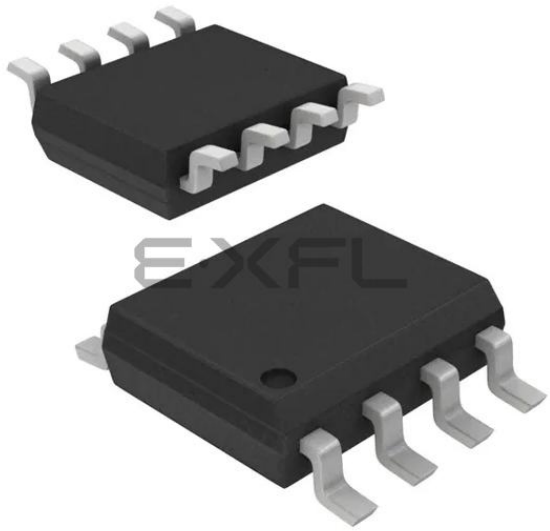


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

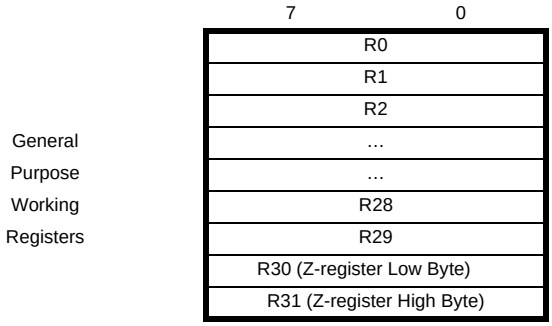
| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 1.6MHz |
| Connectivity | - |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 6 |
| Program Memory Size | 1KB (512 x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 64 x 8 |
| RAM Size | - |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 4x10b |
| Oscillator Type | Internal |
| Operating Temperature | 0°C ~ 70°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 8-SOIC (0.209", 5.30mm Width) |
| Supplier Device Package | 8-SOIC |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/attiny15l-1sc |

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.

## The General Purpose Register File

Figure 3 shows the structure of the 32 general purpose registers in the CPU.

**Figure 3.** 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, and 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.

Registers 30 and 31 form a 16-bit pointer (the Z-pointer) which is used for indirect Flash memory and Register File access. When the Register File is accessed, the contents of R31 is discarded by the CPU.

## The 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. Some microcontrollers in the AVR product family feature a hardware multiplier in the arithmetic part of the ALU.

## The Flash Program Memory

The ATtiny15L contains 1K byte On-chip, In-System Programmable Flash memory for program storage. Since all instructions are single 16-bit words, the Flash is organized as 512 x 16 words. The Flash memory has an endurance of at least 1,000 write/erase cycles.

The ATtiny15L Program Counter is nine bits wide, thus addressing the 512 words Flash Program memory.

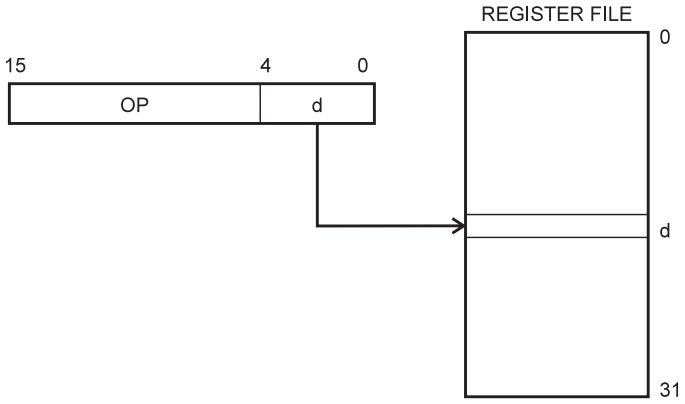
See page 54 for a detailed description on Flash memory programming.

## The Program and Data Addressing Modes

The ATtiny15L AVR RISC Microcontroller supports powerful and efficient addressing modes. This section describes the various addressing modes supported in the ATtiny15L. 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

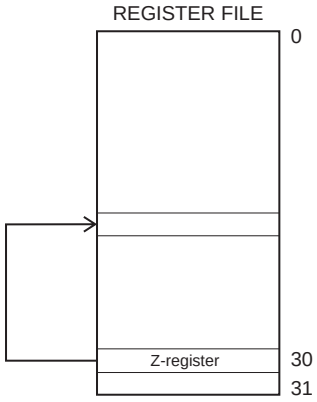
**Figure 4.** Direct Single-register Addressing




The operand is contained in register d (Rd).

### Register Indirect

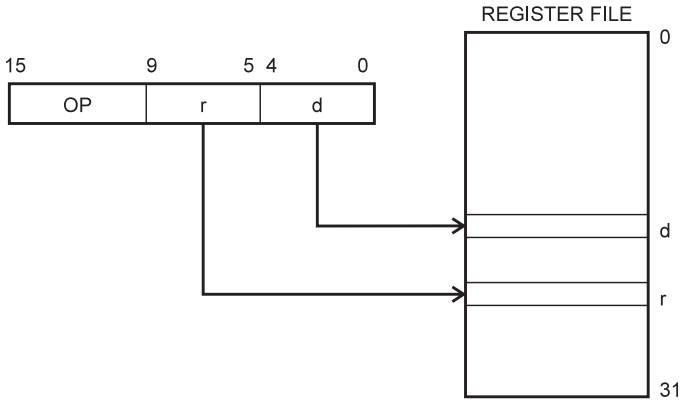
**Figure 5.** Indirect Register Addressing




The register accessed is the one pointed to by the Z-register low byte (R30).

### Register Direct, Two Registers Rd and Rr

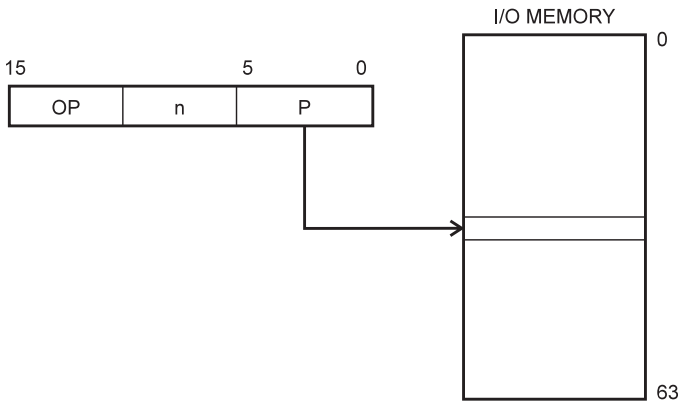
**Figure 6.** Direct Register Addressing, Two Registers



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

**I/O Direct**

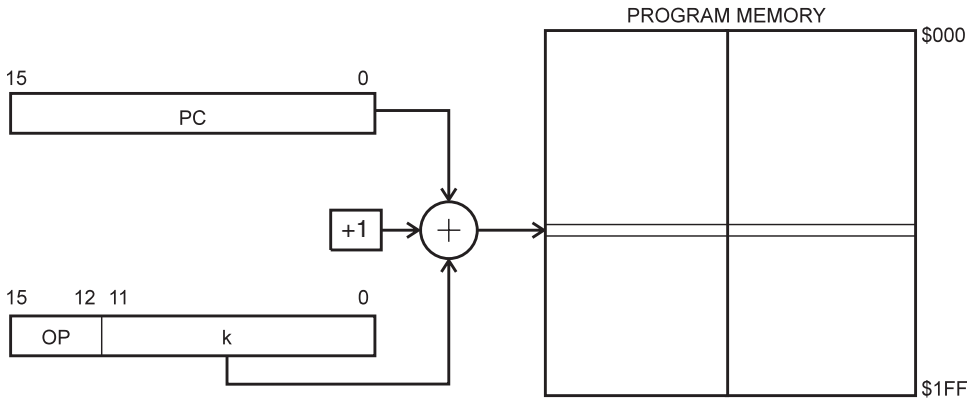
**Figure 7.** I/O Direct Addressing




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

**Relative Program Addressing, RJMP and RCALL**

**Figure 8.** Relative Program Memory Addressing




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

**Constant Addressing using the LPM Instruction**

**Figure 9.** Code Memory Constant Addressing

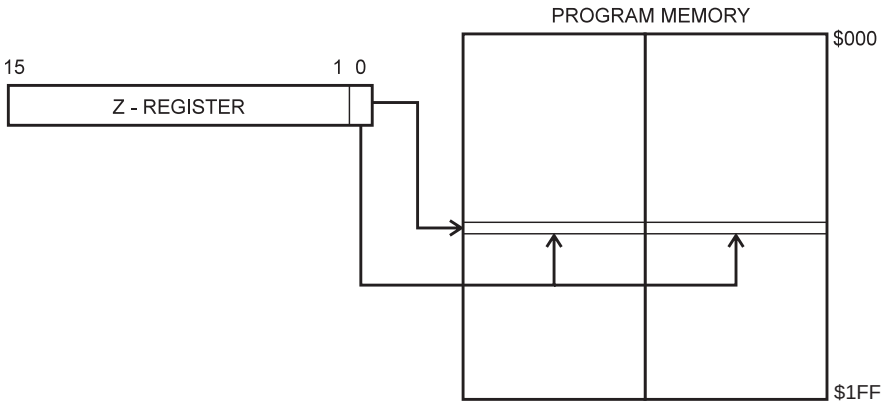
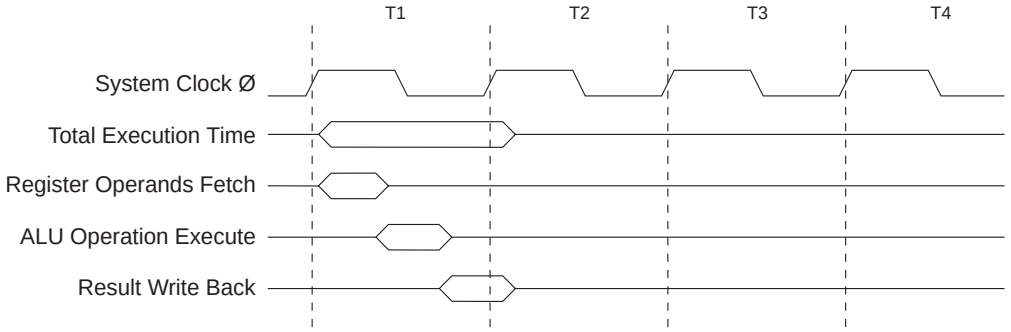


**Figure 11.** Single Cycle ALU Operation




## I/O Memory

The I/O space definition of the ATtiny15L is shown in Table 2.

**Table 2.** ATtiny15L I/O Space[1]

| Address Hex | Name | Function |
|---|---|---|
| $3F | SREG | Status Register |
| $3B | GIMSK | General Interrupt Mask Register |
| $3A | GIFR | General Interrupt Flag Register |
| $39 | TIMSK | Timer/Counter Interrupt Mask Register |
| $38 | TIFR | Timer/Counter Interrupt Flag Register |
| $35 | MCUCR | MCU Control Register |
| $34 | MCUSR | MCU Status Register |
| $33 | TCCR0 | Timer/Counter0 Control Register |
| $32 | TCNT0 | Timer/Counter0 (8-bit) |
| $31 | OSCCAL | Oscillator Calibration Register |
| $30 | TCCR1 | Timer/Counter1 Control Register |
| $2F | TCNT1 | Timer/Counter1 (8-bit) |
| $2E | OCR1A | Timer/Counter1 Output Compare Register A |
| $2D | OCR1B | Timer/Counter1 Output Compare Register B |
| $2C | SFIOR | Special Function I/O Register |
| $21 | WDTCR | Watchdog Timer Control Register |
| $1E | EEAR | EEPROM Address Register |
| $1D | EEDR | EEPROM Data Register |
| $1C | EECR | EEPROM Control Register |
| $18 | PORTB | Data Register, Port B |
| $17 | DDRB | Data Direction Register, Port B |
| $16 | PINB | Input Pins, Port B |
| $08 | ACSR | Analog Comparator Control and Status Register |
| $07 | ADMUX | ADC Multiplexer Select Register |

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result after the different arithmetic and logic operations. See the Instruction Set description for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result after the different arithmetic and logic operations. See the Instruction Set description for detailed information.

- **Bit 0 – C: Carry Flag**

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

## Reset and Interrupt Handling

The ATtiny15L provides eight interrupt sources. These interrupts and the separate Reset Vector each have a separate Program Vector in the Program memory space. All the interrupts are assigned individual enable bits that 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 3. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 (the External Interrupt Request 0), etc.

**Table 3.** Reset and Interrupt Vectors

| Vector No. | Program Address | Source | Interrupt Definition |
|---|---|---|---|
| 1 | $000 | RESET | External Reset, Power-on Reset, Brown-out Reset, and Watchdog Reset |
| 2 | $001 | INT0 | External Interrupt Request 0 |
| 3 | $002 | I/O Pins | Pin Change Interrupt |
| 4 | $003 | TIMER1, COMPA | Timer/Counter1 Compare Match A |
| 5 | $004 | TIMER1, OVF | Timer/Counter1 Overflow |
| 6 | $005 | TIMER0, OVF | Timer/Counter0 Overflow |
| 7 | $006 | EE_RDY | EEPROM Ready |
| 8 | $007 | ANA_COMP | Analog Comparator |
| 9 | $008 | ADC | ADC Conversion Complete |

The most typical and general 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     PIN_CHANGE   ; Pin change handler
$003              rjmp     TIM1_CMP     ; Timer1 compare match
$004              rjmp     TIM1_OVF     ; Timer1 overflow handler
$005              rjmp     TIM0_OVF     ; Timer0 overflow handler
$006              rjmp     EE_RDY       ; EEPROM Ready handler
$007              rjmp     ANA_COMP     ; Analog Comparator handler
$008              rjmp     ADC          ; ADC Conversion Handler
;
$009      MAIN:   <instr>  xxx          ; Main program start
  …         …       …      …
```

## ATtiny15L Reset Sources

The ATtiny15L has four sources of Reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold ($V_{POR}$).
- External Reset. The MCU is reset when a low-level is present on the $\overline{\text{RESET}}$ pin for more than 500 ns.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires, and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage $V_{CC}$ is below the Brown-out Reset threshold ($V_{BOT}$).

During Reset, all I/O Registers are then 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 12 shows the reset logic. Table 4 and Table 5 define the timing and electrical parameters of the reset circuitry. Note that the Register File is unchanged by a reset.

**Figure 13.** "MCU Start-up, $\overline{\text{RESET}}$ Tied to $V_{CC}$

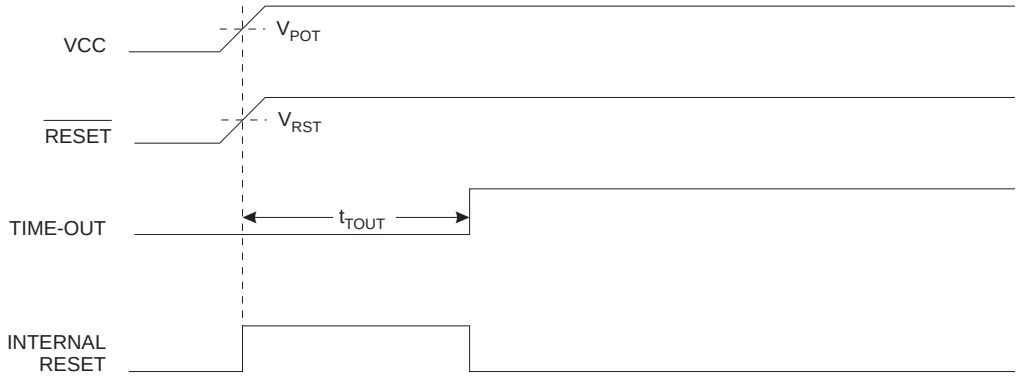
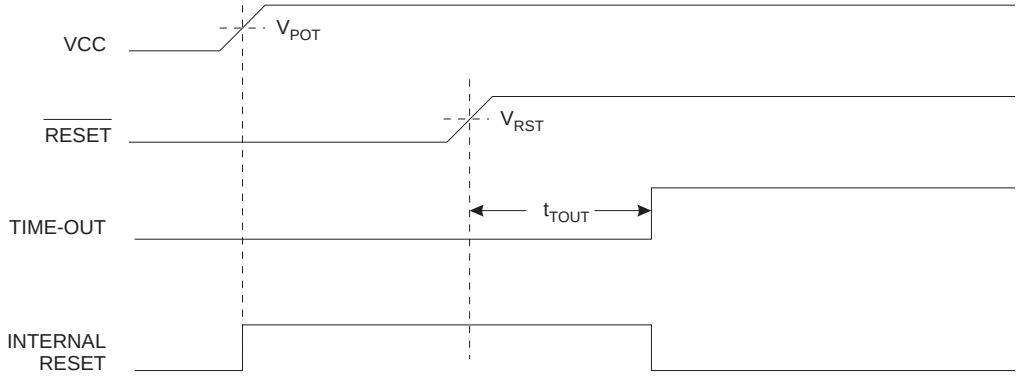


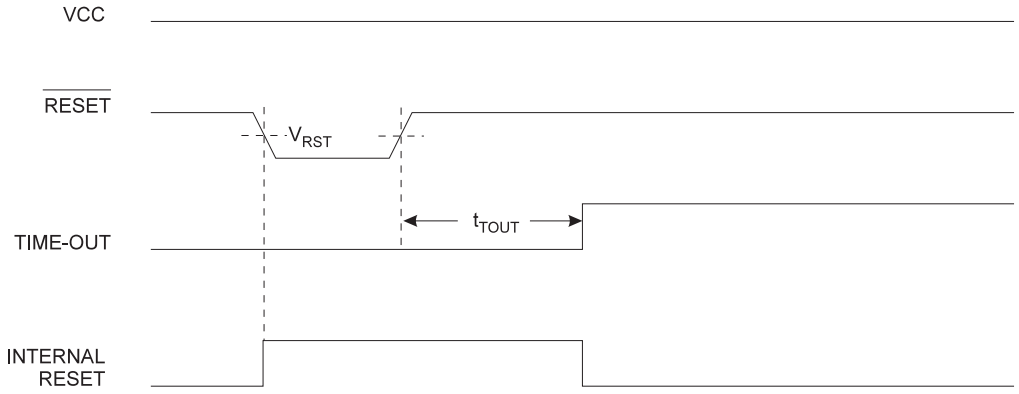

**Figure 14.** MCU Start-up, $\overline{\text{RESET}}$ Extended Externally




## External Reset

An External Reset is generated by a low-level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 500 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 15.** External Reset during Operation



- **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 “1” to the flag. When the SREG I-bit, 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 ATtiny15L and always reads as 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, a pin change, 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).

**Pin Change Interrupt**

The pin change interrupt is triggered by any change in logical value on any input or I/O pin. Change on pins PB4..0 will always cause an interrupt. Change on pin PB5 will cause an interrupt if the pin is configured as input or I/O, as described in the section “Pin Descriptions” on page 4. Observe that, if enabled, the interrupt will trigger even if the changing pin is configured as an output. This feature provides a way of generating a software interrupt. Also observe that the pin change interrupt will trigger even if the pin activity triggers another interrupt, for example the external interrupt. This implies that one external event might cause several interrupts. The values on the pins are sampled before detecting edges. If pin change interrupt is enabled, pulses that last longer than one CPU clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt.

**The 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 | **–** | **PUD** | **SE** | **SM1** | **SM0** | **–** | **ISC01** | **ISC00** | **MCUCR** |
| Read/Write | R | R/W | R/W | R/W | R/W | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7 – Res: Reserved Bit**

This bit is a reserved bit in the ATtiny15L and always reads as zero.

- **Bit 6- PUD: Pull-up Disable**

This PUD bit must be set (one) to disable internal pull-up registers at Port B.

- **Bit 5 – SE: Sleep Enable**

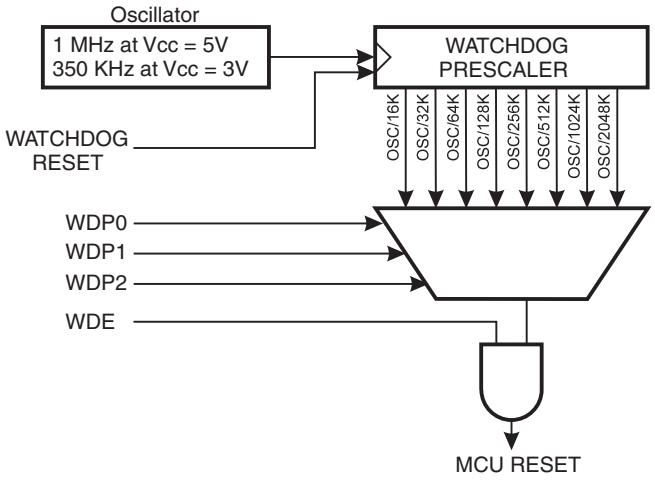
The SE bit must be set (one) to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer’s purpose, it is recommended to set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

## The Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator that runs at 1 MHz. This is the typical value at $V_{CC}$ = 5V. See “Typical Characteristics” on page 66 for typical values at other $V_{CC}$ levels. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted from 16 to 2,048 ms, as shown in Table 15. 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 ATtiny15L resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 17.

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 23.** Watchdog Timer




### The Watchdog Timer Control Register – WDTCR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $21 | – | – | – | 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 ATtiny15L and will always read as zero.

- **Bit 4 – WDTOE: Watchdog 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: Watchdog 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 be cleared only when the WDTOE bit is set (one). To disable an enabled Watchdog Timer, the following procedure must be followed:

## EEPROM Read/Write Access

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

The write access time is in the range of 4.6 - 8.2 ms, depending on the frequency of the calibrated RC Oscillator. See Table 16 for details. A self-timing function however, lets the user software detect when the next byte can be written. If the user code contains code that writes the EEPROM, some precautions must be taken. In heavily filtered power supplies, $V_{CC}$ is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. CPU operation under these conditions is likely to cause the Program Counter to perform unintentional jumps and eventually execute the EEPROM write code. To secure EEPROM integrity, the user is advised to use an external under-voltage reset circuit in this case.

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

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

### The EEPROM Address Register – EEAR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $1E | – | – | EEAR5 | EEAR4 | EEAR3 | EEAR2 | EEAR1 | EEAR0 | EEAR |
| Read/Write | R | R | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial vAlue | 0 | 0 | X | X | X | X | X | X | |

- **Bit 7, 6 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny15L and will always read as zero.

- **Bit 5..0 – EEAR5..0: EEPROM Address**

The EEPROM Address Register (EEAR) specifies the EEPROM address in the 64 bytes EEPROM space. The EEPROM data bytes are addresses linearly between 0 and 63. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

### The EEPROM Data Register – EEDR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $1D | 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.

mode, the ADC is constantly sampling and updating the ADC Data Register. The ADFR bit in ADCSR selects between the two available modes.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSR. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering Power-saving sleep modes.

A conversion is started by writing a logical “1” to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be set to zero by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

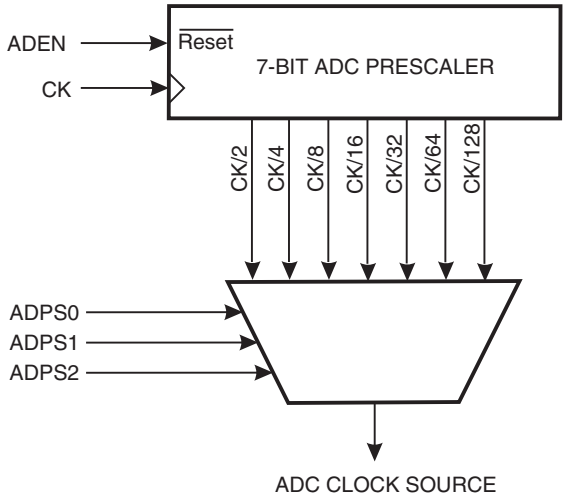
The ADC generates a 10-bit result, which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right-adjusted, but can optionally be presented left-adjusted by setting the ADLAR bit in ADMUX.

If the result is left-adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to data registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt, which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

## Prescaling and Conversion Timing

**Figure 26.** ADC Prescaler




The successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz. Using a higher input frequency will affect the conversion accuracy, see “ADC Characteristics” on page 50. The ADC module contains a prescaler, which divides the system clock to an acceptable ADC clock frequency.

The ADPSn bits in ADCSR are used to generate a proper ADC clock input frequency from any CK frequency above 100 kHz. The prescaler starts counting from the moment

the ADC is switched on by setting the ADEN bit in ADCSR. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

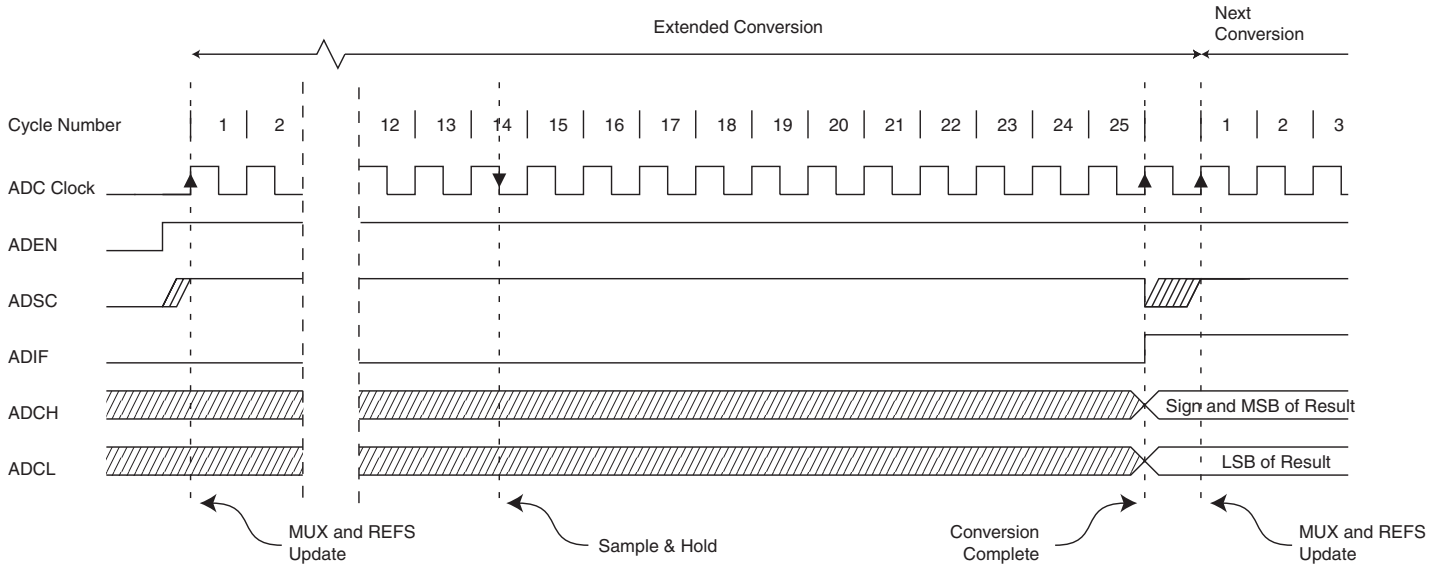
When initiating a conversion by setting the ADSC bit in ADCSR, the conversion starts at the following rising edge of the ADC clock cycle. If differential channels are selected, the conversion will only start at every other rising edge of the ADC clock cycle after ADEN was set.

A normal conversion takes 13 ADC clock cycles. In certain situations, the ADC needs more clock cycles to perform initialization and minimize offset errors. These extended conversions take 25 ADC clock cycles and occur as the first conversion after one of the following events:

- The ADC is switched on (ADEN in ADCSR is set).
- The voltage reference source is changed (the REFS1..0 bits in ADMUX change value).
- A differential channel is selected (MUX2 in ADMUX is “1”). Note that subsequent conversions on the same channel are not extended conversions.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an extended conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge. In Free Running mode, a new conversion will be started immediately after the conversion completes while ADSC remains high. Using Free Running mode and an ADC clock frequency of 200 kHz gives the lowest conversion time, 65 µs, equivalent to 15 kSPS. For a summary of conversion times, see Table 18.

**Figure 27.** ADC Timing Diagram, First Conversion (Single Conversion Mode)



- **Bits 4..3 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny15L and always read as zero.

- **Bits 2..0 – MUX2..MUX0: Analog Channel and Gain Selection Bits 2..0**

The value of these bits selects which analog input is connected to the ADC. In case of differential input (PB3 - PB4), gain selection is also made with these bits. Selecting PB3 as both inputs to the differential gain stage enables offset measurements. Refer to Table 20 for details. If these bits are changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSR is set).

**Table 20.** Input Channel and Gain Selections

| MUX2..0 | Single-ended Input | Positive Differential Input | Negative Differential Input | Gain |
|---|---|---|---|---|
| 000 | ADC0 (PB5) | N/A | | |
| 001 | ADC1 (PB2) | | | |
| 010 | ADC2 (PB3) | | | |
| 011 | ADC3 (PB4) | | | |
| 100(1) | N/A | ADC2 (PB3) | ADC2 (PB3) | 1x |
| 101(1) | | ADC2 (PB3) | ADC2 (PB3) | 20x |
| 110 | | ADC2 (PB3) | ADC3 (PB4) | 1x |
| 111 | | ADC2 (PB3) | ADC3 (PB4) | 20x |

Note: 1. For offset calibration only. See “Operation” on page 42.

### The ADC Control and Status Register – ADCSR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $06 | ADEN | ADSC | ADFR | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 | ADCSR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – ADEN: ADC Enable**

Writing a logical “1” to this bit enables the ADC. By clearing this bit to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, a logical “1” must be written to this bit to start each conversion. In Free Running mode, a logical “1” must be written to this bit to start the first conversion.

When the conversion completes, ADSC returns to zero in Single Conversion mode and stays high in Free Running mode.

Writing a “0” to this bit has no effect.

- **Bit 5 – ADFR: ADC Free Running Select**

When this bit is set (one), the ADC operates in Free Running mode. In this mode, the ADC samples and updates the Data Registers continuously. Clearing this bit (zero) will terminate Free Running mode. If active channels are used (MUX2 in ADMUX set), the

### The ADC Data Register – ADCL and ADCH

*ADLAR = 0*

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| $05 | – | – | – | – | – | – | ADC9 | ADC8 | ADCH |
| $04 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADC1 | ADC0 | ADCL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

*ADLAR = 1*

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| $05 | ADC9 | ADC8 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADCH |
| $04 | ADC1 | ADC0 | – | – | – | – | – | – | ADCL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

When an ADC conversion is complete, the result is found in these two registers. When ADCL is read, the ADC Data Register is not updated until ADCH is read. If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH. The ADLAR bit in ADMUX affects the way the result is read from the registers. If ADLAR is set, the result is left-adjusted. If ADLAR is cleared (default), the result is right-adjusted.

- **ADC9..0: ADC Conversion Result**

These bits represent the result from the conversion. For the differential channel, this is the value after gain adjustment, as indicated in Table 20 on page 47. For single-ended conversion, or if ADLAR or SIGN is zero, $000 represents ground and $3FF represents the selected reference voltage minus one LSB.

### Scanning Multiple Channels

Since change of analog channel always is delayed until a conversion is finished, the Free Running mode can be used to scan multiple channels without interrupting the converter. Typically, the ADC Conversion Complete Interrupt will be used to perform the channel shift. However, the user should take the following fact into consideration:

The interrupt triggers once the result is ready to be read. In Free Running mode, the next conversion will start immediately when the interrupt triggers. If ADMUX is changed after the interrupt triggers, the next conversion has already started, and the old setting is used.

## ADC Noise-canceling Techniques

Digital circuitry inside and outside the ATtiny15L generates EMI, which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. The analog part of the ATtiny15L and all analog components in the application should have a separate analog ground plane on the PCB. This ground plane is connected to the digital ground plane via a single point on the PCB.
2. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If some Port B pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

## ADC Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Resolution | Single-ended Conversion | | 10.0 | | Bits |
| | | Differential Conversion<br>Gain = 1x or 20x | | 8.0 | | Bits |
| | Absolute Accuracy | Single-ended Conversion<br>$V_{REF}$ = 4V<br>ADC Clock = 200 kHz | | 1.0 | 2.0 | LSB |
| | | Single-ended Conversion<br>$V_{REF}$ = 4V<br>ADC Clock = 1 MHz | | 4.0 | | LSB |
| | | Single-ended Conversion<br>$V_{REF}$ = 4V<br>ADC Clock = 2 MHz | | 16.0 | | LSB |
| | Integral Non-linearity | $V_{REF}$ > 2V | | 0.5 | | LSB |
| | Differential Non-linearity | $V_{REF}$ > 2V | | 0.5 | | LSB |
| | Zero Error (Offset) | $V_{REF}$ > 2V | | 1.0 | | LSB |
| | Conversion Time | Free Running Conversion | 65.0 | | 260.0 | µs |
| | Clock Frequency | | 50.0 | | 200.0 | kHz |
| $V_{REF}$ | Reference Voltage | Single-ended Conversion | 2.0 | | $V_{CC}$ | V |
| | | Differential Conversion | 2.0 | | $V_{CC}$ - 0.2 | V |
| $V_{INT}$ | Internal Voltage Reference | | 2.4 | 2.56 | 2.7 | V |
| $R_{REF}$ | Reference Input Resistance | | 6.0 | 10.0 | 13.0 | kΩ |
| $R_{AIN}$ | Analog Input Resistance | | | 100.0 | | MΩ |

## 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 6-bit bi-directional I/O port.

Three data memory address locations are allocated for Port B, one each for the Data Register – PORTB, $18, Data Direction Register – DDRB, $17, and the Port B Input Pins – PINB, $16. The Port B Input Pins address is read-only, while the Data Register and the Data Direction Register are read/write.

Ports PB5..0 have special functions as described in the section “Pin Descriptions” on page 4. If PB5 is not configured as External Reset, it is input with no pull-up or as an open-drain output. All I/O pins have individually selectable pull-ups, which can be overridden with pull-up disable.

The Port B output buffers on PB0 to PB4 can sink 20 mA and thus drive LED displays directly. PB5 can sink 12 mA. When pins PB0 to PB4 are used as inputs and are externally pulled low, they will source current ($I_{IL}$) if the internal pull-ups are activated.

### Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to Vcc or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

### Alternative Functions of Port B

In ATtiny15L four Port B pins – PB2, PB3, PB4, and PB5 – have alternative functions as inputs for the ADC. If some Port B pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion. During Power-down mode and ADC Noise Reduction mode, the Schmitt triggers of the digital inputs are disconnected on these pins. This allows an analog input voltage close to $V_{CC}/2$ to be present during Power-down without causing excessive power consumption. The Port B pins with alternate functions are shown in Table 1 on page 4.

When the pins PB4..0 are used for the alternate function, the DDRB and PORTB Registers have to be set according to the alternate function description. When PB5 is used as External Reset pin, the values in the corresponding DDRB and PORTB bit are ignored.

### The Port B Data Register – PORTB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $18 | – | – | – | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 | PORTB |
| Read/Write | R | R | R | R/W | R/WS | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

### The Port B Data Direction Register – DDRB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $17 | – | – | DDB5 | DDB4 | DDB3 | DDB2 | DDB1 | DDB0 | DDRB |
| Read/Write | R | R | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## High-voltage Serial Programming Algorithm

To program and verify the ATtiny15L in the High-voltage Serial Programming mode, the following sequence is recommended (See instruction formats in Table 25):

1. Power-up sequence:
   Apply 4.5 - 5.5V between $V_{CC}$ and GND. Set PB5 and PB0 to “0” and wait at least 30 µs.
   Set PB3 to “0”. Wait at least 100 ns.
   Apply 12V to PB5 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 PB5 to “0”.
   Turn $V_{CC}$ power off.

When writing or reading serial data to the ATtiny15L, data is clocked on the eigth rising edge of the 16 external clock pulses needed to generate the internal clock. See Figure 31, Figure 32, and Table 26 for an explanation.

**Figure 31.** High-voltage Serial Programming Waveforms

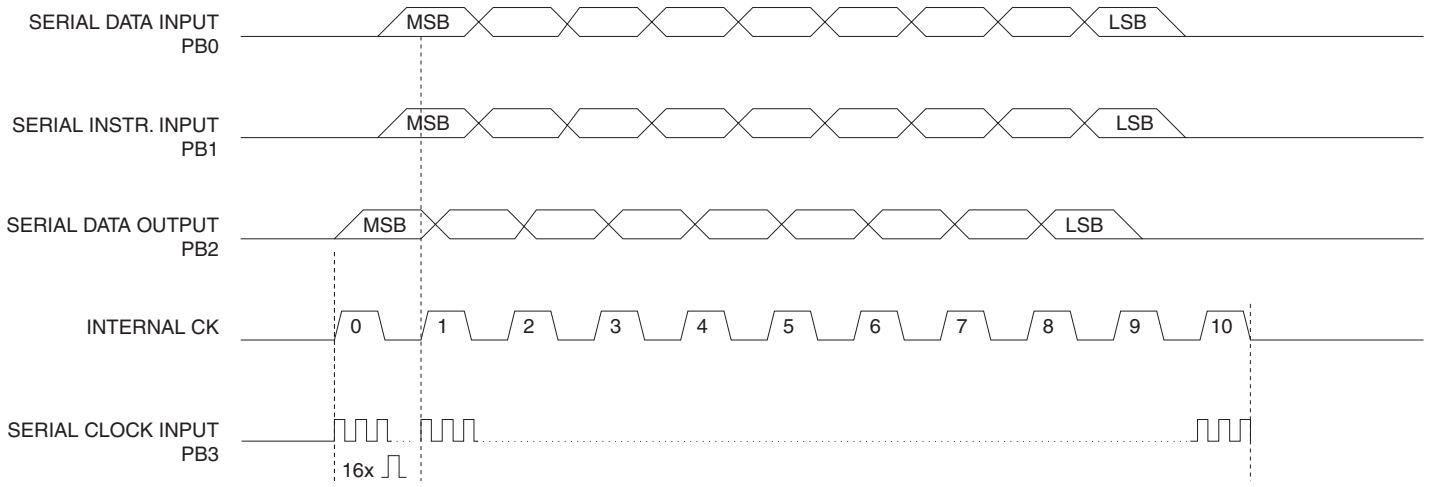


**Table 27.** Low-voltage Serial Programming Instruction Set[1]

| Instruction | Instruction Format | | | | Operation |
|---|---|---|---|---|---|
| | Byte 1 | Byte 2 | Byte 3 | Byte4 | |
| 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 Flash and EEPROM memory arrays. |
| Read Program Memory | 0010 **H**000 | xxxx xxx**a** | **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 | xxxx xxx**a** | **bbbb bbbb** | **iiii iiii** | Write **H** (high or low) data **i** to Program memory at word address **a**:**b**. |
| Read EEPROM Memory | 1010 0000 | xxxx xxxx | xx**bb bbbb** | **oooo oooo** | Read data **o** from EEPROM memory at address **b**. |
| Write EEPROM Memory | 1100 0000 | xxxx xxxx | xx**bb bbbb** | **iiii iiii** | Write data **i** to EEPROM memory at address **b**. |
| 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 Lock Bits | 0101 1000 | xxxx xxxx | xxxx xxxx | xxxx x***21***x | Read Lock bits. “0” = programmed, “1” = unprogrammed. |
| Read Signature Bytes | 0011 0000 | xxxx xxxx | 0000 00**bb** | **oooo oooo** | Read signature byte **o** at address **b**. |
| Write Fuse Bits | 1010 1100 | 101x xxxx | xxxx xxxx | **8765** 11**43** | Set bits **8** - **3** = “0” to program, “1” to unprogram. |
| Read Fuse Bits | 0101 0000 | xxxx xxxx | xxxx xxxx | **8765** xx**43** | Read Fuse bits. “0” = programmed, “1” = unprogrammed. |
| Read Calibration Byte | 0011 1000 | xxxx xxxx | 0000 0000 | **oooo oooo** | |

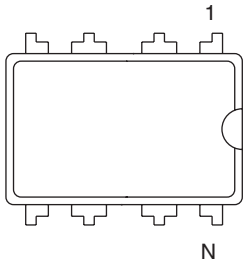
Note: 1. **a** = address high bits
**b** = address low bits
**H** = 0 – low byte, 1 – high byte
**o** = data out
**i** = data in
x = don’t care
***1*** = Lock bit 1
***2*** = Lock bit 2
**3** = CKSEL0 Fuse
**4** = CKSEL1 Fuse
**5** = RSTDISBL Fuse
**6** = SPIEN Fuse
**7** = BODEN Fuse
**8** = BODLEVEL Fuse

# ATtiny15L Instruction Set Summary

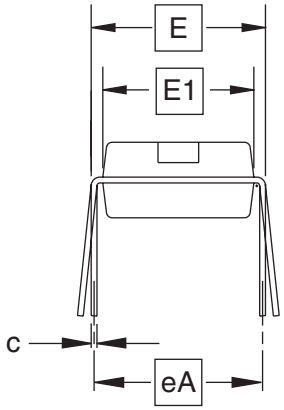
| Mnemonic | Operands | Description | Operation | Flags | # Clocks |
|---|---|---|---|---|---|
| **ARITHMETIC AND LOGIC INSTRUCTIONS** | | | | | |
| ADD | Rd, Rr | Add Two Registers | Rd ← Rd + Rr | Z,C,N,V,H | 1 |
| ADC | Rd, Rr | Add with Carry Two Registers | Rd ← Rd + Rr + C | Z,C,N,V,H | 1 |
| SUB | Rd, Rr | Subtract Two Registers | Rd ← Rd - Rr | Z,C,N,V,H | 1 |
| SUBI | Rd, K | Subtract Constant from Register | Rd ← Rd - K | Z,C,N,V,H | 1 |
| SBC | Rd, Rr | Subtract with Carry Two Registers | Rd ← Rd - Rr - C | Z,C,N,V,H | 1 |
| SBCI | Rd, K | Subtract with Carry Constant from Reg. | Rd ← Rd - K - C | Z,C,N,V,H | 1 |
| 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 • (FFh - 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 |
| RCALL | k | Relative Subroutine Call | PC ← PC + k + 1 | 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 |
| 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 |
| SBRS | Rr, b | Skip if Bit in Register is Set | if (Rr(b) = 1) PC ← PC + 2 or 3 | None | 1/2 |
| SBIC | P, b | Skip if Bit in I/O Register Cleared | if (P(b) = 0) PC ← PC + 2 or 3 | None | 1/2 |
| SBIS | P, b | Skip if Bit in I/O Register is Set | if (P(b) = 1) PC ← PC + 2 or 3 | None | 1/2 |
| BRBS | s, k | Branch if Status Flag Set | if (SREG(s) = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRBC | s, k | Branch if Status Flag Cleared | if (SREG(s) = 0) then PC ← PC + k + 1 | None | 1/2 |
| BREQ | k | Branch if Equal | if (Z = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRNE | k | Branch if Not Equal | if (Z = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRCS | k | Branch if Carry Set | if (C = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRCC | k | Branch if Carry Cleared | if (C = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRSH | k | Branch if Same or Higher | if (C = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRLO | k | Branch if Lower | if (C = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRMI | k | Branch if Minus | if (N = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRPL | k | Branch if Plus | if (N = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRGE | k | Branch if Greater or Equal, Signed | if (N ⊕ V= 0) then PC ← PC + k + 1 | None | 1/2 |
| BRLT | k | Branch if Less Than 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 |
| **DATA TRANSFER INSTRUCTIONS** | | | | | |
| LD | Rd, Z | Load Register Indirect | Rd ← (Z) | None | 2 |
| ST | Z, Rr | Store Register Indirect | (Z) ← Rr | None | 2 |
| MOV | Rd, Rr | Move between Registers | Rd ← Rr | None | 1 |
| LDI | Rd, K | Load Immediate | Rd ← K | None | 1 |
| IN | Rd, P | In Port | Rd ← P | None | 1 |
| OUT | P, Rr | Out Port | P ← Rr | None | 1 |
| LPM | | Load Program Memory | R0 ← (Z) | None | 3 |
| **BIT AND BIT-TEST INSTRUCTIONS** | | | | | |
| SBI | P, b | Set Bit in I/O Register | I/O(P,b) ← 1 | None | 2 |

## Packaging Information

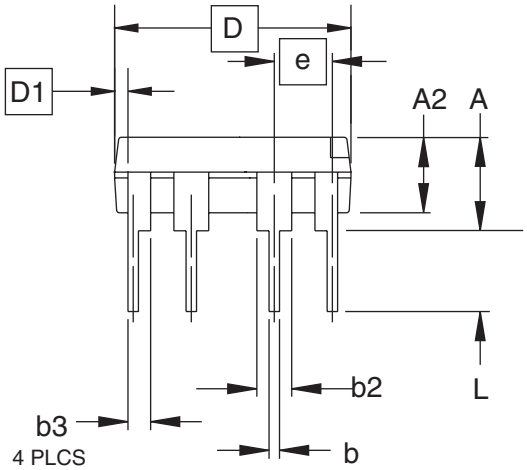
### 8P3




Top View




End View




Side View

**COMMON DIMENSIONS**
(Unit of Measure = inches)

| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | | | 0.210 | 2 |
| A2 | 0.115 | 0.130 | 0.195 | |
| b | 0.014 | 0.018 | 0.022 | 5 |
| b2 | 0.045 | 0.060 | 0.070 | 6 |
| b3 | 0.030 | 0.039 | 0.045 | 6 |
| c | 0.008 | 0.010 | 0.014 | |
| D | 0.355 | 0.365 | 0.400 | 3 |
| D1 | 0.005 | | | 3 |
| E | 0.300 | 0.310 | 0.325 | 4 |
| E1 | 0.240 | 0.250 | 0.280 | 3 |
| e | 0.100 BSC | | | |
| eA | 0.300 BSC | | | 4 |
| L | 0.115 | 0.130 | 0.150 | 2 |

Notes:
1. This drawing is for general information only; refer to JEDEC Drawing MS-001, Variation BA for additional information.
2. Dimensions A and L are measured with the package seated in JEDEC seating plane Gauge GS-3.
3. D, D1 and E1 dimensions do not include mold Flash or protrusions. Mold Flash or protrusions shall not exceed 0.010 inch.
4. E and eA measured with the leads constrained to be perpendicular to datum.
5. Pointed or rounded lead tips are preferred to ease insertion.
6. b2 and b3 maximum dimensions do not include Dambar protrusions. Dambar protrusions shall not exceed 0.010 (0.25 mm).

01/09/02

| | TITLE | DRAWING NO. | REV. |
|---|---|---|---|
| 2325 Orchard Parkway<br>San Jose, CA 95131 | **8P3**, 8-lead, 0.300" Wide Body, Plastic Dual In-line Package (PDIP) | 8P3 | B |