Features

- Utilizes the AVR® RISC Architecture
- AVR - High-performance and Low-power RISC Architecture
  - 118 Powerful Instructions - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Up to 1MIPS Throughput at 1MHz
- Data and Nonvolatile Program Memory
  - 2K Bytes of In-System Programmable Flash
    Endurance 1,000 Write/Erase Cycles
  - 128 Bytes of internal SRAM
  - 128 Bytes of In-System Programmable EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - Programming Lock for Flash Program and EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler
  - Programmable Watchdog Timer with On-chip Oscillator
  - SPI Serial Interface for In-System Programming
- Special Microcontroller Features
  - Low-power Idle and Power Down Modes
  - External and Internal Interrupt Sources
  - Power-on Reset Circuit
  - On-chip RC Oscillator
- Specifications
  - Low-power, High-speed CMOS Process Technology
  - Fully Static Operation
- Power Consumption at 3V, 25°C
  - Active: 1.5 mA
  - Idle Mode: 100 µA
  - Power Down Mode: <1 µA
- I/O and Packages
  - 5 Programmable I/O Lines
  - 8-pin PDIP and SOIC
- Operating Voltages
  - 2.7 - 6.0V
- Speed Grade
  - Internal Oscillator ~1MHz @ 5.0V

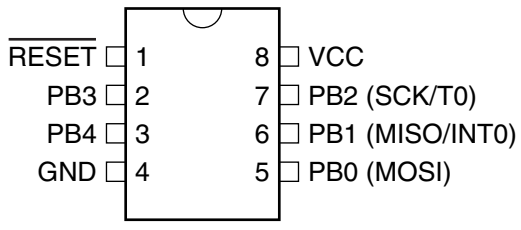
Description

The ATtiny22L is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny22L achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

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

Pin Configuration

PDIP/SOIC



8-bit AVR® Microcontroller with 2K Bytes of In-System Programmable Flash

ATtiny22L

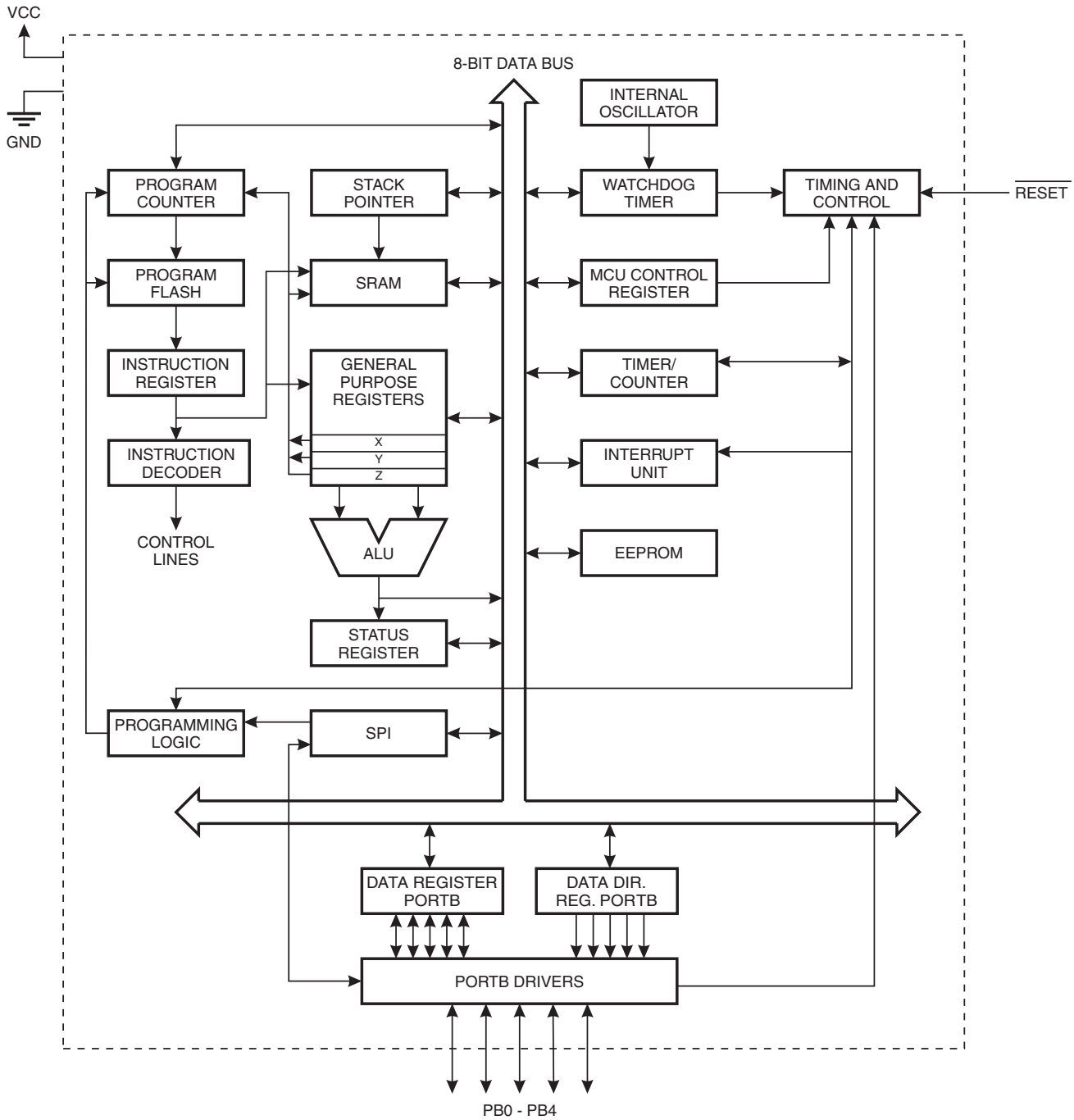
Preliminary

Rev. 1273B–02/00

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.

Block Diagram

Figure 1. The ATtiny22L Block Diagram



The ATtiny22L provides the following features: 2K bytes of In-System Programmable Flash, 128 bytes EEPROM, 128 bytes SRAM, five general purpose I/O lines, 32 general purpose working registers, an 8-bit timer/counter, internal and external interrupts, programmable Watchdog Timer with internal oscillator, an SPI serial port for Flash Memory downloading and two software selectable power saving modes. The Idle Mode stops the CPU while allowing the SRAM, timer/counters, SPI port and interrupt system to continue functioning. The power down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The device is manufactured using Atmel's high density nonvolatile memory technology. The on-chip Flash allows the program memory to be reprogrammed in-system through an SPI serial interface. By combining an 8-bit RISC CPU with ISP Flash on a monolithic chip, the Atmel ATtiny22L is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATtiny22L AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions ATtiny22L

VCC

Supply voltage pin.

GND

Ground pin.

Port B (PB4..PB0)

Port B is a 5-bit bi-directional I/O port with internal pull-up resistors. The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low, will source current if the pull-up resistors are activated.

Port B also serves the functions of various special features.

Port pins can provide internal pull-up resistors (selected for each bit). The port B pins are tri-stated when a reset condition becomes active.

$\overline{\text{RESET}}$

Reset input. An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 50 ns will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

Clock Source

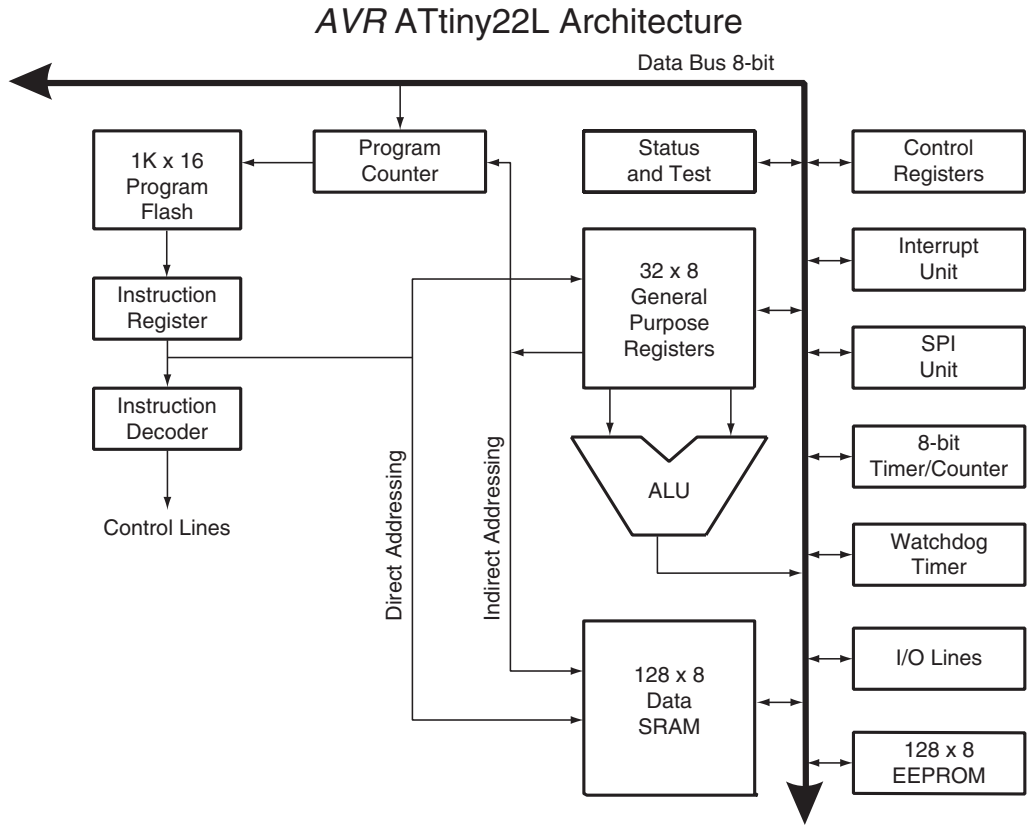
The ATtiny22L is clocked by an on-chip RC oscillator. This RC oscillator runs at a nominal frequency of 1 MHz (VCC = 5V).

Architectural Overview

The fast-access register file concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one arithmetic logic unit (ALU) operation is executed. 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 three address pointers is also used as the address pointer for the constant table look up function. These added function registers are the 16-bit X-register, Y-register and Z-register.

Figure 2. The ATtiny22L AVR RISC Architecture



The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 2 shows the ATtiny22L AVR RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the register file as well. This is enabled by the fact that the register file is assigned the 32 lowermost Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D-converters, 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, $20 - $5F.

The AVR has Harvard architecture - with separate memories and buses for program and data. The program memory is accessed with a two stage pipeline. 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 downloadable Flash memory.

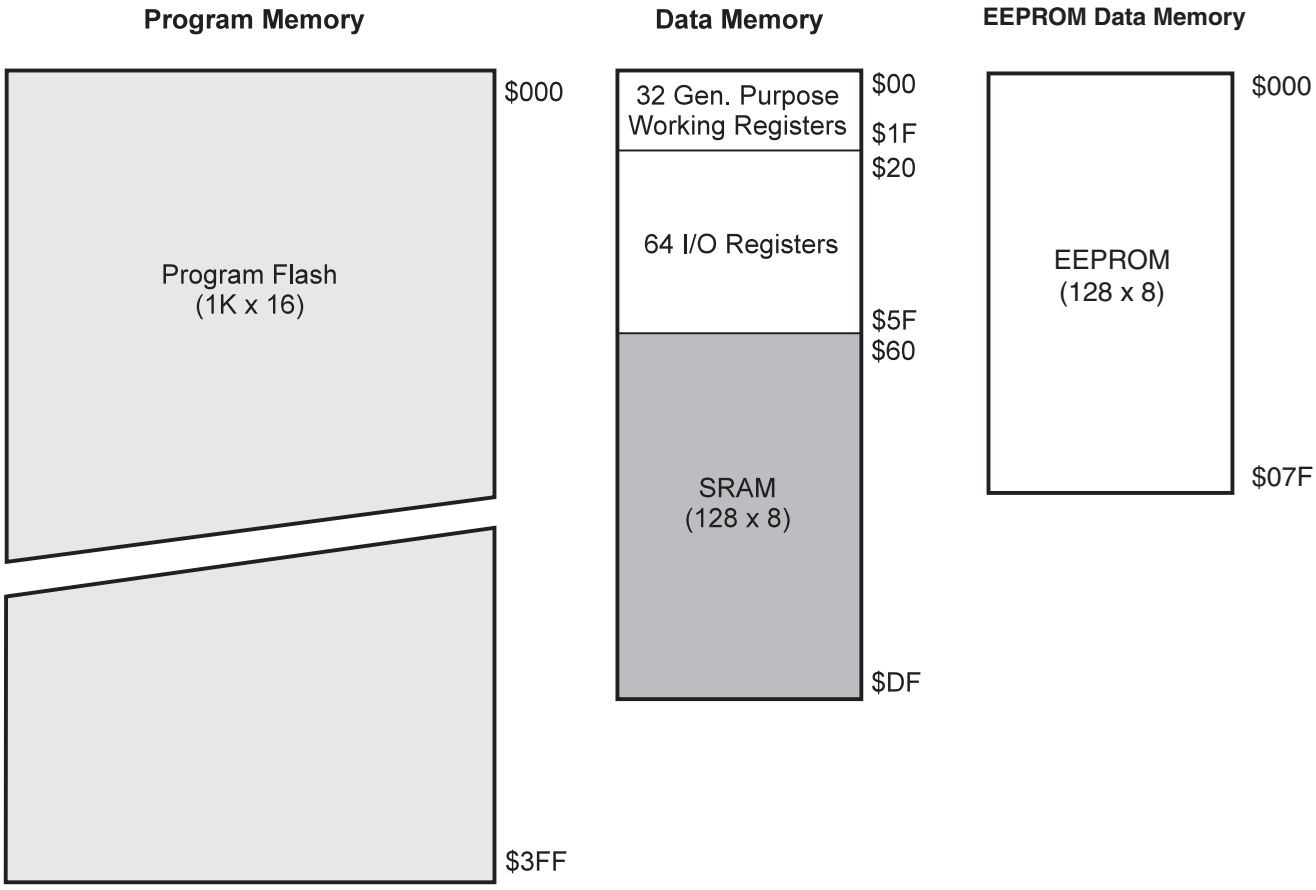
With the relative jump and call instructions, the whole 1K address space is directly accessed. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

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 8-bit stack pointer SP is read/write accessible in the I/O space.

The 128 bytes data SRAM + register file and I/O registers can be easily 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.

Figure 3. Memory Maps

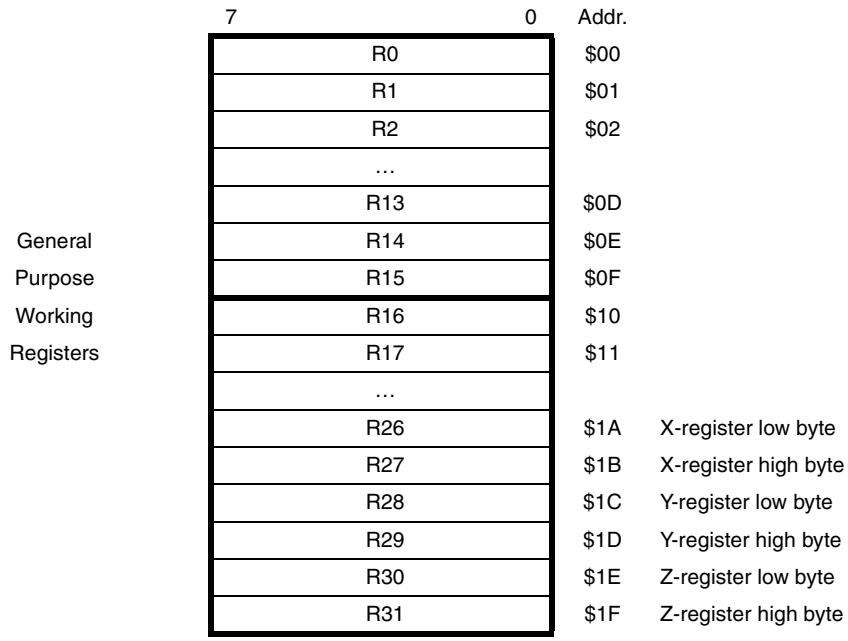


A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

General Purpose Register File

Figure 4 shows the structure of the 32 general purpose registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers



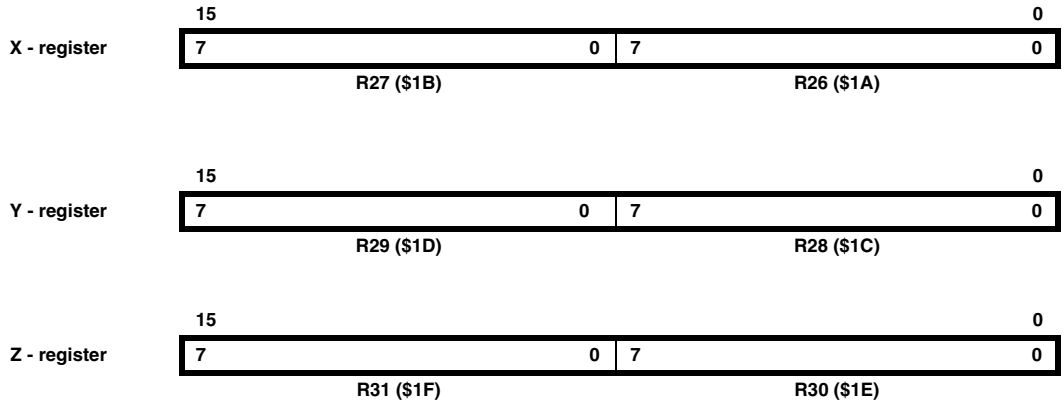
All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI, ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the register file - R16..R31. The general SBC, SUB, CP, AND, OR and all other operations between two registers or on a single register apply to the entire register file.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although the register file is not physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X, Y and Z registers can be set to index any register in the file.

X-Register, Y-Register, and Z-Register

The registers R26..R31 have some added functions to their general purpose usage. These registers are the address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y and Z are defined as:

Figure 5. The X, Y, and Z Registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

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, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories - arithmetic, logic and bit-functions

In-System Programmable Flash Program Memory

The ATtiny22L contains 2K bytes on-chip In-System Programmable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 1K x 16. The Flash memory has an endurance of at least 1000 write/erase cycles.

The ATtiny22L Program Counter PC is 10 bits wide, hence addressing the 1024 program memory addresses. See page 38 for a detailed description on Flash data programming.

Constant tables must be allocated within the address 0-2K (see the LPM - Load Program Memory instruction description).

See page 9 for the different addressing modes.

EEPROM Data Memory

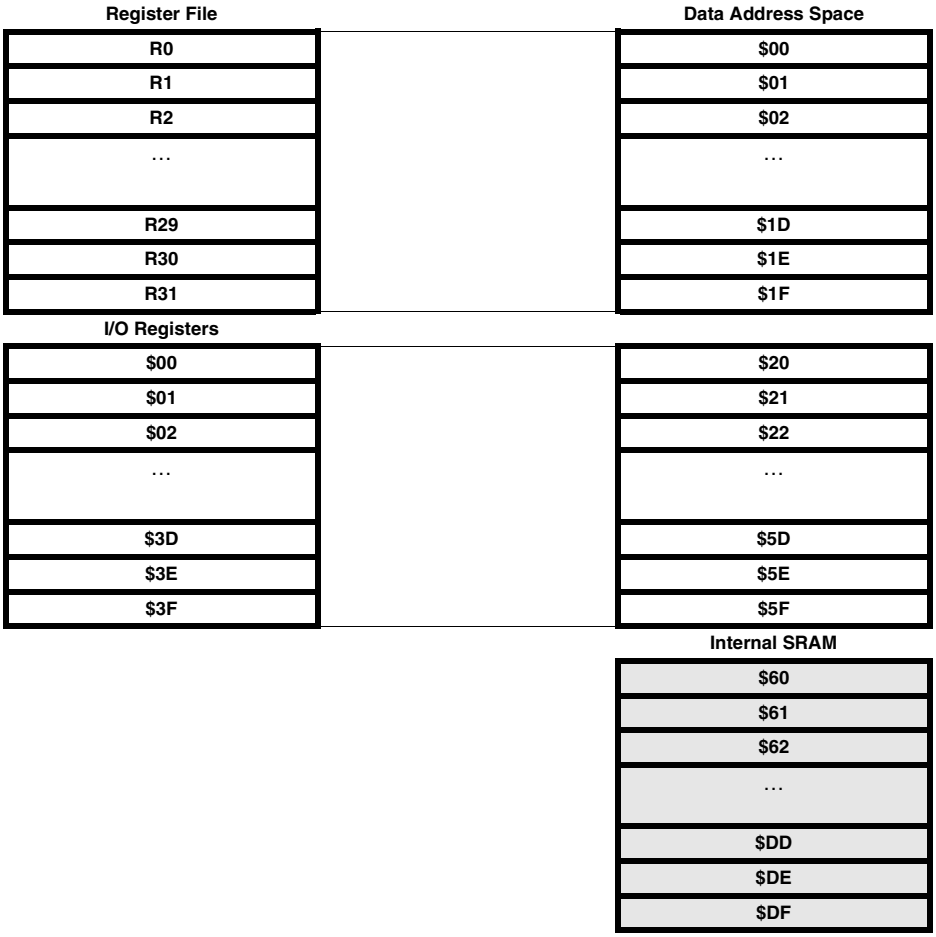
The ATtiny22L contains 128 bytes of EEPROM data memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on page 30 specifying the EEPROM address register, the EEPROM data register, and the EEPROM control register.

For the SPI data downloading, see page 38 for a detailed description.

SRAM Data Memory

The following figure shows how the ATtiny22L Data Memory is organized:

Figure 6. SRAM Organization



The 224 Data Memory locations address the Register file, I/O Memory and the data SRAM. The first 96 locations address the Register File + I/O Memory, and the next 128 locations address the 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 address space.

The Indirect with Displacement mode features 63 address locations reach from the base address given by the Y and Z register.

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

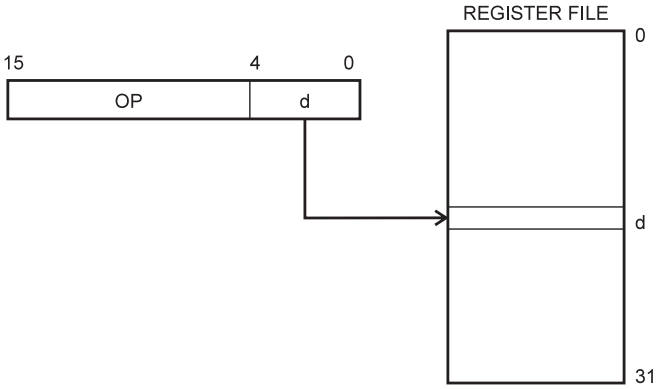
The 32 general purpose working registers, 64 I/O registers and the 128 bytes of data SRAM in the ATtiny22L are all directly accessible through all these addressing modes.

Program and Data Addressing Modes

The ATtiny22L AVR RISC Microcontroller supports powerful and efficient addressing modes for access to the program memory (Flash) and data memory. This section describes the different addressing modes supported by the AVR architecture. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

Register Direct, Single Register Rd

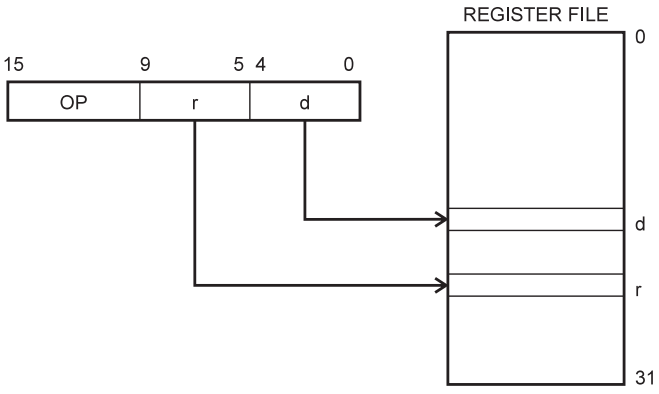
Figure 7. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

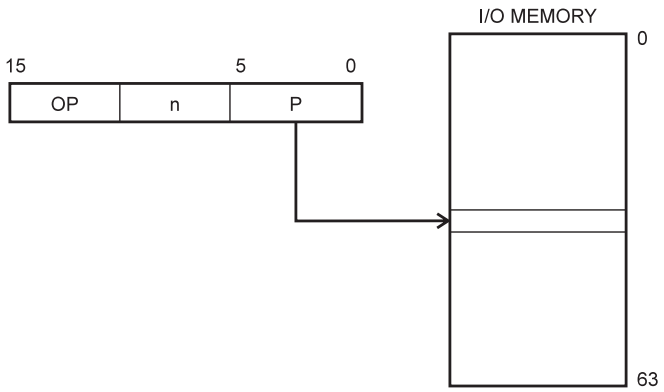
Figure 8. Direct Register Addressing, Two Registers



Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O Direct

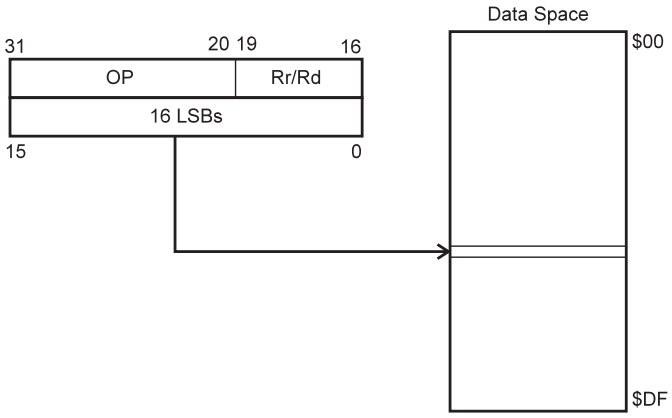
Figure 9. I/O Direct Addressing



Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

Data Direct

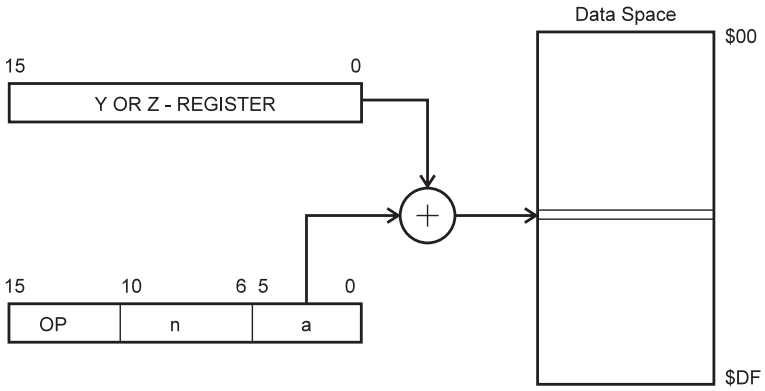
Figure 10. Direct Data Addressing



A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

Data Indirect with Displacement

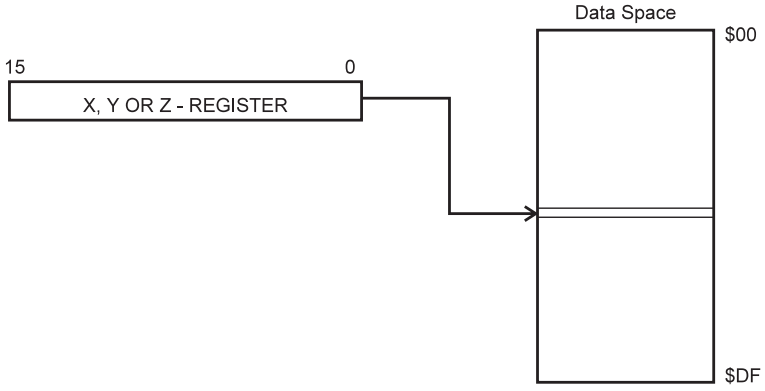
Figure 11. Data Indirect with Displacement



Operand address is the result of the Y or Z-register contents added to the address contained in 6 bits of the instruction word.

Data Indirect

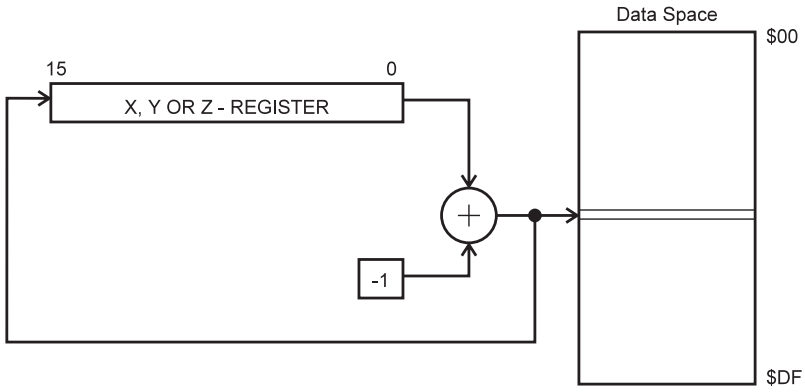
Figure 12. Data Indirect Addressing



Operand address is the contents of the X, Y or the Z-register.

Data Indirect With Pre-Decrement

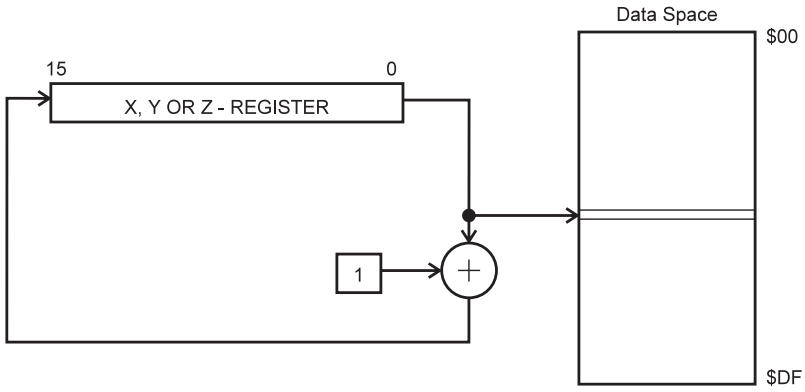
Figure 13. Data Indirect Addressing With Pre-Decrement



The X, Y or the Z-register is decremented before the operation. Operand address is the decremented contents of the X, Y or the Z-register.

Data Indirect With Post-Increment

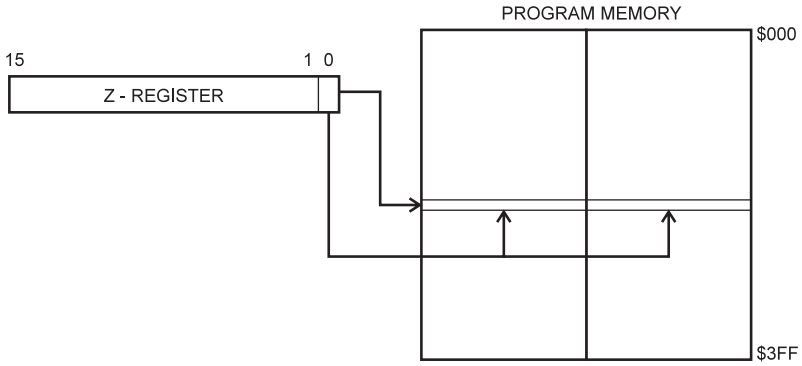
Figure 14. Data Indirect Addressing With Post-Increment



The X, Y or the Z-register is incremented after the operation. Operand address is the content of the X, Y or the Z-register prior to incrementing.

Constant Addressing Using the LPM Instruction

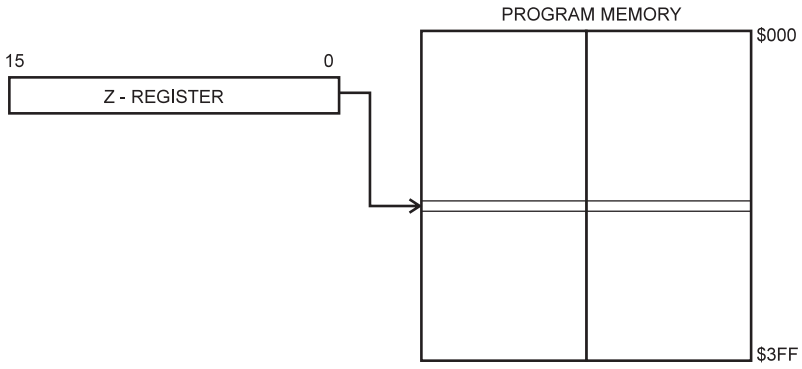
Figure 15. Code Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 1K), the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1).

Indirect Program Addressing, IJMP and ICALL

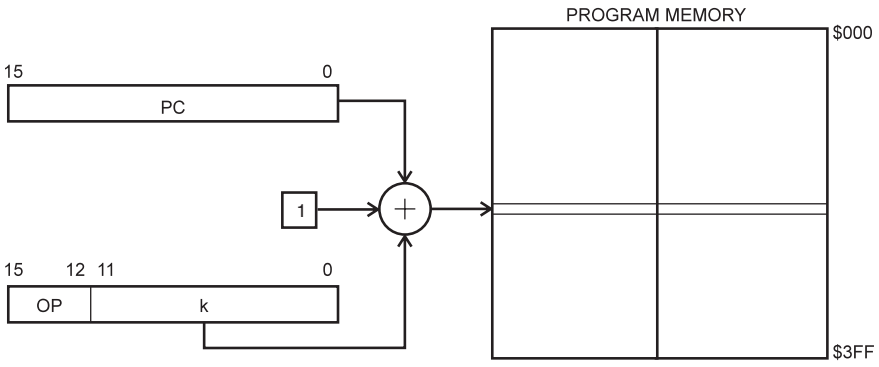
Figure 16. Indirect Program Memory Addressing



Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the content of the Z-register).

Relative Program Addressing, RJMP and RCALL

Figure 17. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is -2048 to 2047.

Memory Access and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the System Clock Ø, directly generated from the internal RC oscillator. No internal clock division is used.

Figure 18. The Parallel Instruction Fetches and Instruction Executions

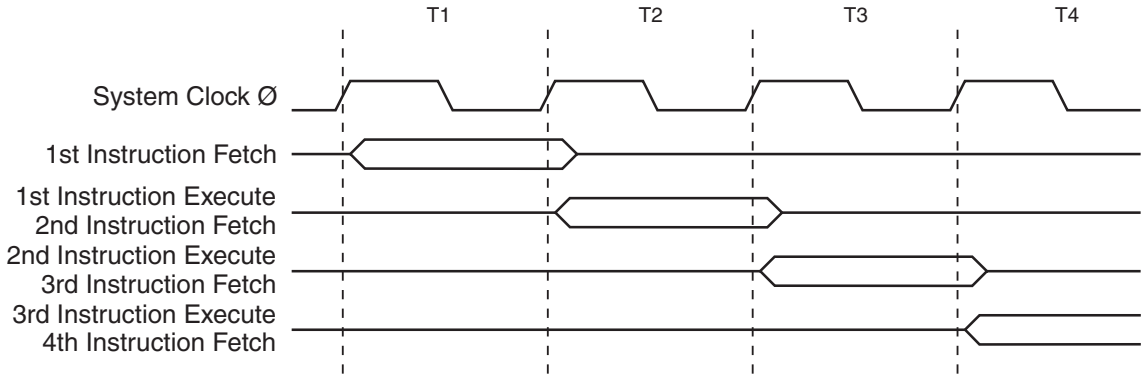


Figure 18 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 19. Single Cycle ALU Operation

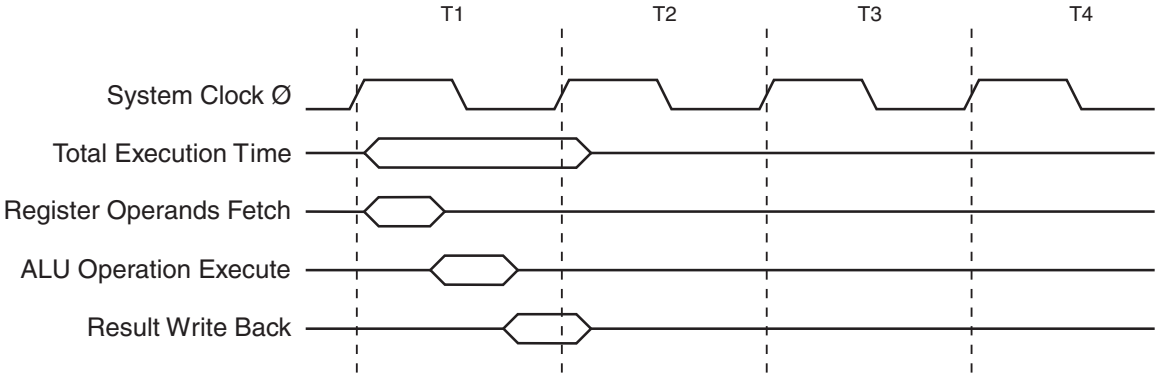
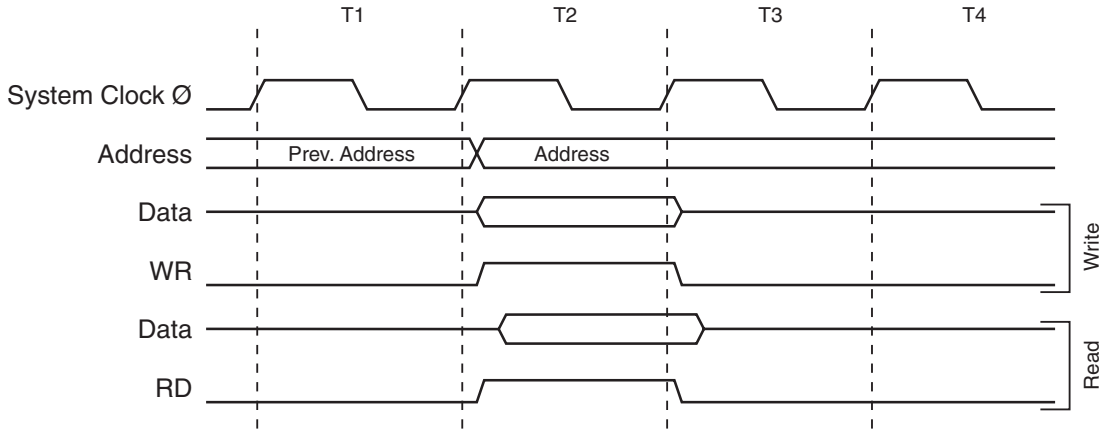


Figure 19 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 20. On-chip Data SRAM Access Cycles



The internal data SRAM access is performed in two System Clock cycles as described in Figure 20.

I/O Memory

The I/O space definition of the ATtiny22L is shown in the following table:

Table 1. ATtiny22L I/O Space

Address Hex	Name	Function
$3F ($5F)	SREG	Status REGister
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag register
$35 ($55)	MCUCR	MCU Control Register
$34 ($54)	MCUSR	MCU Status Register
$33 ($53)	TCCR0	Timer/Counter 0 Control Register
$32 ($52)	TCNT0	Timer/Counter 0 (8-bit)
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1E ($3E)	EEAR	EEPROM Address Register
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$18 ($38)	PORTB	Data Register, Port B
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B

Note: Reserved and unused locations are not shown in the table.

All the different ATtiny22L I/O and peripherals are placed in the I/O space. The different I/O locations are accessed by the IN and OUT instructions transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set chapter for more details. When using the I/O specific commands IN, OUT the I/O addresses $00 - $3F must be used. When addressing I/O registers as SRAM, $20 must be added to this address. All I/O register addresses throughout this document are shown with the SRAM address in parentheses.

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 the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

The different I/O and peripherals control registers are explained in the following sections.

Status Register - SREG

The AVR status register - SREG - at I/O space location $3F ($5F) is defined as:

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	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 (one) 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 (zero), 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.

- **Bit 6 - T: Bit Copy Storage**

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source and 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. See the Instruction Set Description for detailed information.

- **Bit 4 - S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the Instruction Set Description for detailed information.

- **Bit 3 - V: Two's Complement Overflow Flag**

The two's complement overflow flag V supports two's complement arithmetics. See the Instruction Set Description for detailed information.

- **Bit 2 - N: Negative Flag**

The negative flag N indicates a negative result from an arithmetical or logical operation. See the Instruction Set Description for detailed information.

- **Bit 1 - Z: Zero Flag**

The zero flag Z indicates a zero result from an arithmetical or logical operation. See the Instruction Set Description for detailed information.

- **Bit 0 - C: Carry Flag**

The carry flag C indicates a carry in an arithmetical or logical operation. See the Instruction Set Description for detailed information.

Note that the status register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt routine. This must be handled by software.

Stack Pointer - SPL

An 8-bit register at I/O address $3D ($5D) forms the stack pointer of the ATtiny22L. 8 bits are used to address the 128 bytes of SRAM in locations $60 - $DF.

Bit	7	6	5	4	3	2	1	0	
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
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 Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The stack pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when an address is pushed onto the Stack with subroutine calls and interrupts. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when an address is popped from the Stack with return from subroutine RET or return from interrupt RETI.

Reset and Interrupt Handling

The ATtiny22L provides two interrupt sources. These interrupts and the separate reset vector, each have a separate program vector in the program memory space. Both interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 2 . The list also determines the priority levels of the interrupts. The lower the address the higher is the priority level. RESET has the highest priority, next is INT0 - the External Interrupt Request 0, etc.

Table 2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin, Power-on Reset and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	TIMER0, OVF0	Timer/Counter0 Overflow

The most typical program setup for the Reset and Interrupt Vector Addresses are:

```
Address     Labels      Code                    Comments
$000                    rjmp RESET              ; Reset Handler
$001                    rjmp EXT_INT0           ; IRQ0 Handler
$002                    rjmp TIM_OVF0           ; Timer0 Overflow Handler;
$003        MAIN:       ldi r16, low(RAMEND)    ; Main program start
                        out SPL, r16
                        <instr> xxx
 …           …           …                       …
```

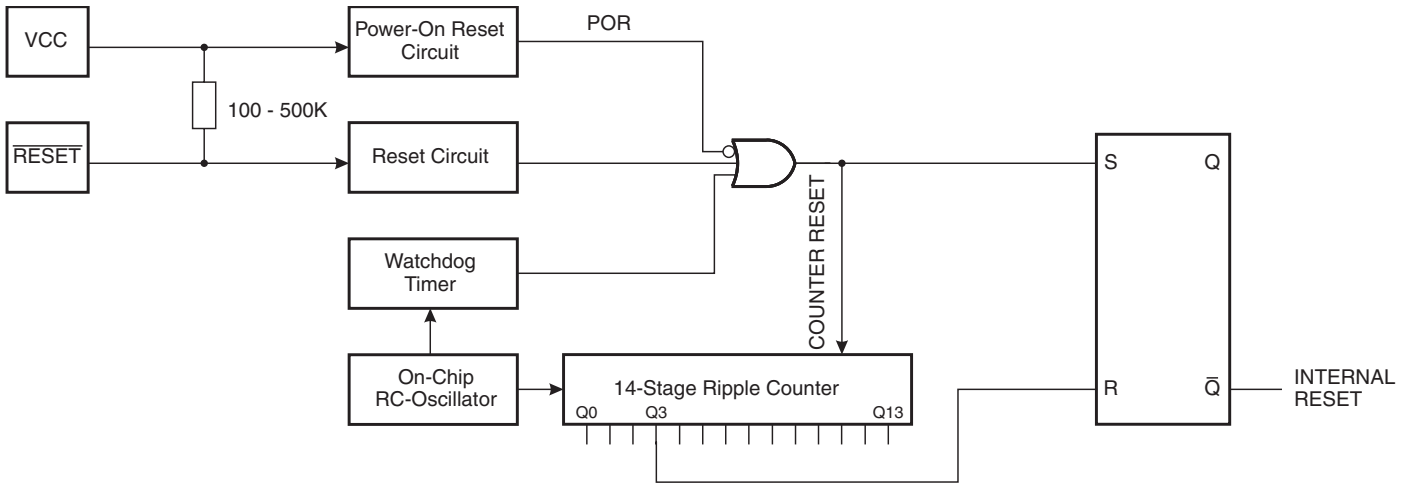
Reset Sources

The ATtiny22L provides three 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 more than 50 ns.
- Watchdog Reset. The MCU is reset when the Watchdog timer period expires and the Watchdog is enabled.

During reset, all I/O registers are set to their initial values, and the program starts execution from address $000. The instruction placed in address $000 must be an RJMP - relative jump - instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 21 shows the reset logic. Table 3 defines the timing and electrical parameters of the reset circuitry.

Figure 21. Reset Logic



The ATtiny22L has a fixed startup time.

Table 3. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}(1)	Power-On Reset Threshold Voltage, rising	1.0	1.4	1.8	V
	Power-On Reset Threshold Voltage, falling	0.4	0.6	0.8	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.6 V_{CC}		V
t_{TOUT}	Reset Delay Time-Out Period ATtiny22L	11	16	21	µs

Notes: 1. The Power-On Reset will not work unless the supply voltage has been below V_{POT} (falling).

Table 4. Reset Characteristics (V_{CC} = 3.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}(1)	Power-On Reset Threshold Voltage, rising	1.0	1.4	1.8	V
	Power-On Reset Threshold Voltage, falling	0.4	0.6	0.8	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.6 V_{CC}		V
t_{TOUT}	Reset Delay Time-Out Period ATtiny22L	22	32	42	µs

Notes: 1. The Power-On Reset will not work unless the supply voltage has been below V_{POT} (falling).

Power-On Reset

The ATtiny22L is designed for use in systems where it can operate from the internal RC oscillator. After V_{CC} has reached V_{POT}, the device will start after the time t_{TOUT} (see Figure 22).

The start-up time t_{TOUT} is one RC-oscillator cycle. The frequency of the RC oscillator is voltage dependent as shown in “Typical characteristics” on page 44.

Figure 22. MCU Start-Up, $\overline{\text{RESET}}$ Tied to V_{CC}.

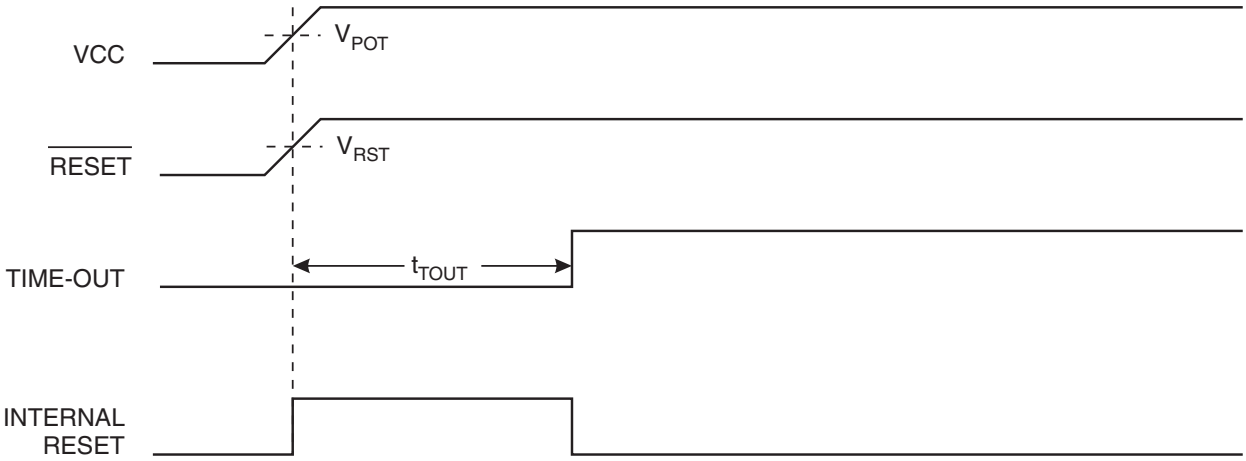
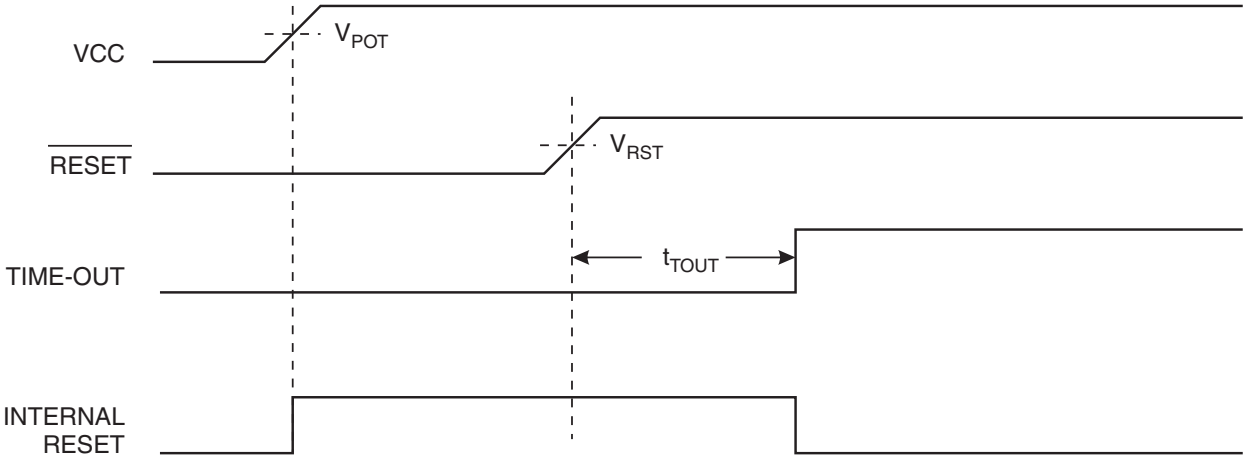


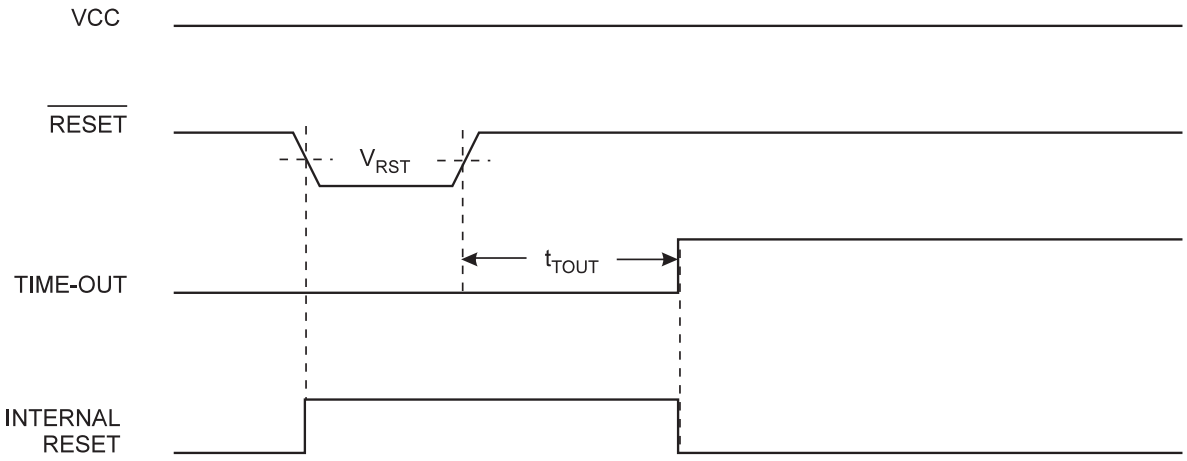
Figure 23. MCU Start-Up, $\overline{\text{RESET}}$ Controlled Externally



External Reset

An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 50 ns 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 timer starts the MCU after the Time-out period t_{TOUT} has expired.

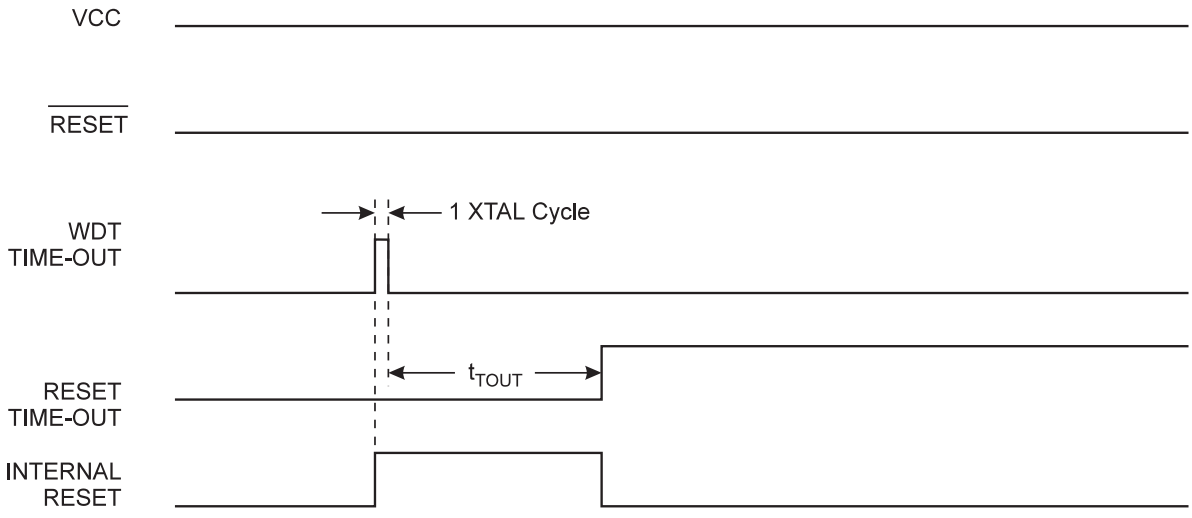
Figure 24. External Reset During Operation



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 clock cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 28 for details on operation of the Watchdog.

Figure 25. Watchdog Reset During Operation



MCU Status Register - MCUSR

The MCU Status Register provides information on which reset source caused a MCU reset:

Bit	7	6	5	4	3	2	1	0	
$34 ($54)	-	-	-	-	-	-	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	See bit description		

- **Bit 7..2 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and always read as zero.

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

After a power-on reset, this bit is undefined (X). It will be set by an external reset. A watchdog reset will leave this bit unchanged.

- **Bit 0 - PORF: Power-On Reset Flag**

This bit is set by a power-on reset. A watchdog reset or an external reset will leave this bit unchanged.

To summarize, the following table shows the value of these two bits after the three modes of reset.

Table 5. PORF and EXTRF Values after Reset

Reset Source	PORF	EXTRF
Power-On Reset	1	undefined
External Reset	unchanged	1
Watchdog Reset	unchanged	unchanged

To make use of these bits to identify a reset condition, the user software should clear both the PORF and EXTRF bits as early as possible in the program. Checking the PORF and EXTRF values is done before the bits are cleared. If the bit is cleared before an external or watchdog reset occurs, the source of reset can be found by using the following truth table:

Table 6. Reset Source Identification

PORF	EXTRF	Reset Source
0	0	Watchdog Reset
0	1	External Reset
1	0	Power-On Reset
1	1	Power-On Reset

Interrupt Handling

The ATtiny22L has two 8-bit Interrupt Mask control registers; GIMSK - General Interrupt Mask register and TIMSK - Timer/Counter Interrupt Mask register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software can set (one) the I-bit to enable nested interrupts. The I-bit is set (one) when a Return from Interrupt instruction - RETI - is executed.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the 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 when the corresponding interrupt enable bit is cleared (zero), the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software.

If one or more interrupt conditions occur when the global interrupt enable bit is cleared (zero), the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set (one), and will be executed by order of priority.

Note that external level interrupt does not have a flag, and will only be remembered for as long as the interrupt condition is active.

Note that the status register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt routine. This must be handled by software.

General Interrupt Mask Register - GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	-	INT0	-	-	-	-	-	-	GIMSK
Read/Write	R	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 - Res: Reserved Bit**

This bit is a reserved bit in the ATtiny22L and always reads as 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 general Control Register (MCUCR) defines whether the external interrupt is activated on rising 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 program memory address $001. See also “External Interrupts.”

- **Bits 5..0 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and always read as zero.

General Interrupt Flag Register - GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	-	INTF0	-	-	-	-	-	-	GIFR
Read/Write	R	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 - Res: Reserved Bit**

This bit is a reserved bit in the ATtiny22L and always reads as zero.

- **Bit 6 - INTF0: External Interrupt Flag0**

When an event 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 interrupt vector at address $001. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bits 5..0 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and always read as zero.

Timer/Counter Interrupt Mask Register - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	-	-	-	-	-	-	TOIE0	-	TIMSK
Read/Write	R	R	R	R	R	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7..2 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and always read zero.

- **Bit 1 - TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt (at vector $002) is executed if an overflow in Timer/Counter0 occurs, i.e., when the Overflow Flag (Timer/Counter0) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

- **Bit 0 - Res: Reserved Bit**

This bit is a reserved bit in the ATtiny22L and always reads as zero.

Timer/Counter Interrupt FLAG Register - TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	-	-	-	-	-	-	TOV0	-	TIFR
Read/Write	R	R	R	R	R	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..2 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and always read zero.

- **Bit 1 - TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logical one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

- **Bit 0 - Res: Reserved Bit**

This bit is a reserved bit in the ATtiny22L and always reads zero.

External Interrupt

The external interrupt is triggered by the INT0 pin. Observe that, if enabled, the interrupt will trigger even if the INT0 pin is configured as an output. This feature provides a way of generating a software interrupt. The external interrupt can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register - MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low.

The external interrupt is set up as described in the specification for the MCU Control Register - MCUCR.

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is 4 clock cycles minimum. 4 clock cycles after the interrupt flag has been set, the program vector address for the actual interrupt handling routine is executed. During these 4 clock cycles, the Program Counter (2 bytes) is popped back from the Stack, the Stack Pointer is incremented by 2, and the I flag in SREG is set. The vector is a relative jump to the interrupt routine, and this jump takes 2 clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.

A return from an interrupt handling routine (same as for a subroutine call routine) takes 4 clock cycles. During these 4 clock cycles, the Program Counter (2 bytes) is popped back from the Stack, and the Stack Pointer is incremented by 2. 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.

MCU Control Register - MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	-	-	SE	SM	-	-	ISC01	ISC00	MCUCR
Read/Write	R	R	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7, 6 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and always read as zero.

- **Bit 5 - SE: Sleep Enable**

The SE bit must be set (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 programmers purpose, it is recommended to set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

- **Bit 4 - SM: Sleep Mode**

This bit selects between the two available sleep modes. When SM is cleared (zero), Idle Mode is selected as Sleep Mode. When SM is set (one), Power Down mode is selected as sleep mode. For details, refer to the section “Sleep Modes” on page 25.

- **Bits 3, 2 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L, and always read as zero.

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

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask is set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 7. The value on the INT01 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 7. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Note: When changing the ISC01/ISC00 bits, INT0 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Sleep Modes

To enter the sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU awakes, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file, SRAM and I/O memory are unaltered. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset vector.

Idle Mode

When the SM bit is cleared (zero), the SLEEP instruction forces the MCU into the Idle Mode stopping the CPU but allowing Timer/Counters, Watchdog and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like Timer Overflow interrupt and watchdog reset.

Power Down Mode

When the SM bit is set (one), the SLEEP instruction forces the MCU into the Power Down Mode. In this mode, the external oscillator is stopped, while the external interrupts and the Watchdog (if enabled) continue operating. Only an external reset, a watchdog reset (if enabled), or an external level interrupt on INT0 can wake up the MCU.

Note that if a level triggered interrupt is used for wake-up from Power Down Mode, the changed level must be held for some time to wake up the MCU. This makes the MCU less sensitive to noise. The changed level is sampled twice by the watchdog oscillator clock, and if the input has the required level during this time, the MCU will wake up. The period of the watchdog oscillator is 1 us (nominal) at 5.0V and 25C. The frequency of the watchdog oscillator is voltage dependent as shown in section “Typical characteristics” on page 44.

When waking up from Power Down Mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is equal to the clock reset period, as shown in Table 3 and Table 4.

If the wake-up condition disappears before the MCU wakes up and starts to execute, e.g. a low level on is not held long enough, the interrupt causing the wake-up will not be executed.

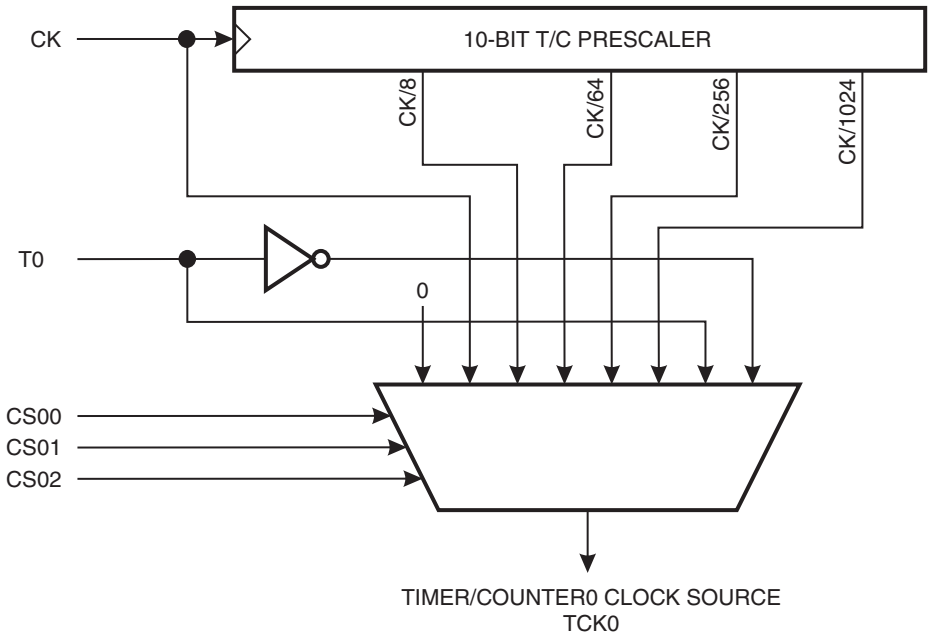
Timer/Counter

The ATtiny22L provides one general purpose 8- bit Timer/Counter - Timer/Counter0. The Timer/Counter has prescaling selection from the 10-bit prescaling timer. The Timer/Counter can either be used as a timer with an internal clock timebase or as a counter with an external pin connection that triggers the counting.

Timer/Counter Prescaler

Figure 26 shows the Timer/Counter prescaler.

Figure 26. Timer/Counter0 Prescaler



The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024 where CK is the oscillator clock. CK, external source and stop, can also be selected as clock sources.

8-Bit Timer/Counter0

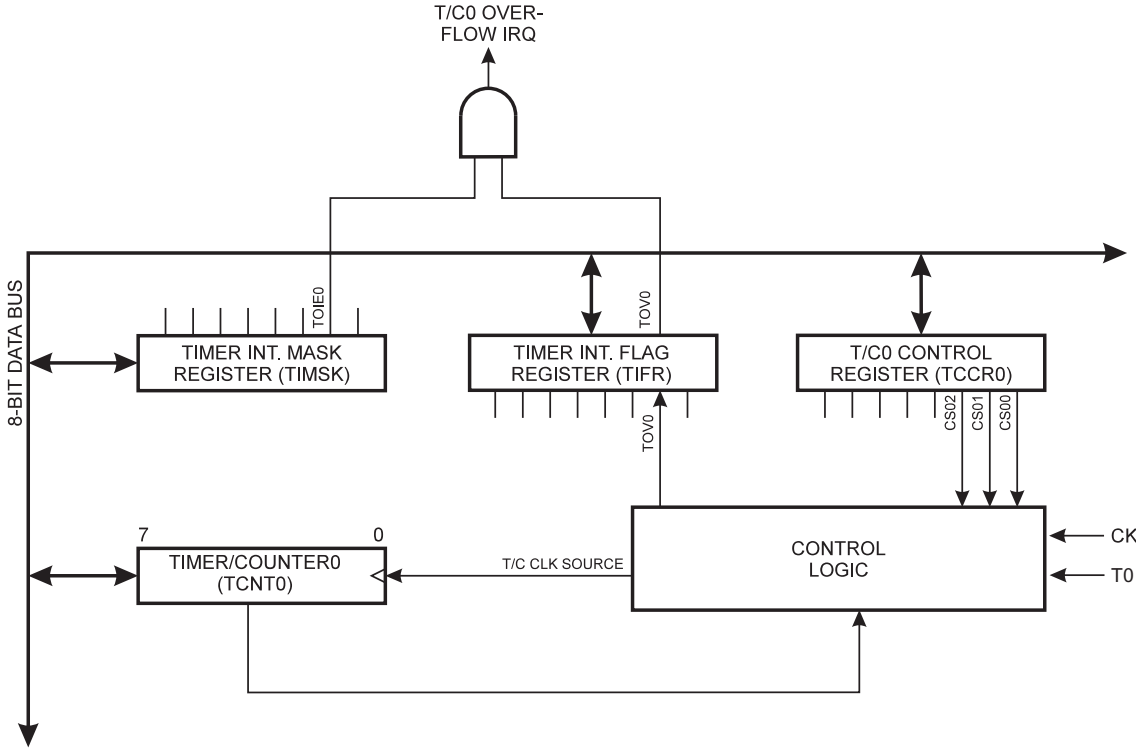
Figure 27 shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition, it can be stopped as described in the specification for the Timer/Counter0 Control Register - TCCR0. The overflow status flag is found in the Timer/Counter Interrupt Flag Register - TIFR. Control signals are found in the Timer/Counter0 Control Register - TCCR0. The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To ensure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

Figure 27. Timer/Counter 0 Block Diagram



Timer/Counter0 Control Register - TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	-	-	-	-	-	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..3 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L, and always read zero.

- **Bits 2,1,0 - CS02, CS01, CS00: Clock Select0, Bit 2,1 and 0**

The Clock Select0 bits 2,1 and 0 define the prescaling source of Timer0.

Table 8. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, the Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used for Timer/Counter0, transitions on PB2/(T0) will clock the counter even if the pin is configured as an output. This feature can give the user SW control of the counting.

Timer Counter 0 - TCNT0

Bit	7	6	5	4	3	2	1	0	
$32 ($52)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

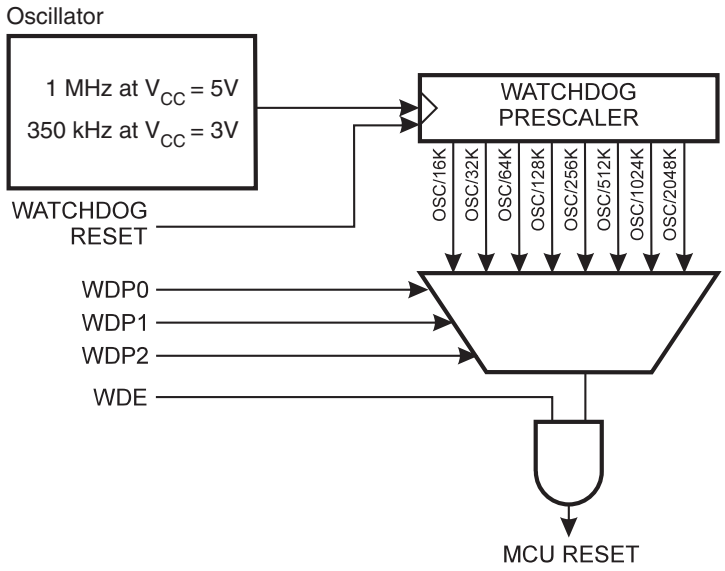
The Timer/Counter0 is realized as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the timer clock cycle following the write operation.

Watchdog Timer

The Watchdog Timer is clocked from the on-chip RC oscillator. By controlling the Watchdog Timer prescaler, the Watchdog reset interval can be adjusted as shown in Table 9. See characterization data for typical values at other V_{CC} levels. The WDR - Watchdog Reset - instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog reset, the ATtiny22L resets and executes from the reset vector. For timing details on the Watchdog reset, refer to page 21.

To prevent unintentional disabling of the watchdog, a special turn-off sequence must be followed when the watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

Figure 28. Watchdog Timer



Watchdog Timer Control Register - WDTCR

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
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 bits in the ATtiny22L and will always read as zero.

- **Bit 4 - WDTOE: Watch Dog Turn-Off Enable**

This bit must be set (one) when the WDE bit is cleared. Otherwise, the watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a watchdog disable procedure.

- **Bit 3 - WDE: Watch Dog Enable**

When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set(one). To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDTOE and WDE. A logical 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 logical 0 to WDE. This disables the watchdog.

- **Bits 2..0 - WDP2, WDP1, WDP0: Watchdog Timer Prescaler 1 and 0**

The WDP2, WDP1 and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding time-out periods are shown in Table 9.

Table 9. Watch Dog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator cycles	Typical time-out at V_{CC} = 3.0V	Typical time-out at V_{CC} = 5.0V
0	0	0	16K cycles	47 ms	15 ms
0	0	1	32K cycles	94 ms	30 ms
0	1	0	64K cycles	0.19 s	60 ms
0	1	1	128K cycles	0.38 s	0.12 s
1	0	0	256K cycles	0.75 s	0.24 s
1	0	1	512K cycles	1.5 s	0.49 s
1	1	0	1,024K cycles	3.0 s	0.97 s
1	1	1	2,048K cycles	6.0 s	1.9 s

Note: The frequency of the watchdog oscillator is voltage dependent as shown in the Electrical Characteristics section.
The WDR - Watchdog Reset - instruction should always be executed before the Watchdog Timer is enabled. This ensures that the reset period will be in accordance with the Watchdog Timer prescale settings. If the Watchdog Timer is enabled without reset, the watchdog timer may not start to count from zero.

EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4ms, depending on the V_{CC} voltages. A self-timing function, however, lets the user software detect when the next byte can be written.

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

When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed. When it is read, the CPU is halted for 4 clock cycles.

EEPROM Address Register - EEAR

Bit	7	6	5	4	3	2	1	0	
$1E ($3E)	-	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
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 - Res: Reserved Bit**

This bit is a reserved bit in the ATtiny22L and will always read as zero.

- **Bit 6..0 - EEAR6..0: EEPROM Address**

The EEPROM Address Register - EEAR6..0 - specifies the EEPROM address in the 128 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 127.

EEPROM Data Register - EEDR

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	MSB							LSB	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	

- **Bit 7..0 - EEDR7..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.

EEPROM Control Register - EECR

Bit	7	6	5	4	3	2	1	0	
$1C ($3C)	-	-	-	-	-	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

These bits are reserved bits in the ATtiny22L and will always read as zero.

- **Bit 2 - EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set(one) setting EEWE will write data to the EEPROM at the selected address. If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for a EEPROM write procedure.

- **Bit 1 - EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is unessential):

1. Wait until EEWE becomes zero
2. Write new EEPROM address to EEAR (optional)
3. Write new EEPROM data to EEDR (optional)
4. Write a logical one to the EEMWE bit in EECR
5. Within four clock cycles after setting EEMWE, write a logical one to EEWE

Caution: An interrupt between step 4 and step 5 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR and EEDR register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the global interrupt flag cleared during the 4 last steps to avoid these problems. When the write access time (typically 2.5 ms at V_{CC} = 5V or 4 ms at V_{CC} = 2.7V) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE 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 set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR register. The EEPROM read access takes one instruction and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two cycles before the next instruction is executed. The user should poll the EEWE bit before starting the read operation. If a write operation is in progress when new data or address is written to the EEPROM I/O registers, the write operation will be interrupted, and the result is undefined.

Prevent 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 the 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 for executing instructions is too low.

EEPROM data 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 is best done by an external low V_{CC} Reset Protection circuit, often referred to as a Brown-Out Detector (BOD). Please refer to application note AVR 180 for design considerations regarding power-on reset and low voltage detection.
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 EEPROM registers from unintentional writes.
3. Store constants in Flash memory if the ability to change memory contents from software is not required. Flash memory can not be updated by the CPU, and will not be subject to corruption.

I/O Port B

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 for changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input).

Port B is a 5-bit bi-directional I/O port.

Three I/O memory address locations are allocated for Port B, one each for the Data Register - PORTB, $18 ($38), Data Direction Register - DDRB, $17($37) and the Port B Input Pins - PINB, $16($36). The Port B Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port B output buffers can sink 20mA and thus drive LED displays directly. When pins PB0 to PB4 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

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

Table 10. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB0	MOSI (Data input line for memory downloading)
PB1	MISO (Data output line for memory uploading) INT0 (External Interrupt0 Input)
PB2	SCK (Serial clock input for serial programming) TO (Timer/Counter0 counter clock input)

When the pins are used for the alternate function the DDRB and PORTB register has to be set according to the alternate function description.

Port B Data Register - PORTB

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	-	-	-	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

Port B Data Direction Register - DDRB

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	-	-	-	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

Port B Input Pins Address - PINB

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	-	-	-	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	N/A	N/A	N/A	N/A	N/A	

The Port B Input Pins address - PINB - is not a register, and this address enables access to the physical value on each Port B pin. When reading PORTB, the Port B Data Latch is read, and when reading PINB, the logical values present on the pins are read.

General Digital I/O

All pins in port B have equal functionality when used as digital I/O pins.

PBn, General I/O pin: The DDBn bit in the DDRB register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTBn has to be cleared (zero) or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 11. DDBn Effects on Port B Pins

DDBn	PORTBn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PBn will source current if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

Alternate Functions of Port B

The alternate pin functions of Port B are:

SCK/T0 - Port B, Bit 2

In serial programming mode, this bit serves as the serial clock input, SCK.

During normal operation, this pin can serve as the external counter clock input. See the timer/counter description for further details. If external timer/counter clocking is selected, activity on this pin will clock the counter even if it is configured as an output.

MISO/INT0 - Port B, Bit 1

In serial programming mode, this bit serves as the serial data output, MISO.

During normal operation, this pin can serve as the external interrupt0 input. See the interrupt description for details on how to enable this interrupt. Note that activity on this pin will trigger the interrupt even if the pin is configured as an output.

MOSI/T0 - Port B, Bit 0

In serial programming mode, this pin serves as the serial data input, MOSI.

Memory Programming

Program and Data Memory Lock Bits

The ATtiny22L MCU provides two lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 12 . The Lock bits can only be erased with the Chip Erase operation.

Table 12. Lock Bit Protection Modes

Memory Lock Bits			
Mode	**LB1**	**LB2**	**Protection Type**
1	1	1	No memory lock features enabled.
2	0	1	Further programming of the Flash and EEPROM is disabled.(1)
3	0	0	Same as mode 2, and verify is also disabled.

Note: 1. In the High-voltage Serial Programming mode, further programming of the Fuse bit is also disabled. Program the fuse bit before programming the lock bits.

Fuse Bit

The ATtiny22L has one Fuse bit, SPIEN.

- When the SPIEN Fuse is programmed (“0”), Serial Program and Data Downloading is enabled. Default value is programmed (“0”). This bit is not accessible in the Low-Voltage Serial Programming mode.

The status of the Fuse bit is not affected by Chip Erase.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. The three bytes reside in a separate address space.

For ATtiny22L(1) they are:

1. $000: $1E (indicates manufactured by Atmel)
2. $001: $91 (indicates 2K bytes Flash memory)
3. $002: $06 (Indicates ATtiny22L when signature byte $001 is $91.)

Note: 1. When both lock bits are programmed (Lock mode 3), the signature bytes can not be read in the Low-voltage Serial mode. Reading the signature bytes will return: $00, $01 and $02.

Programming the Flash and EEPROM

Atmel’s ATtiny22L offers 2K bytes of in-system programmable Flash Program memory and 128 bytes of EEPROM Data memory.

The ATtiny22L is shipped with the on-chip Flash Program and EEPROM Data memory arrays in the erased state (i.e., contents = $FF) and ready to be programmed.

The device supports a High-voltage (12V) Serial Programming mode and a Low-voltage Serial Programming mode. The +12V is used for programming enable only, and no current of significance is drawn by this pin. The Low-voltage Serial Programming mode provides a convenient way to download Program and Data into the device inside the user’s system.

The Program and EEPROM memory arrays in the ATtiny22L are programmed byte-by-byte in either programming modes. For the EEPROM, an auto-erase cycle is provided within the self-timed write instruction in the Low-voltage Serial Programming mode.

During programming, the supply voltage must be in accordance with Table 13.

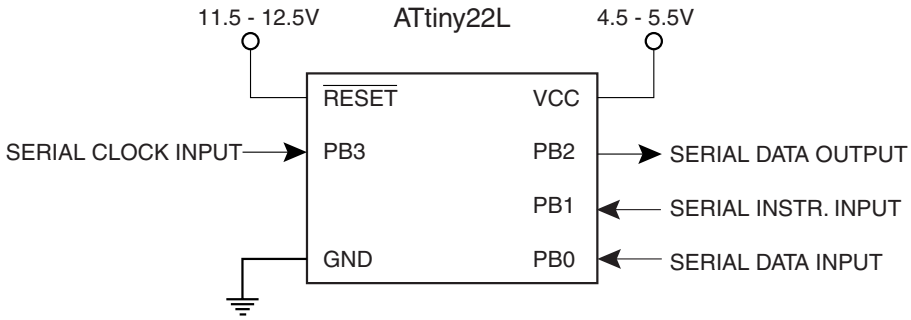
Table 13. Supply Voltage During Programming

Part	Low-voltage Serial Programming	High-voltage Serial Programming
ATtiny22L	2.7 - 6.0V	4.5 - 5.5V

High-Voltage Serial Programming

This section describes how to program and verify Flash Program memory, EEPROM Data memory, Lock bits and Fuse bit in the ATtiny22L.

Figure 29. High-Voltage Serial Programming



High-Voltage Serial Programming Algorithm

To program and verify the ATtiny22L in the high-voltage serial programming mode, the following sequence is recommended (See instruction formats in Table 14):

1. Power-up sequence: Apply 4.5 - 5.5V between V_{CC} and GND. Set $\overline{\text{RESET}}$ and PB0 to “0” and wait at least 100 ns. Set PB3 to “0”. Wait at least 4µs. Apply 12V to $\overline{\text{RESET}}$ and wait at least 100 ns before changing PB0. Wait 8 µs before giving any instructions.
2. The Flash array is programmed one byte at a time by supplying first the address, then the low and high data byte. The write instruction is self-timed, wait until the PB2 (RDY/$\overline{\text{BSY}}$) pin goes high.
3. The EEPROM array is programmed one byte at a time by supplying first the address, then the data byte. The write instruction is self-timed, wait until the PB2 (RDY/$\overline{\text{BSY}}$) pin goes high.
4. Any memory location can be verified by using the Read instruction which returns the contents at the selected address at serial output PB2.
5. Power-off sequence:Set PB3 to “0”.
   Set $\overline{\text{RESET}}$ to “0”.
   Turn V_{CC} power off.

When writing or reading serial data to the device, data is clocked on the rising edge of the serial clock, see Figure 30, Figure 31 and Table 15 for details.

Figure 30. High-Voltage Serial Programming Waveforms

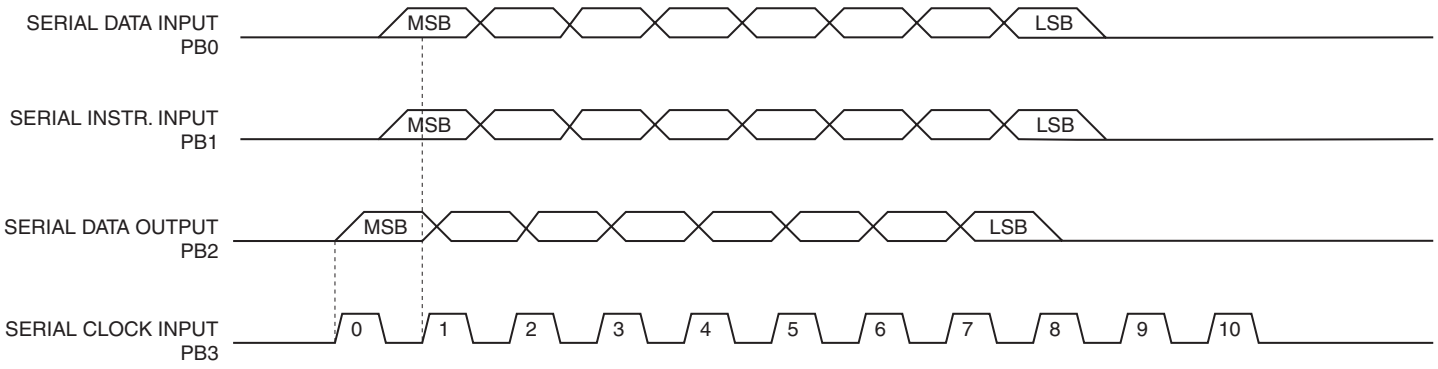


Table 14. High-Voltage Serial Programming Instruction Set

Instruction		Instruction Format				Operation Remarks
		Instr.1	Instr.2	Instr.3	Instr.4	
Chip Erase	PB0 PB1 PB2	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	Wait t_{WLWH_CE} after Instr.3 for the Chip Erase cycle to finish.
Write Flash High and Low Address	PB0 PB1 PB2	0_0001_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**aa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx		Repeat Instr.2 for a new 256 byte page. Repeat Instr.3 for each new address.
Write Flash Low byte	PB0 PB1 PB2	0_**iiii_iiii**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 0_0000_0000_00		Wait after Instr.3 until PB2 goes high. Repeat Instr.1, Instr. 2 and Instr.3 for each new address.
Write Flash High byte	PB0 PB1 PB2	0_**iiii_iiii**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 0_0000_0000_00		Wait after Instr.3 until PB2 goes high. Repeat Instr.1, Instr. 2 and Instr.3 for each new address.
Read Flash High and Low Address	PB0 PB1 PB2	0_0000_0010_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**aa**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx		Repeat Instr.2 and Instr.3 for each new address.
Read Flash Low byte	PB0 PB1 PB2	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **o_oooo_ooo**x_xx			Repeat Instr.1 and Instr.2 for each new address.
Read Flash High byte	PB0 PB1 PB2	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **o_oooo_ooo**x_xx			Repeat Instr.1 and Instr.2 for each new address.
Write EEPROM Low Address	PB0 PB1 PB2	0_0001_0001_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0**bbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx			Repeat Instr.2 for each new address.
Write EEPROM byte	PB0 PB1 PB2	0_**iiii_iiii**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 0_0000_0000_00		Wait after Instr.3 until PB2 goes high
Read EEPROM Low Address	PB0 PB1 PB2	0_0000_0011_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0**bbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx			Repeat Instr.2 for each new address.

Table 14. High-Voltage Serial Programming Instruction Set (Continued)

Instruction		Instruction Format				Operation Remarks
		Instr.1	Instr.2	Instr.3	Instr.4	
Read EEPROM byte	PB0 PB1 PB2	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **o_oooo_ooo**x_xx			Repeat Instr.2 for each new address
Write Fuse bit	PB0 PB1 PB2	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_11**S**1_1110_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Wait t_{WLWH_PFB} after Instr.3 for the Write Fuse bit cycle to finish. Set **S** = “0” to program, “1” to unprogram.
Write Lock bits	PB0 PB1 PB2	0_0010_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_1111_1***21***1_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 0_0000_0000_00	Wait after Instr.4 until PB2 goes high. Write ***2***, ***1*** = “0” to program the Lock bit.
Read Fuse and Lock bits	PB0 PB1 PB2	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 ***1_2***Sxx_xx0x_xx		Reading ***1***, ***2***, **S**= “0” means the Fuse/Lock bit is programmed.
Read Signature Bytes	PB0 PB1 PB2	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**bb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **o_oooo_ooo**x_xx	Repeat Instr.2 - Instr.4 for each Signature byte address

Note: **a** = address high bits
b = address low bits
i = data in
o = data out
x = don’t care
1 = Lock Bit1
2 = Lock Bit2
S = SPIEN Fuse

High-Voltage Serial Programming Characteristics

Figure 31. High-Voltage Serial Programming Timing

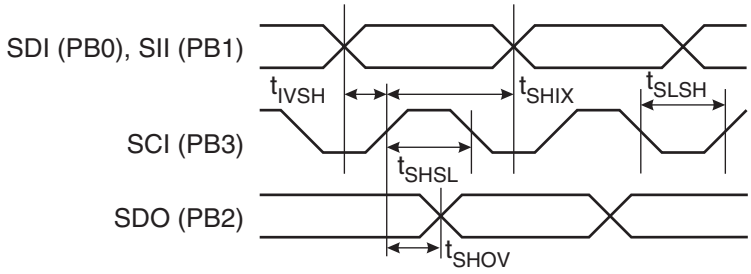


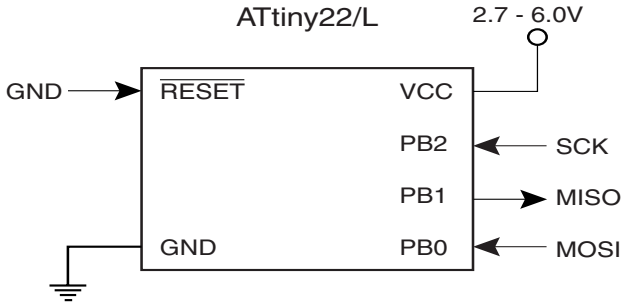
Table 15. High-Voltage Serial Programming Characteristics
T_A = 25°C ± 10%, V_{CC} = 5.0V ± 10% (Unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
t_{SHSL}	SCI (PB3) Pulse Width High	100			ns
t_{SLSH}	SCI (PB3) Pulse Width Low	100			ns
t_{IVSH}	SDI (PB0), SII (PB1) Valid to SCI (PB3) High	50			ns
t_{SHIX}	SDI (PB0), SII (PB1) Hold after SCI (PB3) High	50			ns
t_{SHOV}	SCI (PB3) High to SDO (PB2) Valid	10	16	32	ns
t_{WLWH_CE}	Wait after Instr.3 for Chip Erase	5	10	15	ms
t_{WLWH_PFB}	Wait after Instr.3 for Write Fuse Bit	1.0	1.5	1.8	ms

Low-Voltage Serial Downloading

Both the Program and Data memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output), see Figure 32. After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase instructions can be executed.

Figure 32. Low-voltage Serial Programming and Verify



For the EEPROM, an auto-erase cycle is provided within the self-timed write instruction and there is no need to first execute the Chip Erase instruction. The Chip Erase instruction turns the content of every memory location in both the Program and EEPROM arrays into $FF.

The Program and EEPROM memory arrays have separate address spaces, $0000 to $03FF for Flash Program memory and $000 to $07F for EEPROM Data memory.

The device is clocked from the internal RC-oscillator. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 MCU clock cycles

High: > 2 MCU clock cycles

Low-Voltage Serial Programming Algorithm

When writing serial data to the ATtiny22L, data is clocked on the rising edge of SCK.

When reading data from the ATtiny22L, data is clocked on the falling edge of SCK. See Figure 33, Figure 34 and Table 18 for timing details.

To program and verify the ATtiny22L in the Low-Voltage Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 17):

1. Power-up sequence:

   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0” (if the programmer can not guarantee that SCK is held low during power-up, $\overline{\text{RESET}}$ must be given a positive pulse 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 the MOSI (PB0) pin. Refer to the above section for minimum low and high periods for the serial clock input, SCK.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync, the second byte ($53) will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all 4 bytes of the instruction must be transmitted. If the $53 did not echo back, give SCK a positive pulse and issue a new Programming Enable instruction. If the $53 is not seen within 32 attempts, there is no functional device connected.
4. If a Chip Erase is performed (must be done to erase the Flash), wait t_{WD_ERASE} after the instruction, give $\overline{\text{RESET}}$ a positive pulse, and start over from Step 2. See Table 19 on page 42 for t_{WD_ERASE} value.
5. The Flash or 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. Use Data Polling to detect when the next byte in the Flash or EEPROM can be written. If polling is not used, wait t_{WD_PROG} before transmitting the next instruction. See Table 20 on page 42 for t_{WD_PROG} value. In an erased device, no $FFs in the data file(s) needs to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at the serial output MISO (PB1) pin.
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 “0”.

   Turn V_{CC} power off.

Data Polling EEPROM

When a byte is being programmed into the EEPROM, reading the address location being programmed will give the value P1 until the auto-erase is finished, and then the value P2. See Table 16 for P1 and P2 values.

At the time the device is ready for a new EEPROM byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the values P1 and P2, so when programming these values, the user will have to wait for at least the prescribed time t_{WD_PROG} before programming the next byte. See Table 19 for t_{WD_PROG} value. As a chip-erased device contains $FF in all locations, programming of addresses that are meant to contain $FF, can be skipped. This does not apply if the EEPROM is reprogrammed without first chip-erasing the device.

Table 16. Read back value during EEPROM polling

Part	P1	P2
ATtiny22L	$00	$FF

Data Polling Flash

When a byte is being programmed into the Flash, reading the address location being programmed will give the value $FF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value $FF, so when programming this value, the user will have to wait for at least t_{WD_PROG} before programming the next byte. As a chip-erased device contains $FF in all locations, programming of addresses that are meant to contain $FF, can be skipped.

Figure 33. Low-Voltage Serial Downloading Waveforms

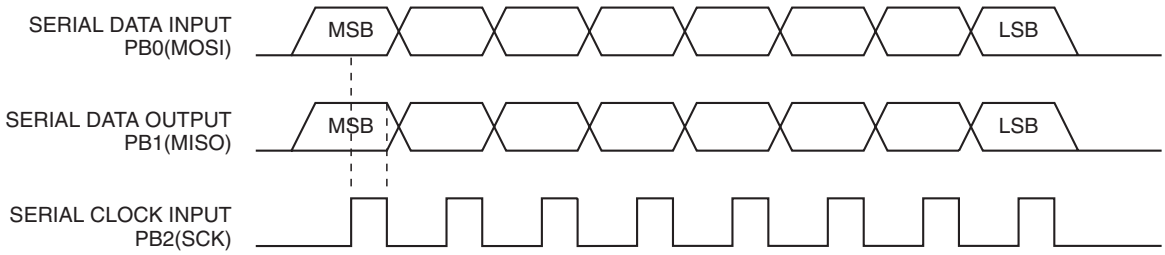


Table 17. Low-Voltage Serial Programming Instruction Set ATtiny22L

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming while $\overline{\text{RESET}}$ is low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase both Flash and EEPROM memory arrays.
Read Program Memory	0010 **H**000	0000 00**aa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**.
Write Program Memory	0100 **H**000	0000 00**aa**	**bbbb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program memory at word address **a**:**b**.
Read EEPROM Memory	1010 0000	0000 0000	x**bbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**.
Write EEPROM Memory	1100 0000	0000 0000	x**bbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**.
Read Lock and Fuse Bit	0101 1000	xxxx xxxx	xxxx xxxx	***12S***x xxx0	Read Lock and Fuse bit. '0' = programmed, '1' = unprogrammed.
Write Lock Bits	1010 1100	1111 1***21***1	xxxx xxxx	xxxx xxxx	Write Lock bits. Set bits ***1***,***2*** = '0' to program Lock bits.
Read Signature Bytes	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature byte **o** from address **b**(1)

Note: **a** = address high bits
b = address low bits
H = 0 - Low byte, 1- High byte
o = data out
i = data in
x = don't care
1 = lock bit 1
2 = lock bit 2
S = SPIEN Fuse

Notes: 1. The signature bytes are not readable in Lock mode 3, i.e. both Lock bits programmed.

Low-Voltage Serial Programming Characteristics

Figure 34. Low-voltage Serial Programming Timing

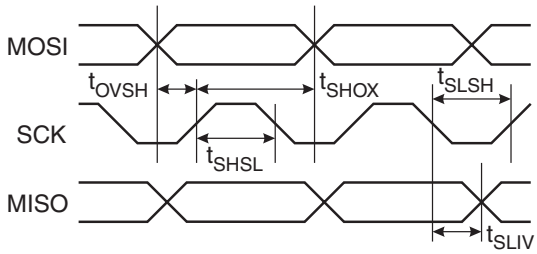


Table 18. Low-voltage Serial Programming Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7 - 6.0V (Unless otherwise noted)

The period of the internal RC oscillator - t_{CLCL} is voltage dependend as shown in “Typical characteristics” on page 44.

Symbol	Parameter	Min	Typ	Max	Units
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	10	16	32	ns

Table 19. Minimum wait delay after the Chip Erase instruction

Symbol	3.2V	3.6V	4.0V	5.0V	Units
t_{WD_ERASE}	18	14	12	8	ms

Table 20. Minimum wait delay after writing a Flash or EEPROM location

Symbol	3.2V	3.6V	4.0V	5.0V	Units
t_{WD_PROG}	9	7	6	4	ms

Electrical Characteristics

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	-1.0V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +13.0V
Maximum Operating Voltage	6.6V
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.

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		-0.5		0.3 V_{CC}[1]	V
V_{IH}	Input High Voltage	(Except $\overline{\text{RESET}}$)	0.6 V_{CC}[2]		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage	$\overline{\text{RESET}}$	0.85 V_{CC}[2]		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage Ports B	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.5 0.4	V V
V_{OH}	Output High Voltage Ports B	I_{OH} = -3 mA, V_{CC} = 5V I_{OH} = -1.5 mA, V_{CC} = 3V	4.2 2.4			V V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 6V, Pin Low (Absolute value)			8.0	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 6V, Pin High (Absolute value)			8.0	µA
RRST	Reset Pullup		100		500	kΩ
$R_{I/O}$	I/O Pin Pullup		30		150	kΩ
I_{CC}	Power Supply Current	Active, V_{CC} = 3V			1.5	mA
		Idle, V_{CC} = 3V			100	µA
		Power Down, V_{CC} = 3V WDT Enabled			25.0	µA
		Power Down, V_{CC} = 3V WDT Disabled			20.0	µA

Notes: 1. "Max" means the highest value where the pin is guaranteed to be read as low
2. "Min" means the lowest value where the pin is guaranteed to be read as high
3. Minimum V_{CC} for Power Down is 2V.

Typical characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factor is the operating voltage, as the frequency of ATtiny22L is also a function of the operationg voltage.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L x V_{CC} x f where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

Figure 35. Active Supply Current vs. V_{CC}

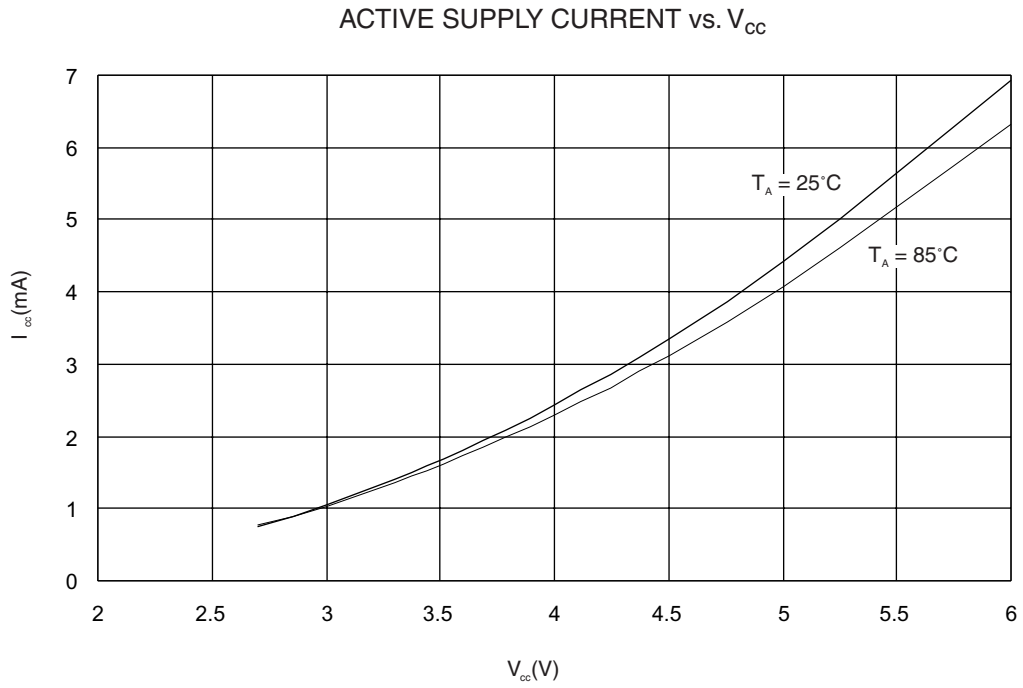


Figure 36. Idle Supply Current vs. V_{CC}

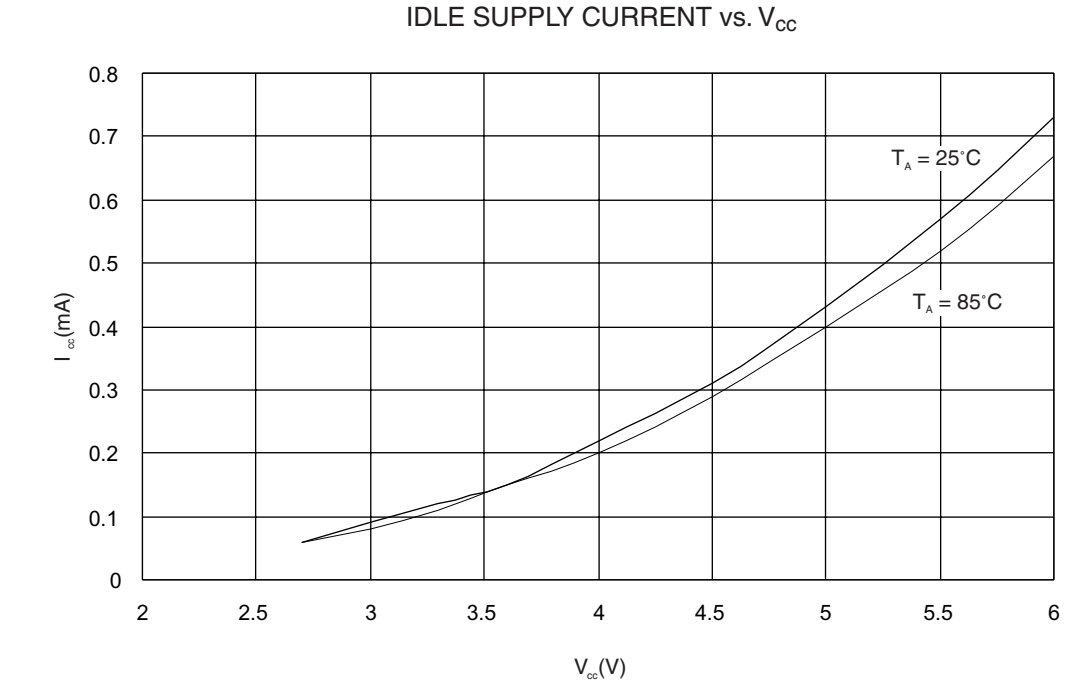


Figure 37. Power Down Supply Current vs. V_{CC}

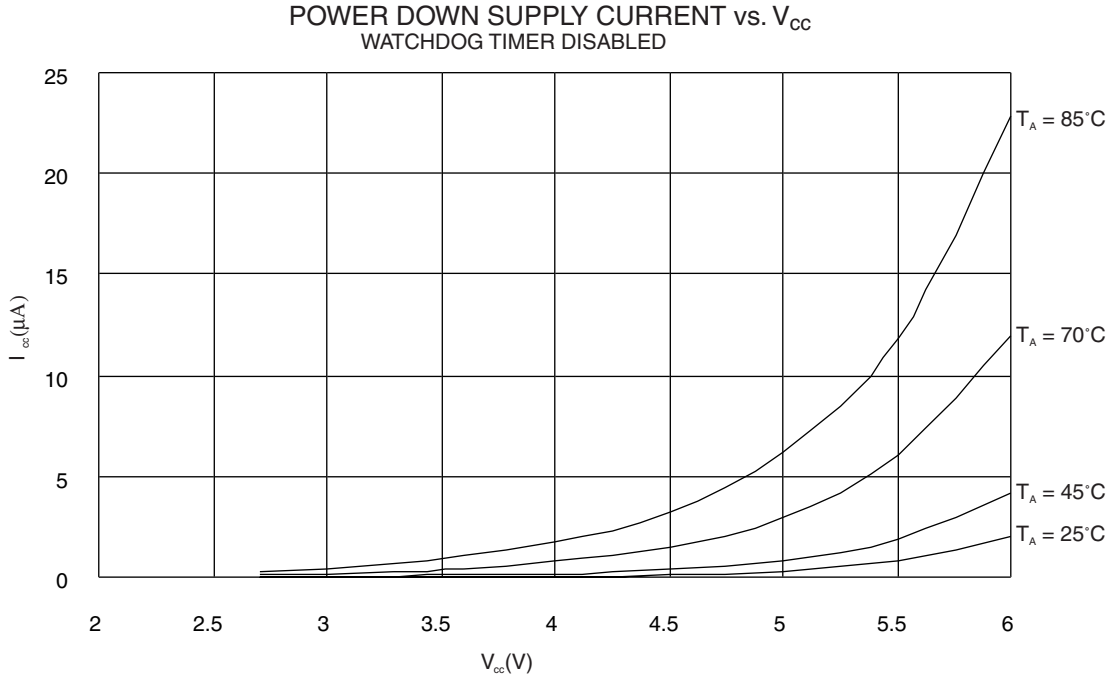


Figure 38. Power Down Supply Current vs. V_{CC}

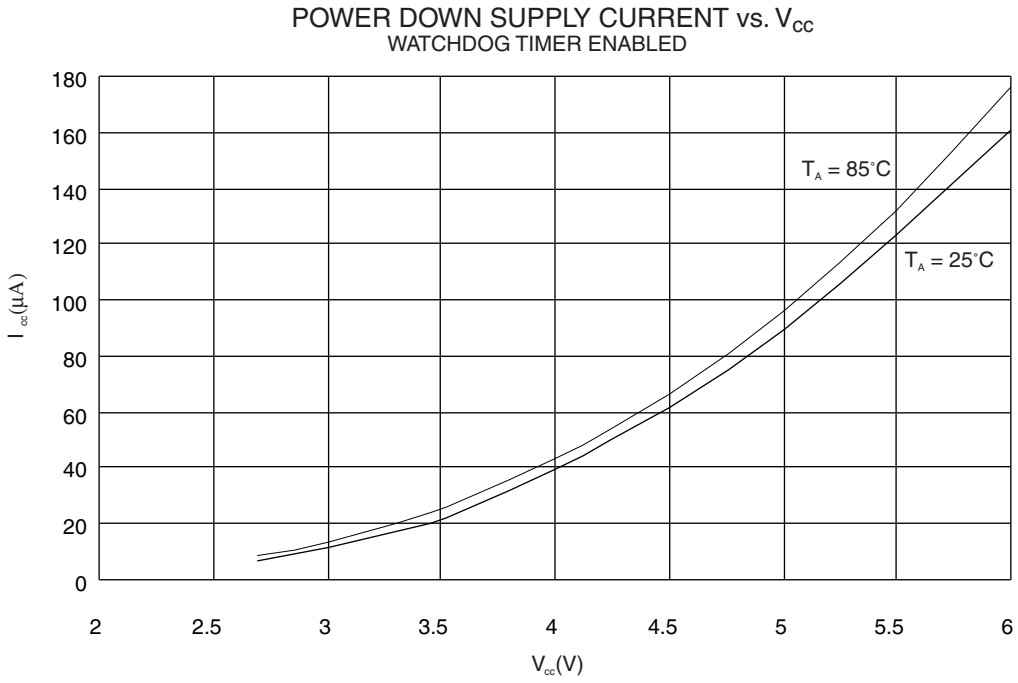
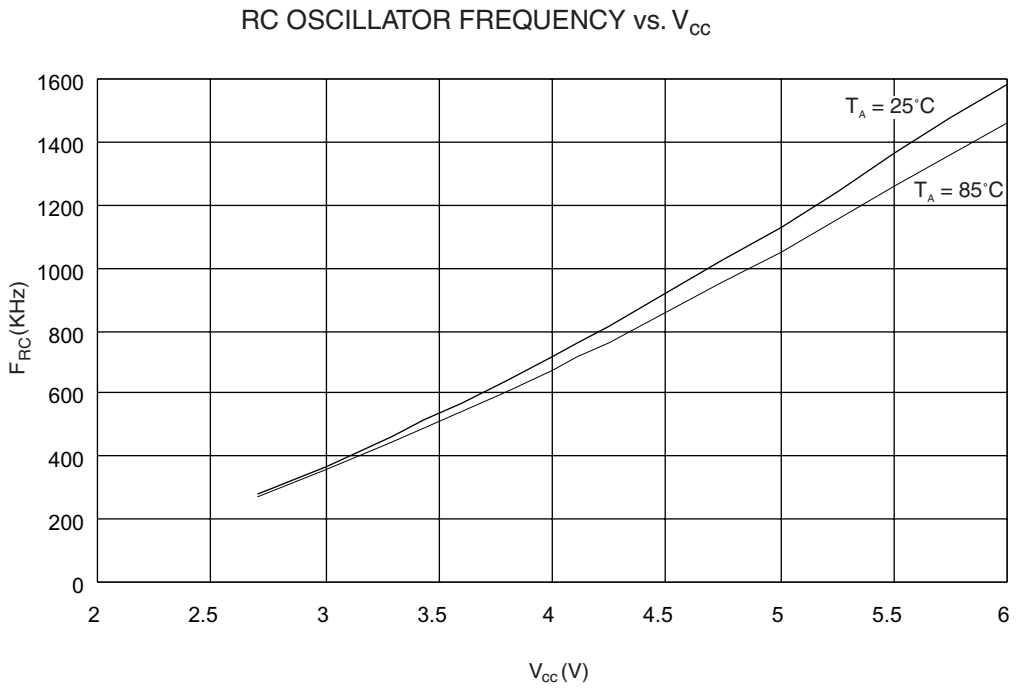


Figure 39. Oscillator Frequency vs. V_{CC}



Note: The frequency of the RC-oscillator may be ±10% off the typical value for a given temperature and V_{CC}.

Sink and source capabilities of I/O ports are measured on one pin at a time.

Figure 40. Pull-Up Resistor Current vs. Input Voltage

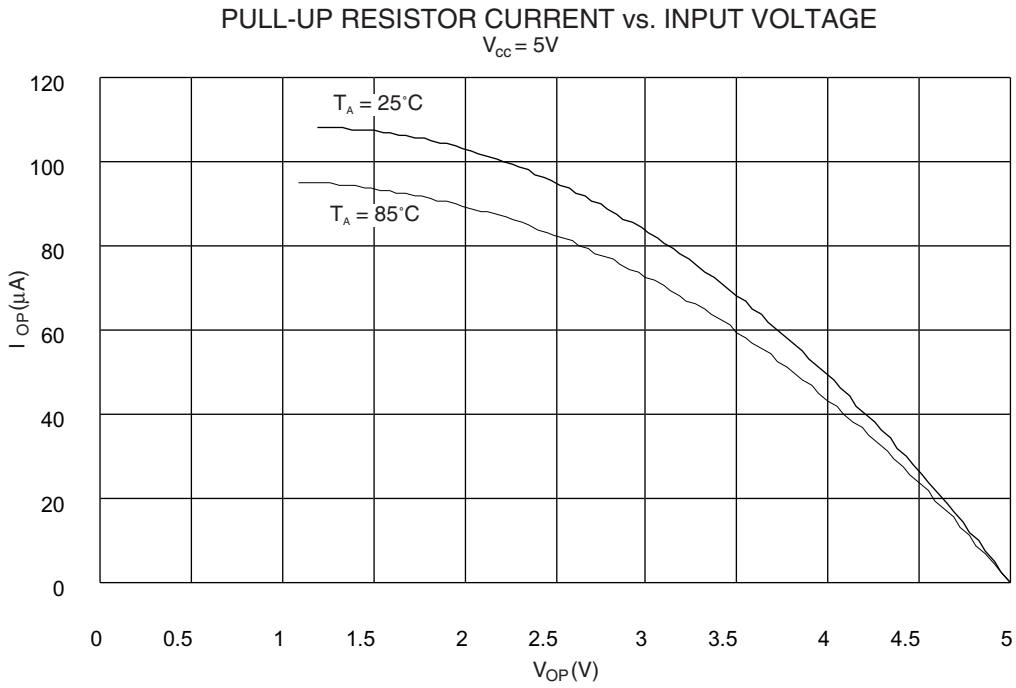


Figure 41. Pull-Up Resistor Current vs. Input Voltage

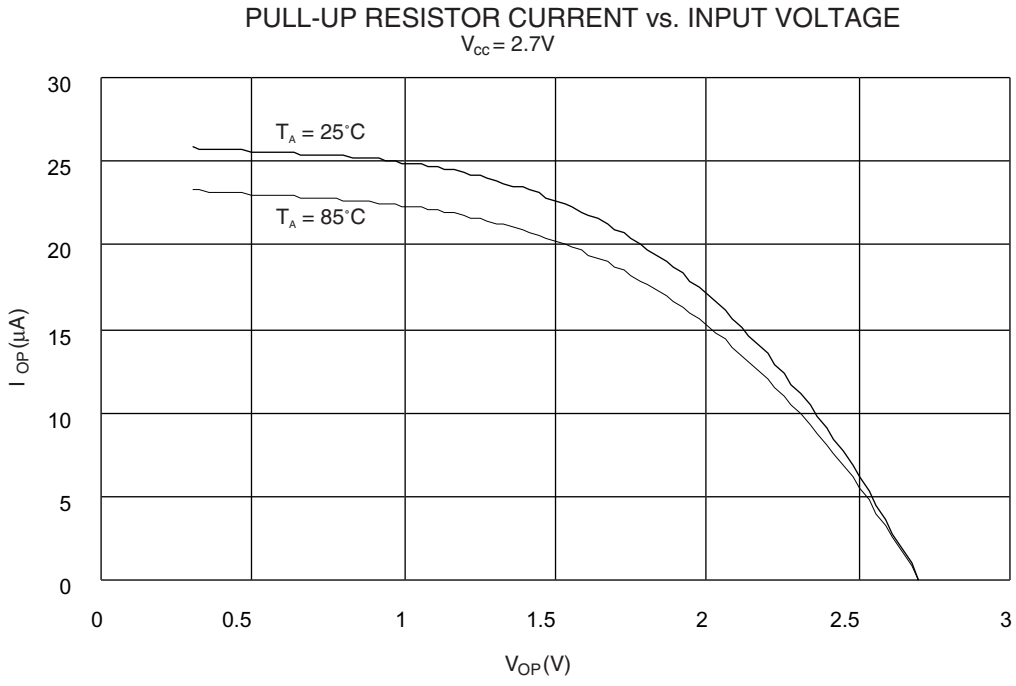


Figure 42. I/O Pin Sink Current vs. Output Voltage

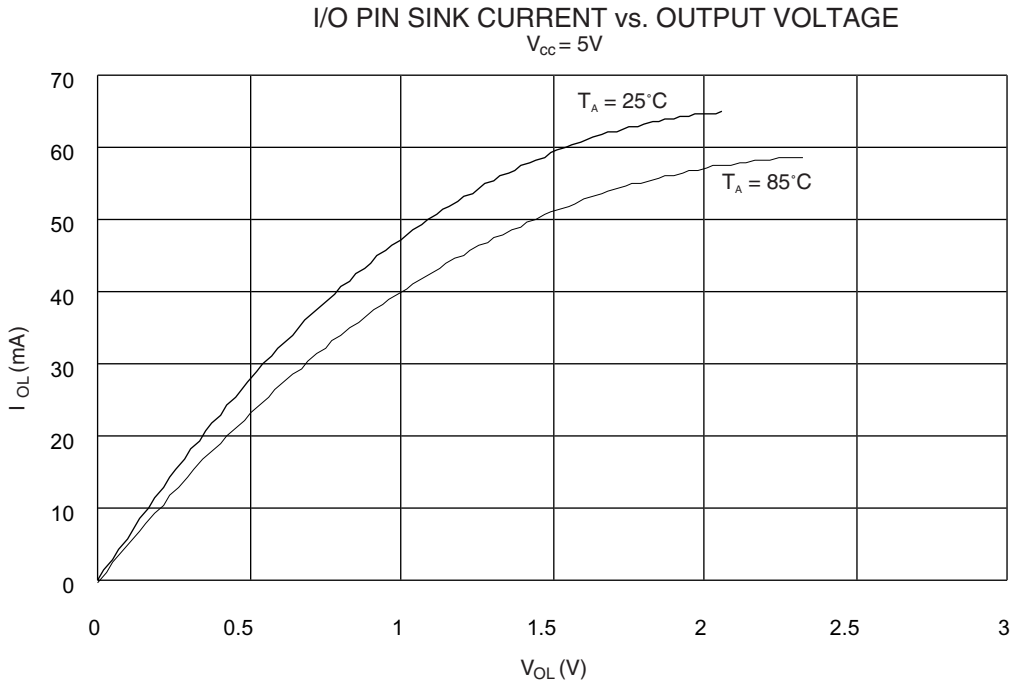


Figure 43. I/O PIn Source Current vs. Output Voltage

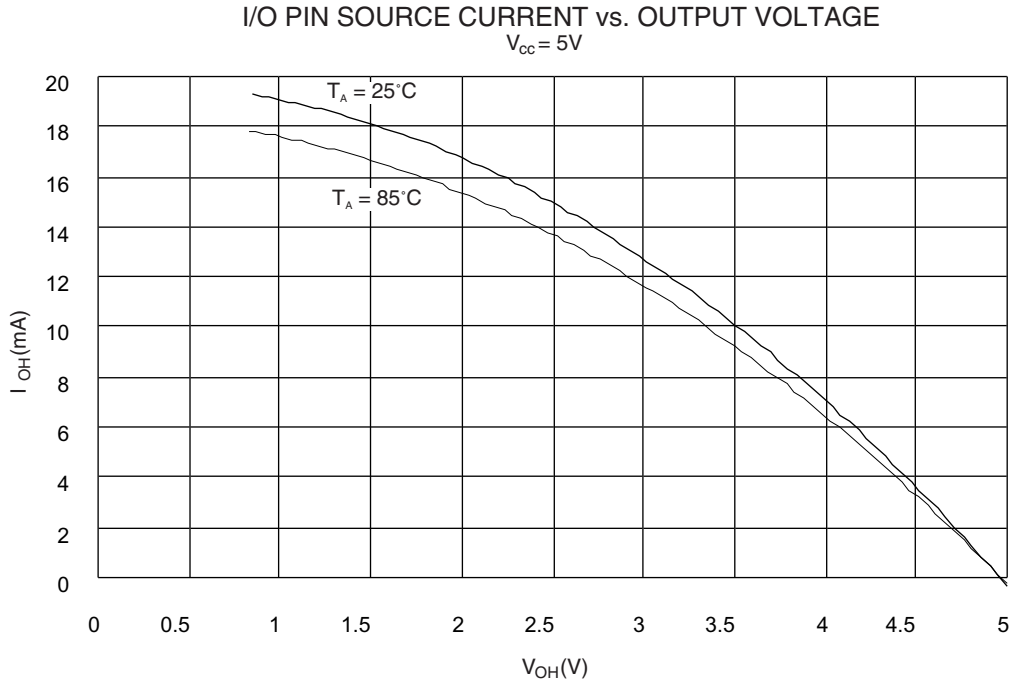


Figure 44. I/O Pin Sink Current vs. Output Voltage

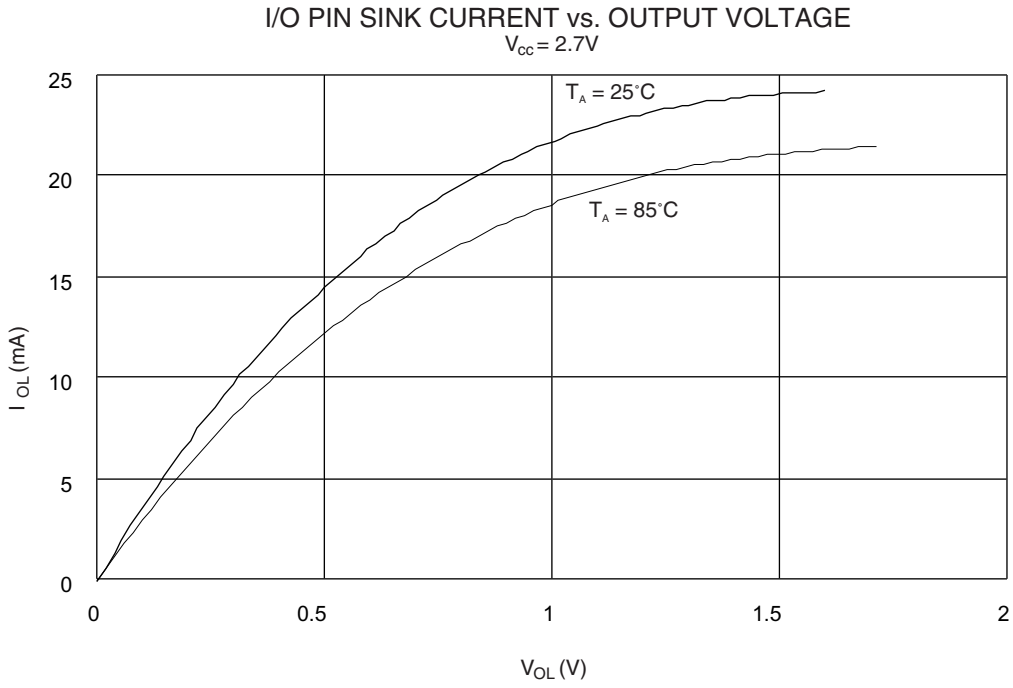


Figure 45. I/O Pin Source Current vs. Output voltage

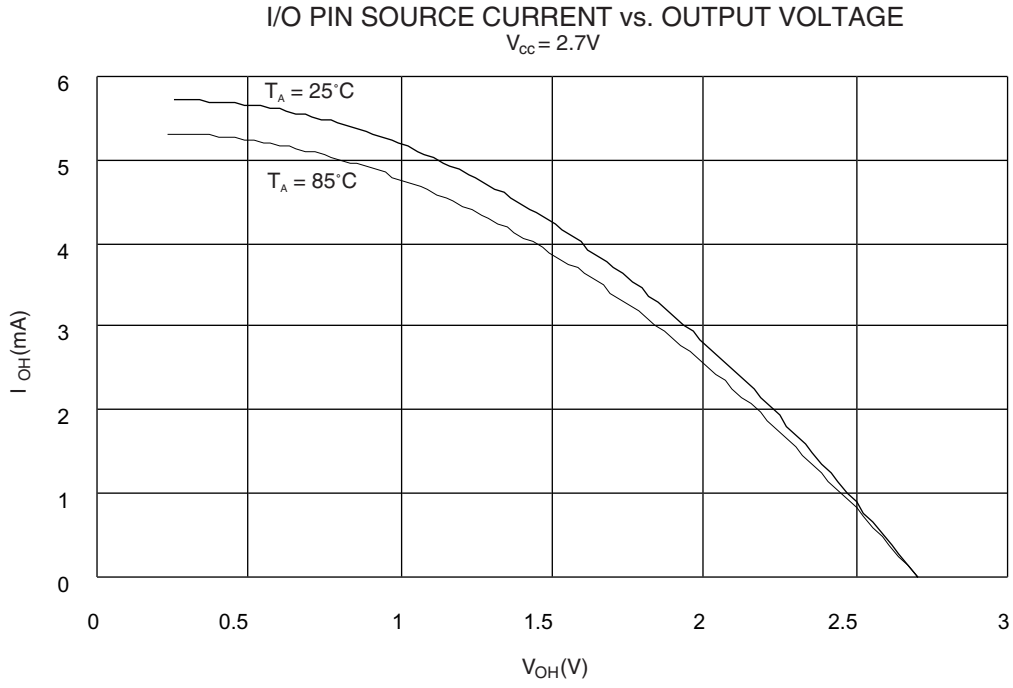


Figure 46. I/O Pin Input Threshold Voltage vs. V_{CC}

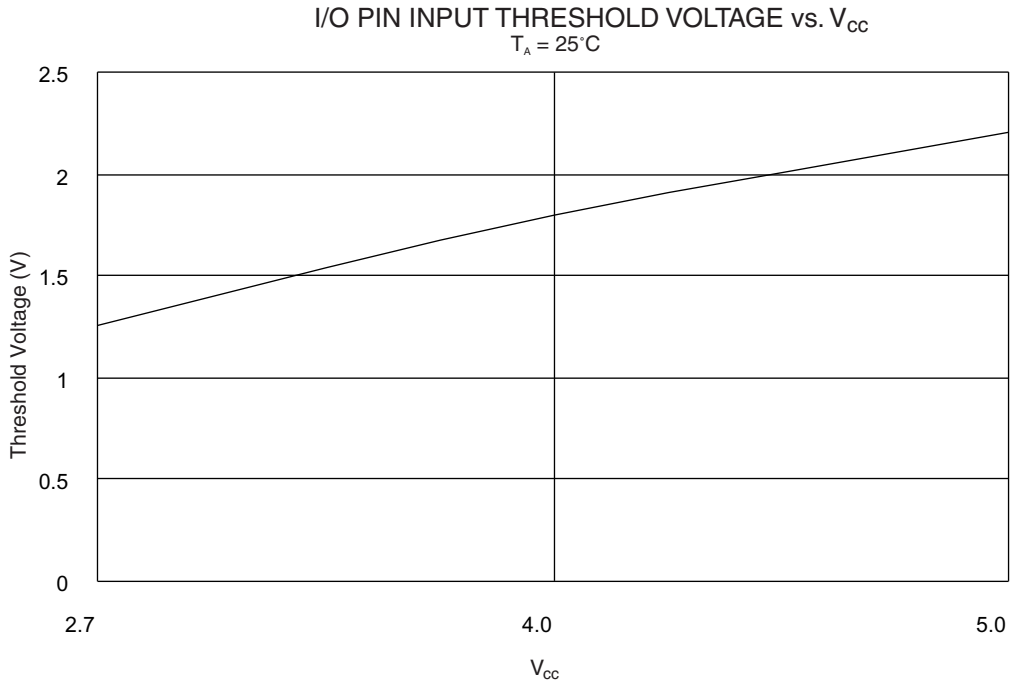
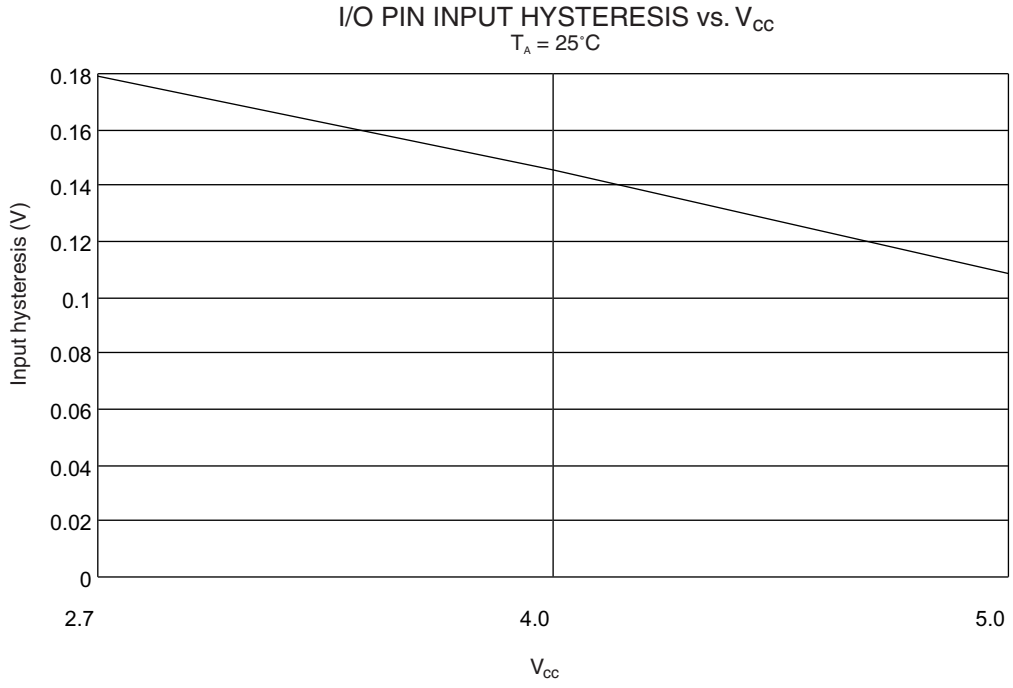


Figure 47. I/O Pin Input Hysteresis vs. V_{CC}



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	page 16
$3E ($5E)	Reserved									
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 17
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	-	INT0	-	-	-	-	-	-	page 23
$3A ($5A)	GIFR	-	INTF0							page 23
$39 ($59)	TIMSK	-	-	-	-	-	-	TOIE0	-	page 23
$38 ($58)	TIFR	-	-	-	-	-	-	TOV0	-	page 24
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	-	-	SE	SM	-	-	ISC01	ISC00	page 24
$34 ($54)	MCUSR	-	-	-	-	-	-	EXTRF	PORF	page 22
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	page 27
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)								page 28
$31 ($51)	Reserved									
$30 ($50)	Reserved									
$2F ($4F)	Reserved									
$2E ($4E)	Reserved									
$2D ($4D)	Reserved									
$2C ($4C)	Reserved									
$2B ($4B)	Reserved									
$2A ($4A)	Reserved									
$29 ($49)	Reserved									
$28 ($48)	Reserved									
$27 ($47)	Reserved									
$26 ($46)	Reserved									
$25 ($45)	Reserved									
$24 ($44)	Reserved									
$23 ($43)	Reserved									
$22 ($42)	Reserved									
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	page 28
$20 ($40)	Reserved									
$1F ($3F)	Reserved									
$1E ($3E)	EEAR	-	EEPROM Address Register							page 30
$1D ($3D)	EEDR	EEPROM Data register								page 30
$1C ($3C)	EECR	-	-	-	-	-	EEMW	EEWE	EERE	page 30
$1B ($3B)	Reserved									
$1A ($3A)	Reserved									
$19 ($39)	Reserved									
$18 ($38)	PORTB	-	-	-	PORTB	PORTB	PORTB	PORTB	PORTB	page 32
$17 ($37)	DDRB	-	-	-	DDB4	DDB3	DDB2	DDB1	DDB0	page 32
$16 ($36)	PINB	-	-	-	PINB4	PINB3	PINB2	PINB1	PINB0	page 32
$15 ($35)	Reserved									
…	Reserved									
$00 ($20)	Reserved									

Notes: 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. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd – Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd – K	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl – K	Z,C,N,V,S	2
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
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 ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF – 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 ← $FF	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 (R(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

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← 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
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
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
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
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep	None	3
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

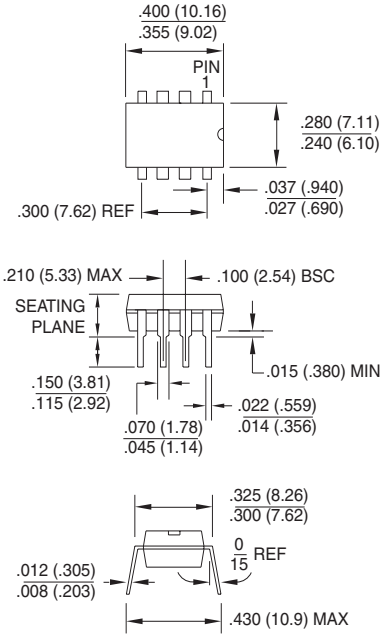
Ordering Information

Power Supply	Speed (MHz)	Ordering Code	Package	Operation Range
2.7 - 6.0V	Internal Osc ~1MHz@5.0V	ATtiny22L-1PC ATtiny22L-1SC	8P3 8S2	Commercial (0°C to 70°C)
		ATtiny22L-1PI ATtiny22L-1SI	8P3 8S2	Industrial (-40°C to 85°C)

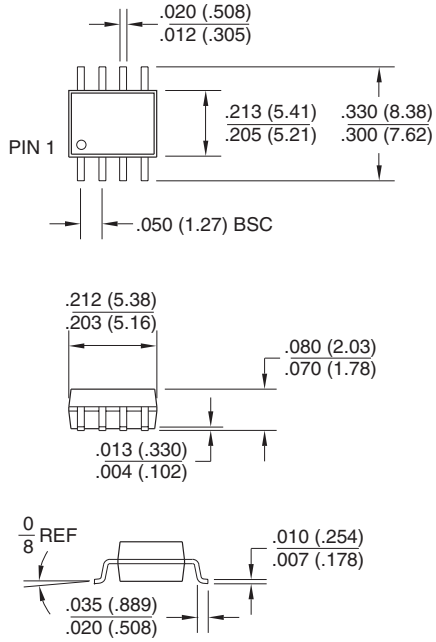
Package Type	
8P3	8-pin, 0.300" Wide, Plastic Dual Inline Package (PDIP)
8S2	8-lead, 0.200" Wide, Plastic Gull-Wing Small Outline (EIAJ SOIC)

Packaging Information

8P3, 8-pin, 0.300" Wide,
Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-001 BA



8S2, 8-lead, 0.200" Wide,
Plastic Gull Wing Small Outline (EIAJ SOIC)
Dimensions in Inches and (Millimeters)



Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

1273B–02/00/xM