Clock Input) SCK (SPI Master Clock)
PA4	PCINT4 (Pin Change Interrupt 4) ADC4 (ADC Input Channel 4) ICP1 (Timer/Counter1 Input Capture Trigger) DI (Three-wire Mode USI *Alternate* Data Input) SDA (Two-wire Mode USI *Alternate* Data Input / Output) MOSI (SPI Master Output / Slave Input)
PA3	PCINT3 (Pin Change Interrupt 3) ADC3 (ADC Input Channel 3) ISRC (Current Source Pin) INT1 (External Interrupt1 Input)
PA2	PCINT2 (Pin Change Interrupt 2) ADC2 (ADC Input Channel 2) OC0A (Output Compare and PWM Output A for Timer/Counter0) DO (Three-wire Mode USI *Alternate* Data Output) MISO (SPI Master Input / Slave Output)
PA1	PCINT1 (Pin Change Interrupt 1) ADC1 (ADC Input Channel 1) TXD (UART Transmit Pin) TXLIN (LIN Transmit Pin)
PA0	PCINT0 (Pin Change Interrupt 0) ADC0 (ADC Input Channel 0) RXD (UART Receive Pin) RXLIN (LIN Receive Pin)

The alternate pin configuration is as follows:

- **Port A, Bit 7 – PCINT7/ADC7/AIN1/XREF/AREF**
  - PCINT7: Pin Change Interrupt, source 7.
  - ADC7: Analog to Digital Converter, channel 7.
  - AIN1: Analog Comparator Positive Input. This pin is directly connected to the positive input of the Analog Comparator.
  - XREF: Internal Voltage Reference Output. The internal voltage reference 2.56V or 1.1V is output when XREFEN is set and if either 2.56V or 1.1V is used as reference for ADC conversion. When XREF output is enabled, the pin port pull-up and digital output driver are turned off.
  - AREF: External Voltage Reference Input for ADC. The pin port pull-up and digital output driver are disabled when the pin is used as an external voltage reference input for ADC or as when the pin is only used to connect a bypass capacitor for the voltage reference of the ADC.

- **Port A, Bit 6 – PCINT6/ADC6/AIN0/$\overline{SS}$**
  - PCINT6: Pin Change Interrupt, source 6.
  - ADC6: Analog to Digital Converter, channel 6.
  - AIN0: Analog Comparator Negative Input. This pin is directly connected to the negative input of the Analog Comparator.
  - $\overline{SS}$: SPI Slave Select Input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDA6. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDA6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTA6 bit.

- **Port A, Bit 5 – PCINT5/ADC5/T1/USCK/SCL/SCK**
  - PCINT5: Pin Change Interrupt, source 5.
  - ADC5: Analog to Digital Converter, channel 5.
  - T1: Timer/Counter1 Clock Input.
  - USCK: Three-wire Mode USI Clock Input.
  - SCL: Two-wire Mode USI Clock Input.
  - SCK: SPI Master Clock output, Slave Clock input pin. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDA5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDA5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTA5 bit.

- **Port A, Bit 4 – PCINT4/ADC4/ICP1/DI/SDA/MOSI**
  - PCINT4: Pin Change Interrupt, source 4.
  - ADC4: Analog to Digital Converter, channel 4.
  - ICP1: Timer/Counter1 Input Capture Trigger. The PA3 pin can act as an Input Capture pin for Timer/Counter1.
  - DI: Three-wire Mode USI Data Input. USI Three-wire mode does not override normal port functions, so pin must be configure as an input for DI function.
  - SDA: Two-wire Mode Serial Interface (USI) Data Input / Output.

- MOSI: SPI Master Output / Slave Input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDA4. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDA4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTA4 bit.

- **Port A, Bit 3 – PCINT3/ADC3/ISRC/INT1**
  - PCINT3: Pin Change Interrupt, source 3.
  - ADC3: Analog to Digital Converter, channel 3.
  - ISCR: Current Source Output pin. While current is sourced by the Current Source module, the user can use the Analog to Digital Converter channel 3 (ADC3) to measure the pin voltage.
  - INT1: External Interrupt, source 1. The PA3 pin can serve as an external interrupt source.

- **Port A, Bit 2 – PCINT2/ADC2/OC0A/DO/MISO**
  - PCINT2: Pin Change Interrupt, source 2.
  - ADC2: Analog to Digital Converter, channel 2.
  - OC0A: Output Compare Match A or output PWM A for Timer/Counter0. The pin has to be configured as an output (DDA2 set (one)) to serve these functions.
  - DO: Three-wire Mode USI Data Output. Three-wire mode data output overrides PORTA2 and it is driven to the port when the data direction bit DDA2 is set. PORTA2 still enables the pull-up, if the direction is input and PORTA2 is set (one).
  - MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDA2. When the SPI is enabled as a Salve, the data direction of this pin is controlled by DDA2. When the pin is forced to be an input, the pull-up can still be controlled by PORTA2.

- **Port A, Bit 1 – PCINT1/ADC1/TXD/TXLIN**
  - PCINT1: Pin Change Interrupt, source 1.
  - ADC1: Analog to Digital Converter, channel 1.
  - TXD: UART Transmit pin. When the UART transmitter is enabled, this pin is configured as an output regardless the value of DDA1. PORTA1 still enables the pull-up, if the direction is input and PORTA2 is set (one).
  - TXLIN: LIN Transmit pin. When the LIN is enabled, this pin is configured as an output regardless the value of DDA1. PORTA1 still enables the pull-up, if the direction is input and PORTA2 is set (one).

- **Port A, Bit 0 – PCINT0/ADC0/RXD/RXLIN**
  - PCINT0: Pin Change Interrupt, source 0.
  - ADC0: Analog to Digital Converter, channel 0.
  - RXD: UART Receive pin. When the UART receiver is enabled, this pin is configured as an input regardless of the value of DDA0. When the pin is forced to be an input, a logical one in PORTA0 will turn on the internal pull-up.
  - RXLIN: LIN Receive pin. When the LIN is enabled, this pin is configured as an input regardless of the value of DDA0. When the pin is forced to be an input, a logical one in PORTA0 will turn on the internal pull-up.

Table 9-4 and Table 9-5 relate the alternate functions of Port A to the overriding signals shown in Figure 9-6 on page 81.

Table 9-4. Overriding Signals for Alternate Functions in PA[7:4]

Signal Name	PA7/PCINT7/ ADC7/AIN1/ XREF/AREF	PA6/PCINT6/ ADC6/AIN0/SS	PA5/PCINT5/ ADC5/T1/USCK/ SCL/SCK	PA4/PCINT4/ ADC4/ICP1/DI/ SDA/MOSI
PUOE	0	SPE **&** $\overline{\text{MSTR}}$	SPE **&** $\overline{\text{MSTR}}$	SPE **&** $\overline{\text{MSTR}}$
PUOV	0	PORTA6 **&** $\overline{\text{PUD}}$	PORTA5 **&** $\overline{\text{PUD}}$	PORTA4 **&** $\overline{\text{PUD}}$
DDOE	0	SPE **&** $\overline{\text{MSTR}}$	(SPE **&** $\overline{\text{MSTR}}$) **|** (USI_2_WIRE **&** USIPOS)	(SPE **&** $\overline{\text{MSTR}}$) **|** (USI_2_WIRE **&** USIPOS)
DDOV	0	0	(USI_SCL_HOLD **|** $\overline{\text{PORTA5}}$) **&** DDRA6	**{** (SPE **&** $\overline{\text{MSTR}}$) **?** (0) **:** ($\overline{\text{USI_SHIFTOUT}}$ **|** $\overline{\text{PORTA4}}$) **&** DDRA4) **}**
PVOE	0	0	(SPE **&** MSTR) **|** (USI_2_WIRE **&** USIPOS **&** DDRA5)	(SPE **&** MSTR) **|** (USI_2_WIRE **&** USIPOS **&** DDRA4)
PVOV	0	0	**{** (SPE **&** MSTR) **?** (SCK_OUTPUT) **:** **~** (USI_2_WIRE **&** USIPOS **&** DDRA5) **}**	**{** (SPE **&** MSTR) **?** (MOSI_OUTPUT) **:** **~** (USI_2_WIRE **&** USIPOS **&** DDRA4) **}**
PTOE	0	0	USI_PTOE **&** USIPOS	0
DIEOE	ADC7D **|** (PCIE0 **&** PCMSK07)	ADC6D **|** (PCIE0 **&** PCMSK06)	ADC5D **|** (USISIE **&** USIPOS) **|** (PCIE0 **&** PCMSK05)	ADC4D **|** (USISIE **&** USIPOS) **|** (PCIE0 **&** PCMSK04)
DIEOV	PCIE0 & PCMSK07	PCIE0 & PCMSK06	(USISIE & USIPOS) **|** (PCIE0 & PCMSK05)	(USISIE & USIPOS) **|** (PCIE0 & PCMSK04)
DI	PCINT7	PCINT6 **-/-** $\overline{\text{SS}}$	PCINT5 **-/-** T1 **-/-** USCK **-/-** SCL **-/-** SCK	PCINT4 **-/-** ICP1 **-/-** DI **-/-** SDA **-/-** MOSI
AIO	ADC7 **-/-** AIN1 **-/-** XREF **-/-** AREF	ADC6 **-/-** AIN0	ADC5	ADC4

Table 9-5. Overriding Signals for Alternate Functions in PA[3:0]

Signal Name	PA3/PCINT3/ADC3/ ISRC/INT1	PA2/PCINT2/ADC2/ OC0A/DO/MISO	PA1/PCINT1/ADC1/ TXD/TXLIN	PA0/PCINT0/ADC0/ RXD/RXLIN
PUOE	0	SPE **&** $\overline{\text{MSTR}}$	LIN_TX_ENABLE	LIN_RX_ENABLE
PUOV	PORTA3 **&** $\overline{\text{PUD}}$	PORTA2 **&** $\overline{\text{PUD}}$	**{** (LIN_TX_ENABLE) **?** (0) **:** (PORTA1 **&** $\overline{\text{PUD}}$) **}**	PORTA0 **&** $\overline{\text{PUD}}$
DDOE	0	SPE **&** $\overline{\text{MSTR}}$	LIN_TX_ENABLE	LIN_RX_ENABLE
DDOV	0	0	LIN_TX_ENABLE	0
PVOE	0	(SPE **&** MSTR) **|** (USI_2_WIRE **&** USI_3_WIRE **&** USIPOS) **|** OC0A	LIN_TX_ENABLE	0
PVOV	0	**{** (SPE **&** MSTR) **?** (MISO_OUTPUT) **:** (($\overline{\text{USI_2_WIRE}}$ **&** USI_3_WIRE **&** USIPOS) **?** (USI_SHIFTOUT) **:** (OC0A)) **}**	**{** (LIN_TX_ENABLE) **?** (LIN_TX) **:** (0) **}**	0
PTOE	0	0	0	0
DIEOE	ADC3D **|** INT1_ENABLE **|** (PCIE0 **&** PCMSK03)	ADC2D **|** (PCIE0 **&** PCMSK02)	ADC1D **|** (PCIE0 **&** PCMSK01)	ADC0D **|** (PCIE0 **&** PCMSK00)
DIEOV	INT1_ENABLE **|** (PCIE0 **&** PCMSK03)	PCIE0 **&** PCMSK02	PCIE0 **&** PCMSK01	PCIE0 **&** PCMSK00
DI	PCINT3 **-/-** INT1	PCINT2 **-/-** MISO	PCINT1	PCINT0
AIO	ADC3 **-/-** ISRC	ADC2	ADC1	ADC0

9.3.4 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 9-6.

Table 9-6. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	PCINT15 (Pin Change Interrupt 15) ADC10 (ADC Input Channel 10) OC1BX (Output Compare and PWM Output B-X for Timer/Counter1) $\overline{\text{RESET}}$ (Reset input pin) dW (debugWIRE I/O)
PB6	PCINT14 (Pin Change Interrupt 14) ADC9 (ADC Input Channel 9) OC1AX (Output Compare and PWM Output A-X for Timer/Counter1) INT0 (External Interrupt0 Input)
PB5	PCINT13 (Pin Change Interrupt 13) ADC8 (ADC Input Channel 8) OC1BW (Output Compare and PWM Output B-W for Timer/Counter1) XTAL2 (Chip clock Oscillator pin 2) CLKO (System clock output)
PB4	PCINT12 (Pin Change Interrupt 12) OC1AW (Output Compare and PWM Output A-W for Timer/Counter1) XTAL1 (Chip clock Oscillator pin 1) CLKI (External clock input)
PB3	PCINT11 (Pin Change Interrupt 11) OC1BV (Output Compare and PWM Output B-V for Timer/Counter1)
PB2	PCINT10 (Pin Change Interrupt 10) OC1AV (Output Compare and PWM Output A-V for Timer/Counter1) USCK (Three-wire Mode USI *Default* Clock Input) SCL (Two-wire Mode USI *Default* Clock Input)
PB1	PCINT9 (Pin Change Interrupt 9) OC1BU (Output Compare and PWM Output B-U for Timer/Counter1) DO (Three-wire Mode USI *Default* Data Output)
PB0	PCINT8 (Pin Change Interrupt 8) OC1AU (Output Compare and PWM Output A-U for Timer/Counter1) DI (Three-wire Mode USI *Default* Data Input) SDA (Two-wire Mode USI *Default* Data Input / Output)

The alternate pin configuration is as follows:

- **Port B, Bit 7 – PCINT15/ADC10/OC1BX/$\overline{\text{RESET}}$/dW**
  - PCINT15: Pin Change Interrupt, source 15.
  - ADC10: Analog to Digital Converter, channel 10.
  - OC1BX: Output Compare and PWM Output B-X for Timer/Counter1. The PB7 pin has to be configured as an output (DDB7 set (one)) to serve this function. The OC1BX pin is also the output pin for the PWM mode timer function (c.f. OC1BX bit of TCCR1D register).

- $\overline{\text{RESET}}$: Reset input pin. When the RSTDISBL Fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on Power-on Reset and Brown-out Reset as its reset sources. When the RSTDISBL Fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.
  If PB7 is used as a reset pin, DDB7, PORTB7 and PINB7 will all read 0.
- dW: When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

- **Port B, Bit 6 – PCINT14/ADC9/OC1AX/INT0**
  - PCINT14: Pin Change Interrupt, source 14.
  - ADC9: Analog to Digital Converter, channel 9.
  - OC1AX: Output Compare and PWM Output A-X for Timer/Counter1. The PB6 pin has to be configured as an output (DDB6 set (one)) to serve this function. The OC1AX pin is also the output pin for the PWM mode timer function (c.f. OC1AX bit of TCCR1D register).
  - INT0: External Interrupt0 Input. The PB6 pin can serve as an external interrupt source.

- **Port B, Bit 5 – PCINT13/ADC8/OC1BW/XTAL2/CLKO**
  - PCINT13: Pin Change Interrupt, source 13.
  - ADC8: Analog to Digital Converter, channel 8.
  - OC1BW: Output Compare and PWM Output B-W for Timer/Counter1. The PB5 pin has to be configured as an output (DDB5 set (one)) to serve this function. The OC1BW pin is also the output pin for the PWM mode timer function (c.f. OC1BW bit of TCCR1D register).
  - XTAL2: Chip clock Oscillator pin 2. Used as clock pin for crystal Oscillator or Low-frequency crystal Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.
  - CLKO: Divided system clock output. The divided system clock can be output on the PB5 pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTB5 and DDB5 settings. It will also be output during reset.

- **Port B, Bit 4 – PCINT12/OC1AW/XTAL1/CLKI**
  - PCINT12: Pin Change Interrupt, source 12.
  - OC1AW: Output Compare and PWM Output A-W for Timer/Counter1. The PB4 pin has to be configured as an output (DDB4 set (one)) to serve this function. The OC1AW pin is also the output pin for the PWM mode timer function (c.f. OC1AW bit of TCCR1D register).
  - XTAL1: Chip clock Oscillator pin 1. Used for all chip clock sources except internal calibrated RC Oscillator. When used as a clock pin, the pin can not be used as an I/O pin.
  - CLKI: External clock input. When used as a clock pin, the pin can not be used as an I/O pin.

  Note: If PB4 is used as a clock pin (XTAL1 or CLKI), DDB4, PORTB4 and PINB4 will all read 0.

- **Port B, Bit 3 – PCINT11/OC1BV**
  - PCINT11: Pin Change Interrupt, source 11.
  - OC1BV: Output Compare and PWM Output B-V for Timer/Counter1. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC1BV pin is also the output pin for the PWM mode timer function (c.f. OC1BV bit of TCCR1D register).

- **Port B, Bit 2 – PCINT10/OC1AV/USCK/SCL**

- PCINT10: Pin Change Interrupt, source 10.
- OC1AV: Output Compare and PWM Output A-V for Timer/Counter1. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC1AV pin is also the output pin for the PWM mode timer function (c.f. OC1AV bit of TCCR1D register).
- USCK: Three-wire Mode USI Clock Input.
- SCL: Two-wire Mode USI Clock Input.

- **Port B, Bit 1 – PCINT9/OC1BU/DO**
  - PCINT9: Pin Change Interrupt, source 9.
  - OC1BU: Output Compare and PWM Output B-U for Timer/Counter1. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC1BU pin is also the output pin for the PWM mode timer function (c.f. OC1BU bit of TCCR1D register).
  - DO: Three-wire Mode USI Data Output. Three-wire mode data output overrides PORTB1 and it is driven to the port when the data direction bit DDB1 is set. PORTB1 still enables the pull-up, if the direction is input and PORTB1 is set (one).

- **Port B, Bit 0 – PCINT8/OC1AU/DI/SDA**
  - IPCINT8: Pin Change Interrupt, source 8.
  - OC1AU: Output Compare and PWM Output A-U for Timer/Counter1. The PB0 pin has to be configured as an output (DDB0 set (one)) to serve this function. The OC1AU pin is also the output pin for the PWM mode timer function (c.f. OC1AU bit of TCCR1D register).
  - DI: Three-wire Mode USI Data Input. USI Three-wire mode does not override normal port functions, so pin must be configure as an input for DI function.
  - SDA: Two-wire Mode Serial Interface (USI) Data Input / Output.

Table 9-7 and Table 9-8 relate the alternate functions of Port B to the overriding signals shown in Figure 9-6 on page 81.

Table 9-7. Overriding Signals for Alternate Functions in PB[7:4]

Signal Name	PB7/PCINT15/ ADC10/OC1BX/ RESET/dW	PB6/PCINT14/ ADC9/OC1AX/INT0	PB5/PCINT13/ ADC8/OC1BW/ XTAL2/CLKO	PB4/PCINT12/ OC1AW/XTAL1/CLKI
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC1B_ENABLE **&** OC1BX	OC1A_ENABLE **&** OC1AX	OC1B_ENABLE **&** OC1BW	OC1A_ENABLE **&** OC1AW
PVOV	OC1B	OC1A	OC1B	OC1A
PTOE	0	0	0	0
DIEOE	ADC10D **|** (PCIE1 **&** PCMSK15)	ADC9D **|** INT0_ENABLE **|** (PCIE1 **&** PCMSK14)	ADC8D **|** (PCIE1 **&** PCMSK13)	(PCIE1 **&** PCMSK13)
DIEOV	PCIE1 **&** PCMSK15	INT0_ENABLE **|** (PCIE1 **&** PCMSK14)	PCIE1 **&** PCMSK13	1
DI	PCINT15	PCINT14 **-/-** INT1	PCINT13	PCINT12
AIO	RESET **-/-** ADC10 **-/-**	ADC9 **-/-** ISRC	ADC8 **-/-** XTAL2	XTAL1 **-/-** CLKI

Table 9-8. Overriding Signals for Alternate Functions in PB[3:0]

Signal Name	PB3/PCINT11/ OC1BV	PB2/PCINT10/ OC1AV/USCK/SCL	PB1/PCINT9/ OC1BU/DO	PB0/IPCINT8/ OC1AU/DI/SDA
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$)	0	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$)
DDOV	0	(USI_SCL_HOLD **|** $\overline{\text{PORTB2}}$) **&** DDRB2	0	($\overline{\text{USI_SHIFTOUT}}$ **|** $\overline{\text{PORTB0}}$) **&** DDRB0)
PVOE	OC1B_ENABLE **&** OC1BV	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB2) **|** (OC1A_ENABLE **&** OC1AV)	($\overline{\text{USI_2_WIRE}}$ **&** USI_3_WIRE **&** $\overline{\text{USIPOS}}$) **|** (OC1B_ENABLE **&** OC1BU)	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB0) **|** (OC1A_ENABLE **&** OC1AU)
PVOV	OC1B	**{** (USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB2) **?** (0) **:** (OC1A) **}**	**{** ($\overline{\text{USI_2_WIRE}}$ **&** USI_3_WIRE **&** $\overline{\text{USIPOS}}$) **?** (USI_SHIFTOUT) **:** (OC1B) **}**	**{** (USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB0) **?** (0) **:** (OC1A) **}**
PTOE	0	USI_PTOE **&** $\overline{\text{USIPOS}}$	0	0
DIEOE	PCIE1 **&** PCMSK11	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK10)	PCIE1 **&** PCMSK9	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK8)
DIEOV	1	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK10)	1	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK8)
DI	PCINT11	PCINT10 **-/-** USCK **-/-** SCL	PCINT9	PCINT8 **-/-** DI **-/-** SDA
AIO	0	0	0	0

9.4 Register Description

9.4.1 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

9.4.2 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

9.4.3 PINA – Port A Input Pins Register

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

9.4.4 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

9.4.5 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

9.4.6 PINB – Port B Input Pins Register

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

10. 8-bit Timer/Counter0 and Asynchronous Operation

Timer/Counter0 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

10.1 Features

- **Single Channel Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV0 and OCF0A)**
- **Allows Clocking from External Crystal (i.e. 32 kHz Watch Crystal) Independent of the I/O Clock**

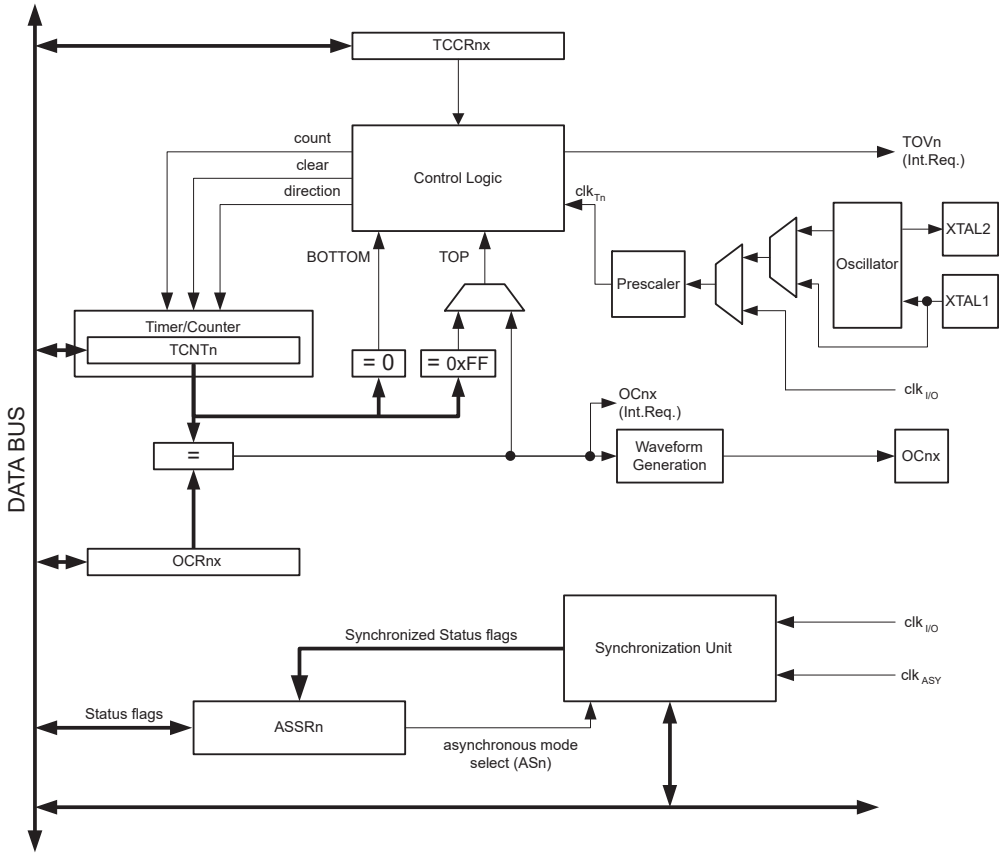
10.2 Overview

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

- A lower case “n” replaces the Timer/Counter number, in this case 0. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.
- A lower case “x” replaces the Output Compare unit channel, in this case A. However, when using the register or bit defines in a program, the precise form must be used, i.e., OCR0A for accessing Timer/Counter0 output compare channel A value and so on.

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 10-1. For the actual placement of I/O pins, refer to “Pin Configuration” on page 11. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “Register Description” on page 107.

Figure 10-1. 8-bit Timer/Counter0 Block Diagram



The Timer/Counter (TCNT0) and Output Compare Register (OCR0A) are 8-bit registers. Interrupt request (shorten as Int.Req.) 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 asynchronously clocked from the XTAL1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source 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 Register (OCR0A) is compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC0A). See ”Output Compare Unit” on page 97. for details. The compare match event will also set the compare flag (OCF0A) which can be used to generate an Output Compare interrupt request.

10.2.1 Definitions

The following definitions are used extensively throughout the section:

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

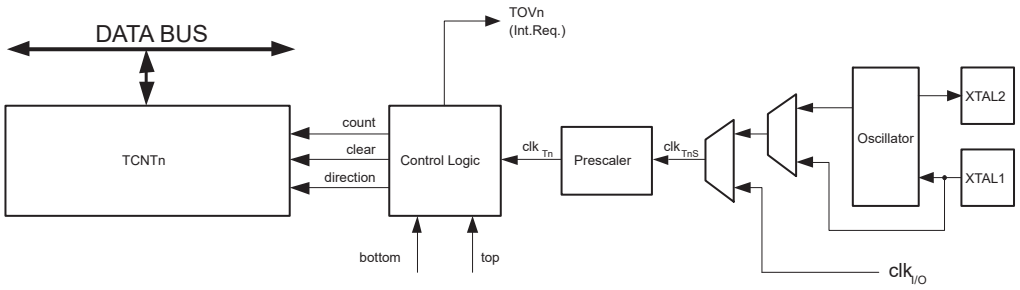
10.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source is selected by the clock select logic which is controlled by the clock select (CS0[2:0]) bits located in the Timer/Counter control register (TCCR0).The clock source clk_T0 is by default equal to the MCU clock, $clk_{I/O}$. When the AS0 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to XTAL1 and XTAL2 or directly from XTAL1. For details on asynchronous operation, see “ASSR – Asynchronous Status Register” on page 110. For details on clock sources and prescaler, see “Timer/Counter0 Prescaler” on page 107.

10.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 10-2 shows a block diagram of the counter and its surrounding environment.

Figure 10-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_T0	Timer/Counter0 clock.
top	Signalizes that TCNT0 has reached maximum value.
bottom	Signalizes that TCNT0 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_T0). clk_T0 can be generated from an external or internal clock source, selected by the Clock Select bits (CS0[2:0]). When no clock source is selected (CS0[2:0] = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_T0 is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC0A. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 99.

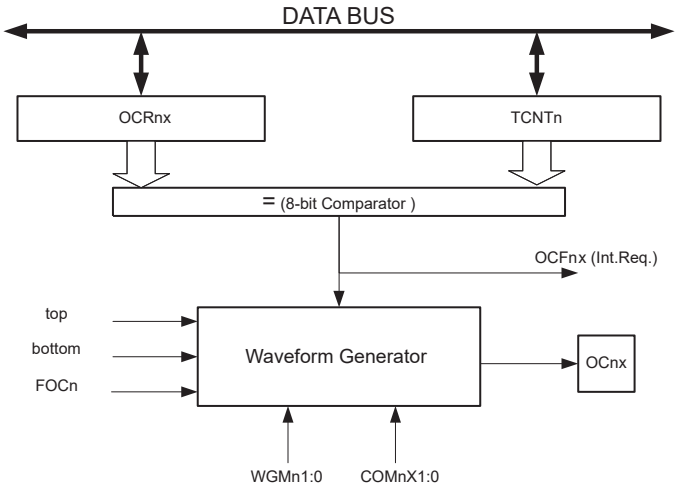
The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM0[1:0] bits. TOV0 can be used for generating a CPU interrupt.

10.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0A). Whenever TCNT0 equals OCR0A, the comparator signals a match. A match will set the Output Compare Flag (OCF0A) at the next timer clock cycle. If enabled (OCIE0A = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF0A flag is automatically cleared when the interrupt is executed. Alternatively, the OCF0A 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 WGM0[1:0] bits and Compare Output mode (COM0A[1:0]) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (“Modes of Operation” on page 99).

Figure 10-3 shows a block diagram of the Output Compare unit.

Figure 10-3. Output Compare Unit, Block Diagram



The OCR0A Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0A Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0A Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0A Buffer Register, and if double buffering is disabled the CPU will access the OCR0A directly.

10.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0A) bit. Forcing compare match will not set the OCF0A flag or reload/clear the timer, but the OC0A pin will be updated as if a real compare match had occurred (the COM0A[1:0] bits settings define whether the OC0A pin is set, cleared or toggled).

10.5.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0A to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

10.5.3 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 channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0A 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 downcounting.

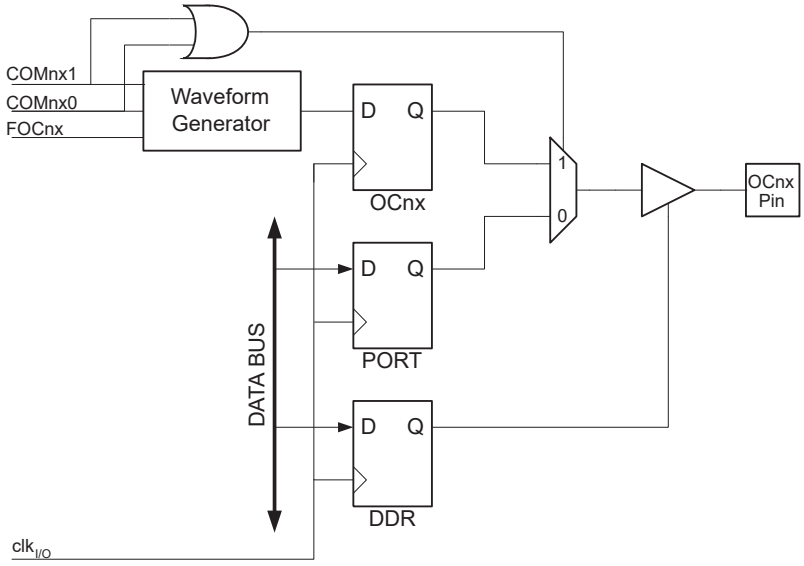
The setup of the OC0A should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0A value is to use the Force Output Compare (FOC0A) strobe bit in Normal mode. The OC0A Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM0A[1:0] bits are not double buffered together with the compare value. Changing the COM0A[1:0] bits will take effect immediately.

10.6 Compare Match Output Unit

The Compare Output mode (COM0A[1:0]) bits have two functions. The Waveform Generator uses the COM0A[1:0] bits for defining the Output Compare (OC0A) state at the next compare match. Also, the COM0A[1:0] bits control the OC0A pin output source. Figure 10-4 shows a simplified schematic of the logic affected by the COM0A[1:0] bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0A[1:0] bits are shown. When referring to the OC0A state, the reference is for the internal OC0A Register, not the OC0A pin.

Figure 10-4. Compare Match Output Logic



10.6.1 Compare Output Function

The general I/O port function is overridden by the Output Compare (OC0A) from the Waveform Generator if either of the COM0A[1:0] bits are set. However, the OC0A pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC0A pin (DDR_OC0A) must be set as output before the OC0A 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 OC0A state before the output is enabled. Note that some COM0A[1:0] bit settings are reserved for certain modes of operation. See ”Register Description” on page 107.

10.6.2 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0A[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0A[1:0] = 0 tells the Waveform Generator that no action on the OC0A Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 10-1 on page 108. For fast PWM mode, refer to Table 10-2 on page 108, and for phase correct PWM refer to Table 10-3 on page 108.

A change of the COM0A[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 FOC0A strobe bits.

10.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM0[1:0]) and Compare Output mode (COM0A[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0A[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0A[1:0] bits control whether the output should be set, cleared, or toggled at a compare match (See ”Compare Match Output Unit” on page 98.).

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 104.

10.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM0[1:0] = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

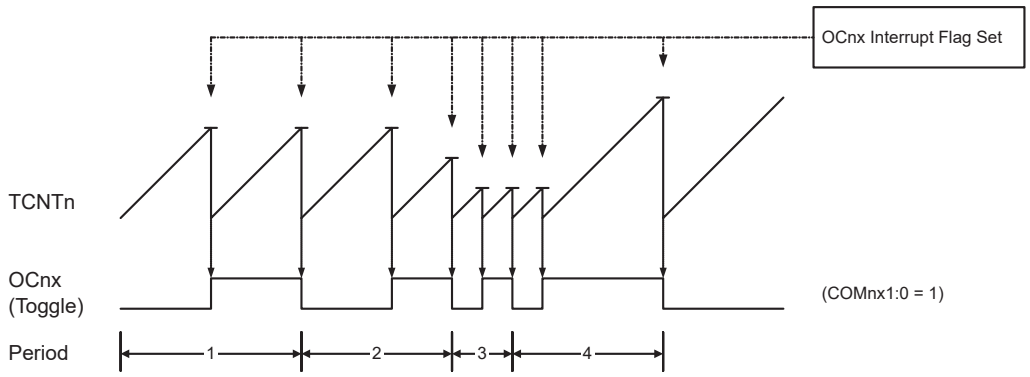
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

10.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM0[1:0] = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 10-5. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 10-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the 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] = 1). The OC0A value will not be visible on the port pin unless the data direction

for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC}0_A$ = $f_{clk_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

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

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

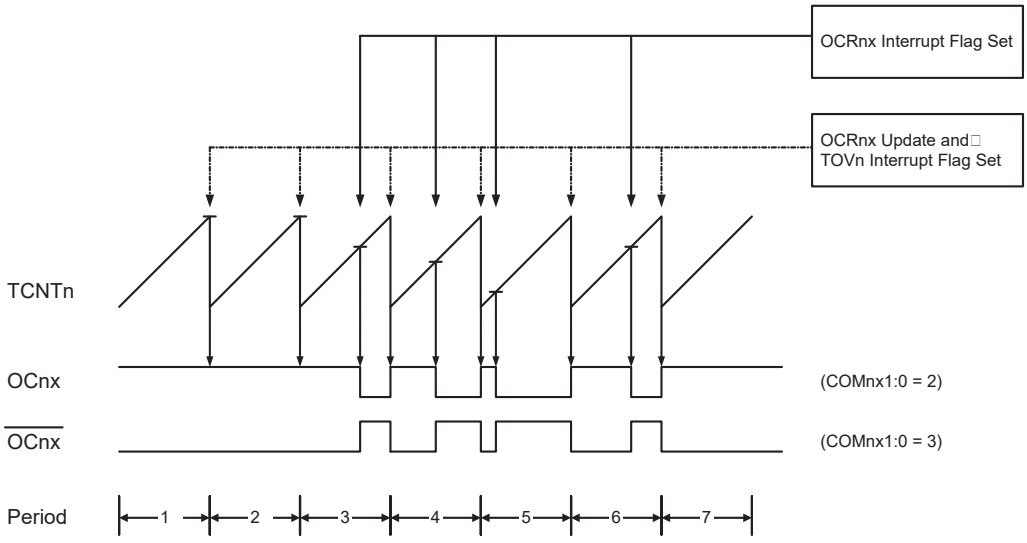
As for the Normal mode of operation, the TOV0 flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

10.7.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM0[1:0] = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0A) is cleared on the compare match between TCNT0 and OCR0A, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 10-6. 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 line marks on the TCNT0 slopes represent compare matches between OCR0A and TCNT0.

Figure 10-6. Fast PWM Mode, Timing Diagram



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

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pin. Setting the COM0A[1:0] bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0A[1:0] to three (See Table 10-2 on page 108). The actual OC0A value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0A Register at the compare match between OCR0A and TCNT0, and clearing (or setting) the OC0A Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

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

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot 256}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

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

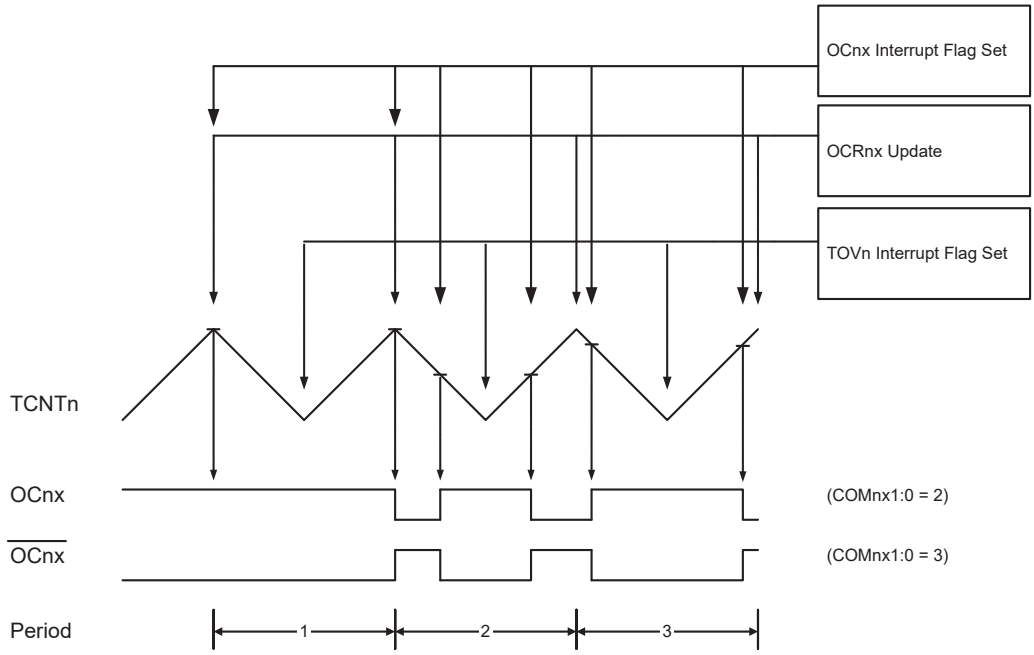
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0A to toggle its logical level on each compare match (COM0A[1:0] = 1). The waveform generated will have a maximum frequency of $f_{oc}0_A = f_{clk_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

10.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM0[1:0] = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0A) is cleared on the compare match between TCNT0 and OCR0A while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT0 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 10-7. 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 OCR0A and TCNT0.

Figure 10-7. Phase Correct PWM Mode, Timing Diagram



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

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pin. Setting the COM0A[1:0] bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0A[1:0] to three (See Table 10-3 on page 108). The actual OC0A value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0A Register at the compare match between OCR0A and TCNT0 when the counter increments, and setting (or clearing) the OC0A Register at compare match between OCR0A and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

10.8 Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_T0) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when interrupt flags are set. Figure 10-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 10-8. Timer/Counter Timing Diagram, no Prescaling

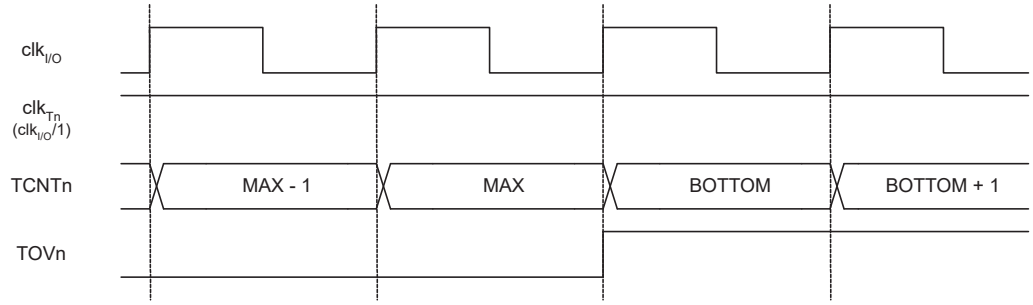


Figure 10-9 shows the same timing data, but with the prescaler enabled.

Figure 10-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)

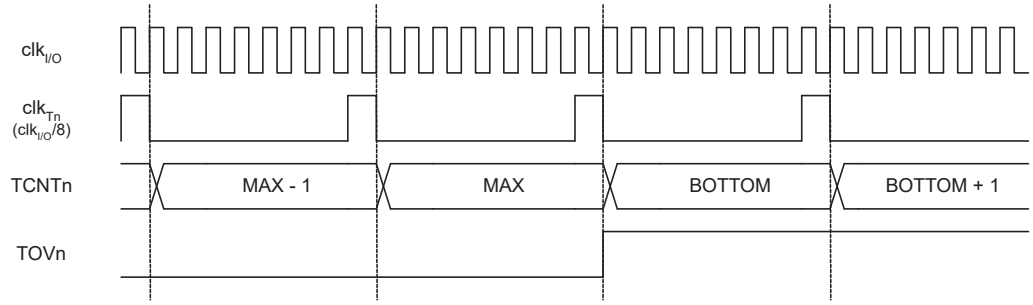


Figure 10-10 shows the setting of OCF0A in all modes except CTC mode.

Figure 10-10. Timer/Counter Timing Diagram, Setting of OCF0A, with Prescaler ($f_{clk_I/O}/8$)

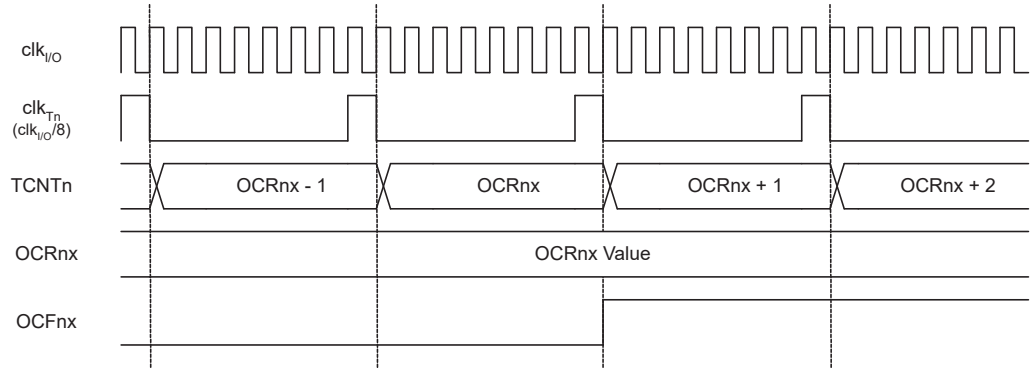
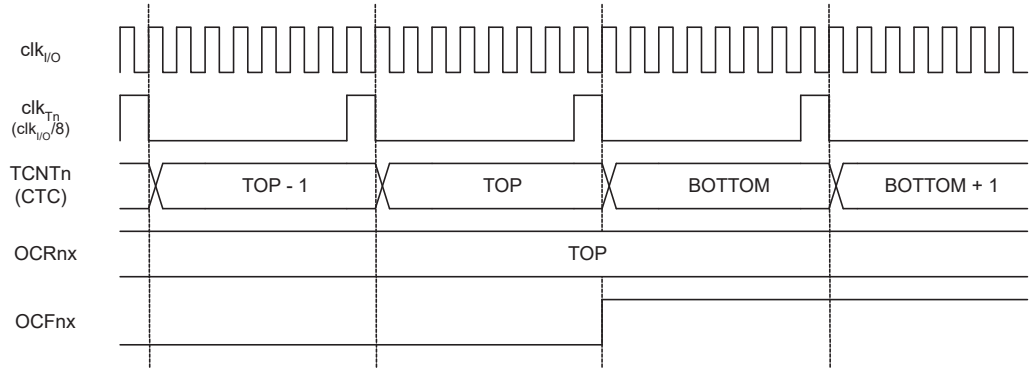


Figure 10-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode.

Figure 10-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk_I/O}/8$)



10.9 Asynchronous Operation of Timer/Counter0

When Timer/Counter0 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter0, the timer registers TCNT0, OCR0A, and TCCR0A might be corrupted. A safe procedure for switching clock source is:
  a. Disable the Timer/Counter0 interrupts by clearing OCIE0A and TOIE0.
  b. Select clock source by setting AS0 and EXCLK as appropriate.
  c. Write new values to TCNT0, OCR0A, and TCCR0A.
  d. To switch to asynchronous operation: Wait for TCN0UB, OCR0UB, and TCR0UB.
  e. Clear the Timer/Counter0 interrupt flags.
  f. Enable interrupts, if needed.
- If an 32.768 kHz watch crystal is used, the CPU main clock frequency must be more than four times the Oscillator or external clock frequency.
- When writing to one of the registers TCNT0, OCR0A, or TCCR0A, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that e.g. writing to TCNT0 does not disturb an OCR0A write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save mode after having written to TCNT0, OCR0A, or TCCR0A, the user must wait until the written register has been updated if Timer/Counter0 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare0 interrupt is used to wake up the device, since the Output Compare function is disabled during writing to OCR0A or TCNT0. If the write cycle is not finished, and the MCU enters sleep mode before the OCR0UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter0 is used to wake the device up from Power-save mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less

than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether the time before re-entering Power-save mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:

a. Write a value to TCCR0A, TCNT0, or OCR0A.
b. Wait until the corresponding Update Busy flag in ASSR returns to zero.
c. Enter Power-save or ADC Noise Reduction mode.

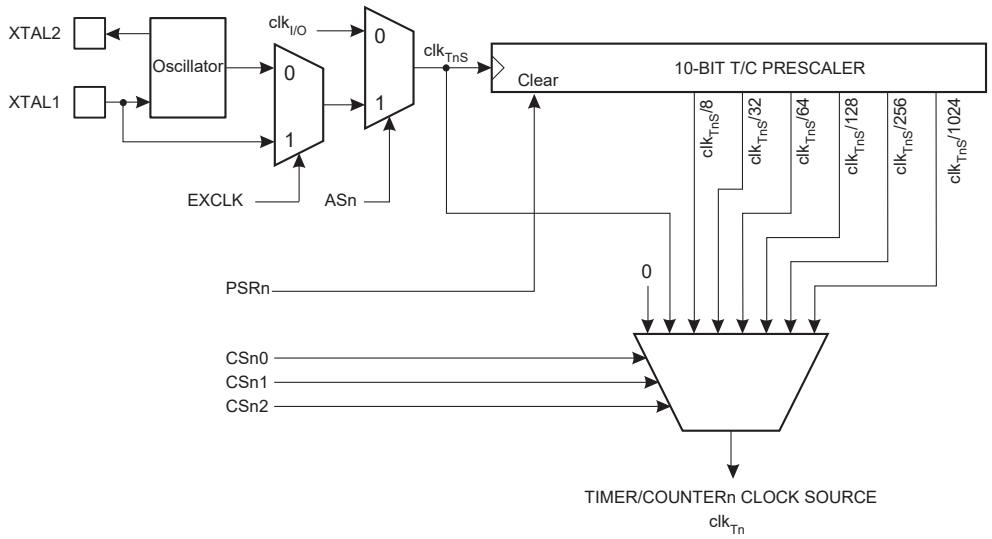
- When the asynchronous operation is selected, the oscillator for Timer/Counter0 is always running, except in Power-down mode. After a Power-up Reset or wake-up from Power-down mode, the user should be aware of the fact that this oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter0 after power-up or wake-up from Power-down mode. The contents of all Timer/Counter0 Registers must be considered lost after a wake-up from Power-down mode due to unstable clock signal upon start-up, no matter whether the oscillator is in use or a clock signal is applied to the XTAL1 pin.
- Description of wake up from Power-save mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the TCNT0 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT0 is clocked on the asynchronous clock, reading TCNT0 must be done through a register synchronized to the internal I/O clock domain (CPU main clock). Synchronization takes place for every rising XTAL1 edge. When waking up from Power-save mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT0 will read as the previous value (before entering sleep) until the next rising XTAL1 edge. The phase of the XTAL1 clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT0 is thus as follows:

  a. Write any value to either of the registers OCR0A or TCCR0A.
  b. Wait for the corresponding Update Busy Flag to be cleared.
  c. Read TCNT0.

- During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timer takes 3 processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the interrupt flag. The Output Compare pin is changed on the timer clock and is not synchronized to the processor clock.

10.10 Timer/Counter0 Prescaler

Figure 10-12. Prescaler for Timer/Counter0



The clock source for Timer/Counter0 is named clk_T0_S. clk_T0_S is by default connected to the main system I/O clock clk_{IO}. By setting the AS0 bit in ASSR, Timer/Counter0 is asynchronously clocked from the XTAL oscillator or XTAL1 pin. This enables use of Timer/Counter0 as a Real Time Counter (RTC).

A crystal can then be connected between the XTAL1 and XTAL2 pins to serve as an independent clock source for Timer/Counter0.

A external clock can also be used using XTAL1 as input. Setting AS0 and EXCLK enables this configuration.

For Timer/Counter0, the possible prescaled selections are: $clk_T0_S/8$, $clk_T0_S/32$, $clk_T0_S/64$, $clk_T0_S/128$, $clk_T0_S/256$, and $clk_T0_S/1024$. Additionally, clk_T0_S as well as 0 (stop) may be selected. Setting the PSR0 bit in GTCCR resets the prescaler. This allows the user to operate with a predictable prescaler.

10.11 Register Description

10.11.1 TCCR0A – Timer/Counter0 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	**COM0A1**	**COM0A0**	**–**	**–**	**–**	**–**	**WGM01**	**WGM00**	**TCCR0A**
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM0A[1:0]: Compare Match Output Mode A**

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

When OC0A is connected to the pin, the function of the COM0A[1:0] bits depends on the WGM0[1:0] bit setting. Table 10-1 shows the COM0A[1:0] bit functionality when the WGM0[1:0] bits are set to a normal or CTC mode (non-PWM).

Table 10-1. Compare Output Mode, non-PWM Mode

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

Table 10-2 shows the COM0A[1:0] bit functionality when the WGM0[1:0] bits are set to fast PWM mode.

Table 10-2. Compare Output Mode, Fast PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	
1	0	Clear OC0A on Compare Match. Set OC0A at BOTTOM (non-inverting mode).
1	1	Set OC0A on Compare Match. Clear OC0A at BOTTOM (inverting mode).

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 101 for more details.

Table 10-3 shows the COM0[1:0] bit functionality when the WGM0[1:0] bits are set to phase correct PWM mode.

Table 10-3. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Phase Correct PWM Mode” on page 102 for more details.

- **Bits 5:2 – Res: Reserved Bits**

These bits are reserved in the ATtiny87/167 and will always read as zero.

- **Bits 1:0 – WGM0[1:0]: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used, see Table 10-4. Modes of operation supported by the Timer/Counter unit are: Normal mode (Counter), Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (See "Modes of Operation" on page 99.).

Table 10-4. Waveform Generation Mode Bit Description

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0A at	TOV0 Flag Set on(1)(2)
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0A	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

Notes: 1. MAX = 0xFF,
2. BOTTOM = 0x00.

10.11.2 TCCR0B – Timer/Counter0 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	FOC0A	–	–	–	–	CS02	CS01	CS00	TCCR0B
Read/Write	W	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

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

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

The FOC0A bit is always read as zero.

- **Bits 6:3 – Res: Reserved Bits**

These bits are reserved in the ATtiny87/167 and will always read as zero.

- **Bits 2:0 – CS0[2:0]: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Table 10-5.

Table 10-5. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_T0_S (No prescaling)
0	1	0	clk_T0_S/8 (From prescaler)
0	1	1	clk_T0_S/32 (From prescaler)
1	0	0	clk_T0_S/64 (From prescaler)
1	0	1	clk_T0_S/128 (From prescaler)
1	1	0	clk_T0_S/256 (From prescaler)
1	1	1	clk_T0_S/1024 (From prescaler)

10.11.3 TCNT0 – Timer/Counter0 Register

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Register.

10.11.4 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	OCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

10.11.5 ASSR – Asynchronous Status Register

Bit	7	6	5	4	3	2	1	0	
(0xB6)	–	EXCLK	AS0	TCN0UB	OCR0AUB	–	TCR0AUB	TCR0BUB	ASSR
Read/Write	R	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is reserved in the ATtiny87/167 and will always read as zero.

- **Bit 6 – EXCLK: Enable External Clock Input**

When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on XTAL1 pin instead of an external crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal oscillator will only run when this bit is zero.

- **Bit 5 – AS0: Asynchronous Timer/Counter0**

When AS0 is written to zero, Timer/Counter0 is clocked from the I/O clock, clkI/O and the Timer/Counter0 acts as a synchronous peripheral.
When AS0 is written to one, Timer/Counter0 is clocked from the low-frequency crystal oscillator (See ”Low-frequency Crystal Oscillator” on page 37.) or from external clock on XTAL1 pin (See ”External Clock” on page 38.) depending on EXCLK setting. When the value of AS0 is changed, the contents of TCNT0, OCR0A, and TCCR0A might be corrupted.

AS0 also acts as a flag: Timer/Counter0 is clocked from the low-frequency crystal or from external clock ONLY IF the calibrated internal RC oscillator or the internal watchdog oscillator is used to drive the system clock. After setting AS0, if the switching is available, AS0 remains to 1, else it is forced to 0.

- **Bit 4 – TCN0UB: Timer/Counter0 Update Busy**

When Timer/Counter0 operates asynchronously and TCNT0 is written, this bit becomes set. When TCNT0 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT0 is ready to be updated with a new value.

- **Bit 3 – OCR0AUB: Output Compare 0 Register A Update Busy**

When Timer/Counter0 operates asynchronously and OCR0A is written, this bit becomes set. When OCR0A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR0A is ready to be updated with a new value.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved in the ATtiny87/167 and will always read as zero.

- **Bit 1 – TCR0AUB: Timer/Counter0 Control Register A Update Busy**

When Timer/Counter0 operates asynchronously and TCCR0A is written, this bit becomes set. When TCCR0A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR0A is ready to be updated with a new value.

- **Bit 0 – TCR0BUB: Timer/Counter0 Control Register B Update Busy**

When Timer/Counter0 operates asynchronously and TCCR0B is written, this bit becomes set. When TCCR0B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR0B is ready to be updated with a new value.

If a write is performed to any of the four Timer/Counter0 Registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT0, OCR0A, TCCR0A and TCCR0B are different. When reading TCNT0, the actual timer value is read. When reading OCR0A, TCCR0A or TCCR0B the value in the temporary storage register is read.

10.11.6 TIMSK0 – Timer/Counter0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	–	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved in the ATtiny87/167 and will always read as zero.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter0 Interrupt Flag Register – TIFR0.

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

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

10.11.7 TIFR0 – Timer/Counter0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	–	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved in the ATtiny87/167 and will always read as zero.

- **Bit 1 – OCF0A: Output Compare Flag 0 A**

The OCF0A bit is set (one) when a compare match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0 (Timer/Counter0 Compare match Interrupt Enable), and OCF0A are set (one), the Timer/Counter0 Compare match Interrupt is executed.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The TOV0 bit is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0A (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter0 changes counting direction at 0x00.

10.11.8 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	–	–	–	–	–	PSR0	PSR1	GTCCR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – PSR0: Prescaler Reset Timer/Counter0**

When this bit is one, the Timer/Counter0 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter0 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set. Refer to the description of the “Bit 7 – TSM: Timer/Counter Synchronization Mode” on page 116 for a description of the Timer/Counter Synchronization mode.

11. Timer/Counter1 Prescaler

11.1 Overview

Most bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number.

11.1.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn[2:0] = 1). 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 $f_{CLK_I/O}/1024$.

11.1.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter. 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 (6 > CSn[2:0] > 1). 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).

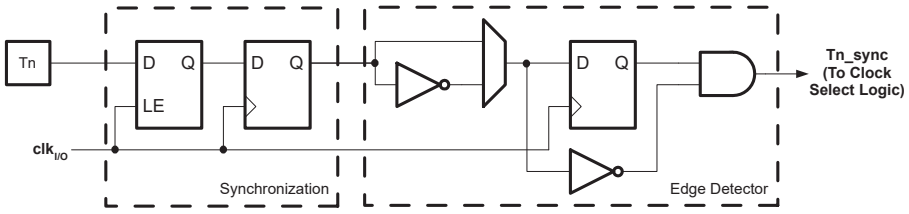
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

11.1.3 External Clock Source

An external clock source applied to the T1 pin can be used as Timer/Counter clock (clk_T1). The T1 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 11-1 shows a functional equivalent block diagram of the T1 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T1} pulse for each positive (CSn[2:0] = 7) or negative (CSn[2:0] = 6) edge it detects.

Figure 11-1. T1 Pin Sampling



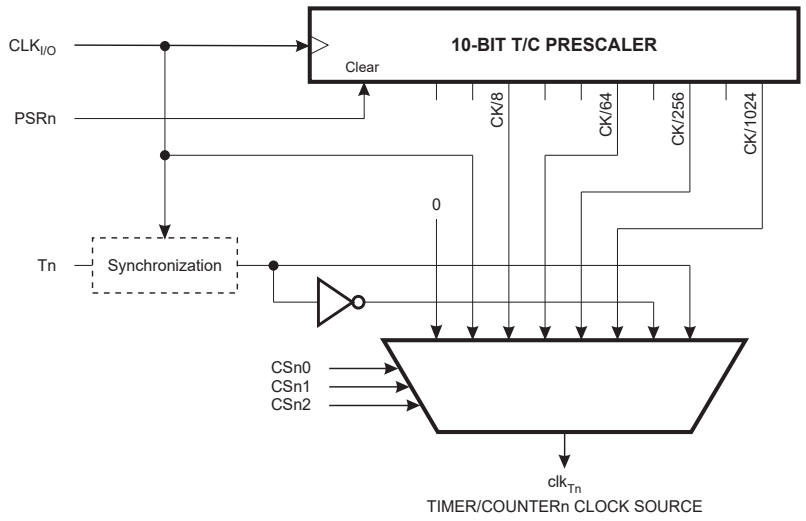
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1 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 ensured to have less than half the system clock frequency ($f_{ExtClk} < f_{clk_I/O}/2$) given a 50/50 % 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 Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk_I/O}/2.5$.

An external clock source can not be prescaled.

Figure 11-2. Prescaler for Timer/Counter1 (1)



Note: 1. The synchronization logic on the input pin (T1) is shown in Figure 11-1.

11.2 Register Description

11.2.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	–	–	–	–	–	PSR0	PSR1	GTCCR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR0 and PSR1 bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSR0 and PSR1 bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

- **Bit 0 – PSR1: Prescaler Reset Timer/Counter1**

When this bit is one, Timer/Counter1 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

12. 16-bit Timer/Counter1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

12.1 Features

- **True 16-bit Design (i.e., Allows 16-bit PWM)**
- **Two independent Output Compare Units**
- **Four Controlled Output Pins per Output Compare Unit**
- **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**
- **Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

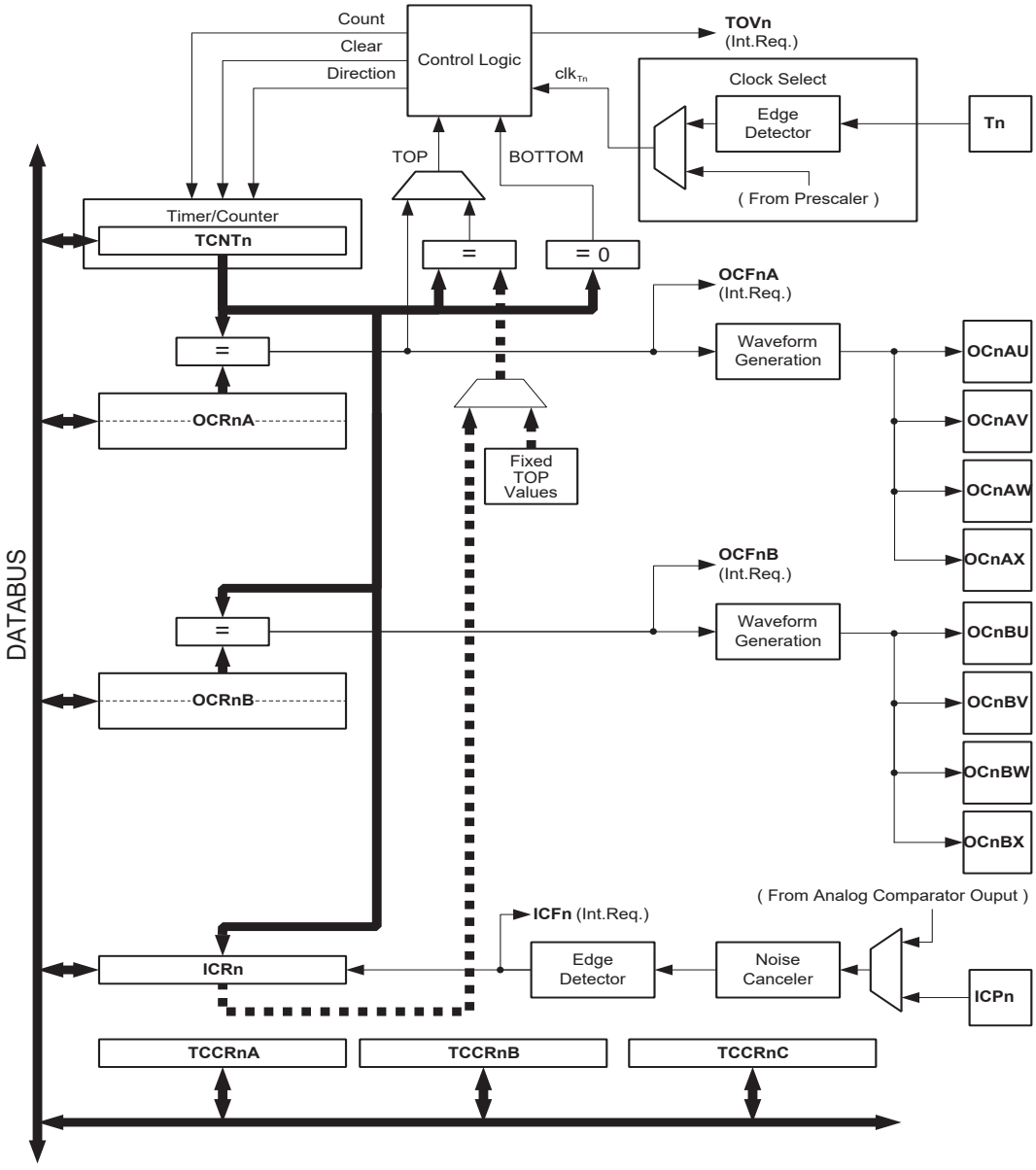
12.2 Overview

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

- A lower case “n” replaces the Timer/Counter number, in this case 1. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.
- A lower case “x” replaces the Output Compare unit channel, in this case A or B. However, when using the register or bit defines in a program, the precise form must be used, i.e., OCR1A for accessing Timer/Counter1 output compare channel A value and so on.
- A lower case “i” replaces the index of the Output Compare output pin, in this case U, V, W or X. However, when using the register or bit defines in a program, the precise form must be used.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 12-1. For the actual placement of I/O pins, refer to “Pin Configuration” on page 11. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “Register Description” on page 139.

Figure 12-1. 16-bit Timer/Counter1 Block Diagram[1]



Note: 1. Refer to Figure 1-2 on page 11, Table 9-6 on page 89, and Table 9-3 on page 84 for Timer/Counter1 pin placement and description.

12.2.1 Registers

The Timer/Counter (TCNT1), Output Compare Registers (OCR1A/B), and Input Capture Register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 119. The Timer/Counter Control Registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR1). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK1). TIFR1 and TIMSK1 are not shown in the figure.

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

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins See ”Output Compare Units” on page 126.. The compare match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an Output Compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICP1) or on the Analog Comparator pins (See ”AnaComp - Analog Comparator” on page 218.). The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register 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 ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

12.2.2 Definitions

The following definitions are used extensively throughout the section:

BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65,535).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 Register. The assignment is dependent of the mode of operation.

12.3 Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 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 stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

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

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

12.3.1 Code Examples

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly Code Examples[1]

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

C Code Examples[1]

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

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

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

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 do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[1]

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

C Code Example[1]

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

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

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

The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[1]

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

C Code Example[1]

```
void TIM16_WriteTCNT1(unsigned int i)
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Set TCNT1 to i */
  TCNT1 = i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

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

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

12.3.2 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, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

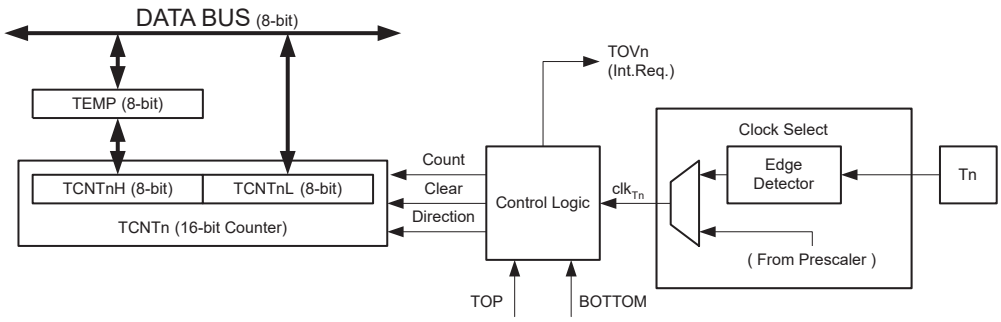
12.4 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS1[2:0]) bits located in the Timer/Counter control Register B (TCCR1B). For details on clock sources and prescaler, see “Timer/Counter1 Prescaler” on page 114.

12.5 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 12-2 shows a block diagram of the counter and its surroundings.

Figure 12-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_T1	Timer/Counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT1H) containing the upper eight bits of the counter, and Counter Low (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_T1). The clk_T1 can be generated from an external or internal clock source, selected by the Clock Select bits (CS1[2:0]). When no clock source is selected (CS1[2:0] = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_T1 is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the Waveform Generation mode bits (WGM1[3:0]) located in the Timer/Counter Control Registers A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC1A/B. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 129.

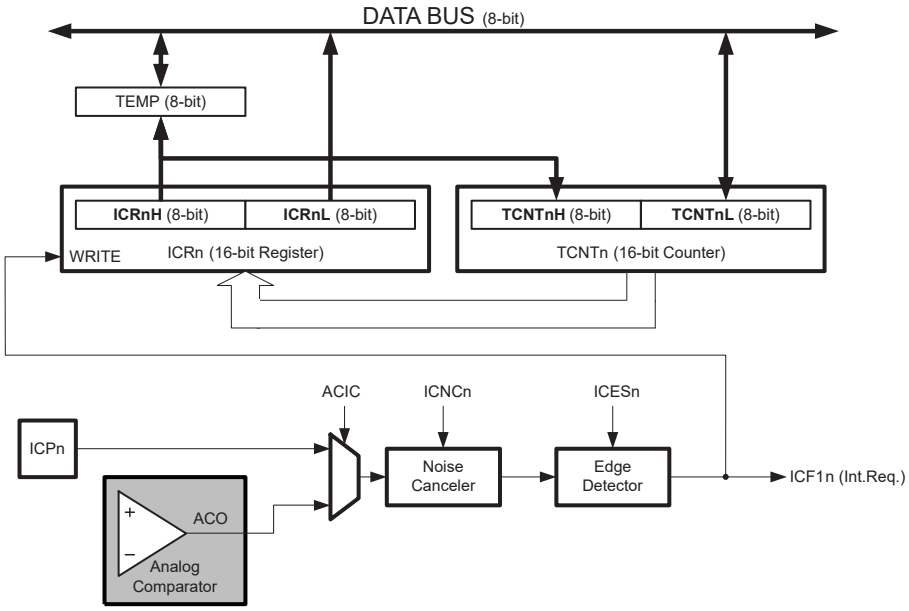
The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation selected by the WGM1[3:0] bits. TOV1 can be used for generating a CPU interrupt.

12.6 Input Capture Unit

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

The Input Capture unit is illustrated by the block diagram shown in Figure 12-3. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 12-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the Input Capture pin (ICP1), alternatively on the Analog Comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the Input Capture Register (ICR1). The Input Capture Flag (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (ICIE1 = 1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 flag can be cleared by software by writing a logical one to its I/O bit location.

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

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In these cases the Waveform Generation mode (WGM1[3:0]) bits must be set before the TOP value can be written to the ICR1

Register. When writing the ICR1 Register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 119.

12.6.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the Input Capture pin (ICP1). Only Timer/Counter1 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 (ICP1) and the Analog Comparator output (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 11-1 on page 114). 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. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

12.6.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the Input Capture Noise Canceler (ICNC1) bit in Timer/Counter Control Register B (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

12.6.3 Using the Input Capture Unit

The main challenge when using the Input Capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 Register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the Input Capture interrupt, the ICR1 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 ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 flag is not required (if an interrupt handler is used).

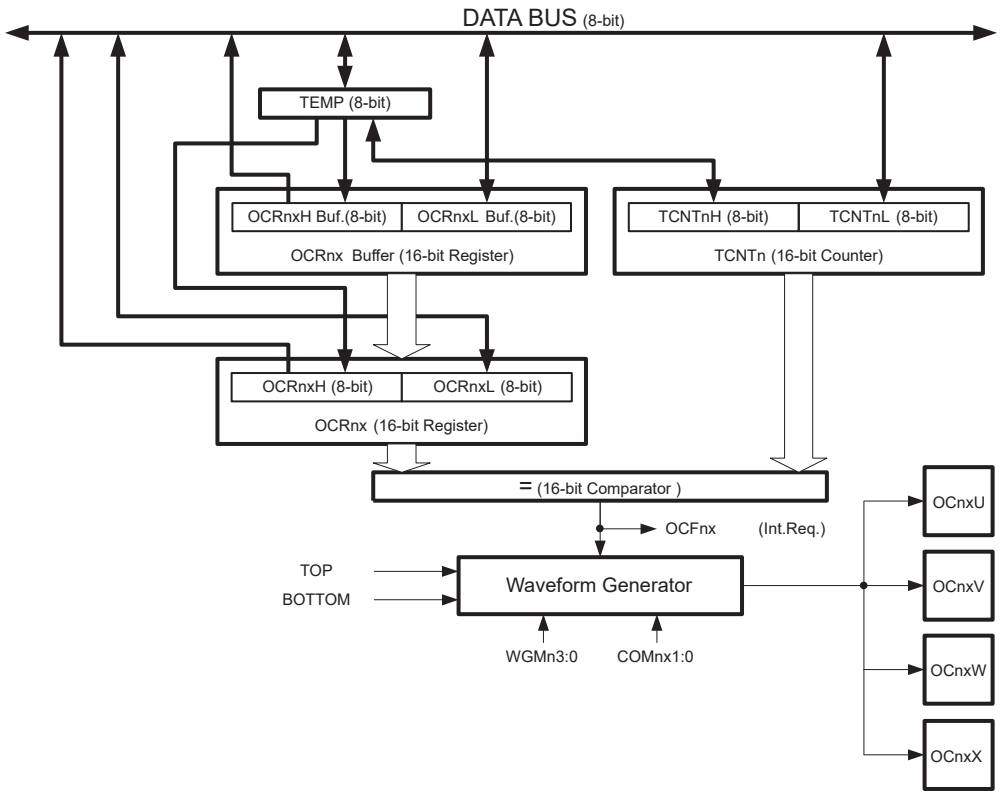
12.7 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the Output Compare Register (OCR1A/B). If TCNT equals OCR1A/B the comparator signals a match. A match will set the Output Compare Flag (OCF1A/B) at the next timer clock cycle. If enabled (OCIE1A/B = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF1A/B flag is automatically cleared when the interrupt is executed. Alternatively the OCF1A/B flag can be cleared by software by writing a logical one to its I/O bit locations. The Waveform Generator uses the match signal to generate an output according to operating mode set by the Waveform Generation mode (WGM1[3:0]) bits and Compare Output mode (COM1A/B[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” on page 129.)

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.

Figure 12-4 shows 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 12-4. Output Compare Unit, Block Diagram



The OCR1A/B 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 OCR1A/B Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1A/B Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1A/B Buffer Register, and if double buffering is disabled the CPU will access the OCR1A/B directly. The content of the OCR1A/B (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1A/B is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCR1A/B Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR1A/BH) 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 (OCR1A/BL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR1A/B buffer or OCR1A/B Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 119.

12.7.1 Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC1A/B) bit. Forcing compare match will not set the OCF1A/B flag or reload/clear the timer, but the OC1A/Bi pins will be updated as if a real compare match had occurred (the COM1A/B[1:0] bits settings define whether the OC1A/Bi pins are set, cleared or toggled - if the respective OCnxi bit is set).

12.7.2 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1A/B to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

12.7.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1A/B value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

The setup of the OC1A/B should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1A/B value is to use the Force Output Compare (FOC1A/B) strobe bits in Normal mode. The OC1A/B Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM1A/B[1:0] bits are not double buffered together with the compare value. Changing the COM1A/B[1:0] bits will take effect immediately.

12.8 Compare Match Output Unit

The Compare Output mode (COM1A/B[1:0]) bits have two functions. The Waveform Generator uses the COM1A/B[1:0] bits for defining the Output Compare (OC1A/B) state at the next compare match. Secondly the COM1A/B[1:0] and OCnxi bits control the OC1A/Bi pin output source. Figure 12-6 shows a simplified schematic of the logic affected by the COM1A/B[1:0] and OCnxi bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts

of the general I/O port control registers (DDR and PORT) that are affected by the COM1A/B[1:0] and OCnxi bits are shown. When referring to the OC1A/B state, the reference is for the internal OC1A/B Register, not the OC1A/Bi pin. If a system reset occur, the OC1A/B Register is reset to “0”.

Figure 12-5. Compare Match Output

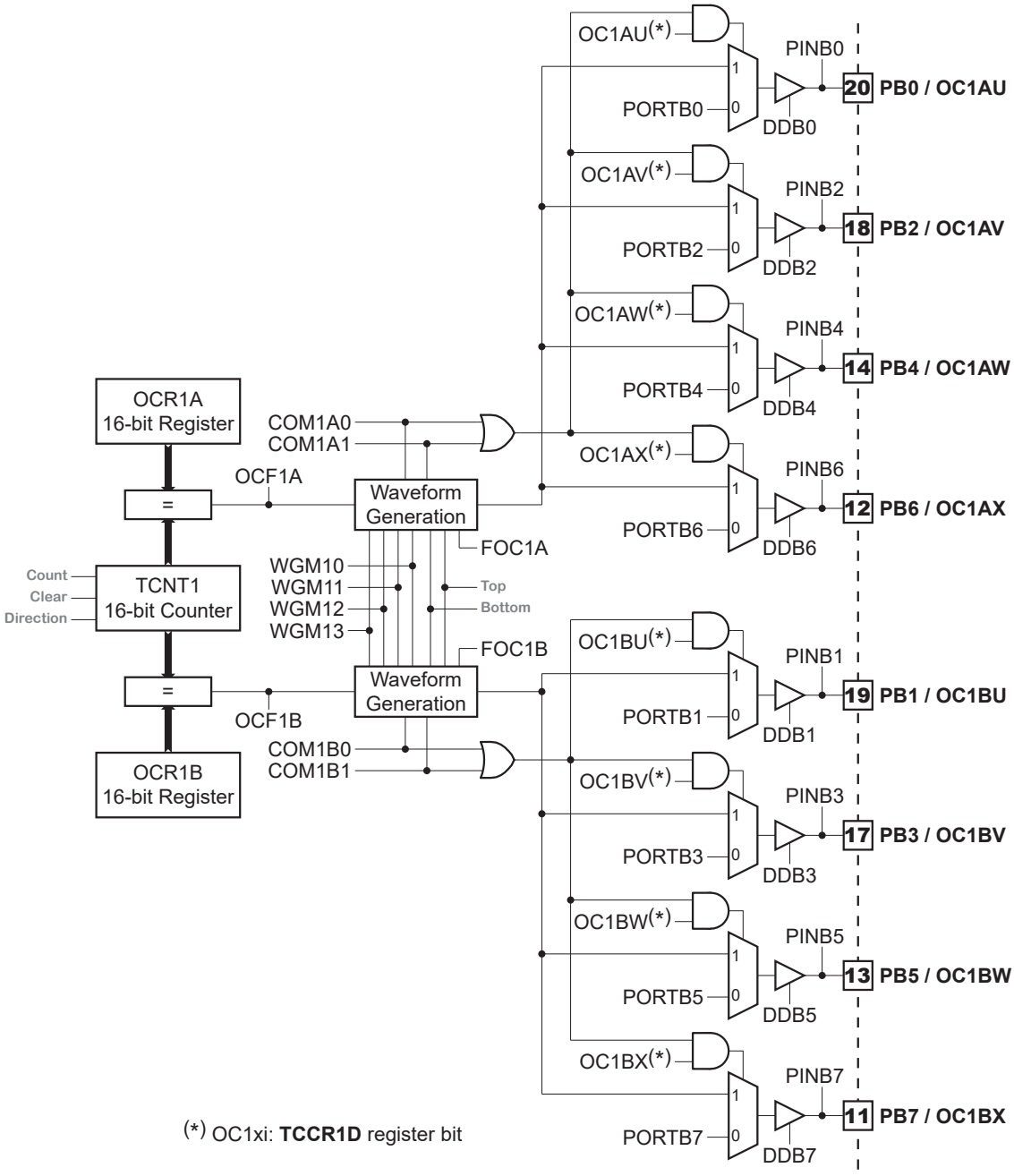
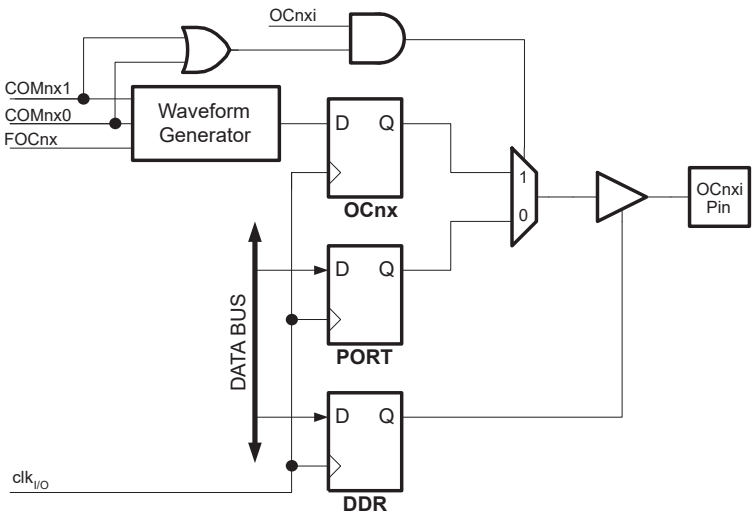


Figure 12-6. Compare Match Output Logic



12.8.1 Compare Output Function

The general I/O port function is overridden by the Output Compare (OC1A/B) from the Waveform Generator if either of the COM1A/B[1:0] bits are set and if OCnxi respective bit is set in TCCR1D register. However, the OC1A/Bi pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC1A/Bi pin (DDR_OC1A/Bi) must be set as output before the OC1A/B value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 12-1, Table 12-2 and Table 12-3 for details.

The design of the Output Compare pin logic allows initialization of the OC1A/B state before the output is enabled. Note that some COM1A/B[1:0] bit settings are reserved for certain modes of operation. See “Register Description” on page 139.

The COM1A/B[1:0] bits have no effect on the Input Capture unit.

12.8.2 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM1A/B[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1A/B[1:0] = 0 tells the Waveform Generator that no action on the OC1A/B Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 12-1 on page 139. For fast PWM mode refer to Table 12-2 on page 140, and for phase correct and phase and frequency correct PWM refer to Table 12-3 on page 140.

A change of the COM1A/B[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 FOC1A/B strobe bits.

12.9 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM1[3:0]) and Compare Output mode (COM1A/B[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1A/B[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1A/B[1:0] bits control whether the output should be set, cleared

or toggle at a compare match (See ”Compare Match Output Unit” on page 127.). The OCnxi bits over control the setting of the COM1A/B[1:0] bits as shown in Figure 12-6 on page 129.

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 137.

12.9.1 Normal Mode

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

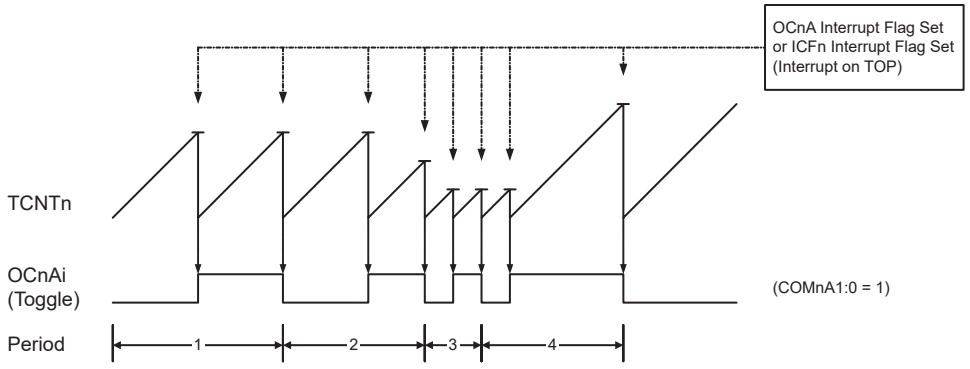
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

12.9.2 Clear Timer on Compare Match (CTC) Mode

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

The timing diagram for the CTC mode is shown in Figure 12-7. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 12-7. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM1[3:0] = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM1A[1:0] = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1) and OC1Ai is set. The waveform generated will have a maximum frequency of $f_{OC}1_A = f_{clk_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

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

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

As for the Normal mode of operation, the TOV1 flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

12.9.3 Fast PWM Mode

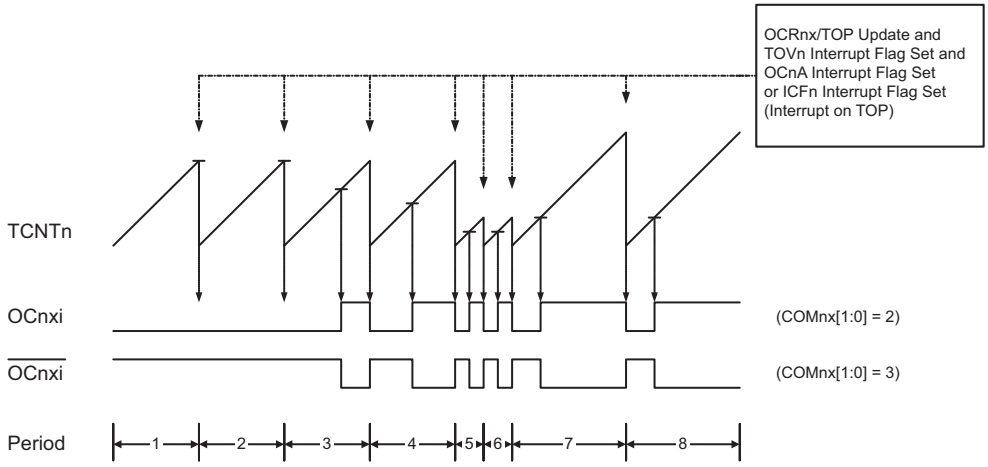
The fast Pulse Width Modulation or fast PWM mode (WGM1[3:0] = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1A/B) is set on the compare match between TCNT1 and OCR1A/B, and cleared at TOP. In inverting Compare Output mode output is cleared on compare match and set at TOP. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

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

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

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM1[3:0] = 5, 6, or 7), the value in ICR1 (WGM1[3:0] = 14), or the value in OCR1A (WGM1[3:0] = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 12-8. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1A/B and TCNT1. The OC1A/B interrupt flag will be set when a compare match occurs.

Figure 12-8. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OC1A or ICF1 flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1A/B. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1A/B Registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 flag is set.

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

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1A/B pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1A/B[1:0] to three (see Table 12-2 on page 140). The actual OC1A/B value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1A/B) and OC1A/Bi is set. The PWM waveform is generated by setting (or clearing) the OC1A/B Register at the compare match between OCR1A/B and TCNT1, and clearing (or setting) the OC1A/B 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:

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

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1A/B Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1A/B is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1A/B equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1A/B[1:0] bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each compare match (COM1A[1:0] = 1). The waveform generated will have a maximum frequency of $f_{OC1_A} = f_{clk_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

12.9.4 Phase Correct PWM Mode

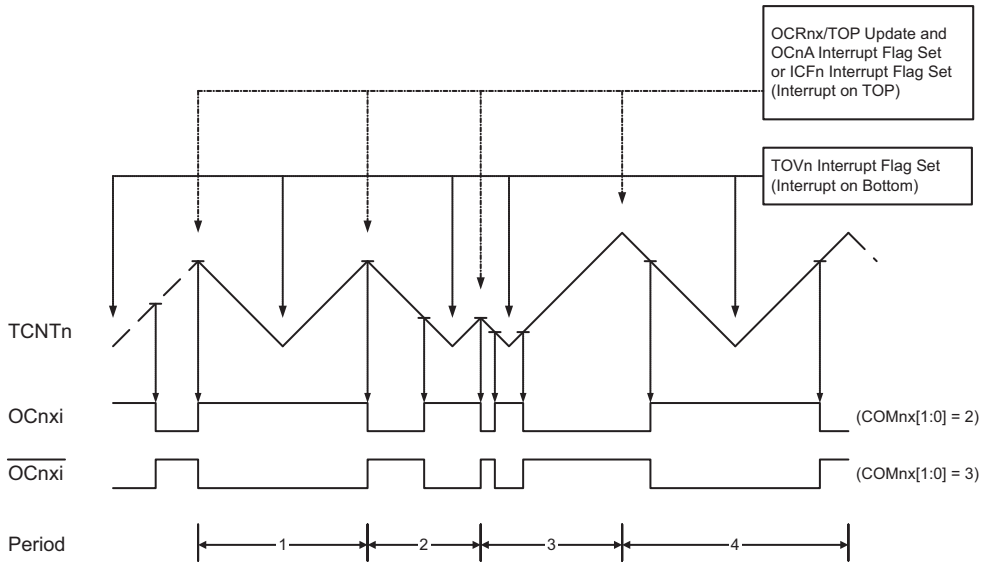
The phase correct Pulse Width Modulation or phase correct PWM mode (WGM1[3:0] = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1A/B) is cleared on the compare match between TCNT1 and OCR1A/B while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

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

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

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM1[3:0] = 1, 2, or 3), the value in ICR1 (WGM1[3:0] = 10), or the value in OCR1A (WGM1[3:0] = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 12-9. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1A/B and TCNT1. The OC1A/B interrupt flag will be set when a compare match occurs.

Figure 12-9. Phase Correct PWM Mode, Timing Diagram



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

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1A/B. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1A/B Registers are written. As the third period shown in Figure 12-9 illustrates, 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 OCR1A/B Register. Since the OCR1A/B 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 OC1A/B pins. Setting the COM1A/B[1:0] bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1A/B[1:0] to three (See Table on page 140). The actual OC1A/B value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1A/B) and OC1A/Bi is set. The PWM waveform is generated by setting (or clearing) the OC1A/B Register at the compare match between OCR1A/B and TCNT1 when the counter increments, and clearing (or setting) the OC1A/B Register at compare match between OCR1A/B and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1A/B Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1A/B 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.

12.9.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation, or phase and frequency correct PWM mode (WGM1[3:0] = 8 or 9) 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 (OC1A/B) is cleared on the compare match between TCNT1 and OCR1A/B while upcounting, and set on the compare match while downcounting. 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 OCR1A/B Register is updated by the OCR1A/B Buffer Register, (see Figure 12-9 and Figure 12-10).

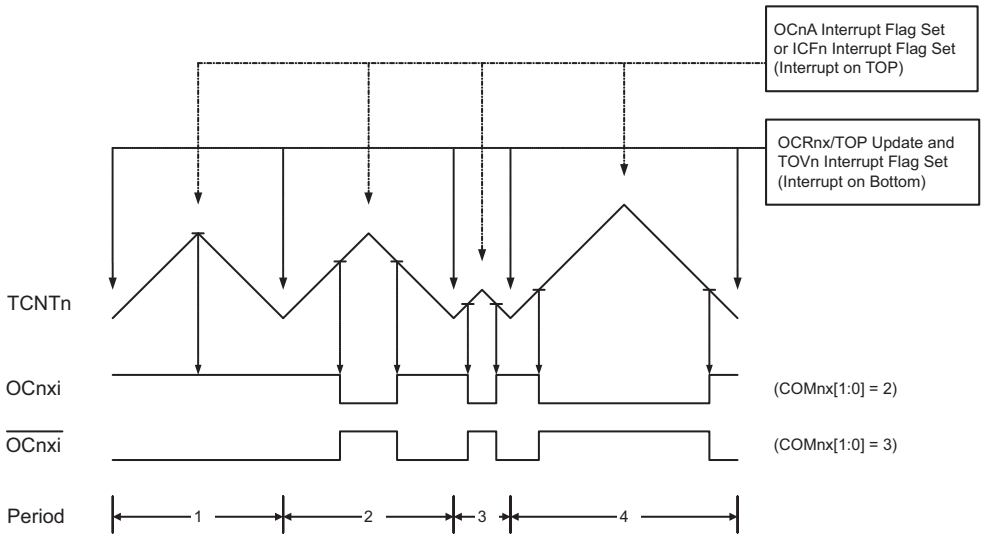
The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following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

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

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM1[3:0] = 8), or the value in OCR1A (WGM1[3:0] = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency

correct PWM mode is shown on Figure 12-10. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1A/B and TCNT1. The OC1A/B interrupt flag will be set when a compare match occurs.

Figure 12-10. Phase and Frequency Correct PWM Mode, Timing Diagram



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

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1A/B.

As Figure 12-10 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1A/B Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

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

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1A/B pins. Setting the COM1A/B[1:0] bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1A/B[1:0] to three (See Table on page 140). The actual OC1A/B value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1A/B) and OC1A/Bi is set. The PWM waveform is generated by setting (or clearing) the OC1A/B Register at the compare match between

OCR1A/B and TCNT1 when the counter increments, and clearing (or setting) the OC1A/B Register at compare match between OCR1A/B and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

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

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1A/B Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1A/B 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.

12.10 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_T1) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR1A/B Register is updated with the OCR1A/B buffer value (only for modes utilizing double buffering). Figure 12-11 shows a timing diagram for the setting of OCF1A/B.

Figure 12-11. Timer/Counter Timing Diagram, Setting of OCF1A/B, no Prescaling

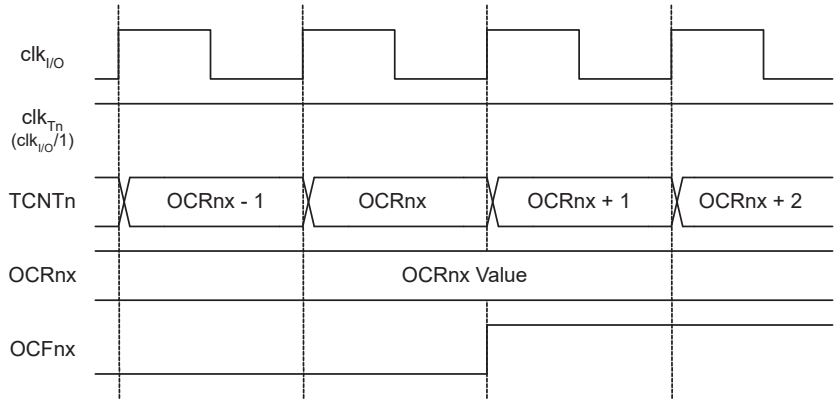


Figure 12-12 shows the same timing data, but with the prescaler enabled.

Figure 12-12. Timer/Counter Timing Diagram, Setting of OCF1A/B, with Prescaler ($f_{clk_I/O}/8$)

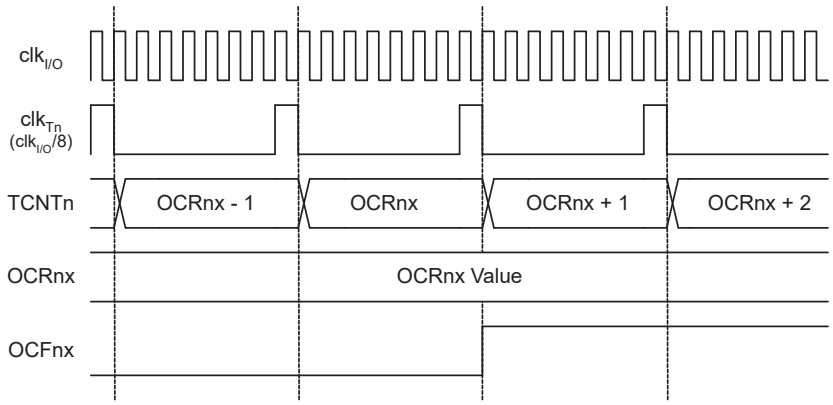


Figure 12-13 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1A/B Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 flag at BOTTOM.

Figure 12-13. Timer/Counter Timing Diagram, no Prescaling

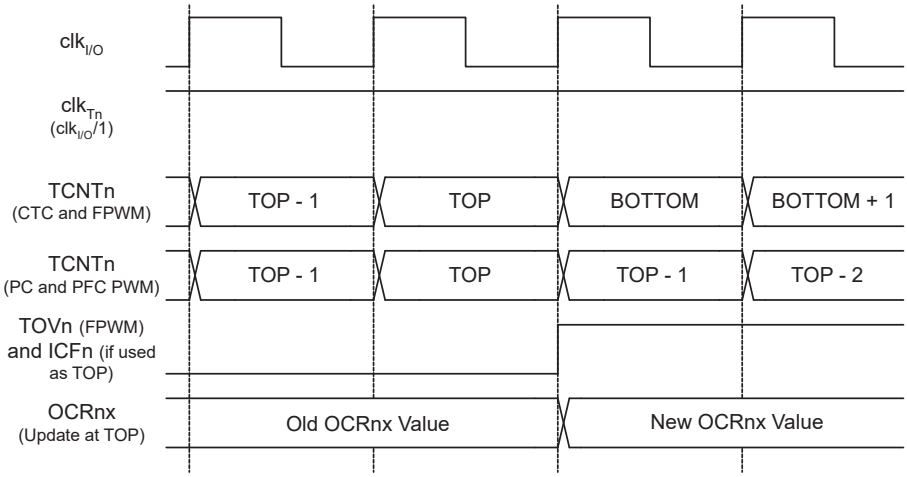
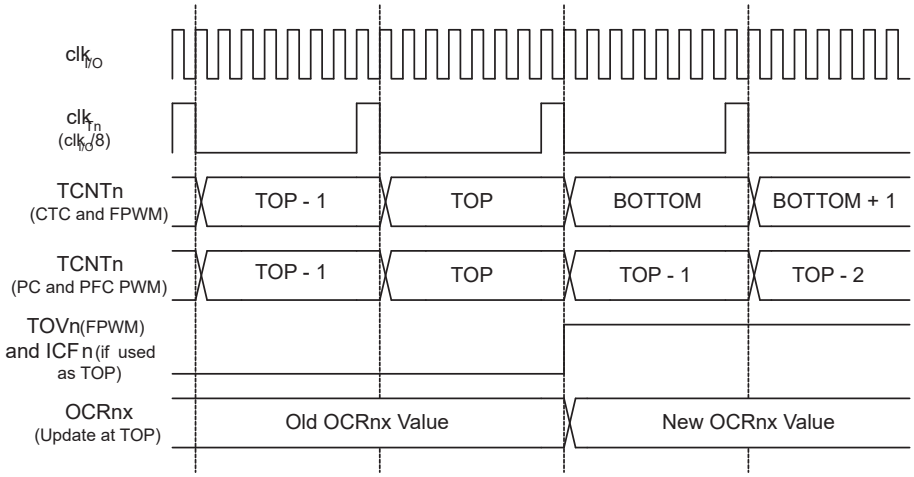


Figure 12-14 shows the same timing data, but with the prescaler enabled.

Figure 12-14. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)



12.11 Register Description

12.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM1A[1:0]: Compare Output Mode for Channel A**

- **Bits 5:4 – COM1B[1:0]: Compare Output Mode for Channel B**

The COM1A[1:0] and COM1B[1:0] control the Output Compare pins (OC1Ai and OC1Bi respectively) behavior. If one or both of the COM1A[1:0] bits are written to one, the OC1Ai output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B[1:0] bit are written to one, the OC1Bi output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit and OC1xi bit (TCCR1D) corresponding to the OC1Ai or OC1Bi pin must be set in order to enable the output driver.

When the OC1Ai or OC1Bi is connected to the pin, the function of the COM1A/B[1:0] bits is dependent of the WGM1[3:0] bits setting. Table 12-1 shows the COM1A/B[1:0] bit functionality when the WGM1[3:0] bits are set to a Normal or a CTC mode (non-PWM).

Table 12-1. Compare Output Mode, non-PWM

OC1Ai OC1Bi	COM1A1 COM1B1	COM1A0 COM1B0	Description
0	x	x	Normal port operation, OC1A/OC1B disconnected.
1	0	0	
	0	1	Toggle OC1A/OC1B on Compare Match.
	1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
	1	1	Set OC1A/OC1B on Compare Match (Set output to high level).

Table 12-2 shows the COM1A/B[1:0] bit functionality when the WGM1[3:0] bits are set to the fast PWM mode.

Table 12-2. Compare Output Mode, Fast PWM (1)

OC1Ai OC1Bi	COM1A1 COM1B1	COM1A0 COM1B0	Description
0	x	x	Normal port operation, OC1A/OC1B disconnected.
1	0	0	
1	0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected. WGM13=1: Toggle OC1A on Compare Match, OC1B reserved.
1	1	0	Clear OC1A/OC1B on Compare Match Set OC1A/OC1B at TOP
1	1	1	Set OC1A/OC1B on Compare Match Clear OC1A/OC1B at TOP

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 131. for more details.

Table 12-3 shows the COM1A/B[1:0] bit functionality when the WGM1[3:0] bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 12-3. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM(1)

OC1Ai OC1Bi	COM1A1 COM1B1	COM1A0 COM1B0	Description
0	x	x	Normal port operation, OC1A/OC1B disconnected.
1	0	0	
1	0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected. WGM13=1: Toggle OC1A on Compare Match, OC1B reserved.
1	1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

Note: 1. A special case occurs when OC1A/OC1B equals TOP and COM1A1/COM1B1 is set. See “Phase Correct PWM Mode” on page 133. for more details.

- **Bits 3:2 – Res: Reserved Bits**

These bits are reserved for future use.

- **Bits 1:0 – WGM1[1:0]: Waveform Generation Mode**

Combined with the WGM1[3:2] bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 12-4. 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” on page 129.).

Table 12-4. Waveform Generation Mode Bit Description (1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1A/B at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

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

12.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: 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 (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

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

This bit selects which edge on the Input Capture pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture Register (ICR1). The event will also set the Input Capture Flag (ICF1), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM1[3:0] bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

- **Bit 5 – Res: Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

- **Bits 4:3 – WGM1[3:2]: Waveform Generation Mode**

See TCCR1A Register description.

- **Bits 2:0 – CS1[2:0]: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 12-11 and Figure 12-12.

Table 12-5. Clock Select Bit Description

CS12	CS11	CS10	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	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

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

12.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**

- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM1[3:0] bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1nx output is changed according to its COM1A/B[1:0] and OC1nx bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1A/B[1:0] bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

12.11.4 TCCR1D – Timer/Counter1 Control Register D

Bit	7	6	5	4	3	2	1	0	
(0x83)	OC1BX	OC1BW	OC1BV	OC1BU	OC1AX	OC1AW	OC1AV	OC1AU	TCCR1D
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – OC1Bi: Output Compare Pin Enable for Channel B**

The OC1Bi bits enable the Output Compare pins of Channel B as shown in Figure 12-6 on page 129.

- **Bits 3:0 – OC1Ai: Output Compare Pin Enable for Channel A**

The OC1Ai bits enable the Output Compare pins of Channel A as shown in Figure 12-6 on page 129.

12.11.5 TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 119.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1A/B Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

12.11.6 OCR1AH and OCR1AL – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.11.7 OCR1BH and OCR1BL – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A/B pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 119.

12.11.8 ICR1H and ICR1L – Input Capture Register

Bit	7	6	5	4	3	2	1	0	
(0x87)	ICR1[15:8]								ICR1H
(0x86)	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the Analog Comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 119.

12.11.9 TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved for future use.

- **Bit 5 – ICIE1: 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/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector (See ”Interrupt Vectors in ATtiny87/167” on page 67.) is executed when the ICF1 flag, located in TIFR1, is set.

- **Bits 4:3 – Res: Reserved Bits**

These bits are reserved for future use.

- **Bit 2 – OCIE1B: 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/Counter1 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector (See ”Interrupt Vectors in ATtiny87/167” on page 67.) is executed when the OCF1B flag, located in TIFR1, is set.

- **Bit 1 – OCIE1A: 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/Counter1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (See ”Interrupt Vectors in ATtiny87/167” on page 67.) is executed when the OCF1A flag, located in TIFR1, is set.

- **Bit 0 – TOIE1: Timer/Counter 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/Counter1 Overflow interrupt is enabled. The corresponding Interrupt Vector (See ”Interrupt Vectors in ATtiny87/167” on page 67.) is executed when the TOV1 flag, located in TIFR1, is set.

12.11.10 TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved for future use.

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

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM1[3:0] to be used as the TOP value, the ICF1 flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

- **Bits 4:3 – Res: Reserved Bits**

These bits are reserved for future use.

- **Bit 2 – OCF1B: Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a Forced Output Compare (FOC1B) strobe will not set the OCF1B flag.

OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

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

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV1: Timer/Counter Overflow Flag**

The setting of this flag is dependent of the WGM1[3:0] bits setting. In Normal and CTC modes, the TOV1 flag is set when the timer overflows. Refer to Table 12-4 on page 141 for the TOV1 flag behavior when using another WGM1[3:0] bit setting.

TOV1 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

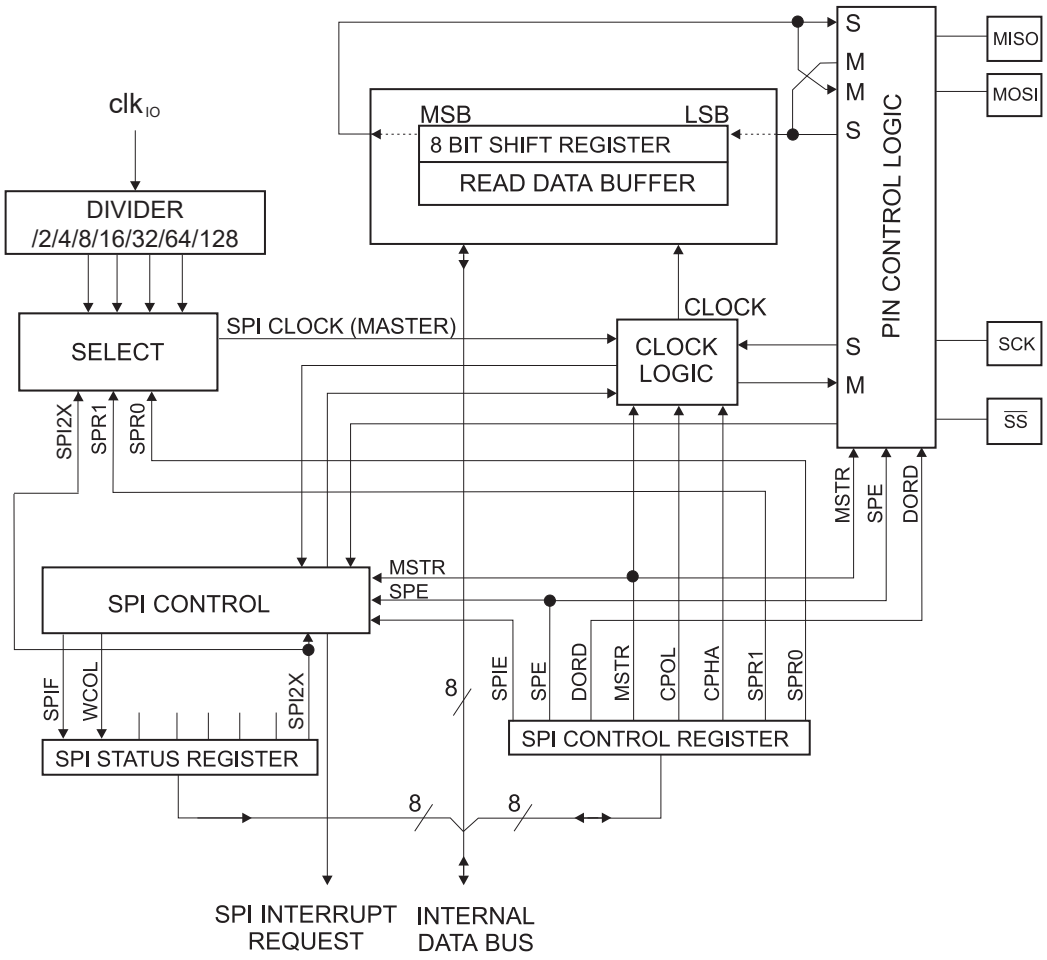
13. SPI - Serial Peripheral Interface

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATtiny87/167 and peripheral devices or between several AVR devices. The ATtiny87/167 SPI includes the following features:

13.1 Features

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

Figure 13-1. SPI Block Diagram(1)



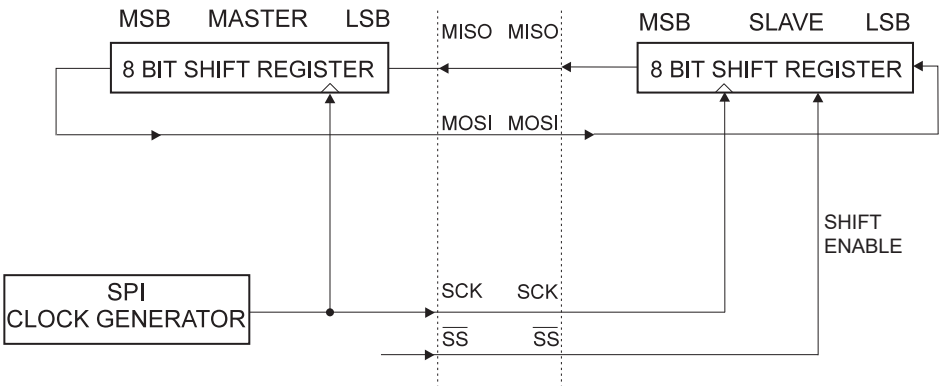
Note: 1. Refer to Figure 1.4 on page 11, and Table 9-3 on page 84 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 13-2. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

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

Figure 13-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed $f_{clkio}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 13-1. For more details on automatic port overrides, refer to “Alternate Port Functions” on page 80.

Table 13-1. SPI Pin Overrides[1]

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

Note: 1. See “Alternate Functions of Port B” on page 89 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission.

DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB2, replace DD_MOSI with DDB2 and DDR_SPI with DDRB.

Assembly Code Example[1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi      r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out      DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi      r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out      SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out      SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  in       r17,SPSR
  sbrs     r17,SPIF
  rjmp     Wait_Transmit
  ret
```

C Code Example[1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)));
}
```

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

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example[1]

```
SPI_SlaveInit:
  ; Set MISO output, all others input
  ldi     r17,(1<<DD_MISO)
  out     DDR_SPI,r17
  ; Enable SPI
  ldi     r17,(1<<SPE)
  out     SPCR,r17
  ret

SPI_SlaveReceive:
  ; Wait for reception complete
  sbis    SPSR,SPIF
  rjmp    SPI_SlaveReceive
  ; Read received data and return
  in      r16,SPDR
  ret
```

C Code Example[1]

```
void SPI_SlaveInit(void)
{
  /* Set MISO output, all others input */
  DDR_SPI = (1<<DD_MISO);
  /* Enable SPI */
  SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
  /* Wait for reception complete */
  while(!(SPSR & (1<<SPIF)));
  /* Return data register */
  return SPDR;
}
```

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

13.2 $\overline{SS}$ Pin Functionality

13.2.1 Slave Mode

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

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

13.2.2 Master Mode

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

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

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

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

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

13.2.3 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	**SPIE**	**SPE**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**	**SPCR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

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

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

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

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

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 13-3 and Figure 13-4 for an example. The CPOL functionality is summarized below:

Table 13-2. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 13-3 and Figure 13-4 for an example. The CPOL functionality is summarized below:

Table 13-3. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1:0 – SPR[1:0]: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the clk_{IO} frequency f_{clkio} is shown in the following table:

Table 13-4. Relationship Between SCK and the Oscillator Frequency

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

13.2.4 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

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

- **Bit 6 – WCOL: Write COLlision Flag**

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

- **Bits 5:1 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and will always read as zero.

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

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 13-4). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only ensured to work at $f_{clkio}/4$ or lower.

The SPI interface on the ATtiny87/167 is also used for program memory and EEPROM down-loading or uploading. See “Serial Downloading” on page 245 for serial programming and verification.

13.2.5 SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

- **Bits 7:0 – SPD[7:0]: SPI Data**

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

13.3 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 13-3 and Figure 13-4. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 13-2 and Table 13-3, as done below:

Table 13-5. CPOL Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)	0
CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)	1
CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)	2
CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)	3

Figure 13-3. SPI Transfer Format with CPHA = 0

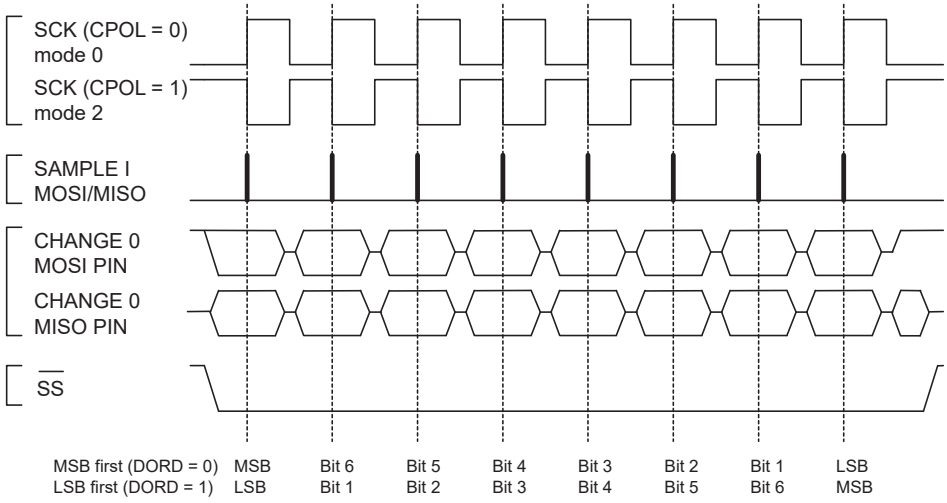
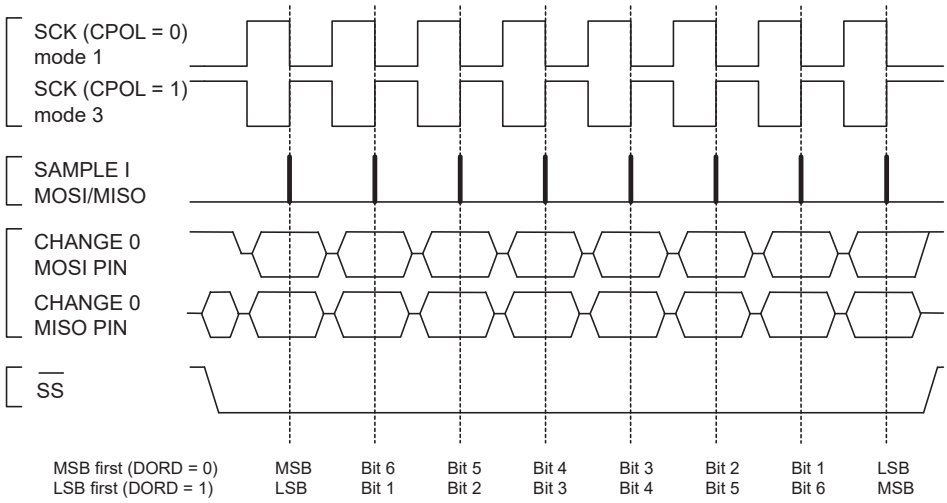


Figure 13-4. SPI Transfer Format with CPHA = 1



14. USI – Universal Serial Interface

14.1 Features

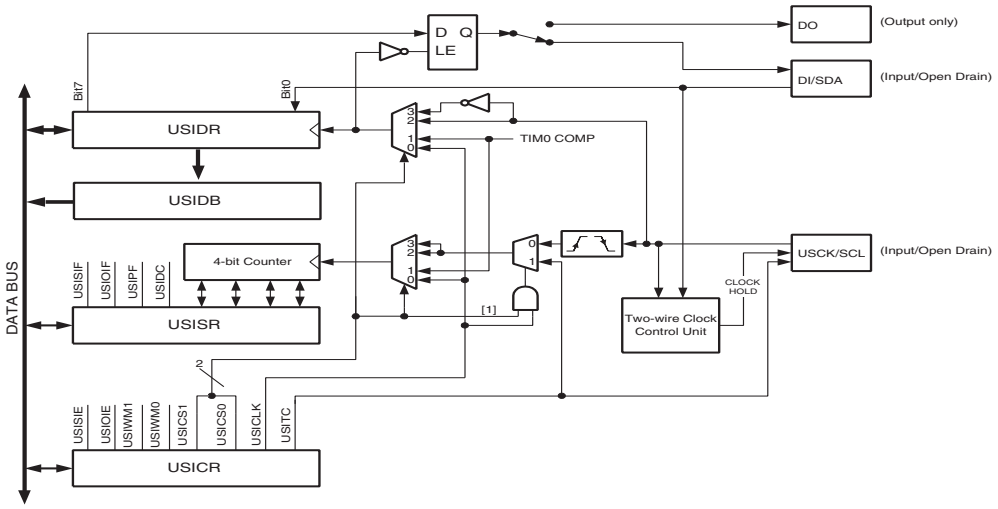
- **Two-wire Synchronous Data Transfer (Master or Slave)**
- **Three-wire Synchronous Data Transfer (Master or Slave)**
- **Data Received Interrupt**
- **Wakeup from Idle Mode**
- **In Two-wire Mode: Wake-up from All Sleep Modes, Including Power-down Mode**
- **Two-wire Start Condition Detector with Interrupt Capability**

14.2 Overview

The Universal Serial Interface, or USI, provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize the processor load.

A simplified block diagram of the USI is shown on Figure 14-1 For the actual placement of I/O pins, refer to “Pin Configuration” on page 11. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “Register Description” on page 163.

Figure 14-1. Universal Serial Interface, Block Diagram



The 8-bit USI Data Register is directly accessible via the data bus and contains the incoming and outgoing data. The register has no buffering so the data must be read as quickly as possible to ensure that no data is lost. The USI Data Register is a serial shift register and the most significant bit that is the output of the serial shift register is connected to one of two output pins depending of the wire mode configuration.

A transparent latch is inserted between the USI Data Register Output and output pin, which delays the change of data output to the opposite clock edge of the data input sampling. The serial input is always sampled from the Data Input (DI) pin independent of the configuration.

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the USI Data Register and the counter are clocked simultaneously by the same clock source. This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected the counter counts both clock edges. In this case the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: The USCK pin, Timer/Counter0 Compare Match or from software.

The Two-wire clock control unit can generate an interrupt when a start condition is detected on the Two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

14.3 Functional Descriptions

14.3.1 Three-wire Mode

The USI Three-wire mode is compliant to the Serial Peripheral Interface (SPI) mode 0 and 1, but does not have the slave select ($\overline{SS}$) pin functionality. However, this feature can be implemented in software if necessary. Pin names used by this mode are: DI, DO, and USCK.

Figure 14-2. Three-wire Mode Operation, Simplified Diagram

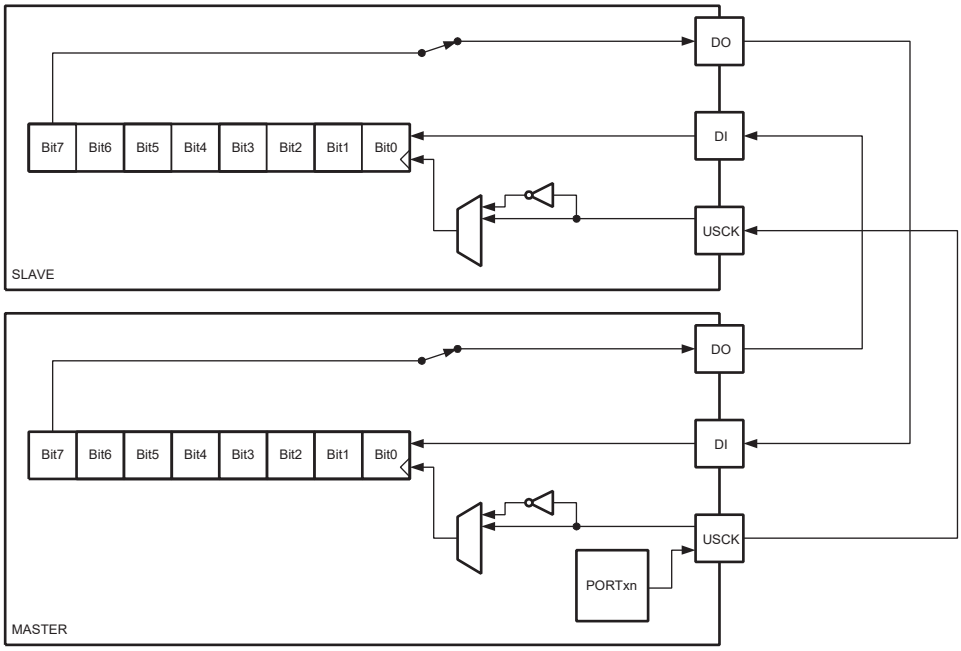
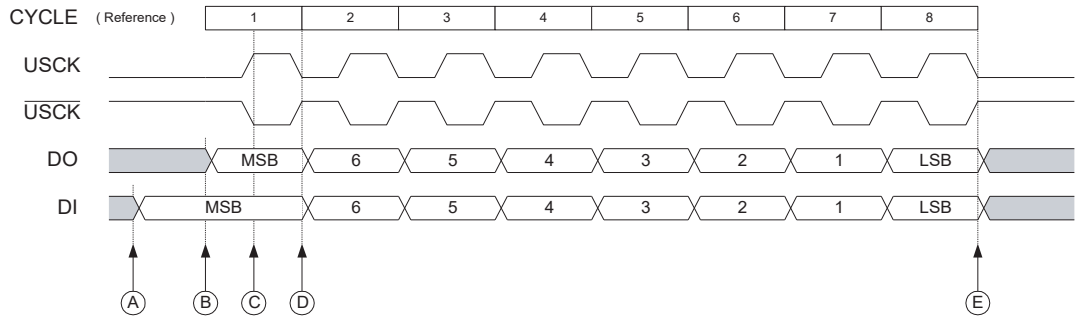


Figure 14-2 shows two USI units operating in Three-wire mode, one as Master and one as Slave. The two USI Data Register are interconnected in such way that after eight USCK clocks, the data in each register are interchanged. The same clock also increments the USI's 4-bit counter. The Counter Overflow (interrupt) Flag, or USIOIF, can therefore be used to determine when a transfer is completed.

The clock is generated by the Master device software by toggling the USCK pin via the PORT Register or by writing a one to the USITC bit in USICR.

Figure 14-3. Three-wire Mode, Timing Diagram



The Three-wire mode timing is shown in Figure 14-3 At the top of the figure is a USCK cycle reference. One bit is shifted into the USI Data Register (USIDR) for each of these cycles. The USCK timing is shown for both external clock modes. In External Clock mode 0 (USICS0 = 0), DI is sampled at positive edges, and DO is changed (Data Register is shifted by one) at negative edges. External Clock mode 1 (USICS0 = 1) uses the opposite edges versus mode 0, i.e., samples data at negative and changes the output at positive edges. The USI clock modes corresponds to the SPI data mode 0 and 1.

Referring to the timing diagram (Figure 14-3), a bus transfer involves the following steps:

1. The Slave device and Master device sets up its data output and, depending on the protocol used, enables its output driver (mark A and B). The output is set up by writing the data to be transmitted to the USI Data Register. Enabling of the output is done by setting the corresponding bit in the port Data Direction Register. Note that point A and B does not have any specific order, but both must be at least one half USCK cycle before point C where the data is sampled. This must be done to ensure that the data setup requirement is satisfied. The 4-bit counter is reset to zero.
2. The Master generates a clock pulse by software toggling the USCK line twice (C and D). The bit value on the slave and master's data input (DI) pin is sampled by the USI on the first edge (C), and the data output is changed on the opposite edge (D). The 4-bit counter will count both edges.
3. Step 2. is repeated eight times for a complete register (byte) transfer.
4. After eight clock pulses (i.e., 16 clock edges) the counter will overflow and indicate that the transfer is completed. The data bytes transferred must now be processed before a new transfer can be initiated. The overflow interrupt will wake up the processor if it is set to Idle mode. Depending of the protocol used the slave device can now set its output to high impedance.

14.3.2 SPI Master Operation Example

The following code demonstrates how to use the USI module as a SPI Master:

```
SPITransfer:
    sts     USIDR,r16
    ldi     r16,(1<<USIOIF)
    sts     USISR,r16
    ldi     r16,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)
SPITransfer_loop:
```

```
    sts    USICR,r16
    lds    r16, USISR
    sbrs   r16, USIOIF
    rjmp   SPITransfer_loop
    lds    r16,USIDR
    ret
```

The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO and USCK pins are enabled as output in the DDRA or DDRB Register. The value stored in register r16 prior to the function is called is transferred to the Slave device, and when the transfer is completed the data received from the Slave is stored back into the r16 Register.

The second and third instructions clears the USI Counter Overflow Flag and the USI counter value. The fourth and fifth instruction set Three-wire mode, positive edge Shift Register clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI module as a SPI Master with maximum speed (fsck = fck/4):

```
SPITransfer_Fast:

    sts    USIDR,r16
    ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
    ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

    sts    USICR,r16 ; MSB
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16 ; LSB
    sts    USICR,r17

    lds    r16,USIDR
    ret
```

14.3.3 SPI Slave Operation Example

The following code demonstrates how to use the USI module as a SPI Slave:

```
init:
    ldi     r16,(1<<USIWM0)|(1<<USICS1)
    sts     USICR,r16
...
SlaveSPITransfer:
    sts     USIDR,r16
    ldi     r16,(1<<USIOIF)
    sts     USISR,r16
SlaveSPITransfer_loop:
    lds     r16, USISR
    sbrs    r16, USIOIF
    rjmp    SlaveSPITransfer_loop
    lds     r16,USIDR
    ret
```

The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO is configured as output and USCK pin is configured as input in the DDR Register. The value stored in register r16 prior to the function is called is transferred to the master device, and when the transfer is completed the data received from the Master is stored back into the r16 Register.

Note that the first two instructions is for initialization only and needs only to be executed once.These instructions sets Three-wire mode and positive edge USI Data Register clock. The loop is repeated until the USI Counter Overflow Flag is set.

14.3.4 Two-wire Mode

The USI Two-wire mode is compliant to the Inter IC (TWI) bus protocol, but without slew rate limiting on outputs and input noise filtering. Pin names used by this mode are SCL and SDA.

Figure 14-4. Two-wire Mode Operation, Simplified Diagram

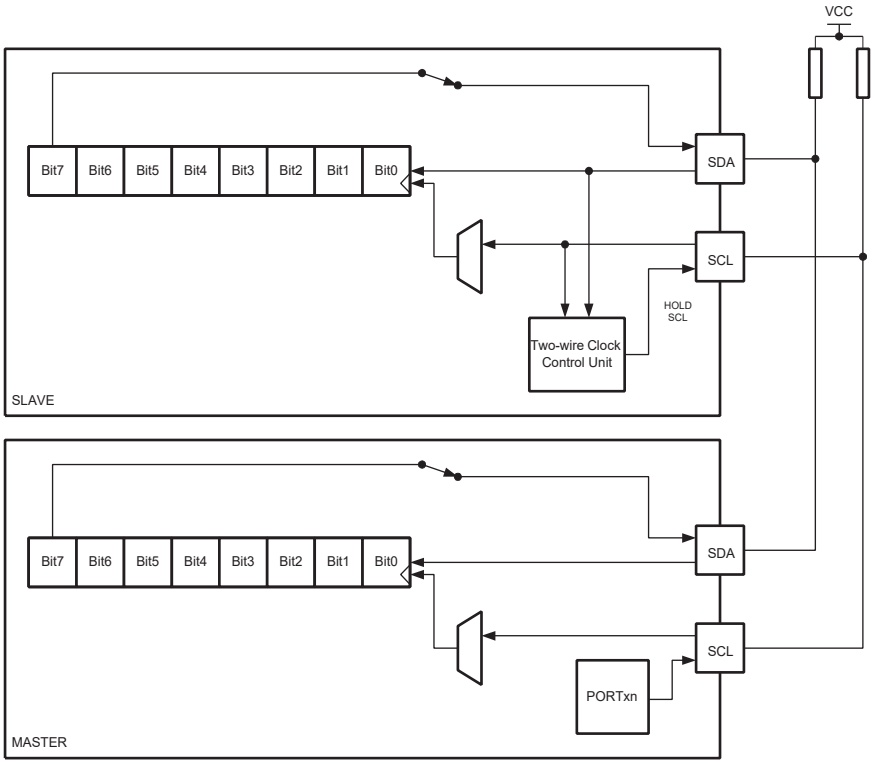
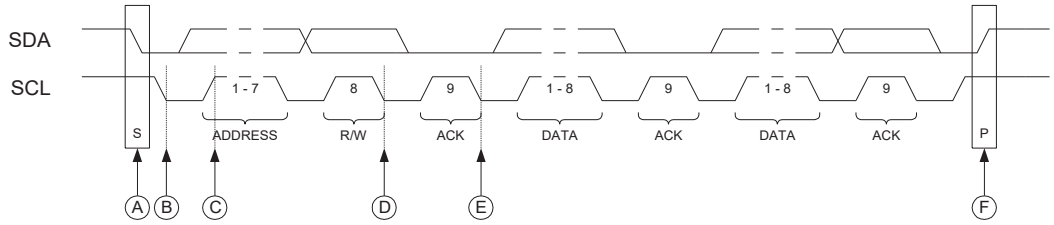


Figure 14-4 shows two USI units operating in Two-wire mode, one as Master and one as Slave. It is only the physical layer that is shown since the system operation is highly dependent of the communication scheme used. The main differences between the Master and Slave operation at this level, is the serial clock generation which is always done by the Master, and only the Slave uses the clock control unit. Clock generation must be implemented in software, but the shift operation is done automatically by both devices. Note that only clocking on negative edge for shifting data is of practical use in this mode. The slave can insert wait states at start or end of transfer by forcing the SCL clock low. This means that the Master must always check if the SCL line was actually released after it has generated a positive edge.

Since the clock also increments the counter, a counter overflow can be used to indicate that the transfer is completed. The clock is generated by the master by toggling the USCK pin via the PORT Register.

The data direction is not given by the physical layer. A protocol, like the one used by the TWI-bus, must be implemented to control the data flow.

Figure 14-5. Two-wire Mode, Typical Timing Diagram

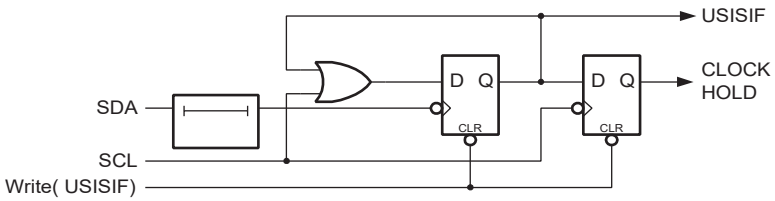


Referring to the timing diagram (Figure 14-5), a bus transfer involves the following steps:

1. The a start condition is generated by the Master by forcing the SDA low line while the SCL line is high (A). SDA can be forced low either by writing a zero to bit 7 of the Shift Register, or by setting the corresponding bit in the PORT Register to zero. Note that the USI Data Register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 14-6.) detects the start condition and sets the USISIF Flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the Master has forced an negative edge on this line (B). This allows the Slave to wake up from sleep or complete its other tasks before setting up the USI Data Register to receive the address. This is done by clearing the start condition flag and reset the counter.
3. The Master set the first bit to be transferred and releases the SCL line (C). The Slave samples the data and shift it into the USI Data Register at the positive edge of the SCL clock.
4. After eight bits are transferred containing slave address and data direction (read or write), the Slave counter overflows and the SCL line is forced low (D). If the slave is not the one the Master has addressed, it releases the SCL line and waits for a new start condition.
5. If the Slave is addressed it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the Counter Register must be set to 14 before releasing SCL at (D)). Depending of the R/W bit the Master or Slave enables its output. If the bit is set, a master read operation is in progress (i.e., the slave drives the SDA line) The slave can hold the SCL line low after the acknowledge (E).
6. Multiple bytes can now be transmitted, all in same direction, until a stop condition is given by the Master (F). Or a new start condition is given.

If the Slave is not able to receive more data it does not acknowledge the data byte it has last received. When the Master does a read operation it must terminate the operation by force the acknowledge bit low after the last byte transmitted.

Figure 14-6. Start Condition Detector, Logic Diagram



14.3.5 Start Condition Detector

The start condition detector is shown in Figure 14-6. The SDA line is delayed (in the range of 50 to 300 ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in Two-wire mode.

The start condition detector is working asynchronously and can therefore wake up the processor from the Power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case the Oscillator start-up time set by the CKSEL Fuses (see "Clock Systems and their Distribution" on page 32) must also be taken into the consideration. Refer to the USISIF bit description on page 164 for further details.

14.4 Alternative USI Usage

When the USI unit is not used for serial communication, it can be set up to do alternative tasks due to its flexible design.

14.4.1 Half-duplex Asynchronous Data Transfer

By utilizing the USI Data Register in Three-wire mode, it is possible to implement a more compact and higher performance UART than by software only.

14.4.2 4-bit Counter

The 4-bit counter can be used as a stand-alone counter with overflow interrupt. Note that if the counter is clocked externally, both clock edges will generate an increment.

14.4.3 12-bit Timer/Counter

Combining the USI 4-bit counter and Timer/Counter0 allows them to be used as a 12-bit counter.

14.4.4 Edge Triggered External Interrupt

By setting the counter to maximum value (F) it can function as an additional external interrupt. The Overflow Flag and Interrupt Enable bit are then used for the external interrupt. This feature is selected by the USICS1 bit.

14.4.5 Software Interrupt

The counter overflow interrupt can be used as a software interrupt triggered by a clock strobe.

14.5 Register Description

14.5.1 USIDR – USI Data Register

Bit	7	6	5	4	3	2	1	0	
(0xBA)	**USID7**	**USID6**	**USID5**	**USID4**	**USID3**	**USID2**	**USID1**	**USID0**	**USIDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – USID[7:0]: USI Data**

When accessing the USI Data Register (USIDR) the Serial Register can be accessed directly. If a serial clock occurs at the same cycle the register is written, the register will contain the value written and no shift is performed. A (left) shift operation is performed depending of the USICS[1:0] bits setting. The shift operation can be controlled by an external clock edge, by a Timer/Counter0 Compare Match, or directly by software using the USICLK strobe bit. Note that even when no wire mode is selected (USIWM[1:0] = 0) both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the USI Data Register.

The output pin in use, DO or SDA depending on the wire mode, is connected via the output latch to the most significant bit (bit 7) of the Data Register. The output latch is open (transparent) during the first half of a serial clock cycle when an external clock source is selected (USICS1 = 1), and constantly open when an internal clock source is used (USICS1 = 0). The output will be changed immediately when a new MSB written as long as the latch is open. The latch ensures that data input is sampled and data output is changed on opposite clock edges.

Note that the corresponding Data Direction Register to the pin must be set to one for enabling data output from the USI Data Register.

14.5.2 USIBR – USI Buffer Register

Bit	7	6	5	4	3	2	1	0	
(0xBB)	USIB7	USIB6	USIB5	USIB4	USIB3	USIB2	USIB1	USIB0	USIBR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – USID[7:0]: USI Buffer**

The content of the Serial Register is loaded to the USI Buffer Register when the transfer is completed, and instead of accessing the USI Data Register (the Serial Register) the USI Data Buffer can be accessed when the CPU reads the received data. This gives the CPU time to handle other program tasks too as the controlling of the USI is not so timing critical. The USI flags as set same as when reading the USIDR register.

14.5.3 USISR – USI Status Register

The Status Register contains Interrupt Flags, line Status Flags and the counter value.

Bit	7	6	5	4	3	2	1	0	
(0xB9)	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	USISR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – USISIF: Start Condition Interrupt Flag**

When Two-wire mode is selected, the USISIF Flag is set (to one) when a start condition is detected. When output disable mode or Three-wire mode is selected and (USICSx = 11_b & USICLK = 0) or (USICS = 10_b & USICLK = 0), any edge on the SCK pin sets the flag.

An interrupt will be generated when the flag is set while the USISIE bit in USICR and the Global Interrupt Enable Flag are set. The flag will only be cleared by writing a logical one to the USISIF bit. Clearing this bit will release the start detection hold of USCL in Two-wire mode.

A start condition interrupt will wakeup the processor from all sleep modes.

- **Bit 6 – USIOIF: Counter Overflow Interrupt Flag**

This flag is set (one) when the 4-bit counter overflows (i.e., at the transition from 15 to 0). An interrupt will be generated when the flag is set while the USIOIE bit in USICR and the Global Interrupt Enable Flag are set. The flag will only be cleared if a one is written to the USIOIF bit. Clearing this bit will release the counter overflow hold of SCL in Two-wire mode.

A counter overflow interrupt will wakeup the processor from Idle sleep mode.

- **Bit 5 – USIPF: Stop Condition Flag**

When Two-wire mode is selected, the USIPF Flag is set (one) when a stop condition is detected. The flag is cleared by writing a one to this bit. Note that this is not an Interrupt Flag. This signal is useful when implementing Two-wire bus master arbitration.

- **Bit 4 – USIDC: Data Output Collision**

This bit is logical one when bit 7 in the USI Data Register differs from the physical pin value. The flag is only valid when Two-wire mode is used. This signal is useful when implementing Two-wire bus master arbitration.

- **Bits 3:0 – USICNT[3:0]: Counter Value**

These bits reflect the current 4-bit counter value. The 4-bit counter value can directly be read or written by the CPU.

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a Timer/Counter0 Compare Match, or by software using USICLK or USITC strobe bits. The clock source depends of the setting of the USICS[1:0] bits. For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by write a one to the USICLK bit while setting an external clock source (USICS1 = 1).

Note that even when no wire mode is selected (USIWM[1:0] = 0) the external clock input (USCK/SCL) are can still be used by the counter.

14.5.4 USICR – USI Control Register

Bit	7	6	5	4	3	2	1	0	
(0xB8)	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	USICR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The Control Register includes interrupt enable control, wire mode setting, Clock Select setting, and clock strobe.

- **Bit 7 – USISIE: Start Condition Interrupt Enable**

Setting this bit to one enables the Start Condition detector interrupt. If there is a pending interrupt when the USISIE and the Global Interrupt Enable Flag is set to one, this will immediately be executed. Refer to the USISIF bit description on page 164 for further details.

- **Bit 6 – USIOIE: Counter Overflow Interrupt Enable**

Setting this bit to one enables the Counter Overflow interrupt. If there is a pending interrupt when the USIOIE and the Global Interrupt Enable Flag is set to one, this will immediately be executed. Refer to the USIOIF bit description on page 164 for further details.

- **Bits 5:4 – USIWM[1:0]: Wire Mode**

These bits set the type of wire mode to be used. Basically only the function of the outputs are affected by these bits. Data and clock inputs are not affected by the mode selected and will always have the same function. The counter and USI Data Register can therefore be clocked externally, and data input sampled, even when outputs are disabled. The relations between USIWM[1:0] and the USI operation is summarized in Table 14-1.

Table 14-1. Relations between USIWM[1:0] and the USI Operation

USIWM1	USIWM0	Description
0	0	Outputs, clock hold, and start detector disabled. Port pins operates as normal.
0	1	Three-wire mode. Uses DO, DI, and USCK pins. The *Data Output* (DO) pin overrides the corresponding bit in the PORT Register in this mode. However, the corresponding DDR bit still controls the data direction. When the port pin is set as input the pins pull-up is controlled by the PORT bit. The *Data Input* (DI) and *Serial Clock* (USCK) pins do not affect the normal port operation. When operating as master, clock pulses are software generated by toggling the PORT Register, while the data direction is set to output. The USITC bit in the USICR Register can be used for this purpose.
1	0	Two-wire mode. Uses SDA (DI) and SCL (USCK) pins(1). The *Serial Data* (SDA) and the *Serial Clock* (SCL) pins are bi-directional and uses open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDR Register. When the output driver is enabled for the SDA pin, the output driver will force the line SDA low if the output of the USI Data Register or the corresponding bit in the PORT Register is zero. Otherwise the SDA line will not be driven (i.e., it is released). When the SCL pin output driver is enabled the SCL line will be forced low if the corresponding bit in the PORT Register is zero, or by the start detector. Otherwise the SCL line will not be driven. The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the Start Condition Flag (USISIF) releases the line. The SDA and SCL pin inputs is not affected by enabling this mode. Pull-ups on the SDA and SCL port pin are disabled in Two-wire mode.
1	1	Two-wire mode. Uses SDA and SCL pins. Same operation as for the Two-wire mode described above, except that the SCL line is also held low when a counter overflow occurs, and is held low until the Counter Overflow Flag (USIOIF) is cleared.

Note: 1. The DI and USCK pins are renamed to *Serial Data* (SDA) and *Serial Clock* (SCL) respectively to avoid confusion between the modes of operation.

- **Bits 3:2 – USICS[1:0]: Clock Source Select**

These bits set the clock source for the USI Data Register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or Timer/Counter0 Compare Match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS[1:0] bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the USI Data Register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 14-2 on page 167 shows the relationship between the USICS[1:0] and USICLK setting and clock source used for the USI Data Register and the 4-bit counter.

Table 14-2. Relations between the USICS[1:0] and USICLK Setting

USICS1	USICS0	USICLK	USI Data Register Clock Source	4-bit Counter Clock Source
0	0	0	No Clock	No Clock
0	0	1	Software clock strobe (USICLK)	Software clock strobe (USICLK)
0	1	X	Timer/Counter0 Compare Match	Timer/Counter0 Compare Match
1	0	0	External, positive edge	External, both edges
1	1	0	External, negative edge	External, both edges
1	0	1	External, positive edge	Software clock strobe (USITC)
1	1	1	External, negative edge	Software clock strobe (USITC)

- **Bit 1 – USICLK: Clock Strobe**

Writing a one to this bit location strobes the USI Data Register to shift one step and the counter to increment by one, provided that the USICS[1:0] bits are set to zero and by doing so the software clock strobe option is selected. The output will change immediately when the clock strobe is executed, i.e., in the same instruction cycle. The value shifted into the USI Data Register is sampled the previous instruction cycle. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a Clock Select Register. Setting the USICLK bit in this case will select the USITC strobe bit as clock source for the 4-bit counter (see Table 14-2).

- **Bit 0 – USITC: Toggle Clock Port Pin**

Writing a one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the Data Direction Register, but if the PORT value is to be shown on the pin the DDB2 must be set as output (to one). This feature allows easy clock generation when implementing master devices. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1) and the USICLK bit is set to one, writing to the USITC strobe bit will directly clock the 4-bit counter. This allows an early detection of when the transfer is done when operating as a master device.

14.5.5 USIPP – USI Pin Position

Bit	7	6	5	4	3	2	1	0	
(0xBC)	–	–	–	–	–	–	–	USIPOS	USIPP
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny87/167 and always reads as zero.

- **Bit 0 – USIPOS: USI Pin Position**

Setting or clearing this bit changes the USI pin position.

Table 14-3. USI Pin Position

USIPOS		USI Pin Position	
0	PortB (Default)	DI, SDA	PB0 - (PCINT8/OC1AU)
		DO	PB1 - (PCINT9/OC1BU)
		USCK, SCL	PB2 - (PCINT10/OC1AV)
1	Port A (Alternate)	DI, SDA	PA4 - (PCINT4/ADC4/ICP1/MOSI)
		DO	PA2 - (PCINT2/ADC2/OC0A/MISO)
		USCK, SCL	PA5 - (PCINT5/ADC5/T1/SCK)

15. LIN / UART - Local Interconnect Network Controller or UART

The LIN (Local Interconnect Network) is a serial communications protocol which efficiently supports the control of mechatronics nodes in distributed automotive applications, but is equally suited for industrial applications. The main properties of the LIN bus are:

- **Single master with multiple slaves concept**
- **Low cost silicon implementation based on common UART/SCI interface**
- **Self synchronization with on-chip oscillator in slave node**
- **Deterministic signal transmission with signal propagation time computable in advance**
- **Low cost single-wire implementation**
- **Speed up to 20 Kbit/s.**

LIN provides a cost efficient bus communication where the bandwidth and versatility of CAN are not required. The specification of the line driver/receiver needs to match the ISO9141 NRZ-standard.

If LIN is not required, the controller alternatively can be programmed as Universal Asynchronous serial Receiver and Transmitter (UART).

15.1 LIN Features

- **Hardware Implementation of LIN 2.1 (LIN 1.3 Compatibility)**
- **Small, CPU Efficient and Independent Master/Slave Routines Based on “LIN Work Flow Concept” of LIN 2.1 Specification**
- **Automatic LIN Header Handling and Filtering of Irrelevant LIN Frames**
- **Automatic LIN Response Handling**
- **Extended LIN Error Detection and Signaling**
- **Hardware Frame Time-out Detection**
- **“Break-in-data” Support Capability**
- **Automatic Re-synchronization to Ensure Proper Frame Integrity**
- **Fully Flexible Extended Frames Support Capabilities**

15.2 UART Features

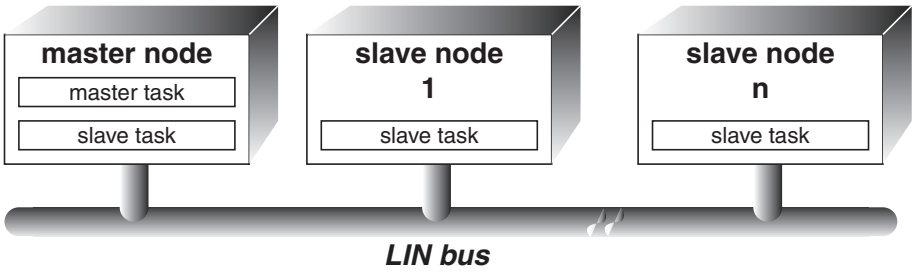
- **Full Duplex Operation (Independent Serial Receive and Transmit Processes)**
- **Asynchronous Operation**
- **High Resolution Baud Rate Generator**
- **Hardware Support of 8 Data Bits, Odd/Even/No Parity Bit, 1 Stop Bit Frames**
- **Data Over-Run and Framing Error Detection**

15.3 LIN Protocol

15.3.1 Master and Slave

A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only.

Figure 15-1. LIN cluster with one master node and “n” slave nodes



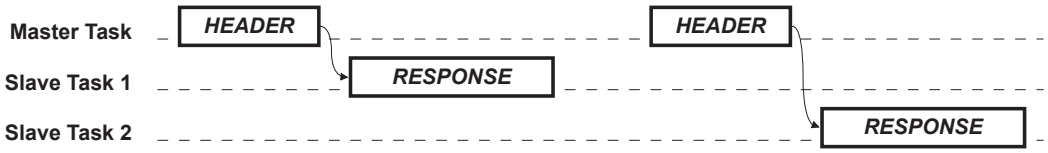
The master task decides when and which frame shall be transferred on the bus. The slave tasks provide the data transported by each frame. Both the master task and the slave task are parts of the Frame handler

15.3.2 Frames

A frame consists of a header (provided by the master task) and a response (provided by a slave task).

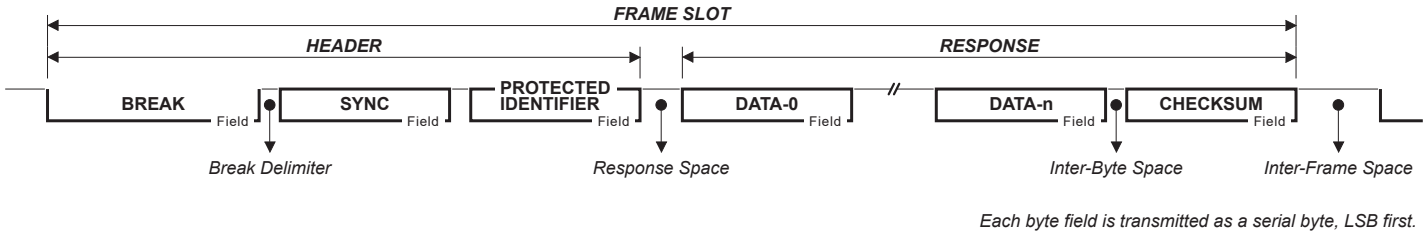
The header consists of a BREAK and SYNC pattern followed by a PROTECTED IDENTIFIER. The identifier uniquely defines the purpose of the frame. The slave task appointed for providing the response associated with the identifier transmits it. The response consists of a DATA field and a CHECKSUM field.

Figure 15-2. Master and slave tasks behavior in LIN frame



The slave tasks waiting for the data associated with the identifier receives the response and uses the data transported after verifying the checksum.

Figure 15-3. Structure of a LIN frame



Each byte field is transmitted as a serial byte, LSB first.

15.3.3 Data Transport

Two types of data may be transported in a frame; signals or diagnostic messages.

- Signals
  Signals are scalar values or byte arrays that are packed into the data field of a frame. A signal is always present at the same position in the data field for all frames with the same identifier.
- Diagnostic messages
  Diagnostic messages are transported in frames with two reserved identifiers. The interpretation of the data field depends on the data field itself as well as the state of the communicating nodes.

15.3.4 Schedule Table

The master task (in the master node) transmits frame headers based on a schedule table. The schedule table specifies the identifiers for each header and the interval between the start of a frame and the start of the following frame. The master application may use different schedule tables and select among them.

15.3.5 Compatibility with LIN 1.3

LIN 2.1 is a super-set of LIN 1.3.

A LIN 2.1 master node can handle clusters consisting of both LIN 1.3 slaves and/or LIN 2.1 slaves. The master will then avoid requesting the new LIN 2.1 features from a LIN 1.3 slave:

- Enhanced checksum,
- Re-configuration and diagnostics,
- Automatic baud rate detection,
- "Response error" status monitoring.

LIN 2.1 slave nodes can not operate with a LIN 1.3 master node (e.g. the LIN1.3 master does not support the enhanced checksum).

The LIN 2.1 physical layer is backwards compatible with the LIN1.3 physical layer. But not the other way around. The LIN 2.1 physical layer sets greater requirements, i.e. a master node using the LIN 2.1 physical layer can operate in a LIN 1.3 cluster.

15.4 LIN / UART Controller

The LIN/UART controller is divided in three main functions:

- Tx LIN Header function,
- Rx LIN Header function,
- LIN Response function.

These functions mainly use two services:

- Rx service,
- Tx service.

Because these two services are basically UART services, the controller is also able to switch into an UART function.

15.4.1 LIN Overview

The LIN/UART controller is designed to match as closely as possible to the LIN software application structure. The LIN software application is developed as independent tasks, several slave tasks and one master task (c.f. Section 15.3.4 on page 171). The ATtiny87/167 conforms to this perspective. The only link between the master task and the slave task will be at the cross-over point where the interrupt routine is called once a new identifier is available. Thus, in a master node, housing both master and slave task, the Tx LIN Header function will alert the slave task of an identifier presence. In the same way, in a slave node, the Rx LIN Header function will alert the slave task of an identifier presence.

When the slave task is warned of an identifier presence, it has first to analyze it to know what to do with the response. Hardware flags identify the presence of one of the specific identifiers from 60 (0x3C) up to 63 (0x3F).

For LIN communication, only four interrupts need to be managed:

- LIDOK: New LIN identifier available,
- LRXOK: LIN response received,
- LTXOK: LIN response transmitted,
- LERR: LIN Error(s).

The wake-up management can be automated using the UART wake-up capability and a node sending a minimum of 5 low bits (0xF0) for LIN 2.1 and 8 low bits (0x80) for LIN 1.3. Pin change interrupt on LIN wake-up signal can be also used to exit the device of one of its sleep modes.

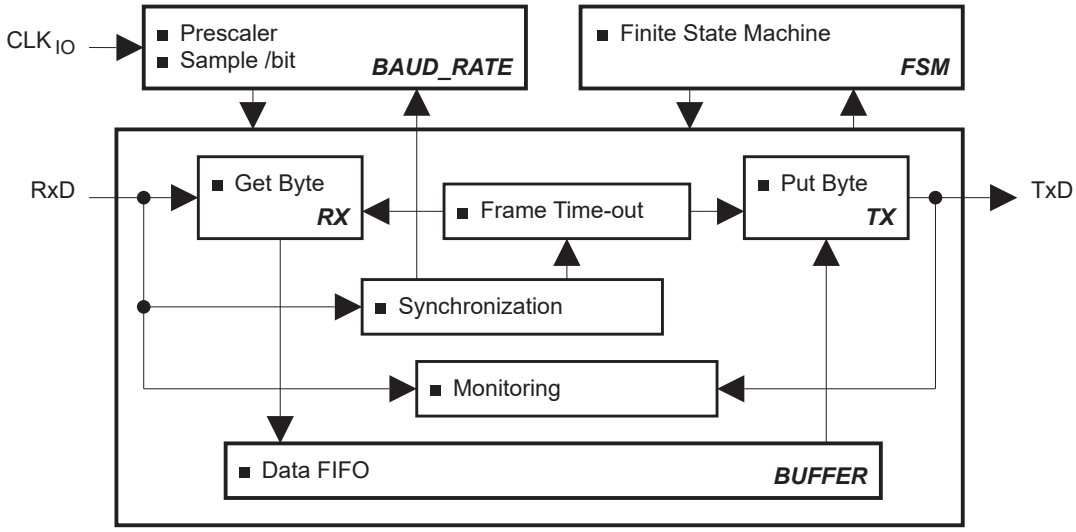
Extended frame identifiers 62 (0x3E) and 63 (0x3F) are reserved to allow the embedding of user-defined message formats and future LIN formats. The byte transfer mode offered by the UART will ensure the upwards compatibility of LIN slaves with accommodation of the LIN protocol.

15.4.2 UART Overview

The LIN/UART controller can also function as a conventional UART. By default, the UART operates as a full duplex controller. It has local loop back circuitry for test purposes. The UART has the ability to buffer one character for transmit and two for receive. The receive buffer is made of one 8-bit serial register followed by one 8-bit independent buffer register. Automatic flag management is implemented when the application puts or gets characters, thus reducing the software overhead. Because transmit and receive services are independent, the user can save one device pin when one of the two services is not used. The UART has an enhanced baud rate generator providing a maximum error of 2% whatever the clock frequency and the targeted baud rate.

15.4.3 LIN/UART Controller Structure

Figure 15-4. LIN/UART Controller Block Diagram



15.4.4 LIN/UART Command Overview

Figure 15-5. LIN/UART Command Dependencies

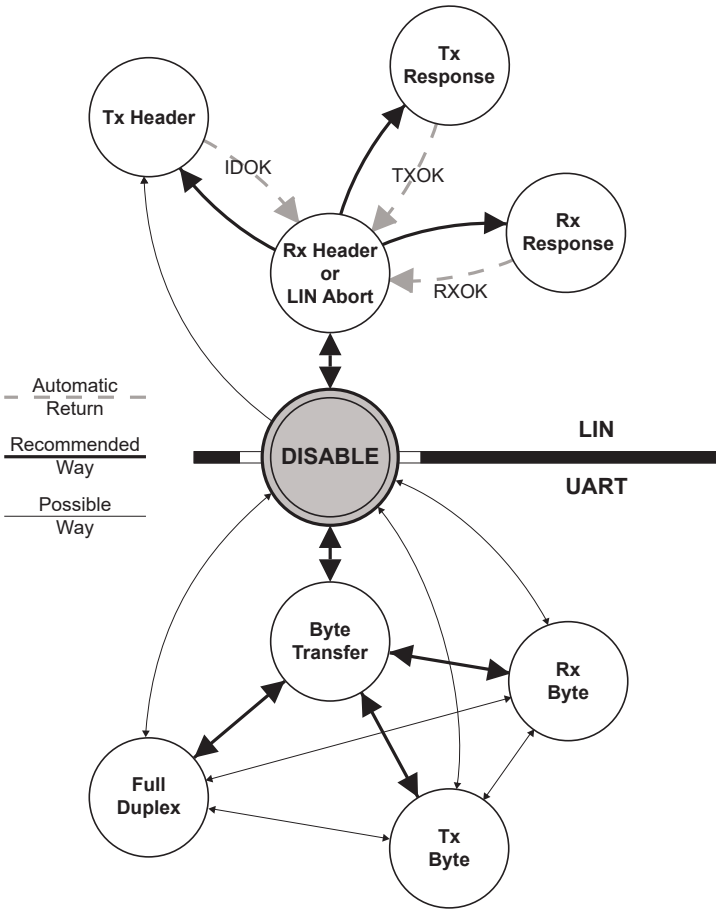


Table 15-1. LIN/UART Command List

LENA	LCMD[2]	LCMD[1]	LCMD[0]	Command	Comment
0	x	x	x	Disable peripheral	
1	0	0	0	Rx Header - LIN Abort	LIN Withdrawal
			1	Tx Header	LCMD[2:0]=000 after Tx
		1	0	Rx Response	LCMD[2:0]=000 after Rx
			1	Tx Response	LCMD[2:0]=000 after Tx
	1	0	0	Byte transfer	no CRC, no Time out LTXDL=LRXDL=0 (LINDLR: read only register)
		1	0	Rx Byte	
		0	1	Tx Byte	
		1	1	Full duplex	

15.4.5 Enable / Disable

Setting the LENA bit in LINCR register enables the LIN/UART controller. To disable the LIN/UART controller, LENA bit must be written to 0. No wait states are implemented, so, the disable command is taken into account immediately.

15.4.6 LIN Commands

Clearing the LCMD[2] bit in LINCR register enables LIN commands.

As shown in Table 15-1 on page 174, four functions controlled by the LCMD[1:0] bits of LINCR register are available (c.f. Figure 15-5 on page 173).

15.4.6.1 Rx Header / LIN Abort Function

This function (or state) is mainly the withdrawal mode of the controller.

When the controller has to execute a master task, this state is the start point before enabling a Tx Header command.

When the controller has only to execute slave tasks, LIN header detection/acquisition is enabled as background function. At the end of such an acquisition (Rx Header function), automatically the appropriate flags are set, and in LIN 1.3, the LINDLR register is set with the uncoded length value.

This state is also the start point before enabling the Tx or the Rx Response command.

A running function (i.e. Tx Header, Tx or Rx Response) can be aborted by clearing LCMD[1:0] bits in LINCR register (See ”Break-in-data” on page 184.). In this case, an abort flag - LABORT - in LINERR register will be set to inform the other software tasks. No wait states are implemented, so, the abort command is taken into account immediately.

Rx Header function is responsible for:

- The BREAK field detection,
- The hardware re-synchronization analyzing the SYNCH field,
- The reception of the PROTECTED IDENTIFIER field, the parity control and the update of the LINDLR register in case of LIN 1.3,
- The starting of the Frame_Time_Out,
- The checking of the LIN communication integrity.

15.4.6.2 Tx Header Function

In accordance with the LIN protocol, only the master task must enable this function. The header is sent in the appropriate timed slots at the programmed baud rate (c.f. LINBRR & LINBTR registers).

The controller is responsible for:

- The transmission of the BREAK field - 13 dominant bits,
- The transmission of the SYNCH field - character 0x55,
- The transmission of the PROTECTED IDENTIFIER field. It is the full content of the LINIDR register (automatic check bits included).

At the end of this transmission, the controller automatically returns to *Rx Header / LIN Abort* state (i.e. LCMD[1:0] = 00) after setting the appropriate flags. This function leaves the controller in the same setting as after the *Rx Header* function. This means that, in LIN 1.3, the LINDLR register is set with the uncoded length value at the end of the *Tx Header* function.

During this function, the controller is also responsible for:

- The starting of the Frame_Time_Out,
- The checking of the LIN communication integrity.

15.4.6.3 Rx & TX Response Functions

These functions are initiated by the slave task of a LIN node. They must be used after sending an header (master task) or after receiving an header (considered as belonging to the slave task). When the TX Response order is sent, the transmission begins. A Rx Response order can be sent up to the reception of the last serial bit of the first byte (before the stop-bit).

In LIN 1.3, the header slot configures the LINDLR register. In LIN 2.1, the user must configure the LINDLR register, either LRXDL[3:0] for *Rx Response* either LTXDL[3:0] for *Tx Response*.

When the command starts, the controller checks the LIN13 bit of the LINCR register to apply the right rule for computing the checksum. Checksum calculation over the DATA bytes and the PROTECTED IDENTIFIER byte is called enhanced checksum and it is used for communication with LIN 2.1 slaves. Checksum calculation over the DATA bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Note that identifiers 60 (0x3C) to 63 (0x3F) shall always use classic checksum.

At the end of this reception or transmission, the controller automatically returns to *Rx Header / LIN Abort* state (i.e. LCMD[1:0] = 00) after setting the appropriate flags.

If an LIN error occurs, the reception or the transmission is stopped, the appropriate flags are set and the LIN bus is left to recessive state.

During these functions, the controller is responsible for:

- The initialization of the checksum operator,
- The transmission or the reception of ‘n’ data with the update of the checksum calculation,
- The transmission or the checking of the CHECKSUM field,
- The checking of the Frame_Time_Out,
- The checking of the LIN communication integrity.

While the controller is sending or receiving a response, BREAK and SYNCH fields can be detected and the identifier of this new header will be recorded. Of course, specific errors on the previous response will be maintained with this identifier reception.

15.4.6.4 *Handling Data of LIN response*

A FIFO data buffer is used for data of the LIN response. After setting all parameters in the LINSEL register, repeated accesses to the LINDAT register perform data read or data write (c.f. “Data Management” on page 186).

Note that LRXDL[3:0] and LTXDL[3:0] are not linked to the data access.

15.4.7 UART Commands

Setting the LCMD[2] bit in LINENR register enables UART commands.

Tx Byte and Rx Byte services are independent as shown in Table 15-1 on page 174.

- Byte Transfer: the UART is selected but both Rx and Tx services are disabled,
- Rx Byte: only the Rx service is enable but Tx service is disabled,
- Tx Byte: only the Tx service is enable but Rx service is disabled,
- Full Duplex: the UART is selected and both Rx and Tx services are enabled.

This combination of services is controlled by the LCMD[1:0] bits of LINENR register (c.f. Figure 15-5 on page 173).

15.4.7.1 *Data Handling*

The FIFO used for LIN communication is disabled during UART accesses. LRXDL[3:0] and LTXDL[3:0] values of LINDLR register are then irrelevant. LINDAT register is then used as data register and LINSEL register is not relevant.

15.4.7.2 *Rx Service*

Once this service is enabled, the user is warned of an in-coming character by the LRXOK flag of LINSIR register. Reading LINDAT register automatically clears the flag and makes free the second stage of the buffer. If the user considers that the in-coming character is irrelevant without reading it, he directly can clear the flag (see specific flag management described in Section 15.6.2 on page 189).

The intrinsic structure of the Rx service offers a 2-byte buffer. The fist one is used for serial to parallel conversion, the second one receives the result of the conversion. This second buffer byte is reached reading LINDAT register. If the 2-byte buffer is full, a new in-coming character will overwrite the second one already recorded. An OVRERR error in LINERR register will then accompany this character when read.

A FERR error in LINERR register will be set in case of framing error.

15.4.7.3 *Tx Service*

If this service is enabled, the user sends a character by writing in LINDAT register. Automatically the LTXOK flag of LINSIR register is cleared. It will rise at the end of the serial transmission. If no new character has to be sent, LTXOK flag can be cleared separately (see specific flag management described in Section 15.6.2 on page 189).

There is no transmit buffering.

No error is detected by this service.

15.5 LIN / UART Description

15.5.1 Reset

The AVR core reset logic signal also resets the LIN/UART controller. Another form of reset exists, a software reset controlled by LSWRES bit in LINCR register. This self-reset bit performs a partial reset as shown in Table 15-2.

Table 15-2. Reset of LIN/UART Registers

Register	Name	Reset Value	LSWRES Value	Comment
LIN Control Reg.	LINCR	0000 0000 $_b$	0000 0000 $_b$	
LIN Status & Interrupt Reg.	LINSIR	0000 0000 $_b$	0000 0000 $_b$	
LIN Enable Interrupt Reg.	LINENIR	0000 0000 $_b$	xxxx 0000 $_b$	
LIN Error Reg.	LINERR	0000 0000 $_b$	0000 0000 $_b$	
LIN Bit Timing Reg.	LINBTR	0010 0000 $_b$	0010 0000 $_b$	x=unknown
LIN Baud Rate Reg. Low	LINBRRL	0000 0000 $_b$	uuuu uuuu $_b$	
LIN Baud Rate Reg. High	LINBRRH	0000 0000 $_b$	xxxx uuuu $_b$	u=unchanged
LIN Data Length Reg.	LINDLR	0000 0000 $_b$	0000 0000 $_b$	
LIN Identifier Reg.	LINIDR	1000 0000 $_b$	1000 0000 $_b$	
LIN Data Buffer Selection	LINSEL	0000 0000 $_b$	xxxx 0000 $_b$	
LIN Data	LINDAT	0000 0000 $_b$	0000 0000 $_b$	

15.5.2 Clock

The I/O clock signal ($clk_{i/o}$) also clocks the LIN/UART controller. It is its unique clock.

15.5.3 LIN Protocol Selection

LIN13 bit in LINCR register is used to select the LIN protocol:

- LIN13 = 0 (default): LIN 2.1 protocol,
- LIN13 = 1: LIN 1.3 protocol.

The controller checks the LIN13 bit in computing the checksum (enhanced checksum in LIN2.1 / classic checksum in LIN 1.3).

This bit is irrelevant for UART commands.

15.5.4 Configuration

Depending on the mode (LIN or UART), LCONF[1:0] bits of the LINCR register set the controller in the following configuration (Table 15-3):

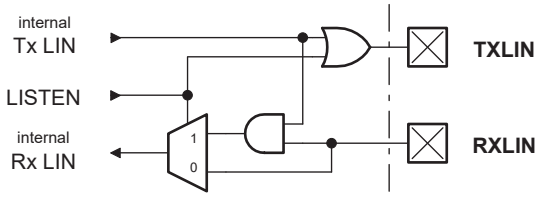
Table 15-3. Configuration Table versus Mode

Mode	LCONF[1:0]	Configuration
LIN	00_b	LIN standard configuration (default)
	01_b	No CRC field detection or transmission
	10_b	Frame_Time_Out disable
	11_b	Listening mode
UART	00_b	8-bit data, no parity & 1 stop-bit
	01_b	8-bit data, even parity & 1 stop-bit
	10_b	8-bit data, odd parity & 1 stop-bit
	11_b	Listening mode, 8-bit data, no parity & 1 stop-bit

The LIN configuration is independent of the programmed LIN protocol.

The listening mode connects the internal Tx LIN and the internal Rx LIN together. In this mode, the TXLIN output pin is disabled and the RXLIN input pin is always enabled. The same scheme is available in UART mode.

Figure 15-6. Listening Mode

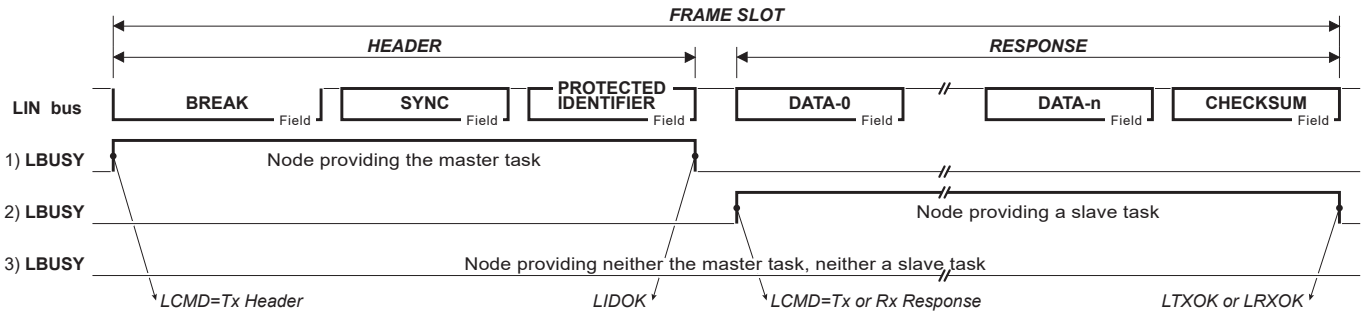


15.5.5 Busy Signal

LBUSY bit flag in LINSIR register is the image of the BUSY signal. It is set and cleared by hardware. It signals that the controller is busy with LIN or UART communication.

15.5.5.1 Busy Signal in LIN Mode

Figure 15-7. Busy Signal in LIN Mode



When the busy signal is set, some registers are locked, user writing is not allowed:

- “LIN Control Register” - LINCR - except LCMD[2:0], LENA & LSWRES,
- “LIN Baud Rate Registers” - LINBRRL & LINBRRH,
- “LIN Data Length Register” - LINDLR,
- “LIN Identifier Register” - LINIDR,
- “LIN Data Register” - LINDAT.

If the busy signal is set, the only available commands are:

- LCMD[1:0] = 00_b, the abort command is taken into account at the end of the byte,
- LENA = 0 and/or LCMD[2] = 0, the kill command is taken into account immediately,
- LSWRES = 1, the reset command is taken into account immediately.

Note that, if another command is entered during busy signal, the new command is not validated and the LOVRERR bit flag of the LINERR register is set. The on-going transfer is not interrupted.

15.5.5.2 Busy Signal in UART Mode

During the byte transmission, the busy signal is set. This locks some registers from being written:

- “LIN Control Register” - LINCR - except LCMD[2:0], LENA & LSWRES,
- “LIN Data Register” - LINDAT.

The busy signal is not generated during a byte reception.

15.5.6 Bit Timing

15.5.6.1 Baud rate Generator

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

- BAUD: Baud rate (in bps),
- $fclk_{i/o}$: System I/O clock frequency,
- LDIV[11:0]: Contents of LINBRRH & LINBRRL registers - (0-4095), the pre-scaler receives $clk_{i/o}$ as input clock.
- LBT[5:0]: Least significant bits of - LINBTR register- (0-63) is the number of samplings in a LIN or UART bit (default value 32).

Equation for calculating baud rate:

$$BAUD = fclk_{i/o} / (LBT[5:0] \times (LDIV[11:0] + 1))$$

Equation for setting LINDIV value:

$$LDIV[11:0] = (fclk_{i/o} / (LBT[5:0] \times BAUD)) - 1$$

Note that in reception a majority vote on three samplings is made.

15.5.6.2 Re-synchronization in LIN Mode

When waiting for Rx Header, LBT[5:0] = 32 in LINBTR register. The re-synchronization begins when the BREAK is detected. If the BREAK size is not in the range (10.5 bits min., 28 bits max. — 13 bits nominal), the BREAK is refused. The re-synchronization is done by adjusting LBT[5:0] value to the SYNCH field of the received header (0x55). Then the PROTECTED IDENTIFIER is sampled using the new value of LBT[5:0].

The re-synchronization implemented in the controller tolerates a clock deviation of ± 20% and adjusts the baud rate in a ± 2% range.

The new LBT[5:0] value will be used up to the end of the response. Then, the LBT[5:0] will be reset to 32 for the next header.

The LINBTR register can be used to (software) re-calibrate the clock oscillator.

The re-synchronization is not performed if the LIN node is enabled as a master.

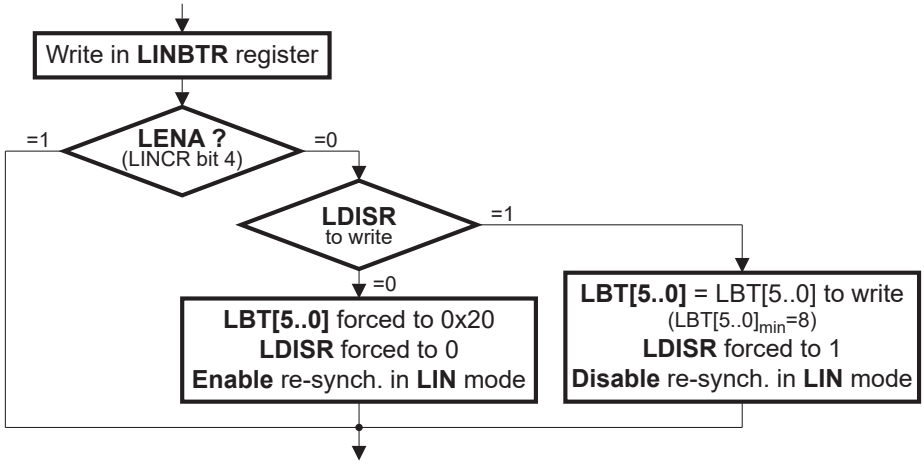
15.5.6.3 Handling LBT[5:0]

LDISR bit of LINBTR register is used to:

- Disable the re-synchronization (for instance in the case of LIN MASTER node),
- To enable the setting of LBT[5:0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of 8 is required for LBT[5:0] due to the sampling operation.

Note that the LENA bit of LINCR register is important for this handling (see Figure 15-8 on page 180).

Figure 15-8. Handling LBT[5:0]



15.5.7 Data Length

Section 15.4.6 “LIN Commands” on page 174 describes how to set or how are automatically set the LRXDL[3:0] or LTXDL[3:0] fields of LINDLR register before receiving or transmitting a response.

In the case of Tx Response the LRXDL[3:0] will be used by the hardware to count the number of bytes already successfully sent.

In the case of Rx Response the LTXDL[3:0] will be used by the hardware to count the number of bytes already successfully received.

If an error occurs, this information is useful to the programmer to recover the LIN messages.

15.5.7.1 Data Length in LIN 2.1

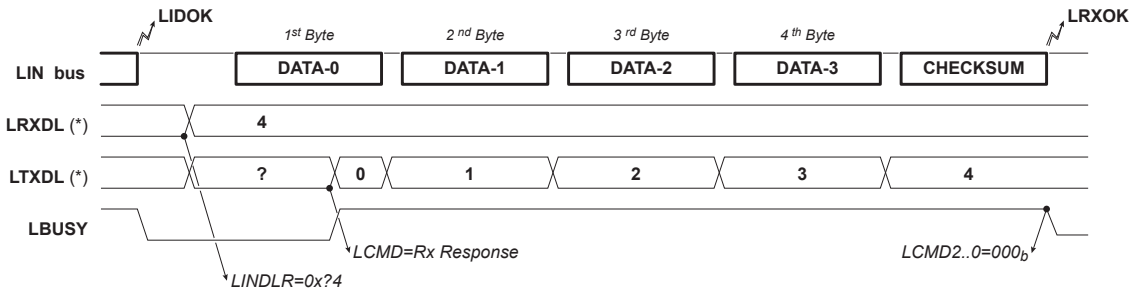
- If LTXDL[3:0]=0 only the CHECKSUM will be sent,
- If LRXDL[3:0]=0 the first byte received will be interpreted as the CHECKSUM,
- If LTXDL[3:0] or LRXDL[3:0] >8, values will be forced to 8 after the command setting and before sending or receiving of the first byte.

15.5.7.2 Data Length in LIN 1.3

- LRXDL and LTXDL fields are both hardware updated before setting LIDOK by decoding the data length code contained in the received PROTECTED IDENTIFIER (LRXDL = LTXDL).
- Via the above mechanism, a length of 0 or >8 is not possible.

15.5.7.3 Data Length in Rx Response

Figure 15-9. LIN2.1 - Rx Response - No error

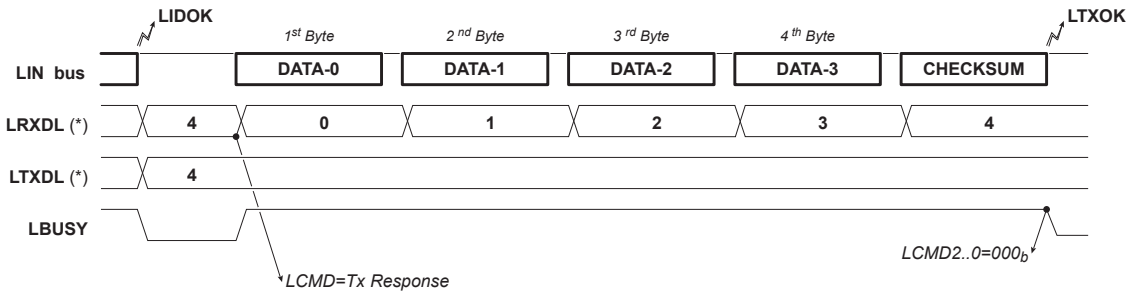


(*) : LRXDL & LTXDL updated by user

- The user initializes LRXDL field before setting the Rx Response command,
- After setting the Rx Response command, LTXDL is reset by hardware,
- LRXDL field will remain unchanged during Rx (during busy signal),
- LTXDL field will count the number of received bytes (during busy signal),
- If an error occurs, Rx stops, the corresponding error flag is set and LTXDL will give the number of received bytes without error,
- If no error occurs, LRXOK is set after the reception of the CHECKSUM, LRXDL will be unchanged (and LTXDL = LRXDL).

15.5.7.4 Data Length in Tx Response

Figure 15-10. LIN1.3 - Tx Response - No error



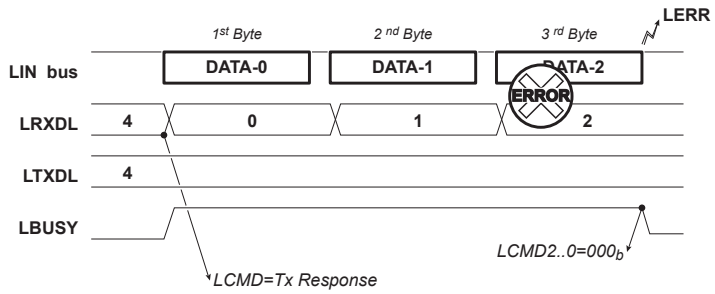
(*) : LRXDL & LTXDL updated by Rx Response or Tx Response task

- The user initializes LTXDL field before setting the Tx Response command,
- After setting the Tx Response command, LRXDL is reset by hardware,
- LTXDL will remain unchanged during Tx (during busy signal),
- LRXDL will count the number of transmitted bytes (during busy signal),

- If an error occurs, Tx stops, the corresponding error flag is set and LRXDL will give the number of transmitted bytes without error,
- If no error occurs, LTXOK is set after the transmission of the CHECKSUM, LTXDL will be unchanged (and LRXDL = LTXDL).

15.5.7.5 Data Length after Error

Figure 15-11. Tx Response - Error



Note: Information on response (ex: error on byte) is only available at the end of the serialization/de-serialization of the byte.

15.5.7.6 Data Length in UART Mode

- The UART mode forces LRXDL and LTXDL to 0 and disables the writing in LINDLR register,
- Note that after reset, LRXDL and LTXDL are also forced to 0.

15.5.8 xxOK Flags

There are three xxOK flags in LINSIR register:

- LIDOK: LIN IDentifier OK
  It is set at the end of the header, either by the Tx Header function or by the Rx Header. In LIN 1.3, before generating LIDOK, the controller updates the LRXDL & LTXDL fields in LINDLR register.
  It is not driven in UART mode.
- LRXOK: LIN RX response complete
  It is set at the end of the response by the Rx Response function in LIN mode and once a character is received in UART mode.
- LTXOK: LIN TX response complete
  It is set at the end of the response by the Tx Response function in LIN mode and once a character has been sent in UART mode.

These flags can generate interrupts if the corresponding enable interrupt bit is set in the LINENIR register (see Section 15.5.13 “Interrupts” on page 185).

15.5.9 xxERR Flags

LERR bit of the LINSIR register is an logical ‘OR’ of all the bits of LINERR register (see Section 15.5.13 “Interrupts” on page 185). There are eight flags:

- LBERR = LIN Bit ERRor.
  A unit that is sending a bit on the bus also monitors the bus. A LIN bit error will be flagged when the bit value that is monitored is different from the bit value that is sent. After detection of a LIN bit error the transmission is aborted.
- LCERR = LIN Checksum ERRor.
  A LIN checksum error will be flagged if the inverted modulo-256 sum of all received data bytes (and the protected identifier in LIN 2.1) added to the checksum does not result in 0xFF.
- LPERR = LIN Parity ERRor (identifier).
  A LIN parity error in the IDENTIFIER field will be flagged if the value of the parity bits does not match with the identifier value. (See LP[1:0] bits in Section 15.6.8 “LINIDR – LIN Identifier Register” on page 193). A LIN slave application does not distinguish between corrupted parity bits and a corrupted identifier. The hardware does not undertake any correction. However, the LIN slave application has to solve this as:
  - known identifier (parity bits corrupted),
  - or corrupted identifier to be ignored,
  - or new identifier.
- LSERR = LIN Synchronization ERRor.
  A LIN synchronization error will be flagged if a slave detects the edges of the SYNCH field outside the given tolerance.
- LFERR = LIN Framing ERRor.
  A framing error will be flagged if dominant STOP bit is sampled.
  Same function in UART mode.
- LTOERR = LIN Time Out ERRor.
  A time-out error will be flagged if the MESSAGE frame is not fully completed within the maximum length $T_{Frame_Maximum}$ by any slave task upon transmission of the SYNCH and IDENTIFIER fields (see Section 15.5.10 “Frame Time Out” on page 184).
- LOVERR = LIN OVerrun ERRor.
  Overrun error will be flagged if a new command (other than LIN Abort) is entered while ‘Busy signal’ is present.
  In UART mode, an overrun error will be flagged if a received byte overwrites the byte stored in the serial input buffer.
- LABORT
  LIN abort transfer reflects a previous *LIN Abort* command (LCMD[2:0] = 000) while ‘Busy signal’ is present.

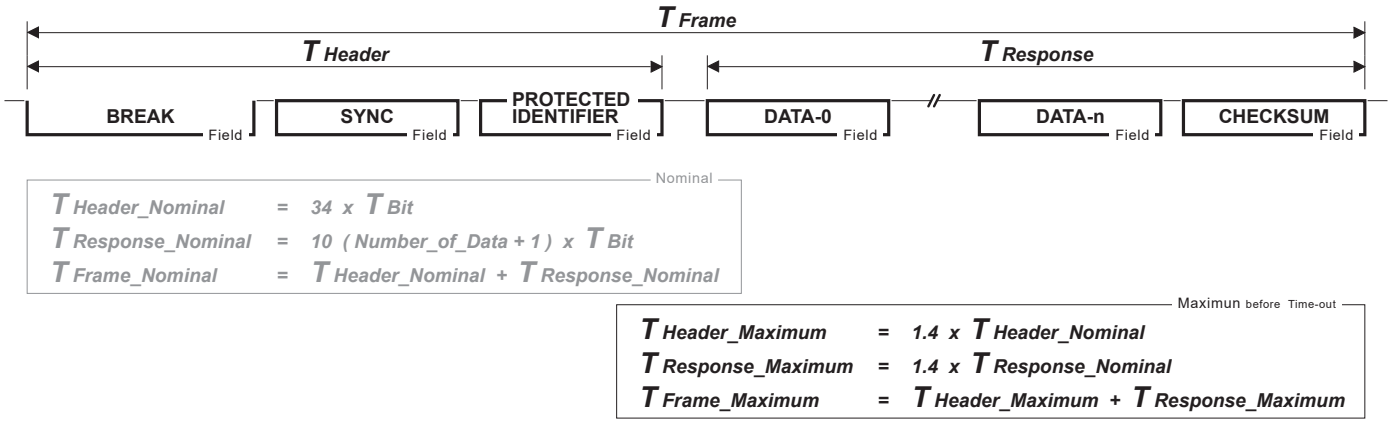
After each LIN error, the LIN controller stops its previous activity and returns to its withdrawal mode (LCMD[2:0] = 000_b) as illustrated in Figure 15-11 on page 182.

Writing 1 in LERR of LINSIR register resets LERR bit and all the bits of the LINERR register.

15.5.10 Frame Time Out

According to the LIN protocol, a frame time-out error is flagged if: $T_{Frame} > T_{Frame_Maximum}$. This feature is implemented in the LIN/UART controller.

Figure 15-12. LIN timing and frame time-out



15.5.11 Break-in-data

According to the LIN protocol, the LIN/UART controller can detect the BREAK/SYNC field sequence even if the break is partially superimposed with a byte of the response. When a BREAK/SYNC field sequence happens, the transfer in progress is aborted and the processing of the new frame starts.

- On slave node(s), an error is generated (i.e. LBERR in case of *Tx Response* or LFERR in case of *Rx Response*). Information on data error is also available, refer to the Section 15.5.7.5.
- On master node, the user (code) is responsible for this aborting of frame. To do this, the master task has first to abort the on-going communication (clearing LCMD bits - *LIN Abort* command) and then to apply the *Tx Header* command. In this case, the abort error flag - LABORT - is set.

On the slave node, the BREAK detection is processed with the synchronization setting available when the LIN/UART controller processed the (aborted) response. But the re-synchronization restarts as usual. Due to a possible difference of timing reference between the BREAK field and the rest of the frame, the time-out values can be slightly inaccurate.

15.5.12 Checksum

The last field of a frame is the checksum.

In LIN 2.1, the checksum contains the inverted eight bit sum with carry over all data bytes and the protected identifier. This calculation is called enhanced checksum.

$$\text{CHECKSUM} = 255 - \left(\text{unsigned char} \left(\left(\sum_{0}^{n} \text{DATA}_n \right) + \text{PROTECTED ID.} \right) + \text{unsigned char} \left(\left(\left(\sum_{0}^{n} \text{DATA}_n \right) + \text{PROTECTED ID.} \right) \gg 8 \right) \right)$$

In LIN 1.3, the checksum contains the inverted eight bit sum with carry over all data bytes. This calculation is called classic checksum.

$$CHECKSUM = 255 - \left(\text{unsigned char}\left(\sum_{0}^{n} DATA_n \right) + \text{unsigned char}\left(\left(\sum_{0}^{n} DATA_n \right) \gg 8 \right) \right)$$

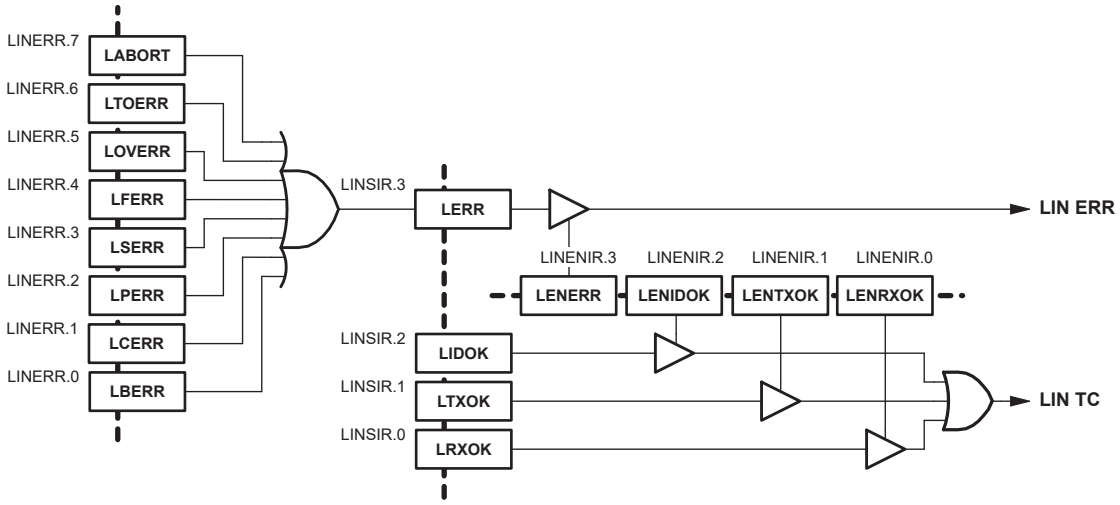
Frame identifiers 60 (0x3C) to 61 (0x3D) shall always use classic checksum.

15.5.13 Interrupts

As shown in Figure 15-13 on page 185, the four communication flags of the LINSIR register are combined to drive two interrupts. Each of these flags have their respective enable interrupt bit in LINENIR register.

(see Section 15.5.8 “xxOK Flags” on page 182 and Section 15.5.9 “xxERR Flags” on page 183).

Figure 15-13. LIN Interrupt Mapping



15.5.14 Message Filtering

Message filtering based upon the whole identifier is not implemented. Only a status for frame headers having 0x3C, 0x3D, 0x3E and 0x3F as identifier is available in the LINSIR register.

Table 15-4. Frame Status

LIDST[2:0]	Frame Status
$0xx_b$	No specific identifier
100_b	60 (0x3C) identifier
101_b	61 (0x3D) identifier
110_b	62 (0x3E) identifier
111_b	63 (0x3F) identifier

The LIN protocol says that a message with an identifier from 60 (0x3C) up to 63 (0x3F) uses a classic checksum (sum over the data bytes only). Software will be responsible for switching correctly the LIN13 bit to provide/check this expected checksum (the insertion of the ID field in the computation of the CRC is set - or not - just after entering the Rx or Tx Response command).

15.5.15 Data Management

15.5.15.1 LIN FIFO Data Buffer

To preserve register allocation, the LIN data buffer is seen as a FIFO (with address pointer accessible). This FIFO is accessed via the LINDX[2:0] field of LINSEL register through the LINDAT register.

LINDX[2:0], the data index, is the address pointer to the required data byte. The data byte can be read or written. The data index is automatically incremented after each LINDAT access if the LAINC (active low) bit is cleared. A roll-over is implemented, after data index=7 it is data index=0. Otherwise, if LAINC bit is set, the data index needs to be written (updated) before each LINDAT access.

The first byte of a LIN frame is stored at the data index=0, the second one at the data index=1, and so on. Nevertheless, LINSEL must be initialized by the user before use.

15.5.15.2 UART Data Register

The LINDAT register is the data register (no buffering - no FIFO). In write access, LINDAT will be for data out and in read access, LINDAT will be for data in.

In UART mode the LINSEL register is unused.

15.5.16 OCD Support

When a debugger break occurs, the state machine of the LIN/UART controller is stopped (included frame time-out) and further communication may be corrupted.

15.6 Register Description

Table 15-5. LIN/UART Register Bits Summary

Name	Bit 7		Bit 6		Bit 5		Bit 4		Bit 3		Bit 2		Bit 1		Bit 0	
LINCR	LSWRES		LIN13		LCONF1		LCONF0		LENA		LCMD2		LCMD1		LCMD0	
	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W
LINSIR	LIDST2		LIDST1		LIDST0		LBUSY		LERR		LIDOK		LTXOK		LRXOK	
	0	R	0	R	0	R	0	R	0	R/W_{one}	0	R/W_{one}	0	R/W_{one}	0	R/W_{one}
LINENIR	—		—		—		—		LENERR		LENIDOK		LENTXOK		LENRXOK	
	0	R	0	R	0	R	0	R	0	R/W	0	R/W	0	R/W	0	R/W
LINERR	LABORT		LTOERR		LOVERR		LFERR		LSERR		LPERR		LCERR		LBERR	
	0	R	0	R	0	R	0	R	0	R	0	R	0	R	0	R
LINBTR	LDISR				LBT5		LBT4		LBT3		LBT2		LBT1		LBT0	
	0	R/W	0	R	1	R/(W)	0	R/(W)	0	R/(W)	0	R/(W)	0	R/(W)	0	R/(W)
LINBRRL	LDIV7		LDIV6		LDIV5		LDIV4		LDIV3		LDIV2		LDIV1		LDIV0	
	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W
LINBRRH	—		—		—		—		LDIV11		LDIV10		LDIV9		LDIV8	
	0	R	0	R	0	R	0	R	0	R/W	0	R/W	0	R/W	0	R/W
LINDLR	LTXDL3		LTXDL2		LTXDL1		LTXDL0		LRXDL3		LRXDL2		LRXDL1		LRXDL0	
	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W
LINIDR	LP1		LP0		LID5/LDL1		LID4/LDL0		LID3		LID2		LID1		LID0	
	1	R	0	R	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W
LINSEL	—		—		—		—		LAINC		LINDX2		LINDX1		LINDX0	
	0	R	0	R	0	R	0	R	0	R/W	0	R/W	0	R/W	0	R/W
LINDAT	LDATA7		LDATA6		LDATA5		LDATA4		LDATA3		LDATA2		LDATA1		LDATA0	
	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W	0	R/W

15.6.1 LINCR – LIN Control Register

Bit	7	6	5	4	3	2	1	0	
(0xC8)	LSWRES	LIN13	LCONF1	LCONF0	LENA	LCMD2	LCMD1	LCMD0	LINCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – LSWRES: Software Reset**
  - 0 = No action,
  - 1 = Software reset (this bit is self-reset at the end of the reset procedure).

- **Bit 6 – LIN13: LIN 1.3 mode**
  - 0 = LIN 2.1 (default),
  - 1 = LIN 1.3.

- **Bits 5:4 – LCONF[1:0]: Configuration**

  a. LIN mode (default = 00):
  - 00 = LIN Standard configuration (listen mode “off”, CRC “on” & Frame_Time_Out “on”,
  - 01 = No CRC, No Frame_Time_Out (listen mode “off”),
  - 10 = No Frame_Time_Out (listen mode “off” & CRC “on”),
  - 11 = Listening mode (CRC “on” & Frame_Time_Out “on”).

  b. UART mode (default = 00):
  - 00 = 8-bit, no parity (listen mode “off”),
  - 01 = 8-bit, even parity (listen mode “off”),
  - 10 = 8-bit, odd parity (listen mode “off”),
  - 11 = Listening mode, 8-bit, no parity.

- **Bit 3 – LENA: Enable**
  - 0 = Disable (both LIN and UART modes),
  - 1 = Enable (both LIN and UART modes).

- **Bits 2:0 – LCMD[2:0]: Command and mode**

  The command is only available if LENA is set.
  - 000 = LIN Rx Header - LIN abort,
  - 001 = LIN Tx Header,
  - 010 = LIN Rx Response,
  - 011 = LIN Tx Response,
  - 100 = UART Rx & Tx Byte disable,
  - 11x = UART Rx Byte enable,
  - 1x1 = UART Tx Byte enable.

15.6.2 LINSIR – LIN Status and Interrupt Register

Bit	7	6	5	4	3	2	1	0	
(0xC9)	LIDST2	LIDST1	LIDST0	LBUSY	LERR	LIDOK	LTXOK	LRXOK	LINSIR
Read/Write	R	R	R	R	R/Wone	R/Wone	R/Wone	R/Wone	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:5 – LIDST[2:0]: Identifier Status**
  - 0xx = no specific identifier,
  - 100 = Identifier 60 (0x3C),
  - 101 = Identifier 61 (0x3D),
  - 110 = Identifier 62 (0x3E),
  - 111 = Identifier 63 (0x3F).

- **Bit 4 – LBUSY: Busy Signal**
  - 0 = Not busy,
  - 1 = Busy (receiving or transmitting).

- **Bit 3 – LERR: Error Interrupt**

  It is a logical OR of LINERR register bits. This bit generates an interrupt if its respective enable bit - LENERR - is set in LINENIR.

  - 0 = No error,
  - 1 = An error has occurred.

  The user clears this bit by writing 1 in order to reset this interrupt. Resetting LERR also resets all LINERR bits.

  In UART mode, this bit is also cleared by reading LINDAT.

- **Bit 2 – LIDOK: Identifier Interrupt**

  This bit generates an interrupt if its respective enable bit - LENIDOK - is set in LINENIR.

  - 0 = No identifier,
  - 1 = Slave task: Identifier present, master task: Tx Header complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

- **Bit 1 – LTXOK: Transmit Performed Interrupt**

  This bit generates an interrupt if its respective enable bit - LENTXOK - is set in LINENIR.

  - 0 = No Tx,
  - 1 = Tx Response complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

  In UART mode, this bit is also cleared by writing LINDAT.

- **Bit 0 – LRXOK: Receive Performed Interrupt**

  This bit generates an interrupt if its respective enable bit - LENRXOK - is set in LINENIR.

  – 0 = No Rx

  – 1 = Rx Response complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

  In UART mode, this bit is also cleared by reading LINDAT.

15.6.3 LINENIR – LIN Enable Interrupt Register

Bit	7	6	5	4	3	2	1	0	
(0xCA)	–	–	–	–	LENERR	LENIDOK	LENTXOK	LENRXOK	LINENIR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – Reserved Bits**

  These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINENIR is written.

- **Bit 3 – LENERR: Enable Error Interrupt**

  – 0 = Error interrupt masked,

  – 1 = Error interrupt enabled.

- **Bit 2 – LENIDOK: Enable Identifier Interrupt**

  – 0 = Identifier interrupt masked,

  – 1 = Identifier interrupt enabled.

- **Bit 1 – LENTXOK: Enable Transmit Performed Interrupt**

  – 0 = Transmit performed interrupt masked,

  – 1 = Transmit performed interrupt enabled.

- **Bit 0 – LENRXOK: Enable Receive Performed Interrupt**

  – 0 = Receive performed interrupt masked,

  – 1 = Receive performed interrupt enabled.

15.6.4 LINERR – LIN Error Register

Bit	7	6	5	4	3	2	1	0	
(0xCB)	LABORT	LTOERR	LOVERR	LFERR	LSERR	LPERR	LCERR	LBERR	LINERR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – LABORT: Abort Flag**

  – 0 = No warning,

  – 1 = LIN abort command occurred.

  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 6 – LTOERR: Frame_Time_Out Error Flag**

– 0 = No error,
– 1 = Frame_Time_Out error.
This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 5 – LOVERR: Overrun Error Flag**
  – 0 = No error,
  – 1 = Overrun error.
  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 4 – LFERR: Framing Error Flag**
  – 0 = No error,
  – 1 = Framing error.
  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 3 – LSERR: Synchronization Error Flag**
  – 0 = No error,
  – 1 = Synchronization error.
  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 2 – LPERR: Parity Error Flag**
  – 0 = No error,
  – 1 = Parity error.
  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 1 – LCERR: Checksum Error Flag**
  – 0 = No error,
  – 1 = Checksum error.
  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 0 – LBERR: Bit Error Flag**
  – 0 = no error,
  – 1 = Bit error.
  This bit is cleared when LERR bit in LINSIR is cleared.

15.6.5 LINBTR – LIN Bit Timing Register

Bit	7	6	5	4	3	2	1	0	
(0xCC)	**LDISR**	**–**	**LBT5**	**LBT4**	**LBT3**	**LBT2**	**LBT1**	**LBT0**	**LINBTR**
Read/Write	R/W	R	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – LDISR: Disable Bit Timing Re synchronization**
  – 0 = Bit timing re-synchronization enabled (default),
  – 1 = Bit timing re-synchronization disabled.

- **Bits 5:0 – LBT[5:0]: LIN Bit Timing**

Gives the number of samples of a bit.

$$\text{sample-time} = (1 / f\text{clk}_{i/o}) \times (\text{LDIV}[11:0] + 1)$$

Default value: LBT[6:0]=32 — Min. value: LBT[6:0]=8 — Max. value: LBT[6:0]=63

15.6.6 LINBRR – LIN Baud Rate Register

Bit	7	6	5	4	3	2	1	0	
(0xCD)	LDIV7	LDIV6	LDIV5	LDIV4	LDIV3	LDIV2	LDIV1	LDIV0	LINBRRL
(0xCE)	–	–	–	–	LDIV11	LDIV10	LDIV9	LDIV8	LINBRRH
Bit	15	14	13	12	11	10	9	8	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 15:12 – Res: Reserved Bits**
  These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINBRR is written.

- **Bits 11:0 – LDIV[11:0]: Scaling of $clk_{i/o}$ Frequency**
  The LDIV value is used to scale the entering $clk_{i/o}$ frequency to achieve appropriate LIN or UART baud rate.

15.6.7 LINDLR – LIN Data Length Register

Bit	7	6	5	4	3	2	1	0	
(0xCF)	LTXDL3	LTXDL2	LTXDL1	LTXDL0	LRXDL3	LRXDL2	LRXDL1	LRXDL0	LINDLR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – LTXDL[3:0]: LIN Transmit Data Length**
  In LIN mode, this field gives the number of bytes to be transmitted (clamped to 8 Max).

  In UART mode this field is unused.

- **Bits 3:0 – LRXDL[3:0]: LIN Receive Data Length**
  In LIN mode, this field gives the number of bytes to be received (clamped to 8 Max).

  In UART mode this field is unused.

15.6.8 LINIDR – LIN Identifier Register

Bit	7	6	5	4	3	2	1	0	
(0xD0)	LP1	LP0	LID5 / LDL1	LID4 / LDL0	LID3	LID2	LID1	LID0	LINIDR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – LP[1:0]: Parity**

  In LIN mode:

  ```
  LP0 = LID4 ^ LID2 ^ LID1 ^ LID0
  LP1 = ! ( LID1 ^ LID3 ^ LID4 ^ LID5 )
  ```

  In UART mode this field is unused.

- **Bits 5:4 – LDL[1:0]: LIN 1.3 Data Length**

  In LIN 1.3 mode:

  – 00 = 2-byte response,
  – 01 = 2-byte response,
  – 10 = 4-byte response,
  – 11 = 8-byte response.

  In UART mode this field is unused.

- **Bits 3:0 – LID[3:0]: LIN 1.3 Identifier**

  In LIN 1.3 mode: 4-bit identifier.

  In UART mode this field is unused.

- **Bits 5:0 – LID[5:0]: LIN 2.1 Identifier**

  In LIN 2.1 mode: 6-bit identifier (no length transported).

  In UART mode this field is unused.

15.6.9 LINSEL – LIN Data Buffer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0xD1)	–	–	–	–	$\overline{\text{LAINC}}$	LINDX2	LINDX1	LINDX0	LINSEL
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W	
Initial Value	–	–	–	–	0	0	0	0	

- **Bits 7:4 – Res: Reserved Bits**

  These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINSEL is written.

- **Bit 3 – $\overline{\text{LAINC}}$: $\overline{\text{Auto Increment}}$ of Data Buffer Index**

  In LIN mode:

  – 0 = Auto incrementation of FIFO data buffer index (default),

  – 1 = No auto incrementation.

  In UART mode this field is unused.

- **Bits 2:0 – LINDX[2:0]: FIFO LIN Data Buffer Index**

  In LIN mode: location (index) of the LIN response data byte into the FIFO data buffer. The FIFO data buffer is accessed through LINDAT.

  In UART mode this field is unused.

15.6.10 LINDAT – LIN Data Register

Bit	7	6	5	4	3	2	1	0	
(0xD2)	**LDATA7**	**LDATA6**	**LDATA5**	**LDATA4**	**LDATA3**	**LDATA2**	**LDATA1**	**LDATA0**	**LINDAT**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – LDATA[7:0]: LIN Data In / Data out**

  In LIN mode: FIFO data buffer port.

  In UART mode: data register (no data buffer - no FIFO).

  – In Write access, data out.

  – In Read access, data in.

16. ISRC – Current Source

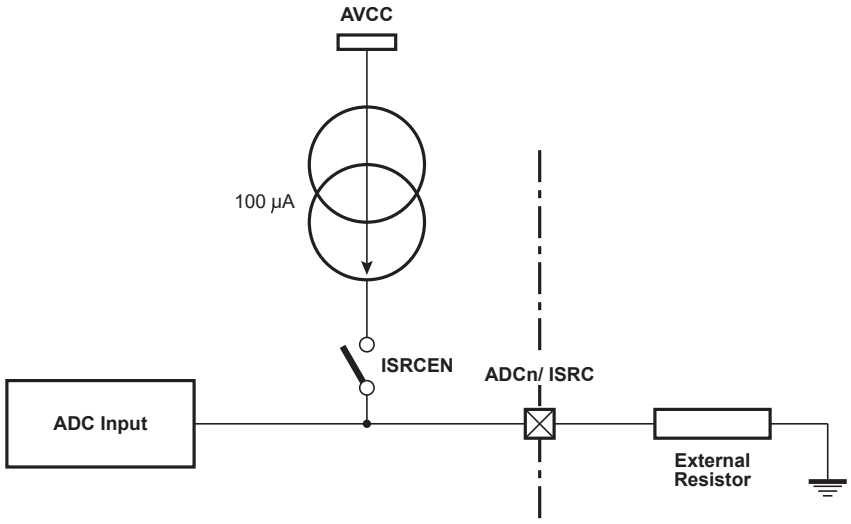
16.1 Features

- **100 µA Constant current source**
- **±10% Absolute Accuracy**

The ATtiny87/167 features a 100 µA ±10% Current Source. Up on request, the current is flowing through an external resistor. The voltage can be measured on the dedicated pin shared with the ADC. Using a resistor in serie with a ≤ 0.5% tolerance is recommended. To protect the device against big values, the ADC must be configured with AV_{CC} as internal reference to perform the first measurement. Afterwards, another internal reference can be chosen according to the previous measured value to refine the result.

When ISRCEN bit is set, the ISRC pin sources 100µA. Otherwise this pin keeps its initial function.

Figure 16-1. Current Source Block Diagram



16.2 Typical applications

16.2.1 LIN Current Source

During the configuration of a LIN node in a cluster, it may be necessary to attribute dynamically an unique physical address to every cluster node. The way to do it is not described in the LIN protocol.

The Current Source offers an excellent solution to associate a physical address to the application supported by the LIN node. A full dynamic node configuration can be used to set-up the LIN nodes in a cluster.

ATtiny87/167 proposes to have an external resistor used in conjunction with the Current Source. The device measures the voltage to the boundaries of the resistance via the Analog to Digital converter. The resulting voltage defines the physical address that the communication handler will use when the node will participate in LIN communication.

The internal Current Source solution of ATtiny87/167 immunizes the address detection against any kind of voltage variations.

Table 16-1. Example of Resistor Values(±5%) for a 8-address System (AV_{CC} = 5V[1])

Physical Address	Resistor Value R_{load} (Ohm)	Typical Measured Voltage (V)	Minimum Reading with a 2.56V ref	Typical Reading with a 2.56V ref	Maximum Reading with a 2.56V ref
0	1 000	0.1		40	
1	2 200	0.22		88	
2	3 300	0.33		132	
3	4 700	0.47		188	
4	6 800	0.68		272	
5	10 000	1		400	
6	15 000	1.5		600	
7	22 000	2.2		880	

Table 16-2. Example of Resistor Values(±1%) for a 16-address System (AV_{CC} = 5V[1])

Physical Address	Resistor Value R_{load} (Ohm)	Typical Measured Voltage (V)	Minimum Reading with a 2.56V ref	Typical Reading with a 2.56V ref	Miximum Reading with a 2.56V ref
0	1 000	0.1	38	40	45
1	1 200	0.12	46	48	54
2	1500	0.15	57	60	68
3	1800	0.18	69	72	81
4	2200	0.22	84	88	99
5	2700	0.27	104	108	122
6	3300	0.33	127	132	149
7	4700	0.47	181	188	212
8	6 800	0.68	262	272	306
9	8 200	0.82	316	328	369
10	10 000	1.0	386	400	450
11	12 000	1.2	463	480	540
12	15 000	1.5	579	600	675
13	18 000	1.8	694	720	810
14	22 000	2.2	849	880	989
15	27 000	2.7	1023	1023	1023

Note: 1. 5V range: Max R_{load} 30KΩ
3V range: Max R_{load} 15KΩ

16.2.2 Current Source for Low Cost Transducer

An external transducer based on a variable resistor can be connected to the Current Source. This can be, for instance:

- A thermistor, or temperature-sensitive resistor, used as a temperature sensor,
- A CdS photoconductive cell, or luminosity-sensitive resistor, used as a luminosity sensor,
- ...

Using the Current Source with this type of transducer eliminates the need for additional parts otherwise required in resistor network or Wheatstone bridge.

16.2.3 Voltage Reference for External Devices

An external resistor used in conjunction with the Current Source can be used as voltage reference for external devices. Using a resistor in serie with a lower tolerance than the Current Source accuracy (≤2%) is recommended. Table 16-2 gives an example of voltage references using standard values of resistors.

16.2.4 Threshold Reference for Internal Analog Comparator

An external resistor used in conjunction with the Current Source can be used as threshold reference for internal Analog Comparator (See ”AnaComp - Analog Comparator” on page 218.). This can be connected to AIN0 (negative Analog Compare input pin) as well as AIN1 (positive Analog Compare input pin). Using a resistor in serie with a lower tolerance than the Current Source accuracy (≤2%) is recommended. Table 16-2 gives an example of threshold references using standard values of resistors.

16.3 Control Register

16.3.1 AMISCR – Analog Miscellaneous Control Register

Bit	7	6	5	4	3	2	1	0	
(0x77)	–	–	–	–	–	AREFEN	XREFEN	ISRCEN	AMISCR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – ISRCEN: Current Source Enable**

Writing this bit to one enables the Current Source as shown in Figure 16-1. It is recommended to use DIDR register bit function when ISRCEN is set. It also recommended to turn off the Current Source as soon as possible (ex: once the ADC measurement is done).

17. ADC – Analog to Digital Converter

17.1 Features

- **10-bit Resolution**
- **1.0 LSB Integral Non-linearity**
- **± 2 LSB Absolute Accuracy**
- **13 - 260 µs Conversion Time (Low - High Resolution)**
- **Up to 15 kSPS at Maximum Resolution**
- **11 Multiplexed Single Ended Input Channels**
- **8 Differential input pairs with selectable gain**
- **Temperature sensor input channel**
- **Voltage from Internal Current Source Driving (ISRC)**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - AV_{CC} ADC Input Voltage Range**
- **Selectable 1.1V / 2.56V ADC Voltage Reference**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**
- **Unipolar / Bipolar Input Mode**
- **Input Polarity Reversal Mode**

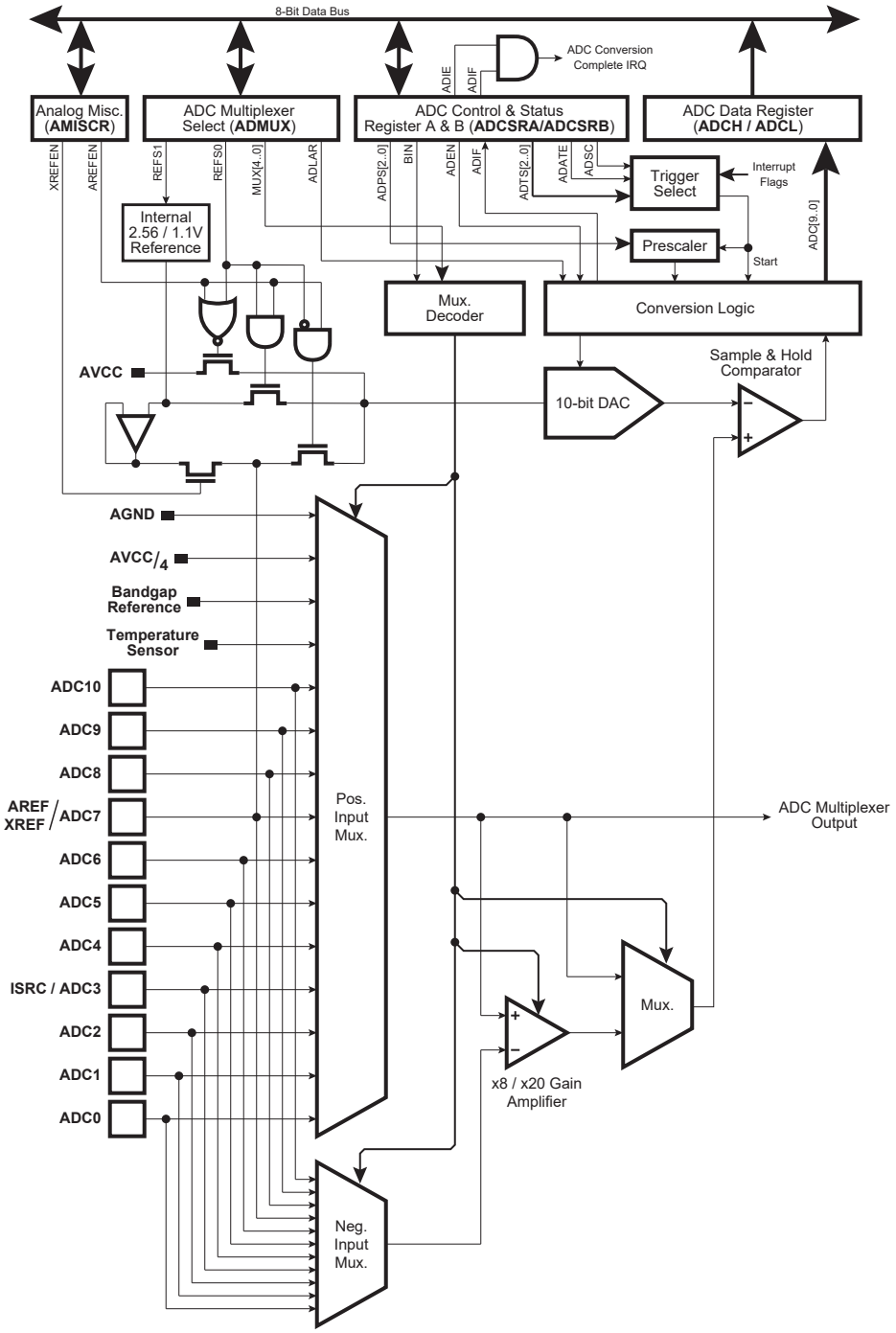
17.2 Overview

The ATtiny87/167 features a 10-bit successive approximation ADC. The ADC is connected to a 11-channel Analog Multiplexer which allows 16 differential voltage input combinations and 11 single-ended voltage inputs constructed from the pins PA[7:0] or PB[7:4]. The differential input is equipped with a programmable gain stage, providing amplification steps of 8x or 20x on the differential input voltage before the A/D conversion. The single-ended voltage inputs refer to 0V (AGND).

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 17-1.

Internal reference voltages of nominally 1.1V or 2.56V are provided On-chip. Alternatively, AV_{CC} can be used as reference voltage for single ended channels. There are also options to output the internal 1.1V or 2.56V reference voltages or to input an external voltage reference and turn-off the internal voltage reference. These options are selected using the REFS[1:0] bits of the ADMUX control register and using AREFEN and XREFEN bits of the AMISCR control register.

Figure 17-1. Analog to Digital Converter Block Schematic



17.3 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents AGND and the maximum value represents the voltage on AV_{CC}, the voltage refrence on AREF pin or an internal 1.1V / 2.56V voltage reference.

The voltage reference for the ADC may be selected by writing to the REFS[1:0] bits in ADMUX and AREFEN bit in AMISCR. The AV_{CC} supply, the AREF pin or an internal 1.1V / 2.56V voltage reference may be selected as the ADC voltage reference.

The analog input channel and differential gain are selected by writing to the MUX[4:0] bits in ADMUX register. Any of the 11 ADC input pins ADC[10:0] can be selected as single ended inputs to the ADC. The positive and negative inputs to the differential gain amplifier are described in Table 17-5.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input pair by the selected gain factor 8x or 20x, according to the setting of the MUX[4:0] bits in ADMUX register. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

The on-chip temperature sensor is selected by writing the code defined in Table 17-5 to the MUX[4:0] bits in ADMUX register when its dedicated ADC channel is used as an ADC input.

A specific ADC channel (defined in Table 17-5) is used to measure the voltage to the boundaries of an external resistance flowing by a current driving by the Internal Current Source (ISRC).

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA register. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

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

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

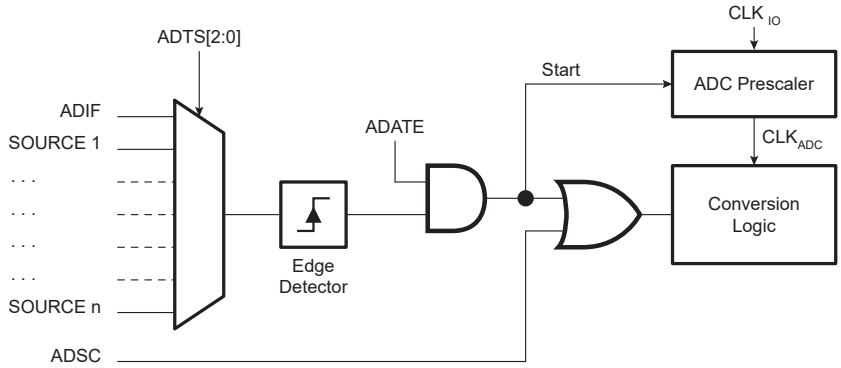
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

17.4 Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be 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, ADATE in ADCSRA register. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB register (see description of the ADTS bits for a list of the 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 SREG register 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 17-2. ADC Auto Trigger Logic

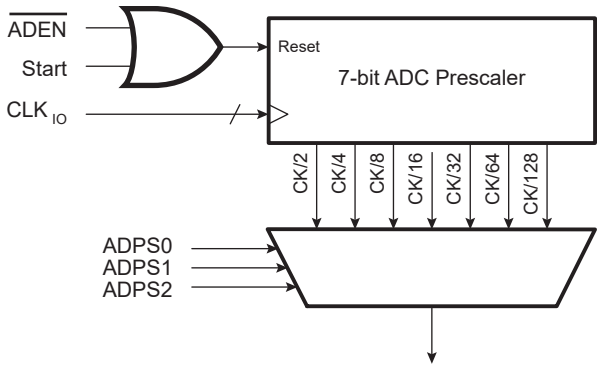


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA register. 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 ADSC in ADCSRA register to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

17.5 Prescaling and Conversion Timing

Figure 17-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200 kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA register. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA register. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

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

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA register is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 14.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place 2 ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 17-1.

Figure 17-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

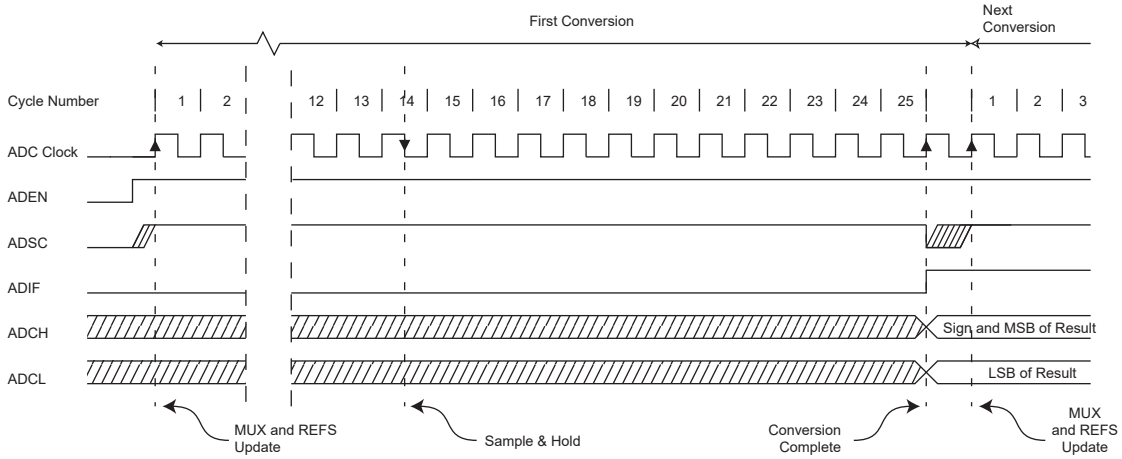


Figure 17-5. ADC Timing Diagram, Single Conversion

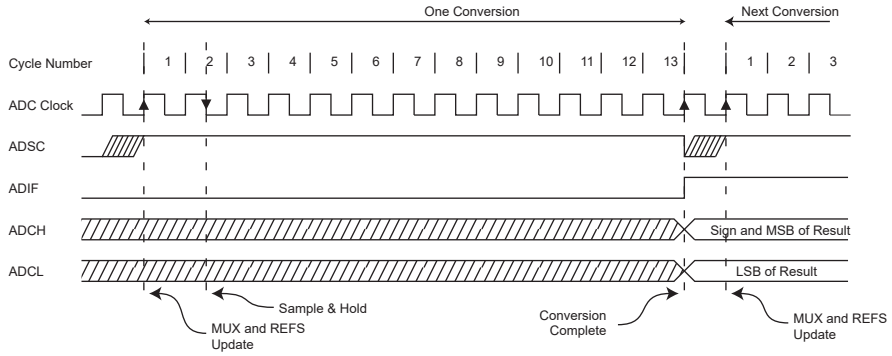


Figure 17-6. ADC Timing Diagram, Auto Triggered Conversion

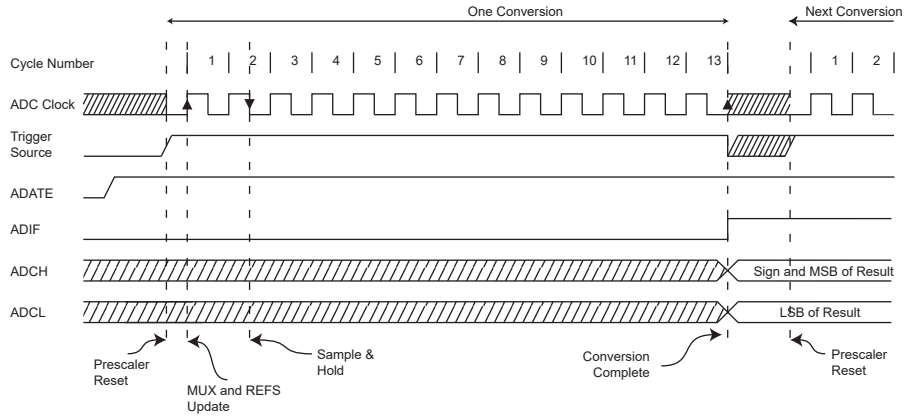


Figure 17-7. ADC Timing Diagram, Free Running Conversion

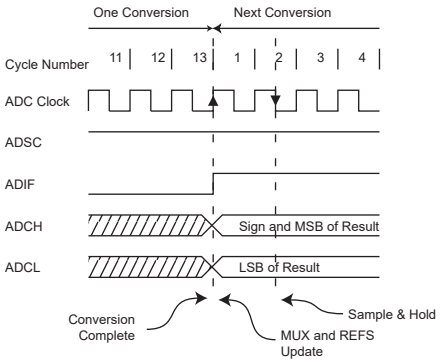


Table 17-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5 cycles	25 cycles
Normal conversions	1.5 cycles	13 cycles
Auto Triggered conversions	2 cycles	13.5 cycles

17.6 Changing Channel or Reference Selection

The MUX[4:0] and REFS[1:0] bits in the ADMUX register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA register is 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 ADSC is 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 ADATE and ADEN is written to one, 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:

a. When ADATE or ADEN is cleared.
b. During conversion, minimum one ADC clock cycle after the trigger event.
c. 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.

17.6.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

17.6.2 ADC Voltage Reference

The voltage reference for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AV_{CC}, internal 1.1V / 2.56V voltage reference or external AREF pin. The first ADC conversion result after switching voltage reference source may be inaccurate, and the user is advised to discard this result.

17.7 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

a. 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.
b. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
c. 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 that 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 ADEN before entering such sleep modes to avoid excessive power consumption.

17.7.1 Analog Input Circuitry

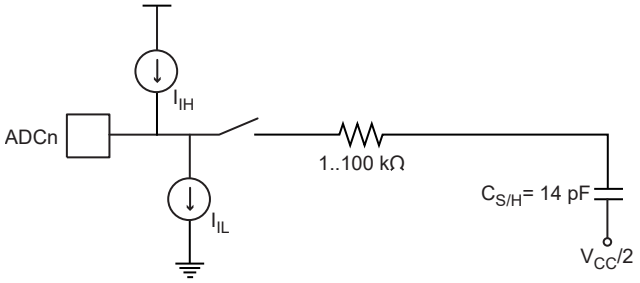
The analog input circuitry for single ended channels is illustrated in Figure 17-8. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive 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 impedant 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 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 17-8. Analog Input Circuitry



17.7.2 Analog Noise Canceling Techniques

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

a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
b. Use the ADC noise canceler function to reduce induced noise from the CPU.
c. If any port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

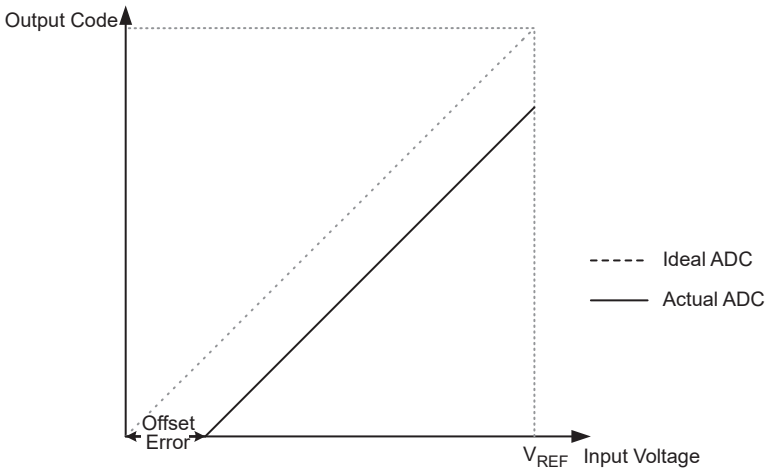
17.7.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n-1.

Several parameters describe the deviation from the ideal behavior:

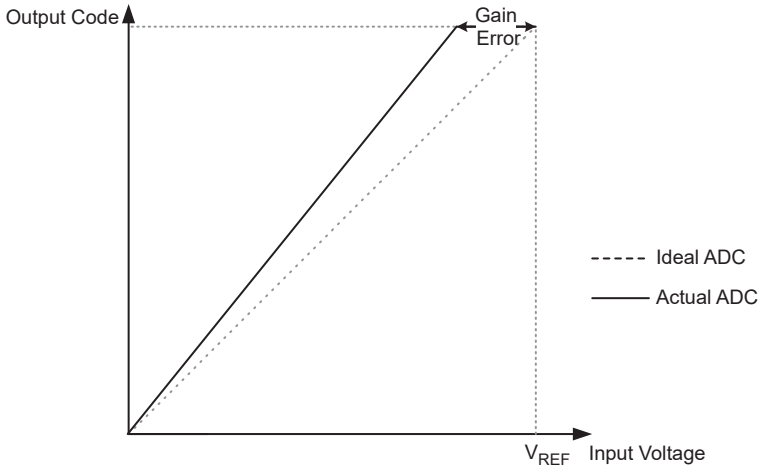
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 17-9. Offset Error



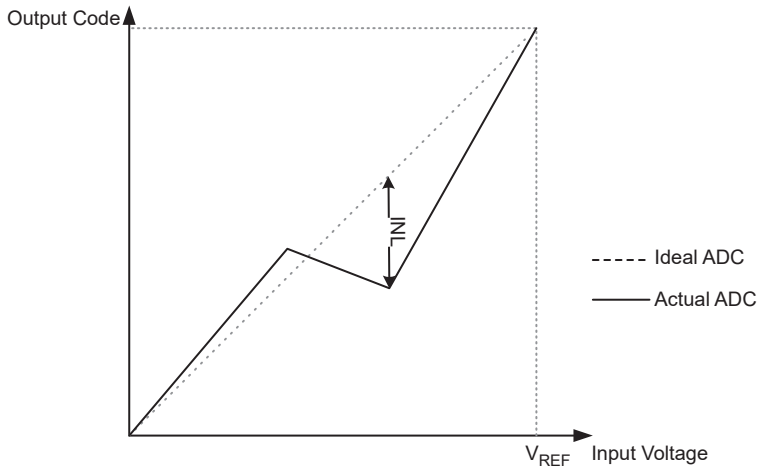
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 17-10. Gain Error



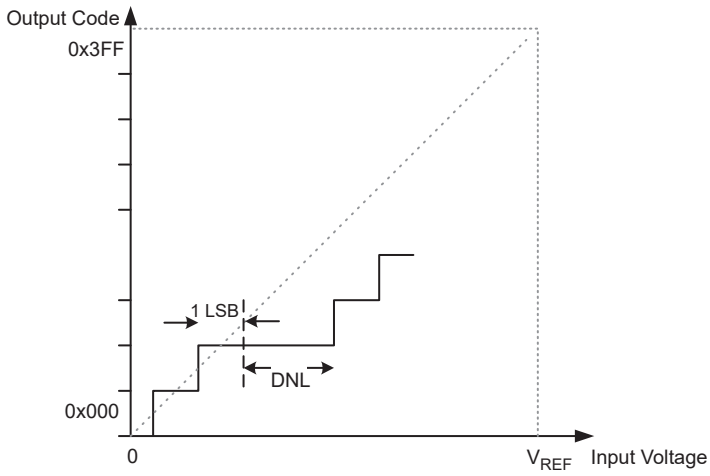
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 17-11. 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 17-12. 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.

17.8 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversion as there are three types of conversions: single ended conversion, unipolar differential conversion and bipolar differential conversion.

17.8.1 Single Ended Conversion

For single ended conversion, the result is:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 17-4 on page 212 and Table 17-5 on page 213). 0x000 represents analog ground, and 0x3FF represents the selected voltage reference minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

17.8.2 Unipolar Differential Conversion

If differential channels and an unipolar input mode are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot 1024}{V_{REF}} \cdot GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference (see Table 17-4 on page 212 and Table 17-5 on page 213). The voltage on the positive pin must always be larger than the voltage on the negative pin or otherwise the voltage difference is saturated to zero. The result is presented in one-sided form, from 0x000 (0_d) to 0x3FF ($+1023_d$). The GAIN is either 8x or 20x.

17.8.3 Bipolar Differential Conversion

As default the ADC converter operates in the unipolar input mode, but the bipolar input mode can be selected by writing the BIN bit in the ADCSRB register to one. In the bipolar input mode two-sided voltage differences are allowed and thus the voltage on the negative input pin can also be larger than the voltage on the positive input pin. If differential channels and a bipolar input mode are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot 512}{V_{REF}} \cdot GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512_d) through 0x000 ($+0_d$) to 0x1FF ($+511_d$). The GAIN is either 8x or 20x.

However, if the signal is not bipolar by nature (9 bits + sign as the 10th bit), this scheme loses one bit of the converter dynamic range. Then, if the user wants to perform the conversion with the maximum dynamic range, the user can perform a quick polarity check of the result and use the unipolar differential conversion with selectable differential input pair. When the polarity check is performed, it is sufficient to read the MSB of the result (ADC9 in ADCH register). If the bit is one, the result is negative, and if this bit is zero, the result is positive.

17.9 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC input. MUX[4:0] bits in ADMUX register enables the temperature sensor. The internal 1.1V voltage reference must also be selected for the ADC voltage reference source in the temperature sensor measurement. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor.

The measured voltage has a linear relationship to the temperature as described in Table 17-2 The sensitivity is approximately 1 LSB / °C and the accuracy depends on the method of user calibration. Typically, the measurement accuracy after a single temperature calibration is ±10°C, assuming calibration at room temperature. Better accuracies are achieved by using two temperature points for calibration.

Table 17-2. Temperature vs. Sensor Output Voltage (Typical Case): Example ADC Values

Temperature/°C	–40°C	+25°C	+85°C
	0x00F6	0x0144	0c01B8

The values described in Table 17-2 are typical values. However, due to process variation the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results the temperature measurement can be calibrated in the application software. The sofware calibration can be done using the formula:

$$T = k * [(ADCH << 8) \mid ADCL] + T_{OS}$$

where ADCH and ADCL are the ADC data registers, T is temperature in Kelvin, k is the fixed slope coefficient and T_{OS} is the temperature sensor offset. Typically, k is very close to 1.0 and in single-point calibration the coefficient may be omitted. Where higher accuracy is required the slope coefficient should be evaluated based on measurements at two temperatures.

17.10 Internal Voltage Reference Output

The internal voltage reference is output on XREF pin as described in Table 17-3 if the ADC is turned on (See “Voltage Reference Enable Signals and Start-up Time” on page 61.). Addition of an external filter capacitor (5 - 10 nF) on XREF pin may be necessary. XREF current load must be from 1 µA to 100 µA with V_{CC} from 2.7V to 5.5V for XREF = 1.1V and with V_{CC} from 4.5V to 5.5V for XREF = 2.56V.

XREF pin can be coupled to an analog input of the ADC (See ”Pin Configuration” on page 11.).

Table 17-3. Internal Voltage Reference Output

XREFEN [1]	REFS1 [2]	REFS0 [2]	Voltage Reference Output ($I_{load} \leq$ 100 µA)
0	x	x	Hi-Z, the pin can be used as AREF input or other alternate functions.
1 [1]	0	1	XREF = 1.1V [3]
1 [1]	1	1	XREF = 2.56V [3][4]

Notes:
1. See ”Bit 1 – **XREFEN: Internal Voltage Reference Output Enable” on page 217.**
2. See ”**Bits 7:6 – REFS[1:0]: Voltage Reference Selection Bits” on page 212.**
3. In these configurations, the pin pull-up must be turned off and the pin digital output must be set in Hi-Z.
4. V_{CC} in range 4.5 - 5.5V.

17.11 Register Description

17.11.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – REFS[1:0]: Voltage Reference Selection Bits**

These bits and AREFEN bit from the Analog Miscellaneous Control Register (AMISCR) select the voltage reference for the ADC, as shown in Table 17-4. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA register is set). Whenever these bits are changed, the next conversion will take 25 ADC clock cycles. If active channels are used, using AVCC or an external AREF higher than (AV_{CC} - 1V) is not recommended, as this will affect ADC accuracy. The internal voltage reference options may not be used if an external voltage is being applied to the AREF pin.

Table 17-4. Voltage Reference Selections for ADC

REFS1	REFS0	AREFEN	Voltage Reference (V_{REF}) Selection
X	0	0	AV_{CC} used as Voltage Reference, diconnected from AREF pin
X	0	1	External Voltage Reference at AREF pin (AREF ≥ 2.0V)
0	1	0	Internal 1.1V Voltage Reference
1	1	0	Internal 2.56V Voltage Reference

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

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see “ADCL and ADCH – ADC Data Register” on page 215.

- **Bits 4:0 – MUX[4:0]: Analog Channel and Gain Selection Bits**

These bits select which combination of analog inputs are connected to the ADC. In case of differential input, gain selection is also made with these bits. Refer to Table 17-5 for details. If these bits are changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSRA register is set).

Table 17-5. Input Channel Selections

MUX[4:0]	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
0 0000	ADC0 (PA0)	NA	NA	NA
0 0001	ADC1 (PA1)			
0 0010	ADC2 (PA2)			
0 0011	ADC3 / ISRC (PA3)			
0 0100	ADC4 (PA4)			
0 0101	ADC5 (PA5)			
0 0110	ADC6 (PA6)			
0 0111	ADC7 / AREF (PA7)			
0 1000	ADC8 (PB5)			
0 1001	ADC9 (PB6)			
0 1010	ADC10 (PB7)			
0 1011	Temperature Sensor			
0 1100	Bandgap Reference (1.1 V)			
0 1101	$AV_{CC}/4$			
0 1110	GND (0V)			
0 1111	(reserved)			
1 0000	N/A	ADC0 (PA0)	ADC1 (PA1)	8x
1 0001				20x
1 0010		ADC1 (PA1)	ADC2 (PA2)	8x
1 0011				20x
1 0100		ADC2 (PA2)	ADC3 (PA3)	8x
1 0101				20x
1 0110		ADC4 (PA4)	ADC5 (PA5)	8x
1 0111				20x
1 1000		ADC5 (PA5)	ADC6 (PA6)	8x
1 1001				20x
1 1010		ADC6 (PA6)	ADC7 (PA7)	8x
1 1011				20x
1 1100		ADC8 (PB5)	ADC9 (PB6)	8x
1 1101				20x
1 1110		ADC9 (PB6)	ADC10 (PB7)	8x
1 1111				20x

17.11.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

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

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

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

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

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

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

- **Bits 2:0 – ADPS[2:0]: ADC Prescaler Select Bits**

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

Table 17-6. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8

Table 17-6. ADC Prescaler Selections (Continued)

ADPS2	ADPS1	ADPS0	Division Factor
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

17.11.3 ADCL and ADCH – ADC Data Register

17.11.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	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	

17.11.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

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

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

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

- **ADC[9:0]: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in “ADC Conversion Result” on page 209.

17.11.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	BIN	ACME	ACIR1	ACIR0	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – BIN: Bipolar Input Mode**

The gain stage is working in the unipolar mode as default, but the bipolar mode can be selected by writing the BIN bit in the ADCSRB register. In the unipolar mode only one-sided conversions are supported and the voltage on the positive input must always be larger than the voltage on the negative input. Otherwise the result is saturated to the voltage reference. In the bipolar mode two-sided conversions are supported and the result is represented in the two's complement form. In the unipolar mode the resolution is 10 bits and the bipolar mode the resolution is 9 bits + 1 sign bit.

- **Bit 3 – Res: Reserved Bit**

This bit is reserved for future use. For compatibility with future devices it must be written to zero when ADCSRB register is written.

- **Bits 2:0 – ADTS[2:0]: ADC Auto Trigger Source**

If ADATE in ADCSRA register 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 register 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 17-7. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter1 Compare Match A
1	0	0	Timer/Counter1 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Capture Event
1	1	1	Watchdog Interrupt Request

17.11.5 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
(0x7E)	ADC7D / AIN1D	ADC6D / AIN0D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – ADC7D:ADC0D: ADC[7:0] Digital Input Disable**

When this bit is written 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.

17.11.6 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
(0x7F)	–	ADC10D	ADC9D	ADC8D	–	–	–	–	DIDR1
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is reserved for future use. For compatibility with future devices it must be written to zero when DIDR1 register is written.

- **Bits 6:4 – ADC10D, ADC9D, ADC8D: ADC[10:8] Digital Input Disable**

When this bit is written 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 ADC10:8 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.

- **Bits 3:0 - Res: Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, they must be written to zero when DIDR1 is written.

17.11.7 AMISCR – Analog Miscellaneous Control Register

Bit	7	6	5	4	3	2	1	0	
(0x77)	–	–	–	–	–	AREFEN	XREFEN	ISRCEN	AMISCR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, they must be written to zero when AMISCR is written.

- **Bit 2 – AREFEN: External Voltage Reference Input Enable**

When this bit is written logic one, the voltage reference for the ADC is input from AREF pin as described in Table 17.10 on page 210. If active channels are used, using AV_{CC} or an external AREF higher than (AV_{CC} - 1V) is not recommended, as this will affect ADC accuracy. The internal voltage reference options may not be used if an external voltage is being applied to the AREF pin. It is recommended to use DIDR register bit function (digital input disable) when AREFEN is set.

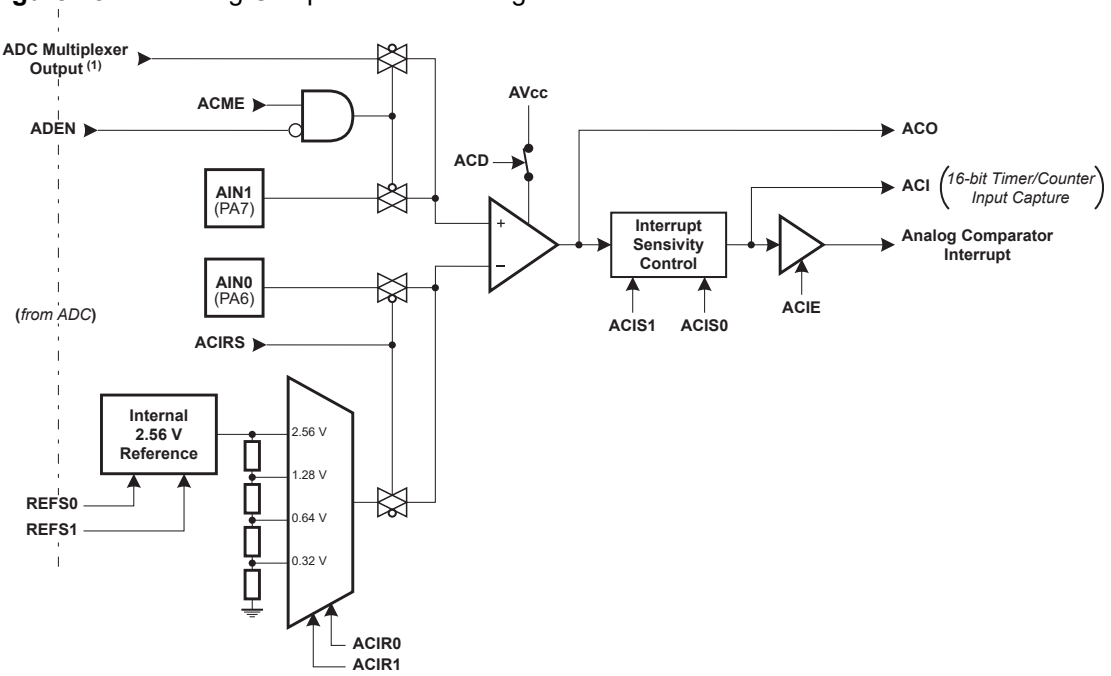
- **Bit 1 – XREFEN: Internal Voltage Reference Output Enable**

When this bit is written logic one, the internal voltage reference 1.1V or 2.56V is output on XREF pin as described in Table 17.10 on page 210. It is recommended to use DIDR register bit function (digital input disable) when XREFEN is set.

18. AnaComp - Analog Comparator

The Analog Comparator compares the input values on the positive pin (AIN1) and negative pin (AIN0). When the voltage on the positive pin is higher than the voltage on the negative pin, the Analog Comparator output, ACO, is set. The comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 18-1.

Figure 18-1. Analog Comparator Block Diagram(1)(2)



Notes:
1. See Table 18-1 on page 219 and Table 18-2 on page 219
2. Refer to Figure 1-2 on page 11 and Table 9-3 on page 84 for Analog Comparator pin placement.

18.1 Analog Comparator Inputs

18.1.1 Analog Compare Positive Input

It is possible to select any of the inputs of the ADC Positive Input Multiplexer to replace the positive input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB register) is set and the ADC is switched off (ADEN in ADCSRA register is zero), MUX[4:0] in ADMUX register select the input pin to replace the positive input to the Analog Comparator, as shown in Table 18-1. If ACME is cleared or ADEN is set, AIN1 pin is applied to the positive input to the Analog Comparator.

Table 18-1. Analog Comparator Positive Input

ACME	ADEN	MUX[4:0]	Analog Comparator Positive Input - Comment	
0	x	x xxxx$_b$	AIN1	ADC Switched On
x	1	x xxxx$_b$	AIN1	
1	0	0 0000$_b$	ADC0	ADC Switched Off.
1	0	0 0001$_b$	ADC1	
1	0	0 0010$_b$	ADC2	
1	0	0 0011$_b$	ADC3 / ISRC	
1	0	0 0100$_b$	ADC4	
1	0	0 0101$_b$	ADC5	
1	0	0 0110$_b$	ADC6	
1	0	0 0111$_b$	ADC7	
1	0	0 1000$_b$	ADC8	
1	0	0 1001$_b$	ADC9	
1	0	0 1010$_b$	ADC10	
1	0	Other	This doesn't make sense - Don't use.	

18.1.2 Analog Compare Negative Input

It is possible to select an internal voltage reference to replace the negative input to the Analog Comparator. The output of a 2-bit DAC using the Internal Voltage Reference of the DAC is available when ACIRS bit of ACSR register is set. The voltage reference division factor is done by ACIR[1:0] of ADCSRB register.
If ACIRS is cleared, AIN0 pin is applied to the negative input to the Analog Comparator.

Table 18-2. Analog Comparator Negative Input

ACIRS	ACIR[1:0]	REFS[1:0]	Analog Comparator Negative Input - Comment
0	x	x	AIN0
1	x	0 0$_b$ 0 1$_b$ 1 0$_b$	Reserved
1	0 0$_b$	1 1$_b$	2.56 V - using Internal 2.56V Voltage Reference
1	0 1$_b$	1 1$_b$	1.28 V ($^1/_2$ of 2.56 V) - using Internal 2.56V Voltage Reference
1	1 0$_b$	1 1$_b$	0.64 V ($^1/_4$ of 2.56 V - using Internal 2.56V Voltage Reference
1	1 1$_b$	1 1$_b$	0.32 V ($^1/_8$ of 2.56 V) - using Internal 2.56V Voltage Reference

18.2 Register Description

18.2.1 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	BIN	ACME	ACIR1	ACIR0	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the positive input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the positive input of the Analog Comparator.

When the Analog to Digital Converter (ADC) is configured as single ended input channel, it is possible to select any of the ADC[10:0] pins to replace the positive input to the Analog Comparator. The ADC multiplexer (MUX[4:0]) is used to select this input, and consequently, the ADC must be switched off to utilize this feature.

- **Bits 5:4 – ACIR[1:0]: Analog Comparator Internal Voltage Reference Select**

When ACIRS bit is set in ADCSRA register, these bits select a voltage reference for the negative input to the Analog Comparator, see Table 18-2 on page 219.

18.2.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	ACD	ACIRS	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is 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 of ACSR register. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACIRS: Analog Comparator Internal Reference Select**

When this bit is set an Internal Reference Voltage replaces the negative input to the Analog Comparator (c.f. Table 18-2 on page 219).
If ACIRS is cleared, AIN0 is applied to the negative input to the Analog Comparator.

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

The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

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

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

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

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

- **Bits 1:0 – ACIS[1:0]: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 18-3.

Table 18-3. ACIS1 / ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

Note: 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.

18.2.3 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
(0x7E)	ADC7D / AIN1D	ADC6D / AIN0D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – AIN1D, AIN0D: AIN1 and AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding Analog Compare 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 AIN0/1 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.

19. DebugWIRE On-chip Debug System

19.1 Features

- **Complete Program Flow Control**
- **Emulates All On-chip Functions, Both Digital and Analog, except RESET Pin**
- **Real-time Operation**
- **Symbolic Debugging Support (Both at C and Assembler Source Level, or for Other HLLs)**
- **Unlimited Number of Program Break Points (Using Software Break Points)**
- **Non-intrusive Operation**
- **Electrical Characteristics Identical to Real Device**
- **Automatic Configuration System**
- **High-Speed Operation**
- **Programming of Non-volatile Memories**

19.2 Overview

The debugWIRE On-chip debug system uses a One-wire, bi-directional interface to control the program flow, execute AVR instructions in the CPU and to program the different non-volatile memories.

19.3 Physical Interface

When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 19-1. The debugWIRE Setup

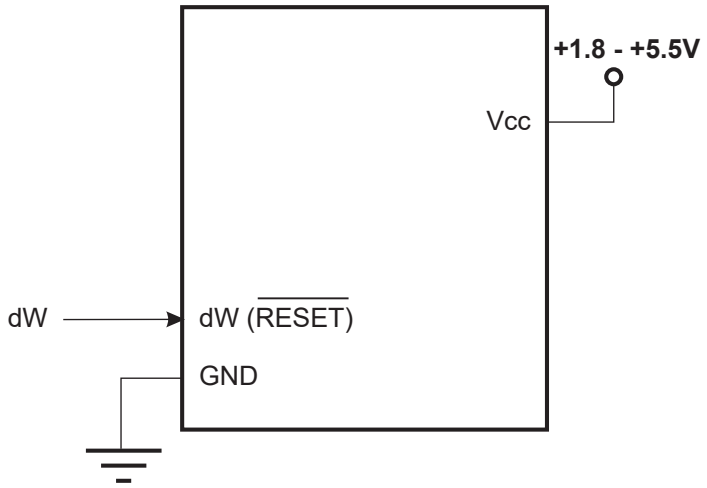


Figure 19-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

19.4 Software Break Points

DebugWIRE supports Program memory break points by the AVR BREAK instruction. Setting a break point in Atmel Studio will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a break point is changed. This is automatically handled by Atmel Studio through the debugWIRE interface. The use of break points will therefore reduce the Flash Data retention. Devices used for debugging purposes should not be shipped to end customers.

19.5 Limitations of DebugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset (RESET). An External Reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O Registers via the debugger (Atmel Studio).

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

19.6 DebugWIRE Related Register in I/O Memory

The following section describes the registers used with the debugWire.

19.6.1 DWDR – DebugWIRE Data Register

Bit	7	6	5	4	3	2	1	0	
0x31 (0x51)	DWDR[7:0]								DWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

20. Flash Programming

The device provides a Self-Programming mechanism for downloading and uploading program code by the MCU itself. The Self-Programming can use any available data interface (i.e. LIN, USART, ...) and associated protocol to read code and write (program) that code into the Program memory.

The Program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the Page Erase command or between a Page Erase and a Page Write operation:

- Alternative 1, fill the buffer before a Page Erase
  - Fill temporary page buffer
  - Perform a Page Erase
  - Perform a Page Write
- Alternative 2, fill the buffer after Page Erase
  - Perform a Page Erase
  - Fill temporary page buffer
  - Perform a Page Write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be re-written. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page.

20.1 Self-Programming the Flash

20.1.1 Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “00000011_b” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- The CPU is halted during the Page Erase operation.

20.1.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001_b” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a Page Write operation or by writing the CTPB bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

20.1.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00000101_b” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- The CPU is halted during the Page Write operation.

20.2 Addressing the Flash During Self-Programming

The Z-pointer is used to address the SPM commands. The Z pointer consists of the Z-registers ZL and ZH in the register file. The number of bits actually used is implementation dependent.

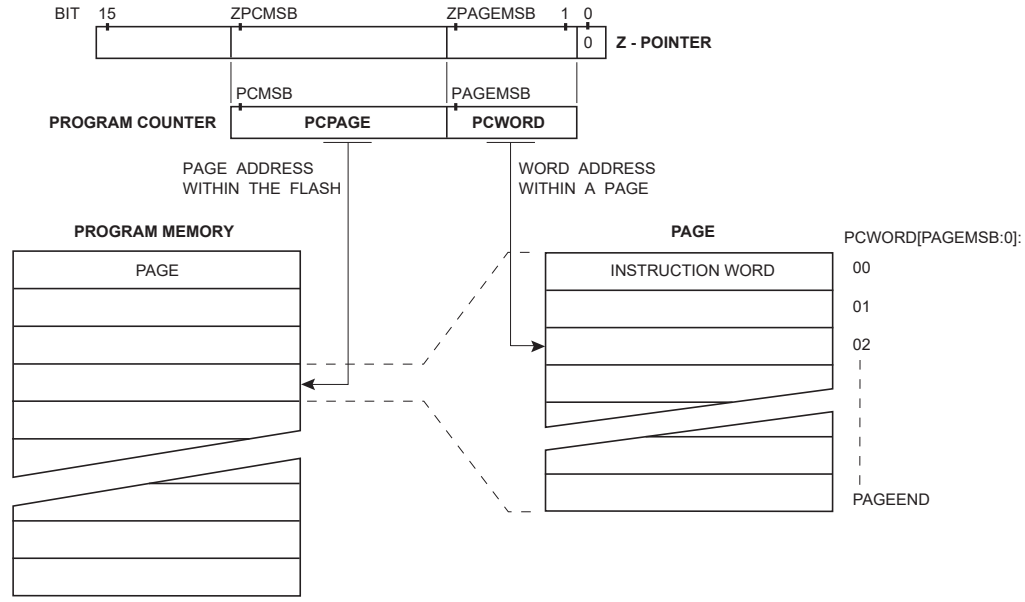
Bit	15	14	13	12	11	10	9	8	
	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8	ZH (R31)
	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	ZL (R30)
Bit	7	6	5	4	3	2	1	0	

Since the Flash is organized in pages (see Table 21-7 on page 235), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 20-1.

Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 20-1. Addressing the Flash During SPM (1)



Note: 1. The different variables used in Table 20-2 are listed in Table 21-7 on page 235.

20.2.1 SPMCSR – Store Program Memory Control and Status Register

The Store Program Memory Control and Status Register contains the control bits needed to control the Boot Loader operations.

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	–	RWWSB	SIGRD	CTPB	RFLB	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is a reserved bit in the ATtiny87/167 and will always read as zero.

- **Bit 6 – RWWSB: Read-While-Write Section Busy**

This bit is for compatibility with devices supporting Read-While-Write. It will always read as zero in ATtiny87/167.

- **Bit 5 – SIGRD: Signature Row Read**

If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register. See ”Reading the Signature Row from Software” on page 228. for details. An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect.

- **Bit 4 – CTPB: Clear Temporary Page Buffer**

If the CTPB bit is written while filling the temporary page buffer, the temporary page buffer will be cleared and the data will be lost.

- **Bit 3 – RFLB: Read Fuse and Lock Bits**

An LPM instruction within three cycles after RFLB and SPMEN are set in the SPMCSR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See ”Reading the Fuse and Lock Bits from Software” on page 227. for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a Page Write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Zpointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 0 – SPMEN: Self Programming Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either SIGRD, CTPB, RFLB, PGWRT, or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During Page Erase and Page Write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than “10 0001 $_b$”, “01 0001 $_b$”, “00 1001 $_b$”, “00 0101 $_b$”, “00 0011 $_b$” or “00 0001 $_b$” in the lower six bits will have no effect.

Note: Only one SPM instruction should be active at any time.

20.2.2 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

20.2.3 Reading the Fuse and Lock Bits from Software

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with 0x0001 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the RFLB and SPMEN bits are set in SPMCSR, the value of the Lock bits will be loaded in the destination register. The RFLB and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0001)	–	–	–	–	–	–	LB2	LB1

The algorithm for reading the Fuse Low byte is similar to the one described above for reading the Lock bits. To read the Fuse Low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the Fuse Low byte (FLB) will be loaded in the destination register as shown below. See Table 21-5 on page 234 for a detailed description and mapping of the Fuse Low byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0000)	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the Fuse High byte (FHB), load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the Fuse High byte will be loaded in the destination register as shown below. See Table 21-4 on page 233 for detailed description and mapping of the Fuse High byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0003)	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Similarly, when reading the Extended Fuse byte (EFB), load 0x0002 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the Extended Fuse byte will be loaded in the destination register as shown below. See Table 21-3 on page 233 for detailed description and mapping of the Extended Fuse byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0002)	–	–	–	–	–	–	–	EFB0

Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

20.2.4 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Table 20-1 on page 228 and set the SIGRD and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the Signature Row Lock bits or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Table 20-1. Signature Row Addressing

Signature Byte	Z-Pointer Address
Device Signature Byte 0	0x0000
Device Signature Byte 1	0x0002
Device Signature Byte 2	0x0004
8MHz RC Oscillator Calibration Byte	0x0001
TSOFFSET - Temp Sensor Offset	0x0003
TSGAIN - Temp Sensor Gain	0x0005

Note: All other addresses are reserved for future use.

20.2.5 Preventing Flash Corruption

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

A Flash program corruption can be caused by two situations when the voltage is too low.

- First, a regular write sequence to the Flash requires a minimum voltage to operate correctly.
- Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

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

1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
2. Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

20.2.6 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 20-2 shows the typical programming time for Flash accesses from the CPU.

Table 20-2. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms

20.2.7 Simple Assembly Code Example for a Boot Loader

Note that the RWWSB bit will always be read as zero in ATtiny87/167. Nevertheless, it is recommended to check this bit as shown in the code example, to ensure compatibility with devices supporting Read-While-Write.

```
  ;- The routine writes one page of data from RAM to Flash
  ;  the first data location in RAM is pointed to by the Y-pointer
  ;  the first data location in Flash is pointed to by the Z-pointer
  ;- Error handling is not included
  ;- Registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
  ;                  loophi (r25), spmcsrval (r20)
  ; -Storing and restoring of registers is not included in the routine
  ;  register usage can be optimized at the expense of code size

.equ PAGESIZEB = PAGESIZE*2   ; AGESIZEB is page size in BYTES, not words
.org SMALLBOOTSTART

Write_page:
  ; Page Erase
  ldi    spmcsrval, (1<<PGERS) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Clear temporary page buffer
  ldi    spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Transfer data from RAM to Flash temporary page buffer
  ldi    looplo, low(PAGESIZEB)    ; init loop variable
  ldi    loophi, high(PAGESIZEB)   ; not required for PAGESIZEB<=256

Wrloop:
  ld     r0, Y+
  ld     r1, Y+
  ldi    spmcsrval, (1<<SELFPGEN)
  rcall  Do_spm
  adiw   ZH:ZL, 2
  sbiw   loophi:looplo, 2         ; use subi for PAGESIZEB<=256
  brne   Wrloop

  ; Execute Page Write
  subi   ZL, low(PAGESIZEB)       ; restore pointer
  sbci   ZH, high(PAGESIZEB)      ; not required for PAGESIZEB<=256
  ldi    spmcsrval, (1<<PGWRT) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Clear temporary page buffer
  ldi    spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Read back and check, optional
  ldi    looplo, low(PAGESIZEB)   ; init loop variable
  ldi    loophi, high(PAGESIZEB)  ; not required for PAGESIZEB<=256
  subi   YL, low(PAGESIZEB)       ; restore pointer
  sbci   YH, high(PAGESIZEB)

Rdloop:
  lpm    r0, Z+
```

```
  ld     r1, Y+
  cpse   r0, r1
  rjmp   Error
  sbiw   loophi:looplo, 1         ; use subi for PAGESIZEB<=256
  brne   Rdloop

  ; To ensure compatibility with devices supporting Read-While-Write
  ; Return to RWW section
  ; Verify that RWW section is safe to read

Return:
  in     temp1, SPMCSR
  sbrs   temp1, RWWSB     ; If RWWSB is set, the RWW section is not ready yet
  ret
  ; Clear temporary page buffer
  ldi    spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
  call   Do_spm
  rjmp   Return

Do_spm:
  ; Check for previous SPM complete
Wait_spm:
  in     temp1, SPMCSR
  sbrc   temp1, SELFPGEN
  rjmp   Wait_spm
  ; Input: spmcsrval determines SPM action
  ; Disable interrupts if enabled, store status
  in     temp2, SREG
  cli
  ; Check that no EEPROM write access is present
Wait_ee:
  sbic   EECR, EEPE
  rjmp   Wait_ee
  ; SPM timed sequence
  out    SPMCSR, spmcsrval
  spm
  ; Restore SREG (to enable interrupts if originally enabled)
  out    SREG, temp2
  ret
```

21. Memory Programming

21.1 Program and Data Memory Lock Bits

The ATtiny87/167 provides two Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 21-2. The Lock bits can only be erased to “1” with the Chip Erase command. The ATtiny87/167 has no separate Boot Loader section.

Table 21-1. Lock Bit Byte(Note:)

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)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

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

Table 21-2. Lock Bit Protection Modes(1)(2)

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.(1)
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.(1)

Notes: 1. Program the Fuse bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

21.2 Fuse Bits

The ATtiny87/167 has three Fuse bytes. Table 21-3, Table 21-4 & Table 21-5 describe briefly the functionality of all the fuses and how they are mapped into the Fuse bytes.

The SPM instruction is enabled for the whole Flash if the SELFPRGEN fuse is programmed (“0”), otherwise it is disabled.

Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 21-3. Extended Fuse Byte

Fuse Extended Byte	Bit No	Description	Default Value
–	7	–	1 (unprogrammed)
–	6	–	1 (unprogrammed)
–	5	–	1 (unprogrammed)
–	4	–	1 (unprogrammed)
–	3	–	1 (unprogrammed)
–	2	–	1 (unprogrammed)
–	1	–	1 (unprogrammed)
SELFPRGEN	0	Self Programming Enable	1 (unprogrammed)

Table 21-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External Reset Disable	1 (unprogrammed)
DWEN	6	DebugWIRE Enable	1 (unprogrammed)
SPIEN(2)	5	Enable Serial Program and Data Downloading	0 (programmed, SPI programming enabled)
WDTON(3)	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(4)	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1(4)	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0(4)	0	Brown-out Detector trigger level	1 (unprogrammed)

Notes:
1. See ”Alternate Functions of Port B” on page 89. for description of RSTDISBL Fuse.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. See ”WDTCR – Watchdog Timer Control Register” on page 65. for details.
4. See Table 22-5 on page 254 for BODLEVEL Fuse coding.

Table 21-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8[4]	7	Divide clock by 8	0 (programmed)
CKOUT[3]	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[1]
SUT0	4	Select start-up time	0 (programmed)[1]
CKSEL3	3	Select Clock source	0 (programmed)[2]
CKSEL2	2	Select Clock source	0 (programmed)[2]
CKSEL1	1	Select Clock source	1 (unprogrammed)[2]
CKSEL0	0	Select Clock source	0 (programmed)[2]

Notes:
1. The default value of SUT[1:0] results in maximum start-up time for the default clock source. See Table 4-4 on page 35 for details.
2. The default setting of CKSEL[3:0] results in internal RC Oscillator @ 8 MHz. See Table 4-3 on page 35 for details.
3. The CKOUT Fuse allows the system clock to be output on PORTB5. See "Clock Output Buffer" on page 39. for details.
4. See "System Clock Prescaler" on page 45.for details.

21.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

21.3 Signature Bytes

All Microchip microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space.

Table 21-6. Signature Bytes

Device	Address	Value	Signature Byte Description
ATtiny87	0	0x1E	Indicates manufactured by Microchip
	1	0x93	Indicates 8 KB Flash memory
	2	0x87	Indicates ATtiny87 device when address 1 contains 0x93
ATtiny167	0	0x1E	Indicates manufactured by Microchip
	1	0x94	Indicates 16 KB Flash memory
	2	0x87	Indicates ATtiny167 device when address 1 contains 0x94

21.4 Calibration Byte

The ATtiny87/167 has a byte calibration value for the internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

21.5 Page Size

Table 21-7. Number of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATtiny87	4K words	64 words	PC[5:0]	64	PC[11:6]	11
ATtiny167	8K words	64 words	PC[5:0]	128	PC[12:6]	12

Table 21-8. Number of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATtiny87 ATtiny167	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

21.6 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATtiny87/167. Pulses are assumed to be at least 250 ns unless otherwise noted.

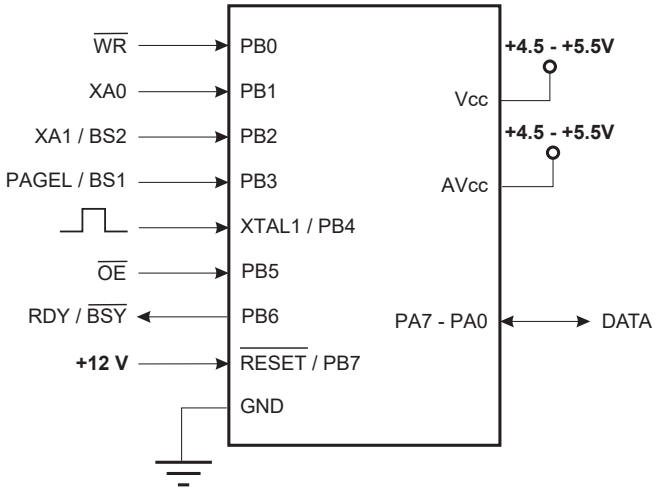
21.6.1 Signal Names

In this section, some pins of the ATtiny87/167 are referenced by signal names describing their functionality during parallel programming, see Figure 21-1 and Figure 21-9. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Figure 21-11.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different commands are shown in Figure 21-12.

Figure 21-1. Parallel programming



Note: V_{CC} - 0.3V < AV_{CC} < Vcc + 0.3V, however, AV_{CC} should always be within 4.5 - 5.5V

Table 21-9. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
$\overline{WR}$	PB0	I	Write Pulse (Active low).
XA0	PB1	I	XTAL1 Action Bit 0
XA1 / BS2	PB2	I	- XTAL1 Action Bit 1 - Byte Select 2 ("0" selects low byte, "1" selects 2'nd high byte)
PAGEL / BS1	PB3	I	- Program Memory and EEPROM data Page Load - Byte Select 1 ("0" selects low byte, "1" selects high byte)
⎍	PB4	I	XTAL1 (Clock input)
$\overline{OE}$	PB5	I	Output Enable (Active low).
RDY / $\overline{BSY}$	PB6	O	0: Device is busy programming, 1: Device is ready for new command.
+12V	PB7	I	- Reset (Active low) - Parallel programming mode (+12V).
DATA	PA[7:0]	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low).

Table 21-10. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PAGEL / BS1	Prog_enable[3]	0
XA1 / BS2	Prog_enable[2]	0
XA0	Prog_enable[1]	0
$\overline{WR}$	Prog_enable[0]	0

Table 21-11. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS1).
0	1	Load Data (High or Low data byte for Flash determined by BS1).
1	0	Load Command
1	1	No Action, Idle

Table 21-12. Command Byte Bit Coding

Command Byte	Command Executed
$1000\ 0000_b$	Chip Erase
$0100\ 0000_b$	Write Fuse bits
$0010\ 0000_b$	Write Lock bits
$0001\ 0000_b$	Write Flash
$0001\ 0001_b$	Write EEPROM
$0000\ 1000_b$	Read Signature bytes and Calibration byte
$0000\ 0100_b$	Read Fuse and Lock bits
$0000\ 0010_b$	Read Flash
$0000\ 0011_b$	Read EEPROM

21.7 Parallel Programming

21.7.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND.
2. Set RESET to “0” and toggle XTAL1 at least six times.
3. Set the Prog_enable pins listed in Table 21-10 on page 237 to “0000 $_b$” and wait at least 100 ns.
4. Apply 11.5 - 12.5V to RESET. Any activity on Prog_enable pins within 100 ns after +12V has been applied to RESET, will cause the device to fail entering programming mode.
5. Wait at least 50 µs before sending a new command.

21.7.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

21.7.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM(Note:) memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

Note: The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”

1. Set XA1, XA0 to “1,0 ”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000 $_b$”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

21.7.4 Programming the Flash

The Flash is organized in pages, see Table 21-7 on page 235. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “1,0”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000 $_b$”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

B. Load Address Low byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “0,1”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “0,1”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

E. Latch Data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (See Figure 21-3 for signal waveforms)

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 21-2 on page 240. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a Page Write.

G. Load Address High byte

1. Set XA1, XA0 to “0,0”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

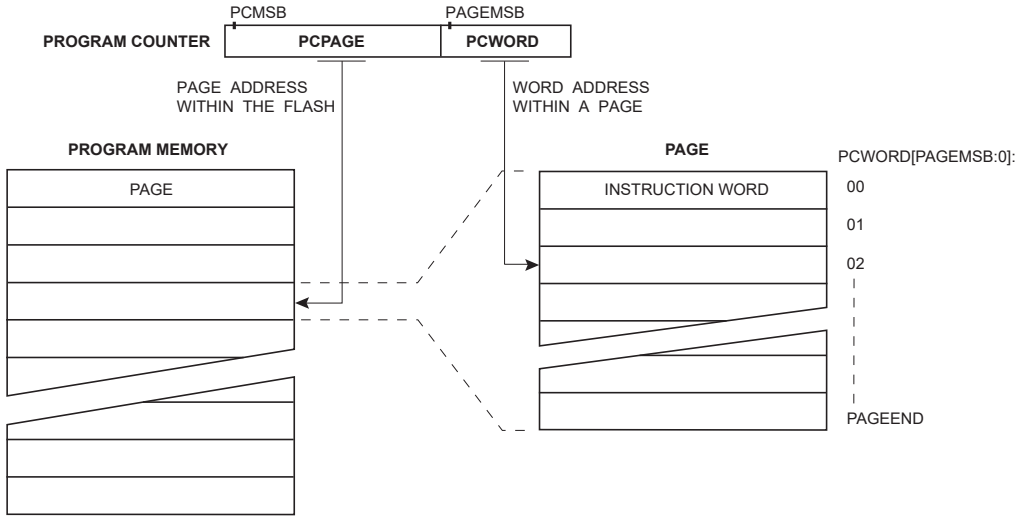
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (See Figure 21-3 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

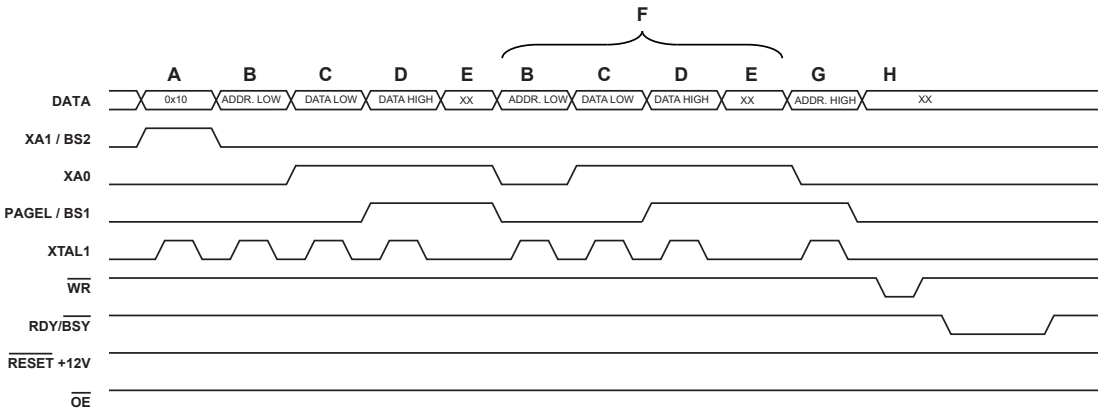
1. 1. Set XA1, XA0 to “1,0”. This enables command loading.
2. Set DATA to “0000 0000 $_b$”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 21-2. Addressing the Flash Which is Organized in Pages



Note: 1. PCPAGE and PCWORD are listed in

Figure 21-3. Programming the Flash Waveforms [1]



Note: 1. "XX" is don't care. The letters refer to the programming description above.

21.7.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 21-8 on page 235. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" on page 239 for details on Command, Address and Data loading):

A: Load Command "0001 0001 $_b$".

G: Load Address High Byte (0x00 - 0xFF).

B: Load Address Low Byte (0x00 - 0xFF).

C: Load Data (0x00 - 0xFF).

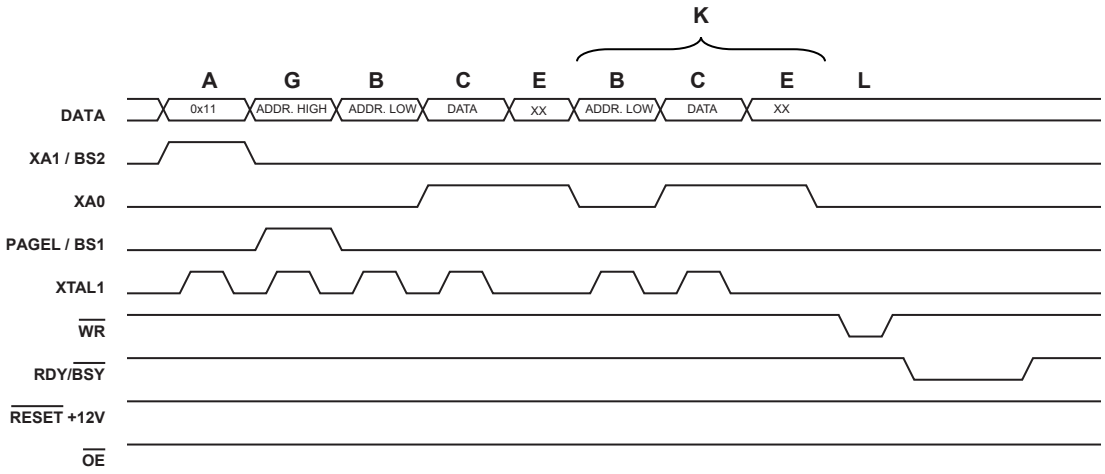
E: Latch data (give PAGEL a positive pulse).

K: Repeat A through E until the entire buffer is filled.

L: Program EEPROM page

1. Set BS1 to "0".
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 21-4 for signal waveforms).

Figure 21-4. Programming the EEPROM Waveforms



21.7.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 239 for details on Command and Address loading):

1. **A:** Load Command “0000 0010 $_b$”.
2. **G:** Load Address High Byte (0x00 - 0xFF).
3. **B:** Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The Flash word low byte can now be read at DATA.
5. Set BS1 to “1”. The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

21.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 239 for details on Command and Address loading):

1. **A:** Load Command “0000 0011 $_b$”.
2. **G:** Load Address High Byte (0x00 - 0xFF).
3. **B:** Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

21.7.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 239 for details on Command and Data loading):

1. **A:** Load Command “0100 0000 $_b$”.
2. **C:** Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

21.7.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 239 for details on Command and Data loading):

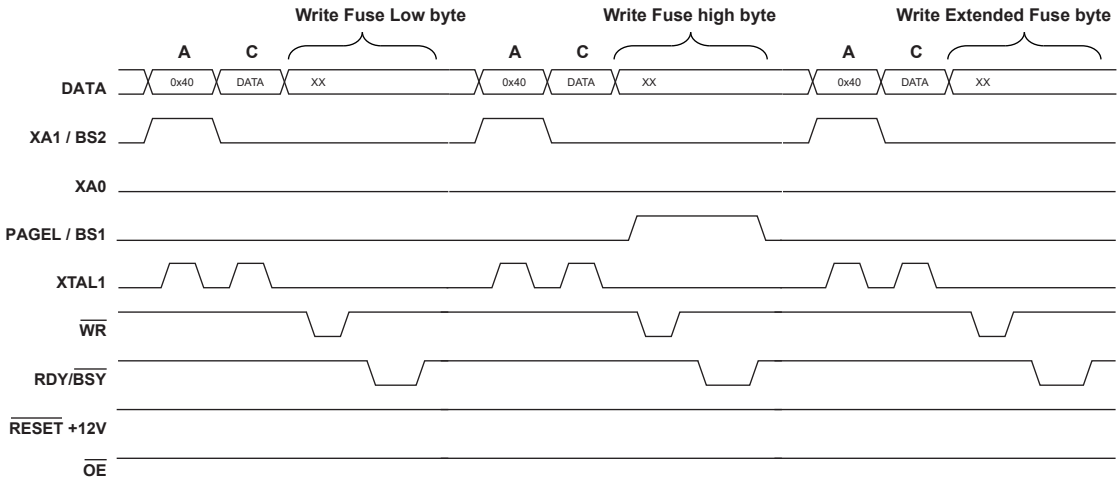
1. **A:** Load Command “0100 0000$_b$”.
2. **C:** Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

21.7.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to “Programming the Flash” on page 239 for details on Command and Data loading):

1. **A:** Load Command “0100 0000$_b$”.
2. **C:** Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2 to “0”. This selects low data byte.

Figure 21-5. Programming the FUSES Waveforms



21.7.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 239 for details on Command and Data loading):

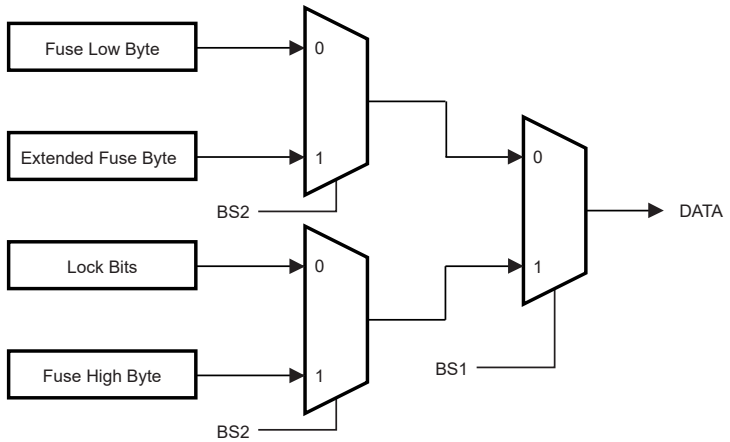
1. **A:** Load Command “0010 0000$_b$”.
2. **C:** Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to re-program the Lock bits by any External Programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
4. The Lock bits can only be cleared by executing Chip Erase.

21.7.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 239 for details on Command loading):

1. **A:** Load Command "0000 0100 $_b$".
2. Set $\overline{OE}$ to "0", BS2 to "0" and BS1 to "0". The status of the Fuse Low bits can now be read at DATA ("0" means programmed).
3. Set $\overline{OE}$ to "0", BS2 to "1" and BS1 to "1". The status of the Fuse High bits can now be read at DATA ("0" means programmed).
4. Set OE to "0", BS2 to "1", and BS1 to "0". The status of the Extended Fuse bits can now be read at DATA ("0" means programmed).
5. Set $\overline{OE}$ to "0", BS2 to "0" and BS1 to "1". The status of the Lock bits can now be read at DATA ("0" means programmed).
6. Set $\overline{OE}$ to "1".

Figure 21-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



21.7.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

1. **A:** Load Command "0000 1000 $_b$".
2. **B:** Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to "0", and BS to "0". The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

21.7.14 Reading the 8 MHz RC Oscillator Calibration Byte

The algorithm for reading the 8 MHz RC Oscillator Calibration byte is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

1. **A:** Load Command "0000 1000 $_b$".
2. **B:** Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to "0", and BS1 to "1". The 8 MHz RC Oscillator Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

21.7.15 Reading the Temperature Sensor Parameter Bytes

The algorithm for reading the Temperature Sensor parameter bytes is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

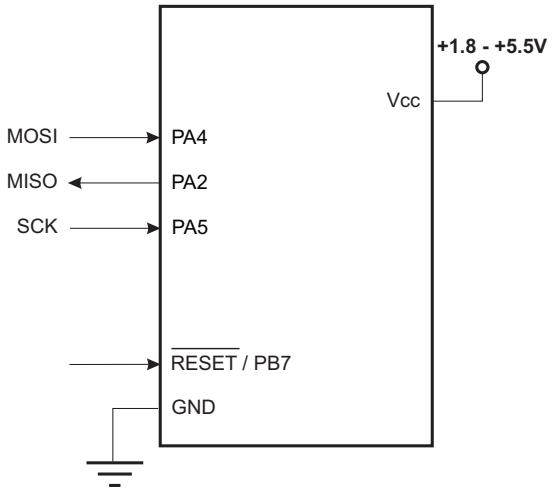
1. **A:** Load Command "0000 1000 $_b$".
2. **B:** Load Address Low Byte, 0x0003 or 0x0005.
3. Set $\overline{OE}$ to "0", and BS1 to "1". The Temperature Sensor parameter byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

21.8 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Note: In Table 21-13, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 21-7. Serial Programming and Verify (Note:)



Note: If the device is clocked by the internal Oscillator, it is no need to connect a clock source to the XTAL1 pin

Table 21-13. Pin Mapping Serial Programming

Symbol	Pin Name	I/O	Function
MOSI	PA4	I	Serial Data In
MISO	PA2	O	Serial Data Out
SCK	PA5	I	Serial Clock

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode **ONLY**) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > **2** CPU clock cycles for f_{ck} < 12 MHz, **3** CPU clock cycles for f_{ck} >= 12 MHz

High: > **2** CPU clock cycles for f_{ck} < 12 MHz, **3** CPU clock cycles for f_{ck} >= 12 MHz

21.8.1 Serial Programming Algorithm

When writing serial data to the ATtiny87/167, data is clocked on the rising edge of SCK.

When reading data from the ATtiny87/167, data is clocked on the falling edge of SCK. See Figure 21-7 and Figure 21-8 for timing details.

To program and verify the ATtiny87/167 in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 21-15 on page 247):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the Load Program memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program memory Page is stored by loading the Write Program memory Page instruction with the 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 21-14) Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 21-14) In a chip erased device, no 0x$\overline{\text{FF}}$s in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The Memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the Load EEPROM Memory Page instruction. The EEPROM Memory Page is stored by loading the Write EEPROM Memory Page Instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM Memory Page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 21-8). In a chip erased device, no 0x$\overline{\text{FF}}$ in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.

7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

Table 21-14. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	4.0 ms
t_{WD_ERASE}	9.0 ms
t_{WD_FUSE}	4.5 ms

21.8.2 Serial Programming Instruction set

Table 21-15 on page 247 and Figure 21-8 on page 249 describes the Instruction set

Table 21-15. Serial Programming Instruction Set

	Instruction Format			
Instruction/Operation	**Byte 1**	**Byte 2**	**Byte 3**	**Byte4**
Programming Enable	0xAC	0x53	0x00	0x00
Chip Erase (Program Memory/EEPROM)	0xAC	0x80	0x00	0x00
Poll RDY/$\overline{\text{BSY}}$	0xF0	0x00	0x00	data byte out
Load Instructions				
Load Extended Address byte[1]	0x4D	0x00	Extended add.	0x00
Load Program Memory Page, High byte	0x48	add. MSB	add. LSB	high data byte in
Load Program Memory Page, Low byte	0x40	add. MSB	add. LSB	low data byte in
Load EEPROM Memory Page (page access)	0xC1	0x00	0000 000aa $_b$	data byte in
Read Instructions				
Read Program Memory, High byte	0x28	add. MSB	add. LSB	high data byte out
Read Program Memory, Low byte	0x20	add. MSB	add. LSB	low data byte out
Read EEPROM Memory	0xA0	0x00	00aa aaaa	data byte out
Read Lock bits	0x58	0x00	0x00	data byte out
Read Signature Byte	0x30	0x00	0000 000aa	data byte out
Read Fuse bits	0x50	0x00	0x00	data byte out
Read Fuse High bits	0x58	0x08	0x00	data byte out
Read Extended Fuse Bits	0x50	0x08	0x00	data byte out
Read Calibration Byte	0x38	0x00	0x00	data byte out
Write Instructions[6]				
Write Program Memory Page	0x4C	add. MSB	add. LSB	0x00

Table 21-15. Serial Programming Instruction Set (Continued)

Instruction/Operation	Instruction Format			
	Byte 1	**Byte 2**	**Byte 3**	**Byte4**
Write EEPROM Memory	0xC0	0x00	00aa aaaa $_b$	data byte in
Write EEPROM Memory Page (page access)	0xC2	0x00	00aa aa00 $_b$	0x00
Write Lock bits	0xAC	0xE0	0x00	data byte in
Write Fuse bits	0xAC	0xA0	0x00	data byte in
Write Fuse High bits	0xAC	0xA8	0x00	data byte in
Write Extended Fuse Bits	0xAC	0xA4	0x00	data byte in

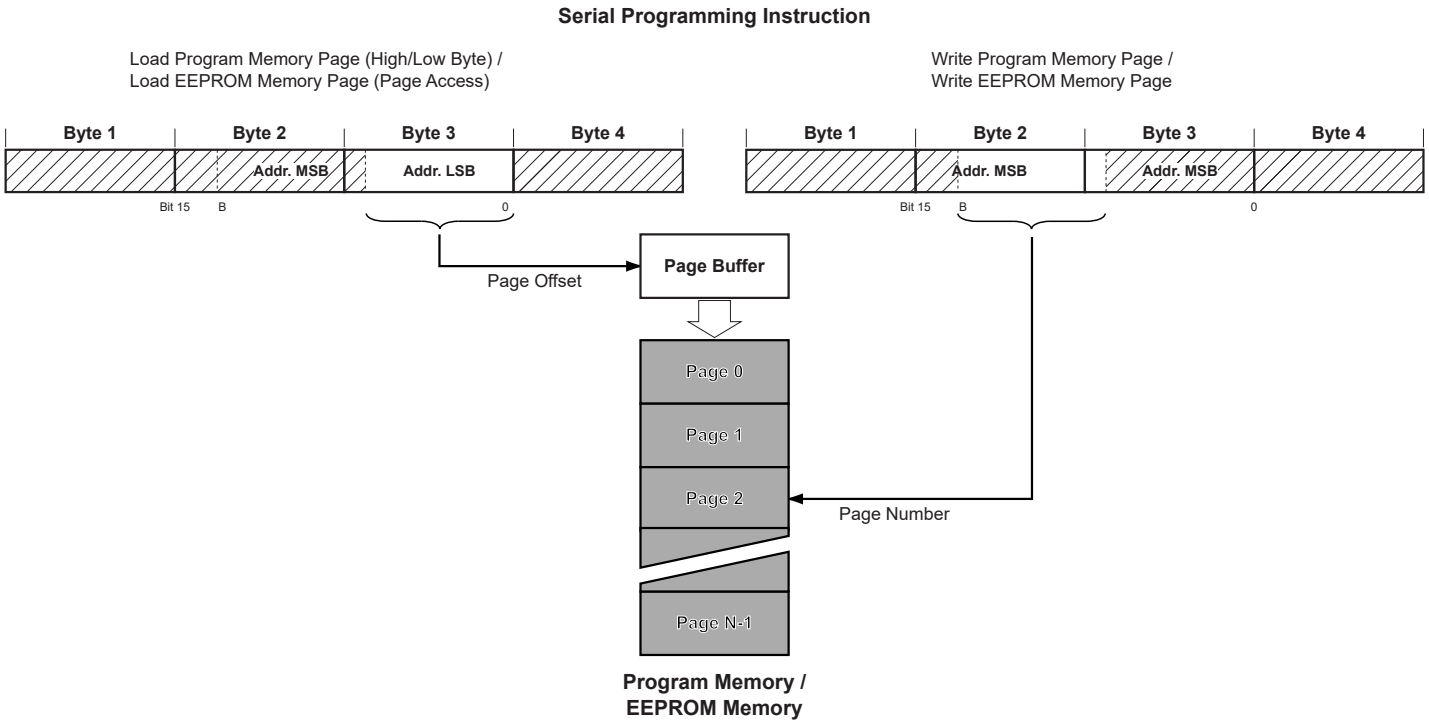
Notes:
1. Not all instructions are applicable for all parts.
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’).
5. Refer to the corresponding section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
6. Instructions accessing program memory use a word address. This address may be random within the page range.
7. See https://www.microchip.com for Application Notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

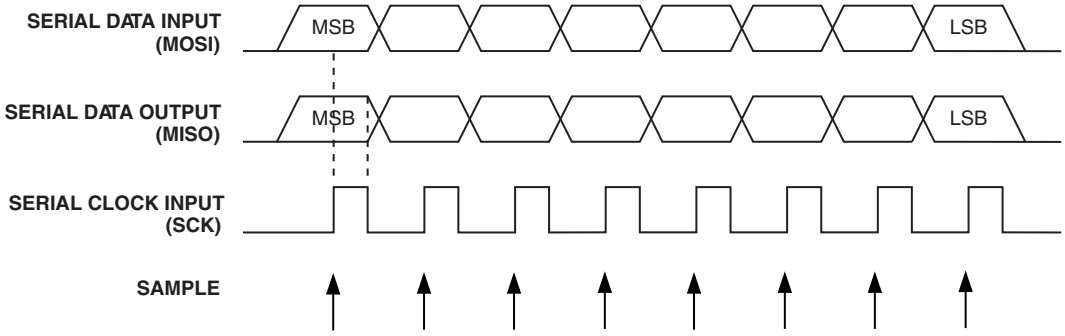
After data is loaded to the page buffer, program the EEPROM page, see Figure 21-8 on page 249.

Figure 21-8. Serial programming Instruction Example



21.9 Serial Programming Characteristics

Figure 21-9. Serial Programming Waveforms



For characteristics of the SPI module, See ”SPI Timing Characteristics” on page 260.

22. Electrical Characteristics

22.1 Absolute Maximum Ratings*

Operating Temperature.....– 40°C to +85°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 Voltage on V_{CC} with respect to Ground .....– 0.5V to 6.0V DC Current per I/O Pin ..... 40.0 mA DC Current V_{CC} and GND Pins ..... 200.0 mA Injection Current at VCC = 0V to 5V(2).....±5.0mA(1)	*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Notes: 1. Maximum current per port = ±30mA
2. Functional corruption may occur .

22.2 DC Characteristics

T_A = -40°C to +85°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 $\overline{\text{RESET}}$ pins	- 0.5		0.2 V_{CC} (2)	V
V_{IL1}		XTAL1 pin - External Clock Selected	- 0.5		0.1 V_{CC} (2)	V
V_{IL2}		$\overline{\text{RESET}}$ pin	- 0.5		0.2 V_{CC} (2)	V
V_{IL3}		$\overline{\text{RESET}}$ pin as I/O	- 0.5		0.2 V_{CC} (2)	V
V_{IH}	Input High Voltage	Except XTAL1 and $\overline{\text{RESET}}$ pins	0.7 V_{CC} (3)		V_{CC} + 0.5	V
V_{IH1}		XTAL1 pin - External Clock Selected	0.8 V_{CC} (3)		V_{CC} + 0.5	V
V_{IH2}		$\overline{\text{RESET}}$ pin	0.9 V_{CC} (3)		V_{CC} + 0.5	V
V_{IH3}		$\overline{\text{RESET}}$ pin as I/O	0.7 V_{CC} (3)		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage (4) (Ports A, B,)	I_{OL} = 10 mA, V_{CC} = 5V I_{OL} = 5 mA, V_{CC} = 3V			0.6 0.5	V
V_{OH}	Output High Voltage (5) (Ports A, B)	I_{OH} = – 10 mA, V_{CC} = 5V I_{OH} = – 5 mA, V_{CC} = 3V	4.3 2.5			V
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

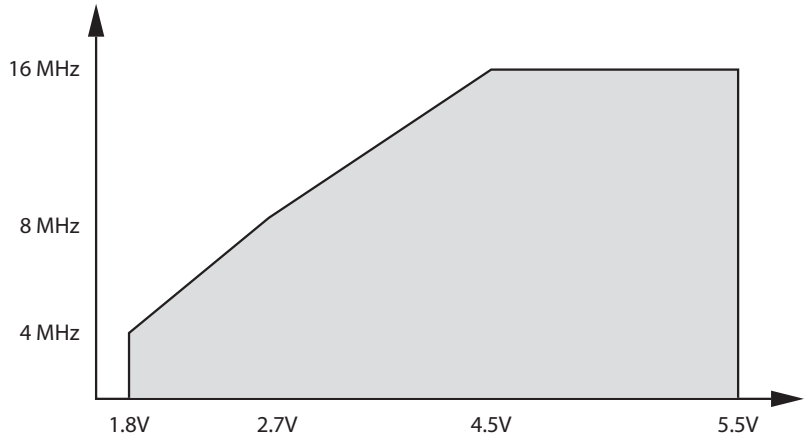
T_A = -40°C to +85°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min.	Typ.(1)	Max.	Units
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{pu}	I/O Pin Pull-up Resistor		20		50	kΩ
I_{CC}	Power Supply Current (6) Active Mode (external clock)	8 MHz, V_{CC} = 5V		5.5	7.0	mA
		4 MHz, V_{CC} = 3V		1.8	2.5	mA
		1 MHz, V_{CC} = 2V		0.3	0.6	mA
	Power Supply Current (6) Idle Mode (external clock)	8 MHz, V_{CC} = 5V		1.8	2.5	mA
		4 MHz, V_{CC} = 3V		0.5	0.8	mA
		1 MHz, V_{CC} = 2V		0.05	0.2	mA
	Power Supply Current (7) Power-down Mode	WDT enabled, V_{CC} = 3V		5	10	µA
		WDT disabled, V_{CC} = 3V		0.15	4	µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2	-10	10	40	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2	-50		50	nA
t_{ACID}	Analog Comparator Propagation Delay Common Mode V_{CC}/2	V_{CC} = 2.7V		170		ns
		V_{CC} = 5.0V		180		ns

Notes:
1. “Typ.", typical values at 25°C. Maximum values are characterized values and not test limits in production.
2. "Max.” means the highest value where the pin is ensured to be read as low.
3. “Min.” means the lowest value where the pin is ensured 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 following must be observed:
   - The sum of all IOL, for all ports, should not exceed 120 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
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 following must be observed:
   - The sum of all IOH, for all ports, should not exceed 120 mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
6. Values are with external clock using methods described in “Minimizing Power Consumption” on page 53. Power Reduction is enabled (PRR = 0xFF) and there is no I/O drive.
7. BOD Disabled.

22.3 Speed

Figure 22-1. Maximum Frequency vs. V_{CC}, ATtiny87/167



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. Voltage and temperature characteristics can be found in Figure 23-29 on page 277.

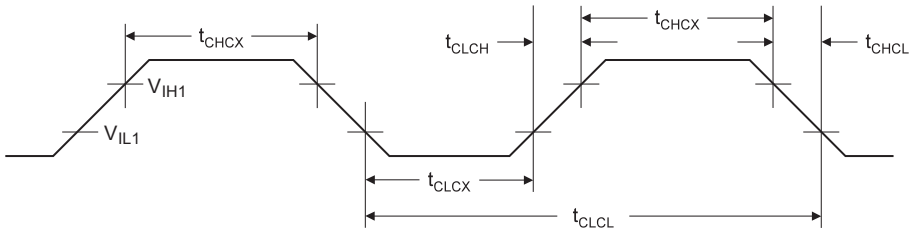
Table 22-1. Calibration Accuracy of Internal Oscillator

Calibration Method	Target Frequency	V_{CC}	Temperature	Accuracy at given voltage & temperature (1)
Factory Calibration	8.0 MHz	3V	25°C	±10%
User Calibration	Fixed frequency within: 7.3 - 8.1 MHz	Fixed voltage within: 1.8 - 5.5V	Fixed temperature within: -40°C to +85°C	±1%

Notes: 1. Accuracy of oscillator frequency at calibration point (fixed temperature and fixed voltage).

22.4.2 External Clock Drive Waveforms

Figure 22-2. External Clock Drive Waveforms



22.4.3 External Clock Drive

Table 22-2. External Clock Drive

Symbol	Parameter	Vcc = 1.8 - 5.5V		Vcc = 2.7 - 5.5V		Vcc = 4.5 - 5.5V		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	8	0	16	MHz
t_{CLCL}	Clock Period	250		125		62.5		ns
t_{CHCX}	High Time	100		50		25		ns
t_{CLCX}	Low Time	100		50		25		ns
t_{CLCH}	Rise Time		2.0		1.6		0.5	µs
t_{CHCL}	Fall Time		2.0		1.6		0.5	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2		2	%

22.5 RESET Characteristics

Table 22-3. External Reset Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage	V_{CC} = 5V	0.2 V_{CC}		0.9 V_{CC}	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin	V_{CC} = 5V			2.5	µs
V_{BG}	Bandgap reference voltage	V_{CC} = 2.7V, T_A = 25°C	1.0	1.1	1.2	V
t_{BG}	Bandgap reference start-up time	V_{CC} = 2.7V, T_A = 25°C		40	70	µs
I_{BG}	Bandgap reference current consumption	V_{CC} = 2.7V, T_A = 25°C		15		µA

Table 22-4. Power On Reset Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)		1.4		V
	Power-on Reset Threshold Voltage (falling)(1)	1.0	1.3	1.6	V
V_{PORMAX}	VCC Max. start voltage to ensure internal Power-on Reset signal			0.4	V
V_{PORMIN}	VCC Min. start voltage to ensure internal Power-on Reset signal	- 0.1			V
V_{CCRR}	VCC Rise Rate to ensure Power-on Reset	0.01			V/ms

Note: 1. Before rising, the supply has to be between VPORMIN and VPORMAX to ensure a Reset.

Table 22-5. BODLEVEL Fuse Coding

BODLEVEL[2:0] Fuses	Min. V_{BOT}[1]	Typ. V_{BOT}	Max. V_{BOT}	Units
$1\,1\,1_b$	BOD Disabled			
$1\,1\,0_b$	1.7	1.8	2.0	V
$1\,0\,1_b$	2.5	2.7	2.9	
$1\,0\,0_b$	4.1	4.3	4.5	
$0\,1\,1_b$	Reserved			
$0\,1\,0_b$				
$0\,0\,1_b$				
$0\,0\,0_b$				

Notes: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This ensures that a Brown-Out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 101 for Low Operating Voltage and BODLEVEL = 100 for High operating Voltage.

Table 22-6. Brown-out Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{HYST}	Brown-out Detector Hysteresis		80		mV
t_{BOD}	Min Pulse Width on Brown-out Reset		2		µs

22.6 Internal Voltage Characteristics

Table 22-7. Internal Voltage Reference Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{BG}	Bandgap reference voltage	V_{CC} = 4.5 T_A = 25°C	1.0	1.1	1.2	V
t_{BG}	Bandgap reference start-up time	V_{CC} = 4.5 T_A = 25°C		40	70	µs
I_{BG}	Bandgap reference current consumption	V_{CC} = 4.5 T_A = 25°C		10		µA

22.7 Current Source Characteristics

Table 22-8. Current Source Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{ISRC}	Current	V_{CC} = 2.7 V / 5.5 V T = -40°C / +85°C	94		106	µA
t_{ISRC}	Current Source start-up time	V_{CC} = 4.5 T_A = 25°C		60		µs

22.8 ADC Characteristics

Table 22-9. ADC Characteristics, Single Ended Channels (-40°C/+85°C)

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution	Single Ended Conversion		10		Bits
TUE	Absolute accuracy	V_{CC} = 4V, VRef = 4V, ADC clock = 200 kHz		2.0	3.5	LSB
INL	Integral Non Linearity	V_{CC} = 4V, VRef = 4V, ADC clock = 200 kHz		0.6	2.0	LSB
DNL	Differential Non Linearity	V_{CC} = 4V, VRef = 4V, ADC clock = 200 kHz		0.3	0.8	LSB
	Gain error	V_{CC} = 4V, VRef = 4V, ADC clock = 200 kHz	-6.0	-2.5	2.0	LSB
	Offset error	V_{CC} = 4V, VRef = 4V, ADC clock = 200 kHz	-3.5	1.5	3.5	LSB
A_{REF}	External voltage reference	Single Ended Conversions	2.0		AV_{CC}	V
	Input bandwidth			38.5		kHz
V_{INT}	Internal 1.1V reference		1.0	1.1	1.2	V
	Internal 2.56V reference	V_{CC} > 3.0V	2.3	2.56	2.8	V
R_{REF}	Reference input resistance			32		kΩ
R_{AIN}	Analog input resistance			100		MΩ

Table 22-10. ADC Characteristics, Differential Channels (-40°C/+85°C)

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution	Differential conversion		8		
TUE	Absolute accuracy	Gain = 8x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		1.0	3.0	LSB
		Gain = 20x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		1.5	3.5	
		Gain = 8x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		2.0	4.5	
		Gain = 20x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		2.0	6.0	
INL	Integral Non Linearity	Gain = 8x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.2	1.0	LSB
		Gain = 20x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.4	1.5	
		Gain = 8x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.5	2.0	
		Gain = 20x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		1.6	5.0	
DNL	Differential Non Linearity	Gain = 8x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.3	0.8	LSB
		Gain = 20x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.3	0.8	
		Gain = 8x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.4	0.8	
		Gain = 20x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz		0.6	1.6	
	Gain error	Gain = 8x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz	-3.0	1.0	3.0	LSB
		Gain = 20x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz	-4.0	1.5	4.0	
		Gain = 8x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz	-5.0	-2.5	0.0	
		Gain = 20x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz	-4.0	-0.5	4.0	

Table 22-10. ADC Characteristics, Differential Channels (-40°C/+85°C) (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Offset error	Gain = 8x or 20x, BIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz	-2.0	0.5	2.0	LSB
		Gain = 8x or 20x, UNIPOLAR V_{REF} = 4V, V_{CC} = 5V ADC clock = 200 kHz	-2.0	0.5	2.0	
A_{REF}	External Reference voltage	Differential conversion	2.0		AV_{CC} - 1.0	V
VDIFF	Input differential voltage		-VREF/Gain		+VREF/Gain	V
AV_{CC}	Analog supply voltage		V_{CC} - 0.3		V_{CC} + 0.3	V
VIN	Input voltage	Differential conversion	0		AV_{CC}	V
	ADC conversion output		-511		+511	LSB
	Input bandwidth	Differential conversion		4		kHz
V_{INT}	Internal 1.1V reference		1.0	1.1	1.2	V
	Internal 2.56V reference	V_{CC} > 3.0V	2.3	2.56	2.8	V
RREF	Reference input resistance			32		kΩ
RAIN	Analog input resistance			100		MΩ

22.9 Parallel Programming Characteristics

Figure 22-3. Parallel Programming Timing, Including some General Timing Requirements

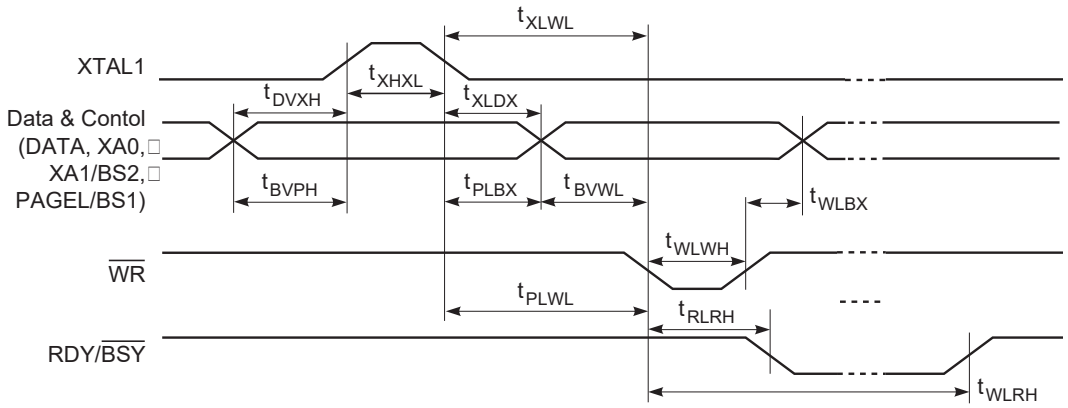
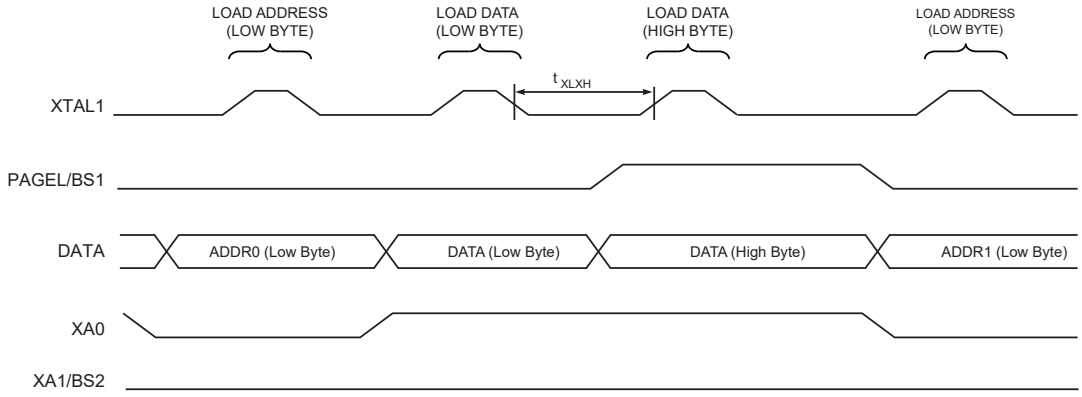
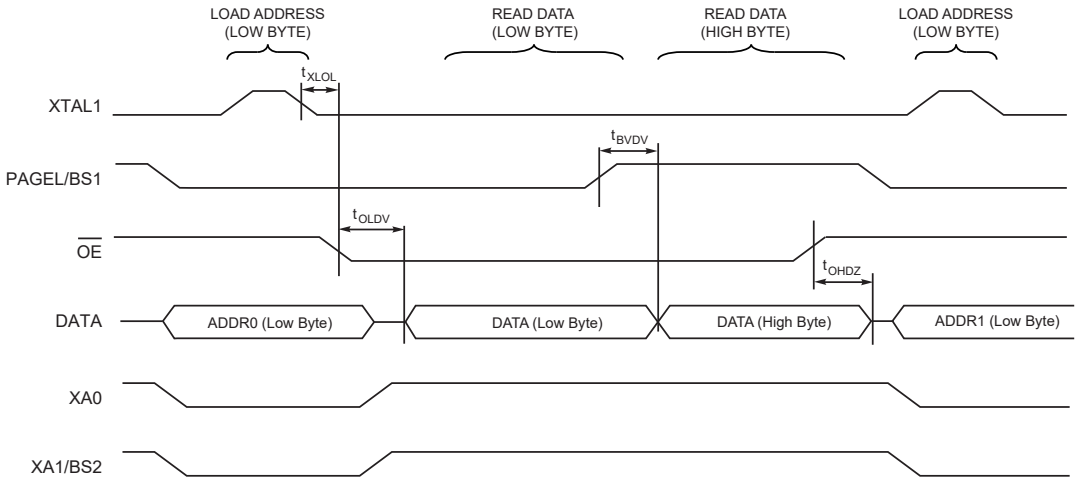


Figure 22-4. Parallel Programming Timing, Loading Sequence with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 22-3 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 22-5. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements(Note:)



Note: The timing requirements shown in Figure 22-3 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 22-11. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High(1)	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase(2)	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

22.10 SPI Timing Characteristics

See Figure 22-6 and Figure 22-7 for details.

Table 22-12. SPI Timing Parameters

	Description	Mode	Min.	Typ.	Max.	
1	SCK period	Master		See Table 13-4		
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t_{sck}		ns
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		
10	SCK period	Slave	4 • t_{ck}			
11	SCK high/low (1)	Slave	2 • t_{ck}			
12	Rise/Fall time	Slave			1.6	µs
13	Setup	Slave	10			
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		ns
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	SS low to SCK	Slave	2 • t_{ck}			

Note: In SPI Programming mode the minimum SCK high/low period is:
- 2 t_{CLCL} for f_{CK} < 12 MHz
- 3 t_{CLCL} for f_{CK} >12 MHz

Figure 22-6. SPI Interface Timing Requirements (Master Mode)

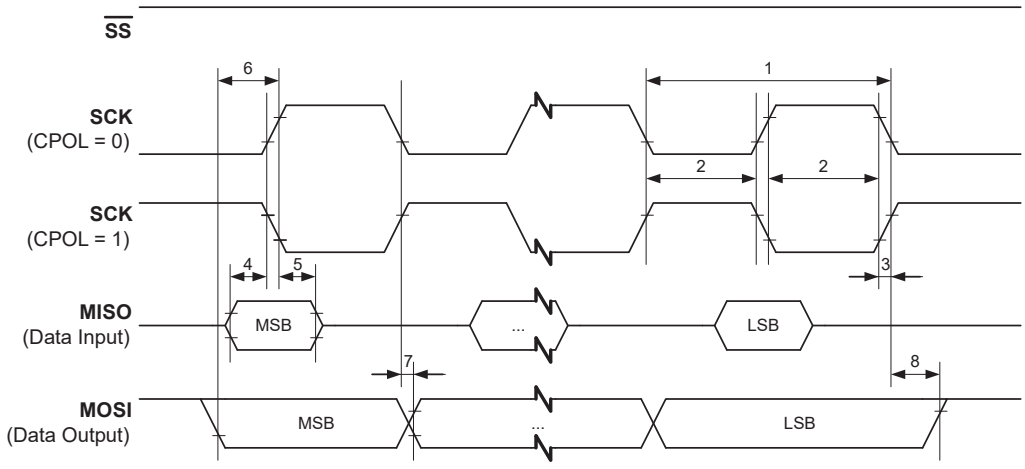
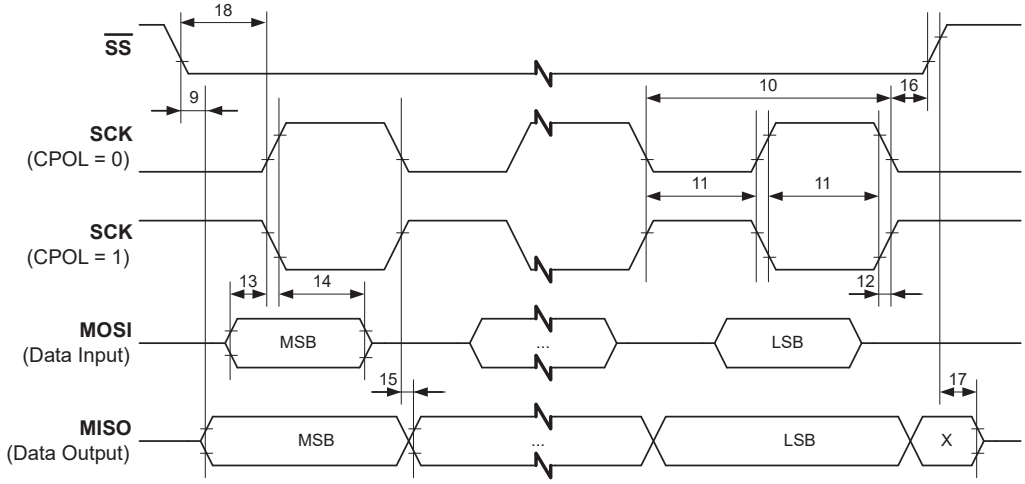


Figure 22-7. SPI Interface Timing Requirements (Slave Mode)



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. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

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

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

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

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

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

23.1 Current Consumption in Active Mode

Figure 23-1. Active Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

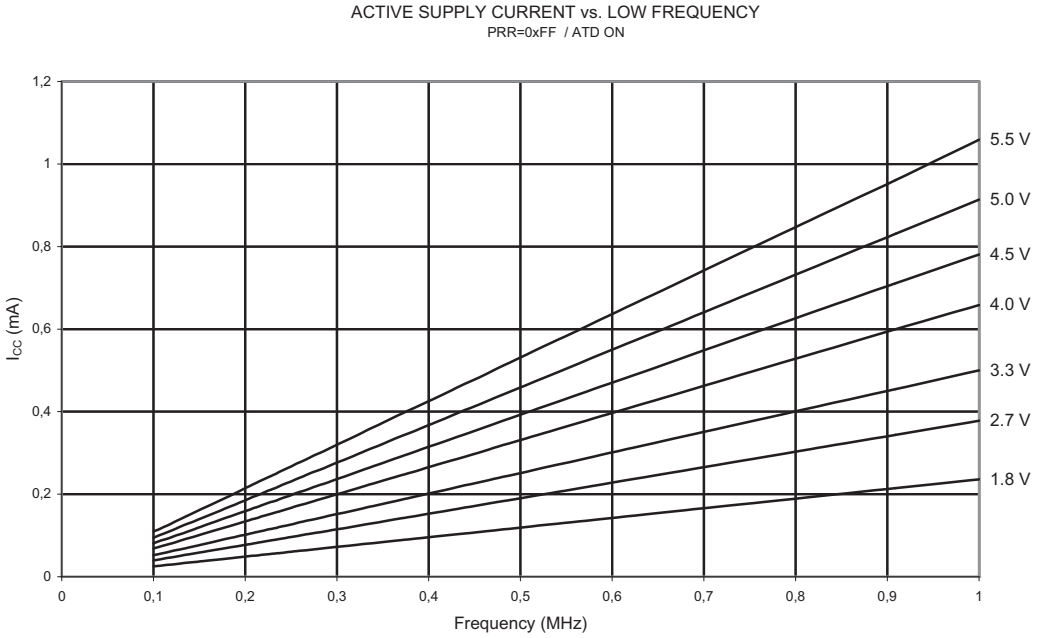


Figure 23-2. Active Supply Current vs. Frequency (1 - 16 MHz)

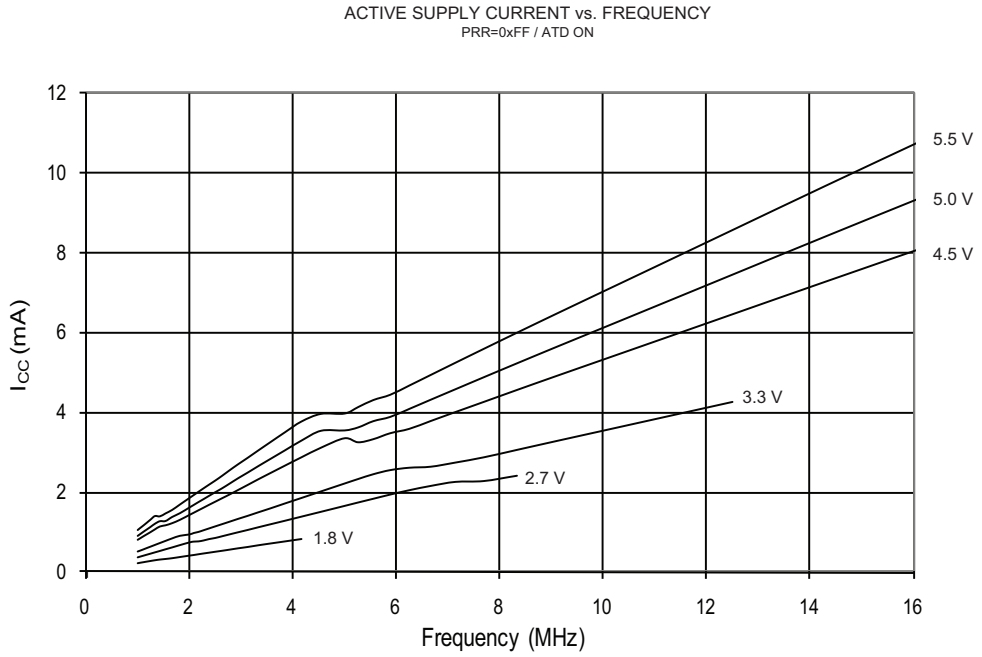


Figure 23-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

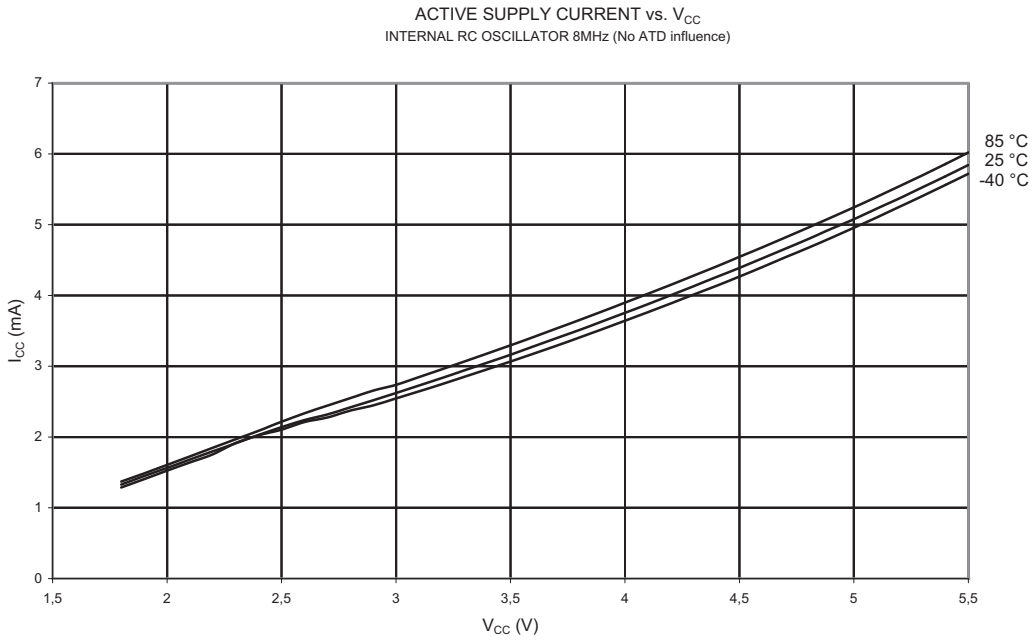
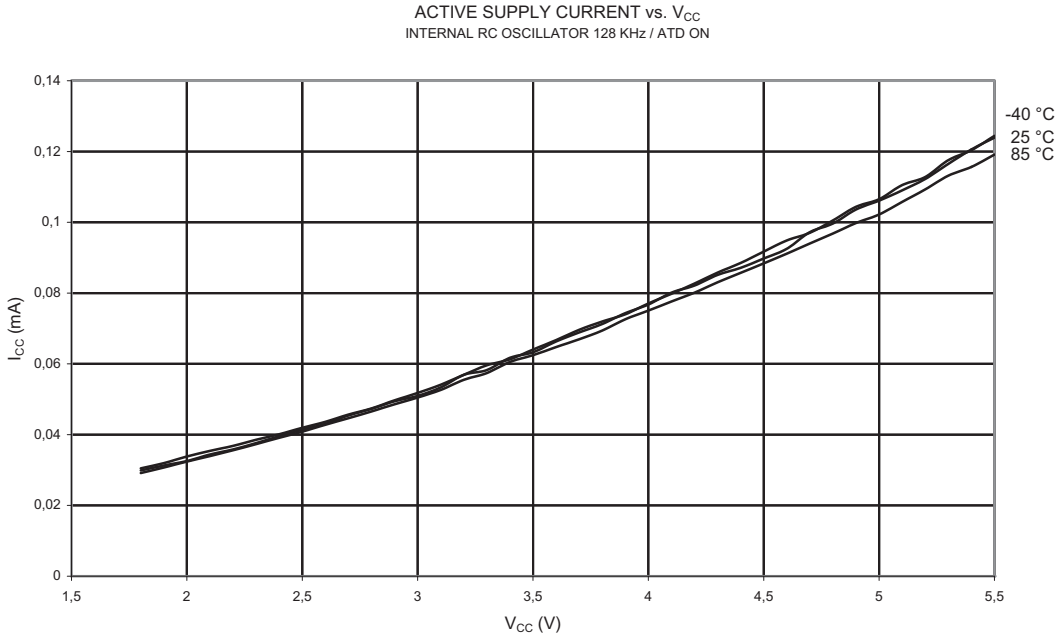


Figure 23-4. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 128 kHz)



23.2 Current Consumption in Idle Mode

Figure 23-5. Idle Supply Current vs. Frequency (1 - 16 MHz)

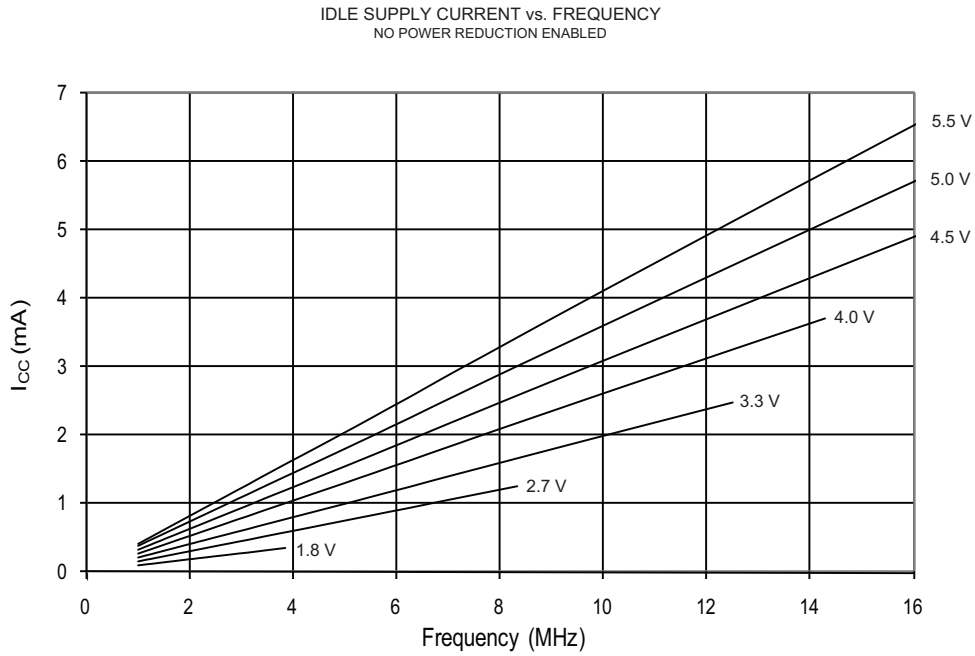


Figure 23-6. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

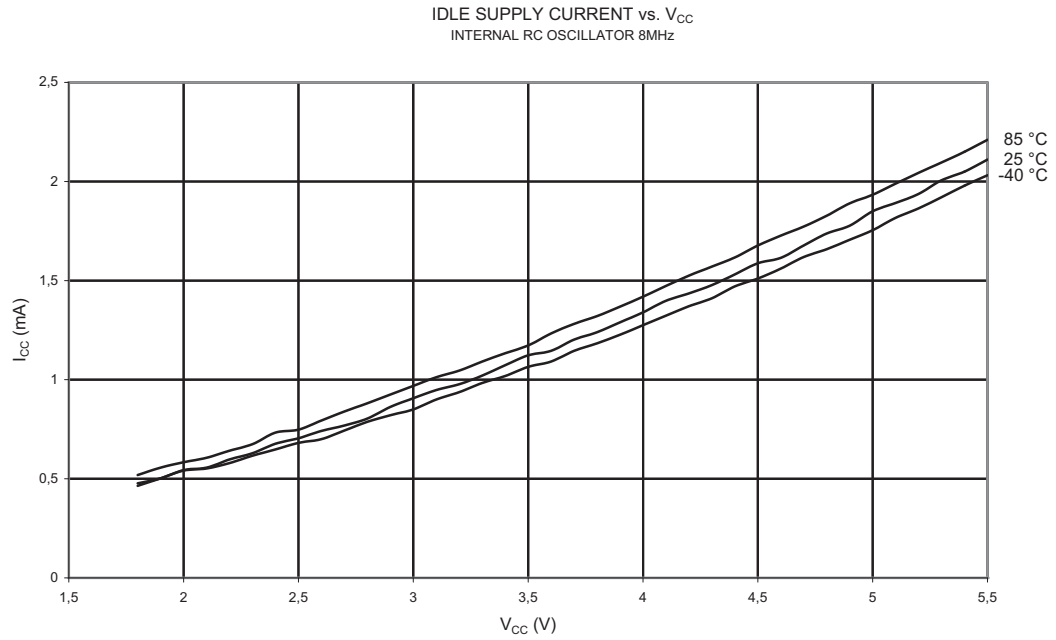
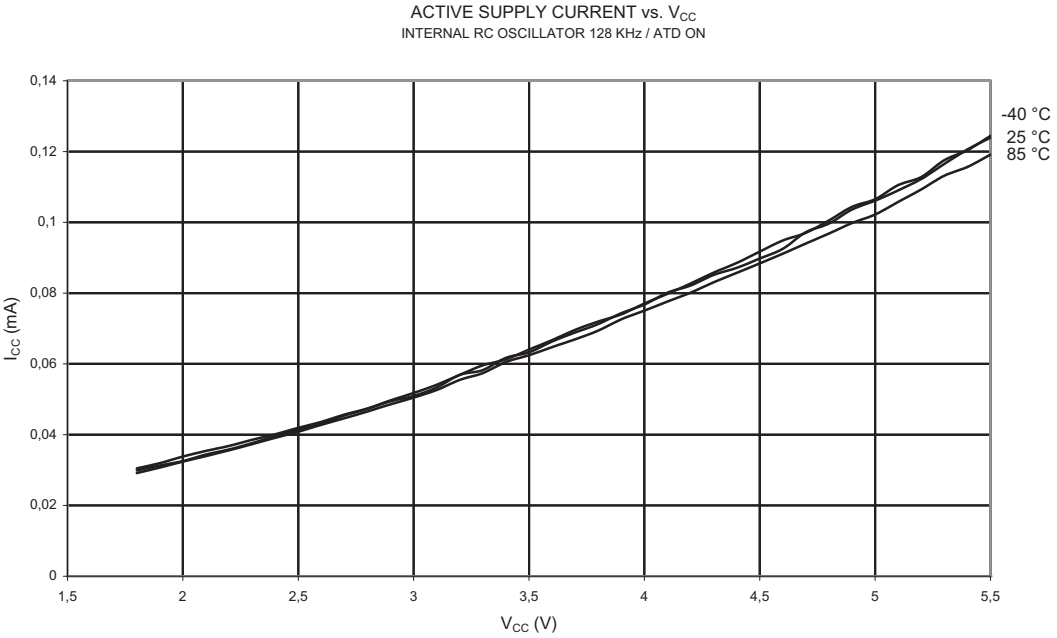


Figure 23-7. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 128 kHz)



23.3 Supply Current of I/O modules

The table below can be used to calculate the additional current consumption for the different I/O modules Idle mode. The enabling or disabling of the I/O modules are controlled by the Power Reduction Register. See Section 5.9.3 “PRR – Power Reduction Register” on page 55 for details.

Table 23-1. Additional Current Consumption for the different I/O modules (absolute values)

Module	V_{CC} = 5.0 V Freq. = 16 MHz	V_{CC} = 5.0 V Freq. = 8 MHz	V_{CC} = 3.0 V Freq. = 8 MHz	V_{CC} = 3.0 V Freq. = 4 MHz	Units
LIN/UART	0.77	0.37	0.20	0.10	mA
SPI	0.31	0.14	0.08	0.04	mA
TIMER-1	0.28	0.13	0.08	0.04	mA
TIMER-0	0.41	0.20	0.10	0.05	mA
USI	0.14	0.05	0.04	0.02	mA
ADC	0.48	0.22	0.10	0.05	mA

23.4 Current Consumption in Power-down Mode

Figure 23-8. Power-down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

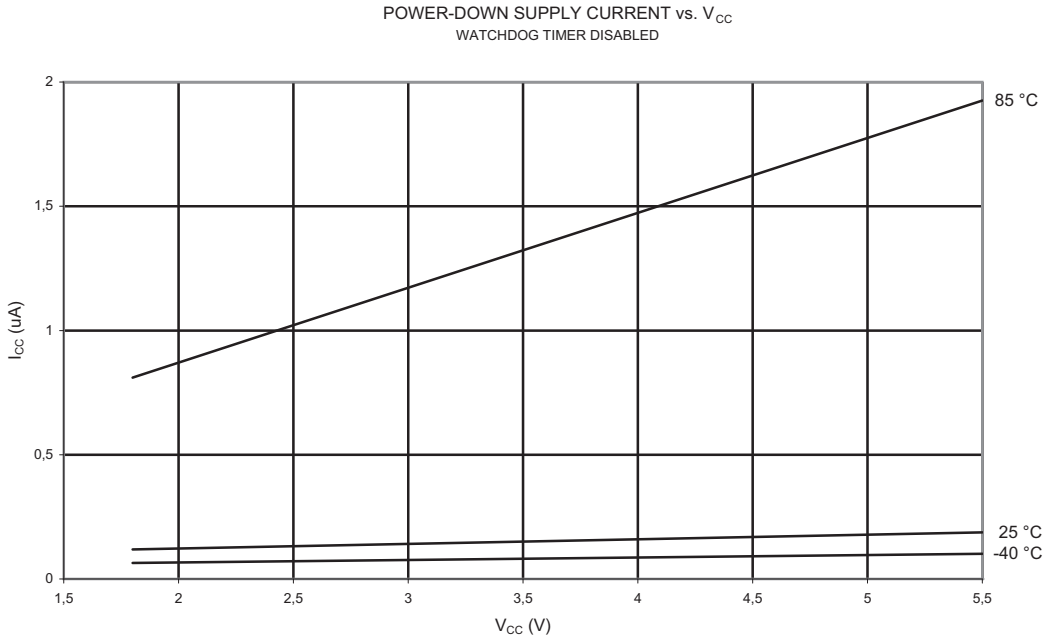
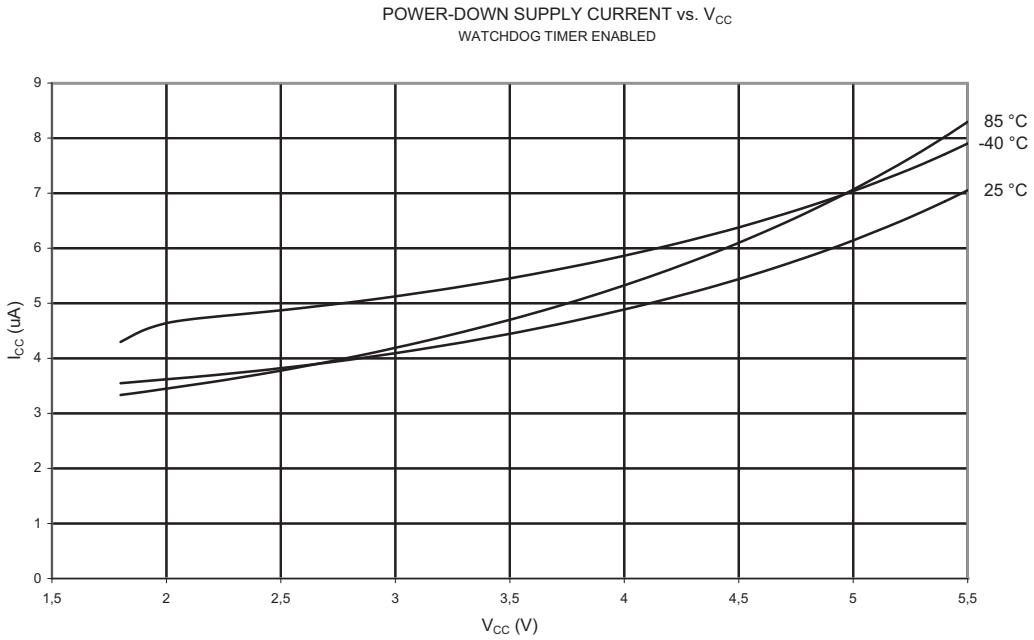


Figure 23-9. Power-down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



23.5 Current Consumption in Reset

Figure 23-10. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, Excluding Current Through the Reset Pull-up)

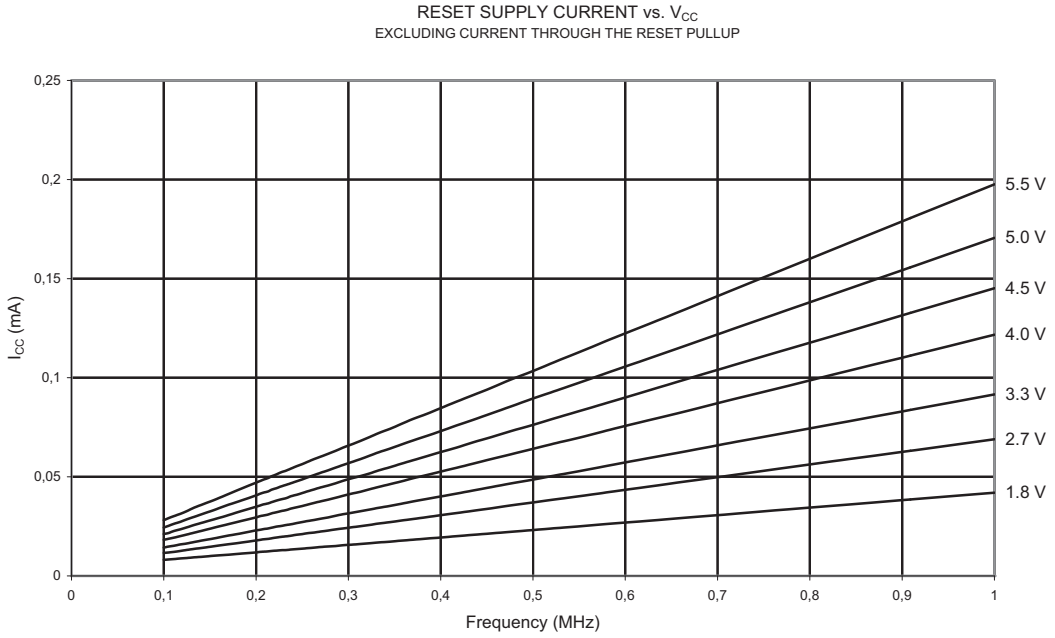
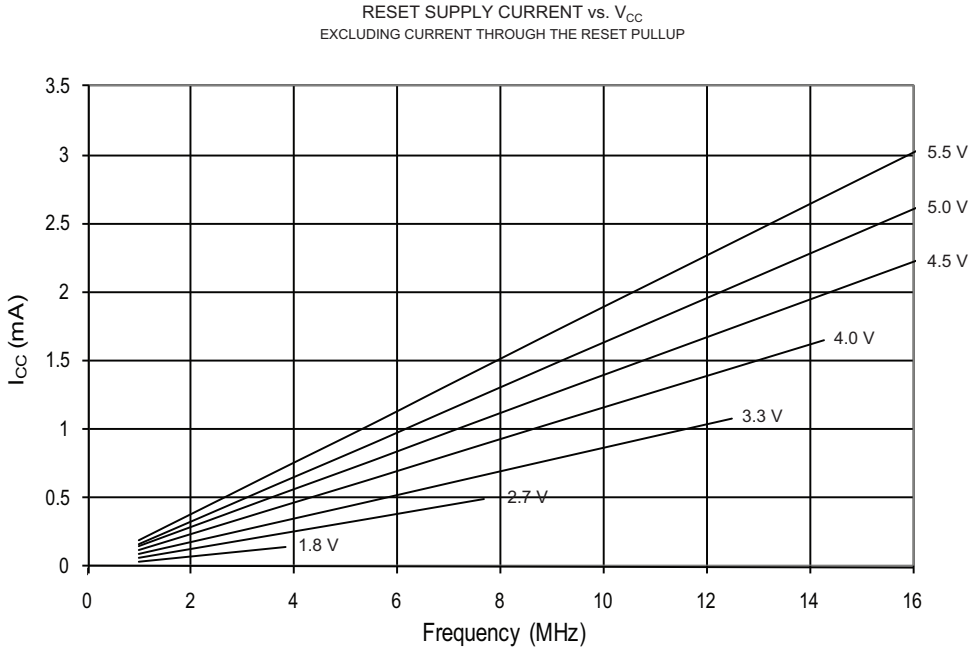


Figure 23-11. Reset Supply Current vs. V_{CC} (1 - 16 MHz, Excluding Current Through the Reset Pull-up)



23.6 Pull-up Resistors

Figure 23-12. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

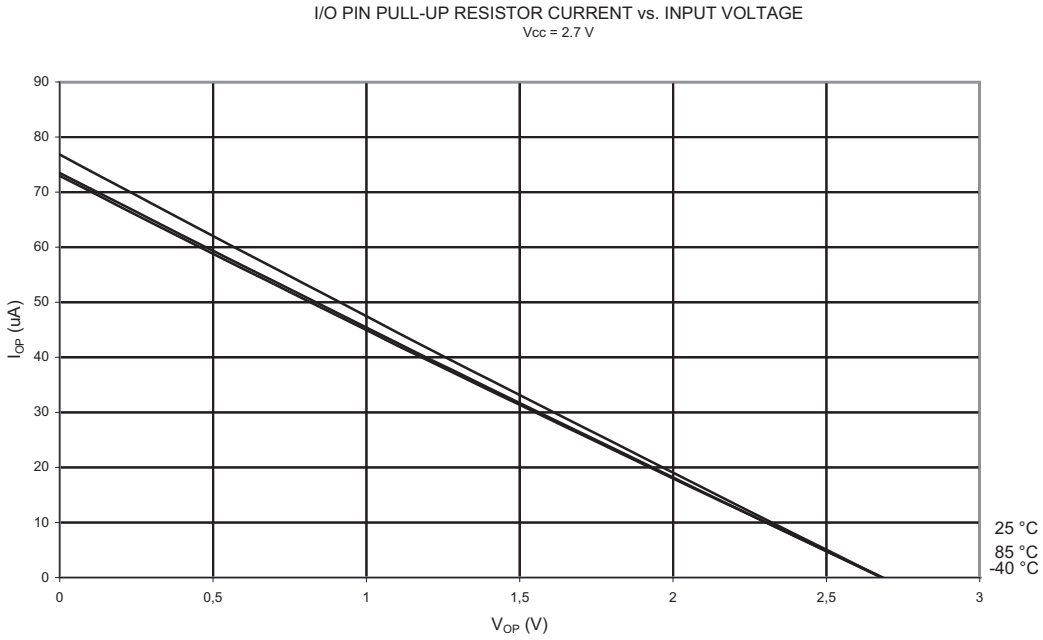


Figure 23-13. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

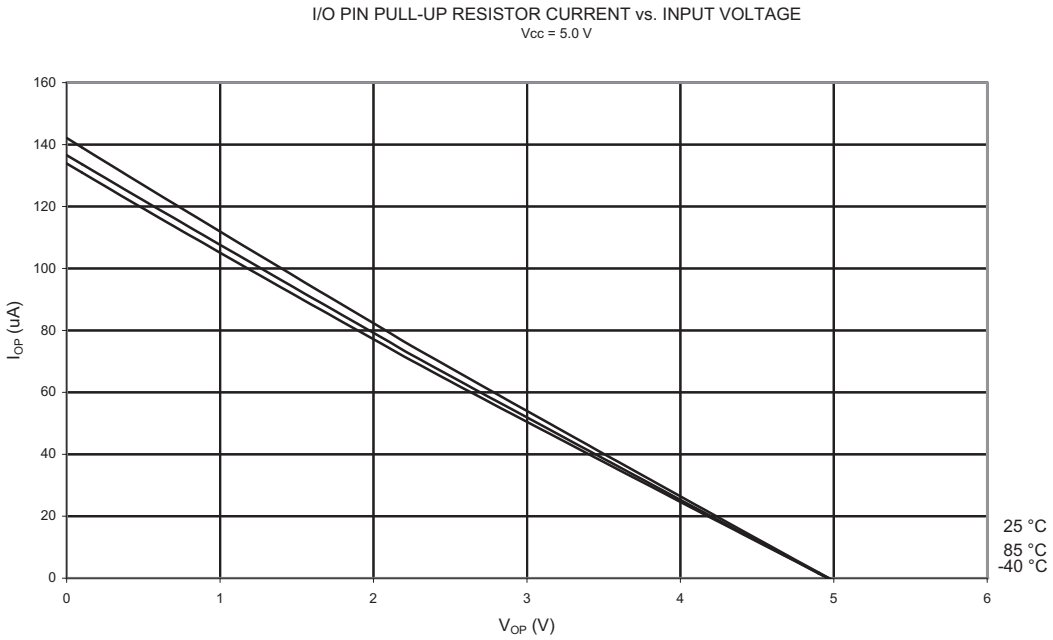


Figure 23-14. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)

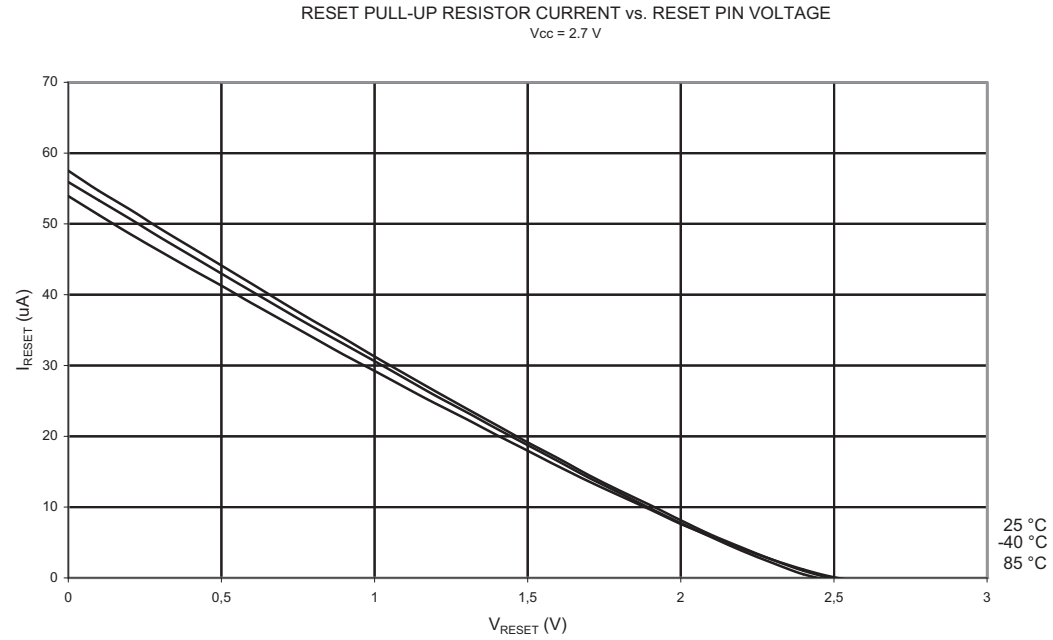
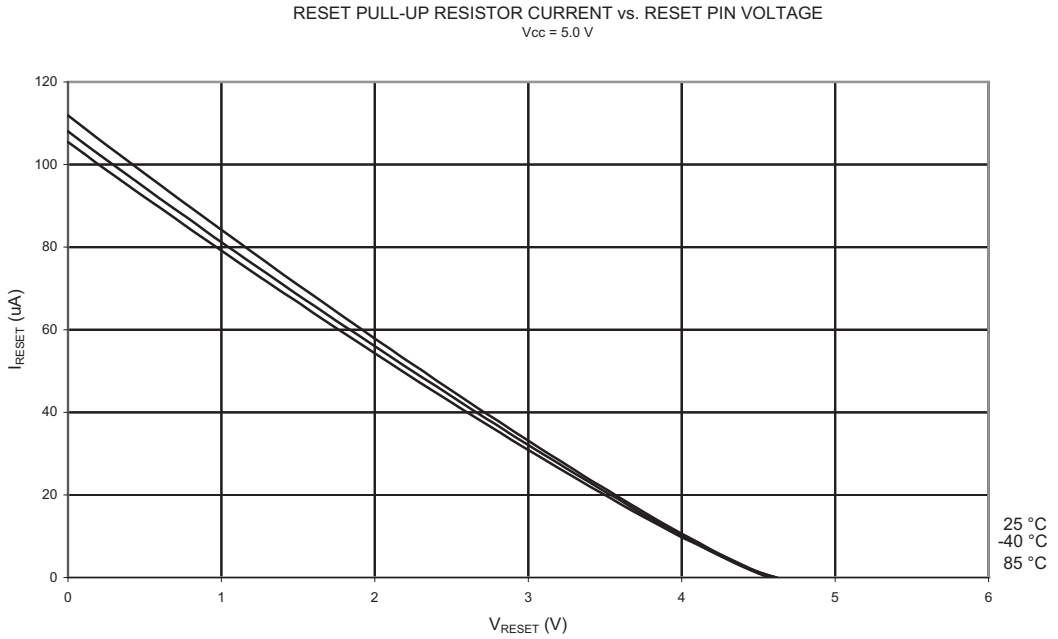


Figure 23-15. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)



23.7 Output Driver Strength

Figure 23-16. V_{OL}: I/O Pin Output Voltage vs. Sink Current (V_{CC} = 3V)

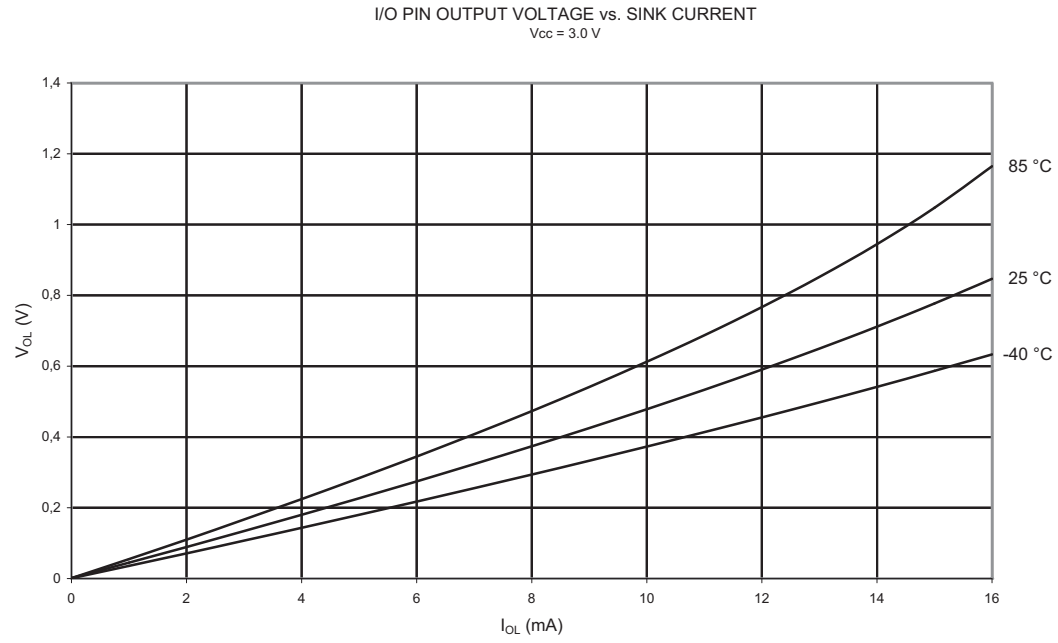


Figure 23-17. V_{OL}: I/O Pin Output Voltage vs. Sink Current (V_{CC} = 5V)

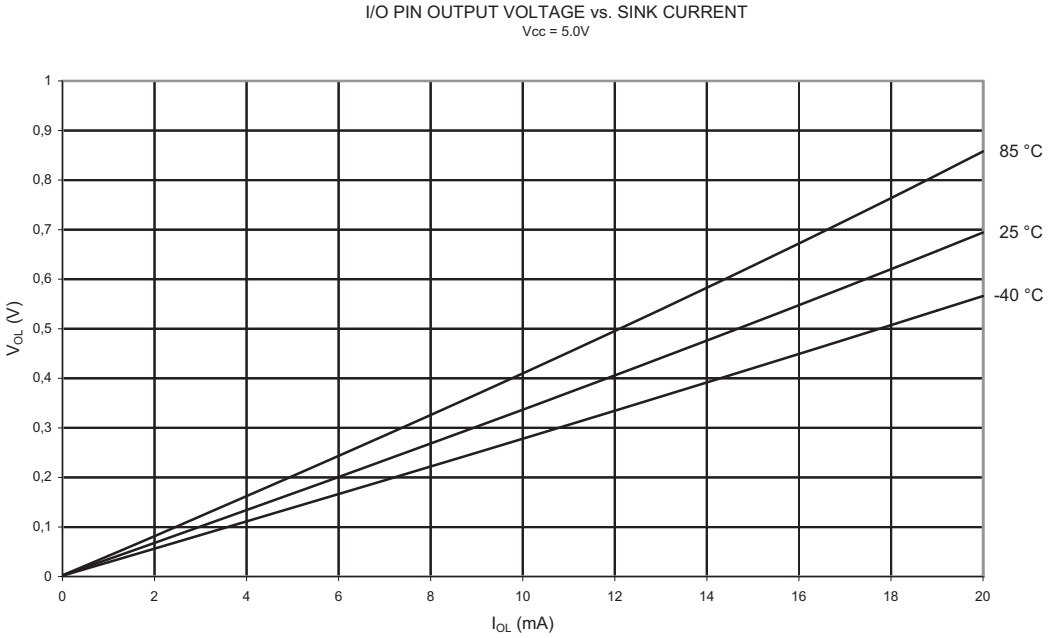


Figure 23-18. V_{OH}: I/O Pin Output Voltage vs. Source Current (V_{CC} = 3V)

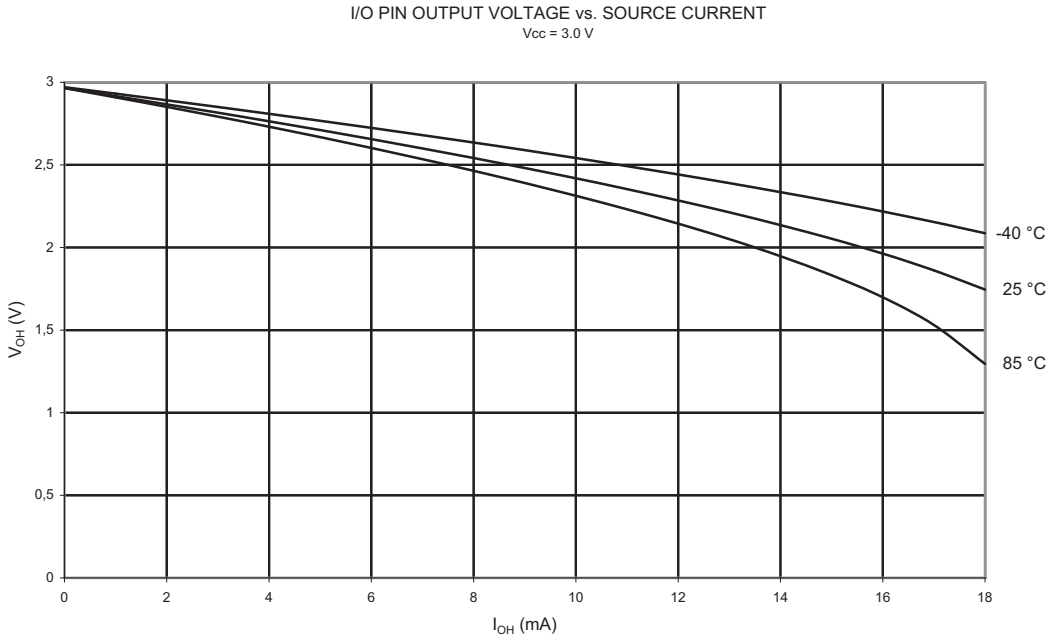
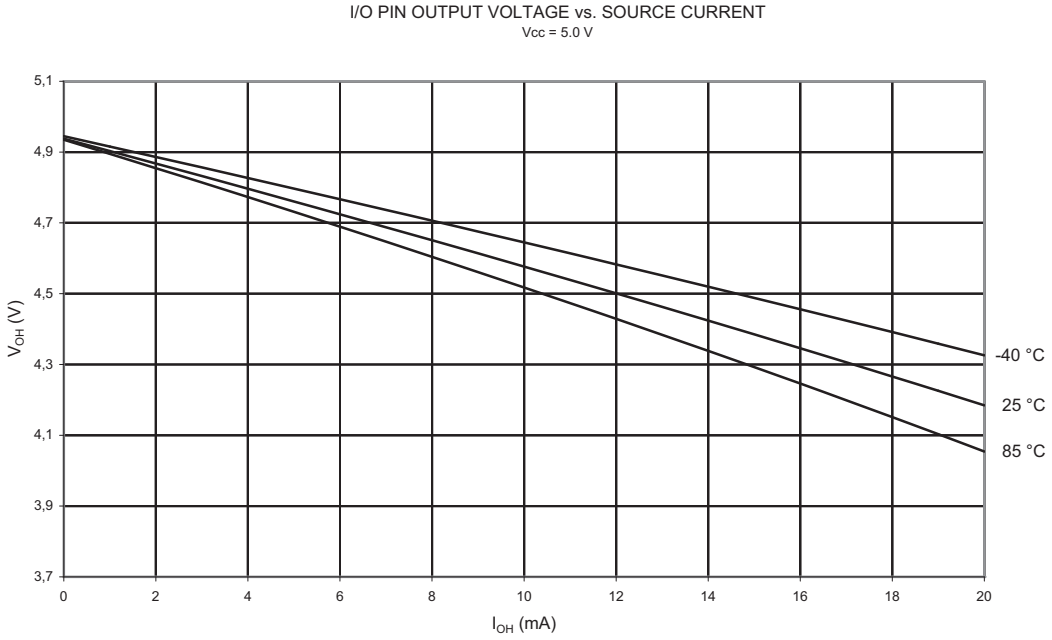


Figure 23-19. V_{OH}: I/O Pin Output Voltage vs. Source Current (V_{CC} = 5V)



23.8 Input Thresholds and Hysteresis (for I/O Ports)

Figure 23-20. V_{IH}: Input Threshold Voltage vs. V_{CC} (I/O Pin Read as ‘1’)

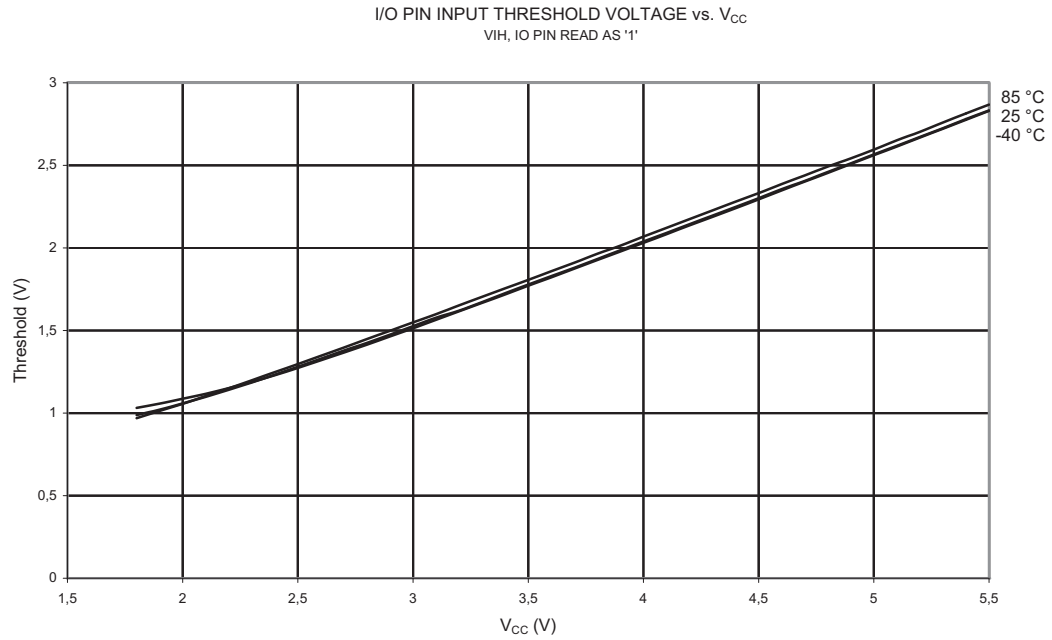
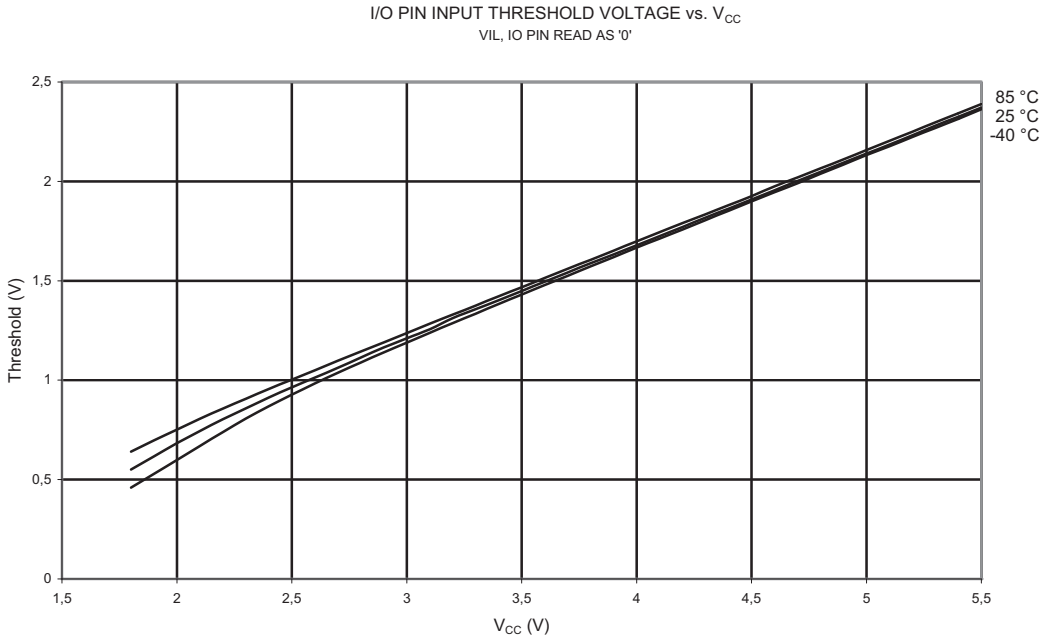


Figure 23-21. V_{IL}: Input Threshold Voltage vs. V_{CC} (I/O Pin, Read as '0')



23.9 BOD, Bandgap and Reset

Figure 23-22. BOD Thresholds vs. Temperature (BOD Level is 4.3V)

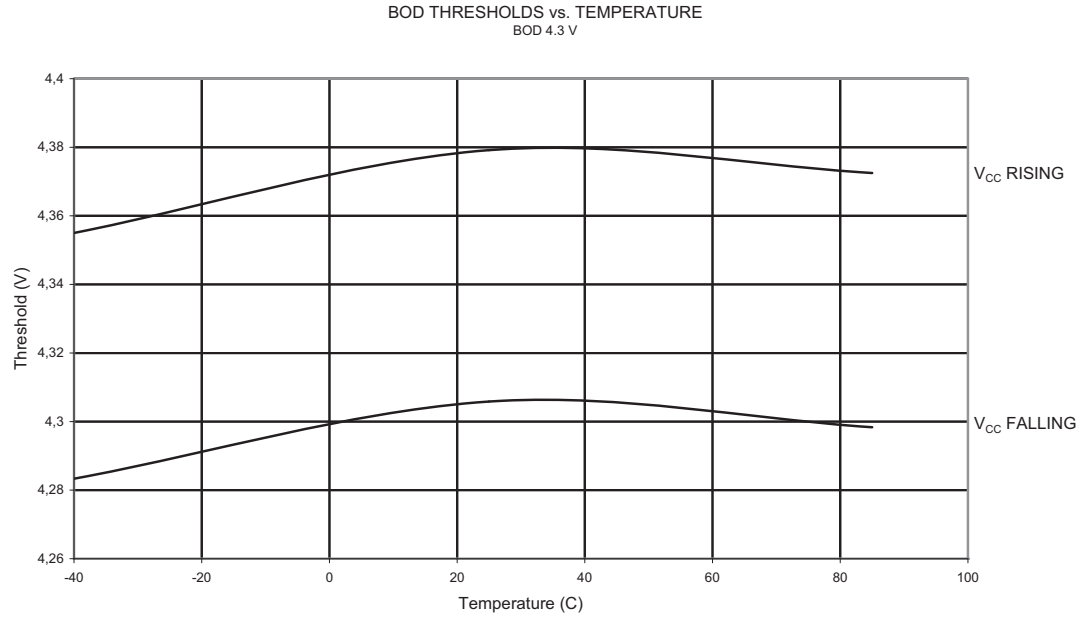


Figure 23-23. BOD Thresholds vs. Temperature (BOD Level is 2.7V)

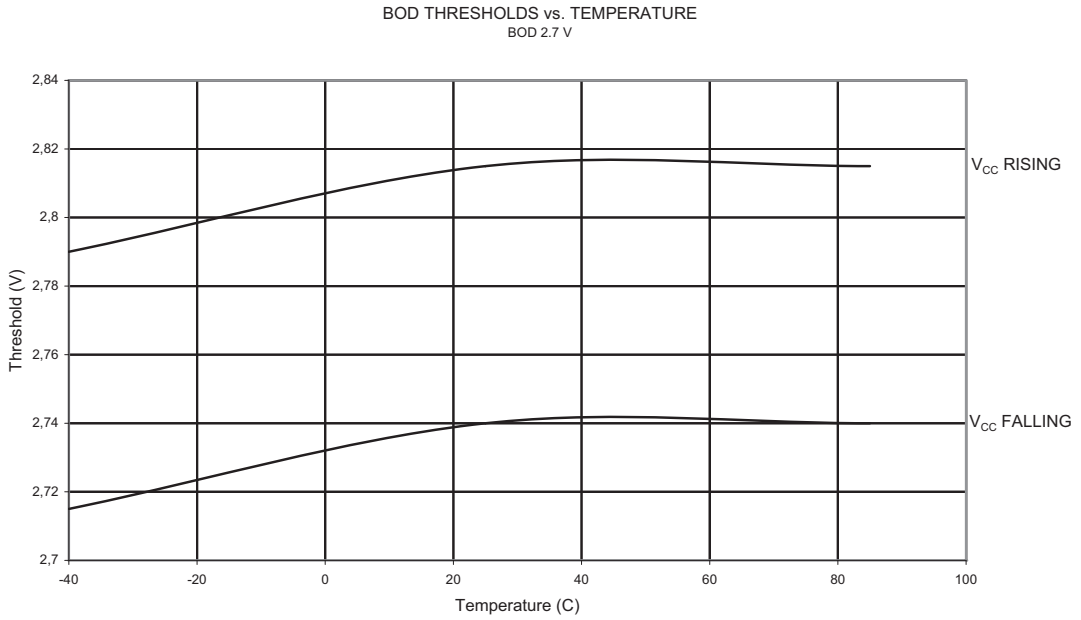


Figure 23-24. BOD Thresholds vs. Temperature (BOD Level is 1.8V)

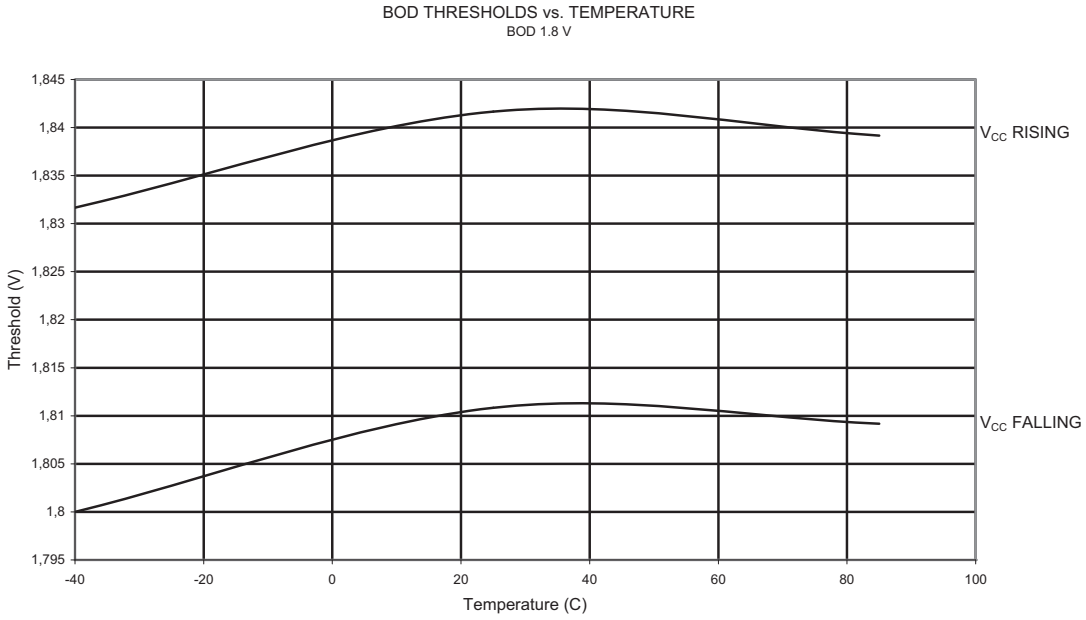


Figure 23-25. Bandgap Voltage vs. Supply Voltage

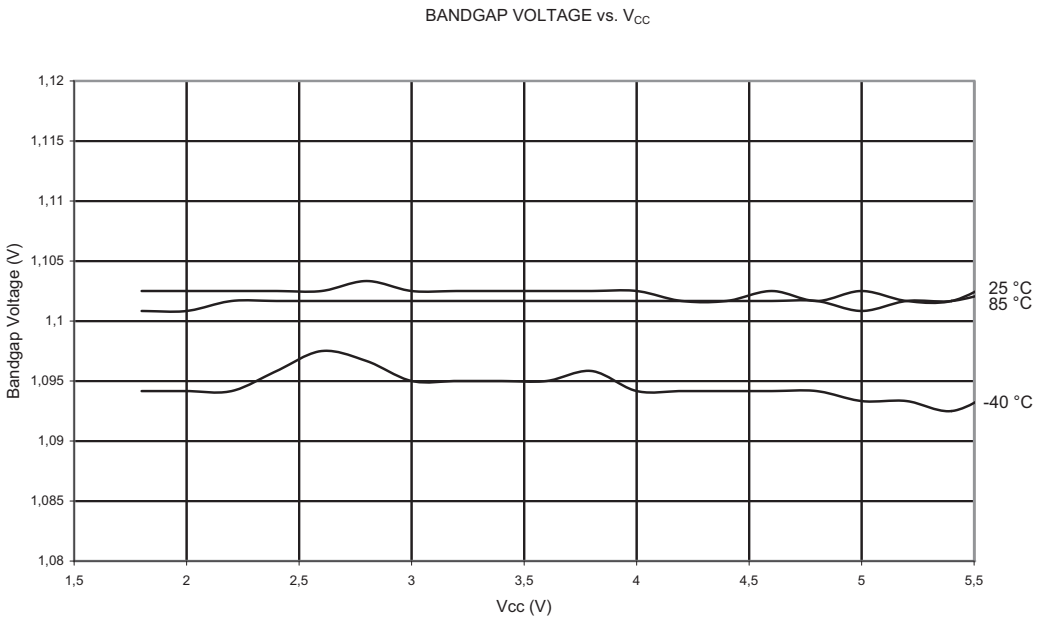


Figure 23-26. V_{IH}: Input Threshold Voltage vs. V_{CC} (Reset Pin, Read as ‘1’)

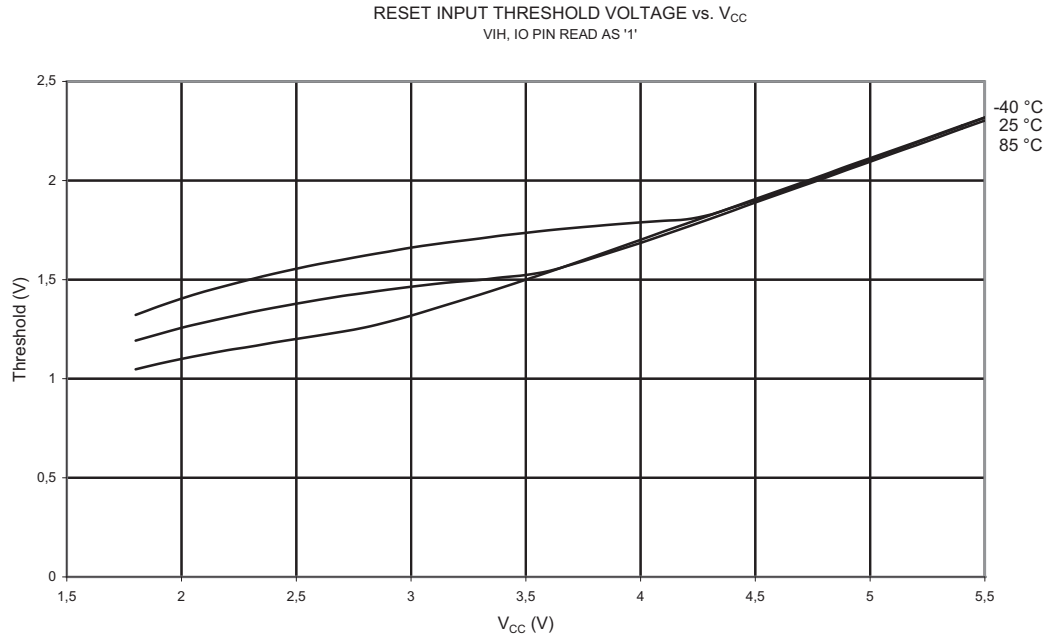
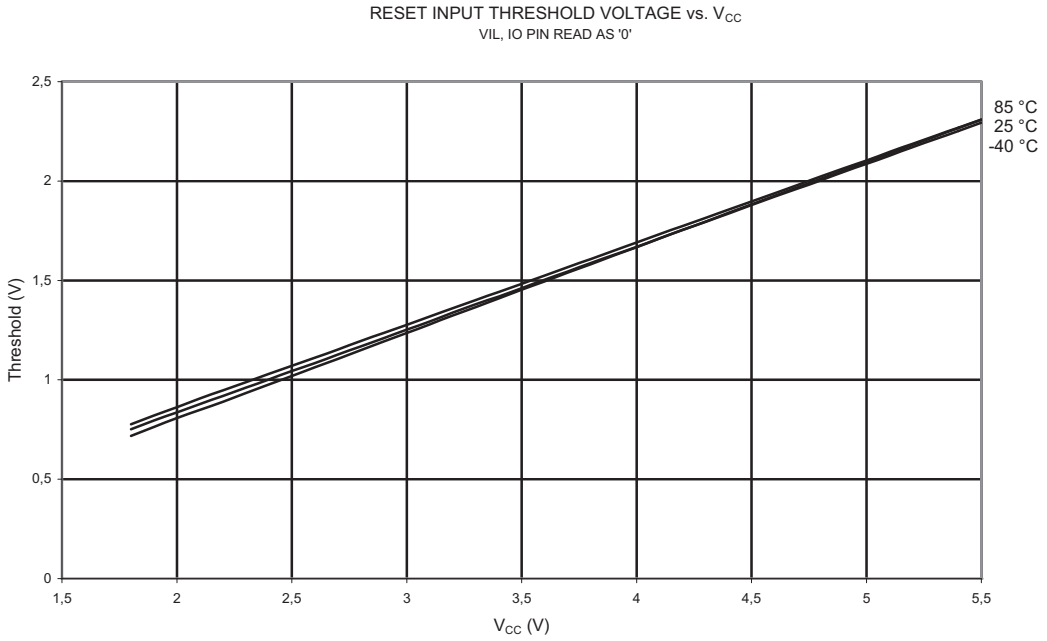


Figure 23-27. V_{IL}: Input Threshold Voltage vs. V_{CC} (Reset Pin, Read as '0')



23.10 Internal Oscillator Speed

Figure 23-28. Frequency of Watchdog Oscillator vs. Temperature

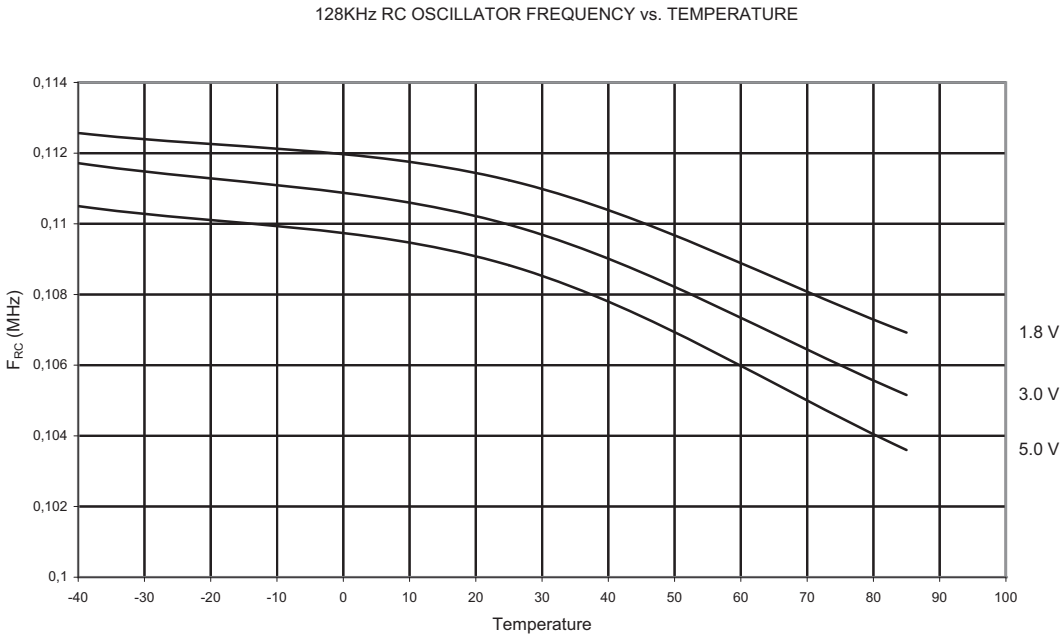
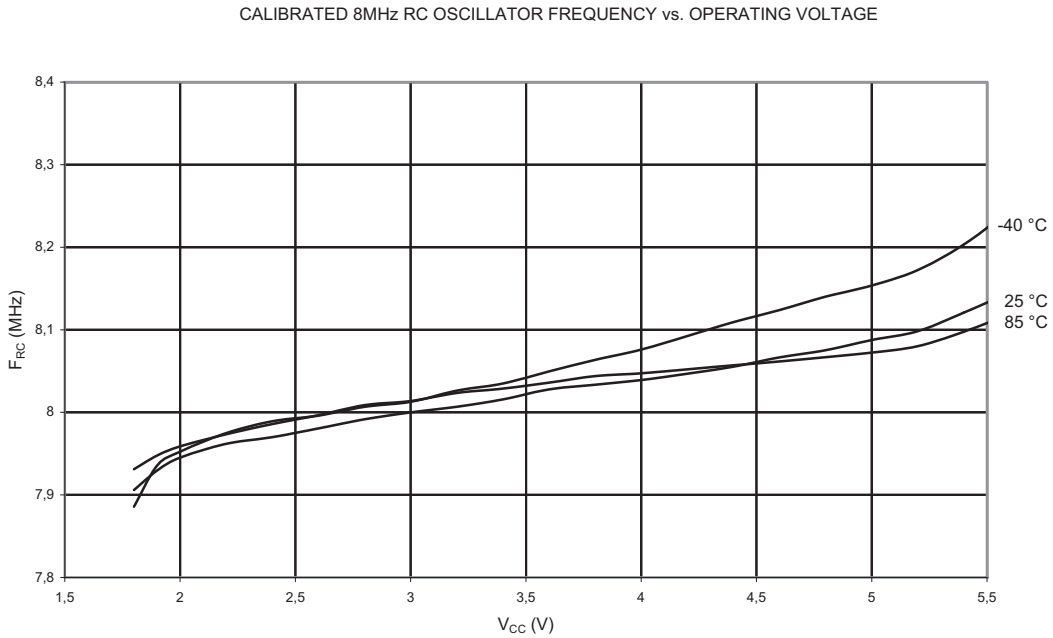


Figure 23-29. Calibrated 8.0 MHz RC Oscillator Frequency vs. V_{CC}



24. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	**Reserved**									
(0xFE)	**Reserved**									
(0xFD)	**Reserved**									
(0xFC)	**Reserved**									
(0xFB)	**Reserved**									
(0xFA)	**Reserved**									
(0xF9)	**Reserved**									
(0xF8)	**Reserved**									
(0xF7)	**Reserved**									
(0xF6)	**Reserved**									
(0xF5)	**Reserved**									
(0xF4)	**Reserved**									
(0xF3)	**Reserved**									
(0xF2)	**Reserved**									
(0xF1)	**Reserved**									
(0xF0)	**Reserved**									
(0xEF)	**Reserved**									
(0xEE)	**Reserved**									
(0xED)	**Reserved**									
(0xEC)	**Reserved**									
(0xEB)	**Reserved**									
(0xEA)	**Reserved**									
(0xE9)	**Reserved**									
(0xE8)	**Reserved**									
(0xE7)	**Reserved**									
(0xE6)	**Reserved**									
(0xE5)	**Reserved**									
(0xE4)	**Reserved**									
(0xE3)	**Reserved**									
(0xE2)	**Reserved**									
(0xE1)	**Reserved**									
(0xE0)	**Reserved**									
(0xDF)	**Reserved**									
(0xDE)	**Reserved**									
(0xDD)	**Reserved**									
(0xDC)	**Reserved**									
(0xDB)	**Reserved**									
(0xDA)	**Reserved**									
(0xD9)	**Reserved**									
(0xD8)	**Reserved**									
(0xD7)	**Reserved**									
(0xD6)	**Reserved**									
(0xD5)	**Reserved**									
(0xD4)	**Reserved**									
(0xD3)	**Reserved**									
(0xD2)	**LINDAT**	LDATA7	LDATA6	LDATA5	LDATA4	LDATA3	LDATA2	LDATA1	LDATA0	page 194
(0xD1)	**LINSEL**	–	–	–	–	/LAINC	LINDX2	LINDX1	LINDX0	page 193
(0xD0)	**LINIDR**	LP1	LP0	LID5 / LDL1	LID4 / LDL0	LID3	LID2	LID1	LID0	page 193
(0xCF)	**LINDLR**	LTXDL3	LTXDL2	LTXDL1	LTXDL0	LRXDL3	LRXDL2	LRXDL1	LRXDL0	page 192
(0xCE)	**LINBRRH**	–	–	–	–	LDIV11	LDIV10	LDIV9	LDIV8	page 192
(0xCD)	**LINBRRL**	LDIV7	LDIV6	LDIV5	LDIV4	LDIV3	LDIV2	LDIV1	LDIV0	page 192
(0xCC)	**LINBTR**	LDISR	–	LBT5	LBT4	LBT3	LBT2	LBT1	LBT0	page 191
(0xCB)	**LINERR**	LABORT	LTOERR	LOVERR	LFERR	LSERR	LPERR	LCERR	LBERR	page 190
(0xCA)	**LINENIR**	–	–	–	–	LENERR	LENIDOK	LENTXOK	LENRXOK	page 190
(0xC9)	**LINSIR**	LIDST2	LIDST1	LIDST0	LBUSY	LERR	LIDOK	LTXOK	LRXOK	page 189
(0xC8)	**LINCR**	LSWRES	LIN13	LCONF1	LCONF0	LENA	LCMD2	LCMD1	LCMD0	page 188
(0xC7)	**Reserved**									
(0xC6)	**Reserved**									
(0xC5)	**Reserved**									
(0xC4)	**Reserved**									
(0xC3)	**Reserved**									
(0xC2)	**Reserved**									
(0xC1)	**Reserved**									
(0xC0)	**Reserved**									
(0xBF)	**Reserved**									

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBE)	**Reserved**									
(0xBD)	**Reserved**									
(0xBC)	**USIPP**								USIPOS	page 168
(0xBB)	**USIBR**	USIB7	USIB6	USIB5	USIB4	USIB3	USIB2	USIB1	USIB0	page 164
(0xBA)	**USIDR**	USID7	USID6	USID5	USID4	USID3	USID2	USID1	USID0	page 163
(0xB9)	**USISR**	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	page 164
(0xB8)	**USICR**	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	page 165
(0xB7)	**Reserved**									
(0xB6)	**ASSR**	–	EXCLK	AS0	TCN0UB	OCR0AUB	–	TCR0AUB	TCR0BUB	page 110
(0xB5)	**Reserved**									
(0xB4)	**Reserved**									
(0xB3)	**Reserved**									
(0xB2)	**Reserved**									
(0xB1)	**Reserved**									
(0xB0)	**Reserved**									
(0xAF)	**Reserved**									
(0xAE)	**Reserved**									
(0xAD)	**Reserved**									
(0xAC)	**Reserved**									
(0xAB)	**Reserved**									
(0xAA)	**Reserved**									
(0xA9)	**Reserved**									
(0xA8)	**Reserved**									
(0xA7)	**Reserved**									
(0xA6)	**Reserved**									
(0xA5)	**Reserved**									
(0xA4)	**Reserved**									
(0xA3)	**Reserved**									
(0xA2)	**Reserved**									
(0xA1)	**Reserved**									
(0xA0)	**Reserved**									
(0x9F)	**Reserved**									
(0x9E)	**Reserved**									
(0x9D)	**Reserved**									
(0x9C)	**Reserved**									
(0x9B)	**Reserved**									
(0x9A)	**Reserved**									
(0x99)	**Reserved**									
(0x98)	**Reserved**									
(0x97)	**Reserved**									
(0x96)	**Reserved**									
(0x95)	**Reserved**									
(0x94)	**Reserved**									
(0x93)	**Reserved**									
(0x92)	**Reserved**									
(0x91)	**Reserved**									
(0x90)	**Reserved**									
(0x8F)	**Reserved**									
(0x8E)	**Reserved**									
(0x8D)	**Reserved**									
(0x8C)	**Reserved**									
(0x8B)	**OCR1BH**	OCR1B15	OCR1B14	OCR1B13	OCR1B12	OCR1B11	OCR1B10	OCR1B9	OCR1B8	page 144
(0x8A)	**OCR1BL**	OCR1B7	OCR1B6	OCR1B5	OCR1B4	OCR1B3	OCR1B2	OCR1B1	OCR1B0	page 144
(0x89)	**OCR1AH**	OCR1A15	OCR1A14	OCR1A13	OCR1A12	OCR1A11	OCR1A10	OCR1A9	OCR1A8	page 144
(0x88)	**OCR1AL**	OCR1A7	OCR1A6	OCR1A5	OCR1A4	OCR1A3	OCR1A2	OCR1A1	OCR1A0	page 144
(0x87)	**ICR1H**	ICR115	ICR114	ICR113	ICR112	ICR111	ICR110	ICR19	ICR18	page 145
(0x86)	**ICR1L**	ICR17	ICR16	ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	page 145
(0x85)	**TCNT1H**	TCNT115	TCNT114	TCNT113	TCNT112	TCNT111	TCNT110	TCNT19	TCNT18	page 144
(0x84)	**TCNT1L**	TCNT17	TCNT16	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	page 144
(0x83)	**TCCR1D**	OC1BX	OC1BW	OC1BV	OC1BU	OC1AX	OC1AW	OC1AV	OC1AU	page 143
(0x82)	**TCCR1C**	FOC1A	FOC1B	–	–	–	–	–	–	page 143
(0x81)	**TCCR1B**	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	page 142
(0x80)	**TCCR1A**	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	page 139
(0x7F)	**DIDR1**	–	ADC10D	ADC9D	ADC8D	–	–	–	–	page 217
(0x7E)	**DIDR0**	ADC7D/AIN1D	ADC6D/AIN0D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	page 216, page 221
(0x7D)	**Reserved**									

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7C)	**ADMUX**	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	page 212
(0x7B)	**ADCSRB**	BIN	ACME	ACIR1	ACIR0	–	ADTS2	ADTS1	ADTS0	page 216, page 220
(0x7A)	**ADCSRA**	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	page 214
(0x79)	**ADCH**	- / ADC9	- / ADC8	- / ADC7	- / ADC6	- / ADC5	- / ADC4	ADC9 / ADC3	ADC8 / ADC2	page 215
(0x78)	**ADCL**	ADC7 / ADC1	ADC6 / ADC0	ADC5 / -	ADC4 / -	ADC3 / -	ADC2 / -	ADC1 / -	ADC0 /	page 215
(0x77)	**AMISCR**	–	–	–	–	–	AREFEN	XREFEN	ISRCEN	page 197, page 217
(0x76)	**Reserved**									
(0x75)	**Reserved**									
(0x74)	**Reserved**									
(0x73)	**Reserved**									
(0x72)	**Reserved**									
(0x71)	**Reserved**									
(0x70)	**Reserved**									
(0x6F)	**TIMSK1**	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	page 145
(0x6E)	**TIMSK0**	–	–	–	–	–	–	OCIE0A	TOIE0	page 112
(0x6D)	**Reserved**									
(0x6C)	**PCMSK1**	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	page 73
(0x6B)	**PCMSK0**	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	page 73
(0x6A)	**Reserved**									
(0x69)	**EICRA**	–	–	–	–	ISC11	ISC10	ISC01	ISC00	page 71
(0x68)	**PCICR**	–	–	–	–	–	–	PCIE1	PCIE0	page 72
(0x67)	**Reserved**									
(0x66)	**OSCCAL**	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	page 45
(0x65)	**Reserved**									
(0x64)	**PRR**	–	–	PRLIN	PRSPI	PRTIM1	PRTIM0	PRUSI	PRADC	page 55
(0x63)	**CLKSELR**	–	COUT	CSUT1	CSUT0	CSEL3	CSEL2	CSEL1	CSEL0	page 48
(0x62)	**CLKCSR**	CLKCCE	–	–	CLKRDY	CLKC3	CLKC2	CLKC1	CLKC0	page 47
(0x61)	**CLKPR**	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	page 46
(0x60)	**WDTCR**	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	page 65
0x3F (0x5F)	**SREG**	I	T	H	S	V	N	Z	C	page 17
0x3E (0x5E)	**SPH**	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	page 19
0x3D (0x5D)	**SPL**	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 19
0x3C (0x5C)	**Reserved**									
0x3B (0x5B)	**Reserved**									
0x3A (0x5A)	**Reserved**									
0x39 (0x59)	**Reserved**									
0x38 (0x58)	**Reserved**									
0x37 (0x57)	**SPMCSR**	–	RWWSB	SIGRD	CTPB	RFLB	PGWRT	PGERS	SPMEN	page 226
0x36 (0x56)	**Reserved**	–	–	–	–	–	–	–	–	
0x35 (0x55)	**MCUCR**	–	BODS	BODSE	PUD	–	–	–	–	page 55, page 83
0x34 (0x54)	**MCUSR**	–	–	–	–	WDRF	BORF	EXTRF	PORF	page 60
0x33 (0x53)	**SMCR**	–	–	–	–	–	SM1	SM0	SE	page 54
0x32 (0x52)	**Reserved**									
0x31 (0x51)	**DWDR**	DWDR7	DWDR6	DWDR5	DWDR4	DWDR3	DWDR2	DWDR1	DWDR0	page 223
0x30 (0x50)	**ACSR**	ACD	ACIRS	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	page 220
0x2F (0x4F)	**Reserved**									
0x2E (0x4E)	**SPDR**	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	page 154
0x2D (0x4D)	**SPSR**	SPIF	WCOL	–	–	–	–	–	SPI2X	page 154
0x2C (0x4C)	**SPCR**	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	page 152
0x2B (0x4B)	**GPIOR2**	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	page 31
0x2A (0x4A)	**GPIOR1**	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	page 31
0x29 (0x49)	**Reserved**									
0x28 (0x48)	**OCR0A**	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	page 110
0x27 (0x47)	**TCNT0**	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	page 110
0x26 (0x46)	**TCCR0B**	FOC0A	–	–	–	–	CS02	CS01	CS00	page 109
0x25 (0x45)	**TCCR0A**	COM0A1	COM0A0	–	–	–	–	WGM01	WGM00	page 107
0x24 (0x44)	**Reserved**									
0x23 (0x43)	**GTCCR**	TSM	–	–	–	–	–	PSR0	PSR1	page 113, page 116
0x22 (0x42)	**EEARH**[1]	–	–	–	–	–	–	–	EEAR8	page 29
0x21 (0x41)	**EEARL**	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	page 29
0x20 (0x40)	**EEDR**	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	page 30
0x1F (0x3F)	**EECR**	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	page 30
0x1E (0x3E)	**GPIOR0**	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	page 31
0x1D (0x3D)	**EIMSK**	–	–	–	–	–	–	INT1	INT0	page 71
0x1C (0x3C)	**EIFR**	–	–	–	–	–	–	INTF1	INTF0	page 72
0x1B (0x3B)	**PCIFR**	–	–	–	–	–	–	PCIF1	PCIF0	page 73

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1A (0x3A)	**Reserved**									
0x19 (0x39)	**Reserved**									
0x18 (0x38)	**Reserved**									
0x17 (0x37)	**Reserved**									
0x16 (0x36)	**TIFR1**	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	page 146
0x15 (0x35)	**TIFR0**	–	–	–	–	–	–	OCF0A	TOV0	page 112
0x14 (0x34)	**Reserved**									
0x13 (0x33)	**Reserved**									
0x12 (0x32)	**PORTCR**	–	–	BBMB	BBMA	–	–	PUDB	PUDA	page 83
0x11 (0x31)	**Reserved**									
0x10 (0x30)	**Reserved**									
0x0F (0x2F)	**Reserved**									
0x0E (0x2E)	**Reserved**									
0x0D (0x2D)	**Reserved**									
0x0C (0x2C)	**Reserved**									
0x0B (0x2B)	**Reserved**									
0x0A (0x2A)	**Reserved**									
0x09 (0x29)	**Reserved**									
0x08 (0x28)	**Reserved**									
0x07 (0x27)	**Reserved**									
0x06 (0x26)	**Reserved**									
0x05 (0x25)	**PORTB**	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 93
0x04 (0x24)	**DDRB**	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 93
0x03 (0x23)	**PINB**	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 93
0x02 (0x22)	**PORTA**	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	page 93
0x01 (0x21)	**DDRA**	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	page 93
0x00 (0x20)	**PINA**	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	page 93

Notes:
1. Address bits exceeding EEAMSB (Table 21-8 on page 235) are don't care.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
4. Some of the status flags are cleared by writing a logical one to them. 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 to 0x1F only.
5. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

25. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock
		ARITHMETIC AND LOGIC INSTRUCTIONS			
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
		BRANCH INSTRUCTIONS			
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
		BIT AND BIT-TEST INSTRUCTIONS			
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1

Mnemonics	Operands	Description	Operation	Flags	#Clock
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

26. Ordering Information

26.1 ATtiny87

Speed (MHz)	Power Supply (V)	Ordering Code	Package[1]	Operational Range
16	1.8 – 5.5	ATtiny87-MU ATtiny87-MUR[2] ATtiny87-SU ATtiny87-SUR[2] ATtiny87-XU ATtiny87-XUR[2]	32PN 32PN 20S2 20S2 20X 20X	Industrial (-40°C to +85°C)[3]

Notes:
1. All packages are Pb-free, halide-free and fully green and they comply with the European directive for Restriction of Hazardous Substances (RoHS).
2. Tape and reel.
3. These devices can also be supplied in wafer form. Please contact your local Microchip sales office for detailed ordering information and minimum quantities.

Package Type	
32PN	32-lead, 0.5mm pitch, 5 x 5 mm Very Thin Quad Flat No Lead Package (VQFN) Sawn
20S2	20-lead, 0.300" Wide, Plastic Gull Wing Small Outline Package (SOIC)
20X	20-lead, 4.4 mm Wide, Plastic Thin Shrink Small Outline Package (TSSOP)

26.2 ATtiny167

Speed (MHz)	Power Supply (V)	Ordering Code	Package[1]	Operational Range
16	1.8 – 5.5	ATtiny167-MU ATtiny167-MUR[2] ATtiny167-MMU ATtiny167-MMUR[2] ATtiny167-SU ATtiny167-SUR[2] ATtiny167-XU ATtiny167-XUR[2]	32PN 32PN TSB TSB 20S2 20S2 20X 20X	Industrial (-40°C to +85°C)[3]

Notes:
1. All packages are Pb-free, halide-free and fully green and they comply with the European directive for Restriction of Hazardous Substances (RoHS).
2. Tape and reel.
3. These devices can also be supplied in wafer form. Please contact your local Microchip sales office for detailed ordering information and minimum quantities.

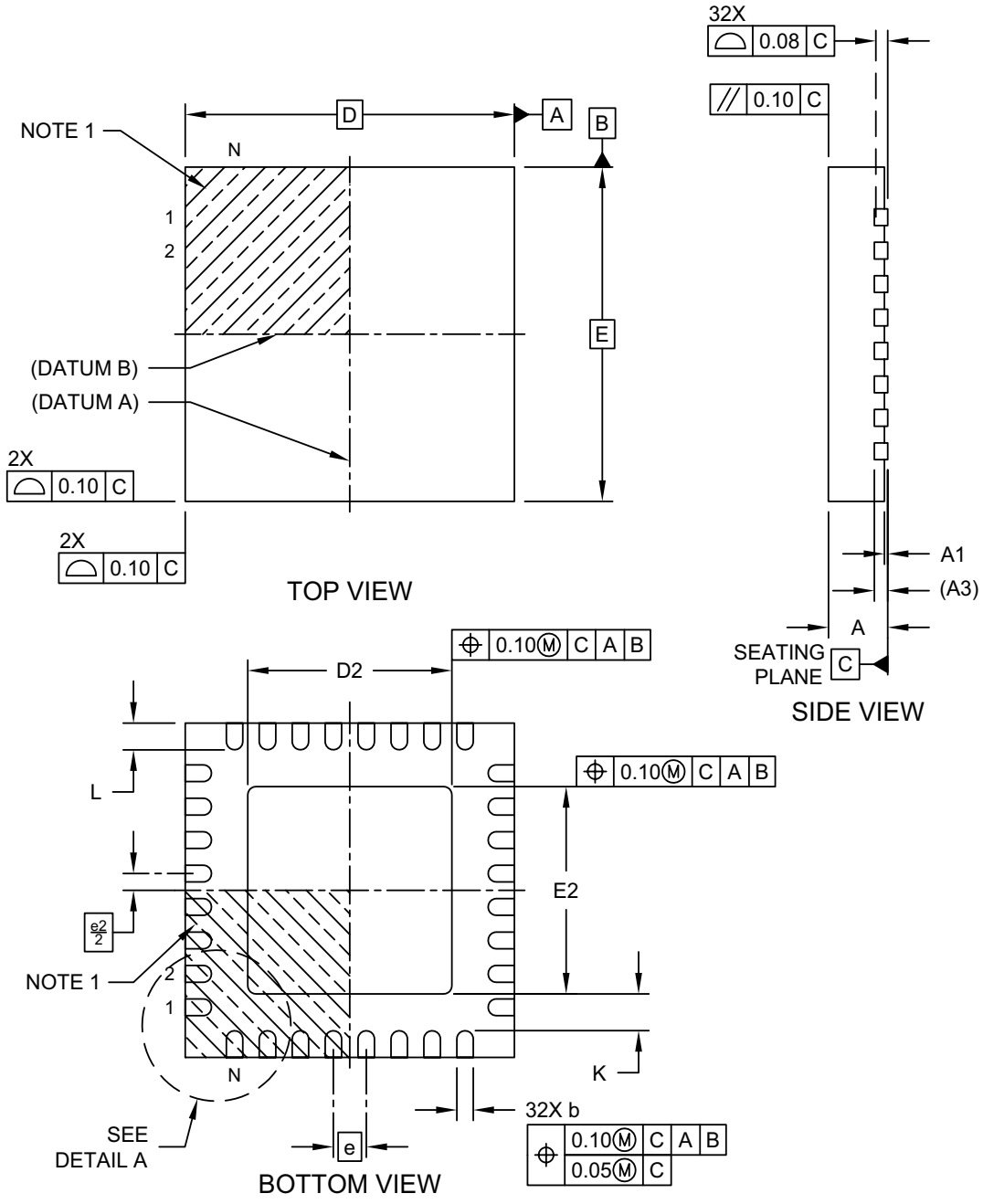
Package Type	
32PN	32-lead, 0.5mm pitch, 5 x 5 mm Very Thin Quad Flat No Lead Package (VQFN) Sawn
20S2	20-lead, 0.300" Wide, Plastic Gull Wing Small Outline Package (SOIC)
20X	20-lead, 4.4 mm Wide, Plastic Thin Shrink Small Outline Package (TSSOP)
TSB	20-lead, Very Thin Quad Flat No Lead Package (WQFN)

27. Packaging Information

27.1 32PN

32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

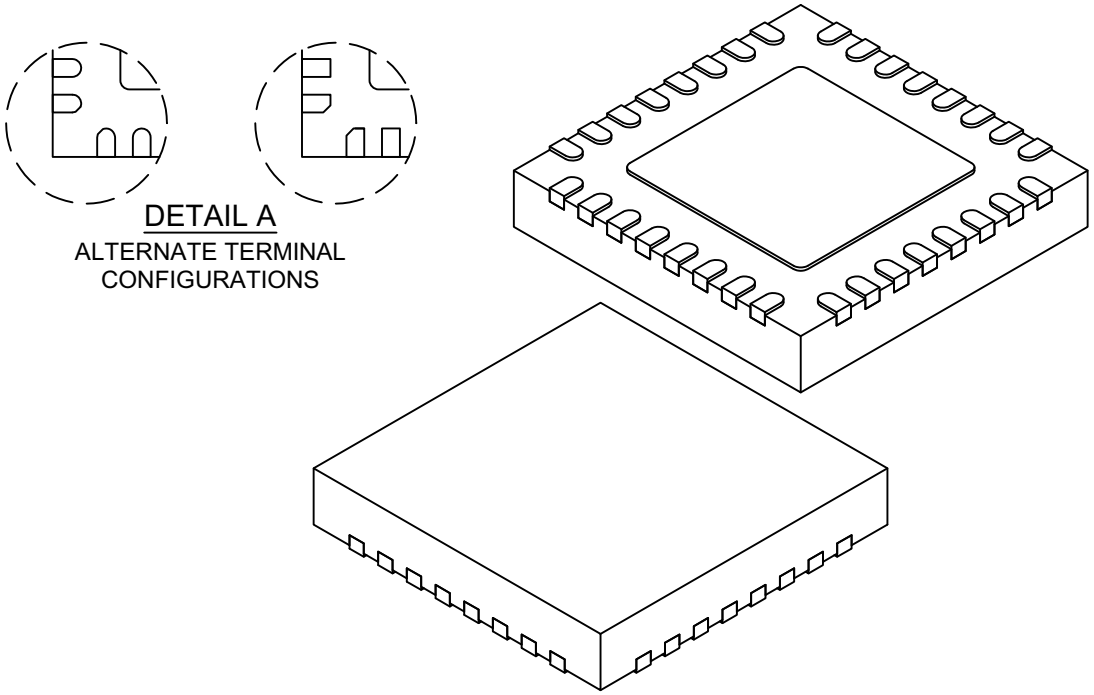
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-21395-RXB Rev B Sheet 1 of 2

32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		32	
Pitch	e		0.50 BSC	
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		5.00 BSC	
Exposed Pad Length	D2	3.00	3.10	3.20
Overall Width	E		5.00 BSC	
Exposed Pad Width	E2	3.00	3.10	3.20
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K	0.20	-	-

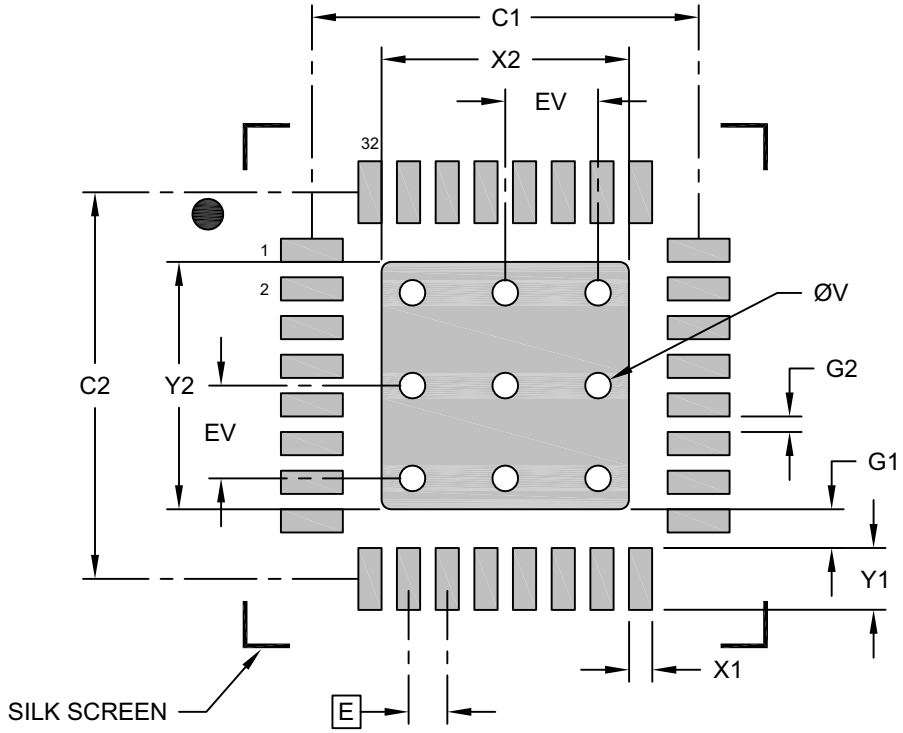
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21395-RXB Rev B Sheet 2 of 2

32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Center Pad Width	X2			3.20
Center Pad Length	Y2			3.20
Contact Pad Spacing	C1		5.00	
Contact Pad Spacing	C2		5.00	
Contact Pad Width (X32)	X1			0.30
Contact Pad Length (X32)	Y1			0.80
Contact Pad to Center Pad (X32)	G1	0.20		
Contact Pad to Contactr Pad (X28)	G2	0.20		
Thermal Via Diameter	V		0.33	
Thermal Via Pitch	EV		1.20	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

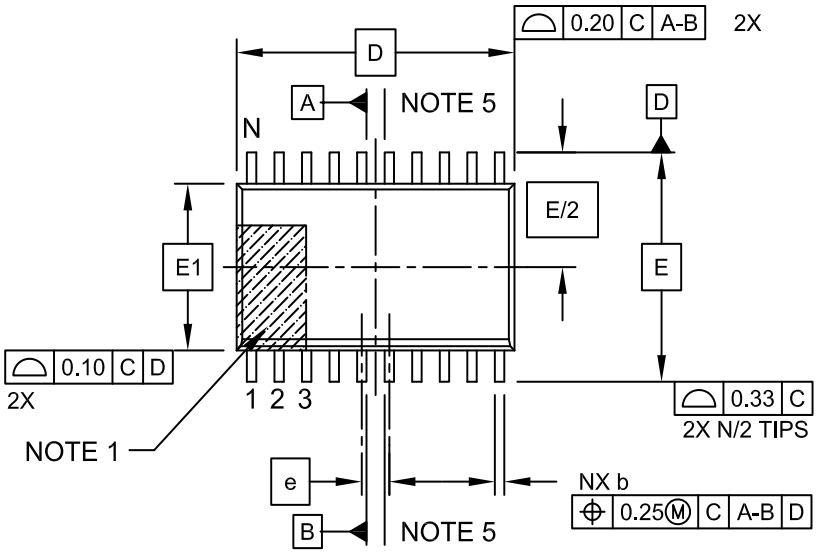
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23395-RXB Rev B

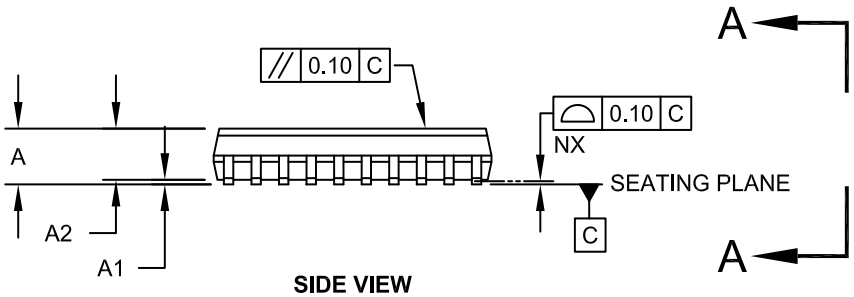
27.2 20S2

20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

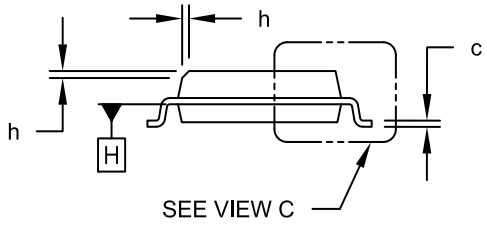
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

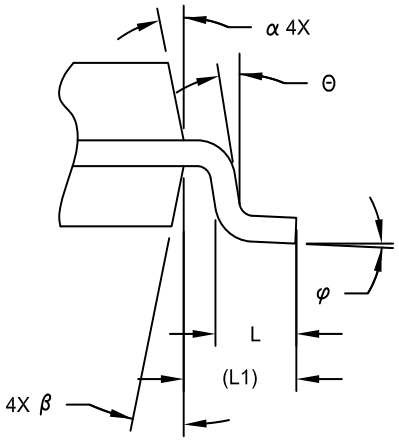


VIEW A-A

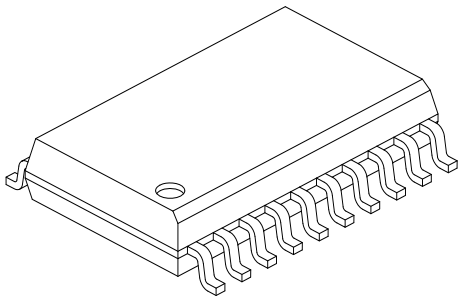
Microchip Technology Drawing C04-094C Sheet 1 of 2

20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



VIEW C



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		20	
Pitch	e		1.27 BSC	
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff §	A1	0.10	-	0.30
Overall Width	E		10.30 BSC	
Molded Package Width	E1		7.50 BSC	
Overall Length	D		12.80 BSC	
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1		1.40 REF	
Lead Angle	Θ	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.20	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

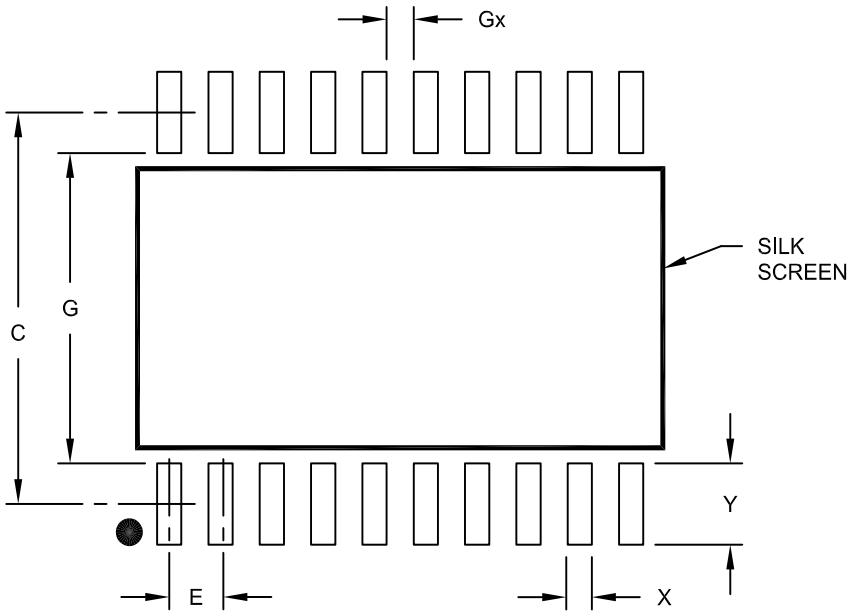
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-094C Sheet 2 of 2

20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		9.40	
Contact Pad Width (X20)	X			0.60
Contact Pad Length (X20)	Y			1.95
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.45		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

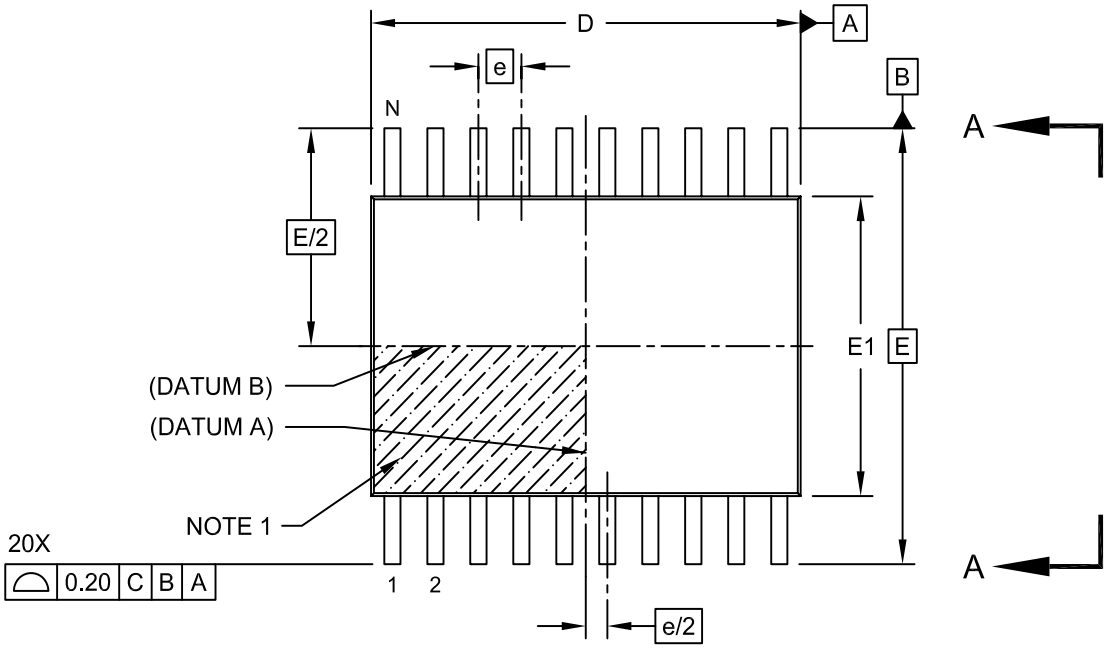
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2094A

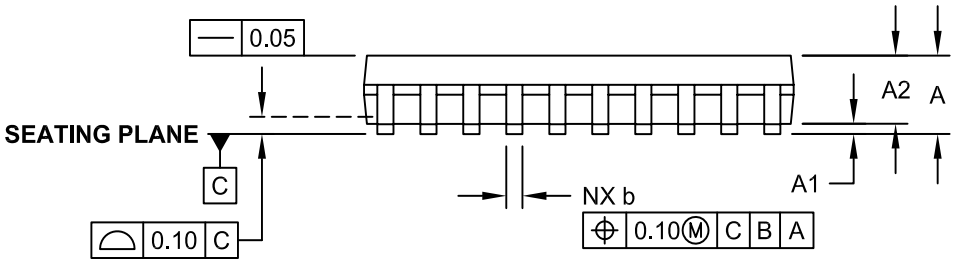
27.3 20X

20-Lead Plastic Thin Shrink Small Outline (ST) - 4.4 mm Body [TSSOP]

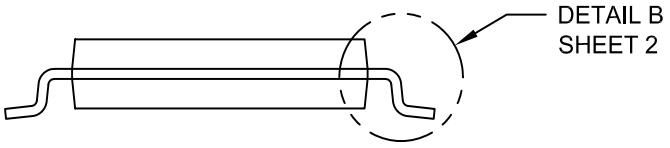
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

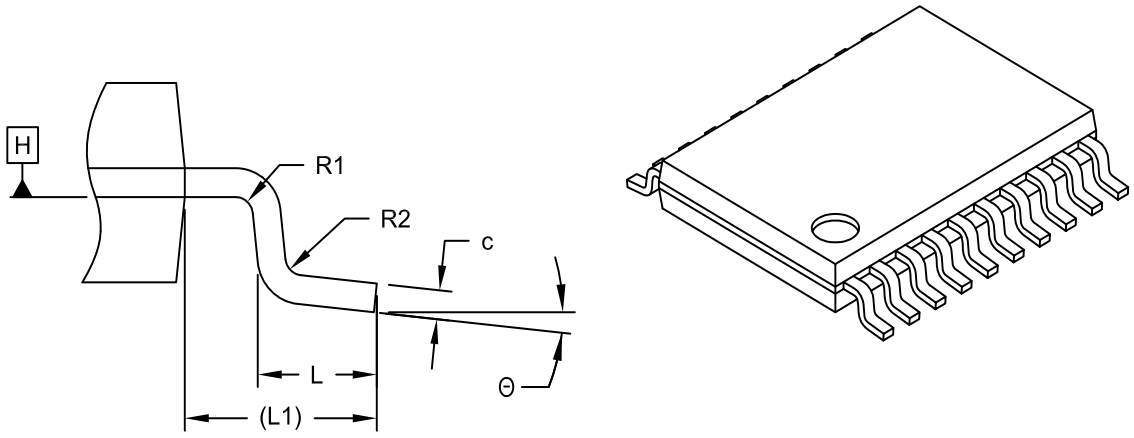


VIEW A—A

Microchip Technology Drawing C04-088C Sheet 1 of 2

20-Lead Plastic Thin Shrink Small Outline (ST) - 4.4 mm Body [TSSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



DETAIL B

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		20	
Pitch	e		0.65 BSC	
Overall Height	A	-	-	1.20
Molded Package Thickness	A2	0.80	1.00	1.05
Standoff	A1	0.05	-	0.15
Overall Width	E		6.40 BSC	
Molded Package Width	E1	4.30	4.40	4.50
Molded Package Length	D	6.40	6.50	6.60
Foot Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	Θ	0°	-	8°
Lead Width	b	0.19	-	0.30
Lead Thickness	c	0.09	-	0.20
Bend Radius	R1	0.09	-	-
Bend Radius	R2	0.09	-	-

Notes:

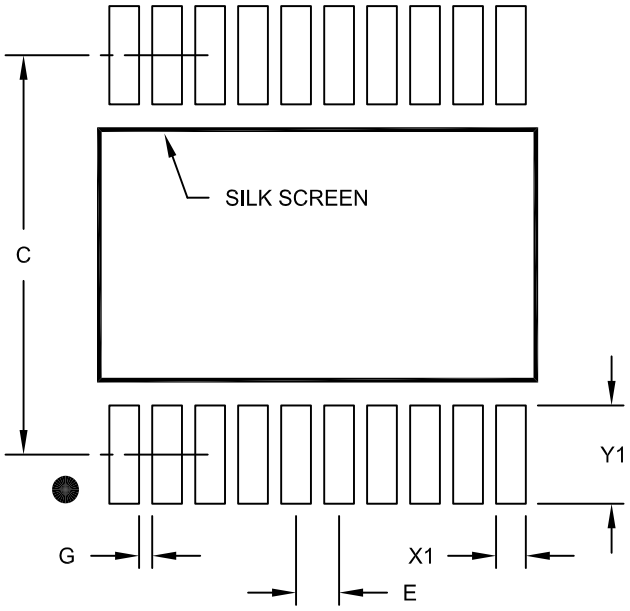
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-088C Sheet 2 of 2

20-Lead Plastic Thin Shrink Small Outline (ST) - 4.4 mm Body [TSSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		5.90	
Contact Pad Width (X20)	X1			0.45
Contact Pad Length (X20)	Y1			1.45
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

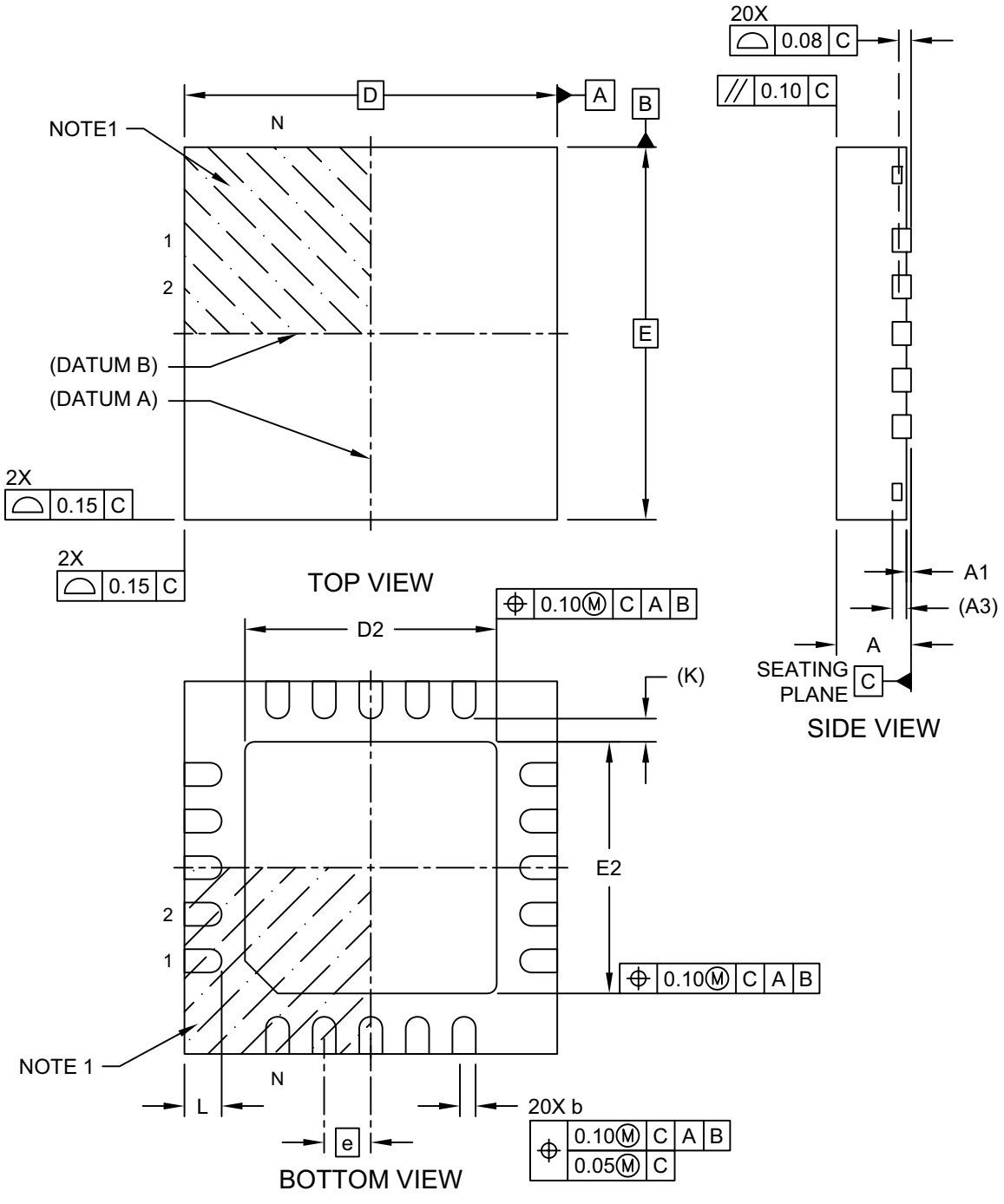
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2088A

27.4 TSB

20-Lead Very, Very Thin Quad Flat, No Lead Package (TSB) - 4x4x0.8 mm Body [WQFN] With 2.7 mm Exposed Pad; Atmel Legacy Global Package Code ZAJ

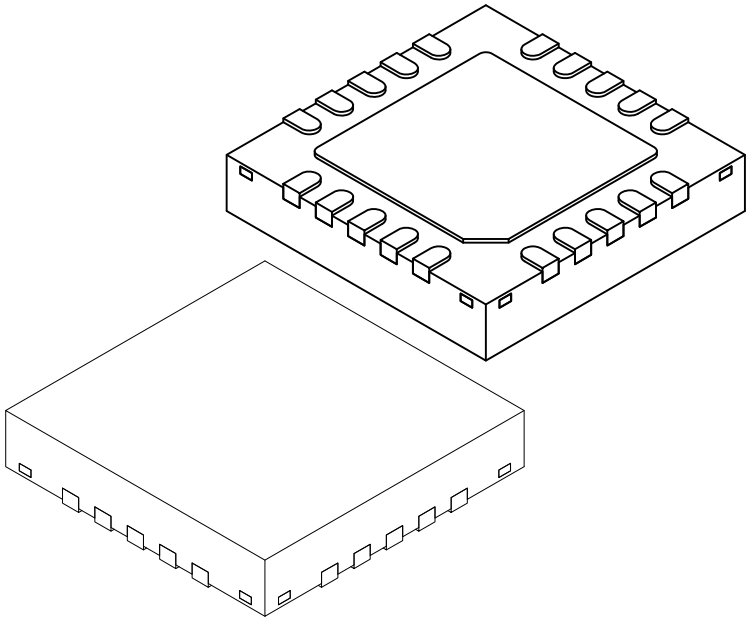
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-21444 Rev A Sheet 1 of 2

20-Lead Very, Very Thin Quad Flat, No Lead Package (TSB) - 4x4x0.8 mm Body [WQFN] With 2.7 mm Exposed Pad; Atmel Legacy Global Package Code ZAJ

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		20	
Pitch	e		0.50 BSC	
Overall Height	A	0.70	0.75	0.80
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Overall Length	D		4.00 BSC	
Exposed Pad Length	D2	2.60	2.70	2.80
Overall Width	E		4.00 BSC	
Exposed Pad Width	E2	2.60	2.70	2.80
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.35	0.40	0.45
Terminal-to-Exposed-Pad	K		0.25 REF	

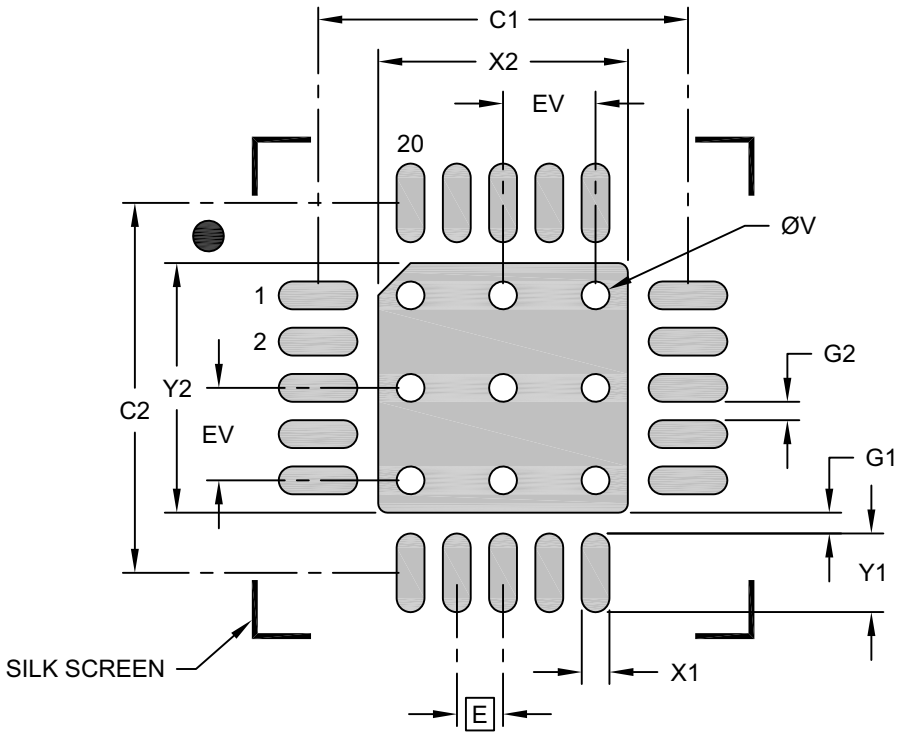
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21444 Rev A Sheet 2 of 2

20-Lead Very, Very Thin Quad Flat, No Lead Package (TSB) - 4x4x0.8 mm Body [WQFN] With 2.7 mm Exposed Pad; Atmel Legacy Global Package Code ZAJ

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	X2			2.70
Optional Center Pad Length	Y2			2.70
Contact Pad Spacing	C1		4.00	
Contact Pad Spacing	C2		4.00	
Contact Pad Width (X20)	X1			0.30
Contact Pad Length (X20)	Y1			0.85
Contact Pad to Center Pad (X20)	G1	0.23		
Contact Pad to Contact Pad (X16)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23444 Rev A

28. Errata

The errata content has been moved to a separate document, *ATtiny87/167 Silicon Errata and Data Sheet Clarification*, located at http://www.microchip.com/DS80000850.

29. Datasheet Revision History

29.1 Rev. A – 01/2020

1. New Microchip document number. Previous version was Atmel data sheet rev. 8265D.
2. Updated Package Information to Microchip style.
3. Updated Ordering Information.
4. Errata has been moved to separate document.
5. Updated "Interrupt Vectors in ATtiny87/167" table.
6. Added parentheses to equations in "Baud rate Generator" section to emphasize the correct order of operations.
7. Added pinout diagram for WQFN20.

29.2 Rev. 8265D – 01/2014

Updated:

- The datasheet updated with Atmel "new blue logo" and addresses
- Corrected the speed (Frequency) specified in the table "ATtiny167" on page 285 to 16 MHz
- The Figure 23-2 on page 263, Figure 23-5 on page 264 and Figure 23-11 on page 268 updated according to the 16 MHz Frequency

29.3 Rev. 8265C – 03/12

Updated:

- "Memory Size Summary" on page 9
- The datasheet status changes from "Preliminary" to "Complete"
- "Ordering Information" on page 284. 32M1-A package replaced by 32NP
- "Packaging Information" on page 286. 32M1-A package replaced by 32NP
- The whole document with Atmel new template that include blue logo and new addresses on the last page.

29.4 Rev. 8265B – 09/10

Updated:

- Section 9.3.3 "Alternate Functions of Port A" on page 84, Bit 3 and Bit 4
- Section 26.2 "ATtiny167" on page 285, ordering codes
- Section 10.11.1 "TCCR0A – Timer/Counter0 Control Register A" on page 107, Bit 1 and Bit 0 descriptions
- Bit syntax throughout the datasheet, e.g. from CS02:0 to CS0[2:0]

29.5 Rev. 8265A – 08/10

Initial revision.

DEVELOPMENT SUPPORT

Move a design from concept to production in record time with Microchip's award-winning development tools. Microchip tools work together to provide state of the art debugging for any project with easy-to-use Graphical User Interfaces (GUIs) in our free MPLAB® X and Atmel Studio Integrated Development Environments (IDEs), and our code generation tools. Providing the ultimate ease-of-use experience, Microchip's line of programmers, debuggers and emulators work seamlessly with our software tools. Microchip development boards help evaluate the best silicon device for an application, while our line of third party tools round out our comprehensive development tool solutions.

Microchip's MPLAB X and Atmel Studio ecosystems provide a variety of embedded design tools to consider, which support multiple devices, such as PIC® MCUs, AVR® MCUs, SAM MCUs and dsPIC® DSCs. MPLAB X tools are compatible with Windows®, Linux® and Mac® operating systems while Atmel Studio tools are compatible with Windows.

Go to the following website for more information and details:

https://www.microchip.com/development-tools/

THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://microchip.com/support

© 2020 Microchip Technology Inc.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2020, Microchip Technology Incorporated, All Rights Reserved.

ISBN:978-1-5224-5396-3

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

© 2020 Microchip Technology Inc.