Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- Non-Volatile Program and Data Memories
  - 4K Bytes of In-System Programmable Program Memory Flash
  - 64 Bytes of In-System Programmable EEPROM
  - 256 Bytes of Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C(1)
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-Bit Timer/Counters with two PWM Channels, Each
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-Chip Analog Comparator
  - 10-bit ADC
    - 4 Single-Ended Channels
  - Universal Serial Interface
  - Boost Converter
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Pin Change Interrupt on 16 Pins
  - Low Power Idle, ADC Noise Reduction and Power-Down Modes
  - Enhanced Power-On Reset Circuit
  - Programmable Brown-Out Detection Circuit
  - Internal Calibrated Oscillator
  - Temperature Sensor On Chip
- I/O and Packages
  - Available in 20-Pin SOIC and 20-Pin QFN/MLF
  - 16 Programmable I/O Lines
- Operating Voltage:
  - 0.7 – 1.8V (via On-Chip Boost Converter)
  - 1.8 – 5.5V (Boost Converter Bypassed)
- Speed Grade
  - Using On-Chip Boost Converter
    - 0 – 4 MHz
  - External Power Supply
    - 0 – 4 MHz @ 1.8 – 5.5V
    - 0 – 8 MHz @ 2.7 – 5.5V
- Low Power Consumption
  - Active Mode, 1 MHz System Clock (Without Boost Converter)
    - 400 µA @ 3V
  - Power-Down Mode (Without Boost Converter)
    - 150 nA @ 3V

Note: 1. See “Data Retention” on page 6 for details.

8-bit AVR® Microcontroller with 4K Bytes In-System Programmable Flash and Boost Converter

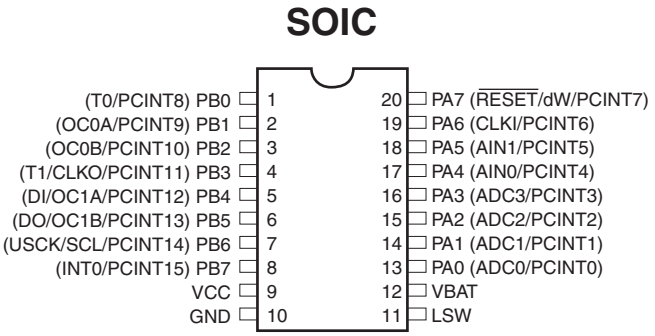
ATtiny43U

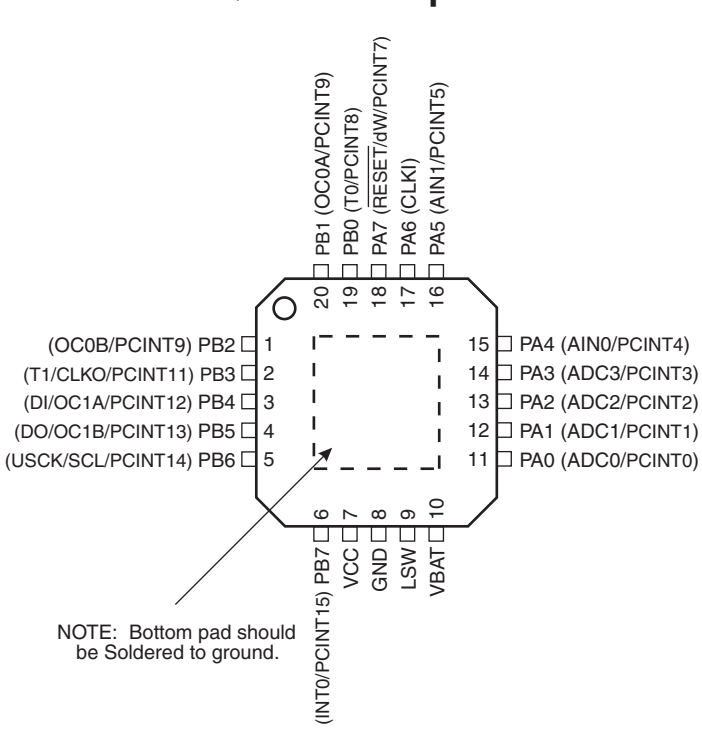
Preliminary

Rev. 8048B–AVR–03/09

1. Pin Configurations

Figure 1-1. Pinout of ATtiny43U





1.1 Pin Descriptions

1.1.1 V_{CC}

Supply voltage.

1.1.2 GND

Ground.

1.1.3 Port A (PA7:PA0)

Port A is a 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 except PA7 which has the $\overline{\text{RESET}}$ capability. To use pin PA7 as an I/O pin, instead of RESET pin, program (‘0’) RSTDISBL fuse. 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 has an alternate functions as analog inputs for the ADC, analog comparator, timer/counter, SPI and pin change interrupt as described in “Alternate Port Functions” on page 67.

1.1.4 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 20-4 on page 158. Shorter pulses are not guaranteed to generate a reset.

1.1.5 Port B (PB7:PB0)

Port B is a 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 as listed in Section 11.3 “Alternate Port Functions” on page 67.

1.1.6 LSW

Boost converter external inductor connection. Connect to ground when boost converter is disabled permanently.

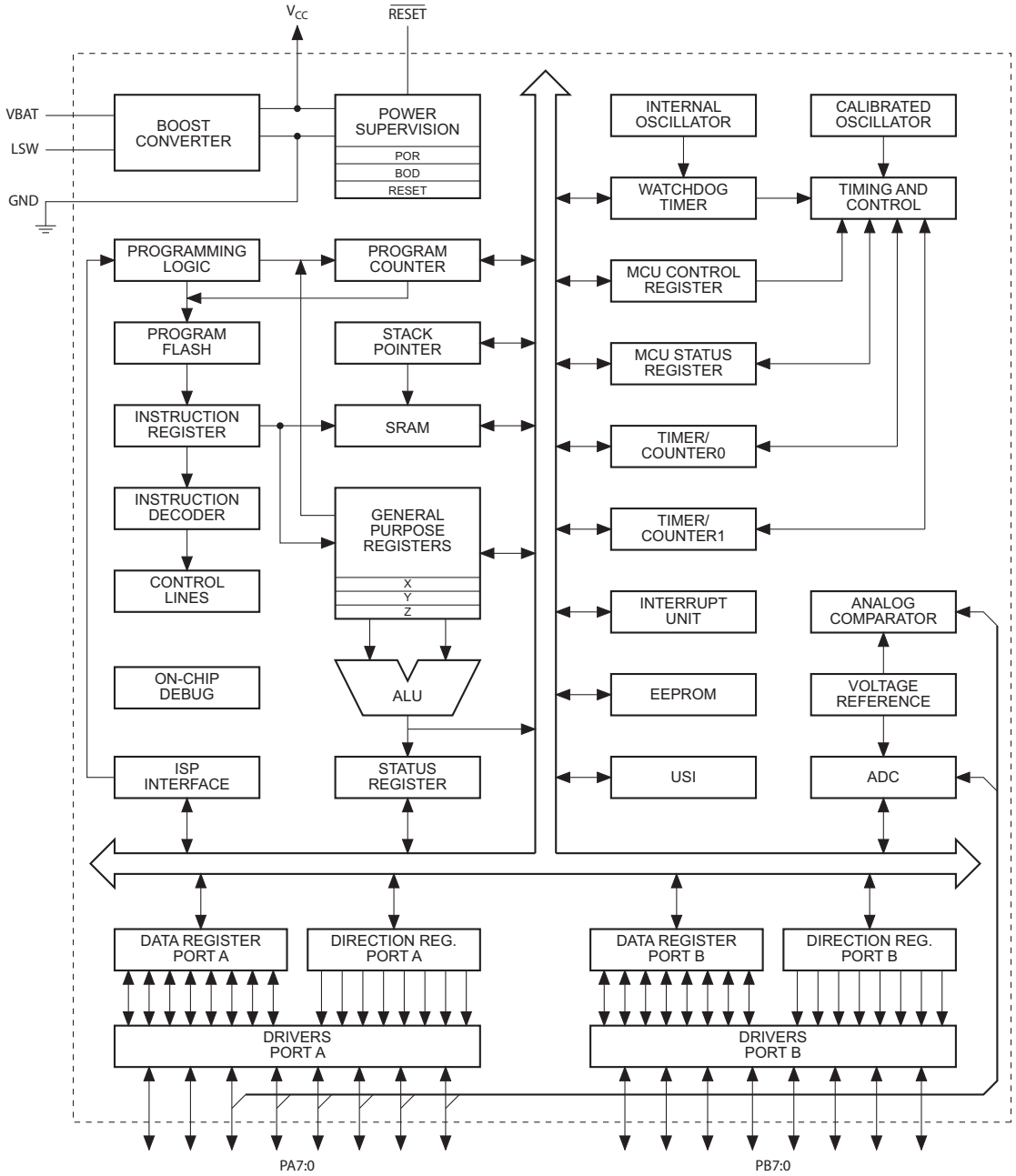
1.1.7 V_{BAT}

Battery supply voltage. Connect to ground when boost converter is disabled permanently.

2. Overview

The ATtiny43U 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 ATtiny43U achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Figure 2-1. Block Diagram



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

architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATtiny43U provides the following features: 4K byte of In-System Programmable Flash, 64 bytes EEPROM, 256 bytes SRAM, 16 general purpose I/O lines, 32 general purpose working registers, two 8-bit Timer/Counters with two PWM channels, Internal and External Interrupts, a 4-channel 10-bit ADC, Universal Serial Interface, a programmable Watchdog Timer with internal Oscillator, internal calibrated 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.

A special feature of ATtiny43U is the built-in boost voltage converter, which provides 3V supply voltage from an external, low voltage.

The device is manufactured using Atmel'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 ATtiny43U AVR is supported by a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

3. About

3.1 Resources

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

3.2 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.

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

3.3 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.

3.4 Disclaimer

Typical values contained in this data sheet 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 is characterized.

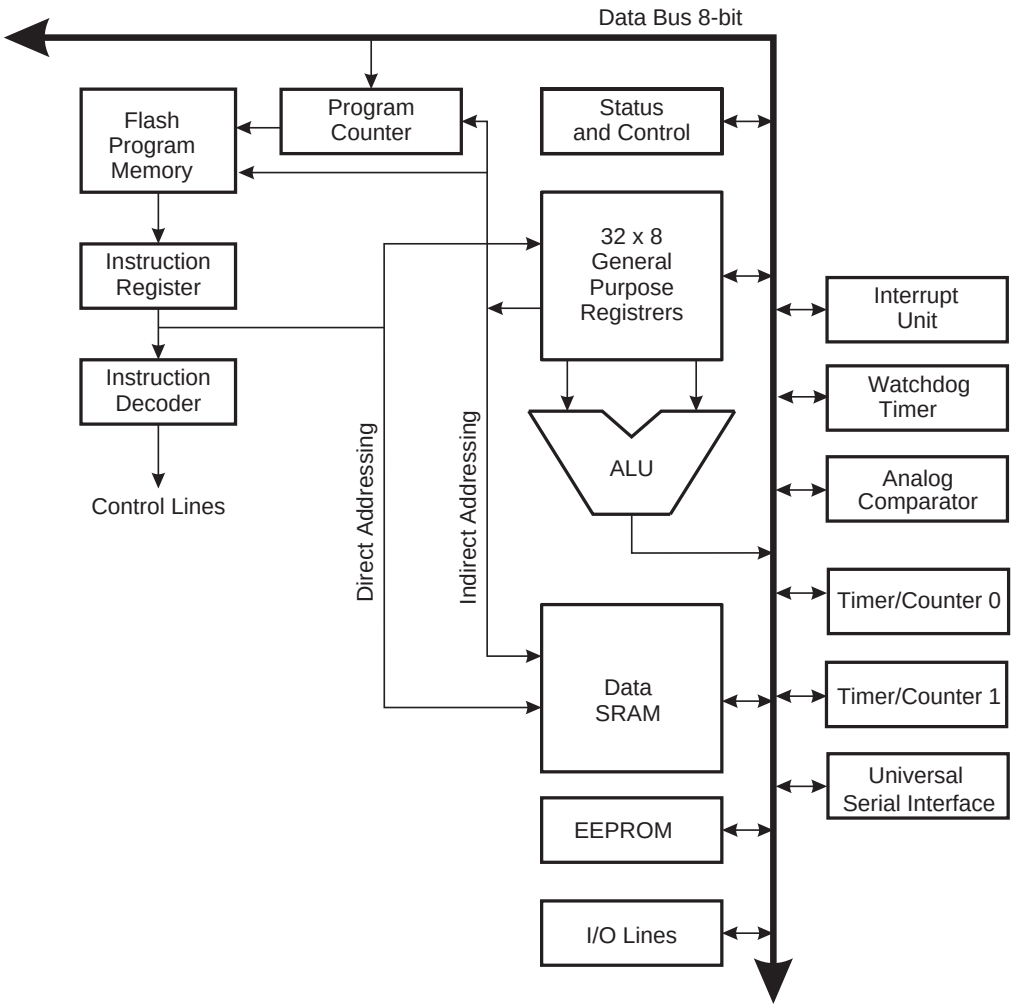
4. AVR CPU Core

4.1 Introduction

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.

4.2 Architectural Overview

Figure 4-1. 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.

4.3 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.

4.4 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.

4.4.1 SREG - AVR Status Register

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 $\oplus$ V**

The S-bit is always an exclusive or between the Negative Flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

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

- **Bit 1 – Z: Zero Flag**

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

- **Bit 0 – C: Carry Flag**

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

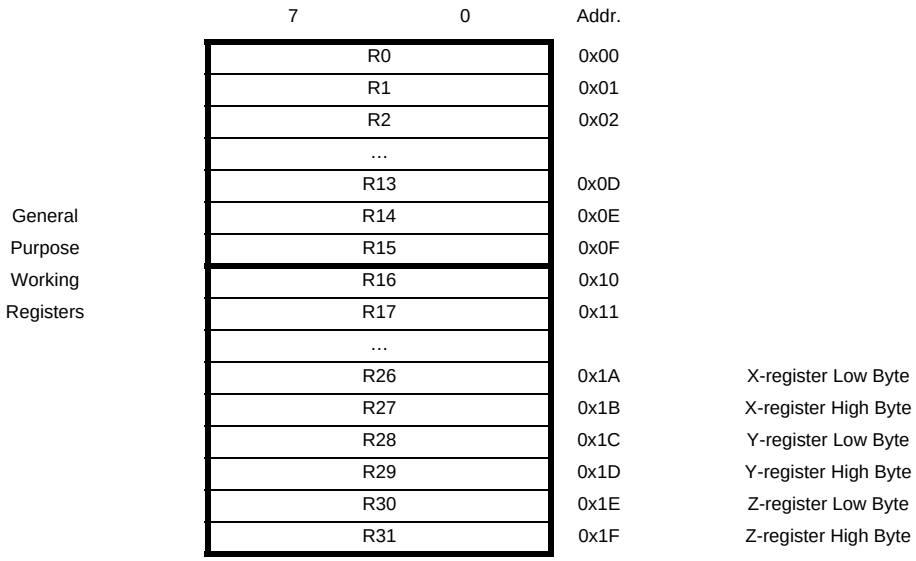
4.5 General Purpose Register File

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 4-2 on page 10 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4-2. 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 4-2, 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.

4.5.1 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 4-3 on page 11.

Figure 4-3. 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).

4.6 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. Note that the Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack. The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer.

The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM, see Figure 5-2 on page 16.

See Table 4-1 for Stack Pointer details.

Table 4-1. Stack Pointer instructions

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

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

4.6.1 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	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

4.7 Instruction Execution Timing

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

Figure 4-4 on page 12 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 4-4. The Parallel Instruction Fetches and Instruction Executions

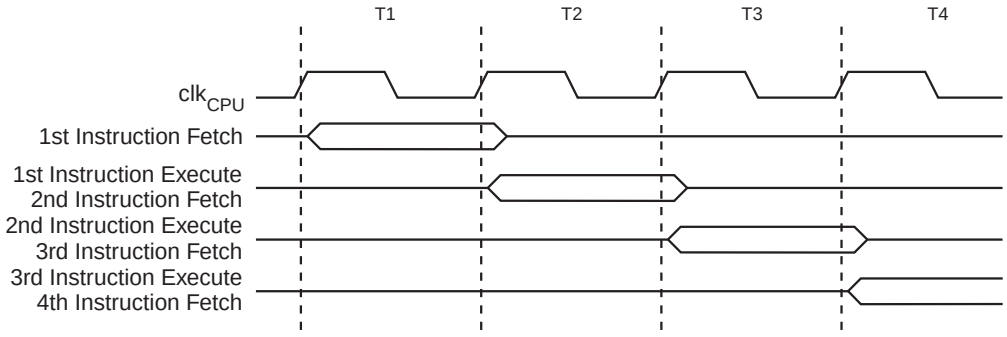
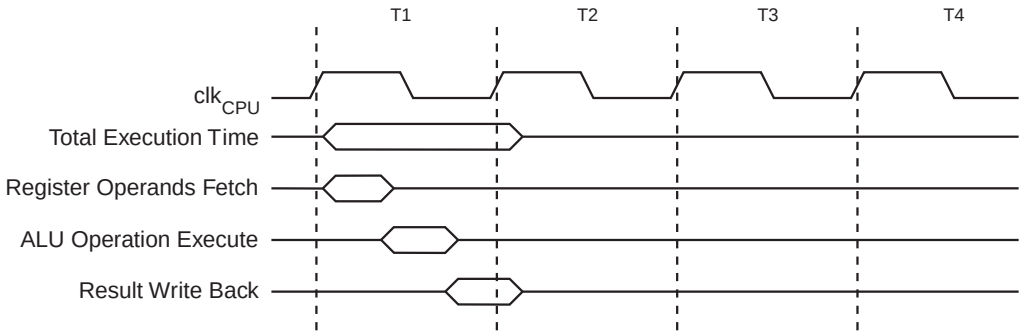


Figure 4-5 on page 12 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 4-5. Single Cycle ALU Operation



4.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate Program Vector in the Program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.

The lowest addresses in the Program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 57. 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.

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) */
```

4.8.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the Program Vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

5. Memories

5.1 Overview

This section describes the different memories in ATtiny43U. The AVR architecture has two main memory spaces, the Data memory and the Program memory space. In addition, the ATtiny43U features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

5.2 In-System Re-programmable Flash Program Memory

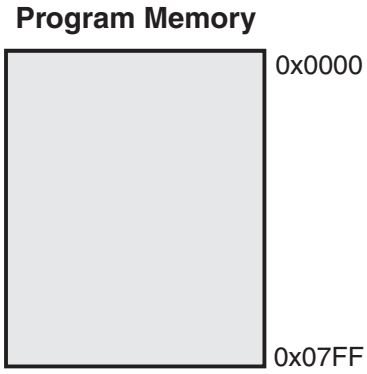
The ATtiny43U contains 4K byte On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 2048 x 16.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATtiny43U Program Counter (PC) is 11 bits wide, thus addressing the 2048 Program memory locations. “Memory Programming” on page 139 contains a detailed description on Flash data downloading.

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 “Instruction Execution Timing” on page 12.

Figure 5-1. Program Memory Map



5.3 SRAM Data Memory

Figure 5-2 on page 16 shows how the ATtiny43U SRAM Memory is organized.

The low Data memory locations address both the Register File, the I/O memory and the internal data SRAM, as follows:

- The first 32 locations address the Register File
- The next 64 locations address the standard I/O memory
- The last 256 locations address the internal data SRAM

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

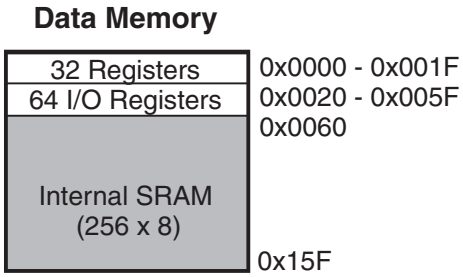
The direct addressing reaches the entire data space.

The Indirect with Displacement mode 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, and the 256 bytes of internal data SRAM in ATtiny43U are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 10.

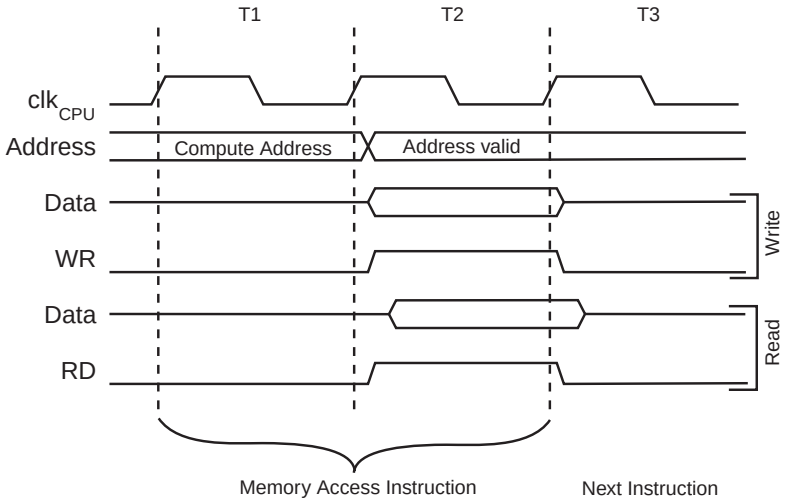
Figure 5-2. Data Memory Map



5.3.1 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 5-3 on page 16.

Figure 5-3. On-chip Data SRAM Access Cycles



5.4 EEPROM Data Memory

The ATtiny43U contains 64 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and

the EEPROM Control Register. For a detailed description of Serial data downloading to the EEPROM, see “Serial Programming” on page 151.

5.4.1 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 5-1 on page 21. 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 19 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. See “Atomic Byte Programming” on page 17 and “Split Byte Programming” on page 17 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.

5.4.2 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 EEAR 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.

5.4.3 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).

5.4.4 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.

5.4.5 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 28.

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 (r17) in address register
  out EEAR, r17
  ; Write data (r19) to data register
  out EEDR,r19
  ; 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 (r17) in address register
  out  EEAR, 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;
}
```

5.4.6 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.

5.5 I/O Memory

The I/O space definition of the ATtiny43U is shown in “Register Summary” on page 167.

All 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. See 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.

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.

5.5.1 General Purpose I/O Registers

ATtiny43U 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. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

5.6 Register Description

5.6.1 EEAR – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	-	-	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	X	X	X	X	

- **Bits 7:6 – Res: Reserved Bit**

These bits are reserved and will always read zero.

- **Bits 5:0 – EEAR[5:0]: EEPROM Address**

The EEPROM Address Register – EEAR – specifies the EEPROM address. The EEPROM data bytes are addressed linearly in the range 0...(64-1). The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

5.6.2 EEDR – EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	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.

5.6.3 EECR – EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	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	

- **Bit 7 – Res: Reserved Bit**

These bits are reserved and will always read zero. For compatibility with future AVR devices, always write this bit to zero. After reading, mask out this bit.

- **Bit 6 – Res: Reserved Bit**

These bits are reserved and will always read zero.

- **Bits 5, 4 – EEPM1 and EEPM0: EEPROM 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 5-1. 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 5-1. EEPROM Mode Bits

EEPM1	EEPM0	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.

5.6.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	MSB							LSB	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	

5.6.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x14 (0x34)	MSB							LSB	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	

5.6.6 GPIOR0 – General Purpose I/O Register 0

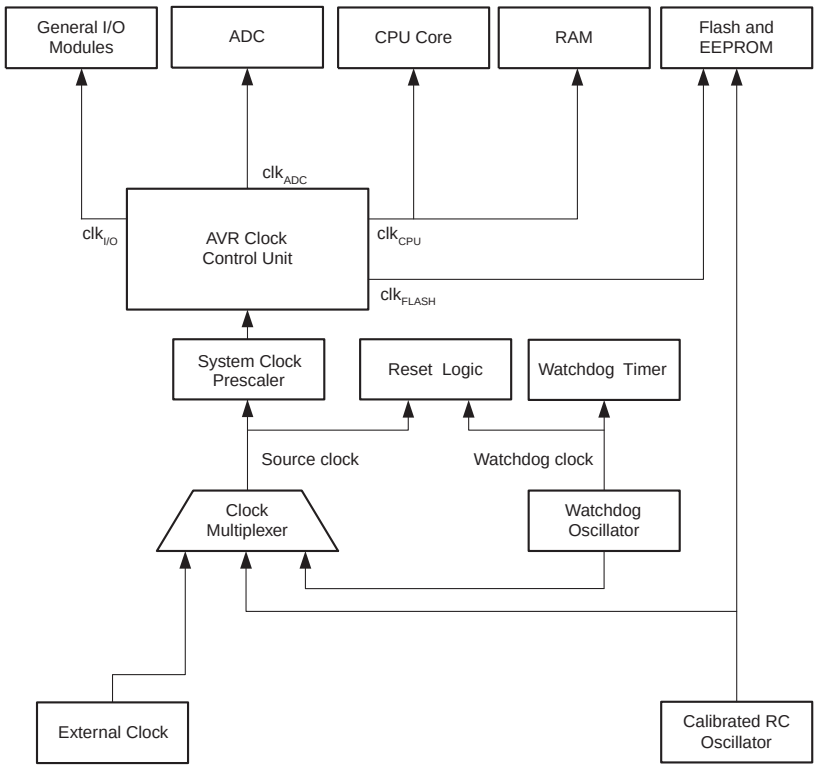
Bit	7	6	5	4	3	2	1	0	
0x13 (0x33)	MSB							LSB	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6. System Clock and Clock Options

6.1 Clock Systems and their Distribution

Figure 6-1 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in "Power Management and Sleep Modes" on page 31. The clock systems are detailed below.

Figure 6-1. Clock Distribution



6.1.1 CPU Clock – clk_{CPU}

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

6.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters. 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. Also note that start condition detection in the USI module is carried out asynchronously when $clk_{I/O}$ is halted.

6.1.3 Flash Clock – clk_{FLASH}

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

6.1.4 ADC Clock – clk_{ADC}

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

6.2 Clock Sources

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

Table 6-1. Device Clocking Options Select[1]

Device Clocking Option	CKSEL3..0
External Clock (see page 24)	0000
Calibrated Internal 8 MHz Oscillator (see page 25)	0010
Internal 128 kHz Oscillator (see page 26)	0011
Reserved	0001, 0100-1111

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

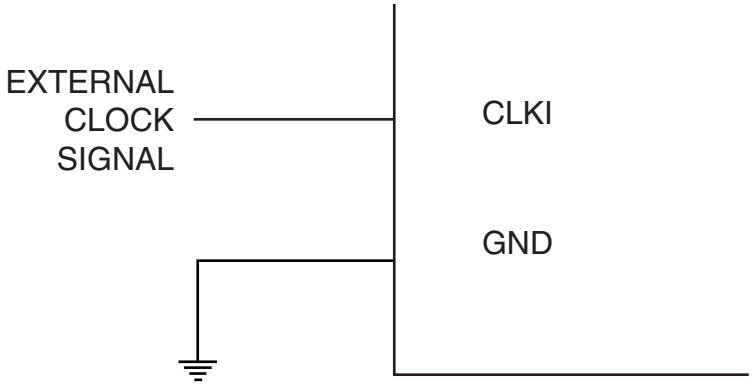
6.2.1 External Clock

To drive the device from an external clock source, CLKI should be driven as shown in Figure 6-2 on page 24. To run the device on an external clock, the CKSEL Fuses must be programmed to “0000” (see Table 6-2).

Table 6-2. Crystal Oscillator Clock Frequency

CKSEL3..0	Frequency
0000	0 - 8 MHz

Figure 6-2. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by SUT Fuses as shown in Table 6-3.

Table 6-3. Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset	Recommended Usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

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

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 "Power Management and Sleep Modes" on page 31 for details.

6.2.2 Calibrated Internal 8 MHz Oscillator

By default, the Internal RC Oscillator provides an approximate 8.0 MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. See Table 20-2 on page 157 for more details. The device is shipped with the CKDIV8 Fuse programmed. See "System Clock Prescaler" on page 27 for more details.

This clock may be selected as the system clock by programming the CKSEL Fuses as shown in Table 6-4. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. The accuracy of this calibration is shown as Factory calibration in Table 20-2 on page 157.

By changing the OSCCAL register from SW, see "OSCCAL – Oscillator Calibration Register" on page 28, 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 20-2 on page 157.

When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. For more information on the pre-programmed calibration value, see the section "Calibration Byte" on page 142.

Table 6-4. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency (MHz)
0010(1)	8.0

Notes: 1. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 6-5 below.

Table 6-5. Start-up times for Internal Calibrated RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	14CK[1]	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10[2]	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

Note: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4 ms to ensure programming mode can be entered.
2. The device is shipped with this option selected.

6.2.3 Internal 128 kHz 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 select as the system clock by programming the CKSEL Fuses to “11” as shown in Table 6-6 below.

Table 6-6. 128 kHz Internal Oscillator Operating Modes

CKSEL3..0	Nominal Frequency
0011	128 kHz

When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 6-7 below.

Table 6-7. Start-up Times for the 128 kHz Internal Oscillator

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset	Recommended Usage
00	6 CK	14CK[1]	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

Note: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4 ms to ensure programming mode can be entered.

6.2.4 Default Clock Source

The device is shipped with CKSEL = "0010", SUT = "10", and CKDIV8 programmed. The default clock source is therefore the internal RC oscillator running at 8.0 MHz with the longest start-up time and an initial system clock prescale setting of 8, resulting in a 1 MHz system clock. The default setting ensures every user can make the desired clock source setting using any available programming interface.

6.2.5 Clock Startup Sequence

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

To ensure sufficient V_{CC}, the device issues an internal reset with a time-out delay (t_{TOUT}) after the device reset is released by all other reset sources. The section "System Control and Reset" on page 48 describes the start conditions for the internal reset. The delay (t_{TOUT}) is timed from the Watchdog Oscillator and the number of cycles in the delay is set by the SUTn and CKSELn fuse bits. The available delays are shown in Table 6-8.

Table 6-8. Number of Watchdog Oscillator Cycles

Typ Time-out (V_{CC} = 5.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
0 ms	0 ms	0
4.1 ms	4.3 ms	512
65 ms	69 ms	8K (8,192)

Note: The frequency of the Watchdog Oscillator is voltage dependent as shown in TBD.

The main purpose of the delay is to keep the AVR in reset until V_{CC} has risen to a sufficient level. The delay will not monitor the actual voltage and, hence, the user must make sure the delay time is longer than the V_{CC} rise time. If this is not possible, an internal or external Brown-Out Detection circuit should be used. A BOD circuit ensures there is sufficient V_{CC} before it releases the reset line, and the time-out delay can then be disabled. It is not recommended to disable the time-out delay without implementing a Brown-Out Detection circuit.

The oscillator is required to oscillate for a minimum number of cycles before the clock is considered stable. An internal ripple counter monitors the oscillator output clock, and keeps the internal reset active for a given number of clock cycles. The reset is then released and the device will start to execute.

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

6.3 System Clock Prescaler

The ATtiny43U has a system clock prescaler, which means the system clock can be divided as described in section "CLKPR – Clock Prescale Register" on page 28. This feature can be used to lower system clock frequency and decrease the power consumption at times when requirements 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. Clock signals $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 20-4 on page 158.

6.3.1 Switching Time

When changing prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures that no intermediate frequency is higher than either the clock frequency corresponding to the previous setting or the clock frequency corresponding to the new setting. The ripple counter of the prescaler runs at the same frequency as the undivided clock, which may be higher than the CPU's clock frequency. Hence, even if it was readable, it is not possible to determine the state of the prescaler, and it is not possible to predict the exact time it takes to switch from one clock division to the other. 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, two active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

6.4 Clock Output Buffer

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

6.5 Register Description

6.5.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
	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 20-2 on page 157. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 20-2 on page 157. 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 CAL6..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.

6.5.2 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
	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 CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

- **Bits 3:0 – CLKPS[3:0]: Clock Prescaler Select 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 affected. The division factors are given in Table 6-9.

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

Table 6-9. 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

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 bitsin 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 to make sure the write procedure is not interrupted.

7. 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 "Software BOD Disable" on page 32 for more details.

7.1 Sleep Modes

Figure 6-1 on page 23 presents the different clock systems in ATtiny43U, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 7-1 below shows the different sleep modes and their wake-up sources.

Table 7-1. 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}	Main Clock Source Enabled	INT0 and Pin Change	SPM/EEPROM Ready	ADC	Other I/O	Watchdog Interrupt
Idle			X	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X(1)	X	X		X
Power-down						X(1)				X

Note: 1. For INT0, only level interrupt.

To enter any of the sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM1:0 bits in the MCUCR Register select which sleep mode (Idle, ADC Noise Reduction or Power-down) will be activated by the SLEEP instruction. See Table 7-2 on page 34 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.

Note that if a level triggered interrupt is used for wake-up the changed level must be held for some time to wake up the MCU (and for the MCU to enter the interrupt service routine). See "External Interrupts" on page 58 for details.

7.1.1 Idle Mode

When the SM1:0 bits are written to 00, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing Analog Comparator, ADC, 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 Timer Overflow. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the “ACSR – Analog Comparator Control and Status Register” on page 113. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

7.1.2 ADC Noise Reduction Mode

When the SM1..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, and the Watchdog to continue operating (if enabled). This sleep mode 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 form the ADC Conversion Complete interrupt, only an External Reset, a Watchdog Reset, a Brown-out Reset, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or a pin change interrupt can wake up the MCU from ADC Noise Reduction mode.

7.1.3 Power-Down Mode

When the SM1..0 bits are written to 10, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the Oscillator is stopped, while the external interrupts, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, an external level interrupt on INT0, or a pin change interrupt can wake up the MCU. This sleep mode halts all generated clocks, allowing operation of asynchronous modules only.

7.2 Software BOD Disable

When the Brown-out Detector (BOD) is enabled by BODLEVEL fuses (see Table 19.2 on page 140), the BOD is actively monitoring the power supply voltage during a sleep period. To save power, it is possible for software to disable the BOD in Power-Down Mode (see “Power-Down Mode” on page 32). The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses. If disabled by software, the BOD is turned off immediately after entering the sleep mode and automatically turned on upon wake-up. 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 the same as the wake-up time from RESET. This is in order to ensure the BOD is working correctly before the MCU continues executing code.

BOD disable is controlled by bit 7 (BODS — BOD Sleep) of MCU Control Register, see “MCUCR – MCU Control Register” on page 34. Writing this bit to one turns off the BOD in Power-Down Mode, while a zero in this bit keeps BOD active. The default setting is zero, i.e. BOD active.

Writing to the BODS bit is controlled by a timed sequence and an enable bit, see “MCUCR – MCU Control Register” on page 34.

7.3 Power Reduction Register

The Power Reduction Register (PRR), see “PRR – Power Reduction Register” on page 35, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozenand 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

7.4 Minimizing Power Consumption

There are several issues 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.

7.4.1 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. See "Analog to Digital Converter" on page 115 for details on ADC operation.

7.4.2 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 the 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. See "Analog Comparator" on page 112 for details on how to configure the Analog Comparator.

7.4.3 Brown-out Detector

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled it will be active in all sleep modes and, hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. See "Brown-out Detection" on page 50 and "Software BOD Disable" on page 32 for details on how to configure the Brown-out Detector.

7.4.4 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. See "Internal Voltage Reference" on page 51 for details on the start-up time.

7.4.5 Watchdog Timer

If the Watchdog Timer is not needed in the application, this 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. See "Watchdog Timer" on page 51 for details on how to configure the Watchdog Timer.

7.4.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing 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. See the section "Digital Input Enable and Sleep Modes" on page 66 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or has 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 Register (DIDR0). See "DIDR0 – Digital Input Disable Register 0" on page 130 for details.

7.5 Register Description

7.5.1 MCUCR – MCU Control Register

The MCU Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – BODS: BOD Sleep**

The BODS bit must be written to logic one in order to turn off BOD during sleep, see Table 7-1 on page 31. 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 – 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.

- **Bits 4, 3 – SM[1:0]: Sleep Mode Select Bits 2:0**

These bits select between the three available sleep modes as shown in Table 7-2 below.

Table 7-2. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle

Table 7-2. Sleep Mode Select

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

- **Bit 2 – 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.

7.5.2 PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	PRE2	PRE1	PRE0	–	PRTIM1	PRTIM0	PRUSI	PRADC	PRR
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 7:5 – PRE[2:0]: Prepared Read Enable**

These bits are used for prepared read operations. See sections “Software Control of Boost Converter” on page 42 and “ADCSRB – ADC Control and Status Register B” on page 47.

- **Bit 4 – Res: Reserved Bit**

This bit is reserved and will always read zero.

- **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. 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**

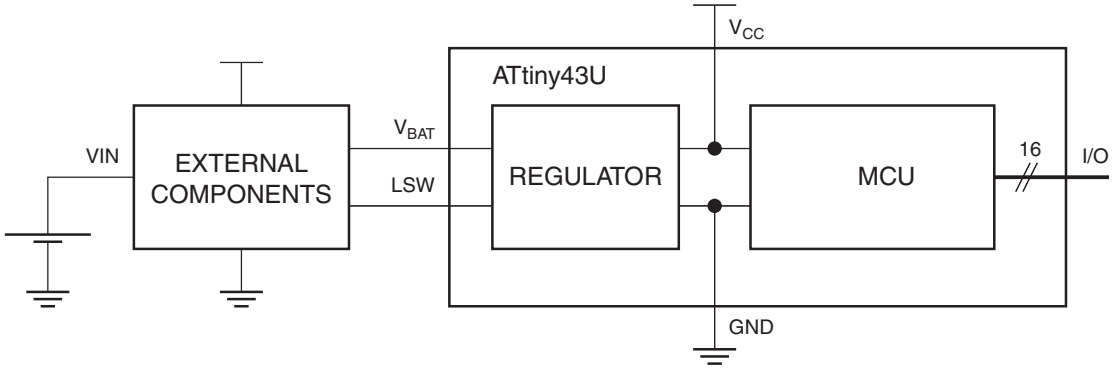
Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

8. Power Supply and On-Chip Boost Converter

In order to work properly microcontrollers typically require a supply voltage level that can not be provided by battery packs of less than two or three battery cells. This constraint adds to size, cost and complexity of the design. The integrated boost converter of ATtiny43U bridges the gap between minimum supply voltage of the device and typical output voltages of single-cell standard, alkaline, Lithium, NiCd or NiMH batteries. The boost converter enables the device to be powered from a source with a supply voltage well below 1V.

A block diagram illustrating the use of the boost converter is shown in Figure 8-1, below.

Figure 8-1. Block Diagram of Boost Converter Usage.



8.1 Overview

A boost converter is a device that converts a DC voltage to a higher level. The integrated boost converter of ATtiny43U provides the microcontroller (and its peripherals) with a fixed supply voltage, generated from an external supply of lower voltage.

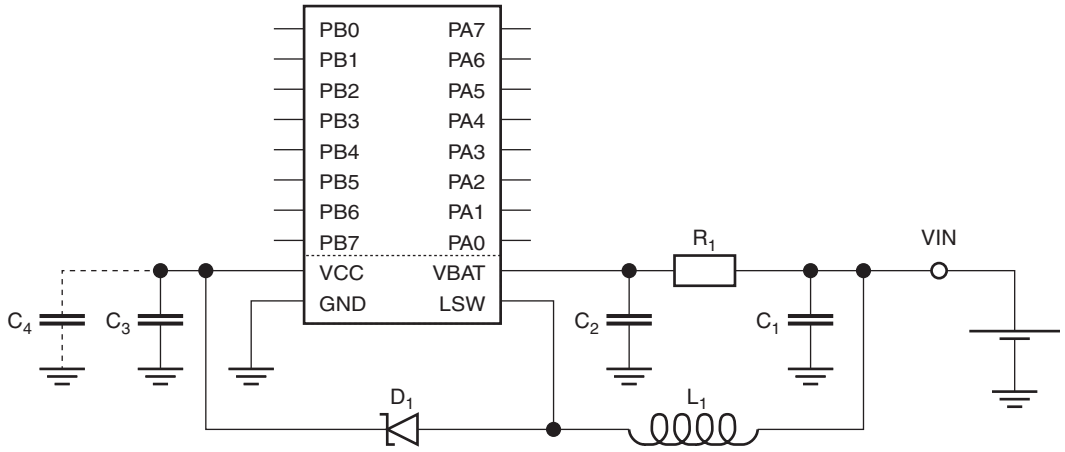
The ATtiny43U boost converter is a switching type, step-up converter that uses an external inductor, a diode and bypass capacitors. The boost converter is self-sufficient, completely independent and does not need any control from the MCU. The converter starts automatically as soon as there is sufficient voltage at the V_{BAT} pin. See Table 20-7 on page 159 for electrical characteristics.

The microcontroller starts as soon as the regulated output of the boost converter rises above power-on and brown-out reset levels (if enabled), as described in section "System Control and Reset" on page 48. After the MCU is released from reset and has started running the application software can then measure the battery voltage and decide if there is sufficient voltage to continue operation.

The boost converter continuously switches between storing energy in and draining energy from the external inductor. During the charge phase the current through the inductor ramps up at a rate determined by the converter input voltage. During the discharge phase energy stored in the inductor is released to the load and the current in the inductor ramps down at a rate determined by the difference between the input and output voltages.

The boost converter requires some external components to operate. See Figure 8-2 on page 37 for component placement. The circuit is completed by inserting an inductor between node VIN and pin LSW, and a Schottky diode between pins LSW and V_{CC}. In addition, an input capacitor and external bypass capacitor from V_{CC} to GND are typically required. See "Typical Applications" on page 46 for more details.

Figure 8-2. Typical Connection of Boost Converter.



When the boost converter is not connected the microcontroller can be powered directly from an external source and is then subject to the standard supply voltage limits defined in “Electrical Characteristics” on page 155.

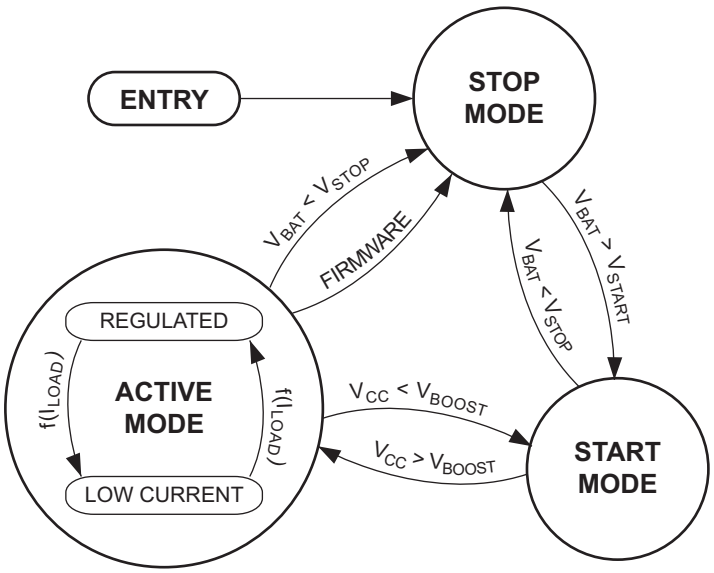
It is recommended to disable the Brown-Out Detection (BOD) circuit when using the integrated boost converter. This is because supply voltage of the microcontroller can drop to lowest BOD levels during regular operation. See “Brown-out Detection” on page 50.

8.2 Modes of Operation

The boost converter has three main modes of operation; Stop, Start, and Active. Operation begins from Stop Mode and is transferred to Start Mode when input voltage, V_{BAT}, is sufficiently high for stable operation. When the converter has managed to raise the output voltage, V_{CC}, to a sustainable level control is then transferred to the main mode of operation, Active Mode.

The modes of operation are illustrated in Figure 8-3, below.

Figure 8-3. Operating Modes of Boost Converter.



8.2.1 Stop Mode

The boost converter enters Stop Mode (see Figure 8-3 on page 37 for modes of operation) when input voltage ,V_{BAT}, is below the shutdown voltage, V_{STOP} (see Table 20-7 on page 159). Alternatively, the boost converter enters Stop Mode when instructed by firmware to do so.

In this mode of operation the boost converter is not active and current consumption is decreased to a minimum. This is in order to prevent battery discharge and to avoid battery damage.

The voltage at the converter output, V_{CC}, has no effect in this mode. The converter monitors the voltage of the input pin, V_{BAT}, and waits for it to rise above the start voltage, V_{START} (see Table 20-7 on page 159). When there is sufficient voltage at the input the converter exits Stop Mode and enters Start Mode.

8.2.2 Start Mode

The converter switches from Stop Mode to Start Mode (see Figure 8-3 on page 37) as soon as the input voltage, V_{BAT}, goes above the start voltage, V_{START} (see Table 20-7 on page 159). Alternatively, the boost converter switches from Active Mode to Start Mode when the output voltage drops below V_{BOOST}.

In this mode of operation the boost converter pumps up the V_{CC} voltage by switcing at a 50% duty cycle and high frequency, until it reaches V_{BOOST}.

All loads should be disconnected during this stage. The boost converter is designed to remain in Start Mode for a short moment, but it is optimised to bring the microcontroller on line, only. If there are additional loads connected to the V_{CC} pin the boost converter may not reach the V_{CC} voltage required to go into Active Mode.

8.2.3 Active Mode

The converter enters Active Mode of operation (see Figure 8-3 on page 37) when both input and output voltages are sufficiently high. This means that V_{BAT} is above V_{START} and V_{CC} is above V_{BOOST}. If input voltage drops below V_{STOP} or output voltage drops below V_{BOOST} the converter will exit Active Mode. Alternatively, firmware can force the boost converter to exit Active Mode and enter Stop Mode.

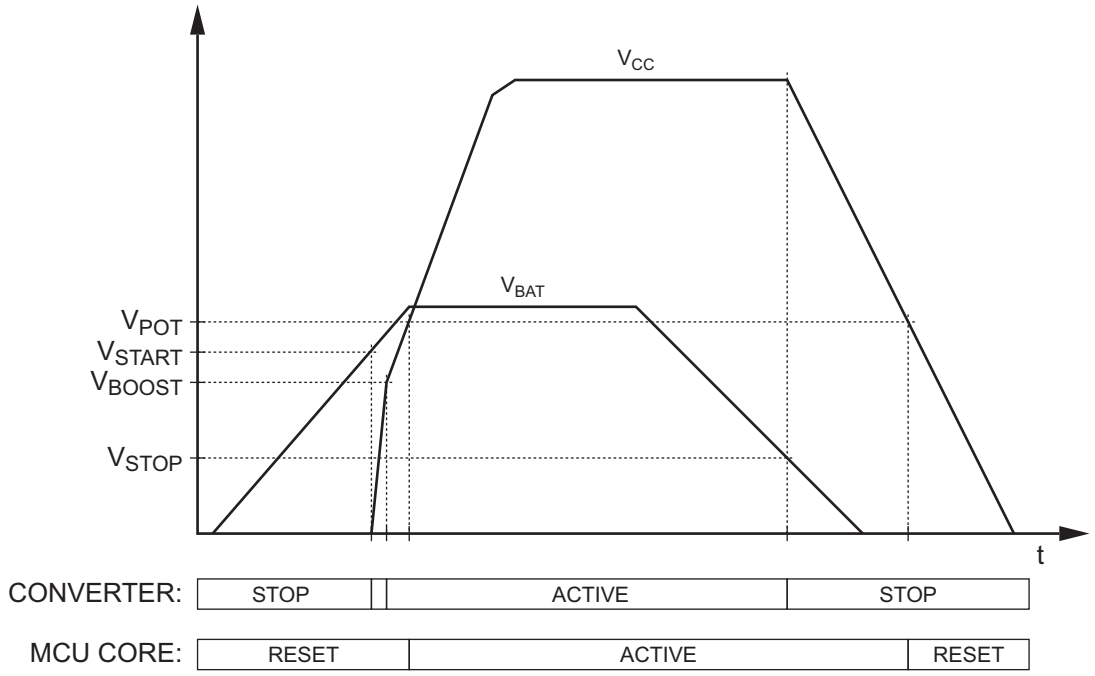
In this mode of operation, the boost converter keeps V_{CC} within limits given in Table 20-7 on page 159 by constantly adjusting the duty cycle between energy charge and discharge phases. The duty cycle is affected mainly by input voltage, V_{BAT}, load current, I_{LOAD}, and temperature.

By default, the boost converter operates in Active Regulated Mode but when load current drops sufficiently low it will enter Active Low Current Mode, as explained in “Output Voltage versus Load Current” . In Active Low Current Mode current consumption is minimised on the expense of output voltage regulation.

8.2.4 Examples

Figure 8-4 illustrates operating modes and input and output voltages of the boost converter. As input voltage, V_{BAT}, rises above V_{START} (see “Boost Converter Characteristics” on page 159) the converter enters Start Mode and output voltage, V_{CC}, begins to rise. At V_{BOOST} output voltage the converter exits Start Mode and goes into Active Mode. When output voltage exceeds the power-on threshold V_{POT} (see “System and Reset Characteristics” on page 158) the microcontroller is released from reset.

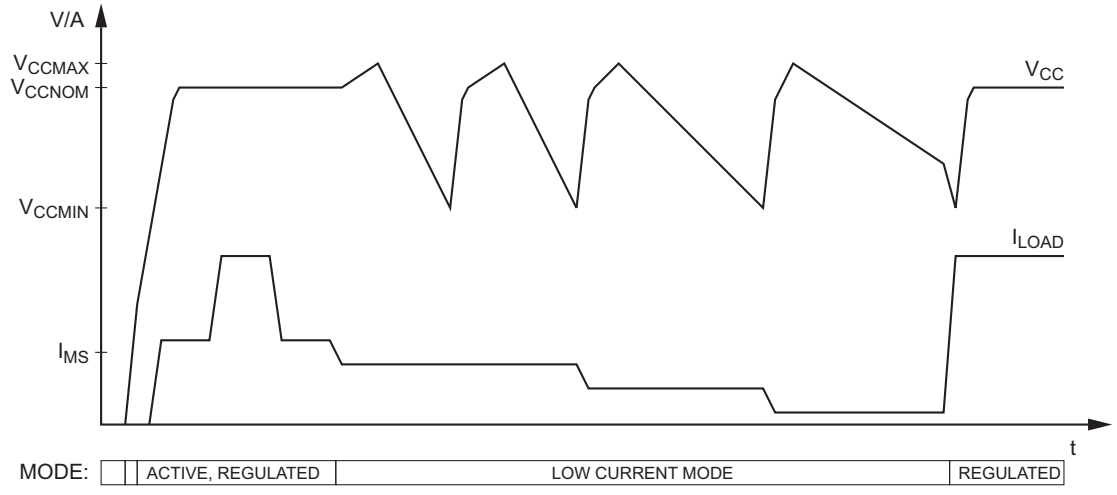
Figure 8-4. Input and Output Voltages of Boost Converter.



When input voltage V_{BAT} falls below V_{STOP} the converter enters Stop Mode and output voltage V_{CC} begins to fall. When converter output voltage, i.e. the supply voltage of the microcontroller, falls below V_{POT} the MCU will go into reset.

Figure 8-5 illustrates how the boost converter output changes with load current. As converter output voltage rises above the power-on threshold the microcontroller is brought on-line and current consumption steps up to a level sufficiently high for the converter to remain in Active Regulated Mode of operation.

Figure 8-5. Output Voltage vs. Load Current of Boost Converter.



Note: The figure is not to scale. Typically, the switching time (rising voltage) is measured in hundreds of microseconds and idle time (falling voltage) is measured in seconds.

As current consumption goes below I_{MS} (see Figure 8-5) the converter goes from Active Regulated Mode to Active Low Current Mode. After this, the more the load current is decreased the longer the discharge time of the output capacitor, i.e. the time when the converter is not switching, will be. Similarly, the charge time, i.e. the time when the converter is switching, will be shorter. Note that in Active Low Current Mode only the last part of the rising/switching slope is changed.

Charge time can be minimised by forcing the device into Full Duty Cycle mode of operation (see "Full Duty Cycle" on page 41).

When load current increases above I_{MS} the converter goes back to Active Regulated Mode.

8.3 Output Voltage versus Load Current

The output voltage of the boost converter depends on the amount of load and method of duty cycle control.

In Active Mode the boost converter operates in one of two sub-modes called (Active) Regulated Mode and (Active) Low Current Mode. In Regulated Mode the emphasis is on maintaining a stable output voltage, while in Low Current Mode the emphasis is on minimising current consumption. The converter always enters Active Regulated Mode at first and switches to Low Current Mode automatically when required but it is possible to design the application such that the converter always stays in Active Regulated Mode.

The boost converter goes from Active Regulated Mode to Active Low Current Mode when the duty cycle reaches its minimum and the output voltage reaches its maximum. At this point the converter stops switching and the output voltage starts to fall. The converter starts switching again when the output voltage has fallen to the low limit defined for Low Current Mode. If load current increases sufficiently the converter will go back from Active Low Current Mode to Active Regulated Mode. See Figure 21-1 on page 165.

The boost converter goes back to Start Mode if output voltage drops below V_{BOOST}, and starts over from Stop Mode if input voltage drops below V_{STOP}, or when instructed by firmware to do so.

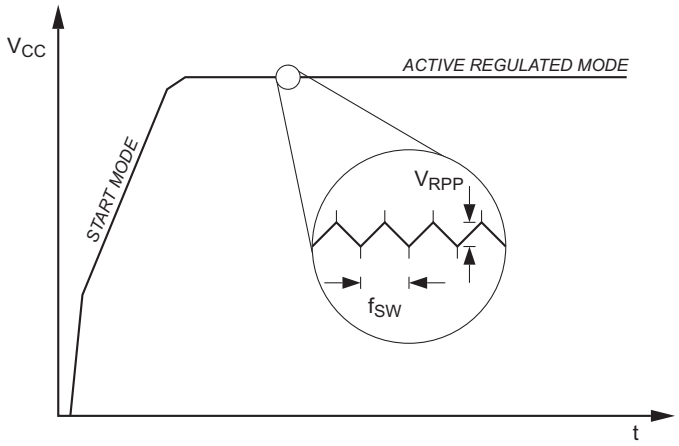
8.3.1 Active Regulated Mode

This is the default method of operation in Active Mode. The converter will remain in this mode provided that load current is sufficiently high. See "Active Low Current Mode" on page 41.

In this mode of operation the output voltage is constantly regulated. This means a stable output voltage with a low amplitude, high frequency ripple superimposed. See Figure 8-6 and Table 20-7 on page 159.

The firmware can instruct the converter to leave this mode and enter Stop Mode. See "Software Control of Boost Converter" on page 42.

Figure 8-6. Typical Output Voltage of Boost Converter in Active Regulated Mode.



8.3.2 Active Low Current Mode

The boost converter enters Active Low Current Mode from Active Regulated Mode when output voltage reaches its maximum and duty cycle is at its minimum. In practice, this means that the load current drops below a threshold. The threshold varies with converter input voltage and temperature but a typical plot is shown in Figure 21-1 on page 165.

From Figure 21-1 on page 165 can be seen that at low input voltages (V_{BAT} typically below 1.0V) and high load currents (I_{LOAD} typically above 0.6mA) the boost converter will never enter Low Current Mode. Using Full Duty Cycle mode the boost converter can be forced to enter Active Low Current Mode at input voltages lower than those shown in Figure 21-1 on page 165. See "Full Duty Cycle" on page 41.

In Low Current Mode the boost converter stops switching and reduces current consumption to a minimum, while still remaining active. Provided there are no external loads active the boost converter enters Low Current Mode automatically when the microcontroller goes into Power Down Mode (see "Sleep Modes" on page 31).

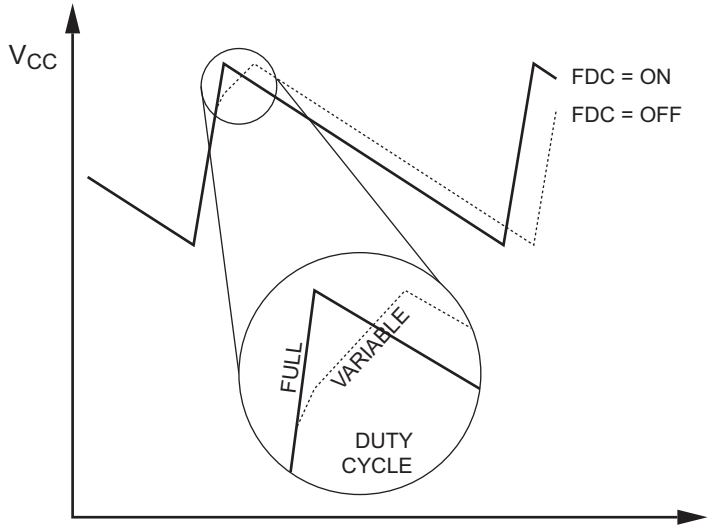
In this mode of operation the converter periodically reaches its duty cycle low limit. When this happens the converter stops switching and the output voltage starts dropping. The converter starts switching again when the output voltage has decreased to the low limit of Active Low Current Mode. This results in a periodical pattern as illustrated in Figure 8-5 on page 39.

If the output voltage, V_{CC}, drops below V_{BOOST} (due to an overload or a short circuit) the converter goes back to Start Mode. In addition, the firmware can instruct the converter to leave this mode and enter Stop Mode. See "Software Control of Boost Converter" on page 42.

8.3.3 Full Duty Cycle

By default, the boost converter keeps V_{CC} within limits by controlling the duty cycle of the switching waveform. It is possible to bypass the duty cycle regulation and lock the duty cycle at its maximum, resulting in a V_{CC} voltage that quickly ramps up to the maximum limit and then starts dropping when the boost converter enters Low Current Mode. See Figure 8-7, below.

Figure 8-7. Typical Output Voltage of Boost Converter at Constant Full Duty Cycle.



See section "Software Control of Boost Converter" on page 42 for instructions on how to turn on and off duty cycle control.

The use of Full Duty Cycle mode is recommended only at low load currents.

8.4 Overload Behaviour

The output is considered overloaded when the load current, I_{LOAD}, exceeds the maximum given in Table 20-7 on page 159. During an overload condition the boost converter operates at maximum duty cycle and can no longer regulate V_{CC}. If the overload condition prevails the output voltage will drop as load current increases. If V_{CC} drops below its minimum level the converter will switch to Start Mode.

In Start Mode the converter has a low load current capability, which means nearly all overload current will be drained straight from the battery (or other power source) via the inductor and the diode. The resistance of the inductor is typically very low and, provided the voltage of the power source remains constant, the output voltage during overload will stabilise to battery voltage, V_{BAT}, minus the forward voltage drop, V_F, of the diode used.

8.5 Software Control of Boost Converter

The boost converter is an independent hardware module that requires no interaction by the microcontroller, although some features can be controlled by firmware. Features that can be controlled by firmware are described in the following sections.

8.5.1 Stopping the Boost Converter

The device firmware can stop the boost converter on demand. When issued a stop signal, the boost converter will exit Active Mode and enter Stop Mode, as illustrated in Figure 8-3 on page 37. This procedure allows the device to read true battery voltage using the on-board ADC, assess if the voltage is sufficient for the selected battery chemistry and then control the boost converter accordingly.

To stop the boost converter, follow the below procedure:

1. Write 110x xxxx to the Power Reduction Register, PRR
2. Within 3 clock cycles of the above, write 10xx xxxx to PRR
3. Within 4 clock cycles of the first step, write 01xx xxxx to PRR

8.5.2 Switching to Full Duty Cycle Mode of Operation

When duty cycle control is disabled the output voltage of the boost converter will rise as fast as possible, resulting in a minimum switching time and a maximum idle time for the converter.

To turn on Full Duty Cycle (FDC) mode, follow the below procedure:

1. Write 110x xxxx to the Power Reduction Register, PRR
2. Within 3 clock cycles of the above, write 10xx xxxx to PRR
3. Within 4 clock cycles of the first step, write 111x xxxx to PRR

8.5.3 Switching to Normal (Variable Duty Cycle) Mode of Operation

To return duty cycle control to the boost converter, follow the below procedure:

1. Write 111x xxxx to the Power Reduction Register, PRR

8.6 Component Selection

Refer to Figure 8-2 on page 37 for component placement and numbering.

8.6.1 Inductor

Low inductance increases peak currents of the inductor, creating more interference noise and lowering the overall efficiency of the converter. Too high inductance values force the converter into non-stable operation. The boost converter has been optimized for a certain size inductance, L, and may not work reliably if other inductance values are used. See “Boost Converter Component Values” on page 45.

The inductor must be able to tolerate the following input current:

$$I_{IN} = \frac{V_{CC} \times I_{LOAD}}{V_{BAT} \times \eta}$$

... where η is the efficiency of the converter at given voltages and load current. See “Boost Converter Efficiency vs. Load Current and VBAT Voltage” on page 166.

The inductor must also be able to tolerate short peak currents. At steady state, i.e. when the converter has stabilised after a constant load current has been introduced, the peak current is calculated as follows:

$$I_{PEAK} = \frac{V_{BAT} \times T_S \times D}{L}$$

... where D is the duty cycle and T_S the switching period of the boost converter. See "Boost Converter Characteristics" on page 159 for limits. The steady-state duty cycle is calculated as follows:

$$D = \left(\frac{V_{CC}}{V_{BAT}} - \frac{1}{2}\right) \times \sqrt{I_{LOAD}}$$

Overall efficiency of the boost converter is also affected by the resistance of the inductor.

8.6.2 Diode

It is recommended to use a Schottky diode with forward voltage, V_F, and reverse leakage current, I_R, values as low as possible. This is because converter efficiency mainly depends on the forward voltage of the diode when I_{LOAD} is at maximum and V_{BAT} is at minimum. It should be noted that the reverse leakage current easily becomes a dominant factor, especially in Active Low Current Mode. For reference, see converter current consumption during Low Current Mode in Table 20-7 on page 159.

The diode is subject to peak currents the same magnitude as the inductor. See "Inductor" on page 43.

It should be noted that reverse leakage current is a highly temperature dependent variable.

8.6.3 Input Capacitors

A voltage drop occurs between the voltage source and inductor L_1 because voltage sources are not ideal and tracks have a non-zero resistance. The voltage drop is application specific and depends on the quality of the voltage source, inductor current, track size and track length. Capacitor C_1 counteracts the voltage drop by providing energy to the inductor during high peak currents.

The highest inductor peak currents are reached at highest load current and lowest V_{BAT} of the application. The input capacitor stabilises the input voltage and guarantees stable operation at all load currents. The size of the capacitor can be decreased if load currents remain low, or if a voltage supply with low internal resistance is available. Also, a good low-pass filter design (see section "RC Filter" below) reduces the size requirements of the capacitor.

8.6.4 RC Filter

A secondary input capacitor, C_2, and a series input resistor, R_1, are recommended. Together they form a Low-Pass Filter (LPF), the purpose of which is to reduce the voltage ripple at the V_{BAT} pin. The corner frequency of the filter can be calculated as follows:

$$f_{LPF} = \frac{1}{2 \times \pi \times R_1 \times C_2}$$

Component values are application specific and depend on the stability of the supply voltage. The LPF reduces voltage ripple at the V_{BAT} pin and helps to prevent the boost converter unintentionally entering Stop Mode.

Too high resistor values may lead to Start Mode failures. See “Boost Converter Component Values” on page 45 for component recommendations and limits.

Capacitor C_2 should be located close to the device.

8.6.5 Output Capacitors

An output capacitor, C_3, is required to keep the output voltage stable at times when energy is transferred to the inductor. It is recommended to use a capacitor with high capacitance and low Equivalent Series Resistance, ESR. A large capacitance helps to reduce the voltage ripple at the output and a low ESR reduces voltage ripple and helps to keep the temperature of the capacitor within limits.

The recommended capacitance at a given, steady-state load is calculated as follows:

$$C_{OUT} = \frac{I_{LOAD} \times T_S \times D}{V_{PP}}$$

... where T_S is the switching frequency of the boost converter, V_{PP} is the allowed voltage ripple and D is the duty cycle, calculated as shown in “Inductor” on page 43.

The recommended ESR is calculated as follows:

$$ESR \leq \frac{V_{PP}}{I_{PEAK}}$$

A secondary output capacitor, C_4, is recommended and should be placed close to the device.

8.6.6 Summary

The table below summarises recommended component values for a typical application.

Table 8-1. Boost Converter Component Values

Component	Recommended Value	Min (2)	Max (2)
C_1	C = 4.7µF	1µF	
C_2	C = 100nF (1)		
C_3	C = 22µF, ESR < 100mΩ	10µF	
C_4	C = 100nF	100nF	100nF
D_1	I_R = 1µA @ 25°C, V_F = 0.5V @ 1A		
L_1	L = 15µH ±20%, I_{MAX} = 700mA, R < 150mΩ	15µH	15µH
R_1	R = 680Ω (1)		1kΩ

Note:
1. With these values the LPF provides a 32dB attenuation at the switching frequency of the boost converter while permitting a supply voltage ripple of about ±200mV
2. Application specific limits may be tighter

8.7 Typical Applications

A typical use of the boost converter is illustrated in Figure 8-2 on page 37. Components can be optimized depending on the type of application. Table 8-2, below, presents recommendations for three different types of applications (cost effective, high output current and long battery life). All values are guidelines, only.

Table 8-2. Recommended Components and Values for Various Designs

Symbol	Component	Cost Effective	High Current	Battery Life	Unit
L_1	Inductor	15	15 (1)	15 (1)	µH
D_1	Schottky diode	10MQ100N	10BQ015 (2)	10BQ040 (3)	
R_1	Resistor	680	680	680	Ω
C_1	Input Capacitor	1 (4)	4.7	4.7	µF
C_2	Secondary Input Cap.	100	100	100	nF
C_3	Output Capacitor	10 (5)	22	22	µF
C_4	Secondary Output Cap.	–	22	22	nF

Notes:
1. Low ESR required.
2. High reverse leakage current, increases current consumption.
3. The diode is the largest individual contributor to battery life. The example diode keeps the boost converter running and maintains a reasonable efficiency level.
4. Depends on internal resistance of power supply.
5. Depends on load current. May not be sufficient for maximum current rating.

8.8 Characteristics

Electrical characteristics of the boost converter are given in Table 20-7 on page 159. Typical characteristics can be found under section "Boost Converter" on page 165.

8.9 Potential Limitations

When the device is powered via the boost converter some usage limitiations may apply. For example, the highest allowed operating frequency of the device depends on supply voltage (see "Speed Grades" on page 156) and the boost converter output voltage varies within the limits given in Table 20-7 on page 159. This means that if the design allows the boost converter to go into Active Low Current Mode the supply voltage will drop periodically, affecting the maximum allowed operating frequency.

Provided the load current remains sufficiently high the boost converter will never enter Active Low Current Mode and the supply voltage will remain high enough to run the device at higher frequencies. The boost converter status bit BS can be used to determine if the boost converter is in Low Current Mode (see "ADCSRB – ADC Control and Status Register B" on page 47).

Since the entire device is powered from the boost converter output variations will show in all peripherals. This means that, for example, high levels of I/O pins may vary with supply voltage.

8.10 Bypassing the Boost Converter

It is possible to bypass and disable the boost converter so that the device can be powered directly from an external supply. To force the boost converter into Stop Mode, connect pin V_{BAT} to ground and provide the device with supply directly to the V_{CC} pin. To permanently disable the

boost converter, connect pins V_{BAT} and LSW to ground and provide the device with supply directly to the V_{CC} pin.

8.11 Register Description

8.11.1 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	**BS**	**ACME**	**–**	**ADLAR**	**–**	**ADTS2**	**ADTS1**	**ADTS0**	**ADCSRB**
Read/Write	R	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 – BS: Boost Status**

The BS bit can be used to identify silent periods of the boost converter. When this bit is one the boost converter is active and switching, i.e. it is either in Active Regulated Mode, or in the switching period of Active Low Current Mode. When this bit is zero the boost converter is not switching, i.e. it is either in Stop Mode or in the non-switching period of Active Low Current Mode.

Alternatively, the BS bit can be programmed to return the state of the duty cycle controller, as follows:

1. Write 11xx xxxx to register PRR
2. Within 3 clock cycles of the above, write 10xx xxxx to register PRR
3. Wait (issue a single-cycle no operation)
4. Within 5 clock cycles of first write, read the BS bit

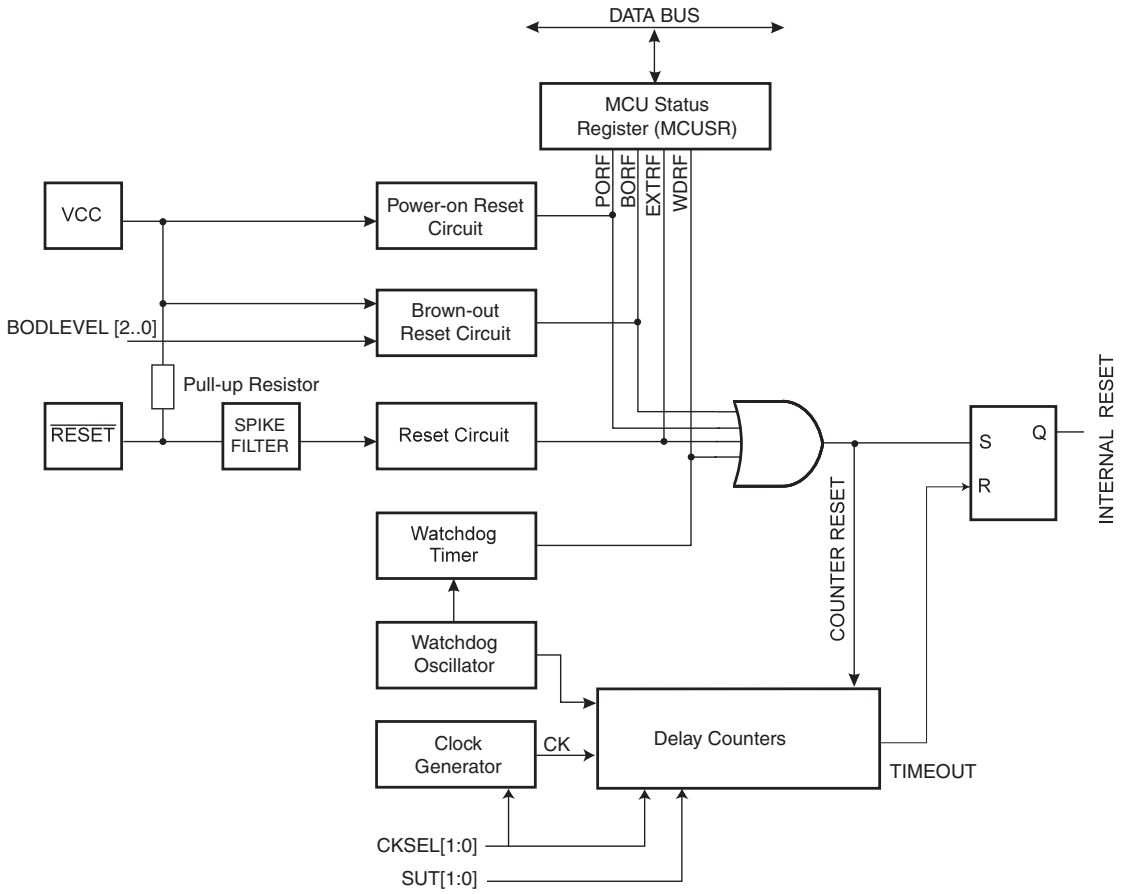
If the BS bit now is zero the converter is operating in normal duty cycle control mode. If the bit is one the converter is working in full duty cycle mode.

9. System Control and Reset

9.1 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 a 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 9-1 shows the reset logic. Table 20-4 on page 158 defines the electrical parameters of the reset circuitry.

Figure 9-1. Reset Logic



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

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. 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 "Clock Sources" on page 24.

9.2 Reset Sources

The ATtiny43U 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 when $\overline{\text{RESET}}$ function is enabled.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.

9.3 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 20-4 on page 158. 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 9-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

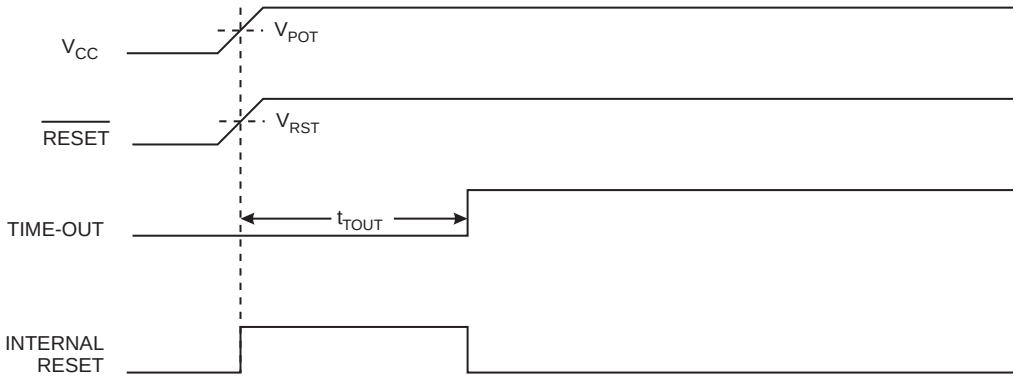
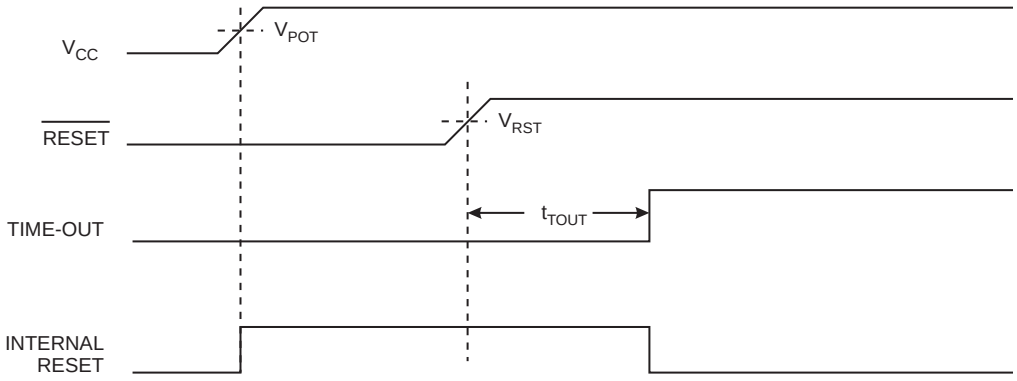


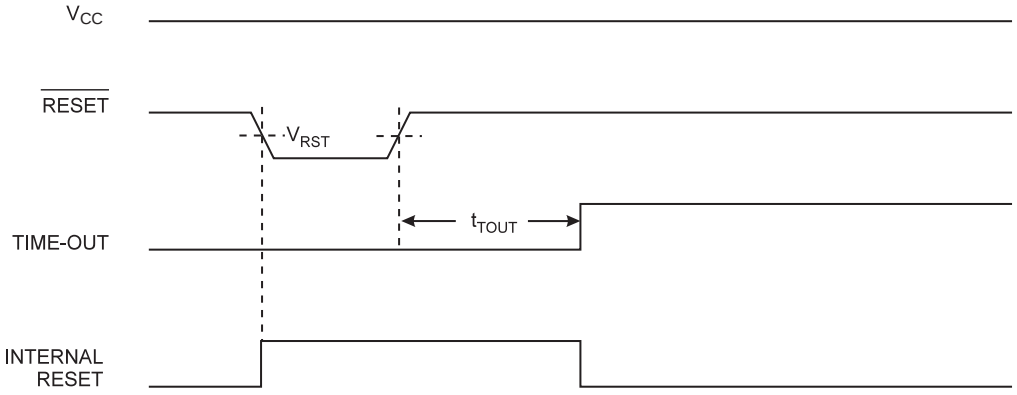
Figure 9-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



9.4 External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin if enabled. Reset pulses longer than the minimum pulse width (see Table 20-4 on page 158) 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.

Figure 9-4. External Reset During Operation



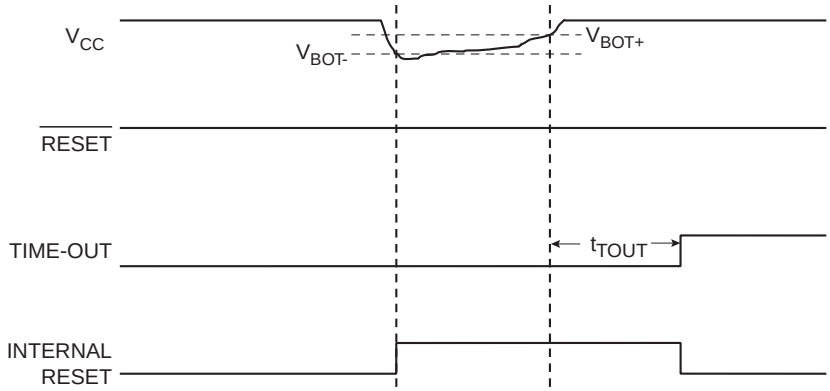
9.5 Brown-out Detection

ATtiny43U has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-5 on page 50), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 9-5 on page 50), 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 20-4 on page 158.

Figure 9-5. Brown-out Reset During Operation

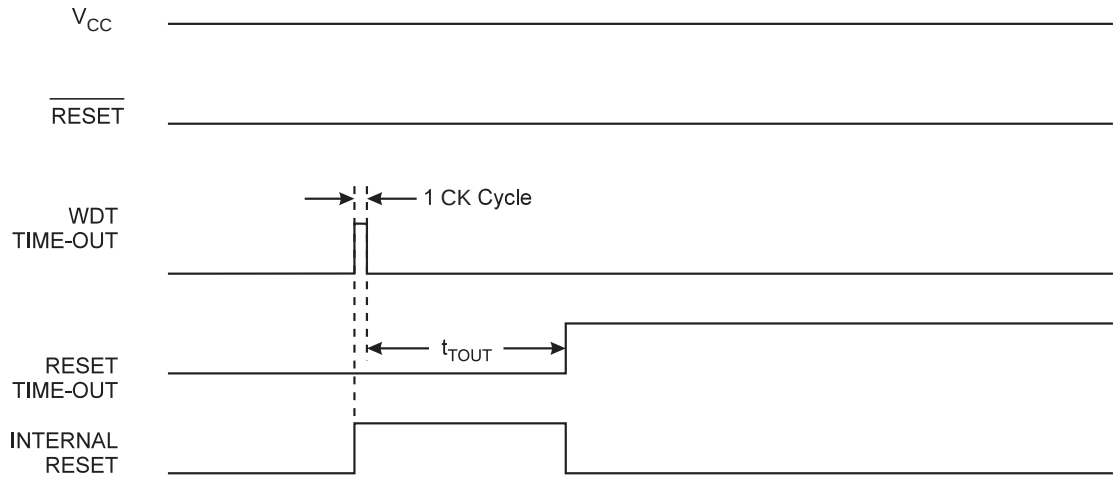


It is recommended to disable the BOD when using the integrated boost converter. See “Power Supply and On-Chip Boost Converter” on page 36

9.6 Watchdog 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}. See “Watchdog Timer” on page 51 for details on operation of the Watchdog Timer.

Figure 9-6. Watchdog Reset During Operation



9.7 Internal Voltage Reference

ATtiny43U 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.

9.7.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 20-4 on page 158. 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] Fuse).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

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

9.8 Watchdog Timer

The Watchdog Timer is clocked from an On-chip Oscillator which runs at 128 kHz. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 9-3 on page 56. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a Chip Reset occurs. Ten different

clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATtiny43U resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to Table 9-3 on page 56.

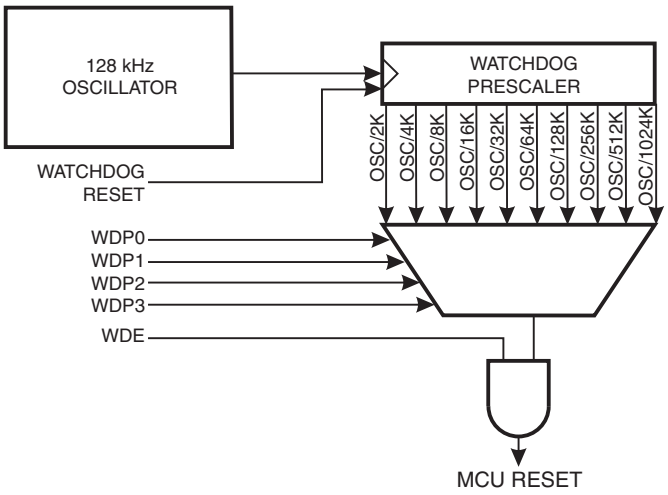
The Wathdog Timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the Watchdog to wake-up from Power-down.

To prevent unintentional disabling of the Watchdog or unintentional change of time-out period, two different safety levels are selected by the fuse WDTON as shown in Table 9-1. See "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 52 for details.

Table 9-1. WDT Configuration as a Function of the Fuse Settings of WDTON

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

Figure 9-7. Watchdog Timer



9.8.1 Timed Sequences for Changing the Configuration of the Watchdog Timer

The sequence for changing configuration differs slightly between the two safety levels. Separate procedures are described for each level.

9.8.1.1 Safety Level 1

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

1. In the same operation, write a logic one to 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, in the same operation, write the WDE and WDP bits as desired, but with the WDCE bit cleared.

9.8.1.2 Safety Level 2

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

1. In the same operation, write a logical one to WDCE and WDE. Even though the WDE always is set, the WDE must be written to one to start the timed sequence.
2. Within the next four clock cycles, in the same operation, write the WDP bits as desired, but with the WDCE bit cleared. The value written to the WDE bit is irrelevant.

9.8.2 Code Example

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

Assembly Code Example[1]

```
WDT_off:
  wdr
  ; Clear WDRF in MCUSR
  ldi  r16, (0<<WDRF)
  out  MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional Watchdog Reset
  in   r16, WDTCR
  ori  r16, (1<<WDCE)|(1<<WDE)
  out  WDTCR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCR, r16
  ret
```

C Code Example[1]

```
void WDT_off(void)
{
  _WDR();
  /* Clear WDRF in MCUSR */
  MCUSR = 0x00
  /* Write logical one to WDCE and WDE */
  WDTCR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCR = 0x00;
}
```

Note: 1. See “Code Examples” on page 6.

9.9 Register Description

9.9.1 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 and will always read zero.

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

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

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

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

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

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

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

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

To make use of the Reset Flags to identify a reset condition, the user should read and then reset the 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.

9.9.2 WDTCSR – Watchdog Timer Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x21 (0x41)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
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 Timeout 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 Timeout Interrupt Enable**

When this bit is written to one, WDE is cleared, and the I-bit in the Status Register is set, the Watchdog Time-out Interrupt is enabled. In this mode the corresponding interrupt is executed instead of a reset if a timeout in the Watchdog Timer occurs.

If WDE is set, WDIE is automatically cleared by hardware when a time-out occurs. This is useful for keeping the Watchdog Reset security while using the interrupt. After the WDIE bit is cleared,

the next time-out will generate a reset. To avoid the Watchdog Reset, WDIE must be set after each interrupt.

Table 9-2. Watchdog Timer Configuration

WDE	WDIE	Watchdog Timer State	Action on Time-out
0	0	Stopped	None
0	1	Running	Interrupt
1	0	Running	Reset
1	1	Running	Interrupt

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

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. See the description of the WDE bit for a Watchdog disable procedure. This bit must also be set when changing the prescaler bits. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 52.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDCE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

In safety level 2, it is not possible to disable the Watchdog Timer, even with the algorithm described above. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 52.

In safety level 1, WDE is overridden by WDRF in MCUSR. See “MCUSR – MCU Status Register” on page 54 for description of WDRF. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared before disabling the Watchdog with the procedure described above. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

Note: If the watchdog timer is not going to be used in the application, it is important to go through a watchdog disable procedure in the initialization of the device. If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset, which in turn will lead to a new watchdog reset. To avoid this situation, the application software should always clear the WDRF flag and the WDE control bit in the initialization routine.

- **Bits 5, 2:0 – WDP[3:0]: Watchdog Timer Prescaler 3, 2, 1, and 0**

The WDP[3:0] bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 9-3 on page 56.

Table 9-3. 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 cycles	16 ms
0	0	0	1	4K cycles	32 ms
0	0	1	0	8K cycles	64 ms
0	0	1	1	16K cycles	0.125 s
0	1	0	0	32K cycles	0.25 s
0	1	0	1	64K cycles	0.5 s
0	1	1	0	128K cycles	1.0 s
0	1	1	1	256K cycles	2.0 s
1	0	0	0	512K cycles	4.0 s
1	0	0	1	1024K 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		

10. Interrupts

This section describes the specifics of the interrupt handling as performed in ATtiny43U. For a general explanation of the AVR interrupt handling, see “Reset and Interrupt Handling” on page 13.

10.1 Interrupt Vectors

Table 10-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset
2	0x0001	INT0	External Interrupt Request 0
3	0x0002	PCINT0	Pin Change Interrupt Request 0
4	0x0003	PCINT1	Pin Change Interrupt Request 1
5	0x0004	WDT	Watchdog Time-out
6	0x0005	TIMER1_COMPA	Timer/Counter1 Compare Match A
7	0x0006	TIMER1_COMPB	Timer/Counter1 Compare Match B
8	0x0007	TIMER1_OVF	Timer/Counter1 Overflow
9	0x0008	TIMER0_COMPA	Timer/Counter0 Compare Match A
10	0x0009	TIMER0_COMPB	Timer/Counter0 Compare Match B
11	0x000A	TIMER0_OVF	Timer/Counter0 Overflow
12	0x000B	ANA_COMP	Analog Comparator
13	0x000C	ADC	ADC Conversion Complete
14	0x000D	EE_RDY	EEPROM Ready
15	0x000E	USI_START	USI Start
16	0x000F	USI_OVF	USI Overflow

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 most typical and general program setup for the Reset and Interrupt Vector Addresses in ATtiny43U is:

```
Address  Labels Code                 Comments
0x0000          rjmp   RESET         ; Reset Handler
0x0001          rjmp   INT0          ; IRQ0 Handler
0x0002          rjmp   PCINT0        ; PCINT0 Handler
0x0003          rjmp   PCINT1        ; PCINT1 Handler
0x0004          rjmp   WDT           ; Watchdog Interrupt Handler
0x0005          rjmp   TIM1_COMPA    ; Timer1 Compare A Handler
0x0006          rjmp   TIM1_COMPB    ; Timer1 Compare B Handler
0x0007          rjmp   TIM1_OVF      ; Timer1 Overflow Handler
```

```
0x0008          rjmp    TIM0_COMPA      ; Timer0 Compare A Handler
0x0009          rjmp    TIM0_COMPB      ; Timer0 Compare B Handler
0x000A          rjmp    TIM0_OVF        ; Timer0 Overflow Handler
0x000B          rjmp    ANA_COMP        ; Analog Comparator Handler
0x000C          rjmp    ADC             ; ADC Conversion Handler
0x000D          rjmp    EE_RDY          ; EEPROM Ready Handler
0x000E          rjmp    USI_STR         ; USI Start Handler
0x000F          rjmp    USI_OVF         ; USI Overflow Handler
;
0x0010   RESET: ldi     r16, low(RAMEND); Main program start
0x0011          out     SPL,r16         ; Set Stack Pointer to top of RAM
0x0012          sei                     ; Enable interrupts
0x0013     <instr>  xxx
   ...     ...     ...  ...
```

10.2 External Interrupts

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

- Pin Change Interrupt PCI0 triggers if a pin in PCINT7:0 is toggled while enabled
- Pin Change Interrupt PCI1 triggers if a pin in PCINT15:8 is toggled while enabled

The PCMSK0 and PCMSK1 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT15..0 are detected asynchronously. This means that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 interrupt can be triggered by a falling or rising edge, or a low level. This is configured as described in “MCUCR – MCU Control Register” on page 59. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Low level and edge interrupts on INT0 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.

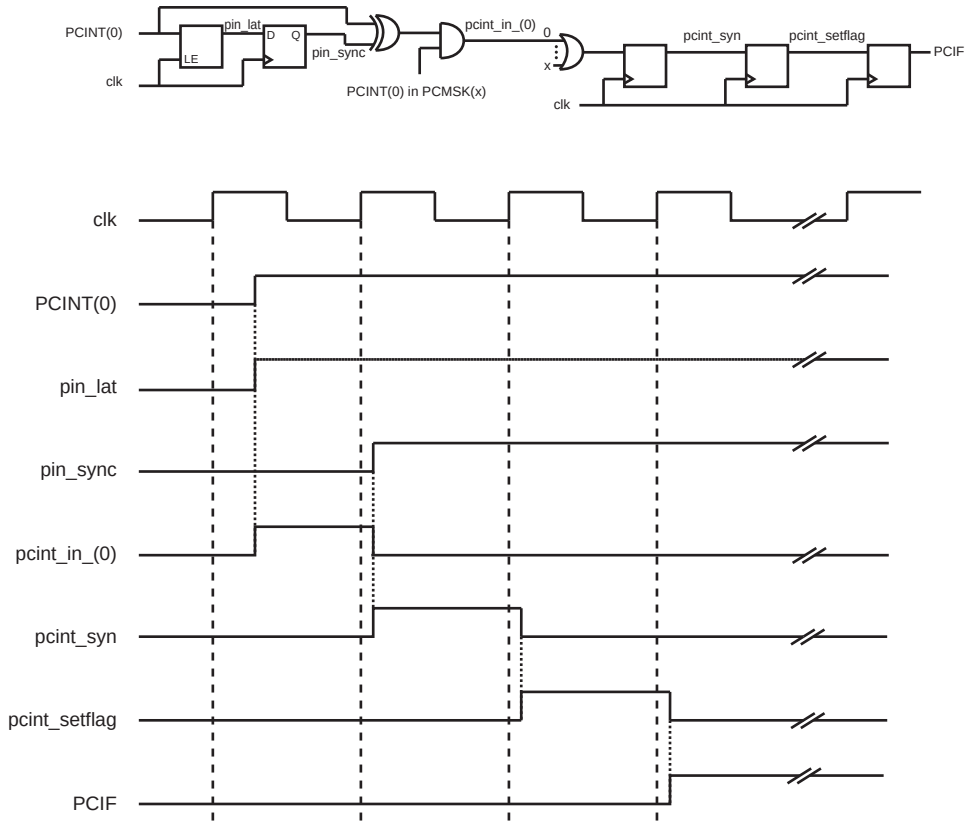
Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt.

If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated and execution will continue from the instruction following the SLEEP command. The start-up time is defined by the SUT and CKSEL fuses, as described in “System Clock and Clock Options” on page 23.

10.2.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 10-1 below.

Figure 10-1. Timing of pin change interrupts



10.3 Register Description

10.3.1 MCUCR – MCU Control Register

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

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	**BODS**	**PUD**	**SE**	**SM1**	**SM0**	**BODSE**	**ISC01**	**ISC00**	**MCUCR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

External Interrupt 0 is activated by the external pin INT0 if the I-flag of SREG and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 10-2 on page 60.

Edges on INT0 are registered asynchronously. Pulses on INT0 wider than the pulse width given in Table 20-6 on page 158 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt.

If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 10-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request asynchronously
0	1	Any logical change on INT0 generates an interrupt request asynchronously
1	0	The falling edge of INT0 generates an interrupt request asynchronously
1	1	The rising edge of INT0 generates an interrupt request asynchronously

10.3.2 GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	–	INT0	PCIE1	PCIE0	–	–	–	–	GIMSK
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 3:0 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU Control Register (MCUCR) 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.

- **Bit 5 – 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 PCINT15..8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT15..8 pins are enabled individually by the PCMSK1 Register.

- **Bit 4 – 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 PCINT7..0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 Register.

10.3.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	–	INTF0	PCIF1	PCIF0	–	–	–	–	GIFR
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 3:0 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 6 – 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 GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

- **Bit 5 – PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT15..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in GIMSK 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 4 – PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in GIMSK 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.

10.3.4 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	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 PCINT15:8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin, or not. If PCINT15:8 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT15:8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

10.3.5 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R	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[7:0]: Pin Change Enable Mask 7:0**

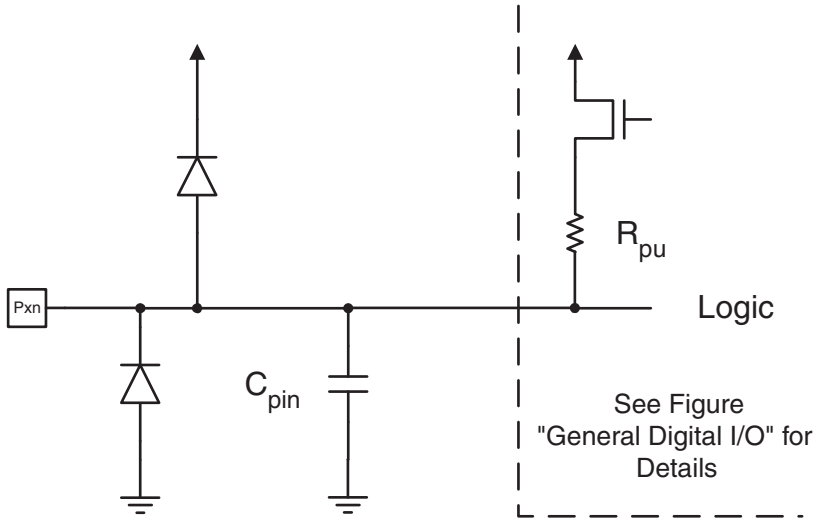
Each PCINT7:0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is set and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11. I/O Ports

11.1 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 11-1 on page 62. See "Electrical Characteristics" on page 155 for a complete list of parameters.

Figure 11-1. 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.

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 disables the pull-up function for all pins in all ports when set.

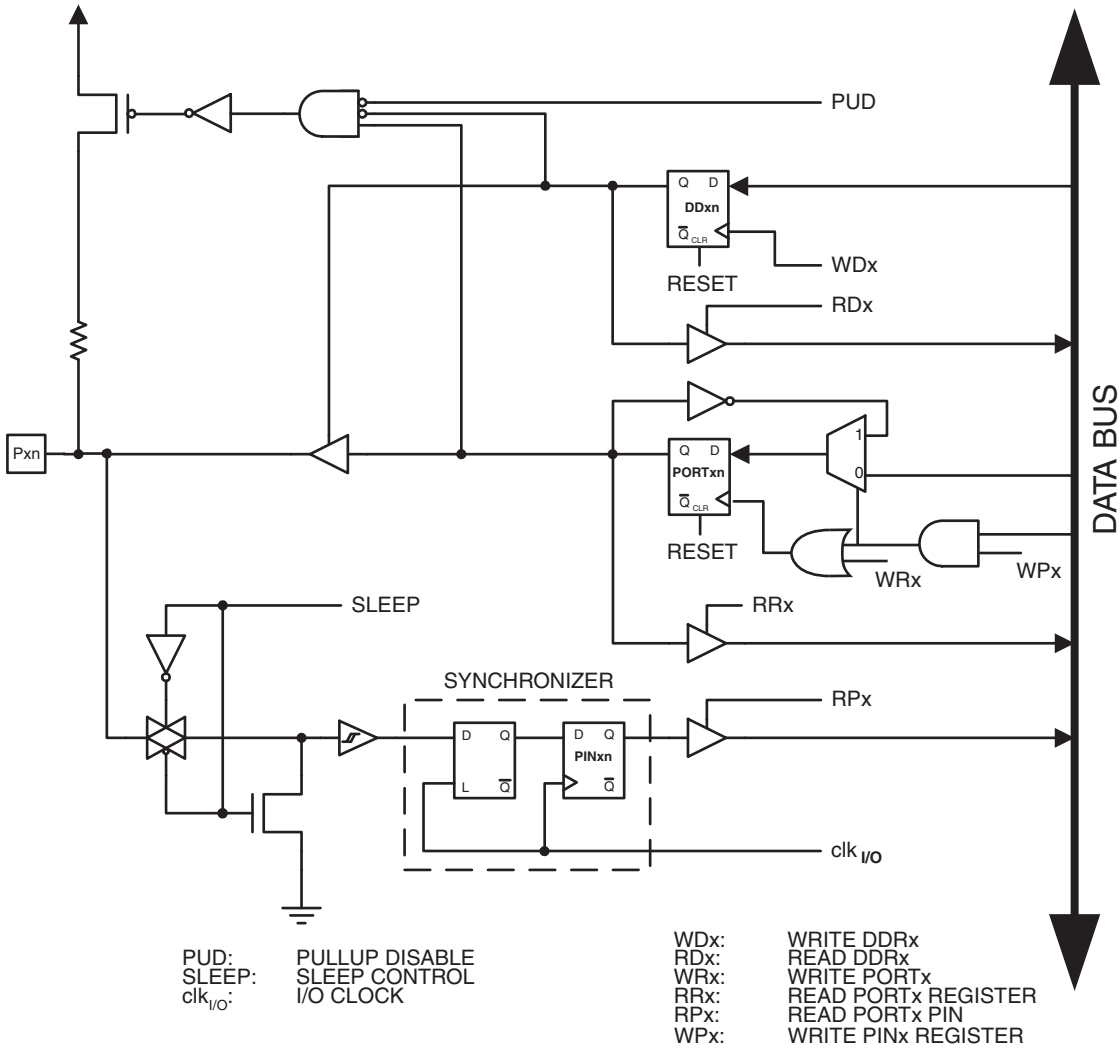
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 63. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 67. 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.

11.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 11-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 11-2. General Digital I/O[1]



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

11.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. 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).

11.2.2 Toggling the Pin

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

11.2.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-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 can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step.

Table 11-1 on page 64 summarizes the control signals for the pin value.

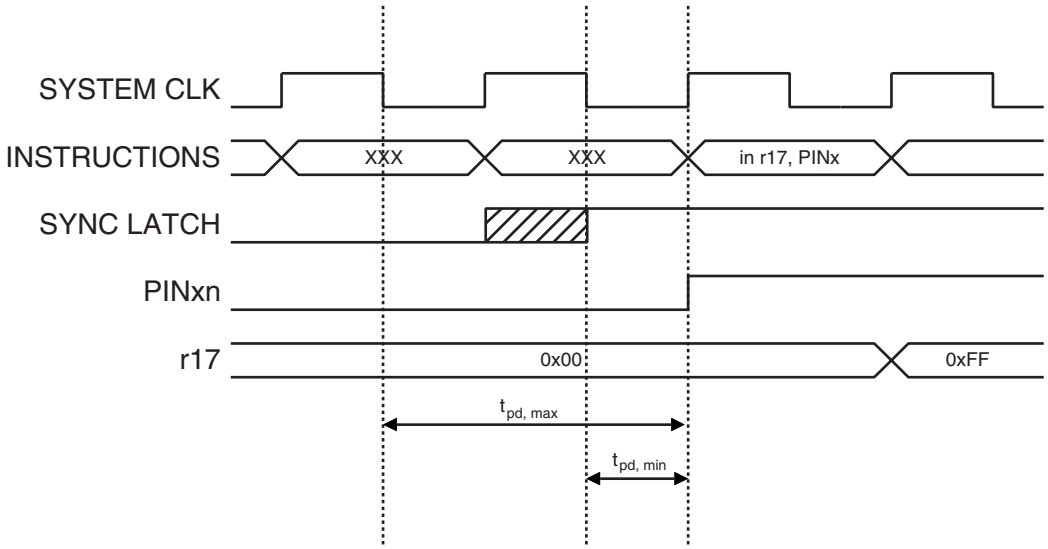
Table 11-1. Port Pin Configurations

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

11.2.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 11-2 on page 63, 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 11-3 on page 65 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.

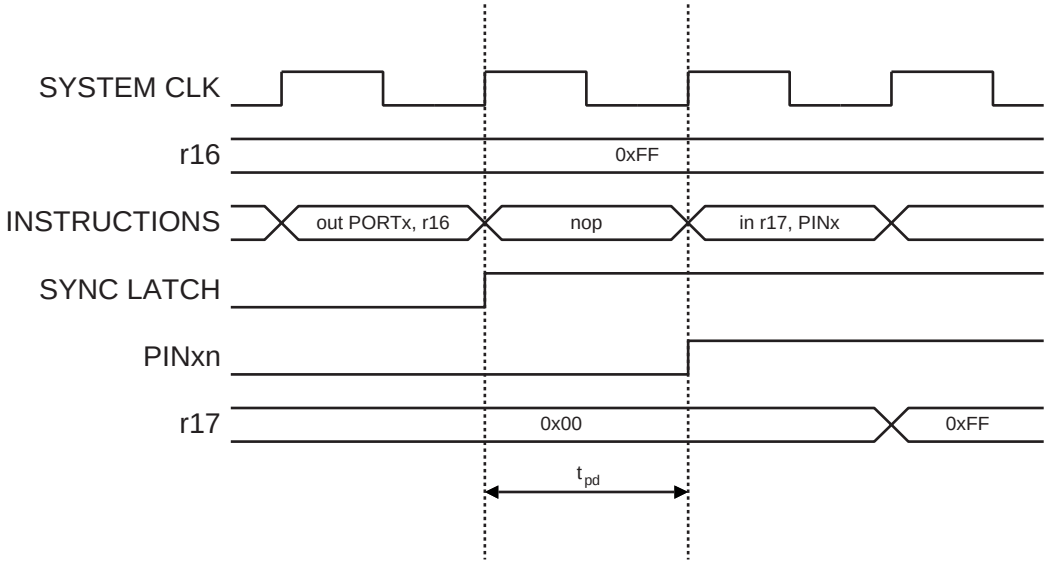
Figure 11-3. Synchronization when Reading an Externally Applied Pin value



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

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 11-4 on page 65. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is one system clock period.

Figure 11-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port A pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 5 as input with a pull-up assigned to port pin 4. 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<<PA4)|(1<<PA1)|(1<<PA0)
ldi  r17,(1<<DDA3)|(1<<DDA2)|(1<<DDA1)|(1<<DDA0)
out  PORTA,r16
out  DDRA,r17
; Insert nop for synchronization
nop
; Read port pins
in   r16,PINA
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTA = (1<<PA4)|(1<<PA1)|(1<<PA0);
DDRA = (1<<DDA3)|(1<<DDA2)|(1<<DDA1)|(1<<DDA0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINA;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1 and 4, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

11.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 11-2 on page 63, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode, Power-save mode, and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 67.

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.

11.2.6 Unconnected Pins

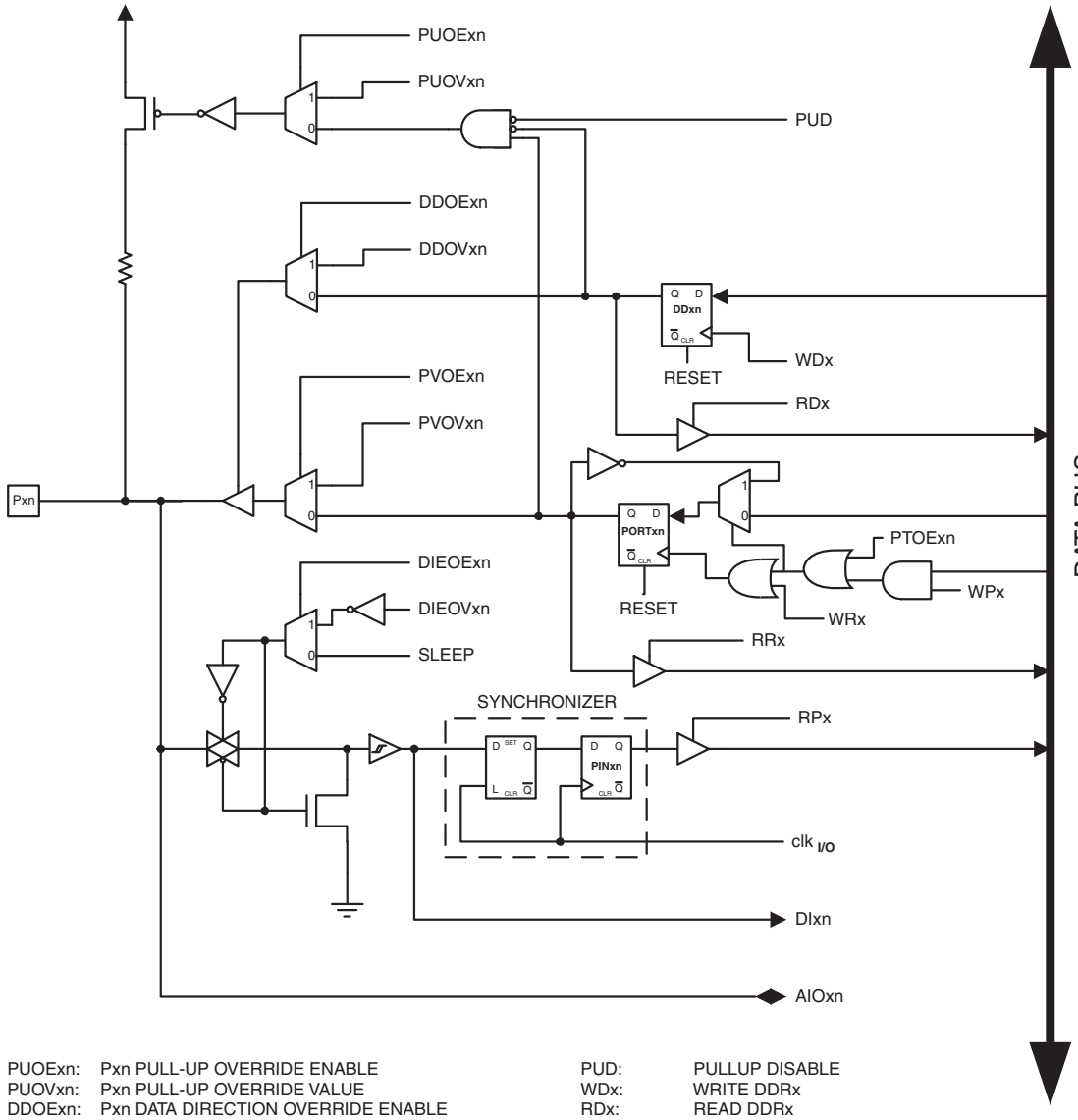
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled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

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pulldown. 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.

11.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 11-5 on page 68 shows how the port pin control signals from the simplified Figure 11-2 on page 63 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 11-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	$clk_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	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 11-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 11-5 on page 68 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 11-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
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.

11.3.1 Alternate Functions of Port A

The Port A pins with alternate function are shown in Table 11-3 on page 70.

Table 11-3. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA0	ADC0: ADC input channel 0. PCINT0: Pin change interrupt 0 source 0.
PA1	ADC1: ADC input channel 1. PCINT1: Pin change interrupt 0 source 1.
PA2	ADC2: ADC input channel 2. PCINT2: Pin change interrupt 0 source 2.
PA3	ADC3: ADC input channel 3. PCINT3: Pin change interrupt 0 source 3.
PA4	AIN0: Analog Comparator Positive Input. PCINT4: Pin change interrupt 0 source 4.
PA5	AIN1: Analog Comparator Negative Input. PCINT5: Pin change interrupt 0 source 5.
PA6	PCINT6: Pin change interrupt 0 source 6. CLKI: External Clock Input.
PA7	$\overline{\text{RESET}}$: Reset pin. dW: debugWire I/O. PCINT7: Pin change interrupt 0 source 7

- **Port A, Bit 0 – ADC0/PCINT0**

ADC0: Analog to Digital Converter, Channel 0.

PCINT0: Pin Change Interrupt source 0. The PA0 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 1 – ADC1/PCINT1**

ADC1: Analog to Digital Converter, Channel 1.

PCINT1: Pin Change Interrupt source 1. The PA1 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 2 – ADC2/PCINT2**

ADC2: Analog to Digital Converter, Channel 2.

PCINT2: Pin Change Interrupt source 2. The PA2 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 3 – ADC3/PCINT3**

ADC3: Analog to Digital Converter, Channel 3.

PCINT3: Pin Change Interrupt source 3. The PA3 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 4 – AIN0/PCINT4**

AIN0: Analog Comparator Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

PCINT4: Pin Change Interrupt source 4. The PA4 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 5 – AIN1/PCINT5**

AIN1: Analog Comparator Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

PCINT5: Pin Change Interrupt source 5. The PA5 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port A, Bit 6 – CLKI/PCINT6**

CLKI: External Clock Input. When used as a clock pin, the pin can not be used as an I/O pin.

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

- **Port A, Bit 7 – $\overline{\text{RESET}}$/dW/PCINT7**

$\overline{\text{RESET}}$: External $\overline{\text{Reset}}$ input is active low and enabled by unprogramming (“1”) the RSTDISBL Fuse. Pullup is activated and output driver and digital input are deactivated when the pin is used as the $\overline{\text{RESET}}$ pin.

dW: 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.

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

Table 11-4 on page 72 to Table 11-6 on page 73 relate the alternate functions of Port A to the overriding signals shown in Figure 11-5 on page 68.

Table 11-4. Overriding Signals for Alternate Functions in PA7..PA6

Signal Name	PA7/$\overline{\text{RESET}}$/dW/PCINT7	PA6/ PCINT6
PUOE	$\overline{\text{RSTDISBL}}$ [1]+ MONCOM_ENABLE	EXT_CLK [2]
PUOV	1	0
DDOE	$\overline{\text{RSTDISBL}}$[1] + MONCOM_ENABLE	EXT_CLK[2]
DDOV	MONCOM_ENABLE • debugWire Transmit	0
PVOE	$\overline{\text{RSTDISBL}}$[1] + MONCOM_ENABLE	EXT_CLK[2]
PVOV	0	0
PTOE	0	0
DIEOE	RSTDISBL[1] + MONCOM_ENABLE + PCINT7 • PCIE0	EXT_CLK[2] + (PCINT6 • PCIE0)
DIEOV	MONCOM_ENABLE + $\overline{\text{MONCOM_ENABLE}}$ • RSTDISBL[1] • PCINT7 • PCIE0)	(EXT_CLK[2] • $\overline{\text{PWR_DOWN}}$) + (EXT_CLK[2] • PCINT7 • PCIE0)
DI	dW/PCINT7 Input	CLKI/PCINT6 Input
AIO		

1. RSTDISBL is 1 when the Fuse is “0” (Programmed).
2. EXT_CLOCK = external clock is selected as system clock

Table 11-5. Overriding Signals for Alternate Functions in PA5..PA4

Signal Name	PA5/AIN1/ PCINT5	PA4/AIN0/PCINT4
PUOE	0	0
PUOV	0	0
DDOE	0	0
DDOV	0	0
PVOE	0	0
PVOV	0	0
PTOE	0	0
DIEOE	(PCINT5 • PCIE) + AIN1D	(PCINT4 • PCIE0) + AIN0D
DIEOV	PCINT5 • PCIE0	PCINT4 • PCIE0
DI	PCINT5 Input	PCINT4 input
AIO	Analog Comparator Negative Input	Analog Comparator Positive Input

Table 11-6. Overriding Signals for Alternate Functions in PA3..PA2

Signal Name	PA3/ADC3/PCINT3	PA2/ADC2/PCINT2
PUOE	0	
PUOV	0	0
DDOE	0	
DDOV	0	0
PVOE	0	
PVOV	0	0
PTOE	0	0
DIEOE	(PCINT3 • PCIE0) + ADC3D	(PCINT2 • PCIE) + ADC2D
DIEOV	PCINT3 • PCIE0	PCINT2 • PCIE0
DI	PCINT3 Input	PCINT2 Input
AIO	ADC3 Input	ADC2

Table 11-7. Overriding Signals for Alternate Functions in PA1..PA0

Signal Name	PA1/ADC1/PCINT1	PA0/ADC0/PCINT0
PUOE	0	
PUOV	0	0
DDOE	0	
DDOV	0	0
PVOE	0	
PVOV	0	0
PTOE	0	0
DIEOE	(PCINT1 • PCIE) + ADC1D	(PCINT0 • PCIE0) + ADC0D
DIEOV	PCINT1 • PCIE0	PCINT0 • PCIE0
DI	PCINT1 Input	PCINT0 Input
AIO	ADC1 Input	ADC0 Input

11.3.2 Alternate Functions of Port B

The Port B pins with alternate function are shown in Table 11-8 on page 74.

Table 11-8. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB0	T0: Timer/Counter0 Counter Source. PCINT8: Pin change interrupt 1 source 8.
PB1	OC0A: Timer/Counter0 Compare Match A output. PCINT9: Pin change interrupt 1 source 9.
PB2	OC0B: Timer/Counter0 Compare Match B output. PCINT10:Pin change interrupt 1 source 10.
PB3	T1: Timer/Counter1 Counter Source. CLKO: System Clock Output. PCINT11:Pin change interrupt 1 source 11.
PB4	OC1A: Timer/Counter1 Compare Match A output. DI: USI Data Input three wire mode. SDA: USI Data Input two wire mode. PCINT12:Pin change interrupt 1 source 12.
PB5	OC1B: Timer/Counter1 Compare Match B output. DO: USI Data Output three wire mode. PCINT13:Pin change interrupt 1 source 13.
PB6	USCK: USI Clock three wire mode. SCL: USI Clock two wire mode. PCINT14:Pin change interrupt 1 source 14.
PB7	INT0: External Interrupt 0 input. PCINT15:Pin change interrupt 1 source 15.

- **Port B, Bit 0 – T0/PCINT8**

T0: Timer/Counter0 Counter Source.

PCINT8: Pin Change Interrupt source 8. The PB0 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 1 – OC0A/PCINT9**

OC0A: Output Compare Match output: The PB1 pin can serve as an external output for the Timer/Counter0 Compare Match A. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.

PCINT9: Pin Change Interrupt source 9. The PB1 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 2 – OC0B/PCINT10**

OC0B: Output Compare Match output: The PB2 pin can serve as an external output for the Timer/Counter0 Compare Match A. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.

PCINT10: Pin Change Interrupt source 10. The PB2 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 3 – T1/CLKO/PCINT11**

T1: Timer/Counter1 Counter source.

CLKO: System Clock Output. The system clock can be output on the PB3 pin. The system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTB3 and DDB3 settings. It will also be output during reset.

PCINT11: Pin Change Interrupt source 11. The PB3 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 4 – DI/SDA/OC1A/PCINT12**

DI: Data Input in USI Three-wire mode. 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 Data.

OC1A, Output Compare Match output: The PB4 pin can serve as an external output for the Timer/Counter1 Compare Match A. The PB4 pin has to be configured as an output (DDB4 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

PCINT12: Pin Change Interrupt source 12. The PB4 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 5 – DO/OC1B/PCINT13**

DO: Data Output in USI Three-wire mode. Data output (DO) overrides PORTB5 value and it is driven to the port when the data direction bit DDB5 is set (one). However the PORTB5 bit still controls the pullup, enabling pullup if direction is input and PORTB5 is set (one).

OC1B: Output Compare Match output: The PB5 pin can serve as an external output for the Timer/Counter1 Compare Match B. The PB5 pin has to be configured as an output (DDB5 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT13: Pin Change Interrupt source 13. The PB5 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 6 – USCK/SCL/PCINT14**

USCK: Three-wire mode Universal Serial Interface Clock.

SCL: Two-wire mode Serial Clock for USI Two-wire mode.

PCINT14: Pin Change Interrupt source 14. The PB6 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port B, Bit 7 – INT0/PCINT15**

INT0: External Interrupt Request 0.

PCINT15: Pin Change Interrupt source 15. The PB7 pin can serve as an external interrupt source for pin change interrupt 1.

Table 11-9 on page 76 to Table 11-12 on page 77 relate the alternate functions of Port B to the overriding signals shown in Figure 11-5 on page 68.

Table 11-9. Overriding Signals for Alternate Functions in PB7..PB6

Signal Name	PB7/INT0/PCINT15	PB6/PCINT14
PUOE	0	0
PUOV	0	0
DDOE	0	USIWM1
DDOV	0	USI_SCL_HOLD + $\overline{\text{PORTB6}}$
PVOE	0	USIWM1
PVOV	0	0
PTOE	0	USIPTOE
DIEOE	(PCINT15 • PCIE1) + INT0	(PCINT14 • PCIE1) + USISIE
DIEOV	(PCINT15 • PCIE1) + INT0	(PCINT14 • PCIE1) + USISIE
DI	INT0/PCINT15	PCINT14/USCK/SCL
AIO		

Table 11-10. Overriding Signals for Alternate Functions in PB5..PB4

Signal Name	PB5/OC1B/PCINT13	PB4/OC1A/PCINT10
PUOE	0	0
PUOV	0	0
DDOE	0	USIWM1
DDOV	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB4}}$) • DDRB4
PVOE	OC1B Enable + ($\overline{\text{USIWM1}}$ • USIWM0)	OC1A Enable + (USIWM1 • DDRB4)
PVOV	OC1B • (~USIWM1• $\overline{\text{USIWM0}}$) + $\overline{\text{USIWM1}}$ • USIWM0 • DO	OC1A • ($\overline{\text{USIWM1• DDRB4}}$)
PTOE	0	0
DIEOE	PCINT13 • PCIE1	PCINT10 • PCIE1 + USISIE
DIEOV	PCINT13 • PCIE1	PCINT10 • PCIE1 + USISIE
DI	PCINT13	PCINT10/DI/SDA
AIO		

Table 11-11. Overriding Signals for Alternate Functions in PB3..PB2

Signal Name	PB3/T1/CLKO/PCINT9	PB2/OC0B/PCINT8
PUOE	CKOUT	0
PUOV	0	0
DDOE	CKOUT	0
DDOV	1'b1	0
PVOE	CKOUT	OC0B Enable
PVOV	CKOUT • System Clock	OC0B
PTOE	0	0
DIEOE	PCINT9 • PCIE1	PCINT8 • PCIE1
DIEOV	PCINT9 • PCIE1	PCINT8 • PCIE1
DI	T1/PCINT9 Input	PCINT8 Input
AIO		

Table 11-12. Overriding Signals for Alternate Functions in PB1..PB0

Signal Name	PB1/OC0A/PCINT7	PB0/T0/PCINT6
PUOE	0	0
PUOV	0	0
DDOE	0	0
DDOV	0	0
PVOE	0	0
PVOV	OC0A Enable	0
PTOE	OC0A	0
DIEOE	PCINT7 • PCIE1	PCINT6 • PCIE1
DIEOV	PCINT7 • PCIE1	PCINT6 • PCIE1
DI	PCINT7 Input	PCINT6 Input
AIO		

11.4 Register Description

11.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See “Configuring the Pin” on page 63 for more details about this feature.

11.4.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	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	

11.4.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	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	

11.4.4 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x19 (0x39)	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	

11.4.5 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	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	

11.4.6 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	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	

11.4.7 PINB – Port BInput Pins Address

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	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	

12. 8-bit Timer/Counter with PWM (Timer/Counter0 and Timer/Counter1)

12.1 Features

- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch Free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **Six Independent Interrupt Sources (TOV0, OCF0A, OCF0B, TOV1, OCF1A, and OCF1B)**

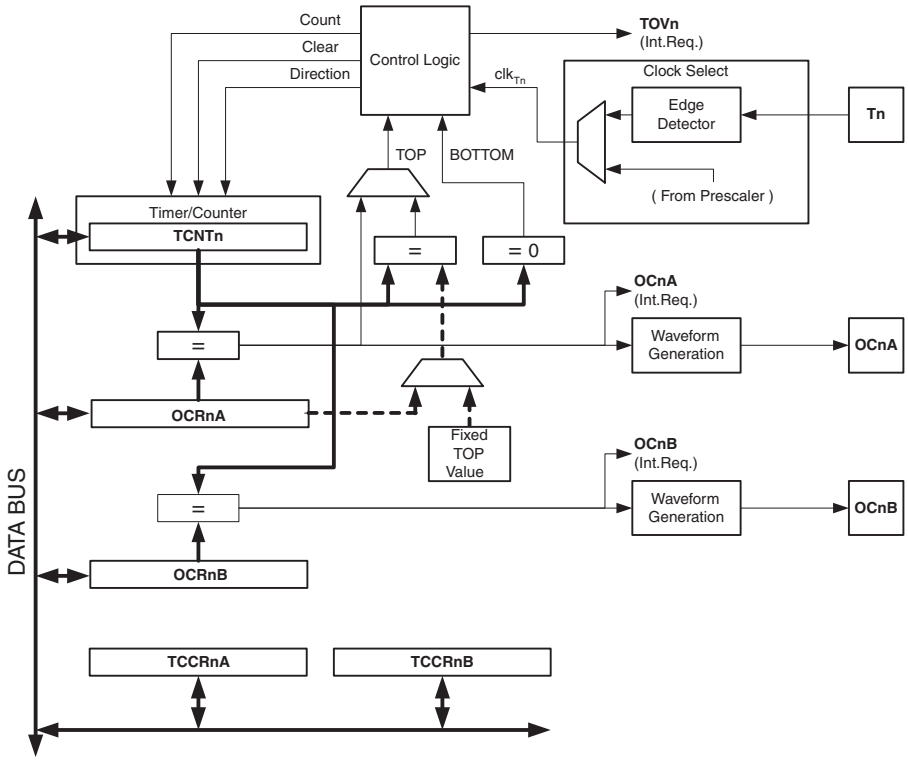
12.2 Overview

Timer/Counter0 and Timer/Conter1 are general purpose Timer/Counter modules with two independent Output Compare Units, each, and with PWM support. They allow accurate program execution timing (event management) and wave generation.

Register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. 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 simplified block diagram of the 8-bit Timer/Counter is shown in Figure 12-1 on page 79. For the actual placement of I/O pins, refer to Figure 1-1 on page 2. 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 90.

Figure 12-1. 8-bit Timer/Counter Block Diagram



12.2.1 Registers

The Timer/Counter (TCNTn) and Output Compare Registers (OCRnA and OCRnB) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFRn). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSKn). TIFRn and TIMSKn are not shown in the figure.

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

The double buffered Output Compare Registers (OCRnA and OCRnB) 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 pins (OCnA and OCnB). See “Output Compare Unit” on page 81 for details. The Compare Match event will also set the Compare Flag (OCFnA or OCFnB) which can be used to generate an Output Compare interrupt request.

12.2.2 Definitions

The definitions in Table 12-1 are used extensively throughout the document.

Table 12-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 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 OCRnA Register. The assignment is dependent on the mode of operation.

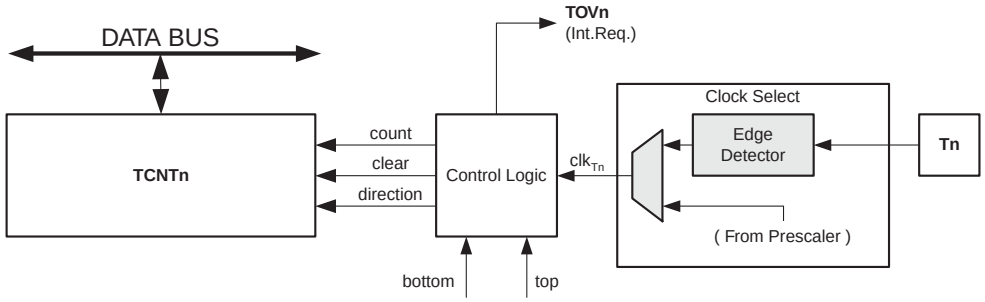
12.3 Timer/Counter Clock Sources

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 (CSn2:0) bits located in the Timer/Counter Control Register (TCCRnB). For details on clock sources and prescaler, see “Timer/Counter Prescaler” on page 98.

12.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 12-2 on page 81 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 TCNTn by 1.
direction	Select between increment and decrement.
clear	Clear TCNTn (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{Tn} in the following.
top	Signalize that TCNTn has reached maximum value.
bottom	Signalize that TCNTn has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{Tn}). clk_{Tn} can be generated from an external or internal clock source, selected by the Clock Select bits (CSn2:0). When no clock source is selected (CSn2:0 = 0) the timer is stopped. However, the TCNTn value can be accessed by the CPU, regardless of whether clk_{Tn} 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 WGMn1 and WGMn0 bits located in the Timer/Counter Control Register (TCCRnA) and the WGMn2 bit located in the Timer/Counter Control Register B (TCCRnB). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OCnA. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 84.

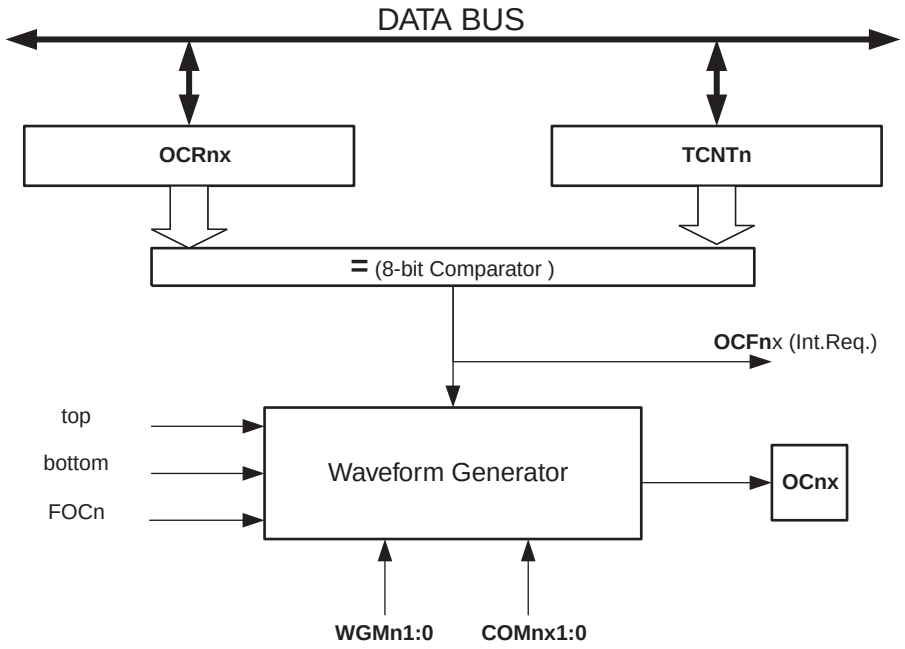
The Timer/Counter Overflow Flag (TOVn) is set according to the mode of operation selected by the WGMn1:0 bits. TOVn can be used for generating a CPU interrupt.

12.5 Output Compare Unit

The 8-bit comparator continuously compares TCNTn with the Output Compare Registers (OCRnA and OCRnB). Whenever TCNTn equals OCRnA or OCRnB, the comparator signals a match. A match will set the Output Compare Flag (OCFnA or OCFnB) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a 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 WGMn2:0 bits and Compare Output mode (COMnx1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation. See “Modes of Operation” on page 84.

Figure 12-3 on page 82 shows a block diagram of the Output Compare unit.

Figure 12-3. Output Compare Unit, Block Diagram



The OCRnx Registers are 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 OCRnx Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

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

12.5.1 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 (nx) bit. Forcing Compare Match will not set the OCFnx Flag or reload/clear the timer, but the OCnx pin will be updated as if a real Compare Match had occurred (the COMnx1:0 bits settings define whether the OCnx pin is set, cleared or toggled).

12.5.2 Compare Match Blocking by TCNTn Write

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

12.5.3 Using the Output Compare Unit

Since writing TCNTn in any mode of operation will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNTn when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNTn equals the OCRnx value, the Compare Match will be missed, resulting in incorrect waveform

generation. Similarly, do not write the TCNTn value equal to BOTTOM when the counter is down-counting.

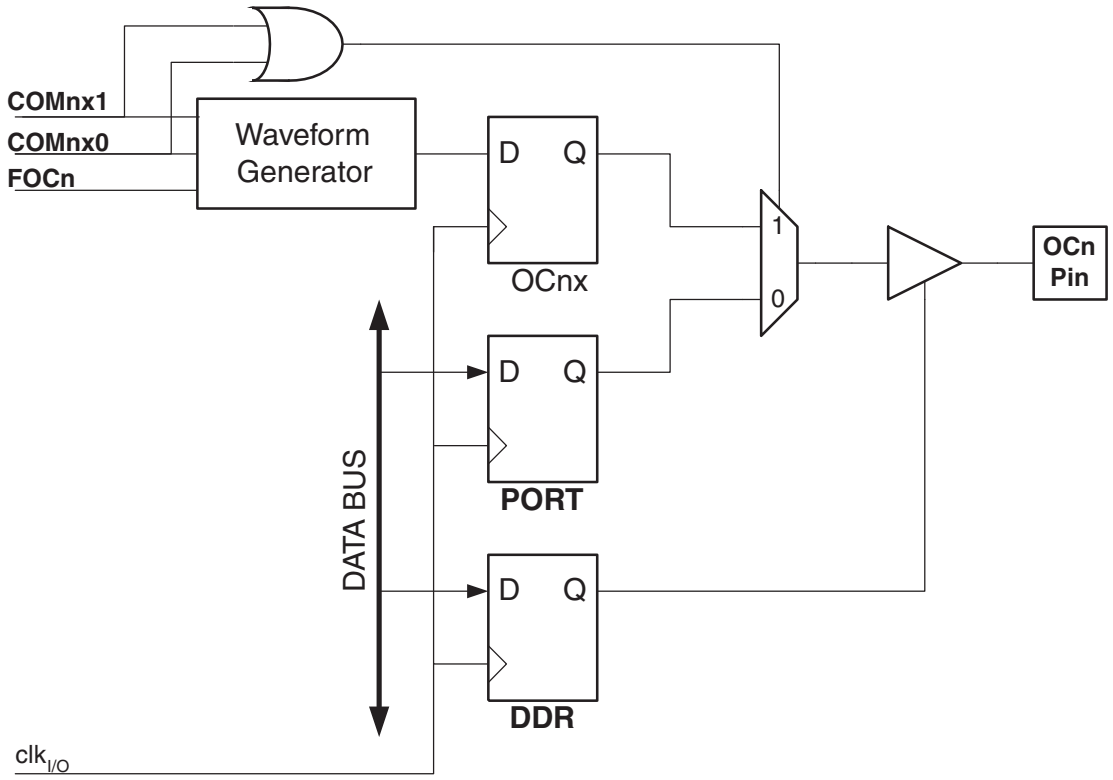
The setup of the OCnx should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OCnx value is to use the Force Output Compare (nx) strobe bits in Normal mode. The OCnx Registers keep their values even when changing between Waveform Generation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

Be aware that the COMnx1:0 bits are not double buffered together with the compare value. Changing the COMnx1:0 bits will take effect immediately.

12.6 Compare Match Output Unit

The Compare Output mode (COMnx1:0) bits have two functions. The Waveform Generator uses the COMnx1:0 bits for defining the Output Compare (OCnx) state at the next Compare Match. Also, the COMnx1:0 bits control the OCnx pin output source. Figure 12-4 on page 83 shows a simplified schematic of the logic affected by the COMnx1:0 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 COMnx1:0 bits are shown. When referring to the OCnx state, the reference is for the internal OCnx Register, not the OCnx pin. If a system reset occur, the OCnx Register is reset to “0”.

Figure 12-4. Compare Match Output Unit, Schematic (non-PWM Mode)



The general I/O port function is overridden by the Output Compare (OCnx) from the Waveform Generator if either of the COMnx1:0 bits are set. However, the OCnx pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OCnx pin (DDR_OCnx) must be set as output before the OCnx value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OCnx state before the output is enabled. Note that some COMnx1:0 bit settings are reserved for certain modes of operation, see “Register Description” on page 90

12.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COMnx1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COMnx1:0 = 0 tells the Waveform Generator that no action on the OCnx Register is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Table 12-2 on page 90. For fast PWM mode, refer to Table 12-3 on page 91, and for phase correct PWM refer to Table 12-4 on page 91.

A change of the COMnx1:0 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 FOCnx strobe bits.

12.7 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 (WGMn2:0) and Compare Output mode (COMnx1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COMnx1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COMnx1:0 bits control whether the output should be set, cleared, or toggled at a Compare Match (See “Modes of Operation” on page 84).

For detailed timing information refer to Figure 12-8 on page 89, Figure 12-9 on page 89, Figure 12-10 on page 89 and Figure 12-11 on page 90 in “Timer/Counter Timing Diagrams” on page 88.

12.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGMn2:0 = 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 (TOVn) will be set in the same timer clock cycle as the TCNTn becomes zero. The TOVn 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 TOVn 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.

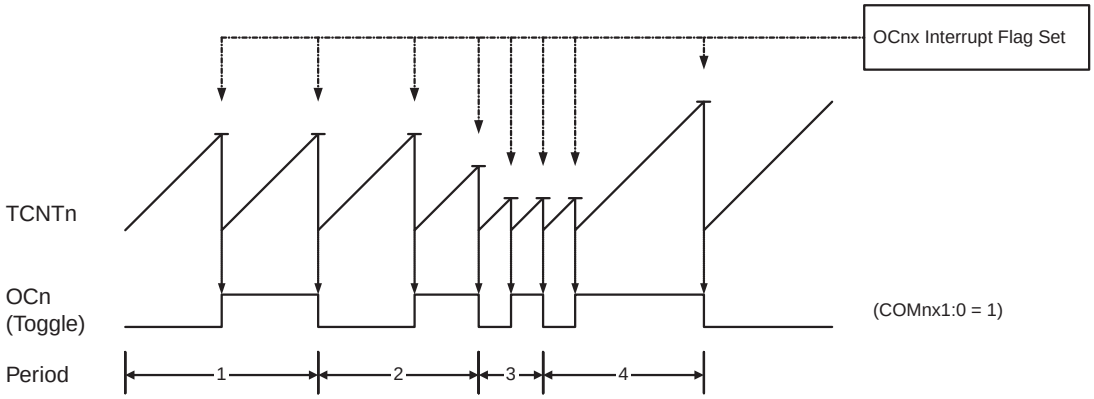
The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

12.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGMn2:0 = 2), the OCRnA Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches the OCRnA. The OCRnA 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 12-5 on page 85. The counter value (TCNTn) increases until a Compare Match occurs between TCNTn and OCRnA, and then counter (TCNTn) is cleared.

Figure 12-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCFnA Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCRnA is lower than the current value of TCNTn, 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 OCnA output can be set to toggle its logical level on each Compare Match by setting the Compare Output mode bits to toggle mode (COMnA1:0 = 1). The OCnA 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 $f_{OCnx} = f_{clk_I/O}/2$ when OCRnA 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, 64, 256, or 1024).

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

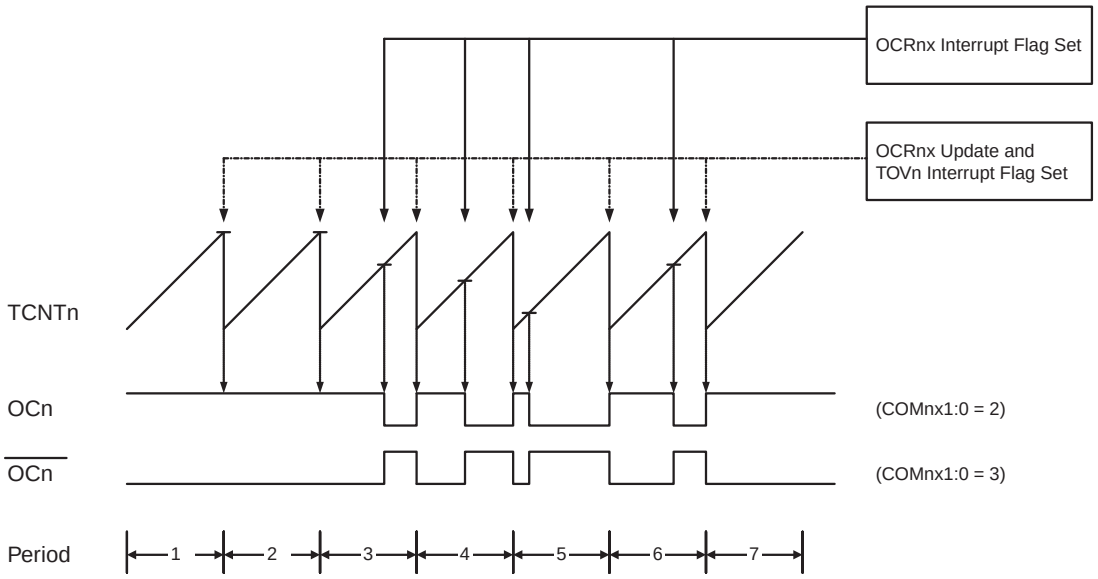
12.7.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGMn2:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGMn2:0 = 3, and OCRnA when WGMn2:0 = 7. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the Compare Match between TCNTn and OCRnx, and set at BOTTOM. In inverting Compare Output mode, 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 use dual-slope operation. This high frequency makes the fast PWM mode well suited

for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 12-6 on page 86. The TCNTn value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNTn slopes represent Compare Matches between OCRnx and TCNTn.

Figure 12-6. Fast PWM Mode, Timing Diagram



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

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three: Setting the COMnA1:0 bits to one allowes the OCnA pin to toggle on Compare Matches if the WGMn2 bit is set. This option is not available for the OCnB pin (See Table 12-3 on page 91). The actual OCnx 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 OCnx Register at the Compare Match between OCRnx and TCNTn, and clearing (or setting) the OCnx Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

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

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

The extreme values for the OCRnA Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCRnA is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCRnA equal to MAX will result

in a constantly high or low output (depending on the polarity of the output set by the COMnA1:0 bits.)

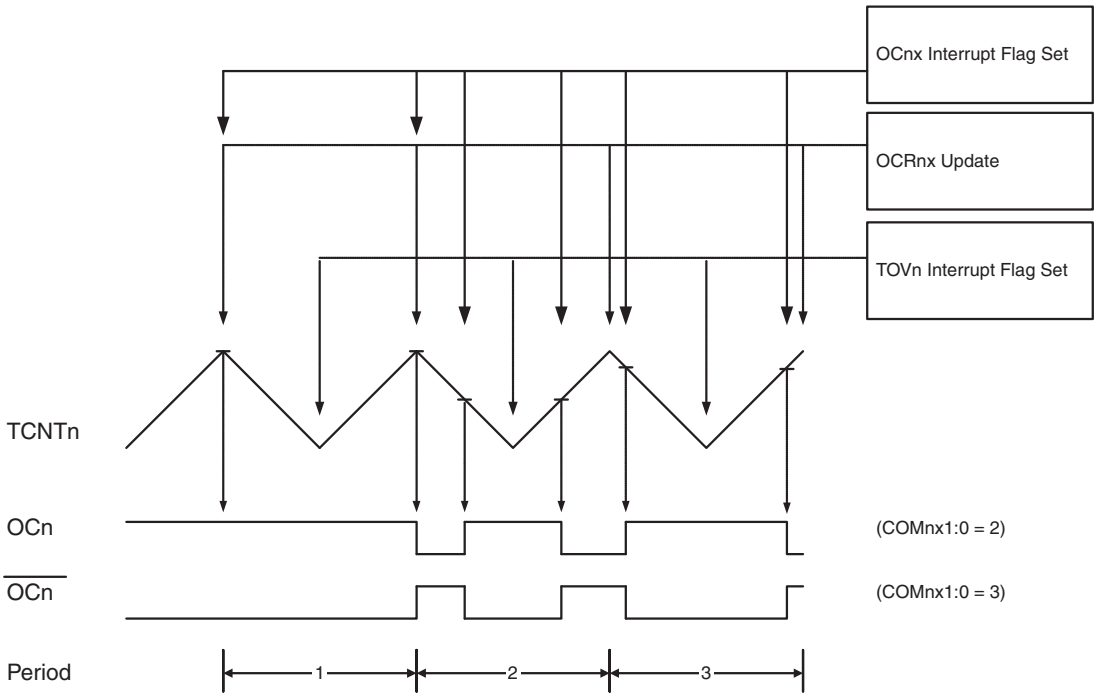
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OCnx to toggle its logical level on each Compare Match (COMnx1:0 = 1). The waveform generated will have a maximum frequency of $f_{OCnx} = f_{clk_I/O}/2$ when OCRnA is set to zero. This feature is similar to the OCnA toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

12.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGMn2:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGMn2:0 = 1, and OCRnA when WGMn2:0 = 5. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the Compare Match between TCNTn and OCRnx while upcounting, and set on the Compare Match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNTn value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 12-7 on page 87. The TCNTn value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNTn slopes represent Compare Matches between OCRnx and TCNTn.

Figure 12-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOVn) 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 OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COMnx1:0 to three: Setting the COMnA0 bits to one allows the OCnA pin to toggle on Compare Matches if the WGMn2 bit is set. This option is not available for the OCnB pin (See Table 12-4 on page 91). The actual OCnx 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 OCnx Register at the Compare Match between OCRnx and TCNTn when the counter increments, and setting (or clearing) the OCnx Register at Compare Match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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

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

At the very start of period 2 in Figure 12-7 on page 87 OCn has a transition from high to low even though there is no Compare Match. The point of this transition is to guaratee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCRnA changes its value from MAX, like in Figure 12-7 on page 87. When the OCRnA value is MAX the OCn pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCRnA, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

12.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 12-8 on page 89 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 12-8. Timer/Counter Timing Diagram, no Prescaling

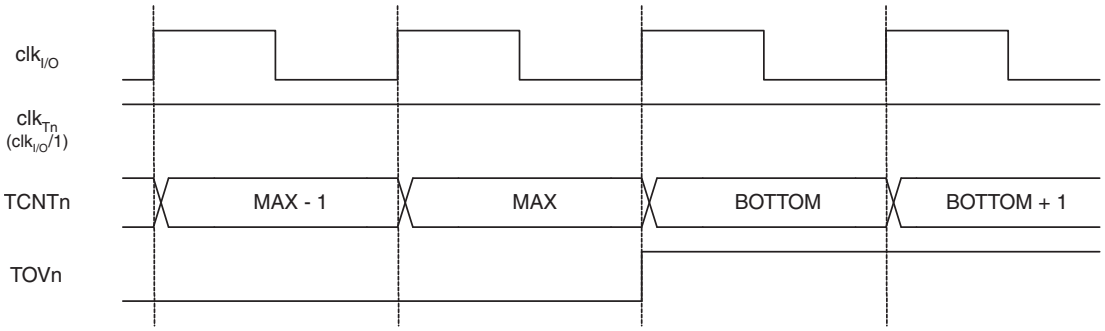


Figure 12-9 on page 89 shows the same timing data, but with the prescaler enabled.

Figure 12-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)

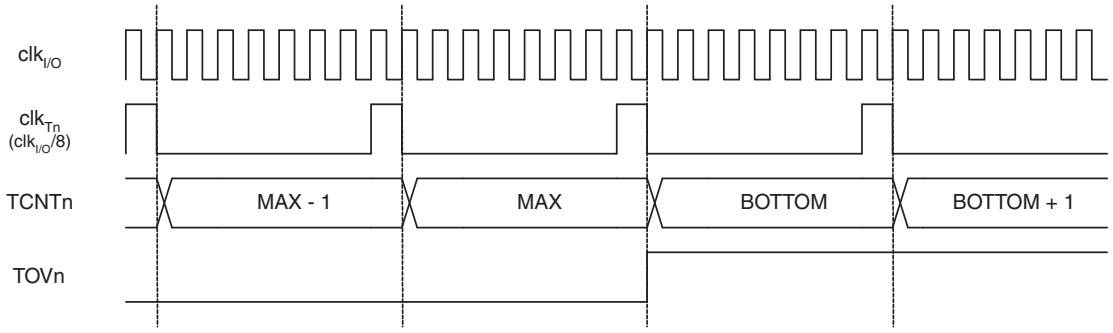


Figure 12-10 on page 89 shows the setting of OCFnB in all modes and OCFnA in all modes except CTC mode and PWM mode, where OCRnA is TOP.

Figure 12-10. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk_I/O}/8$)

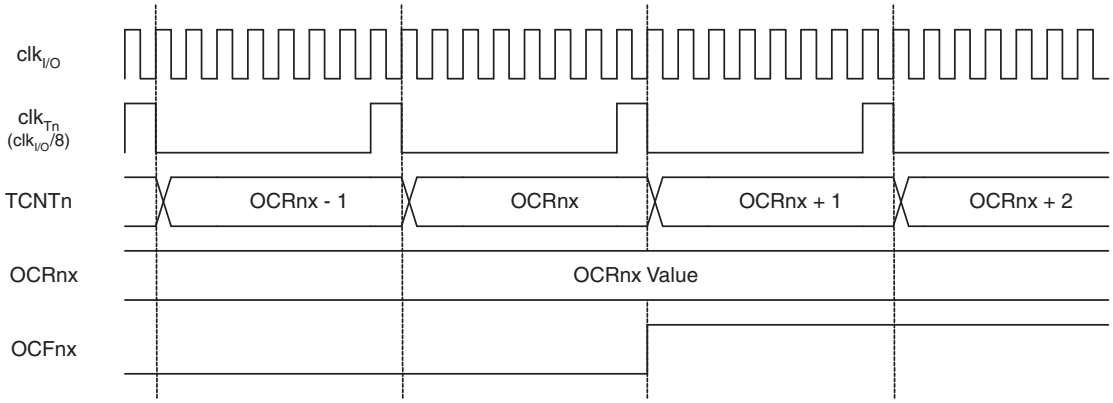
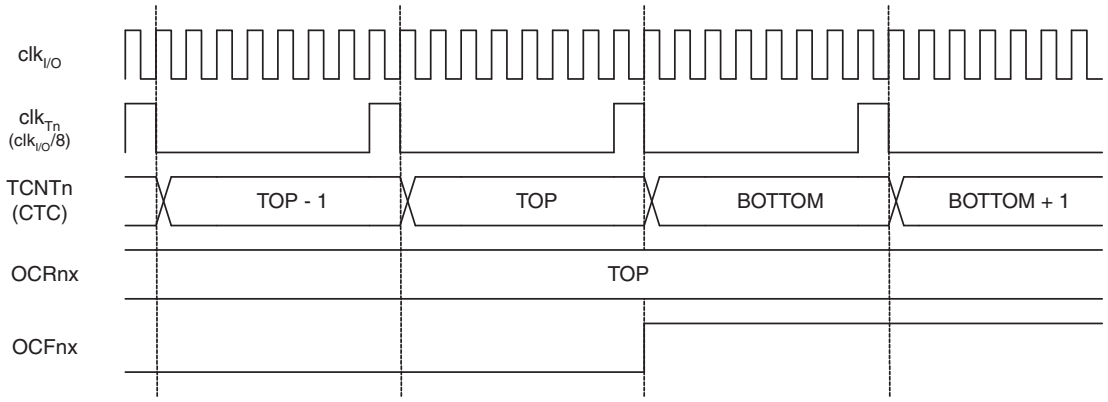


Figure 12-11 on page 90 shows the setting of OCFnA and the clearing of TCNTn in CTC mode and fast PWM mode where OCRnA is TOP.

Figure 12-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk_I/O}/8$)



12.9 Register Description

12.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
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	

12.9.2 TCCR1A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	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 – COMnA[1:0]: Compare Match Output A Mode**

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

When OCnA is connected to the pin, the function of the COMnA[1:0] bits depends on the WGMn[2:0] bit setting. Table 12-2 on page 90 shows the COMnA[1:0] bit functionality when the WGMn[2:0] bits are set to a normal or CTC mode (non-PWM).

Table 12-2. Compare Output Mode, non-PWM Mode

COMnA1	COMnA0	Description
0	0	Normal port operation, OCnA disconnected.
0	1	Toggle OCnA on Compare Match
1	0	Clear OCnA on Compare Match
1	1	Set OCnA on Compare Match

Table 12-3 on page 91 shows the COMnA[1:0] bit functionality when the WGMn[1:0] bits are set to fast PWM mode.

Table 12-3. Compare Output Mode, Fast PWM Mode[1]

COMnA1	COMnA0	Description
0	0	Normal port operation, OCnA disconnected.
0	1	WGMn2 = 0: Normal Port Operation, OCnA Disconnected. WGMn2 = 1: Toggle OCnA on Compare Match.
1	0	Clear OCnA on Compare Match, set OCnA at TOP
1	1	Set OCnA on Compare Match, clear OCnA at TOP

Note: 1. A special case occurs when OCRnA equals TOP and COMnA1 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 85 for more details.

Table 12-4 on page 91 shows the COMnA[1:0] bit functionality when the WGMn[2:0] bits are set to phase correct PWM mode.

Table 12-4. Compare Output Mode, Phase Correct PWM Mode[1]

COMnA1	COMnA0	Description
0	0	Normal port operation, OCnA disconnected.
0	1	WGMn2 = 0: Normal Port Operation, OCnA Disconnected. WGMn2 = 1: Toggle OCnA on Compare Match.
1	0	Clear OCnA on Compare Match when up-counting. Set OCnA on Compare Match when down-counting.
1	1	Set OCnA on Compare Match when up-counting. Clear OCnA on Compare Match when down-counting.

Note: 1. A special case occurs when OCRnA equals TOP and COMnA1 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 87 for more details.

- **Bits 5:4 – COMnB[1:0]: Compare Match Output B Mode**

These bits control the Output Compare pin (OCnB) behavior. If one or both of the COMnB1:0 bits are set, the OCnB output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OCnB pin must be set in order to enable the output driver.

When OCnB is connected to the pin, the function of the COMnB[1:0] bits depends on the WGMn[2:0] bit setting. Table 12-5 on page 91 shows the COMnB[1:0] bit functionality when the WGMn[2:0] bits are set to a normal or CTC mode (non-PWM).

Table 12-5. Compare Output Mode, non-PWM Mode

COMnB1	COMnB0	Description
0	0	Normal port operation, OCnB disconnected.
0	1	Toggle OCnB on Compare Match
1	0	Clear OCnB on Compare Match
1	1	Set OCnB on Compare Match

Table 12-6 on page 92 shows the COMnB[1:0] bit functionality when the WGMn[2:0] bits are set to fast PWM mode.

Table 12-6. Compare Output Mode, Fast PWM Mode[1]

COMnB1	COMnB0	Description
0	0	Normal port operation, OCnB disconnected.
0	1	Reserved
1	0	Clear OCnB on Compare Match, set OC0B at TOP
1	1	Set OCnB on Compare Match, clear OC0B at TOP

Note: 1. A special case occurs when OCRnB equals TOP and COMnB1 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 85 for more details.

Table 12-7 shows the COMnB[1:0] bit functionality when the WGMn2:0 bits are set to phase correct PWM mode.

Table 12-7. Compare Output Mode, Phase Correct PWM Mode[1]

COMnB1	COMnB0	Description
0	0	Normal port operation, OCnB disconnected.
0	1	Reserved
1	0	Clear OCnB on Compare Match when up-counting. Set OCnB on Compare Match when down-counting.
1	1	Set OCnB on Compare Match when up-counting. Clear OCnB on Compare Match when down-counting.

Note: 1. A special case occurs when OCRnB equals TOP and COMnB1 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 87 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved and will always read zero.

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

Combined with the WGMn2 bit found in the TCCRnB 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 12-8 on page 93. 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 84).

Table 12-8. Waveform Generation Mode Bit Description

Mode	WGMn2	WGMn1	WGMn0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRnA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRnA	TOP	TOP

Note: 1. MAX = 0xFF, BOTTOM = 0x00

12.9.3 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.9.4 TCCR1B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	FOC1A	FOC1B	–	–	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

The FOCnA 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 TCCRnB is written when operating in PWM mode. When writing a logical one to the FOCnA bit, an immediate Compare Match is forced on the Waveform Generation unit. The OCnA output is changed according to its COMnA1:0 bits setting. Note that the FOCnA bit is implemented as a strobe. Therefore it is the value present in the COMnA1:0 bits that determines the effect of the forced compare.

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

The FOCnA bit is always read as zero.

- **Bit 6 – FOCnB: Force Output Compare B**

The FOCnB 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 FOCnB bit, an immediate Compare Match is forced on the Waveform Generation unit. The OCnB output is changed according to its COMnB1:0 bits setting. Note that the FOCnB bit is implemented as a strobe. Therefore it is the value present in the COMnB1:0 bits that determines the effect of the forced compare.

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

The FOCnB bit is always read as zero.

- **Bits 5, 4 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 3 – WGMn2: Waveform Generation Mode**

See the description in the “Register Description” on page 90.

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

The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 12-9. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(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 Tn pin. Clock on falling edge.
1	1	1	External clock source on Tn pin. Clock on rising edge.

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

12.9.5 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0[7:0]								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	

12.9.6 TCNT1 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	TCNT1[7:0]								TCNT1
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 TCNTn Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNTn) while the counter is running, introduces a risk of missing a Compare Match between TCNTn and the OCRnx Registers.

12.9.7 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	OCR0A[7:0]								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	

12.9.8 OCR1A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	OCR1A[7:0]								OCR1A
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 (TCNTn). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnA pin.

12.9.9 OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x3C (0x5C)	OCR0B[7:0]								OCR0B
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	

12.9.10 OCR1B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	OCR1B[7:0]								OCR1B
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 B contains an 8-bit value that is continuously compared with the counter value (TCNTn). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnB pin.

12.9.11 TIMSK0 – Timer/Counter 0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.9.12 TIMSK1 – Timer/Counter 1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	–	–	–	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
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 and will always read zero.

- **Bit 2 – OCIEnB: Timer/Countern Output Compare Match B Interrupt Enable**

When the OCIEnB bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Countern occurs, i.e., when the OCFnB bit is set in the Timer/Counter Interrupt Flag Register – TIFRn.

- **Bit 1 – OCIEnA: Timer/Countern Output Compare Match A Interrupt Enable**

When the OCIEnA bit is written to one, and the I-bit in the Status Register is set, the Timer/Countern Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Countern occurs, i.e., when the OCFnA bit is set in the Timer/Counter n Interrupt Flag Register – TIFRn.

- **Bit 0 – TOIEn: Timer/Countern Overflow Interrupt Enable**

When the TOIEn bit is written to one, and the I-bit in the Status Register is set, the Timer/Countern Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Countern occurs, i.e., when the TOVn bit is set in the Timer/Counter n Interrupt Flag Register – TIFRn.

12.9.13 TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

12.9.14 TIFR1 – Timer/Counter 1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	–	–	–	–	–	OCF1B	OCF1A	TOV1	TIFR1
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 and will always read zero.

- **Bit 2 – OCFnB: Output Compare Flag n B**

The OCFnB bit is set when a Compare Match occurs between the Timer/Countern and the data in OCRnB – Output Compare Registern B. OCFnB is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCFnB is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIEnB (Timer/Counter Compare B Match Interrupt Enable), and OCFnB are set, the Timer/Countern Compare Match Interrupt is executed.

- **Bit 1 – OCFnA: Output Compare Flag n A**

The OCFnA bit is set when a Compare Match occurs between the Timer/Countern and the data in OCRnA – Output Compare Registern A. OCFnA is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCFnA is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIEnA (Timer/Countern Compare Match Interrupt Enable), and OCFnA are set, the Timer/Countern Compare Match Interrupt is executed.

- **Bit 0 – TOVn: Timer/Countern Overflow Flag**

The bit TOVn is set when an overflow occurs in Timer/Countern. TOVn is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOVn is cleared by writing a logic one to the flag. When the SREG I-bit, TOIEn (Timer/Countern Overflow Interrupt Enable), and TOVn are set, the Timer/Countern Overflow interrupt is executed.

13. Timer/Counter Prescaler

Timer/Counter0 and Timer/Counter1 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counters. Tn is used as a general name, n = 0, 1.

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 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 $f_{CLK_I/O}/1024$.

13.1 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/CounterCounter, and it is shared by the Timer/Counter Tn. 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 > CSn2:0 > 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).

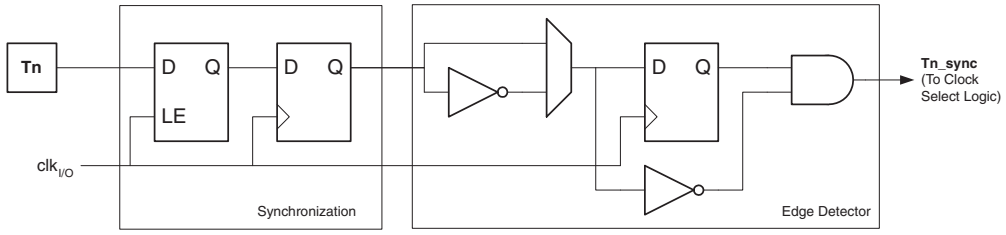
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution.

13.2 External Clock Source

An external clock source applied to the Tn pin can be used as Timer/Counter clock (clk_{Tn}). The Tn 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 13-1 on page 98 shows a functional equivalent block diagram of the Tn 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_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 13-1. Tn 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 Tn pin to the counter is updated.

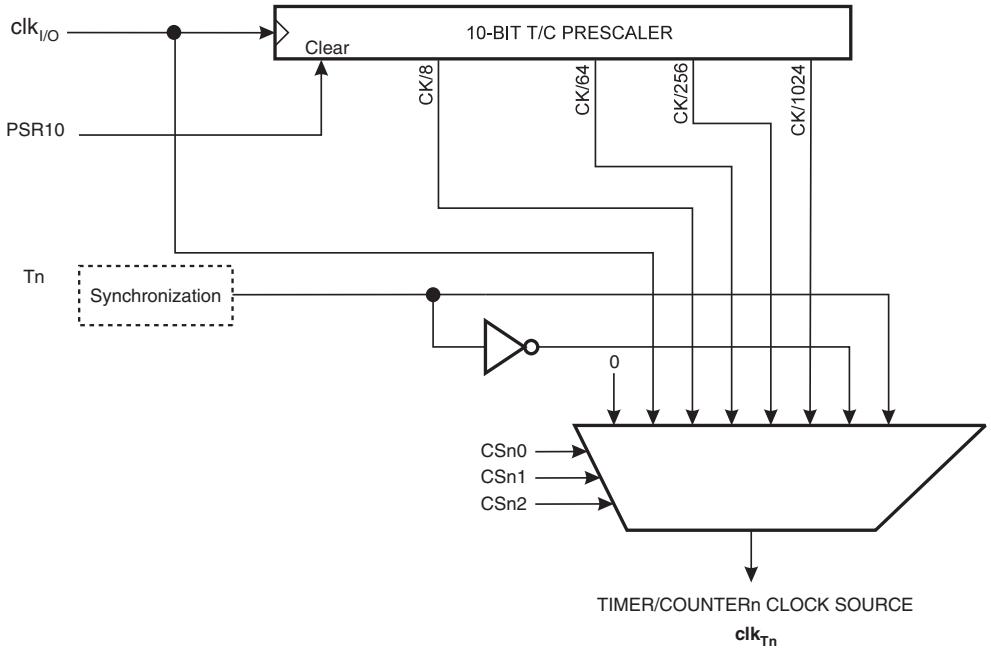
Enabling and disabling of the clock input must be done when Tn has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the sys-

tem 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 13-2. Prescaler for Timer/Countern



Note: 1. The synchronization logic on the input pins (Tn) is shown in Figure 13-1 on page 98.

13.3 Register Description

13.3.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**TSM**	–	–	–	–	–	–	**PSR10**	**GTCCR**
Read/Write	R/W	R	R	R	R	R	R	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 PSR10 bit is kept, hence keeping the Prescaler Reset signal asserted. This ensures that the Timer/Counter is halted and can be configured without the risk of advancing during configuration. When the TSM bit is written to zero, the PSR10 bit is cleared by hardware, and the Timer/Counter start counting.

- **Bit 0 – PSR10: Prescaler Reset Timer/Counter**

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

14. USI – Universal Serial Interface

14.1 Features

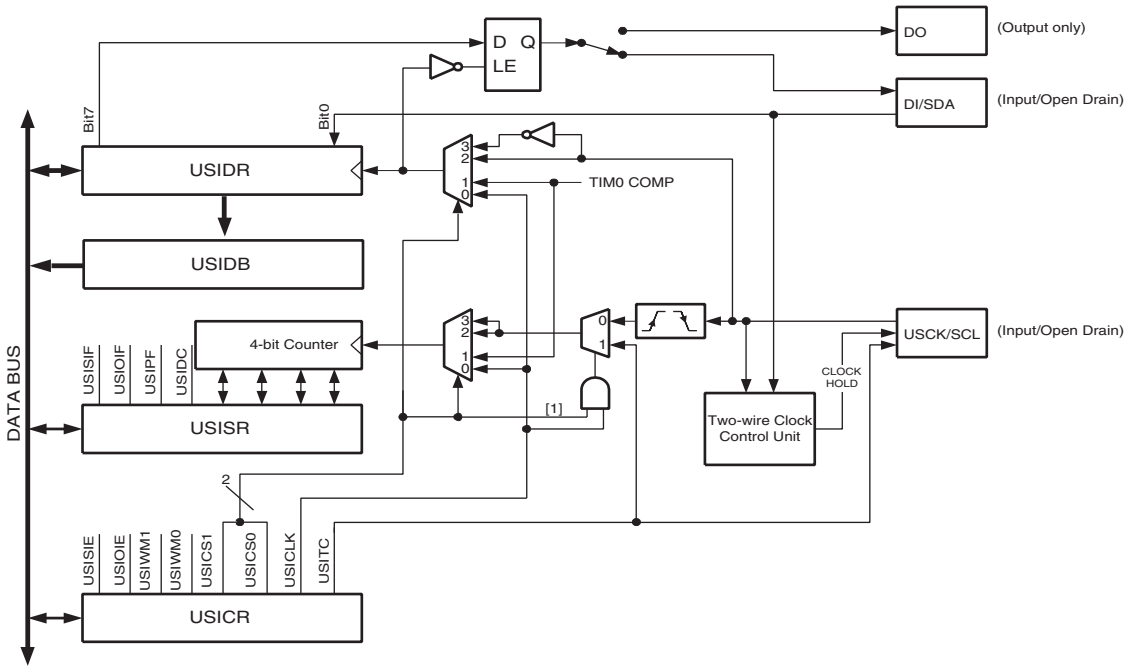
- **Two-wire Synchronous Data Transfer (Master or Slave)**
- **Three-wire Synchronous Data Transfer (Master or Slave)**
- **Data Received Interrupt**
- **Wakeup from Idle Mode**
- **In Two-wire Mode: Wake-up from All Sleep Modes, Including Power-down Mode**
- **Two-wire Start Condition Detector with Interrupt Capability**

14.2 Overview

The Universal Serial Interface (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 in Figure 14-1 on page 100. For actual placement of I/O pins, refer to "Pinout of ATtiny43U" on page 2. 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 Descriptions" on page 107.

Figure 14-1. Universal Serial Interface, Block Diagram



The 8-bit USI Data Register (USIDR) contains the incoming and outgoing data. It is directly accessible via the data bus but a copy of the contents is also placed in the USI Buffer Register (USIBR) where it can be retrieved later. If reading the USI Data Register directly, the register must be read as quickly as possible to ensure that no data is lost.

The most significant bit of the USI Data Register is connected to one of two output pins (depending on the mode configuration, see Table 14-1 on page 108). There is a transparent latch between the output of the USI Data Register and the 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 it 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. This means the counter registers the number of clock edges and not the number of data 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 be configured to generate an interrupt when a start condition has been detected on the two-wire bus. It can also be set to 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 (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

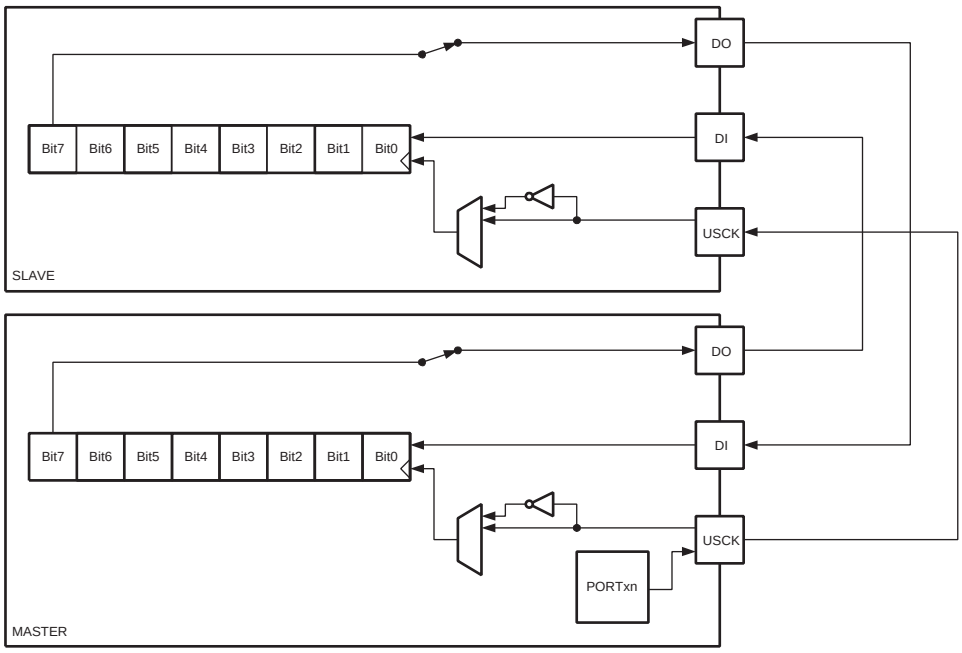
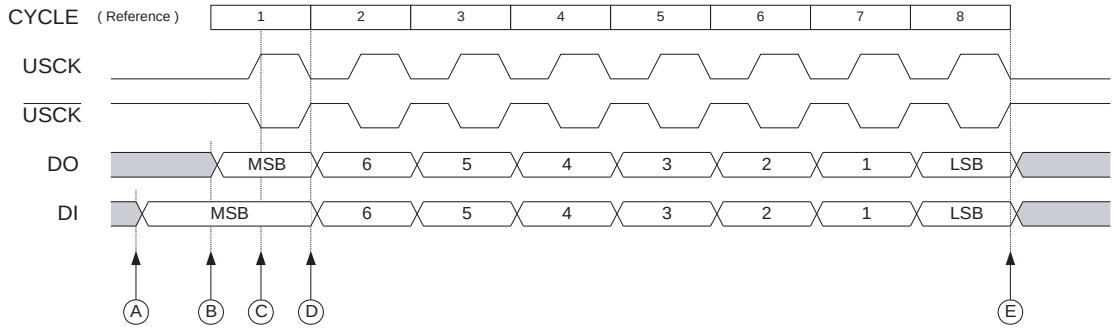


Figure 14-2 shows two USI units operating in three-wire mode, one as Master and one as Slave. The two USI Data Registers are interconnected in such way that after eight USCK clocks, the data in each register has been interchanged. The same clock also increments the USI's 4-bit counter. The Counter Overflow (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 PORTA register or by writing a one to bit 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 (USI Data Register is shifted by one) at negative edges. In external clock mode 1 (USICS0 = 1) the opposite edges with respect to mode 0 are used. In other words, data is sampled at negative and output is changed 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 and master devices set up their data outputs and, depending on the protocol used, enable their output drivers (mark A and B). The output is set up by writing the data to be transmitted to the USI Data Register. The output is enabled by setting the corresponding bit in the Data Direction Register of Port A. Note that there is not a preferred order of points A and B in the figure, but both must be at least one half USCK cycle before point C, where the data is sampled. This is in order to ensure that the data setup requirement is satisfied. The 4-bit counter is reset to zero.
2. The master software generates a clock pulse by toggling the USCK line twice (C and D). The bit values on the data input (DI) pins are 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 has been completed. If USI Buffer Registers are not used the data bytes that have been 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 on 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:
    out     USIDR,r16
    ldi     r16,(1<<USIOIF)
    out     USISR,r16
    ldi     r17,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)

<continues>
```

```
<continued>

SPITransfer_loop:
    out    USICR,r17
    in     r16, USISR
    sbrs   r16, USIOIF
    rjmp   SPITransfer_loop
    in     r16,USIDR
    ret
```

The code is size optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in DDRA. 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 register r16.

The second and third instructions clear the USI Counter Overflow Flag and the USI counter value. The fourth and fifth instructions set three-wire mode, positive edge clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI as an SPI master with maximum speed ($f_{SCK} = f_{CK}/2$):

```
SPITransfer_Fast:
    out    USIDR,r16
    ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
    ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

    out    USICR,r16 ; MSB
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16 ; LSB
    out    USICR,r17

    in     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)
    out     USICR,r16
...
SlaveSPITransfer:
    out     USIDR,r16
    ldi     r16,(1<<USIOIF)
    out     USISR,r16
SlaveSPITransfer_loop:
    in      r16, USISR
    sbrs    r16, USIOIF
    rjmp    SlaveSPITransfer_loop
    in      r16,USIDR
    ret
```

The code is size optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in DDRA. 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 register r16.

Note that the first two instructions are for initialization, only, and need only be executed once. These instructions set three-wire mode and positive edge clock. The loop is repeated until the USI Counter Overflow Flag is set.

14.3.4 Two-wire Mode

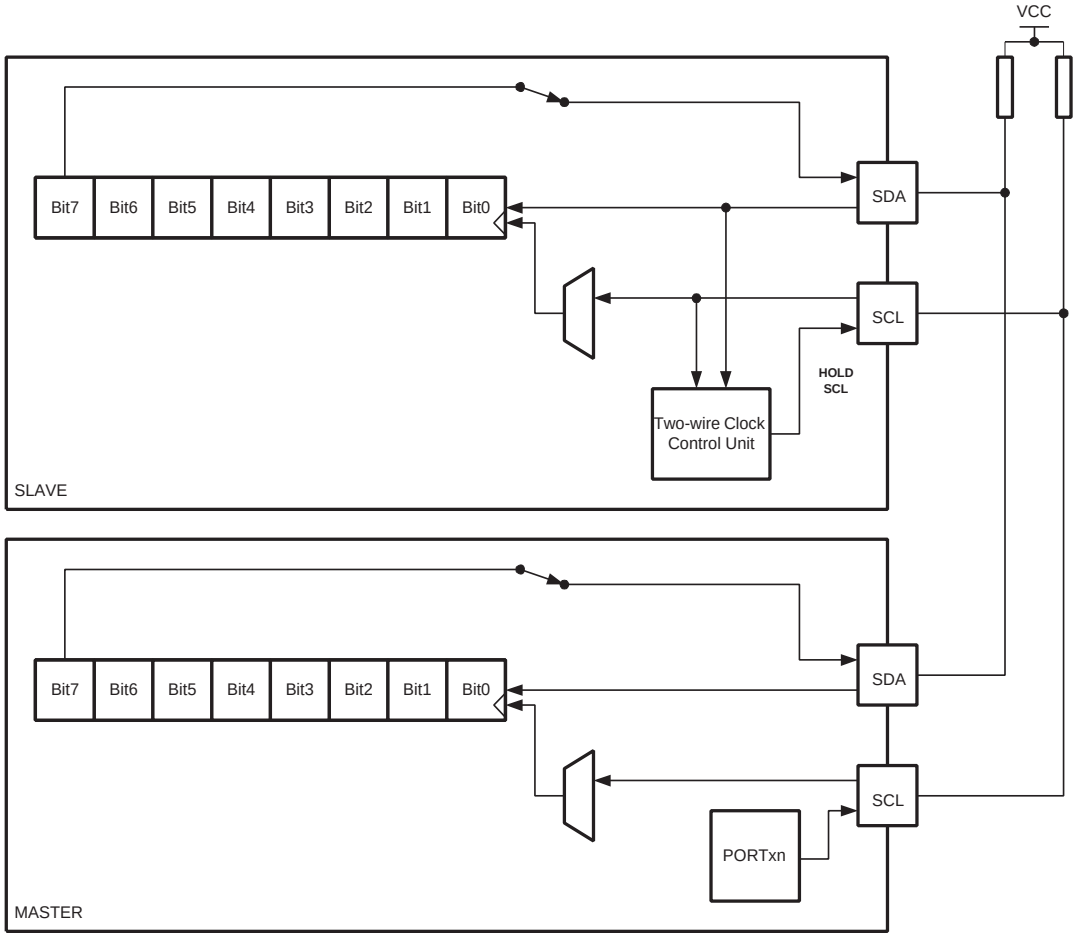
The USI two-wire mode is compliant to the Inter IC (TWI) bus protocol, but without slew rate limiting on outputs and without input noise filtering. Pin names used in this mode are SCL and SDA.

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. Only the slave uses the clock control unit.

Clock generation must be implemented in software, but the shift operation is done automatically in both devices. Note that clocking only on negative edges 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.

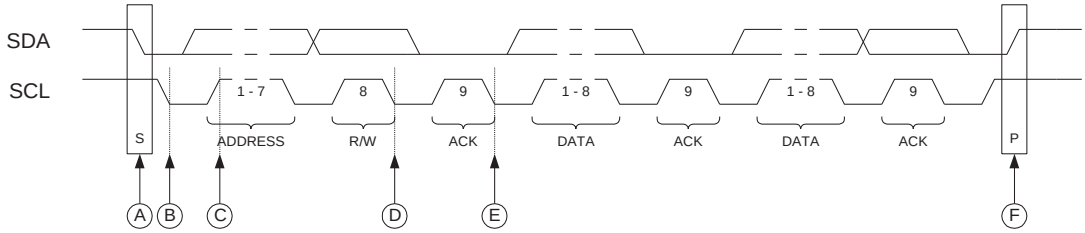
Since the clock also increments the counter, a counter overflow can be used to indicate that the transfer is completed. The clock is generated by the master by toggling the USCK pin via the PORTA register.

Figure 14-4. Two-wire Mode Operation, Simplified Diagram



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 start condition is generated by the master by forcing the SDA low line while keeping the SCL line high (A). SDA can be forced low either by writing a zero to bit 7 of the USI Data Register, or by setting the corresponding bit in the PORTA register to zero. Note that the Data Direction Register bit must be set to one for the output to be enabled. The start detector logic of the slave device (see Figure 14-6 on page 106)

detects the start condition and sets the USISIF Flag. The flag can generate an interrupt if necessary.

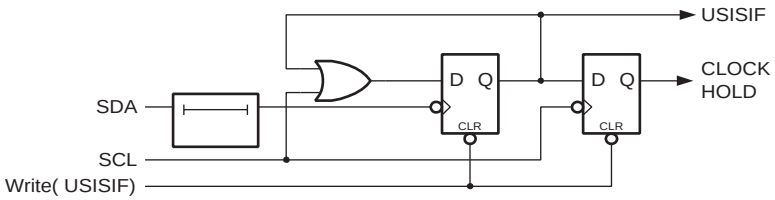
2. In addition, the start detector will hold the SCL line low after the master has forced a negative edge on this line (B). This allows the slave to wake up from sleep or complete other tasks before setting up the USI Data Register to receive the address. This is done by clearing the start condition flag and resetting the counter.
3. The master set the first bit to be transferred and releases the SCL line (C). The slave samples the data and shifts it into the USI Data Register at the positive edge of the SCL clock.
4. After eight bits containing slave address and data direction (read or write) have been transferred, 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. When the slave is addressed, it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the USI Counter Register must be set to 14 before releasing SCL at (D)). Depending on 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 forcing the acknowledge bit low after the last byte transmitted.

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.

Figure 14-6. Start Condition Detector, Logic Diagram



The start condition detector works asynchronously and can therefore wake up the processor from power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature the oscillator start-up time (set by CKSEL fuses, see “Clock Sources” on page 24) must also be taken into consideration. Refer to the description of the USISIF bit on page 112 for further details.

14.4 Alternative USI Usage

The flexible design of the USI allows it to be used for other tasks when serial communication is not needed. Below are some examples.

14.4.1 Half-Duplex Asynchronous Data Transfer

Using 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 increment the counter value.

14.4.3 12-Bit Timer/Counter

Combining the 4-bit USI counter with one of the 8-bit timer/counters creates 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 Descriptions

14.5.1 USICR – USI Control Register

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	**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. See the USISIF bit description in “Analog Comparator” on page 112 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. See the USIOIF bit description in “Analog Comparator” on page 112 for further details.

- **Bit 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 Shift Register can therefore be clocked exter-

nally, and data input sampled, even when outputs are disabled. The relations between USIWM1..0 and the USI operation is summarized in Table 14-1.

Table 14-1. Relationship between USIWM[1:0] and USI Operation

USIWM1	USIWM0	Description
0	0	**Outputs, clock hold, and start detector disabled.** Port pins operate as normal.
0	1	**Three-wire mode. Uses DO, DI, and USCK pins.** The *Data Output* (DO) pin overrides the corresponding bit in the PORTA register. However, the corresponding DDRA bit still controls the data direction. When the port pin is set as input the pin pull-up is controlled by the PORTA 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 PORTA 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 use open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDRA 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 PORTA 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 PORTA 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 in two-wire mode above, except that the SCL line is also held low when a counter overflow occurs, and 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.

- **Bit 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 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. Relationship between the USICS1:0 and USICLK Setting

USICS1	USICS0	USICLK	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 software clock strobe option has been selected by writing USICS1:0 bits to zero. The output will change immediately when the clock strobe is executed, i.e., during the same instruction cycle. The value shifted into the USI Data Register is sampled the previous instruction cycle.

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).

The bit will be read as zero.

- **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 corresponding DDR pin must be set as output (to one). This feature allows easy clock generation when implementing master devices.

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.

The bit will read as zero.

14.5.2 USISR – USI Status Register

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	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	

The Status Register contains interrupt flags, line status flags and the counter value.

- **Bit 7 – USISIF: Start Condition Interrupt Flag**

When two-wire mode is selected, the USISIF Flag is set (to one) when a start condition has been detected. When three-wire mode or output disable mode has been selected any edge on the SCK pin will set the flag.

If USISIE bit in USICR and the Global Interrupt Enable Flag are set, an interrupt will be generated when this flag is 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). If the USIOIE bit in USICR and the Global Interrupt Enable Flag are set an interrupt will also be generated when the flag is 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 has been 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 on the setting of the USICS1:0 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 choosing an external clock source (USICS1 = 1) and writing a one to the USICLK bit.

Note that even when no wire mode is selected (USIWM1..0 = 0) the external clock input (USCK/SCL) can still be used by the counter.

14.5.3 USIDR – USI Data Register

Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	**MSB**							**LSB**	**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	

The USI Data Register can be accessed directly but a copy of the data can also be found in the USI Buffer Register.

Depending on the USICS[1:0] bits of the USI Control Register a (left) shift operation may be performed. The shift operation can be synchronised to an external clock edge, to a Timer/Counter0 Compare Match, or directly to software via the USICLK bit. 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.

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 (DO or SDA, depending on the wire mode) is connected via the output latch to the most significant bit (bit 7) of the USI Data Register. The output latch ensures that data input is sampled and data output is changed on opposite clock edges. The 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 is written as long as the latch is open.

Note that the Data Direction Register bit corresponding to the output pin must be set to one in order to enable data output from the USI Data Register.

14.5.4 USIBR – USI Data Buffer

Bit	7	6	5	4	3	2	1	0	
0x10 (0x30)	MSB							LSB	USIBR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

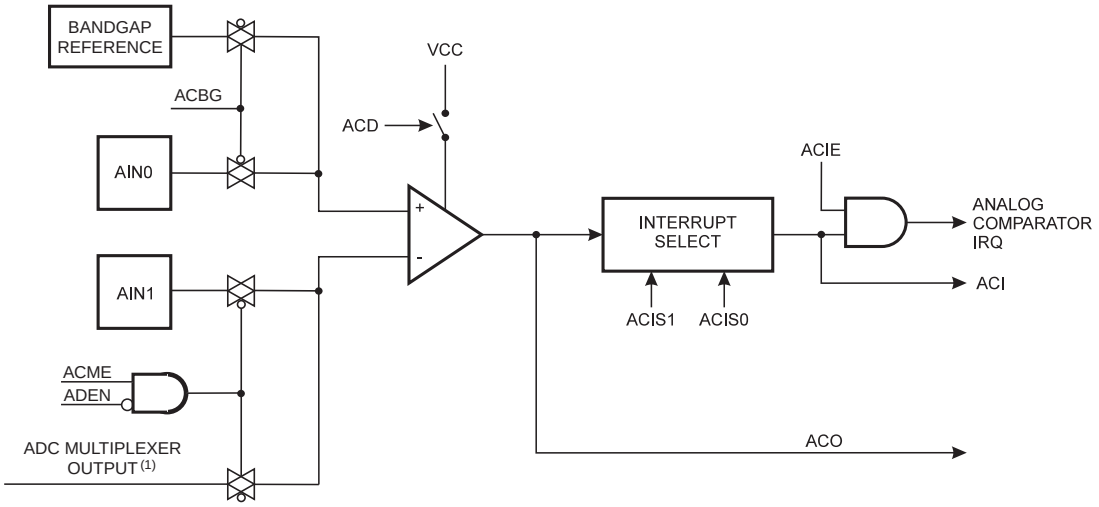
Instead of reading data from the USI Data Register the USI Buffer Register can be used. This makes controlling the USI less time critical and gives the CPU more time to handle other program tasks. USI flags as set similarly as when reading the USIDR register.

The content of the USI Data Register is loaded to the USI Buffer Register when the transfer has been completed.

15. Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator 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 15-1 on page 112.

Figure 15-1. Analog Comparator Block Diagram



Notes: 1. See Table 15-1 on page 113.

See Figure 1-1 on page 2 and Table 11-3 on page 70 for Analog Comparator pin placement.

The ADC Power Reduction bit, PRADC, must be disabled in order to use the ADC input multiplexer. This is done by clearing the PRADC bit in the Power Reduction Register, PRR. See "PRR – Power Reduction Register" on page 35 for more details.

15.1 Analog Comparator Multiplexed Input

It is possible to select any of the ADC[3:0] pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX[1:0] in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 15-1. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 15-1. Analog Comparator Multiplexed Input

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	X	XXX	AIN1
1	1	XXX	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3

15.2 Register Description

15.2.1 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	BS	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	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 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see “Analog Comparator Multiplexed Input” on page 112.

15.2.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	ACD	ACBG	ACO	ACI	ACIE	–	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R	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 in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator.

- **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 – Res: Reserved Bit**

These bits are reserved and will always read zero.

- **Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 15-2.

Table 15-2. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

15.2.3 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	–	–	AIN1D	AIN0D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
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 and will always read zero.

- **Bits 5, 4 – AIN1D, AIN0D: Analog Comparator I/O**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

16. Analog to Digital Converter

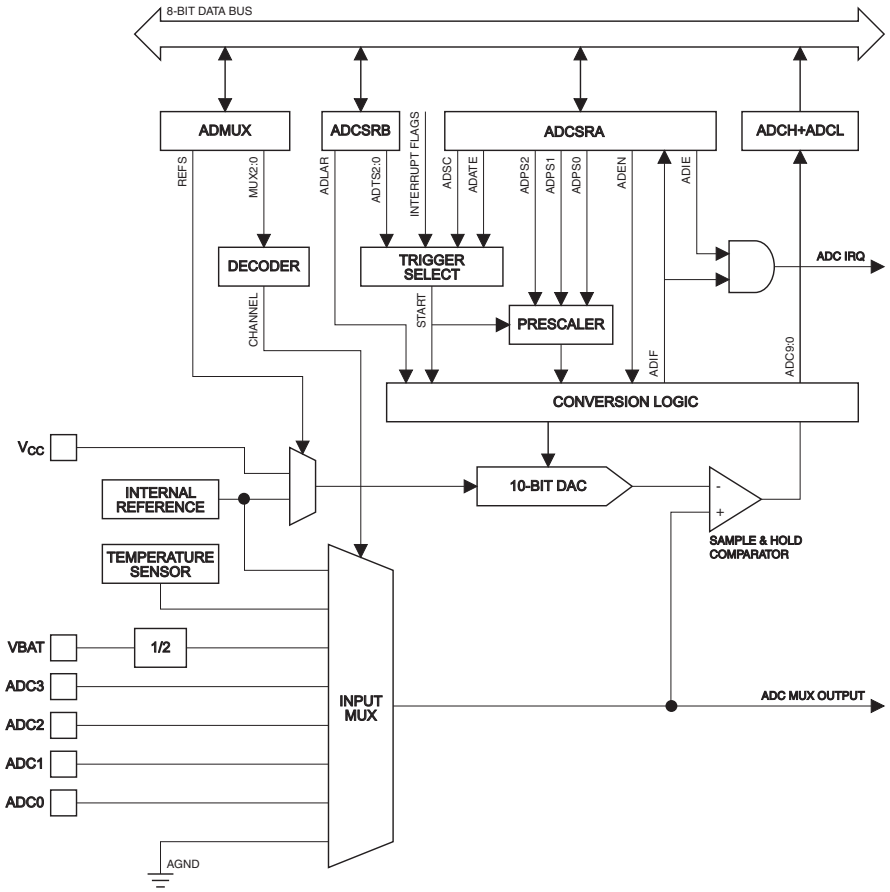
16.1 Features

- 10-bit Resolution
- 1 LSB Integral Non-linearity
- ± 2 LSB Absolute Accuracy
- 13µs Conversion Time
- 15 kSPS at Maximum Resolution
- Four Multiplexed Single Ended Input Channels
- Temperature Sensor Input Channel
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Cancele

16.2 Overview

ATtiny43U features a 10-bit, successive approximation Analog-to-Digital Converter (ADC). A block diagram of the ADC is shown in Figure 16-1.

Figure 16-1. Analog to Digital Converter Block Schematic



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.

The analog multiplexer allows eight single-ended channels to be connected to the ADC, including the low four bits of port A, the internal temperature sensor, the internal voltage reference, supply voltage (V_{BAT}) and ground (GND).

Internal reference voltage of nominally 1.1V is provided on-chip. Alternatively, V_{CC} can be used as reference voltage.

16.3 ADC Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the reference voltage.The voltage reference for the ADC may be selected by writing to the REFS bit in ADMUX. The VCC supply or an internal 1.1V voltage reference may be selected as the ADC voltage reference.

The analog input channel is selected by writing to the MUX2..0 bits in ADMUX. Any of the four ADC input pins ADC3..0, and V_{BAT} input pin can be selected as single ended input to the ADC. The on-chip temperature sensor is selected by writing “111” to the MUX2..0 bits in the ADMUX register.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. 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 ADCSRB.

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.

16.4 Starting a Conversion

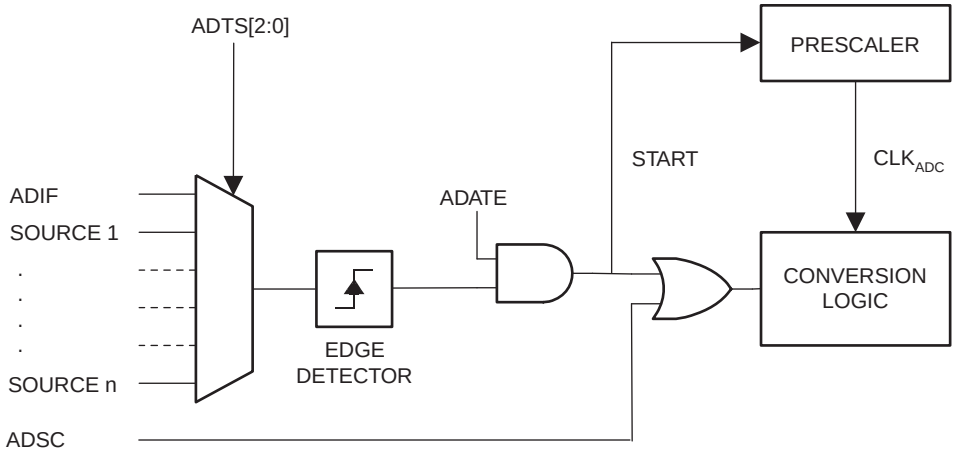
Make sure the ADC is powered by clearing the ADC Power Reduction bit, PRADC, in the Power Reduction Register, PRR (see “PRR – Power Reduction Register” on page 35).

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. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB (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 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 16-2. 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. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

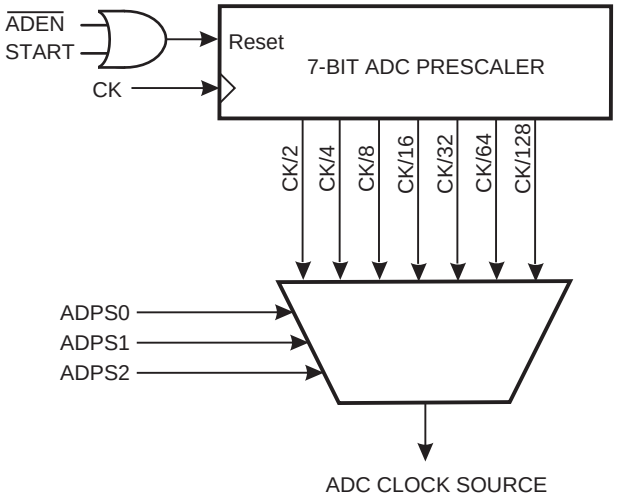
If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

16.5 Prescaling and Conversion Timing

By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. 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. It is not recommended to use a higher input clock frequency than 1 MHz.

The ADC module contains a prescaler, as illustrated in Figure 16-3 on page 118, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

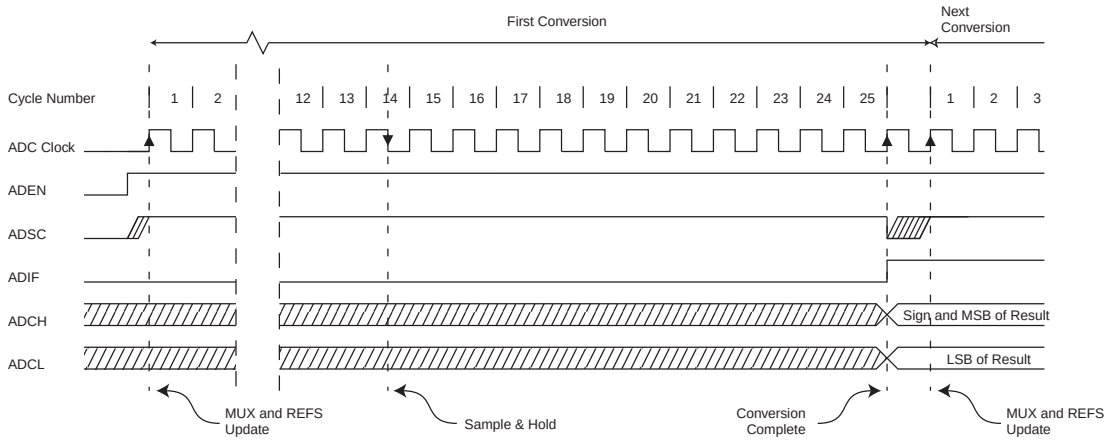
Figure 16-3. ADC Prescaler



When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

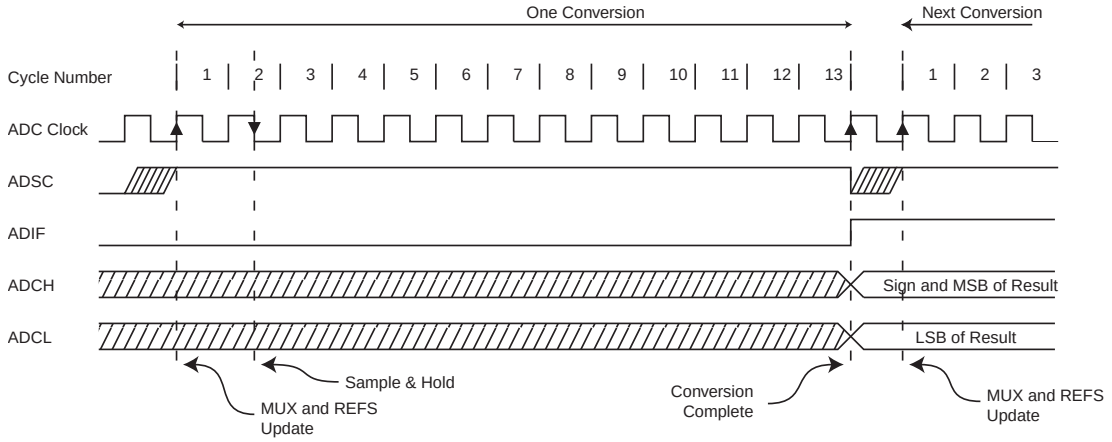
A normal conversion takes 13 ADC clock cycles, as summarised in Table 16-1 on page 120. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry, as shown in Figure 16-4 below.

Figure 16-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)



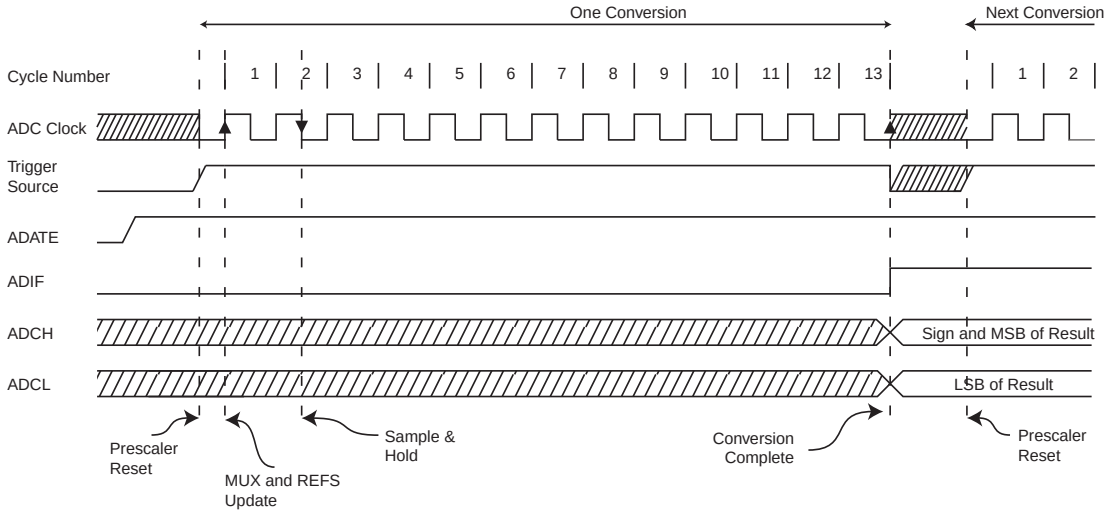
The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of a first conversion. See Figure 16-5. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, 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.

Figure 16-5. ADC Timing Diagram, Single Conversion



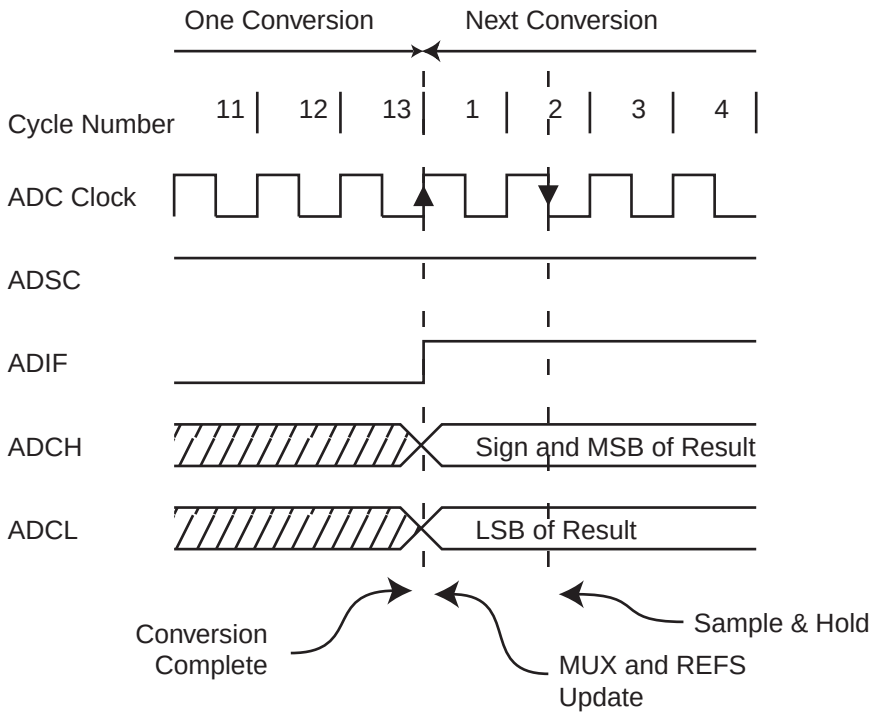
When Auto Triggering is used, the prescaler is reset when the trigger event occurs, as shown in Figure 16-6 below. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

Figure 16-6. ADC Timing Diagram, Auto Triggered Conversion



In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. See Figure 16-7.

Figure 16-7. ADC Timing Diagram, Free Running Conversion



For a summary of conversion times, see Table 16-1.

Table 16-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions	1.5	13
Auto Triggered conversions	2	13.5
Free Running conversion	2.5	14

16.6 Changing Channel or Reference Selection

The MUX[2:0] and REFS 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 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:

- When ADATE or ADEN is cleared.
- During conversion, minimum one ADC clock cycle after the trigger event.
- 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.

16.6.1 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.

16.6.2 ADC Voltage Reference

The ADC reference voltage (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 V_{CC}, or internal 1.1V reference, or external AREF pin. The internal 1.1V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier.

The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

16.7 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode. This reduces 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:

- 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.
- Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
- 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 automatically be 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.

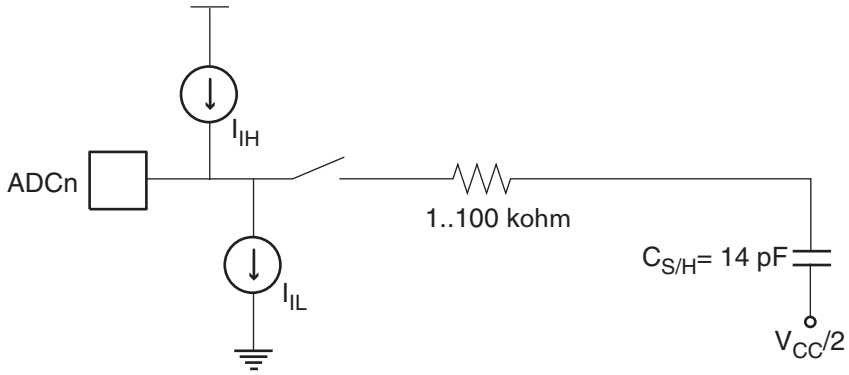
16.8 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 16-8. 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 10kΩ 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, which can vary widely. With slowly varying signals the user is recommended to use sources with low impedance, only, since this minimizes the required charge transfer to the S/H capacitor.

In order to avoid distortion from unpredictable signal convolution, signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 16-8. Analog Input Circuitry



Note: The capacitor in the figure depicts the total capacitance, including the sample/hold capacitor and any stray or parasitic capacitance inside the device. The value given is worst case.

16.9 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

- Keep analog signal paths as short as possible.
- Make sure analog tracks run over the analog ground plane.

- Keep analog tracks well away from high-speed switching digital tracks.
- Use the ADC noise canceler function to reduce induced noise from the CPU.
- If any port pin is used as a digital output, it mustn't switch while a conversion is in progress.
- Place bypass capacitors as close to V_{CC} and GND pins as possible.

Where high ADC accuracy is required it is recommended to use ADC Noise Reduction Mode, as described in Section 16.7 on page 121. This is especially the case when system clock frequency is above 1 MHz, or when the ADC is used for reading the internal temperature sensor. A good system design with properly placed, external bypass capacitors does reduce the need for using ADC Noise Reduction Mode

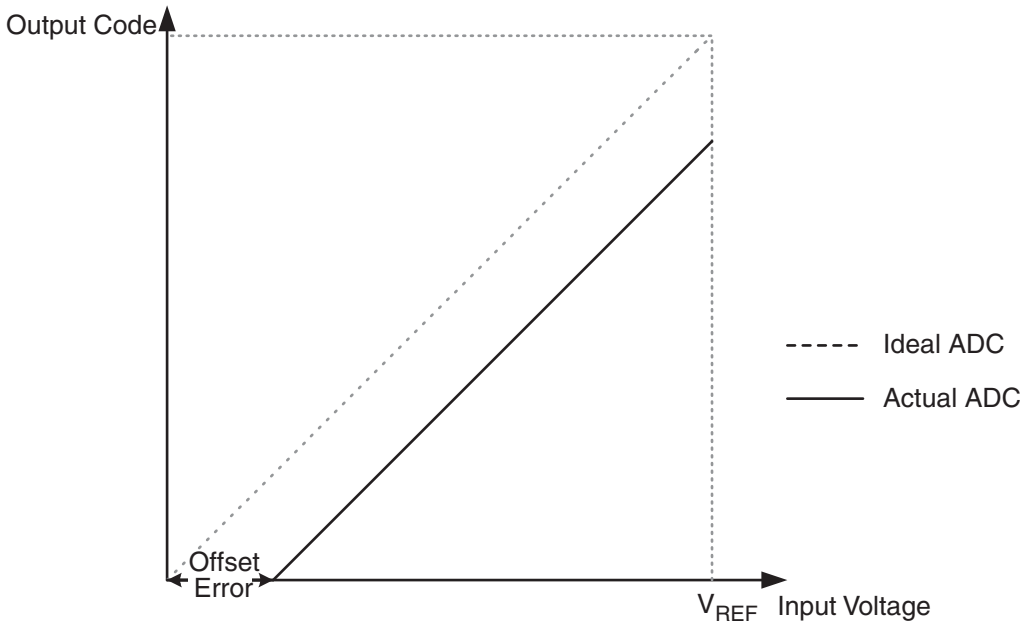
16.10 ADC Accuracy Definitions

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:

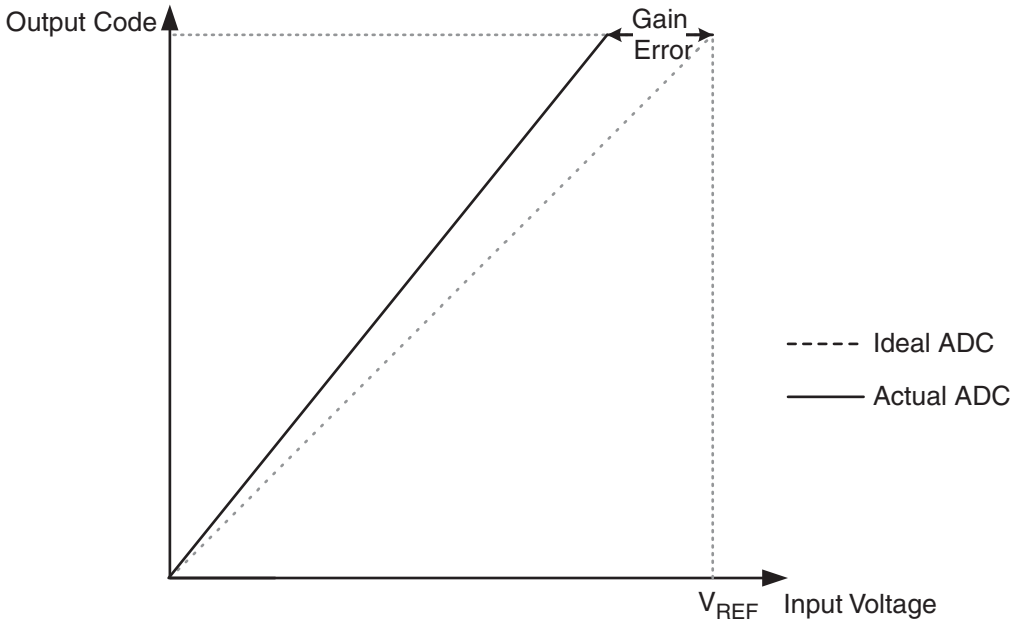
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 16-9. Offset Error



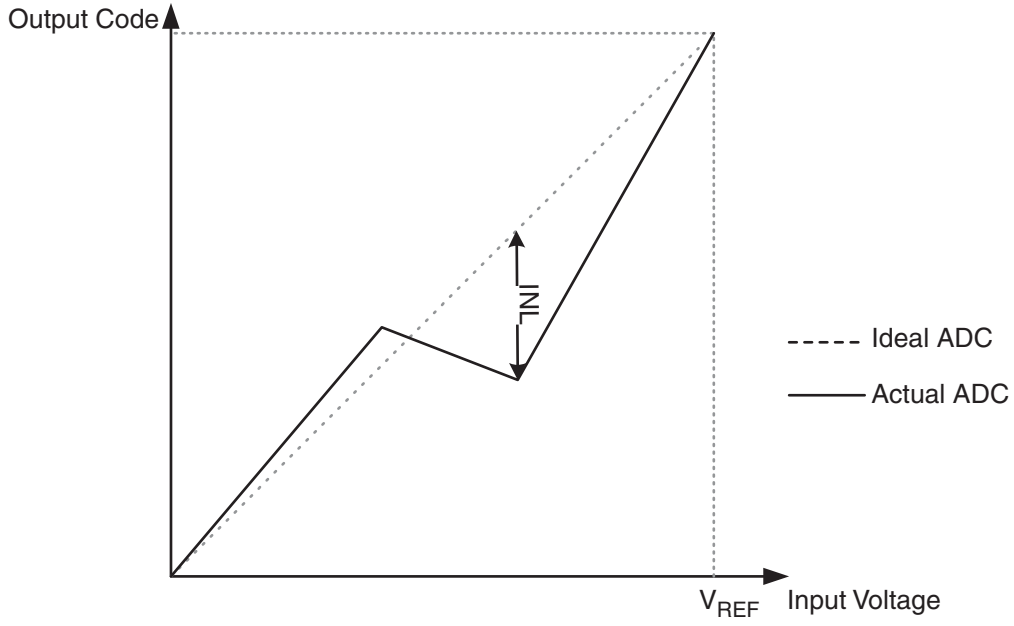
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 16-10. Gain Error



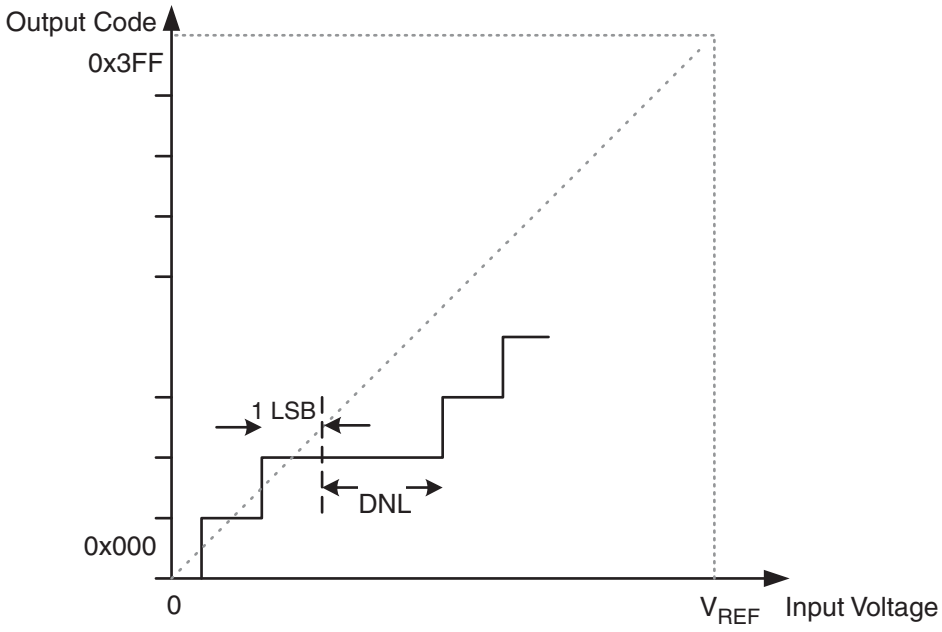
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 16-11. 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 16-12. 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

16.11 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH). 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 16-3 on page 126 and Table 16-4 on page 127). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

16.12 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC channel. Selecting the ADC4 channel by writing the MUX2:0 bits in ADMUX register to “111” enables the temperature sensor. The internal 1.1V reference must also be selected for the ADC reference source in the temperature sensor measurement. When the tem-

perature 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 16-2 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 16-2. Temperature vs. Sensor Output Voltage (Typical Case)

Temperature	**-40 °C**	**+25 °C**	**+85 °C**
ADC	230 LSB	300 LSB	370 LSB

The values described in Table 16-2 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, 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.

16.13 Register Description

16.13.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	–	**REFS**	–	–	–	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is reserved and must always be written zero.

- **Bit 6 – REFS: Reference Selection Bit**

This bit selects the voltage reference for the ADC, as shown in Table 16-3 on page 126. If this bit is changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set).

Table 16-3. Voltage Reference Selections for ADC

REFS	**Voltage Reference Selection**
0	V_{CC} used as analog reference.
1	Internal 1.1V Voltage Reference.

- **Bits 5:3 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bits 2:0 – MUX[2:0]: Analog Channel Selection Bits**

The value of these bits selects which analog input is connected to the ADC, as shown in Table 16-4. Selecting channel ADC4 enables temperature measurement.

Table 16-4. ADC Multiplexer Channel Selections.

Single Ended Input	MUX[2:0]
ADC0 (PA0)	000
ADC1 (PA1)	001
ADC2 (PA2)	010
ADC3 (PA3)	011
0V (GND)	100
1.1V (I Ref)	101
V_{BAT} (1)	110
ADC4 (2)	111

Notes: 1. Due to the voltage divider present, a current will flow from V_{BAT} to ground via a 100kΩ resistor divider as long as this channel is selected.

2. See "Temperature Measurement" on page 125.

If these bits are changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSRA is set).

16.13.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	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 instruction is 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 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 16-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

16.13.3 ADCL and ADCH – ADC Data Register

16.13.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	–	–	–	–	–	–	ADC9	ADC8	ADCH
0x04 (0x24)	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	

16.13.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x04 (0x24)	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 ADCSRB, 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 125.

16.13.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	BS	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	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 5 – Res: Reserved Bit**

This bit is reserved and will always read what was written.

- **Bit 4 – 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 comple the description of this bit, see “ADCL and ADCH – ADC Data Register” on page 128.

- **Bit 3 – Res: Reserved Bit**

This bit is reserved and will always read what was written.

- **Bits 2:0 – ADTS[2:0]: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS[2:0] settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 16-6. 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/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow

Table 16-6. ADC Auto Trigger Source Selections (Continued)

ADTS2	ADTS1	ADTS0	Trigger Source
1	0	1	Timer/Counter1 Compare Match A
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Compare Match B

16.13.5 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	–	–	AIN1D	AIN0D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
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 3:0 – ADC[3:0]D: ADC[3:0] 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[3:0] 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. debugWIRE On-chip Debug System

17.1 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**

17.2 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.

17.3 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 17-1. The debugWIRE Setup

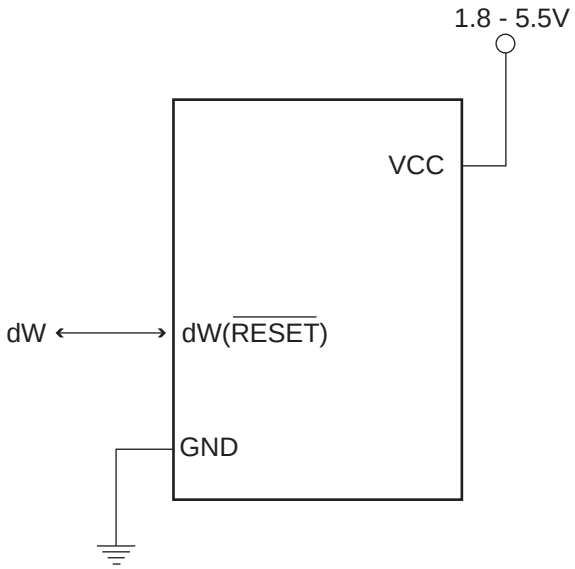


Figure 17-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-Up resistor on the dW/(RESET) line must be in the range of 10k to 20 kΩ. However, the pull-up resistor is optional.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors inserted on the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

17.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR 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 AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Falsh Data retention. Devices used for debugging purposes should not be shipped to end customers.

17.5 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 (AVR Studio). See the debugWIRE documentation for detailed description of the limitations.

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.

17.6 Register Description

The following section describes the registers used with the debugWire.

17.6.1 DWDR – debugWire Data Register

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	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.

18. Self-Programming the Flash

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 and associated protocol to read code and write (program) that code into the Program memory. The SPM instruction is disabled by default but it can be enabled by programming the SELFPRGEN fuse (to “0”).

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.

18.1 Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “00000011” 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.

18.2 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” 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.

18.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00000101” 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.

18.4 Addressing the Flash During Self-Programming

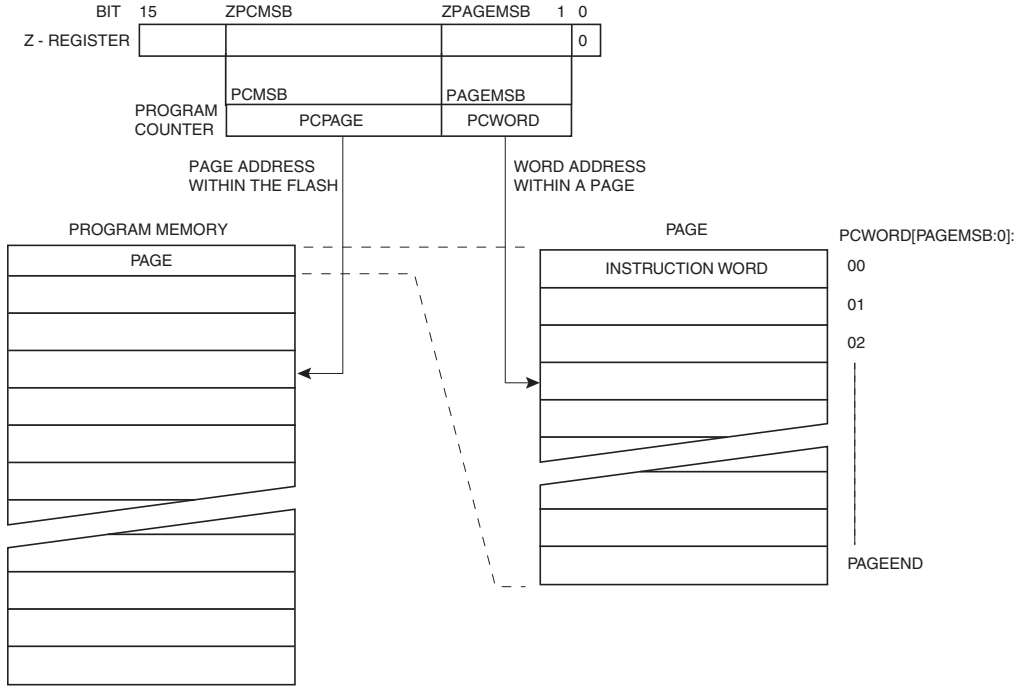
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 19-8 on page 142), 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 19-7 on page 151. 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 18-1. Addressing the Flash During SPM[1]



Note: 1. The different variables used in Figure 18-1 are listed in Table 19-8 on page 142.

18.5 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.

18.6 Reading the Fuse and Lock Bits from Software

It is possible for firmware to read device fuse and lock bits. In addition, firmware can also read data from the device signature imprint table (see page 141).

Note: Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

18.6.1 Reading Lock Bits from Firmware

Lock bit values are returned in the destination register after an LPM instruction has been issued within three CPU cycles after RFLB and SPMEN bits have been set in SPMCSR. The RFLB and SPMEN bits automatically clear upon completion of reading the lock bits, or if no LPM instruction is executed within three CPU cycles, or if no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared LPM functions normally.

To read the lock bits, follow the below procedure:

1. Load the Z-pointer with 0x0001.
2. Set RFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the lock bits from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

See section “Program And Data Memory Lock Bits” on page 139 for more information.

18.6.2 Reading Fuse Bits from Firmware

The algorithm for reading fuse bytes is similar to the one described above for reading lock bits, only the addresses are different. To read the Fuse Low Byte (FLB), follow the below procedure:

1. Load the Z-pointer with 0x0000.
2. Set RFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the FLB from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Refer to Table 19-5 on page 141 for a detailed description and mapping of the Fuse Low Byte.

To read the Fuse High Byte (FHB), simply replace the address in the Z-pointer with 0x0003 and repeat the procedure above. If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Refer to Table 19-4 on page 140 for detailed description and mapping of the Fuse High Byte.

To read the Fuse Extended Byte (FEB), replace the address in the Z-pointer with 0x0002 and repeat the previous procedure. If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FEB7	FEB6	FEB5	FEB4	FEB3	FEB2	FEB1	FEB0

Refer to Table 19-3 on page 140 for detailed description and mapping of the Fuse Extended Byte.

18.6.3 Reading Device Signature Imprint Table from Firmware

To read the contents of the device signature imprint table, follow the below procedure:

1. Load the Z-pointer with the table index.
2. Set RSIG and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read table data from the LPM destination register.

See program example below.

Assembly Code Example

```
DSIT_read:
  ; Uses Z-pointer as table index
  ldi  ZH, 0
  ldi  ZL, 1
  ; Preload SPMCSR bits into R16, then write to SPMCSR
  ldi  r16, (1<<RSIG)|(1<<SPMEN)
  out  SPMCSR, r16
  ; Issue LPM. Table data will be returned into r17
  lpm  r17, Z
  ret
```

Note: See "Code Examples" on page 6.

If successful, the contents of the destination register are as described in section "Device Signature Imprint Table" on page 141.

18.7 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.

18.8 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 18-1 shows the typical programming time for Flash accesses from the CPU.

Table 18-1. 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

18.9 Register Description

18.9.1 SPMCSR – Store Program Memory Control and Status Register

The Store Program Memory Control and Status Register contains the control bits needed to control the Program memory operations.

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	–	–	–	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	

- **Bits 7:5 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **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 “EEPROM Write Prevents Writing to SPMCSR” on page 135 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 Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either 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 “10001”, “01001”, “00101”, “00011” or “00001” in the lower five bits will have no effect.

19. Memory Programming

This section describes the different methods for Programming the ATtiny43U memories.

19.1 Program And Data Memory Lock Bits

The ATtiny43U provides two Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional security listed in Table 19-2 on page 139. The Lock bits can only be erased to “1” with the Chip Erase command.

The device has no separate boot loader section. The SPM instruction is enabled for the whole Flash, if the SELFPROGEN fuse is programmed (“0”), otherwise it is disabled.

Program memory can be read out via the debugWIRE interface when the DWEN fuse is programmed, even if lock bits are set. Thus, when lock bit security is required, debugWIRE should always be disabled by clearing the DWEN fuse.

Table 19-1. Lock Bit Byte(1)

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. “1” means unprogrammed, “0” means programmed

Table 19-2. Lock Bit Protection Modes.

Memory Lock Bits (1) (2)			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 High-voltage and Serial Programming mode. The Fuse bits are locked in both Serial and High-voltage Programming mode.(1) debugWire is disabled.
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in High-voltage and Serial Programming mode. The Fuse bits are locked in both Serial and High-voltage Programming mode.(1) debugWire is disabled.

Notes: 1. Program fuse bits before programming LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

Lock bits can also be read by device firmware. See section “Reading the Fuse and Lock Bits from Software” on page 135.

19.2 Fuse Bytes

The ATtiny43U has three Fuse bytes. Table 19-3, Table 19-4 and Table 19-5 briefly describe the functionality of all the fuses and how they are mapped into the Fuse bytes. Note that the fuses are read as logical zero, “0”, if they are programmed..

Table 19-3. Fuse Extended Byte

Fuse High 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[1]	0	Self-Programming Enable	1 (unprogrammed)

Notes: 1. Enables SPM instruction. See “Self-Programming the Flash” on page 133.

Table 19-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL[1]	7	External Reset disable	1 (unprogrammed)
DWEN[2]	6	DebugWIRE Enable	1 (unprogrammed)
SPIEN[3]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON[4]	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2[5]	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1[5]	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0[5]	0	Brown-out Detector trigger level	1 (unprogrammed)

Notes:
1. See “Alternate Functions of Port B” on page 74 for description of RSTDISBL and DWEN Fuses. When programming the RSTDISBL Fuse, TBD programming has to be used to change fuses to perform further programming
2. DWEN must be unprogrammed when Lock Bit security is required. See “Program And Data Memory Lock Bits” on page 139.
3. The SPIEN Fuse is not accessible in SPI Programming mode.
4. See “WDT Configuration as a Function of the Fuse Settings of WDTON” on page 52 for details.
5. See Table 20-5 on page 158 for BODLEVEL Fuse decoding.

Table 19-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8[1]	7	Divide clock by 8	0 (programmed)
CKOUT	6	Clock Output Enable	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[2]
SUT0	4	Select start-up time	0 (programmed)[2]
CKSEL3	3	Select Clock source	0 (programmed)[3]
CKSEL2	2	Select Clock source	0 (programmed)[3]
CKSEL1	1	Select Clock source	1 (unprogrammed)[3]
CKSEL0	0	Select Clock source	0 (programmed)[3]

Notes:
1. See "Power Management and Sleep Modes" on page 31 for details.
2. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 6-5 on page 26 for details.
3. The default setting of CKSEL3..0 results in internal RC Oscillator @ 8.0 MHz. See Table 6-4 on page 25 for details.

Note that fuse bits are locked if Lock Bit 1 (LB1) is programmed. Fuse bits should be programmed before lock bits. The status of fuse bits is not affected by chip erase.

Fuse bits can also be read by device firmware. See section "Reading the Fuse and Lock Bits from Software" on page 135.

19.2.1 Latching of Fuses

Fuse values are latched when the device enters programming mode and changes to fuse values 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. Fuses are also latched on power-up.

19.3 Device Signature Imprint Table

The device signature imprint table is a dedicated memory area used for storing miscellaneous device information, such as the device signature and oscillator calibaration data. Most of this memory segment is reserved for internal use, as outlined in Table 19-6.

Table 19-6. Contents of Device Signature Imprint Table.

Address	High Byte
0x00	Signature byte 0 [1]
0x01	Calibration data for internal oscillator [2]
0x02	Signature byte 1 [1]
0x03	Reserved for internal use
0x04	Signature byte 2 [1]
0x05 ... 0x2A	Reserved for internal use

Notes:
1. See section "Signature Bytes" for more information.
2. See section "Calibration Byte" for more information.

19.3.1 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and High-voltage Programming mode, also when the device is locked.

Signature bytes can also be read by the device firmware. See section “Reading the Fuse and Lock Bits from Software” on page 135.

The three signature bytes reside in a separate address space called the device signature imprint table. The signature data for ATtiny43U is given in Table 19-7.

Table 19-7. Device Signature Byte

Part	Signature Byte 0	Signature Byte 1	Signature Byte 0
ATtiny43U	0x1E	0x92	0x0C

19.3.2 Calibration Byte

The device signature imprint table of ATtiny43U contains one byte of calibration data for the internal oscillator, as shown in Table 19-6 on page 141. During reset, this byte is automatically written into the OSCCAL register to ensure correct frequency of the calibrated oscillator.

Calibration bytes can also be read by the device firmware. See section “Reading the Fuse and Lock Bits from Software” on page 135.

19.4 Page Size

Table 19-8. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATtiny43U	2K words (4K bytes)	32 words	PC[4:0]	64	PC[10:5]	10

Table 19-9. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATtiny43U	64 bytes	4 bytes	EEA[1:0]	16	EEA[5:2]	5

19.5 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 ATtiny43U. Pulses are assumed to be at least 250 ns unless otherwise noted.

19.5.1 Signal Names

In this section, some pins of the ATtiny43U are referenced by signal names describing their functionality during parallel programming, see Figure 19-1 and Table 19-10. Pins not described in the following table are referenced by pin names.

Figure 19-1. Parallel Programming

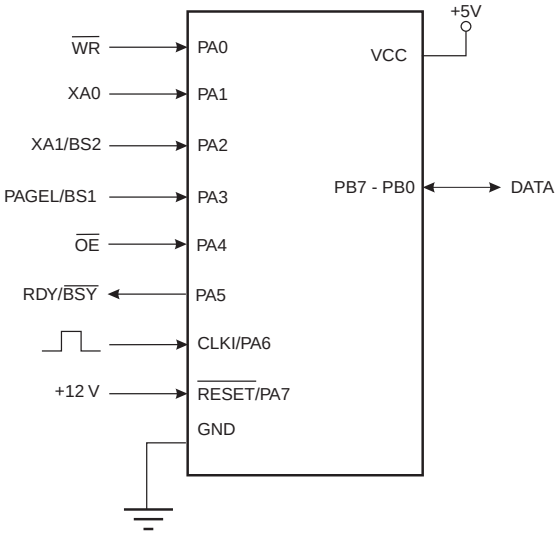


Table 19-10. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
$\overline{WR}$	PA0	I	Write Pulse (Active low).
XA0	PA1	I	CLKI Action Bit 0
XA1/BS2	PA2	I	CLKI Action Bit 1. Byte Select 2 (“0” selects low byte, “1” selects 2’nd high byte).
PAGEL/BS1	PA3	I	Byte Select 1 (“0” selects low byte, “1” selects high byte). Program Memory and EEPROM Data Page Load.
$\overline{OE}$	PA5	I	Output Enable (Active low).
RDY/$\overline{BSY}$	PA6	O	0: Device is busy programming, 1: Device is ready for new command.
DATA I/O	PB7-PB0	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low).

Table 19-11. 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
WR	Prog_enable[0]	0

The XA1/XA0 pins determine the action executed when the CLKI pin is given a positive pulse. The bit coding is shown in Table 19-12.

Table 19-12. XA1 and XA0 Coding

XA1	XA0	Action when CLKI 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

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 19-13.

Table 19-13. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature Bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

19.6 Parallel Programming

19.6.1 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 $\overline{RESET}$ to “0” and toggle CLKI at least six times.
3. Set the Prog_enable pins listed in Table 19-11 on page 143 to “0000” and wait at least 100 ns.
4. Apply 11.5 - 12.5V to $\overline{RESET}$. Any activity on Prog_enable pins within 100 ns after +12V has been applied to $\overline{RESET}$, will cause the device to fail entering programming mode.
5. Wait at least 50 µs before sending a new command.

19.6.2 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.

19.6.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM[1] 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: 1. The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give CLKI 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.

19.6.4 Programming the Flash

The Flash is organized in pages, see Table 19-8 on page 142. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give CLKI 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 CLKI a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give CLKI a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the data byte.

E. No action

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 19-2 on page 147. 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 “00”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the address high byte.

H. Program Page

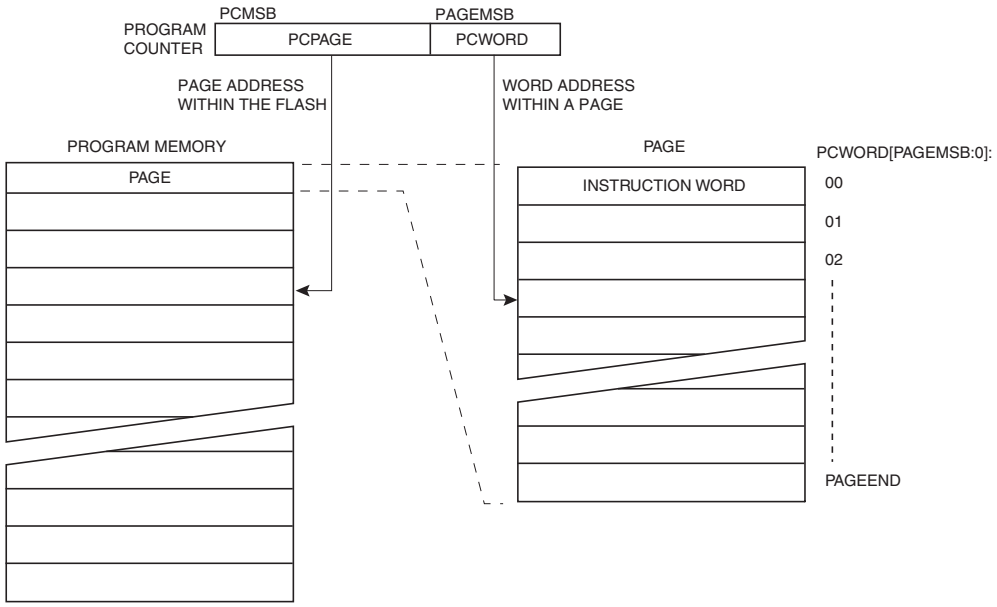
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (See Figure 19-3 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

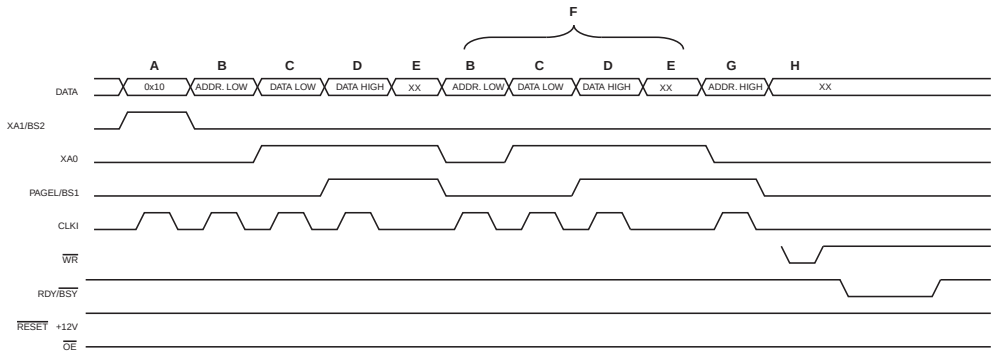
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give CLKI a positive pulse. This loads the command, and the internal write signals are reset.

Figure 19-2. Addressing the Flash Which is Organized in Pages[1]



Note: 1. PCPAGE and PCWORD are listed in Table 19-8 on page 142.

Figure 19-3. Programming the Flash Waveforms[1]



Note: 1. “XX” is don’t care. The letters refer to the programming description above.

19.6.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 19-9 on page 142. 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 145 for details on Command, Address and Data loading):

1. A: Load Command “0001 0001”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. C: Load Data (0x00 - 0xFF).

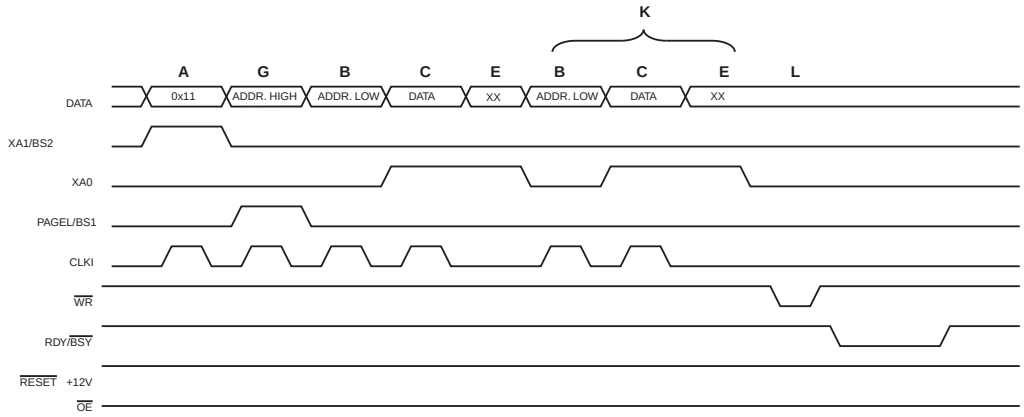
5. E: No action.

K: Repeat 3 through 5 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 19-4 for signal waveforms).

Figure 19-4. Programming the EEPROM Waveforms



19.6.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 145 for details on Command and Address loading):

1. A: Load Command “0000 0010”.
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”.

19.6.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 145 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
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”.

19.6.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 145 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

19.6.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 145 for details on Command and Data loading):

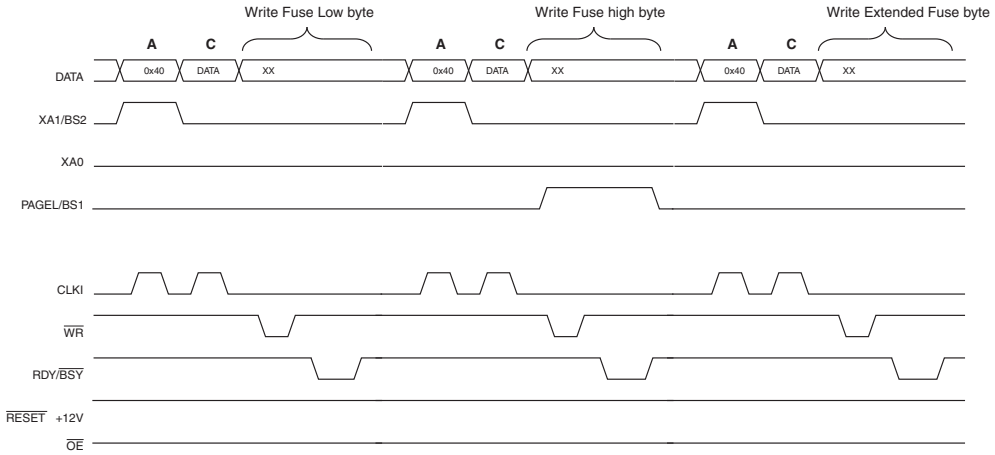
1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

19.6.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to “Programming the Flash” on page 145 for details on Command and Data loading):

1. 1. A: Load Command “0100 0000”.
2. 2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. 3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. 4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. 5. Set BS2 to “0”. This selects low data byte.

Figure 19-5. Programming the FUSES Waveforms



19.6.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 145 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

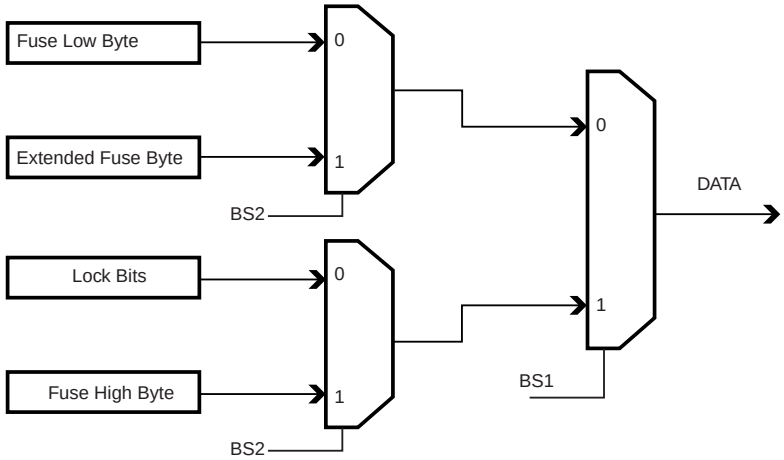
The Lock bits can only be cleared by executing Chip Erase.

19.6.12 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 145 for details on Command loading):

1. A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
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 19-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



19.6.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to “Programming the Flash” on page 145 for details on Command and Address loading):

1. A: Load Command “0000 1000”.
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”.

19.6.14 Reading the Calibration Byte

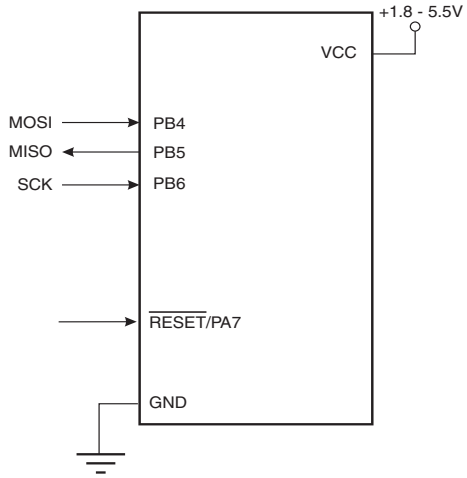
The algorithm for reading the Calibration byte is as follows (refer to “Programming the Flash” on page 145 for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

19.7 Serial Programming

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). See Figure 19-7 below.

Figure 19-7. Serial Programming and Verify



Note: If clocked by internal oscillator there is no need to connect a clock source to the CLKI pin.

After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed

Table 19-14. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB4	I	Serial Data in
MISO	PB5	O	Serial Data out
SCK	PB6	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

19.7.1 Serial Programming Algorithm

When writing serial data to the ATtiny43U, data is clocked on the rising edge of SCK.

When reading data from the ATtiny43U, data is clocked on the falling edge of SCK. See Figure 20-6 and Figure 20-7 for timing details.

To program and verify the ATtiny43U in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 19-16):

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 3 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 19-15 on page 153.) 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 19-15 on page 153.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The Memory page is loaded one byte at a time by supplying the 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 4 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 19-15 on page 153). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.

8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

Table 19-15. 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}	4.0 ms
t_{WD_FUSE}	4.5 ms

19.7.2 Serial Programming Instruction set

Table 19-16 on page 153 and Figure 19-8 on page 154 describes the Instruction set.

Table 19-16. Serial Programming Instruction Set

Instruction/Operation[1]	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming Enable	$AC	$53	$00	$00
Chip Erase (Program Memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{\text{BSY}}$	$F0	$00	$00	data byte out
Load Instructions				
Load Extended Address byte	$4D	$00	Extended adr	$00
Load Program Memory Page, High byte	$48	adr MSB	adr LSB	high data byte in
Load Program Memory Page, Low byte	$40	adr MSB	adr LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	$00	adr LSB	data byte in
Read Instructions				
Read Program Memory, High byte	$28	adr MSB	adr LSB	high data byte out
Read Program Memory, Low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM Memory	$A0	$00	adr LSB	data byte out
Read Lock bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	adr LSB	data byte out
Read Fuse bits	$50	$00	$00	data byte out
Read Fuse High bits	$58	$08	$00	data byte out
Read Extended Fuse Bits	$50	$08	$00	data byte out
Read Calibration Byte	$38	$00	$00	data byte out
Write Instructions				
Write Program Memory Page	$4C	adr MSB	adr LSB	$00
Write EEPROM Memory	$C0	$00	adr LSB	data byte in
Write EEPROM Memory Page (page access)	$C2	$00	adr LSB	$00
Write Lock bits	$AC	$E0	$00	data byte in

Table 19-16. Serial Programming Instruction Set (Continued)

Instruction/Operation[1]	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Write Fuse bits	$AC	$A0	$00	data byte in
Write Fuse High bits	$AC	$A8	$00	data byte in
Write Extended Fuse Bits	$AC	$A4	$00	data byte in

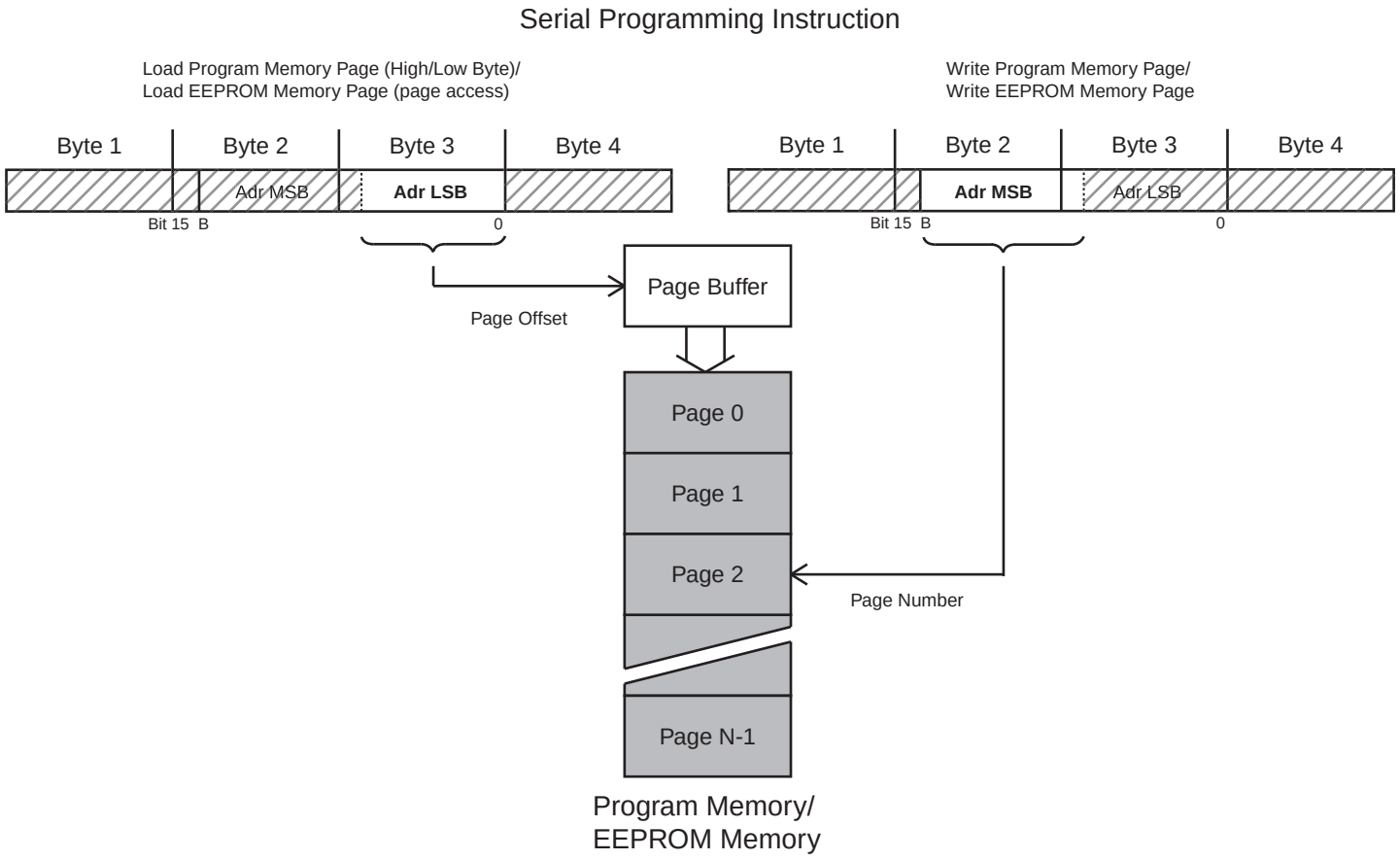
Notes:
1. Not all instructions are applicable for all parts.
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’) .
5. Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
6. See htt://www.atmel.com/avr 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 19-8 on page 154.

Figure 19-8. Serial Programming Instruction example



20. Electrical Characteristics

20.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*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.

20.2 DC Characteristics

Note: All DC Characteristics contained in this data sheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual silicon.

Table 20-1. DC Characteristics. T_A = -40°C to 85°C

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, Any Pin as I/O	V_{CC} = 1.8V - 2.4V	-0.5		0.2V_{CC} (1)	V
		V_{CC} = 2.4V - 5.5V	-0.5		0.3V_{CC} (1)	V
	Input Low Voltage, $\overline{\text{RESET}}$ Pin as Reset (2)	V_{CC} = 1.8V - 5.5V	-0.5		0.2V_{CC} (1)	V
V_{IH}	Input High-voltage, Any Pin as I/O	V_{CC} = 1.8V - 2.4V	0.7V_{CC} (3)		V_{CC} +0.5	V
		V_{CC} = 2.4V - 5.5V	0.6V_{CC} (3)		V_{CC} +0.5	V
	Input High-voltage, $\overline{\text{RESET}}$ Pin as Reset (2)	V_{CC} = 1.8V to 5.5V	0.9 V_{CC} (3)		V_{CC} +0.5	V
V_{OL}	Output Low Voltage (4), Pins PB1, PB2, PB4 and PB5 (5)	I_{OL} = 20 mA, V_{CC} = 5V			0.8	V
		I_{OL} = 10 mA, V_{CC} = 3V			0.8	V
		I_{OL} = 4 mA, V_{CC} = 1.8V			0.8	V
	Output Low Voltage (4), All Other I/O Pins, except $\overline{\text{RESET}}$ pin	I_{OL} = 10 mA, V_{CC} = 5V			0.8	V
		I_{OL} = 5 mA, V_{CC} = 3V			0.8	V
		I_{OL} = 2 mA, V_{CC} = 1.8V			0.8	V
V_{OH}	Output High-voltage (6) All I/O Pins, except $\overline{\text{RESET}}$ pin	I_{OH} = 10 mA, V_{CC} = 5V	4.0			V
		I_{OH} = 5 mA, V_{CC} = 3V	2.3			V
		I_{OH} = 2 mA, V_{CC} = 1.8V	1.4			V
I_{LIL}	Input Leakage Current, I/O Pin	V_{CC} = 5.5V, pin low	-1		1	µA
I_{LIH}	Input Leakage Current, I/O Pin	V_{CC} = 5.5V, pin high	-1		1	µA

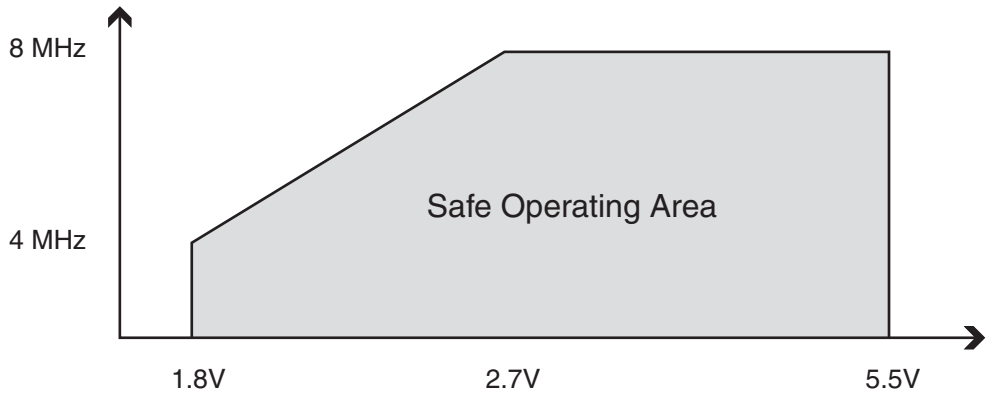
Table 20-1. DC Characteristics. T_A = -40°C to 85°C (Continued)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
R_{PU}	Pull-up Resistor, I/O Pin	V_{CC} = 5.5V, input low	20		50	kΩ
	Pull-up Resistor, $\overline{\text{RESET}}$ Pin	V_{CC} = 5.5V, input low	30		80	kΩ
I_{CC}	Supply Current, Active Mode, Without Boost Converter (7)	f = 1MHz, V_{CC} = 2V		0.2	0.55	mA
		f = 4MHz, V_{CC} = 3V		1.3	2.5	mA
		f = 8MHz, V_{CC} = 5V		4	7	mA
	Supply Current, Idle Mode, Without Boost Converter (7)	f = 1MHz, V_{CC} = 2V		0.04	0.15	mA
		f = 4MHz, V_{CC} = 3V		0.25	0.6	mA
		f = 8MHz, V_{CC} = 5V		1.0	2.0	mA
	Supply Current, Power-Down Mode, Without Boost Converter (7)	WDT enabled, V_{CC} = 3V		4.5	10	µA
		WDT disabled, V_{CC} = 3V		0.35	2	µA

Notes:
1. "Max" means the highest value where the pin is guaranteed to be read as low.
2. Not tested in production.
3. "Min" means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can under steady state (non-transient) conditions sink more than indicated, the sum of all I_{OL} currents may not exceed 60 mA, or the boost converter limits. If I_{OL} limits are exceeded the corresponding V_{OL} levels can not be guaranteed. Pins are not guaranteed to sink currents greater than those listed.
5. Pins PB1, PB2, PB4 and PB5 are high sink I/O pins.
6. Although each I/O port can under steady state (non-transient) conditions source more than indicated, the sum of all I_{OH} currents may not exceed 60 mA, or the boost converter limits. If I_{OH} limits are exceeded the corresponding V_{OH} levels can not be guaranteed. Pins are not guaranteed to source currents greater than those listed.
7. See "Boost Converter Characteristics" on page 159 for current consumption of entire device, including boost converter.

20.3 Speed Grades

Figure 20-1. Maximum Frequency vs. V_{CC} (Boost Converter Disregarded).



20.4 Clock Characteristics

20.4.1 Calibrated Internal Oscillator Accuracy

It is possible to manually calibrate the internal oscillator to be more accurate than default factory calibration. Please note that the oscillator frequency depends on temperature and voltage.

Table 20-2. 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: 1.8V - 5.5V	Fixed temperature within: -40°C - 85°C	±1%

Notes: 1. Accuracy of oscillator frequency at calibration point (fixed temperature and fixed voltage).

20.4.2 External Clock Drive

Figure 20-2. External Clock Drive Waveforms

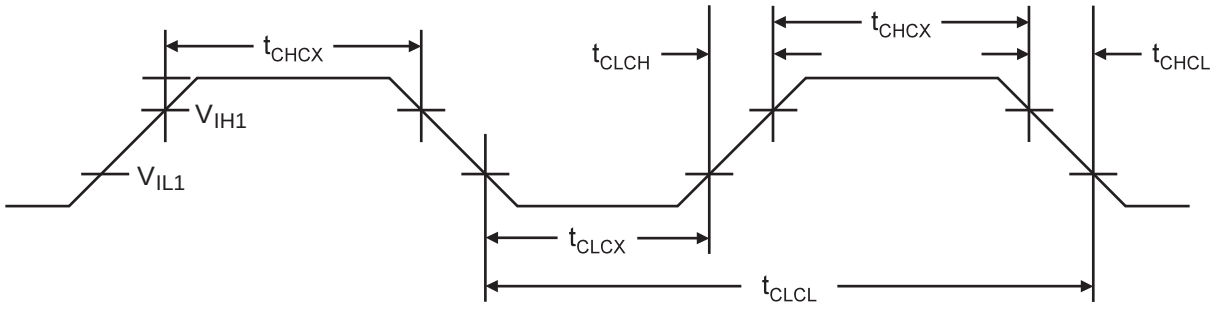


Table 20-3. External Clock Drive Characteristics

Symbol	Parameter	V_{CC} = 1.8 - 5.5V		V_{CC} = 2.7 - 5.5V		Units
		Min.	Max.	Min.	Max.	
$1/t_{CLCL}$	Clock Frequency	0	4	0	8	MHz
t_{CLCL}	Clock Period	250		125		ns
t_{CHCX}	High Time	100		50		ns
t_{CLCX}	Low Time	100		50		ns
t_{CLCH}	Rise Time		2.0		1.6	µs
t_{CHCL}	Fall Time		2.0		1.6	µs
Δt_{CLCL}	Period change from clock cycle to next		2		2	%

20.5 System and Reset Characteristics

Table 20-4. Reset, Brown-Out and Internal Voltage Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)	T_A = -40 - 85°C	1.1	1.4	1.6	V
	Power-on Reset Threshold Voltage (falling)[1]	T_A = -40 - 85°C	0.6	1.3	1.6	V
V_{PSR}	Power-On Slope Rate	T_A = -40 - 85°C	0.01			V/ms
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold		0.2 V_{CC}		0.9 V_{CC}	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin	V_{CC} = 1.8V V_{CC} = 3V V_{CC} = 5V		2000 700 400		ns
V_{HYST}	Brown-out Detector Hysteresis			50		mV
t_{BOD}	Min Pulse Width on Brown-out Reset			2		µs
V_{BG}	Internal bandgap reference voltage	V_{CC} = 2.7V T_A = 25°C	1.0	1.1	1.2	V
t_{BG}	Internal bandgap reference start-up time	V_{CC} = 2.7V T_A = 25°C		40	70	µs
I_{BG}	Internal bandgap reference current consumption	V_{CC} = 2.7V, T_A = 25°C		15		µA

Note: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

Table 20-5. BODLEVEL Fuse Coding[1]

BODLEVEL [2..0] Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.1	4.3	4.5	
011	Reserved			
010				
001				
000				

Note: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to V_{CC} = V_{BOT} during the production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed.

20.6 External Interrupt Characteristics

Table 20-6. Characteristics of Asynchronous External Interrupt

Symbol	Parameter	Condition	Min	Typ	Max	Unit
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

20.7 Boost Converter Characteristics

Table 20-7. Characteristics of Boost Converter. T = -20°C ... +85°C, unless otherwise noted

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{START}	Start Voltage		1.05	1.2	1.35	V
V_{BAT}	Input Voltage		0.7		1.8	V
V_{STOP}	Shutdown Voltage		0.45		0.7	V
V_{BOOST}				1.0		V
V_{CC}	Output Voltage	Active Mode, I_{LOAD} > 1mA	2.7	3.0	3.3	V
		Low Current Mode	1.8		3.6	V
I_{LOAD}	Load Current	V_{BAT} = 0.7V	0		10	mA
		V_{BAT} = 1.0V	0		30	mA
V_{RPP}	Output Voltage Ripple	V_{BAT} = 1.0V, I_{LOAD} = 30mA, C_{LOAD} = 22µF		100		mV
t_{START}	Start-up Time	V_{BAT} = Step Change from 0V to 1.2V		2		ms
I_{IN}	Current consumption of entire device	V_{BAT} = 0.45V, Converter in Stop Mode MCU not powered		1		µA
		V_{BAT} = 1.05V, Converter in Stop Mode MCU not powered		2		µA
		V_{BAT} = 1.2V, Active Low Current Mode MCU in Power Down, WD disabled		5		µA
		V_{BAT} = 1.2V, Active Regulated Mode MCU active, 4MHz		5		mA
f_S	Switching frequency		75	100	125	kHz
T_S	Switching period	$T_S = 1 / f_S$	8	10	13.3	µs
D_{SW}	Duty Cycle		3		72	%
V_{BATOL}	Lowest V_{BAT} voltage where device recovers from overload	Output overload or short circuit removed	TBD	1.05	TBD	V
	Load Regulation: $\frac{V_{CC}\|_{LOADMIN} - V_{CC}\|_{LOADMAX}}{V_{CC}\|_{LOADMAX}}$	T = 25°C, V_{BAT} = 1.2V Boost converter in Active Mode		2.7		%
	Line Regulation: $\frac{V_{CC}\|_{VBATMAX} - V_{CC}\|_{VBATMIN}}{V_{CC}\|_{VBATNOM}}$	T = 25°C, I_{LOAD} = 1mA Boost converter in Active Mode		1.0		%
	Temperature Regulation: $\frac{V_{CC}\|_{TEMPMAX} - V_{CC}\|_{TEMPMIN}}{V_{CC}\|_{TEMPNOM}}$	V_{BAT} = 1.2V, I_{LOAD} = 1mA Boost converter in Active Mode		2.3		%

20.8 ADC Characteristics – Preliminary Data

Table 20-8. ADC Characteristics, Single_Ended Conversion, T_A = -40°C - 85°C, Boost Converter Disabled.

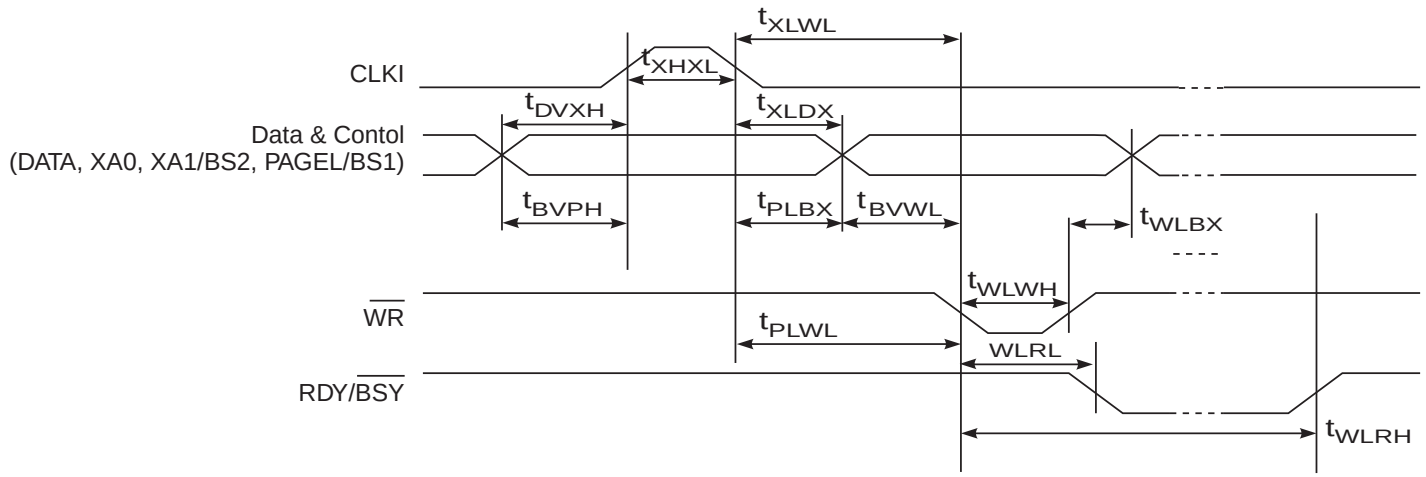
Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution				10	Bits
	Absolute accuracy (Including INL, DNL, and quantization, Gain and Offset Errors)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		4.0	TBD	LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1 MHz		5.0	TBD	LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz Noise Reduction Mode		3.0	TBD	LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1 MHz Noise Reduction Mode		4.0	TBD	LSB
	Integral Non-linearity (INL) (Accuracy after Offset and Gain Calibration)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		1.0	TBD	LSB
	Differential Non-linearity (DNL)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		0.5	TBD	LSB
	Gain Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		5.0	TBD	LSB
	Offset Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		-3.0	TBD	LSB
	Conversion Time	Free Running Conversion	14		280	µs
	Clock Frequency		50		1000	kHz
V_{IN}	Input Voltage		GND		V_{REF}	V
	Input Bandwidth			38.4		kHz
V_{INT}	Internal Voltage Reference		1.0	1.1	1.2	V
R_{AIN}	Analog Input Resistance			100		MΩ

Table 20-9. ADC Characteristics, Single_Ended Conversion, T_A = -40°C - 85°C, Boost Converter Enabled.

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution				10	Bits
	Absolute accuracy (Including INL, DNL, and quantization, Gain and Offset Errors)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		TBD	TBD	LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1 MHz		TBD	TBD	LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz Noise Reduction Mode		TBD	TBD	LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1 MHz Noise Reduction Mode		TBD	TBD	LSB
	Integral Non-linearity (INL) (Accuracy after Offset and Gain Calibration)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		TBD	TBD	LSB
	Differential Non-linearity (DNL)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		TBD	TBD	LSB
	Gain Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		TBD	TBD	LSB
	Offset Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		TBD	TBD	LSB
	Conversion Time	Free Running Conversion	14		280	µs
	Clock Frequency		50		1000	kHz
V_{IN}	Input Voltage		GND		V_{REF}	V
	Input Bandwidth			38.4		kHz
V_{INT}	Internal Voltage Reference		1.0	1.1	1.2	V
R_{AIN}	Analog Input Resistance			100		MΩ

20.9 Parallel Programming Characteristics

Figure 20-3. Parallel Programming Timing, Including some General Timing Requirements



Note: The timing requirements in Figure 20-3 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Figure 20-4. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements

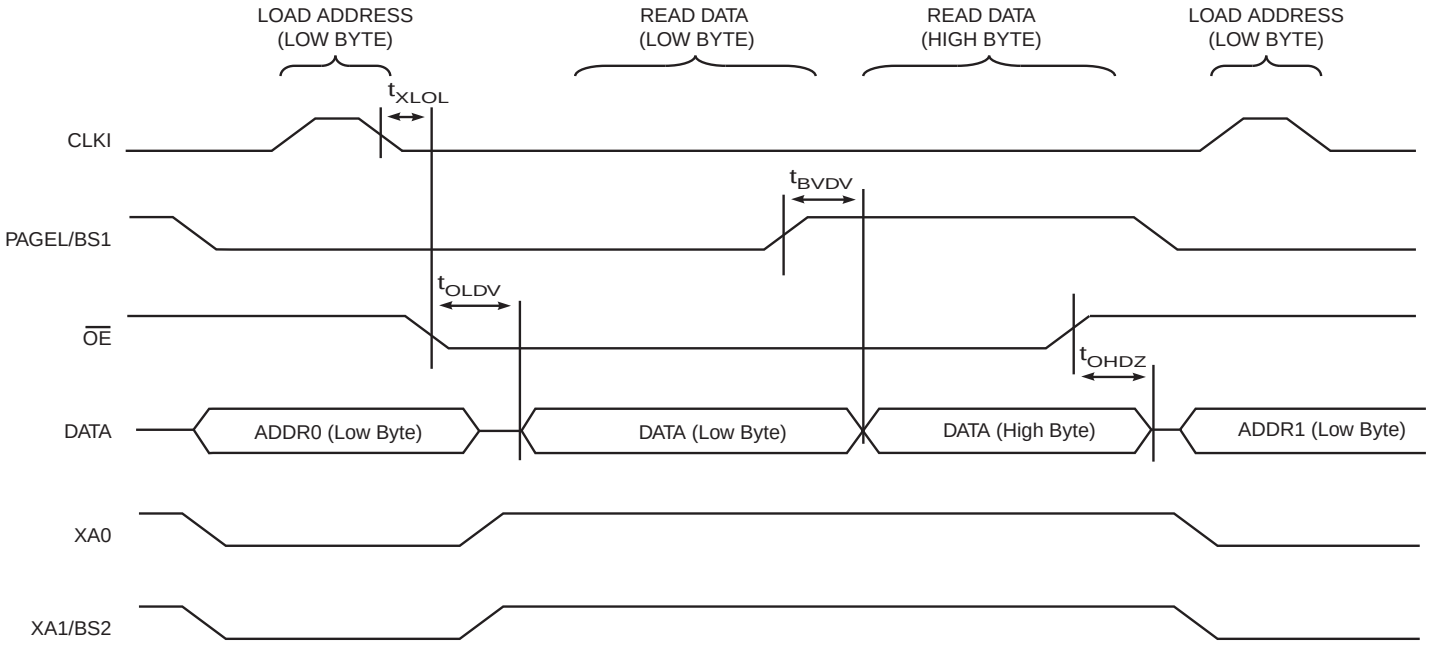


Figure 20-5. Parallel Programming Timing, Loading Sequence with Timing Requirements

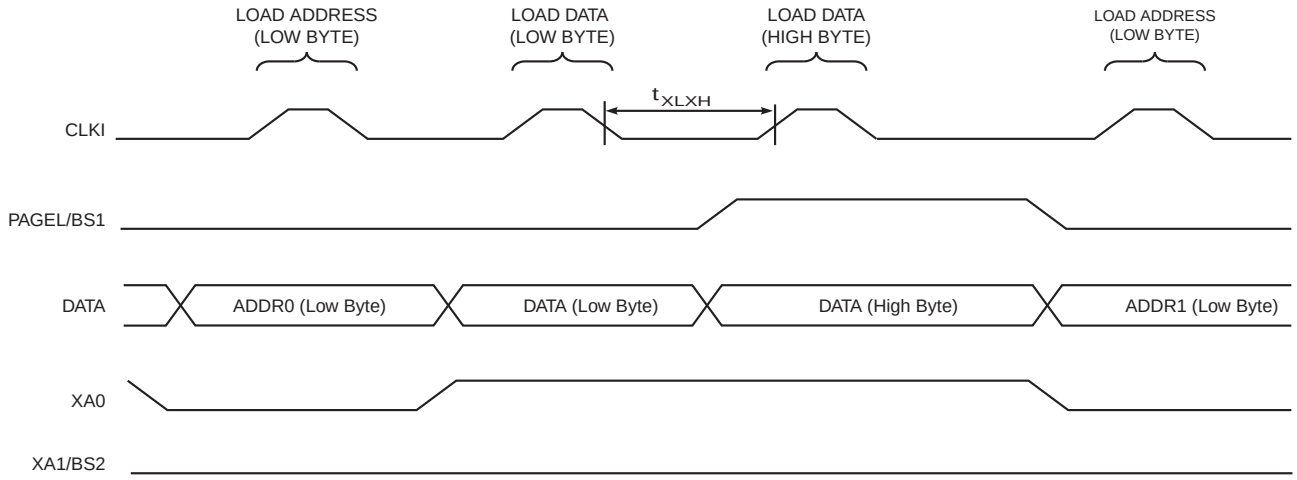


Table 20-10. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

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 CLKI High	67			ns
t_{XLXH}	CLKI Low to CLKI High	200			ns
t_{XHXL}	CLKI Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after CLKI Low	67			ns
t_{XLWL}	CLKI Low to $\overline{WR}$ Low	0			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			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}	CLKI 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

Note: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.
2. t_{WLRH_CE} is valid for the Chip Erase command.

20.10 Serial Programming Characteristics

Figure 20-6. Serial Programming Timing

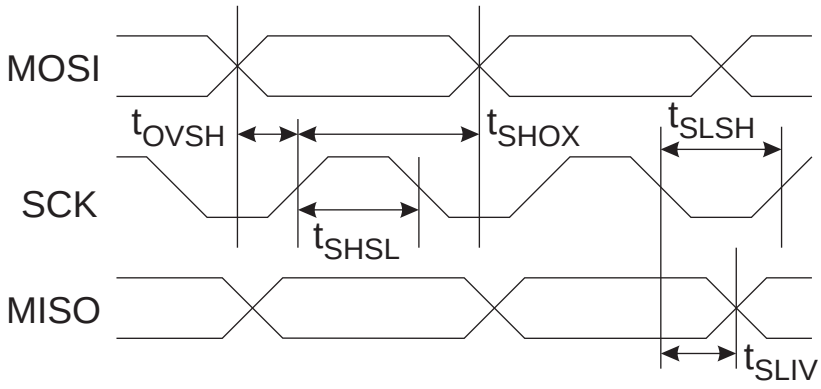


Figure 20-7. Serial Programming Waveforms

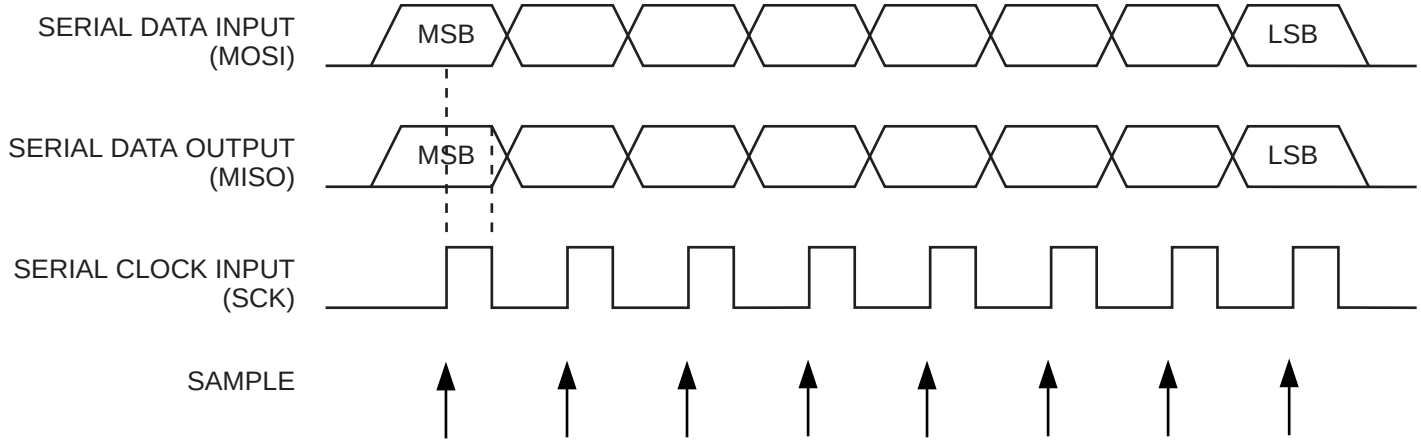


Table 20-11. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 1.8 - 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0		4	MHz
t_{CLCL}	Oscillator Period	250			ns
$1/t_{CLCL}$	Oscillator Frequency (V_{CC} = 4.5V - 5.5V)	0		8	MHz
t_{CLCL}	Oscillator Period (V_{CC} = 4.5V - 5.5V)	125			ns
t_{SHSL}	SCK Pulse Width High	2 t_{CLCL}			ns
t_{SLSH}	SCK Pulse Width Low	2 t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	TBD	TBD	TBD	ns

21. Typical Characteristics – TBD

The data contained in this section is largely based on simulations and characterization of similar devices in the same process and design methods. Thus, the data should be treated as indications of how the part will behave.

The following charts show typical behavior. These figures are not tested during manufacturing. During characterisation devices are operated at frequencies higher than test limits but they are not guaranteed to function properly at frequencies higher than the ordering code indicates.

All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. Current consumption is a function of several factors such as operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

A sine wave generator with rail-to-rail output is used as clock source but current consumption in Power-Down mode is independent of clock selection. The difference between current consumption in Power-Down mode with Watchdog Timer enabled and Power-Down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

The current drawn from pins with a capacitive load may be estimated (for one pin) as follows:

$I_{CP} \approx V_{CC} \times C_L \times f_{SW}$

where V_{CC} = operating voltage, C_L = load capacitance and f_{SW} = average switching frequency of I/O pin.

21.1 Boost Converter

Figure 21-1. Typical Transition Range Between Active Modes of Operation

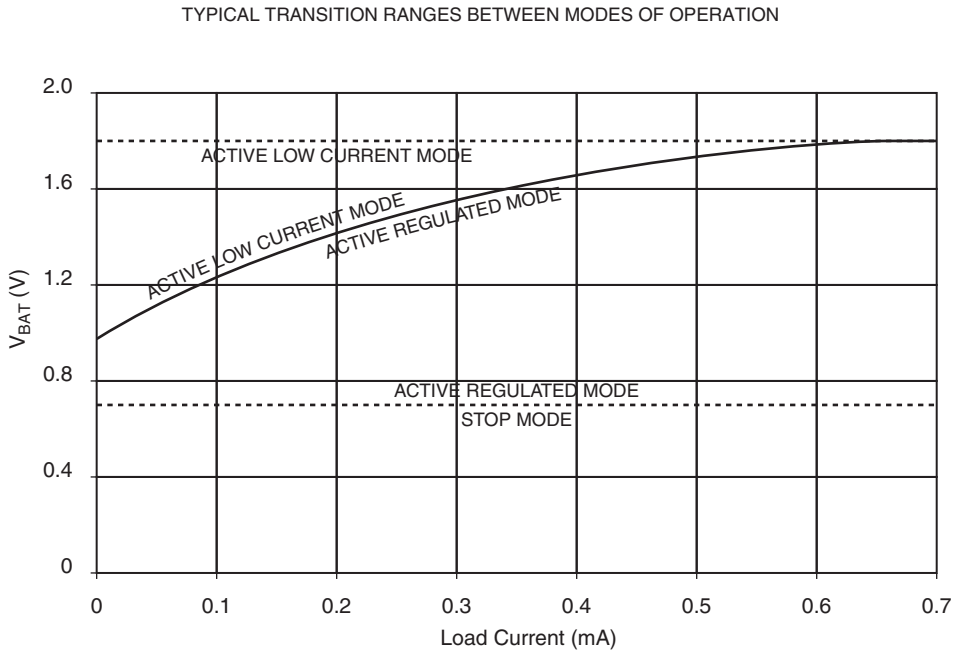


Figure 21-2. Boost Converter Efficiency vs. Load Current and V_{BAT} Voltage

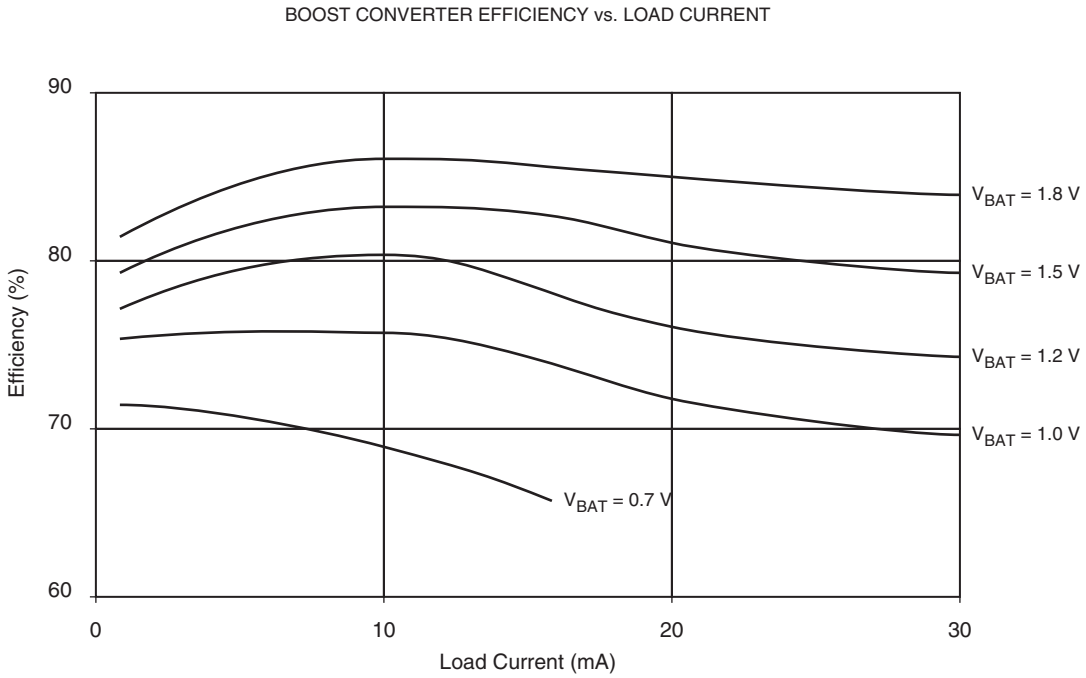
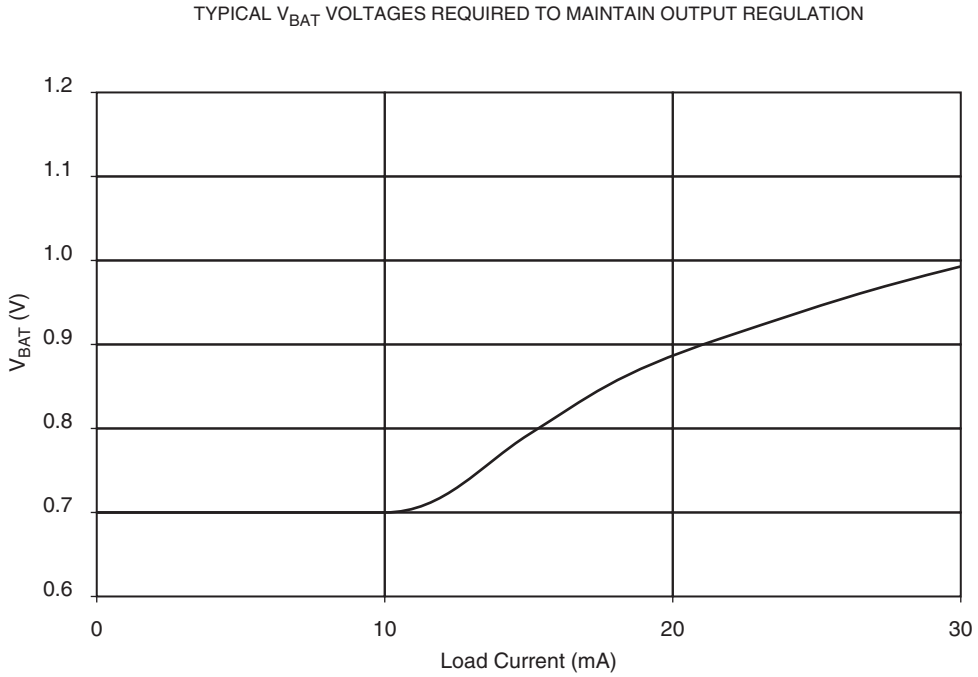


Figure 21-3. Input Voltage Required to Maintain Regulation vs. Load Current



22. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	Page 8
0x3E (0x5E)	SPH	–	–	–	–	–	–	–	SP8	Page 12
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	Page 12
0x3C (0x5C)	OCR0B	Timer/Counter0 – Output Compare Register B								Page 95
0x3B (0x5B)	GIMSK	–	INT0	PCIE1	PCIE0	–	–	–	–	Page 60
0x3A (0x5A)	GIFR	–	INTF0	PCIF1	PCIF0	–	–	–	–	Page 60
0x39 (0x59)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	Page 95
0x38 (0x58)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	Page 96
0x37 (0x57)	SPMCSR	–	–	–	CTPB	RFLB	PGWRT	PGERS	SPMEN	Page 137
0x36 (0x56)	OCR0A	Timer/Counter0 – Output Compare Register A								Page 95
0x35 (0x55)	MCUCR	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	Page 34, Page 59, Page 78
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	Page 54
0x33 (0x53)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	Page 93
0x32 (0x52)	TCNT0	Timer/Counter0								Page 94
0x31 (0x51)	OSCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	Page 28
0x30 (0x50)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–		WGM01	WGM00	Page 90
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–		WGM11	WGM10	Page 90
0x2E (0x4E)	TCCR1B	FOC1A	FOC1B	–	–	WGM12	CS12	CS11	CS10	Page 93
0x2D (0x4D)	TCNT1	Timer/Counter1								Page 95
0x2C (0x4C)	OCR1A	Timer/Counter1 – Output Compare Register A								Page 95
0x2B (0x4B)	OCR1B	Timer/Counter1 – Output Compare Register B								Page 95
0x2A (0x4A)	Reserved	–								
0x29 (0x49)	Reserved	–								
0x28 (0x48)	Reserved	–								
0x27 (0x47)	DWDR	DWDR[7:0]								Page 132
0x26 (0x46)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	Page 28
0x25 (0x45)	Reserved	–								
0x24 (0x44)	Reserved	–								
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	–	PSR10	Page 99
0x22 (0x42)	Reserved	–								
0x21 (0x41)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	Page 54
0x20 (0x40)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	Page 61
0x1F (0x3F)	Reserved	–								
0x1E (0x3E)	EEAR	–	–	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	Page 20
0x1D (0x3D)	EEDR	EEPROM Data Register								Page 21
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	Page 21
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	Page 78
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	Page 78
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	Page 78
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	Page 78
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	Page 78
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	Page 78
0x15 (0x35)	GPIOR2	General Purpose I/O Register 2								Page 22
0x14 (0x34)	GPIOR1	General Purpose I/O Register 1								Page 22
0x13 (0x33)	GPIOR0	General Purpose I/O Register 0								Page 22
0x12 (0x32)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	Page 61
0x11 (0x31)	Reserved	–								
0x10 (0x30)	USIBR	USI Buffer Register								Page 111
0x0F (0x2F)	USIDR	USI Data Register								Page 112
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	Page 112
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	Page 112
0x0C (0x2C)	TIMSK1	–	–	–	–	–	OCIE1B	OCIE1A	TOIE1	Page 96
0x0B (0x2B)	TIFR1	–	–	–	–	–	OCF1B	OCF1A	TOV1	Page 96
0x0A (0x2A)	Reserved	–								
0x09 (0x29)	Reserved	–								
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	–	ACIS1	ACIS0	Page 113
0x07 (0x27)	ADMUX	–	REFS	–	–	–	MUX2	MUX1	MUX0	Page 126
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	Page 127
0x05 (0x25)	ADCH	ADC Data Register High Byte								Page 128
0x04 (0x24)	ADCL	ADC Data Register Low Byte								Page 128
0x03 (0x23)	ADCSRB	BS	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	Pages 47, 113, 129
0x02 (0x22)	Reserved	–								
0x01 (0x21)	DIDR0			AIN1D	AIN0D	ADC3D	ADC2D	ADC1D	ADC0D	Page 114, Page 130
0x00 (0x20)	PRR	PRE2	PRE1	PRE0	–	PRTIM1	PRTIM0	PRUSI	PRADC	Page 35

Note:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. 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.
3. 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 operation 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.

23. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 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
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
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
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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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

24. Ordering Information

24.1 ATtiny43U

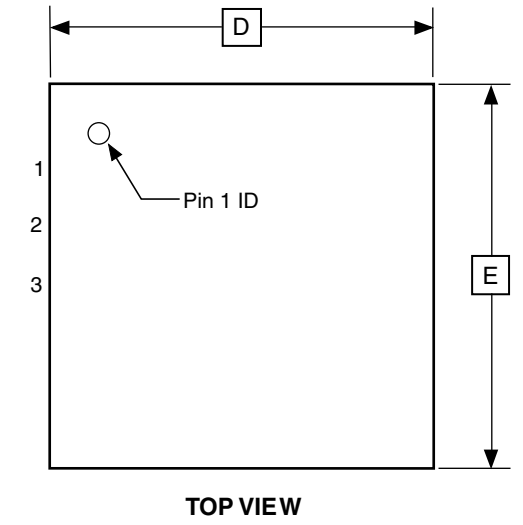
Speed (MHz)	Power Supply	Ordering Code [1]	Package [2]	Operational Range
8	1.8 - 5.5V [3]	ATtiny43U-MU ATtiny43U-SU	20M1 20S2	Industrial (-40°C to 85°C)

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Supply voltage on V_{CC} pin, boost converter disregarded. When boost converter is active the device can be operated from voltages sources lower than indicated here. See table "Characteristics of Boost Converter. T = -20°C ... +85°C, unless otherwise noted" on page 159 for more information.

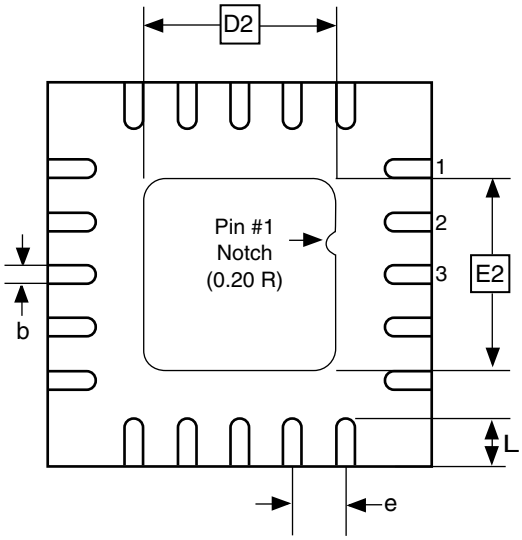
Package Type	
20M1	20-pad, 4 x 4 x 0.8 mm Body, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)
20S2	20-lead, 0.300" Wide Body, Plastic Gull Wing Small Outline Package (SOIC)

25. Packaging Information

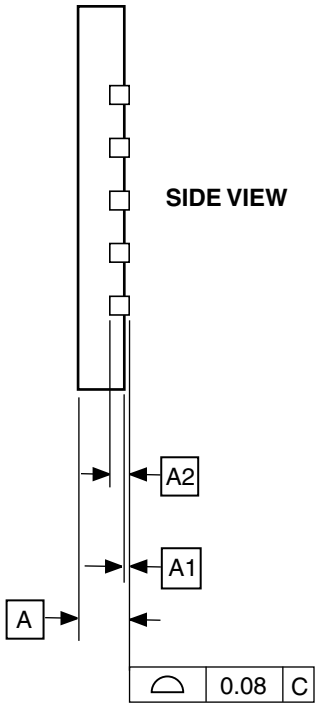
25.1 20M1



TOP VIEW



BOTTOM VIEW



Note: Reference JEDEC Standard MO-220, Fig. 1 (SAW Singulation) WGGD-5.

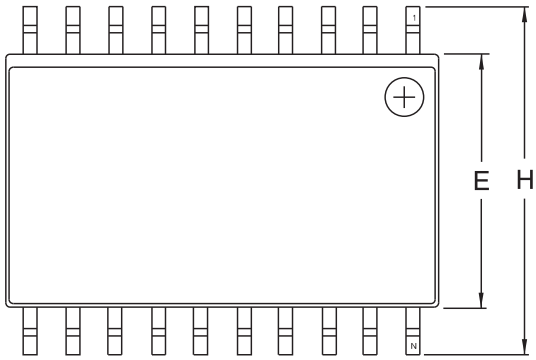
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.70	0.75	0.80	
A1	–	0.01	0.05	
A2		0.20 REF		
b	0.18	0.23	0.30	
D		4.00 BSC		
D2	2.45	2.60	2.75	
E		4.00 BSC		
E2	2.45	2.60	2.75	
e		0.50 BSC		
L	0.35	0.40	0.55	

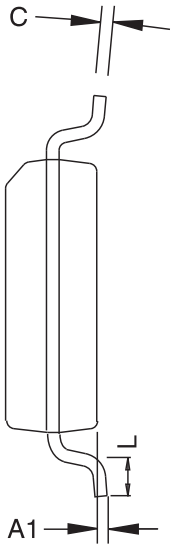
10/27/04

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**20M1**, 20-pad, 4 x 4 x 0.8 mm Body, Lead Pitch 0.50 mm, 2.6 mm Exposed Pad, Micro Lead Frame Package (MLF)	20M1	A

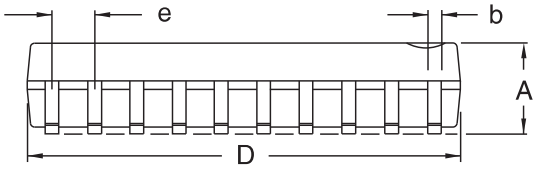
25.2 20S2



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure – mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	2.35		2.65	
A1	0.10		0.30	
b	0.33		0.51	4
C	0.23		0.32	
D	12.60		13.00	1
E	7.40		7.60	2
H	10.00		10.65	
L	0.40		1.27	3
e	1.27 BSC			

Notes.
1. This drawing is for general information only; refer to JEDEC Drawing MS-013, Variation AC for additional information.
2. Dimension 'D' does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shall not exceed 0.15 mm (0.006') per side.
3. Dimension 'E' does not include inter-lead Flash or protrusion. Inter-lead Flash and protrusions shall not exceed 0.25 mm (0.010') per side.
4. 'L' is the length of the terminal for soldering to a substrate.
5. The lead width 'b', as measured 0.36 mm (0.014') or greater above the seating plane, shall not exceed a maximum value of 0.61 mm (0.024') per side.

11/6/06

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**20S2**, 20-lead, 0.300' Wide Body, Plastic Gull Wing Small Outline Package (SOIC)	20S2	B

26. Errata

The revision letter in this section refers to the revision of the ATtiny43U device.

26.1 ATtiny43U

26.1.1 Rev. C

- **Increased Probability of Boost Converter Entering Active Low Current Mode**

1. **Increased Probability of Boost Converter Entering Active Low Current Mode**

   The boost converter may enter and stay in Active Low Current Mode at supply voltages and load currents higher than those specified. This is due to high switching currents in bonding wires of the SOIC package. Devices packaged in MLF are not affected.

   Problem Fix / Workaround

   Add a 1.5nF capacitor between pins LSW and GND of the SOIC package. Also, increase the value of the by-pass capacitor between pins V_{CC} and GND to at least 30µF.

   Alternatively, use the device in MLF, without modifications.

26.1.2 Rev. B

Not sampled.

26.1.3 Rev. A

Not sampled.

27. Datasheet Revision History

27.1 Rev. 8048B-03/09

1. Updated Data retention bullet in “Features” on page 1.

27.2 Rev. 8048A-02/09

1. Initial revision.

Table of Contents

Features *1*

1 Pin Configurations *2*

1.1 Pin Descriptions 2

2 Overview *4*

3 About *6*

3.1 Resources 6

3.2 Code Examples 6

3.3 Data Retention 6

3.4 Disclaimer 6

4 AVR CPU Core *7*

4.1 Introduction 7

4.2 Architectural Overview 7

4.3 ALU – Arithmetic Logic Unit 8

4.4 Status Register 8

4.5 General Purpose Register File 10

4.6 Stack Pointer 11

4.7 Instruction Execution Timing 12

4.8 Reset and Interrupt Handling 13

5 Memories *15*

5.1 Overview 15

5.2 In-System Re-programmable Flash Program Memory 15

5.3 SRAM Data Memory 15

5.4 EEPROM Data Memory 16

5.5 I/O Memory 20

5.6 Register Description 20

6 System Clock and Clock Options *23*

6.1 Clock Systems and their Distribution 23

6.2 Clock Sources 24

6.3 System Clock Prescaler 27

6.4 Clock Output Buffer 28

6.5 Register Description 28

7 Power Management and Sleep Modes *31*

7.1 Sleep Modes 31
7.2 Software BOD Disable 32
7.3 Power Reduction Register 32
7.4 Minimizing Power Consumption 33
7.5 Register Description 34

8 Power Supply and On-Chip Boost Converter 36

8.1 Overview 36
8.2 Modes of Operation 37
8.3 Output Voltage versus Load Current 40
8.4 Overload Behaviour 42
8.5 Software Control of Boost Converter 42
8.6 Component Selection 43
8.7 Typical Applications 46
8.8 Characteristics 46
8.9 Potential Limitations 46
8.10 Bypassing the Boost Converter 46
8.11 Register Description 47

9 System Control and Reset 48

9.1 Resetting the AVR 48
9.2 Reset Sources 49
9.3 Power-on Reset 49
9.4 External Reset 50
9.5 Brown-out Detection 50
9.6 Watchdog Reset 51
9.7 Internal Voltage Reference 51
9.8 Watchdog Timer 51
9.9 Register Description 54

10 Interrupts 57

10.1 Interrupt Vectors 57
10.2 External Interrupts 58
10.3 Register Description 59

11 I/O Ports 62

11.1 Introduction 62
11.2 Ports as General Digital I/O 63
11.3 Alternate Port Functions 67

11.4 Register Description 78

12 8-bit Timer/Counter with PWM (Timer/Counter0 and Timer/Counter1) . 79

12.1 Features 79
12.2 Overview 79
12.3 Timer/Counter Clock Sources 80
12.4 Counter Unit 80
12.5 Output Compare Unit 81
12.6 Compare Match Output Unit 83
12.7 Modes of Operation 84
12.8 Timer/Counter Timing Diagrams 88
12.9 Register Description 90

13 Timer/Counter Prescaler 98

13.1 Prescaler Reset 98
13.2 External Clock Source 98
13.3 Register Description 99

14 USI – Universal Serial Interface 100

14.1 Features 100
14.2 Overview 100
14.3 Functional Descriptions 101
14.4 Alternative USI Usage 107
14.5 Register Descriptions 107

15 Analog Comparator 112

15.1 Analog Comparator Multiplexed Input 112
15.2 Register Description 113

16 Analog to Digital Converter 115

16.1 Features 115
16.2 Overview 115
16.3 ADC Operation 116
16.4 Starting a Conversion 116
16.5 Prescaling and Conversion Timing 117
16.6 Changing Channel or Reference Selection 120
16.7 ADC Noise Canceler 121
16.8 Analog Input Circuitry 122
16.9 Analog Noise Canceling Techniques 122

16.10 ADC Accuracy Definitions 123
16.11 ADC Conversion Result 125
16.12 Temperature Measurement 125
16.13 Register Description 126

17 debugWIRE On-chip Debug System 131

17.1 Features 131
17.2 Overview 131
17.3 Physical Interface 131
17.4 Software Break Points 132
17.5 Limitations of debugWIRE 132
17.6 Register Description 132

18 Self-Programming the Flash 133

18.1 Performing Page Erase by SPM 133
18.2 Filling the Temporary Buffer (Page Loading) 133
18.3 Performing a Page Write 134
18.4 Addressing the Flash During Self-Programming 134
18.5 EEPROM Write Prevents Writing to SPMCSR 135
18.6 Reading the Fuse and Lock Bits from Software 135
18.7 Preventing Flash Corruption 137
18.8 Programming Time for Flash when Using SPM 137
18.9 Register Description 137

19 Memory Programming 139

19.1 Program And Data Memory Lock Bits 139
19.2 Fuse Bytes 140
19.3 Device Signature Imprint Table 141
19.4 Page Size 142
19.5 Parallel Programming Parameters, Pin Mapping, and Commands 142
19.6 Parallel Programming 144
19.7 Serial Programming 151

20 Electrical Characteristics 155

20.1 Absolute Maximum Ratings* 155
20.2 DC Characteristics 155
20.3 Speed Grades 156
20.4 Clock Characteristics 157
20.5 System and Reset Characteristics 158

20.6 External Interrupt Characteristics 158
20.7 Boost Converter Characteristics 159
20.8 ADC Characteristics – Preliminary Data 160
20.9 Parallel Programming Characteristics 162
20.10 Serial Programming Characteristics 164

21 Typical Characteristics – TBD 165

21.1 Boost Converter 165

22 Register Summary 167

23 Instruction Set Summary 169

24 Ordering Information 171

24.1 ATtiny43U 171

25 Packaging Information 172

25.1 20M1 172
25.2 20S2 173

26 Errata 174

26.1 ATtiny43U 174

27 Datasheet Revision History 175

27.1 Rev. 8048B-03/09 175
27.2 Rev. 8048A-02/09 175

Table of Contents i

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2009 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.