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 Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Features

- High-performance, Low-power AVR® 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
- Program and Data Memories
  - 16K Bytes of Non-volatile In-System Programmable Flash Endurance: 1,000 Write/Erase Cycles
  - Optional Boot Code Memory with Independent Lock bits Self-programming of Program and Data Memories
  - 512 Bytes of Non-volatile In-System Programmable EEPROM Endurance: 100,000 Write/Erase Cycles
  - 1K Byte of Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and PWM
  - Expanded 16-bit Timer/Counter System with Separate Prescaler, Compare, Capture Modes and Dual 8-, 9-, or 10-bit PWM
  - Dual Programmable Serial UARTs
  - Master/Slave SPI Serial Interface
  - Real-time Counter with Separate Oscillator
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset
  - External and Internal Interrupt Sources
  - Three Sleep Modes: Idle, Power-save and Power-down
- Power Comsumption at 4 MHz, 3.0V, 25°C
  - Active 3.0 mA
  - Idle Mode 1.2 mA
  - Power-down Mode < 1 µA
- I/O and Packages
  - 35 Programmable I/O Lines
  - 40-lead PDIP and 44-lead TQFP
- Operating Voltages
  - 2.7V - 5.5V for the ATmega161L
  - 4.0V - 5.5V for the ATmega161
- Speed Grades
  - 0 - 4 MHz for the ATmega161L
  - 0 - 8 MHz for the ATmega161
- Commercial and Industrial Temperature Ranges

Disclaimer

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

8-bit AVR® Microcontroller with 16K Bytes of In-System Programmable Flash

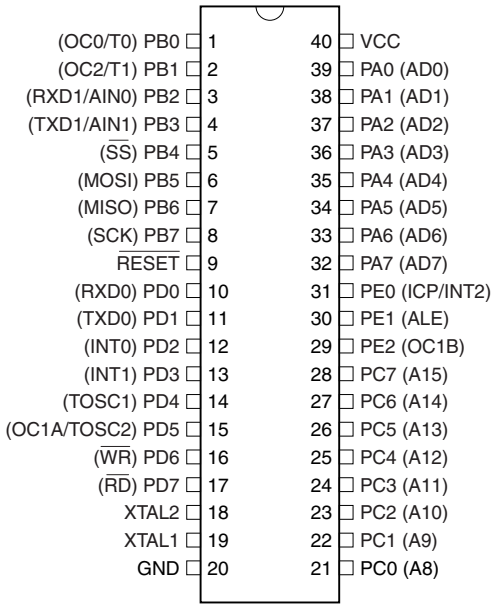
ATmega161
ATmega161L

Note: Not recommended in new designs.

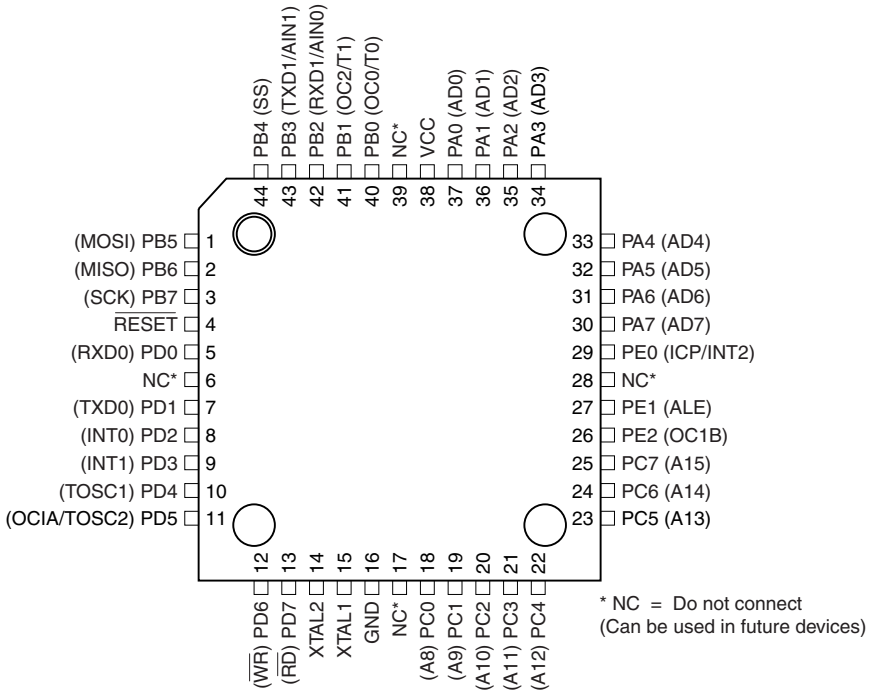
Rev. 1228D–AVR–02/07

Pin Configuration

PDIP



TQFP



Description

The ATmega161 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega161 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed. The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code-efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

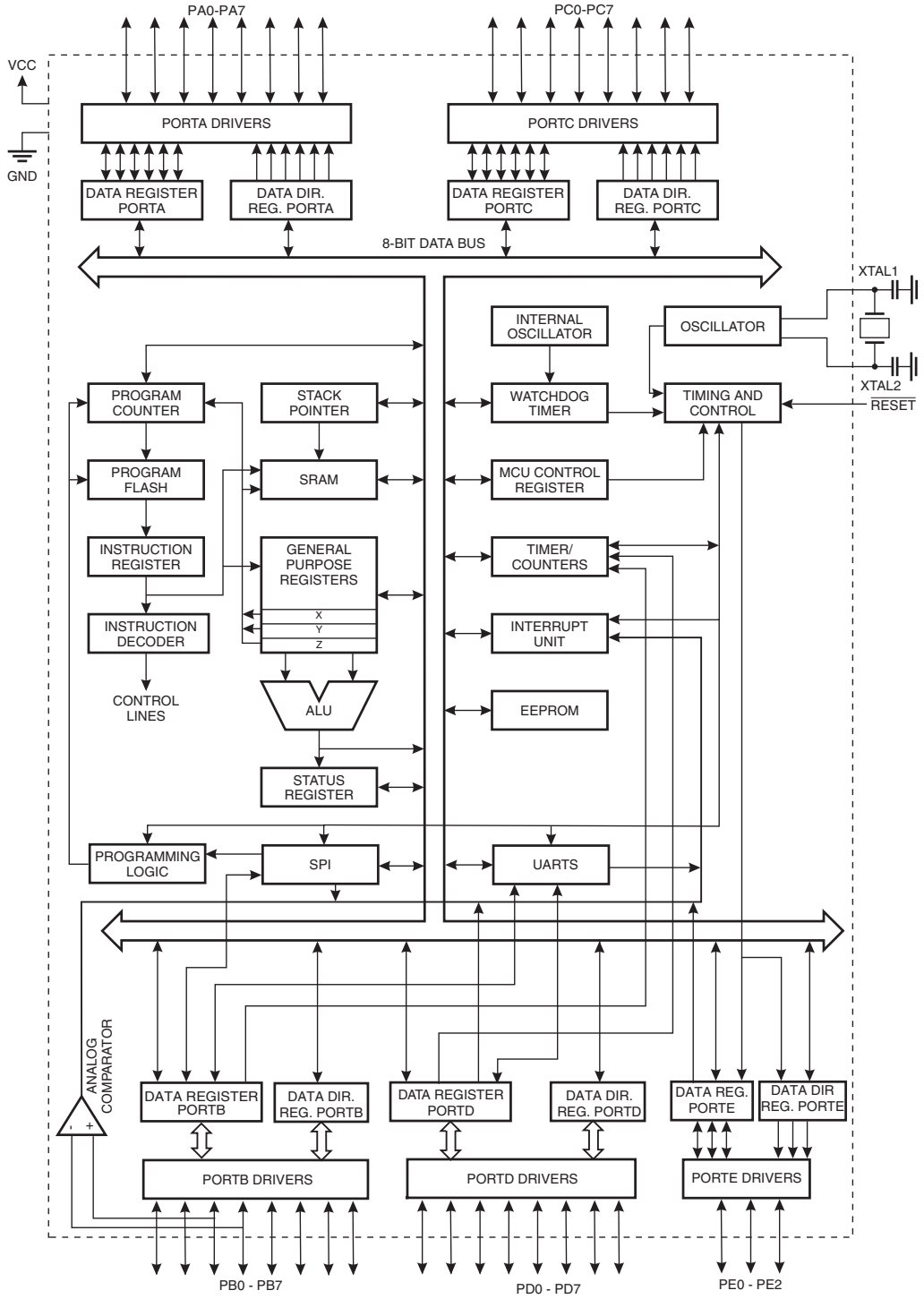
The ATmega161 provides the following features: 16K bytes of In-System or Self-programmable Flash, 512 bytes EEPROM, 1K byte of SRAM, 35 general purpose I/O lines, 32 general purpose working registers, Real-time Counter, three flexible Timer/Counters with Compare modes, internal and external interrupts, two programmable serial UARTs, programmable Watchdog Timer with internal Oscillator, an SPI serial port and three 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 and SRAM contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the timer Oscillator continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping.

The device is manufactured using Atmel's high-density non-volatile memory technology. The On-chip Flash Program memory can be reprogrammed using the Self-programming capability through the Boot Block and an ISP through the SPI port, or by using a conventional non-volatile Memory programmer. By combining an enhanced RISC 8-bit CPU with In-System Programmable Flash on a monolithic chip, the Atmel ATmega161 is a powerful microcontroller that provides a highly flexible and cost-effective solution to many embedded control applications.

The ATmega161 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.

Block Diagram

Figure 1. The ATmega161 Block Diagram



Pin Descriptions

VCC Supply voltage.

GND Ground.

Port A (PA7..PA0) Port A is an 8-bit bi-directional I/O port. 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 A serves as a Multiplexed Address/Data port when using external memory interface.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors. The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. 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 ATmega161 as listed on page 92.

Port C (PC7..PC0) Port C is an 8-bit bi-directional I/O port with internal pull-up resistors. 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 as an address high output when using external memory interface.

Port D (PD7..PD0) Port D is an 8-bit bi-directional I/O port with internal pull-up resistors. 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 ATmega161 as listed on page 101.

Port E (PE2..PE0) Port E is a 3-bit bi-directional I/O port with internal pull-up resistors. The Port E output buffers can sink 20 mA. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega161 as listed on page 107.

$\overline{\text{RESET}}$ 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.

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier that 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. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 3.

Figure 2. Oscillator Connections

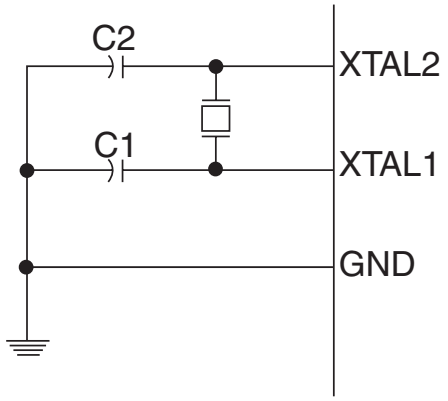
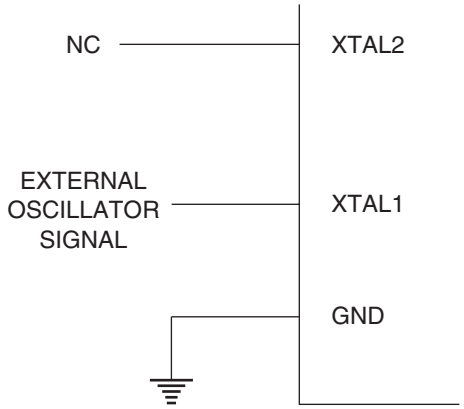


Figure 3. External Clock Drive Configuration



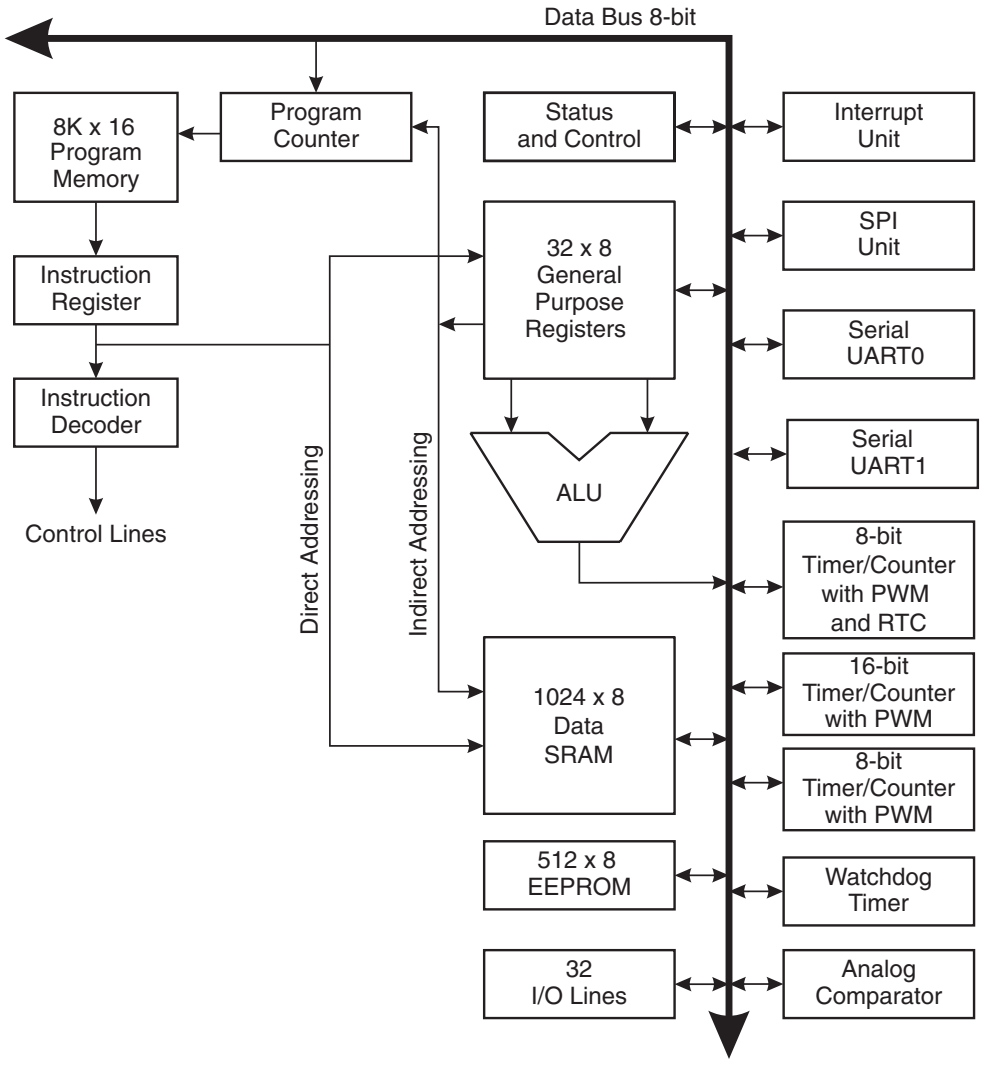
Architectural Overview

The fast-access Register File concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one Arithmetic Logic Unit (ALU) operation is executed. Two operands are output from the Register File, the operation is executed and the result is stored back in the Register File – in one clock cycle.

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

The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the ATmega161 AVR RISC microcontroller architecture.

Figure 4. The ATmega161 AVR RISC Architecture



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

The I/O memory space contains 64 addresses for CPU peripheral functions such as Control Registers, Timer/Counters, and other I/O functions. The I/O memory can be accessed directly or as the Data Space locations following those of the Register File, $20 - $5F.

The AVR uses a Harvard architecture concept – with separate memories and buses for program and data. The Program memory is executed with a two-stage pipeline. While one instruction is being executed, the next instruction is pre-fetched from the Program memory. This concept enables instructions to be executed in every clock cycle. The Program memory is Self-programmable Flash memory.

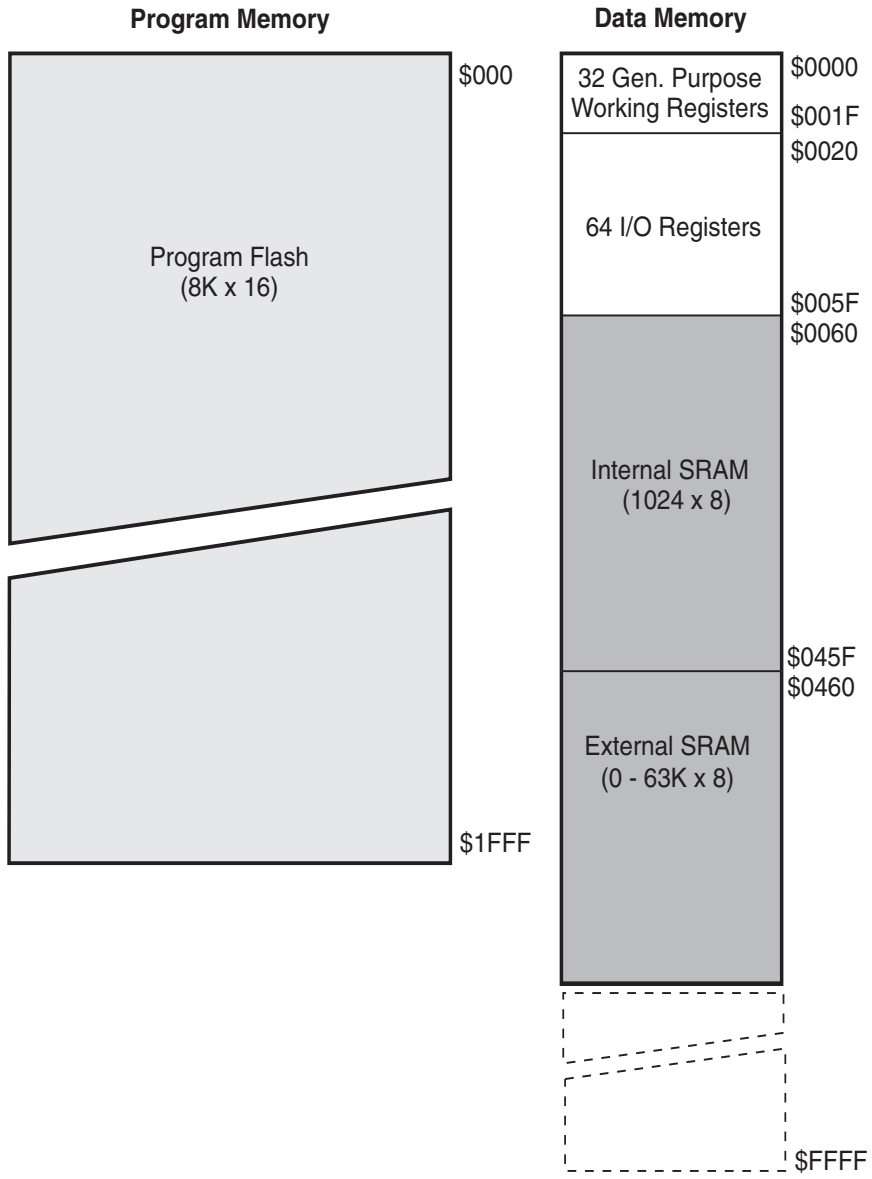
With the jump and call instructions, the whole 8K word address space is directly accessed. Most AVR instructions have a single 16-bit word format. Every Program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM and, consequently, the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP (Stack Pointer) in the reset routine (before subroutines or interrupts are executed). The 16-bit Stack Pointer is read/write accessible in the I/O space.

The 1K byte 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.

Figure 5. Memory Maps

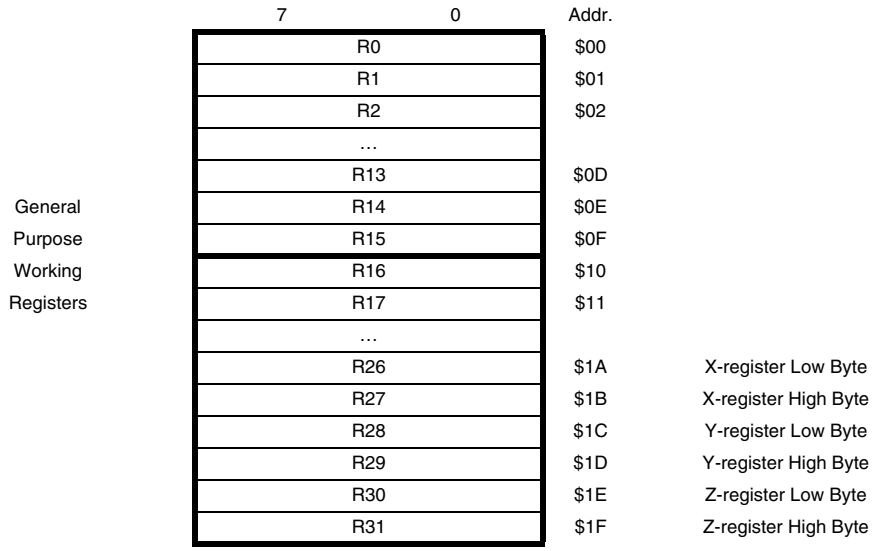


A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All the different interrupts have a separate Interrupt Vector in the Interrupt Vector table at the beginning of the Program memory. The different interrupts have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

The General Purpose Register File

Figure 6 shows the structure of the 32 general purpose working registers in the CPU.

Figure 6. AVR CPU General Purpose Working Registers



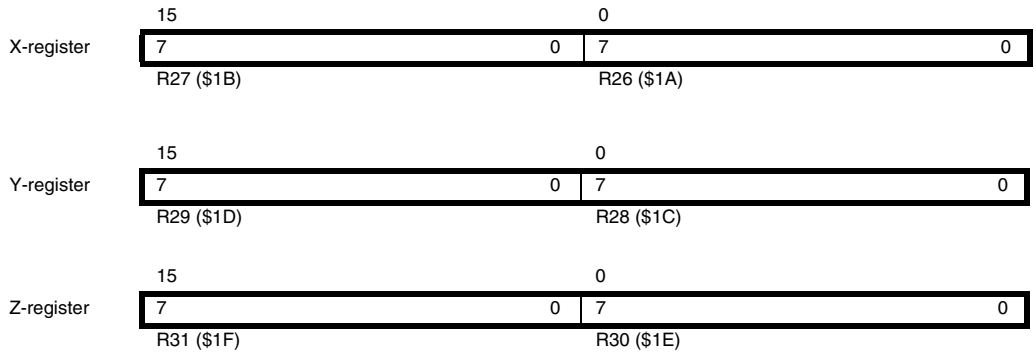
All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exceptions are the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI, and ORI between a constant and a register, and the LDI instruction for load immediate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

As shown in Figure 6, 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 7. X-, Y-, and Z-registers



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

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the Register File are executed. The ALU operations are divided into three main categories – arithmetic, logical and bit functions. ATmega161 also provides a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the Instruction Set section for a detailed description.

Self-programmable Flash Program Memory

The ATmega161 contains 16K bytes of On-chip Self-programmable and In-System Programmable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 8K x 16. The Flash memory has an endurance of at least 1,000 write/erase cycles. The ATmega161 Program Counter (PC) is 13 bits wide, thus addressing the 8,192 Program memory locations.

See page 110 for a detailed description of Flash data downloading.

See page 13 for the different Program Memory Addressing modes.

EEPROM Data Memory

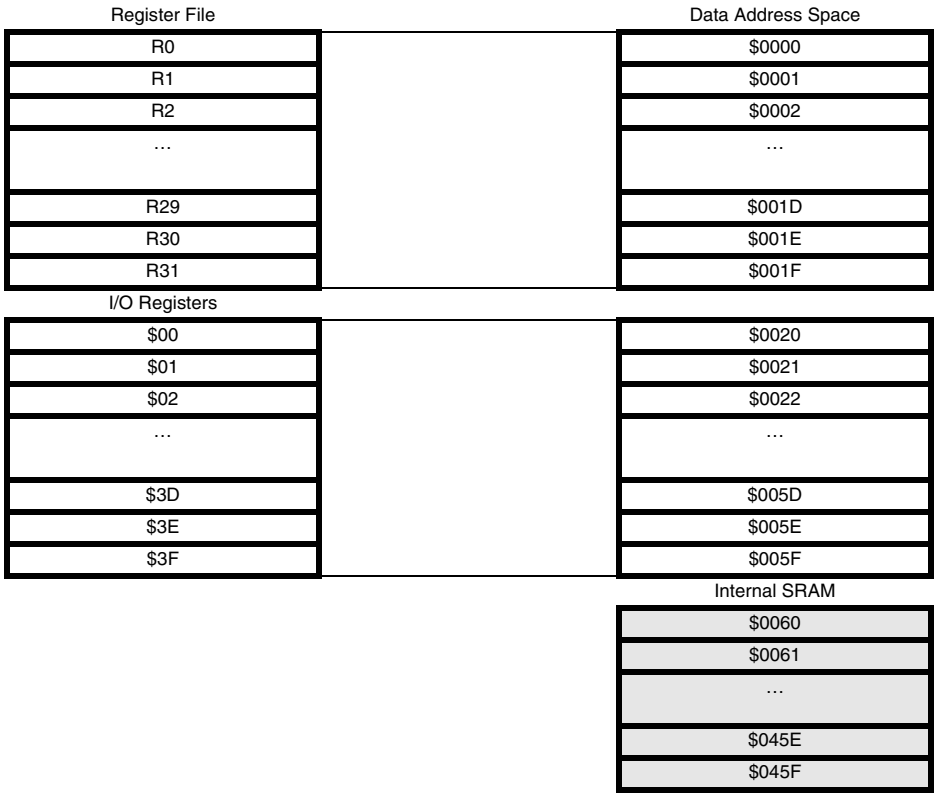
The ATmega161 contains 512 bytes of data EEPROM memory. It is organized as a separate data space in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles per location. The interface between the EEPROM and the CPU is described on page 60, specifying the EEPROM Address Registers, the EEPROM Data Register and the EEPROM Control Register.

For the SPI data downloading, see page 125 for a detailed description.

SRAM Data Memory

Figure 8 shows how the ATmega161 SRAM memory is organized.

Figure 8. SRAM Organization



The lower 1120 Data memory locations address the Register File, the I/O memory and the internal data SRAM. The first 96 locations address the Register File and I/O memory and the next 1K locations address the internal data SRAM. An optional external Data memory device can be placed in the same SRAM memory space. This memory device will occupy the locations following the internal SRAM and up to as much as 64K - 1, depending on external memory size.

When the addresses accessing the Data memory space exceed the internal data SRAM locations, the memory device is accessed using the same instructions as for the internal data SRAM access. When the internal data space is accessed, the read and write strobe pins ($\overline{RD}$ and $\overline{WR}$) are inactive during the whole access cycle. External memory operation is enabled by setting the SRE bit in the MCUCR Register. See “Interface to External Memory” on page 84 for details.

Accessing external memory takes one additional clock cycle per byte compared to access of the internal SRAM. This means that the commands LD, ST, LDS, STS, PUSH, and POP take one additional clock cycle. If the Stack is placed in external memory, interrupts, subroutine calls and returns take two clock cycles extra because the 2-byte Program Counter is pushed and popped. When external memory interface is used with wait state, two additional clock cycles are used per byte. This has the following effect: Data transfer instructions take two extra clock cycles, whereas interrupt, subroutine calls and returns will need four clock cycles more than specified in the Instruction Set manual.

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 1K byte of internal data SRAM in the ATmega161 are all accessible through all these Addressing modes.

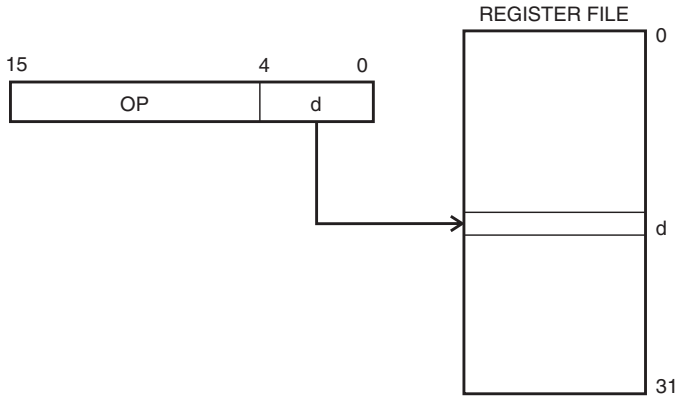
See the next section for a detailed description of the different Addressing modes.

Program and Data Addressing Modes

The ATmega161 AVR 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

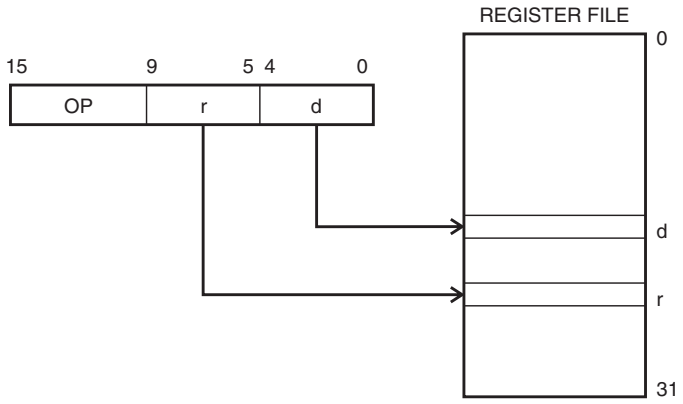
Figure 9. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

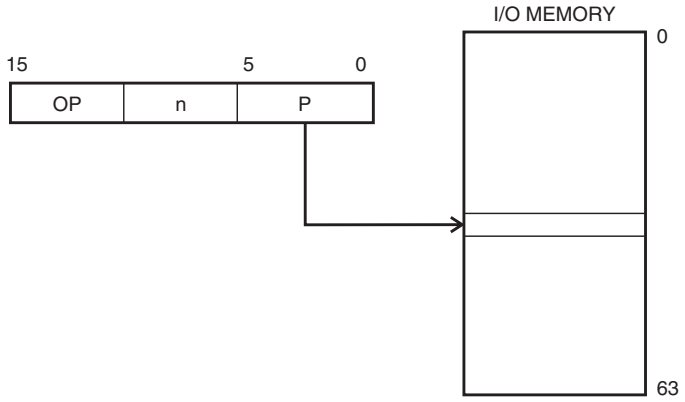
Figure 10. Direct Register Addressing, Two Registers



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

I/O Direct

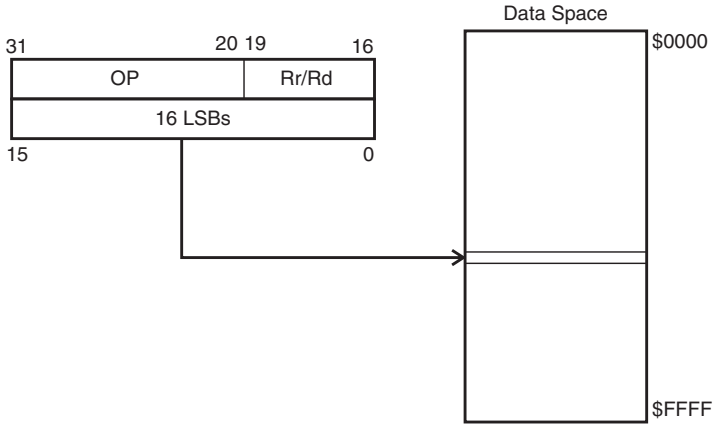
Figure 11. I/O Direct Addressing



Operand address is contained in six bits of the instruction word. n is the destination or Source Register Address.

Data Direct

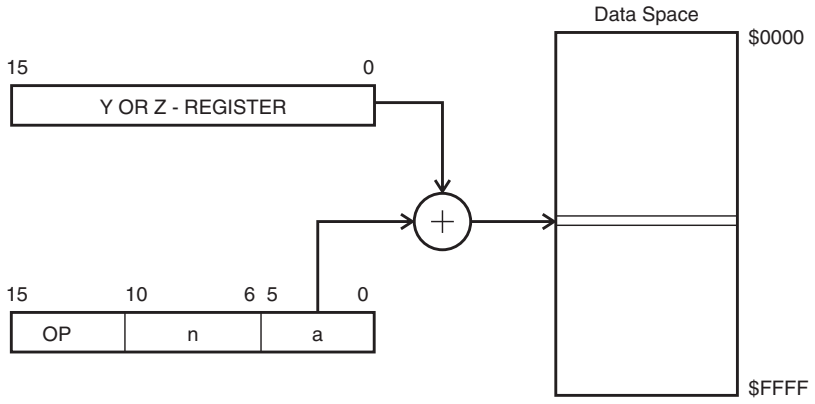
Figure 12. Direct Data Addressing



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

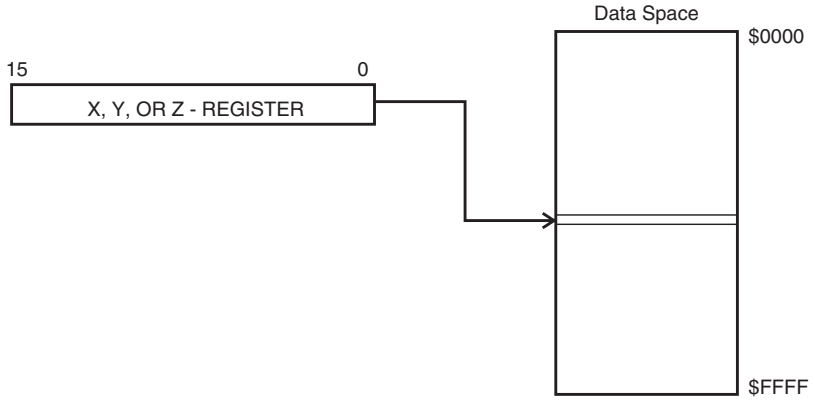
Figure 13. Data Indirect with Displacement



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

Data Indirect

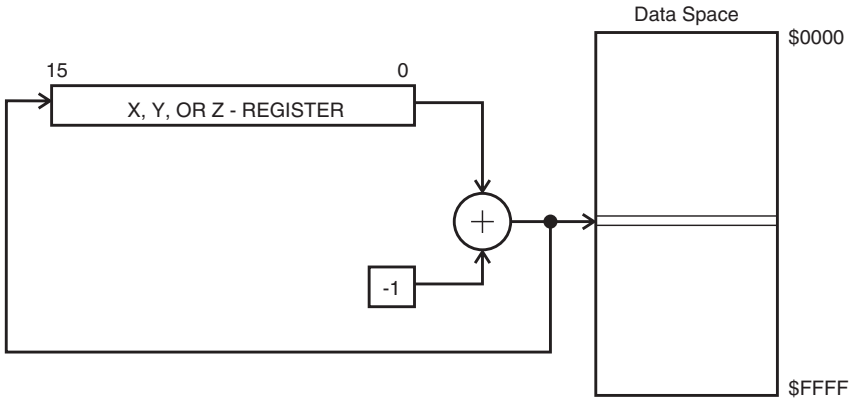
Figure 14. Data Indirect Addressing



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

Data Indirect with Pre-decrement

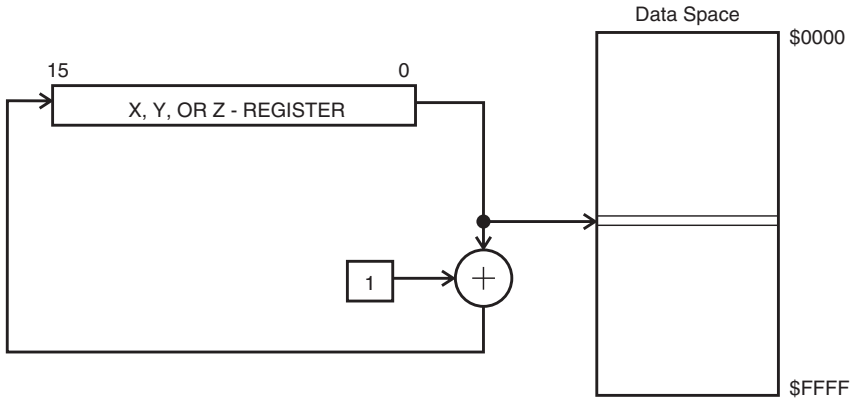
Figure 15. Data Indirect Addressing with Pre-decrement



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

Data Indirect with Post-increment

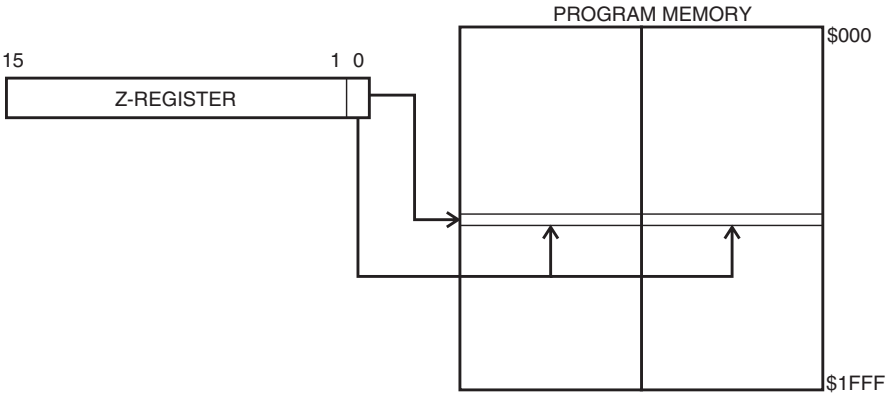
Figure 16. Data Indirect Addressing with Post-increment



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

Constant Addressing Using the LPM Instruction

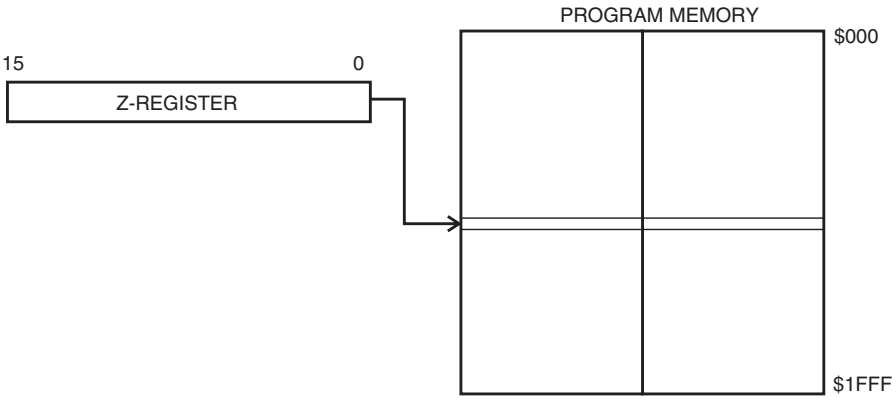
Figure 17. Code Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 8K), the LSB selects Low byte if cleared (LSB = 0) or High byte if set (LSB = 1).

Indirect Program Addressing, IJMP and ICALL

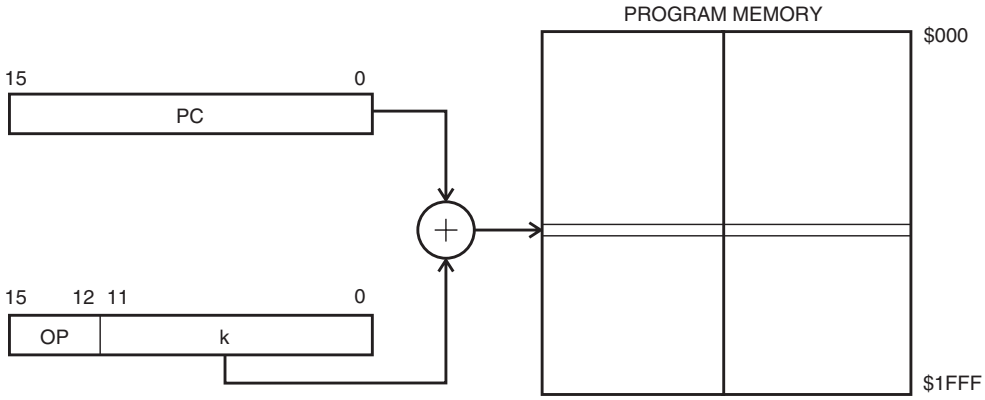
Figure 18. Indirect Program Memory Addressing



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

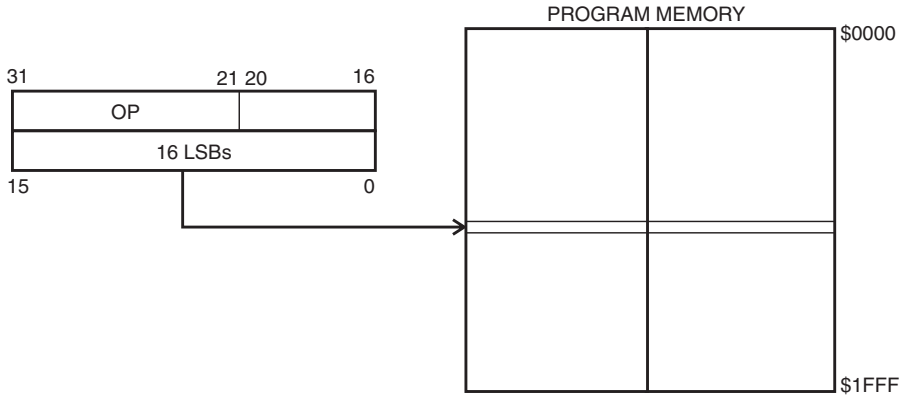
Figure 19. Relative Program Memory Addressing



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

Direct Program Addressing, JMP and CALL

Figure 20. Direct Program Addressing



Program execution continues at the address immediate in the instruction words.

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 Ø, directly generated from the external clock crystal 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

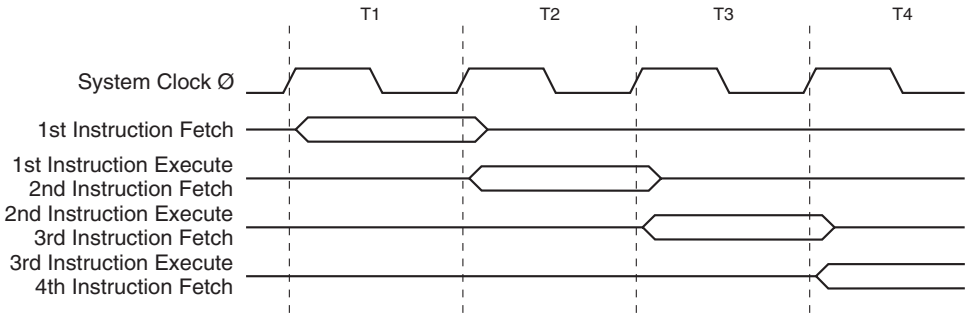
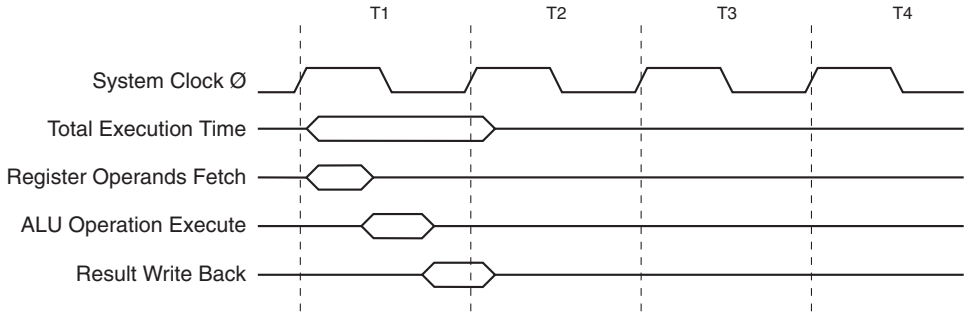


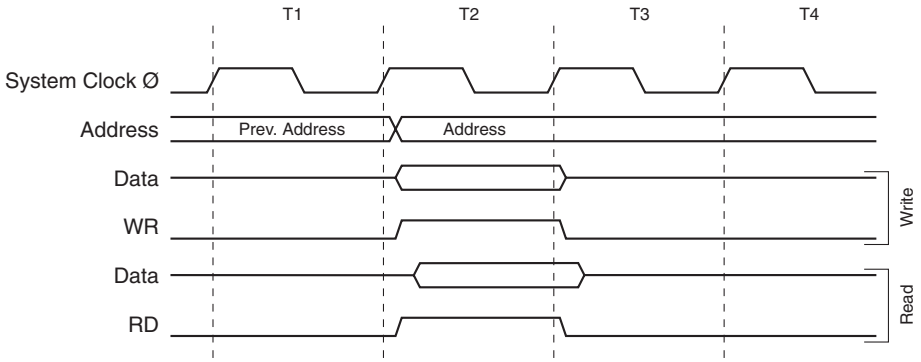
Figure 22 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed and the result is stored back to the destination register.

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 ATmega161 is shown in Table 1.

Table 1. ATmega161 I/O Space[1]

I/O Address (SRAM Address)	Name	Function
$3F($5F)	SREG	Status REGister
$3E ($5E)	SPH	Stack Pointer High
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK Register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK Register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag Register
$37 ($57)	SPMCR	Store Program Memory Control Register
$36 ($56)	EMCUCR	Extended MCU general Control Register
$35 ($55)	MCUCR	MCU general Control Register
$34 ($54)	MCUSR	MCU general Status Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$31 ($51)	OCR0	Timer/Counter0 Output Compare Register
$30 ($50)	SFIOR	Special Function IO 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 RegisterA High Byte
$2A ($4A)	OCR1AL	Timer/Counter1 Output Compare RegisterA Low Byte
$29 ($49)	OCR1BH	Timer/Counter1 Output Compare RegisterB High Byte
$28 ($48)	OCR1BL	Timer/Counter1 Output Compare RegisterB Low Byte
$27 ($47)	TCCR2	Timer/Counter2 Control Register
$26 ($46)	ASSR	Asynchronous mode StatuS Register
$25 ($45)	ICR1H	Timer/Counter1 Input Capture Register High Byte
$24 ($44)	ICR1L	Timer/Counter1 Input Capture Register Low Byte
$23 ($43)	TCNT2	Timer/Counter2 (8-bit)
$22 ($42)	OCR2	Timer/Counter2 Output Compare Register
$21 ($41)	WDTCR	Watchdog Timer Control Register
$20 ($40)	UBRRHI	UART Baud Register HIgh
$1F ($3F)	EEARH	EEPROM Address Register High
$1E ($3E)	EEARL	EEPROM Address Register Low
$1D ($3D)	EEDR	EEPROM Data Register

Table 1. ATmega161 I/O Space[1] (Continued)

I/O Address (SRAM Address)	Name	Function
$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
$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)	UDR0	UART0 I/O Data Register
$0B ($2B)	UCSR0A	UART0 Control and Status Register
$0A ($2A)	UCSR0B	UART0 Control and Status Register
$09 ($29)	UBRR0	UART0 Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register
$07 ($27)	PORTE	Data Register, Port E
$06 ($26)	DDRE	Data Direction Register, Port E
$05 ($25)	PINE	Input Pins, Port E
$03 ($23)	UDR1	UART1 I/O Data Register
$02 ($22)	UCSR1A	UART1 Control and Status Register
$01 ($21)	UCSR1B	UART1 Control and Status Register
$00 ($20)	UBRR1	UART1 Baud Rate Register

Note: 1. Reserved and unused locations are not shown in this table.

All ATmega161 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 section 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 this address. All I/O Register addresses throughout this document are shown with the SRAM address in parentheses.

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

Some of the Status Flags are cleared by writing a logical “1” 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.

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 bit is cleared (zero), none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred and is set by the RETI instruction to enable subsequent interrupts.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source and destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. See the Instruction Set description for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

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

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

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

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result 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.

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

Stack Pointer – SP

The ATmega161 Stack Pointer is implemented as two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the ATmega161 supports up to 64-Kbyte memory, all 16 bits are used.

Bit	15	14	13	12	11	10	9	8	
$3E ($5E)	**SP15**	**SP14**	**SP13**	**SP12**	**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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	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 1 when data is pushed onto the Stack with the PUSH instruction, and it is decremented by 2 when an address is pushed onto the Stack with subroutine calls and interrupts. The Stack Pointer is incremented by 1 when data is popped from the Stack with the POP instruction, and it is incremented by 2 when an address is popped from the Stack with return from subroutine RET or return from interrupt (RETI).

Reset and Interrupt Handling

The ATmega161 provides 20 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 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 2. The list also determines the priority levels of the different interrupts. The lower the address, the higher the priority level. RESET has the highest priority, and next is INT0 (the External Interrupt Request 0) and so on.

Table 2. Reset and Interrupt Vectors[1]

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on 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
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	UART0, RX	UART0, Rx Complete
15	$01c	UART1, RX	UART1, Rx Complete
16	$01e	UART0, UDRE	UART0 Data Register Empty
17	$020	UART1, UDRE	UART1 Data Register Empty
18	$022	UART0, TX	UART0, Tx Complete
19	$024	UART1, TX	UART1, Tx Complete
20	$026	EE_RDY	EEPROM Ready
21	$028	ANA_COMP	Analog Comparator

Note: 1. If BOOTRST fuse is programmed, the Reset Vector is located on program address $1e00, see Table 39 on page 112 for details.

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                 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     UART_RXC0     ; UART0 RX Complete Handler
$01c                jmp     UART_RXC1     ; UART1 RX Complete Handler
$01e                jmp     UART_DRE0     ; UDR0 Empty Handler
$020                jmp     UART_DRE1     ; UDR1 Empty Handler
$022                jmp     UART_TXC0     ; UART0 TX Complete Handler
$024                jmp     UART_TXC1     ; UART1 TX Complete Handler
$026                jmp     EE_RDY        ; EEPROM Ready Handler
$028                jmp     ANA_COMP      ; Analog Comparator Handler
;
$02a      MAIN:     ldi r16,high(RAMEND) ; Main program start
$02b                out SPH,r16
$02c                ldi r16,low(RAMEND)
$02d                out SPL,r16
$02e                <instr>  xxx
…         …         …        …
```

When the BOOTRST fuse is programmed, the most typical and general program setup for the Reset and Interrupt Vector addresses are:

```
Address   Labels     Code                 Comments
.org $002                                 ; Reset is located at $1e000
$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     UART_RXC0     ; UART0 RX Complete Handler
$01c                jmp     UART_RXC1     ; UART1 RX Complete Handler
$01e                jmp     UART_DRE0     ; UDR0 Empty Handler
$020                jmp     UART_DRE1     ; UDR1 Empty Handler
$022                jmp     UART_TXC0     ; UART0 TX Complete Handler
$024                jmp     UART_TXC1     ; UART1 TX Complete Handler
$026                jmp     EE_RDY        ; EEPROM Ready Handler
$028                jmp     ANA_COMP      ; Analog Comparator Handler
;
$02a      MAIN:     ldi r16,high(RAMEND); Main program start
$02b                out SPH,r16
$02c                ldi r16,low(RAMEND)
$02d                out SPL,r16
$02e                <instr>  xxx
;
.org $1e00
$1e00               jmp     RESET         ; Reset handler
…         …         …        …
```

Reset Sources

The ATmega161 has three sources of Reset:

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

During Reset, all I/O Registers are then set to their initial values and the program starts execution from address $000. The instruction placed in address $000 must be a JMP (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 24 shows the Reset Logic. Table 3 and Table 4 define the timing and electrical parameters of the reset circuitry.

Figure 24. Reset Logic