



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of "Quality Parts,Customers Priority,Honest Operation,and Considerate Service",our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: [info@chipsmall.com](mailto:info@chipsmall.com) Web: [www.chipsmall.com](http://www.chipsmall.com)

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



## Features

- High-performance, Low-power AVR<sup>®</sup> 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 8 MIPS Throughput at 8 MHz
  - On-chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 32K Bytes of In-System Self-programmable Flash  
Endurance: 1,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits  
In-System Programming by On-chip Boot Program
  - 1K Byte EEPROM  
Endurance: 100,000 Write/Erase Cycles
  - 2K Bytes Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE Std. 1149.1 Compliant) Interface
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
  - Boundary-Scan Capabilities According to the JTAG Standard
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP and 44-lead TQFP
- Operating Voltages
  - 2.7 - 5.5V (ATmega323L)
  - 4.0 - 5.5V (ATmega323)
- Speed Grades
  - 0 - 4 MHz (ATmega323L)
  - 0 - 8 MHz (ATmega323)



**8-bit AVR<sup>®</sup>  
Microcontroller  
with 32K Bytes  
of In-System  
Programmable  
Flash**

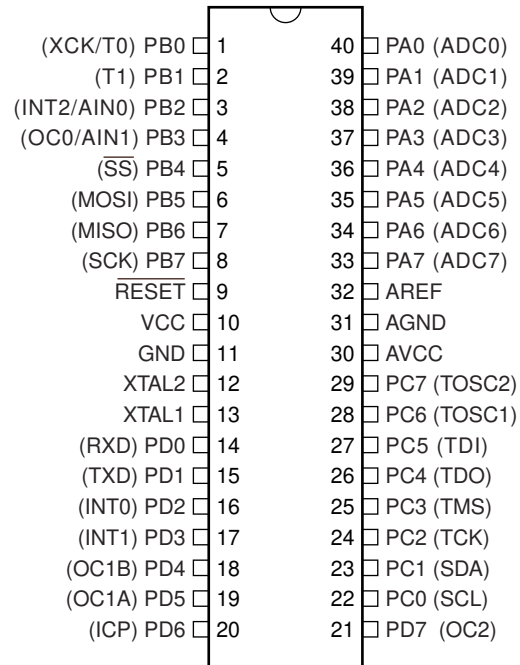
**ATmega323  
ATmega323L**

**Not recommended  
for new designs.  
Use ATmega32.**

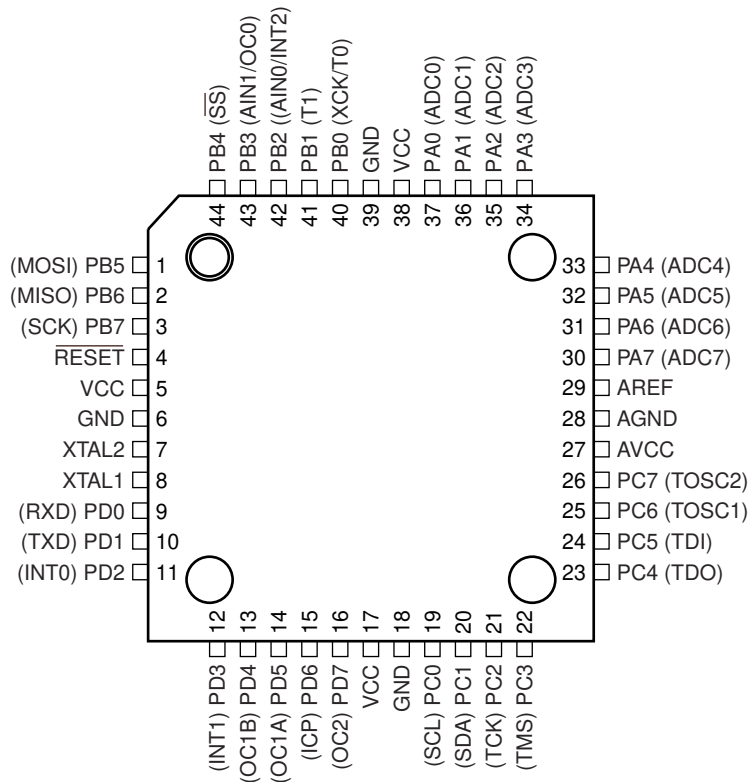


## Pin Configurations

### PDIP



### TQFP



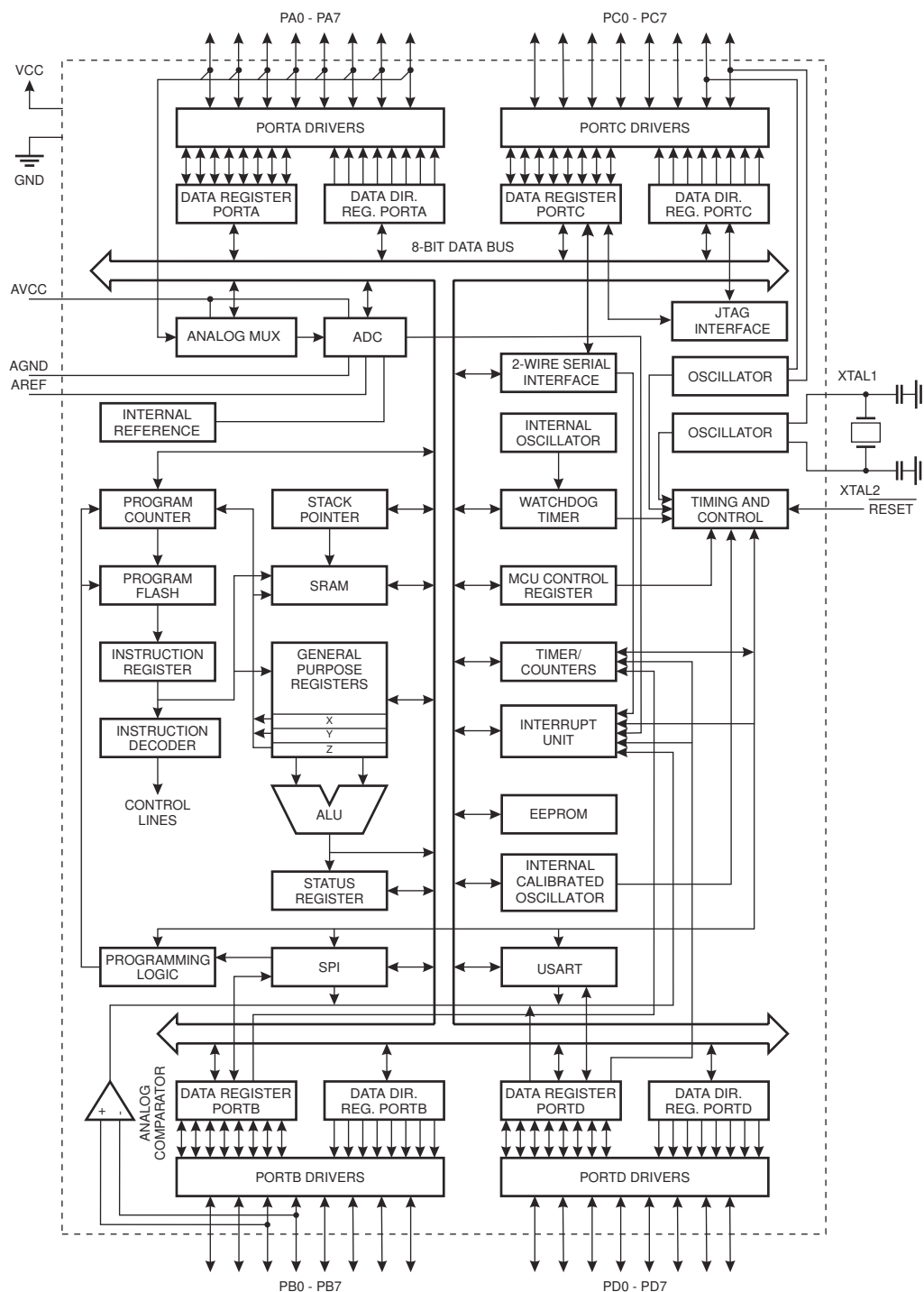


## Overview

The ATmega323 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega323 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

## Block Diagram

Figure 1. Block Diagram





The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega323 provides the following features: 32K bytes of In-System Programmable Flash, 1K bytes EEPROM, 2K bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-Scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC, a programmable Watchdog Timer with internal Oscillator, an SPI serial port, and six 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. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high-density non-volatile memory technology. The On-chip ISP Flash allows the Program memory to be re-programmed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot Program running on the AVR core. The Boot Program can use any interface to download the application program in the Application Flash memory. By combining an 8-bit RISC CPU with In-System Programmable Flash on a monolithic chip, the Atmel ATmega323 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega323 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

**VCC** Digital supply voltage.

**GND** Digital ground.

**Port A (PA7..PA0)** Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers can sink 20 mA and can drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**Port B (PB7..PB0)**

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega323 as listed on page 139.

**Port C (PC7..PC0)**

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers can sink 20 mA. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of the JTAG interface and other special features of the ATmega323 as listed on page 146. If the JTAG interface is enabled, the pull-up resistors on pins PC5 (TDI), PC3 (TMS) and PC2 (TCK) will be activated even if a Reset occurs.

**Port D (PD7..PD0)**

Port D is an 8-bit bidirectional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega323 as listed on page 151.

 **$\overline{\text{RESET}}$** 

Reset input. A low level on this pin for more than 500 ns will generate a Reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a Reset.

**XTAL1**

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

**XTAL2**

Output from the inverting Oscillator amplifier.

**AVCC**

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to  $V_{CC}$ , even if the ADC is not used. If the ADC is used, it should be connected to  $V_{CC}$  through a low-pass filter. See page 127 for details on operation of the ADC.

**AREF**

AREF is the analog reference pin for the A/D Converter. For ADC operations, a voltage in the range 2.56V to AVCC can be applied to this pin.

**AGND**

Analog ground. If the board has a separate analog ground plane, this pin should be connected to this ground plane. Otherwise, connect to GND.

## Clock Options

The device has the following clock source options, selectable by Flash Fuse bits as shown:

**Table 1.** Device Clocking Options Select<sup>(1)</sup>

| Device Clocking Option             | CKSEL3..0   |
|------------------------------------|-------------|
| External Crystal/Ceramic Resonator | 1111 - 1010 |
| External Low-frequency Crystal     | 1001 - 1000 |
| External RC Oscillator             | 0111 - 0101 |
| Internal RC Oscillator             | 0100 - 0010 |
| External Clock                     | 0001 - 0000 |

Note: 1. "1" means unprogrammed, "0" means programmed.

The various choices for each clocking option give different start-up times as shown in Table 6 on page 27.

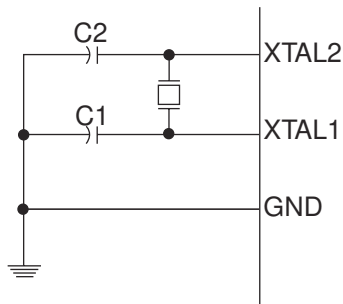
### Internal RC Oscillator

The internal RC Oscillator option is an On-chip Oscillator running at a fixed frequency of nominally 1 MHz. If selected, the device can operate with no external components. See "Calibrated Internal RC Oscillator" on page 41 for information on calibrating this Oscillator.

### Crystal Oscillator

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

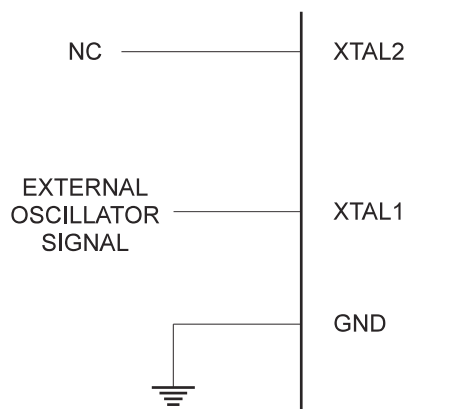
**Figure 2.** Oscillator Connections



## External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 3.

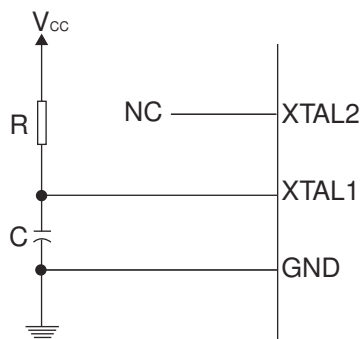
**Figure 3.** External Clock Drive Configuration



## External RC Oscillator

For timing insensitive applications, the external RC configuration shown in Figure 4 can be used. For details on how to choose R and C, see Table 73 on page 215.

**Figure 4.** External RC Configuration



## Timer Oscillator

For the Timer Oscillator pins, PC6(TOSC1) and PC7(TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock source to PC6(TOSC1) is not recommended.



## 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.

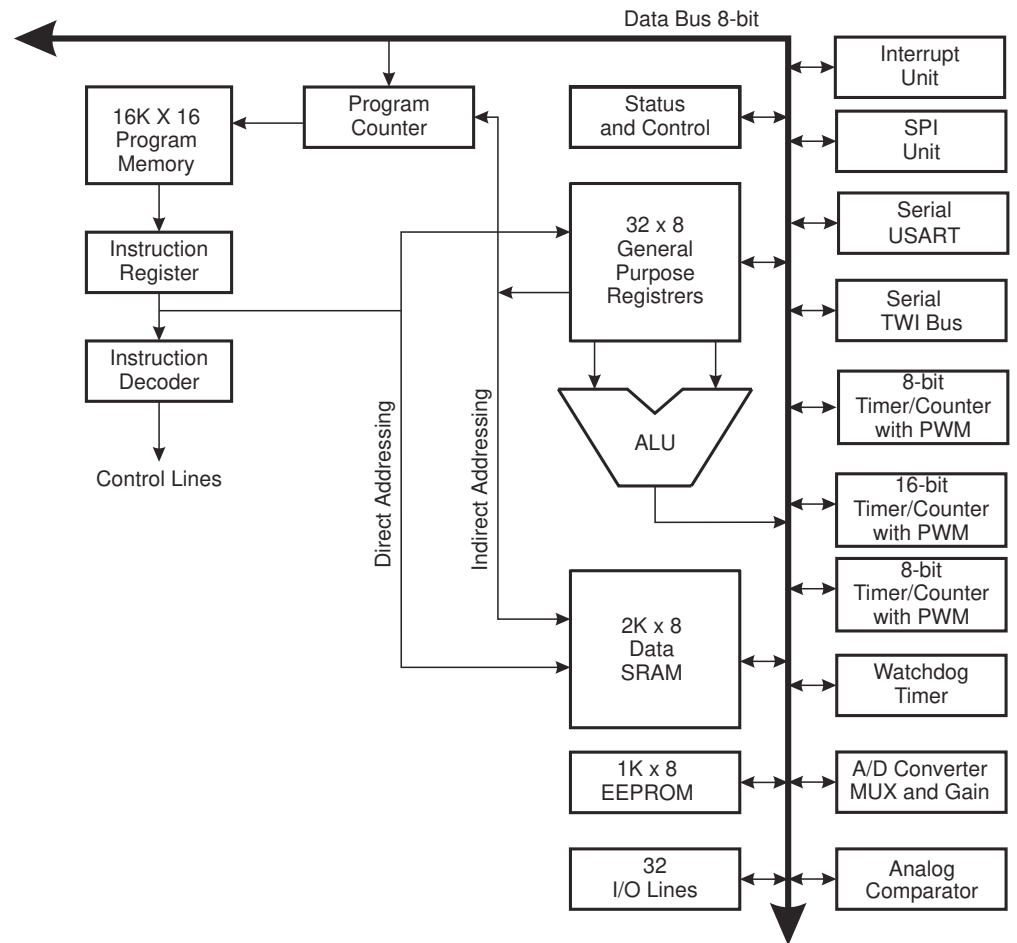
Six of the 32 registers can be used as three 16-bits indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the three address pointers is also used as the address pointer for look up tables in Flash Program memory. These added function registers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

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 5 shows the ATmega323 AVR Enhanced 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 lowest 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.

**Figure 5.** The ATmega323 AVR Enhanced RISC Architecture



The AVR uses a Harvard architecture concept – with separate memories and buses for Program and Data. The Program memory is executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the Program memory. This concept enables instructions to be executed in every clock cycle. The Program memory is In-System Reprogrammable Flash memory.

With the jump and call instructions, the whole 16K 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.

Program Flash memory space is divided in two sections, the Boot Program section (512 to 4K bytes, see page 177) and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section is allowed only in the Boot Program section.

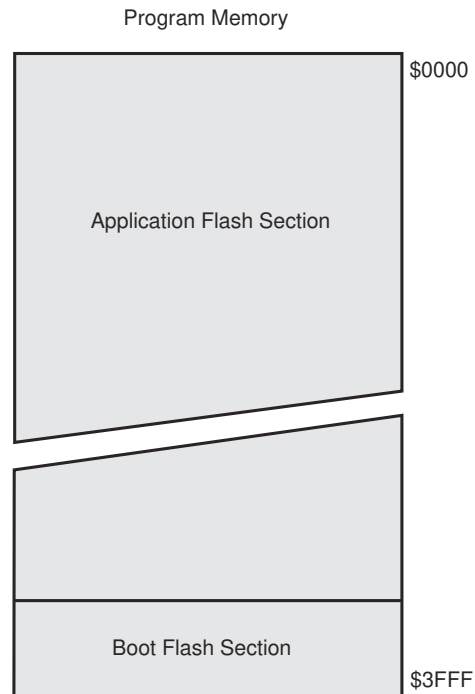
During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset Routine (before subroutines or interrupts are executed). The 12-bit Stack Pointer SP is read/write accessible in the I/O space.

The 2K bytes data SRAM 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.

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

**Figure 6. Memory Maps**



## The General Purpose Register File

Figure 7 shows the structure of the 32 general purpose working registers in the CPU.

**Figure 7.** AVR CPU General Purpose Working Registers

|  | 7   | 0 | Addr. |                      |
|--|-----|---|-------|----------------------|
| General<br>Purpose<br>Working<br>Registers | R0  |   | \$00  |                      |
|  | R1  |   | \$01  |                      |
|  | R2  |   | \$02  |                      |
|  | ... |   |       |                      |
|  | R13 |   | \$0D  |                      |
|  | R14 |   | \$0E  |                      |
|  | R15 |   | \$0F  |                      |
|  | R16 |   | \$10  |                      |
|  | R17 |   | \$11  |                      |
|  | ... |   |       |                      |
|  | R26 |   | \$1A  | X-register Low Byte  |
|  | R27 |   | \$1B  | X-register High Byte |
|  | R28 |   | \$1C  | Y-register Low Byte  |
|  | R29 |   | \$1D  | Y-register High Byte |
|  | R30 |   | \$1E  | Z-register Low Byte  |
|  | R31 |   | \$1F  | Z-register High Byte |

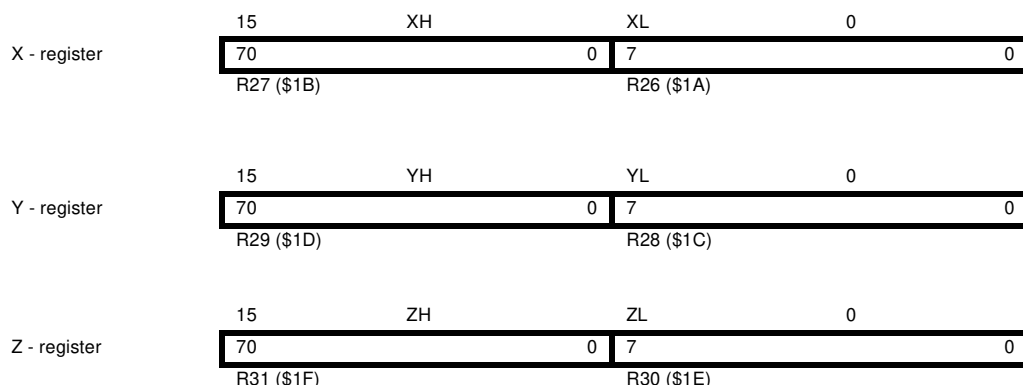
Most register operating instructions in the instruction set have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 7, each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-registers can be set to index any register in the file.

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

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

**Figure 8.** 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).

## **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, logical, and bit-functions. ATmega323 also provides a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

## **The In-System Reprogrammable Flash Program Memory**

The ATmega323 contains 32K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 16K x 16. The Flash Program memory space is divided in two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 1,000 write/erase cycles. The ATmega323 Program Counter (PC) is 14 bits wide, thus addressing the 16K Program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail on page 177. See page 197 for a detailed description on Flash data serial downloading using the SPI pins. See page 202 for details on serial downloading using the JTAG Interface.

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

See also page 13 for the different Program memory addressing modes.

## **The SRAM Data Memory**

Figure 9 shows how the ATmega323 SRAM Memory is organized.

The lower 2,144 Data memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File + I/O Memory, and the next 2,048 locations address the internal data SRAM.

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

The direct addressing reaches the entire data space.

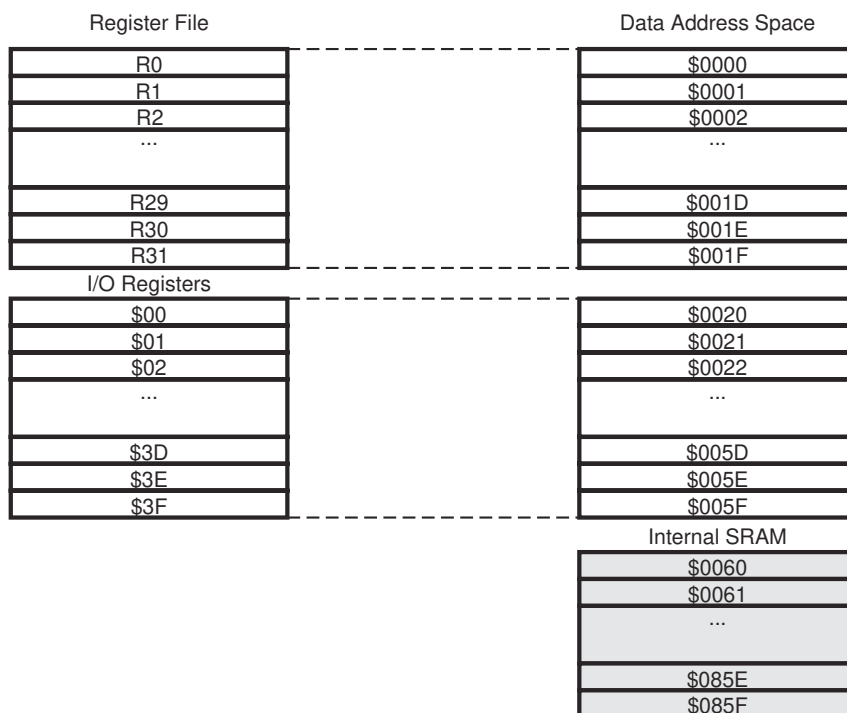
The Indirect with Displacement mode features a 63 address locations reach from the base address given by the Y- or Z-register.

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

The 32 general purpose working registers, 64 I/O Registers, and the 2,048 bytes of internal data SRAM in the ATmega323 are all accessible through all these addressing modes.



**Figure 9. SRAM Organization**

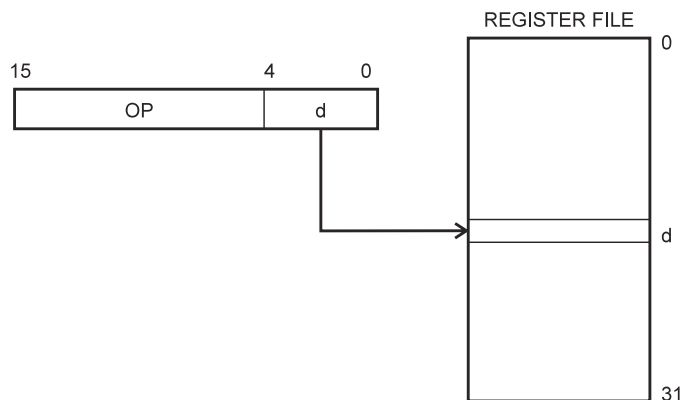


## The Program and Data Addressing Modes

The ATmega323 AVR Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the Program memory (Flash) and Data memory (SRAM, Register File, and I/O 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.

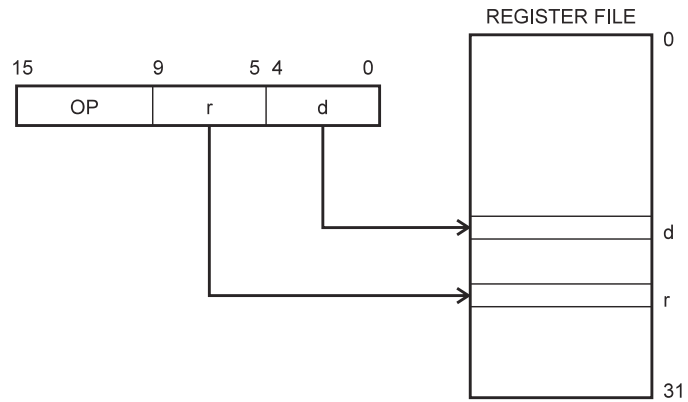
### Register Direct, Single Register Rd

**Figure 10. Direct Single Register Addressing**



The operand is contained in register d (Rd).

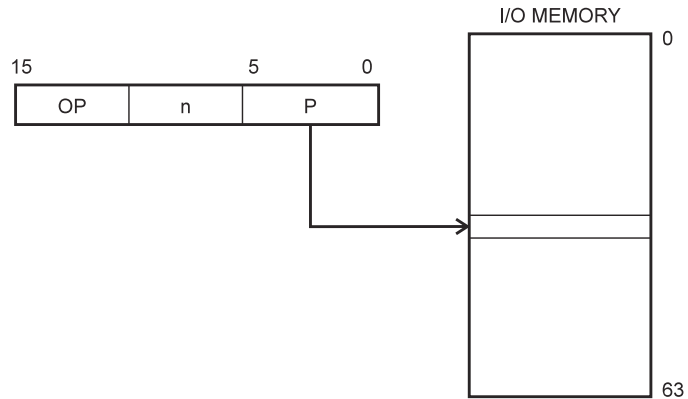
## Register Direct, Two Registers **Figure 11.** Direct Register Addressing, Two Registers Rd and Rr



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

## I/O Direct

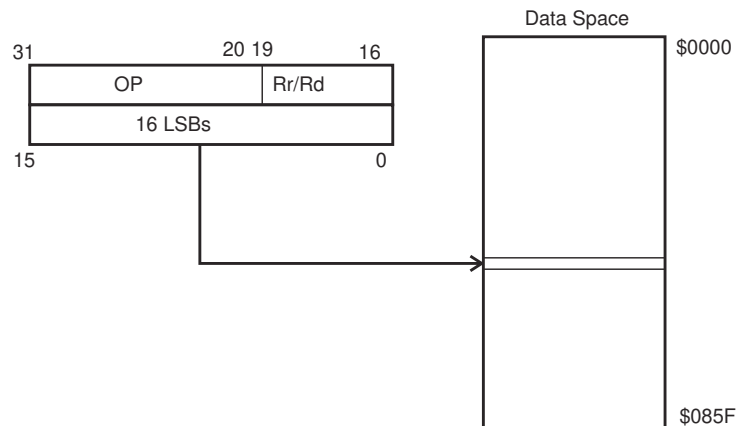
**Figure 12.** I/O Direct Addressing



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

## Data Direct

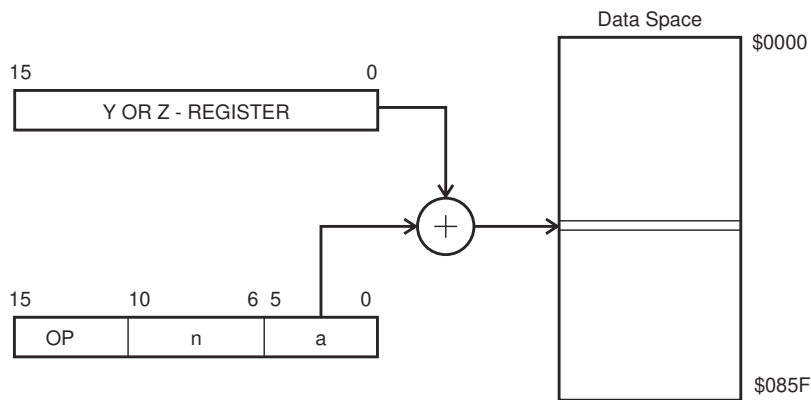
**Figure 13.** Direct Data Addressing



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

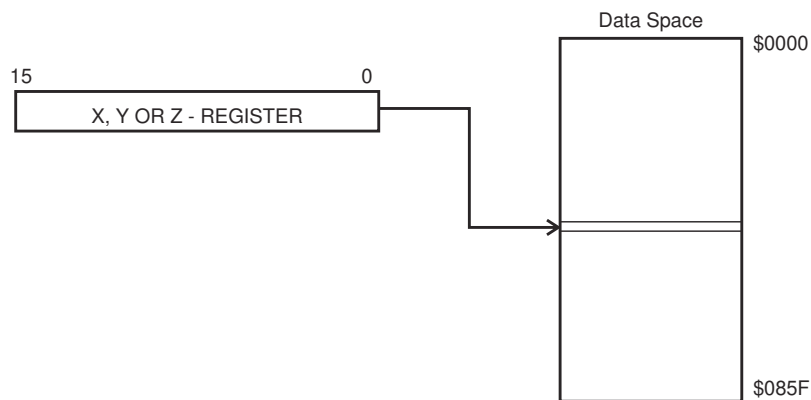
**Figure 14.** Data Indirect with Displacement



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

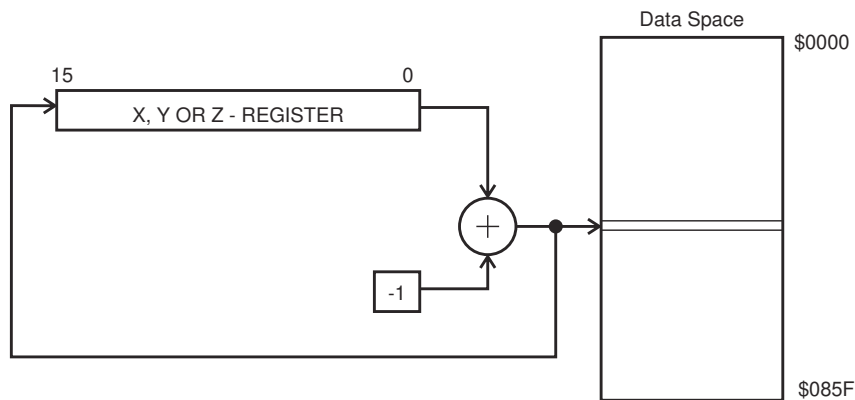
**Figure 15.** Data Indirect Addressing



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

## Data Indirect with Pre-decrement

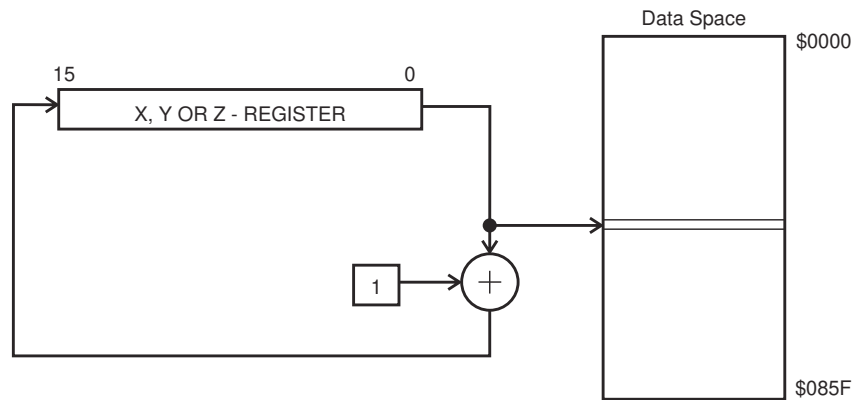
**Figure 16.** Data Indirect Addressing With Pre-decrement



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

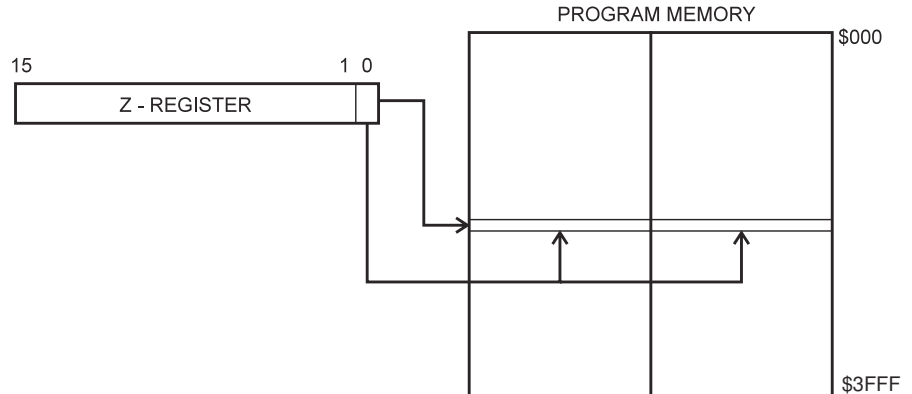
**Figure 17.** Data Indirect Addressing With Post-increment



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 and SPM Instructions

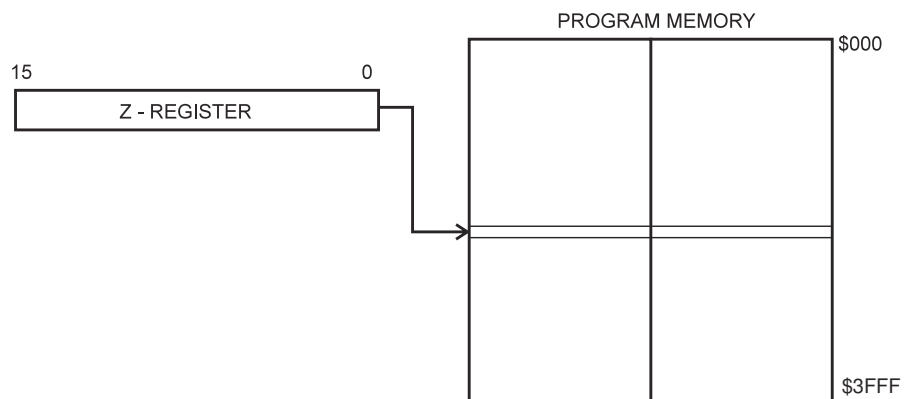
**Figure 18.** Code Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 16K). For LPM, the LSB selects Low Byte if cleared (LSB = 0) or High Byte if set (LSB = 1). For SPM, the LSB should be cleared.

## Indirect Program Addressing, IJMP and ICALL

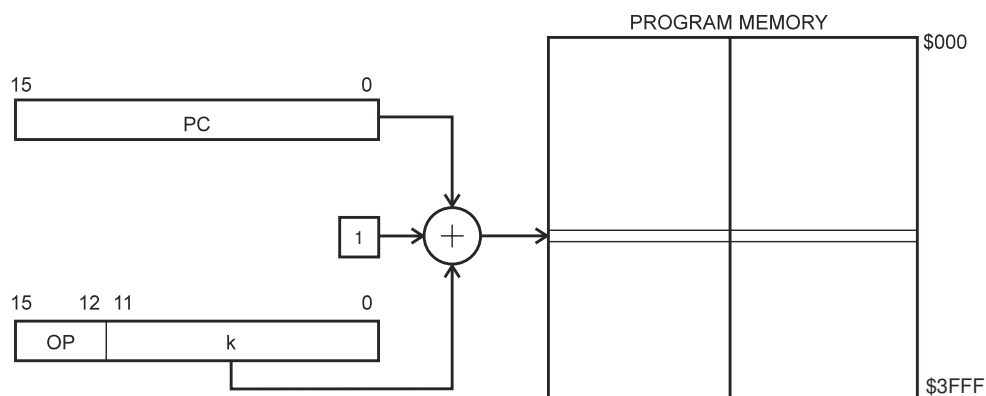
**Figure 19.** Indirect Program Memory Addressing



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

## Relative Program Addressing, Rjmp and Rcall

**Figure 20.** Relative Program Memory Addressing



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

## The EEPROM Data Memory

The ATmega323 contains 1K bytes of data EEPROM Memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on page 66 specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For SPI data downloading of the EEPROM, see page 197 for a detailed description.

## Memory Access Times 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  $\phi$ , directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 21 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 21.** The Parallel Instruction Fetches and Instruction Executions

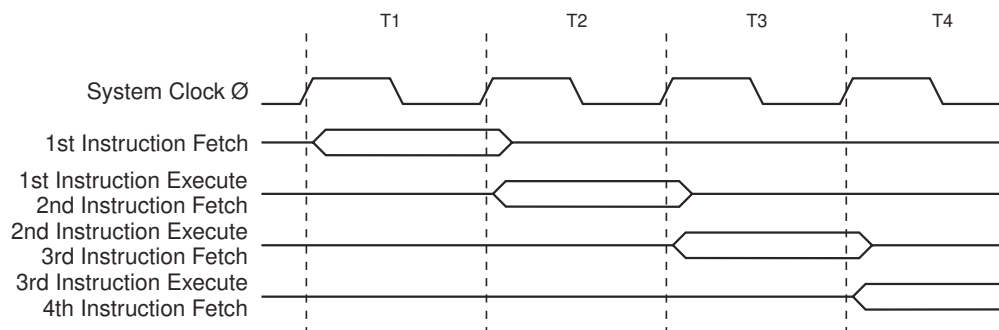
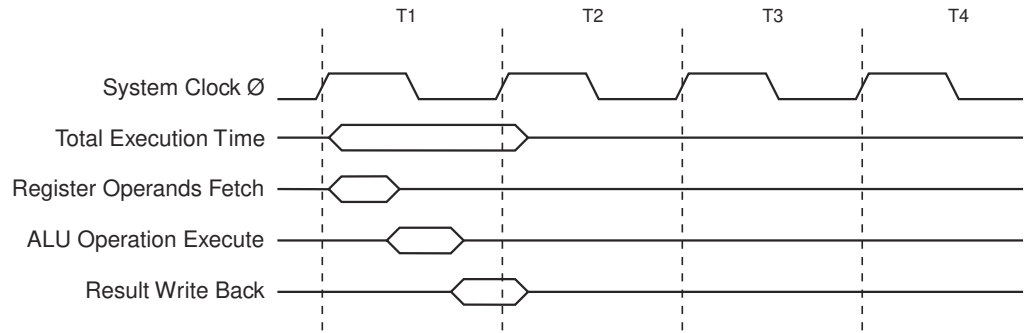


Figure 22 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

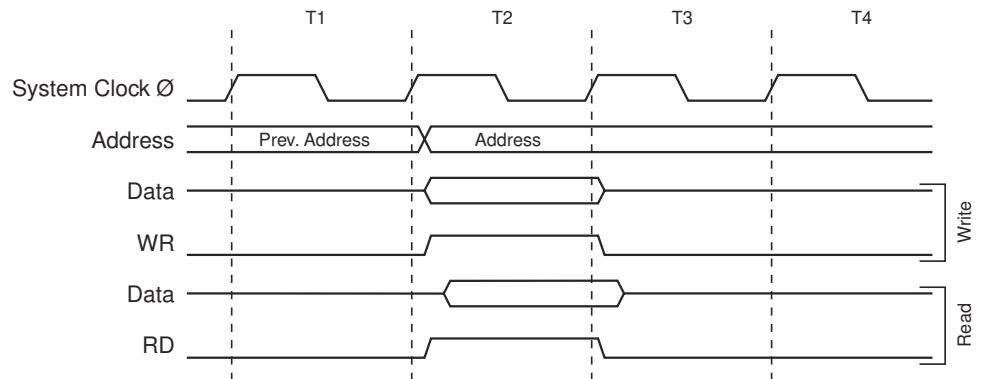


**Figure 22.** Single Cycle ALU Operation



The internal data SRAM access is performed in two System Clock cycles as described in Figure 23.

**Figure 23.** On-chip Data SRAM Access Cycles



## I/O Memory

The I/O space definition of the ATmega323 is shown in Table 2.

**Table 2.** ATmega323 I/O Space

| I/O Address (SRAM Address) | Name   | Function                                   |
|----------------------------|--------|--|
| \$3F (\$5F)                | SREG   | Status Register                            |
| \$3E (\$5E)                | SPH    | Stack Pointer High                         |
| \$3D (\$5D)                | SPL    | Stack Pointer Low                          |
| \$3C (\$3C)                | OCR0   | Timer/Counter0 Output Compare Register     |
| \$3B (\$5B)                | GICR   | General Interrupt Control Register         |
| \$3A (\$5A)                | GIFR   | General Interrupt Flag Register            |
| \$39 (\$59)                | TIMSK  | Timer/Counter Interrupt Mask Register      |
| \$38 (\$58)                | TIFR   | Timer/Counter Interrupt Flag Register      |
| \$37 (\$57)                | SPMCR  | SPM Control Register                       |
| \$36 (\$56)                | TWCR   | Two-wire Serial Interface Control Register |
| \$35 (\$55)                | MCUCR  | MCU general Control Register               |
| \$34 (\$54)                | MCUCSR | MCU general Control and Status Register    |

**Table 2.** ATmega323 I/O Space (Continued)

| I/O Address (SRAM Address) | Name   | Function   |
|----------------------------|--------|--|
| \$33 (\$53)                | TCCR0  | Timer/Counter0 Control Register                    |
| \$32 (\$52)                | TCNT0  | Timer/Counter0 (8-bit)                             |
| \$31 (\$51) <sup>(1)</sup> | OSCCAL | Oscillator Calibration Register                    |
|                            | OCDR   | On-chip Debug Register                             |
| \$30 (\$50)                | SFIOR  | Special Function I/O Register                      |
| \$2F (\$4F)                | TCCR1A | Timer/Counter1 Control Register A                  |
| \$2E (\$4E)                | TCCR1B | Timer/Counter1 Control Register B                  |
| \$2D (\$4D)                | TCNT1H | Timer/Counter1 High Byte                           |
| \$2C (\$4C)                | TCNT1L | Timer/Counter1 Low Byte                            |
| \$2B (\$4B)                | OCR1AH | Timer/Counter1 Output Compare Register A High Byte |
| \$2A (\$4A)                | OCR1AL | Timer/Counter1 Output Compare Register A Low Byte  |
| \$29 (\$49)                | OCR1BH | Timer/Counter1 Output Compare Register B High Byte |
| \$28 (\$48)                | OCR1BL | Timer/Counter1 Output Compare Register B Low Byte  |
| \$27 (\$47)                | ICR1H  | T/C 1 Input Capture Register High Byte             |
| \$26 (\$46)                | ICR1L  | T/C 1 Input Capture Register Low Byte              |
| \$25 (\$45)                | TCCR2  | Timer/Counter2 Control Register                    |
| \$24 (\$44)                | TCNT2  | Timer/Counter2 (8-bit)                             |
| \$23 (\$43)                | OCR2   | Timer/Counter2 Output Compare Register             |
| \$22 (\$42)                | ASSR   | Asynchronous Mode Status Register                  |
| \$21 (\$41)                | WDTCSR | Watchdog Timer Control Register                    |
| \$20 (\$40) <sup>(2)</sup> | UBRRH  | USART Baud Rate Register High Byte                 |
|                            | UCSRC  | USART Control and Status Register C                |
| \$1F (\$3F)                | EEARH  | EEPROM Address Register High Byte                  |
| \$1E (\$3E)                | EEARL  | EEPROM Address Register Low Byte                   |
| \$1D (\$3D)                | EEDR   | EEPROM Data Register                               |
| \$1C (\$3C)                | EECR   | EEPROM Control Register                            |
| \$1B (\$3B)                | PORTA  | Data Register, Port A                              |
| \$1A (\$3A)                | DDRA   | Data Direction Register, Port A                    |
| \$19 (\$39)                | PINA   | Input Pins, Port A                                 |
| \$18 (\$38)                | PORTB  | Data Register, Port B                              |
| \$17 (\$37)                | DDRB   | Data Direction Register, Port B                    |
| \$16 (\$36)                | PINB   | Input Pins, Port B                                 |
| \$15 (\$35)                | PORTC  | Data Register, Port C                              |
| \$14 (\$34)                | DDRC   | Data Direction Register, Port C                    |
| \$13 (\$33)                | PINC   | Input Pins, Port C                                 |

**Table 2.** ATmega323 I/O Space (Continued)

| I/O Address (SRAM Address) | Name   | Function   |
|----------------------------|--------|--|
| \$12 (\$32)                | PORTD  | Data Register, Port D                              |
| \$11 (\$31)                | DDRD   | Data Direction Register, Port D                    |
| \$10 (\$30)                | PIND   | Input Pins, Port D                                 |
| \$0F (\$2F)                | SPDR   | SPI I/O Data Register                              |
| \$0E (\$2E)                | SPSR   | SPI Status Register                                |
| \$0D (\$2D)                | SPCR   | SPI Control Register                               |
| \$0C (\$2C)                | UDR    | USART I/O Data Register                            |
| \$0B (\$2B)                | UCSRA  | USART Control and Status Register A                |
| \$0A (\$2A)                | UCSRB  | USART Control and Status Register B                |
| \$09 (\$29)                | UBRRRL | USART Baud Rate Register Low Byte                  |
| \$08 (\$28)                | ACSR   | Analog Comparator Control and Status Register      |
| \$07 (\$27)                | ADMUX  | ADC Multiplexer Select Register                    |
| \$06 (\$26)                | ADCSR  | ADC Control and Status Register                    |
| \$05 (\$25)                | ADCH   | ADC Data Register High                             |
| \$04 (\$24)                | ADCL   | ADC Data Register Low                              |
| \$03 (\$23)                | TWDR   | Two-wire Serial Interface Data Register            |
| \$02 (\$22)                | TWAR   | Two-wire Serial Interface (Slave) Address Register |
| \$01 (\$21)                | TWSR   | Two-wire Serial Interface Status Register          |
| \$00 (\$20)                | TWBR   | Two-wire Serial Interface Bit Rate Register        |

- Notes:
1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCD Register.
  2. Refer to the USART description for details on how to access UBRRH and UCSRC.

All ATmega323 I/Os and peripherals are placed in the I/O space. The 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 and OUT, the I/O addresses \$00 - \$3F must be used. When addressing I/O Registers as SRAM, \$20 must be added to these addresses. 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 I/O and Peripherals Control Registers are explained in the following sections.

## The 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, $S = N \oplus V$

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

### • Bit 3 – V: Two’s Complement Overflow Flag

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

### • Bit 2 – N: Negative Flag

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

### • Bit 1 – Z: Zero Flag

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

### • Bit 0 – C: Carry Flag

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

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.

## The Stack Pointer – SP

The ATmega323 Stack Pointer is implemented as two 8-bit registers in the I/O space locations \$3E (\$5E) and \$3D (\$5D). As the ATmega323 Data memory has \$860 locations, 12 bits are used.

| Bit           | 15  | 14  | 13  | 12  | 11   | 10   | 9   | 8   |     |
|---------------|-----|-----|-----|-----|------|------|-----|-----|-----|
| \$3E (\$5E)   | –   | –   | –   | –   | SP11 | SP10 | SP9 | SP8 | SPH |
| \$3D (\$5D)   | SP7 | SP6 | SP5 | SP4 | SP3  | SP2  | SP1 | SP0 | SPL |
|               | 7   | 6   | 5   | 4   | 3    | 2    | 1   | 0   |     |
| Read/Write    | R   | R   | R   | R   | R/W  | R/W  | R/W | R/W |     |
|               | R/W | R/W | R/W | R/W | R/W  | R/W  | R/W | R/W |     |
| Initial Value | 0   | 0   | 0   | 0   | 0    | 0    | 0   | 0   |     |
|               | 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 the return address is pushed onto the Stack with subroutine call and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

## Reset and Interrupt Handling

The ATmega323 provides nineteen different interrupt sources. These interrupts and the separate Reset Vector, each have a separate Program Vector in the Program memory space. All interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be disabled when Boot Lock bits BLB02 or BLB12 are set. See the section “Boot Loader Support” on page 177 for details

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. The Interrupt Vectors can be moved to the start of the boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). See the GICR description on page 33 for details..

**Table 3.** Reset and Interrupt Vectors

| Vector No. | Program Address <sup>(2)</sup> | Source       | Interrupt Definition   |
|------------|--------------------------------|--------------|--|
| 1          | \$000 <sup>(1)</sup>           | RESET        | External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset |
| 2          | \$002                          | INT0         | External Interrupt Request 0                                     |
| 3          | \$004                          | INT1         | External Interrupt Request 1                                     |
| 4          | \$006                          | INT2         | External Interrupt Request 2                                     |
| 5          | \$008                          | TIMER2 COMP  | Timer/Counter2 Compare Match                                     |
| 6          | \$00A                          | TIMER2 OVF   | Timer/Counter2 Overflow  |
| 7          | \$00C                          | TIMER1 CAPT  | Timer/Counter1 Capture Event                                     |
| 8          | \$00E                          | TIMER1 COMPA | Timer/Counter1 Compare Match A                                   |
| 9          | \$010                          | TIMER1 COMPB | Timer/Counter1 Compare Match B                                   |



**Table 3.** Reset and Interrupt Vectors (Continued)

| Vector No. | Program Address <sup>(2)</sup> | Source      | Interrupt Definition         |
|------------|--------------------------------|-------------|------------------------------|
| 10         | \$012                          | TIMER1 OVF  | Timer/Counter1 Overflow      |
| 11         | \$014                          | TIMER0 COMP | Timer/Counter0 Compare Match |
| 12         | \$016                          | TIMER0 OVF  | Timer/Counter0 Overflow      |
| 13         | \$018                          | SPI, STC    | Serial Transfer Complete     |
| 14         | \$01A                          | USART, RXC  | USART, Rx Complete           |
| 15         | \$01C                          | USART, UDRE | USART Data Register Empty    |
| 16         | \$01E                          | USART, TXC  | USART, Tx Complete           |
| 17         | \$020                          | ADC         | ADC Conversion Complete      |
| 18         | \$022                          | EE_RDY      | EEPROM Ready                 |
| 19         | \$024                          | ANA_COMP    | Analog Comparator            |
| 20         | \$026                          | TWSI        | Two-wire Serial Interface    |

- Notes:
1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see “Boot Loader Support” on page 177.
  2. When the IVSEL bit in GICR is set, Interrupt Vectors will be moved to the start of the boot Flash section. The address of each Interrupt Vector will then be address in this table plus the start address of the boot Flash section.

Table 4 shows Reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings.

**Table 4.** Reset and Interrupt Vectors Placement

| BOOTRST | IVSEL | Reset address      | Interrupt Vectors Start Address |
|---------|-------|--------------------|---------------------------------|
| 0       | 0     | \$0000             | \$0002                          |
| 0       | 1     | \$0000             | Boot Reset Address + \$0002     |
| 1       | 0     | Boot Reset Address | \$0002                          |
| 1       | 1     | Boot Reset Address | Boot Reset Address + \$0002     |

Note: The Boot Reset Address is shown in Table 59 on page 177.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega323 is:

| Address | Labels | Code           | Comments                  |
|---------|--------|----------------|---------------------------|
| \$000   |        | jmp RESET      | ; Reset Handler           |
| \$002   |        | jmp EXT_INT0   | ; IRQ0 Handler            |
| \$004   |        | jmp EXT_INT1   | ; IRQ1 Handler            |
| \$006   |        | jmp EXT_INT2   | ; IRQ2 Handler            |
| \$008   |        | jmp TIM2_COMP  | ; Timer2 Compare Handler  |
| \$00a   |        | jmp TIM2_OVF   | ; Timer2 Overflow Handler |
| \$00c   |        | jmp TIM1_CAPT  | ; Timer1 Capture Handler  |
| \$00e   |        | jmp TIM1_COMPA | ; Timer1 CompareA Handler |
| \$010   |        | jmp TIM1_COMPB | ; Timer1 CompareB Handler |
| \$012   |        | jmp TIM1_OVF   | ; Timer1 Overflow Handler |
| \$014   |        | jmp TIM0_COMP  | ; Timer0 Compare Handler  |
| \$016   |        | jmp TIM0_OVF   | ; Timer0 Overflow Handler |

```

$018          jmp    SPI_STC;      ; SPI Transfer Complete Handler
$01a          jmp    USART_RXC    ; USART RX Complete Handler
$01c          jmp    USART_UDRE   ; UDR Empty Handler
$01e          jmp    USART_TXC    ; USART TX Complete Handler
$020          jmp    ADC          ; ADC Conversion Complete Interrupt Handler
$022          jmp    EE_RDY       ; EEPROM Ready Handler
$024          jmp    ANA_COMP     ; Analog Comparator Handler
$026          jmp    TWSI         ; Two-wire Serial Interface Interrupt
Handler
;
$028          MAIN: ldi    r16,high(RAMEND); Main program start
$029          out     SPH,r16     ; Set Stack Pointer to top of RAM
$02a          ldi     r16,low(RAMEND)
$02b          out     SPL,r16
$02c          <instr> xxx
...           ...

```

When the BOOTRST Fuse is unprogrammed, the boot section size set to 4K bytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses are:

| Address | Labels | Code                  | Comments                                      |
|---------|--------|-----------------------|---|
| \$000   |        | jmp RESET             | ; Reset handler                               |
|         |        |                       | ;   |
| \$002   | MAIN:  | ldi r16,high(RAMEND); | Main program start                            |
| \$003   |        | out SPH,r16           | ; Set Stack Pointer to top of RAM             |
| \$004   |        | ldi r16,low(RAMEND)   |   |
| \$005   |        | out SPL,r16           |   |
| \$006   |        | <instr> xxx           |   |
|         |        |                       | ;   |
|         |        | .org \$3802           |   |
| \$3802  |        | jmp EXT_INT0          | ; IRQ0 Handler                                |
| \$3804  |        | jmp EXT_INT1          | ; IRQ1 Handler                                |
| ...     |        | ... ..                | ;   |
| \$3826  |        | jmp TWSI              | ; Two-wire Serial Interface Interrupt Handler |

When the BOOTRST Fuse is programmed and the boot section size set to 4K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses are:

| Address | Labels | Code                  | Comments                                      |
|---------|--------|-----------------------|---|
|         |        | .org \$002            |   |
| \$002   |        | jmp EXT_INT0          | ; IRQ0 Handler                                |
| \$004   |        | jmp EXT_INT1          | ; IRQ1 Handler                                |
| ...     |        | ... ..                | ;   |
| \$026   |        | jmp TWSI              | ; Two-wire Serial Interface Interrupt Handler |
|         |        |                       | ;   |
| \$028   | MAIN:  | ldi r16,high(RAMEND); | Main program start                            |
| \$029   |        | out SPH,r16           | ; Set Stack Pointer to top of RAM             |

```

$02a      ldi    r16,low(RAMEND)
$02b      out    SPL,r16
$02c      <instr> xxx
;
.org $3800
$3800      jmp    RESET      ; Reset handler
...        ...      ...      ...

```

When the BOOTRST Fuse is programmed, the boot section size set to 4K bytes and the IVSEL bit in the GICR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses are:

| Address     | Labels | Code                  | Comments                                      |
|-------------|--------|-----------------------|---|
| \$000       | MAIN:  | ldi r16,high(RAMEND); | Main program start                            |
| \$001       |        | out SPH,r16           | ; Set Stack Pointer to top of RAM             |
| \$002       |        | ldi r16,low(RAMEND)   |   |
| \$003       |        | out SPL,r16           |   |
| \$004       |        | <instr> xxx           |   |
| ;           |        |                       |   |
| .org \$3800 |        |                       |   |
| \$3800      |        | jmp RESET             | ; Reset handler                               |
| \$3802      |        | jmp EXT_INT0          | ; IRQ0 Handler                                |
| \$3804      |        | jmp EXT_INT1          | ; IRQ1 Handler                                |
| ...         | ...    | ...                   | ;   |
| \$3826      |        | jmp TWSI              | ; Two-wire Serial Interface Interrupt Handler |

## Reset Sources

The ATmega323 has five 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  $\overline{RESET}$  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 ( $V_{BOT}$ ).
- JTAG AVR Reset. The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system.

During Reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – Absolute 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. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 24 shows the Reset Logic. Table 5 and Table 6 define the timing and electrical parameters of the reset circuitry.