## 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 Nonvolatile 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 Nonvolatile In-System Programmable EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Bytes 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 and Programmable Brown-out Detection
  - External and Internal Interrupt Sources
  - Three Sleep Modes: Idle, Power Save and Power-down
- I/O and Packages
  - 35 Programmable I/O Lines
  - 40-pin PDIP, 44-pin PLCC and TQFP
- Operating Voltages
  - 2.7V - 5.5V (ATmega161L), 4.0V - 5.5V (ATmega161)
- Speed Grades
  - 0 - 4 MHz (ATmega161L), 0 - 8 MHz (ATmega161)
- Commercial and Industrial Temperature Ranges

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

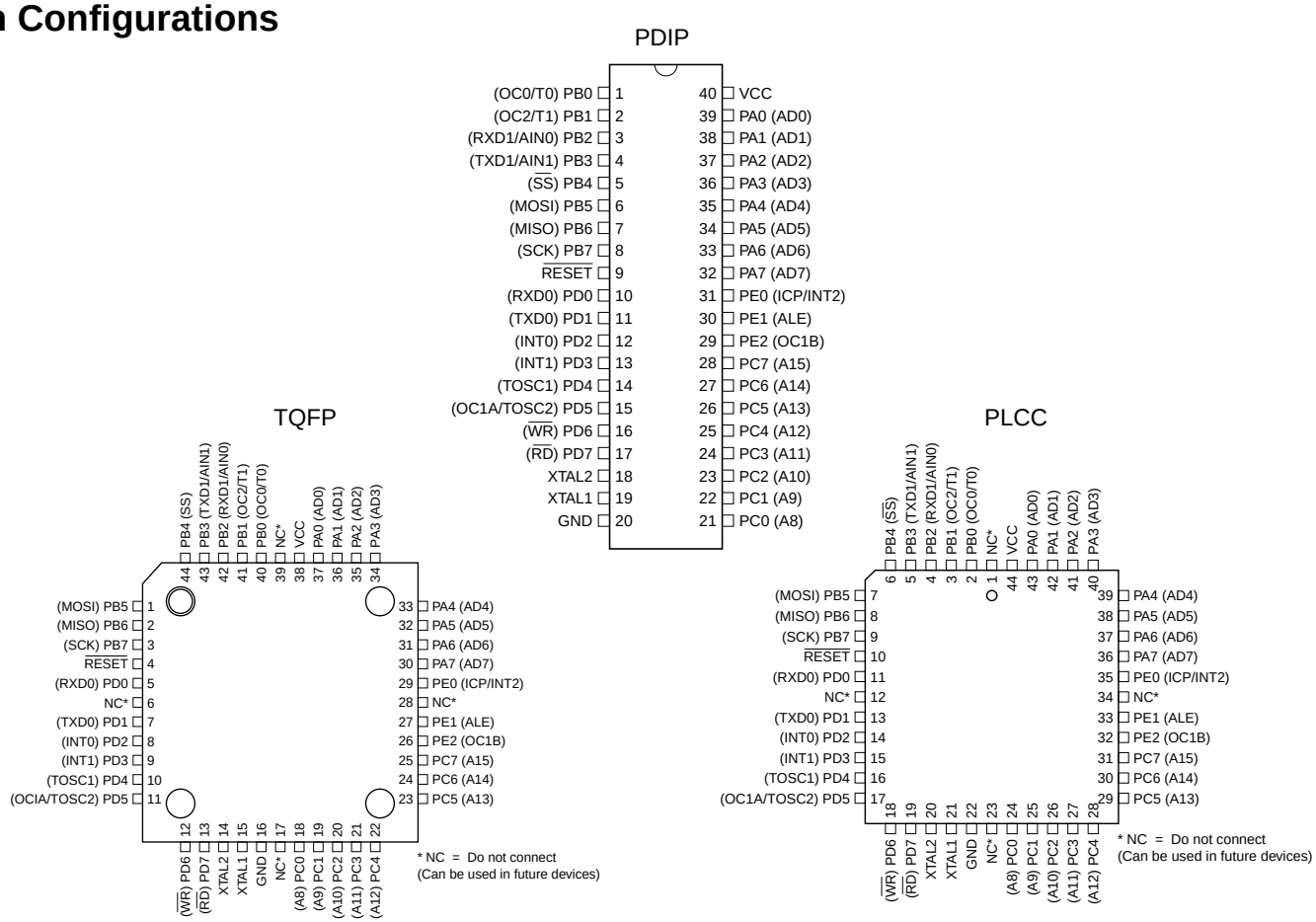
# ATmega161
# ATmega161L

# Advance Information

Rev. 1228AS–08/99

Note: This is a summary document. For the complete 134-page document, please visit our web site at *www.atmel.com* or e-mail at *literature@atmel.com* and request literature #1228A.

## Pin Configurations




## 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.

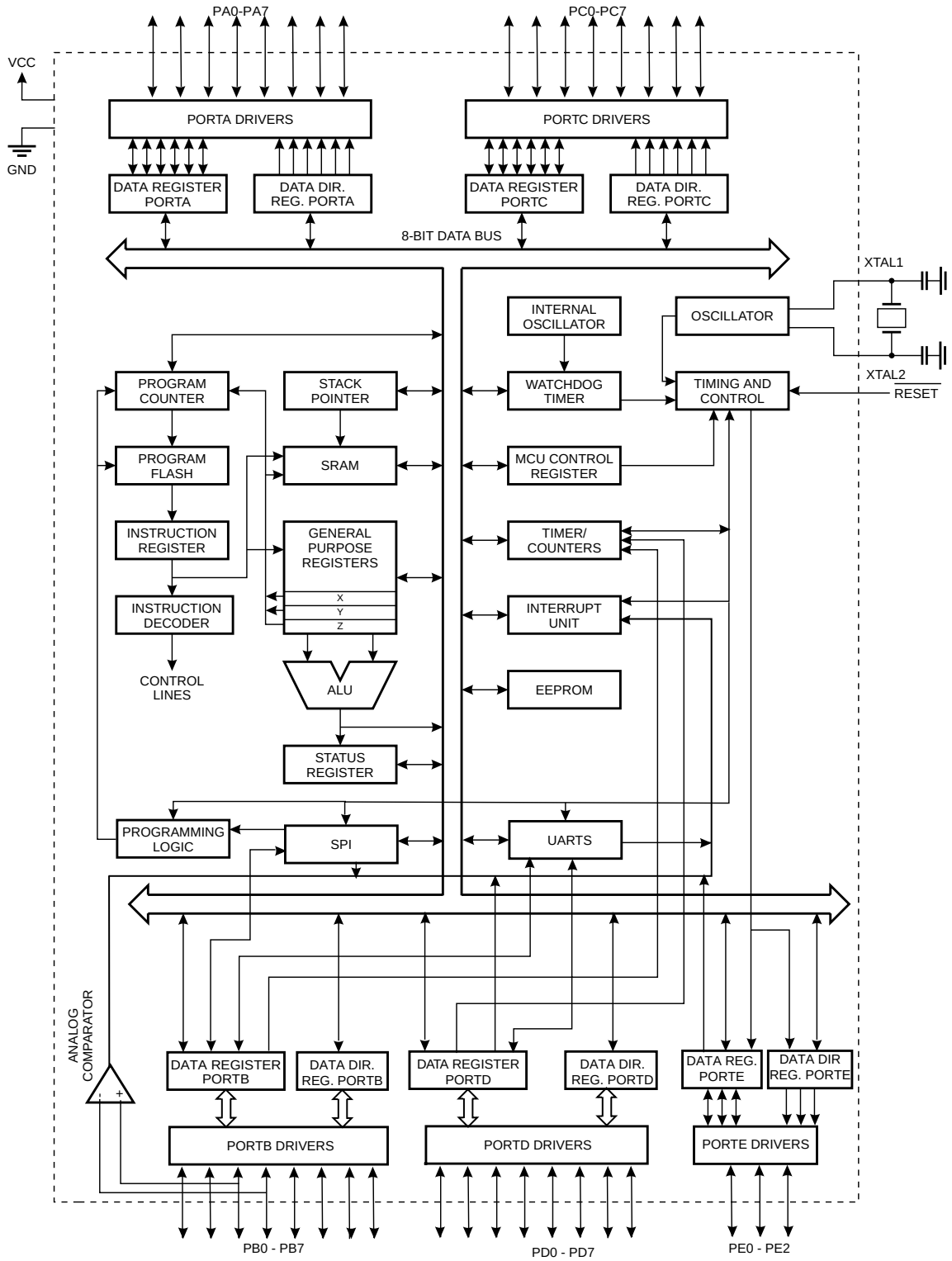
The ATmega161 provides the following features: 16K bytes of In-System- or Self-programmable Flash, 512 bytes EEPROM, 1K bytes SRAM, 35 general purpose I/O lines, 32 general purpose working registers, Real Time Counter, three flexible timer/counters with compare modes, internal and external interrupts, 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 nonvolatile memory technology. The on-chip Flash program memory can be reprogrammed using the self-programming capability through the bootblock, using an ISP through the SPI-port, or by using a conventional nonvolatile 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 bidirectional 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 Multiplexed Address/Data port when using external memory interface.

### Port B (PB7..PB0)

Port B is an 8-bit bidirectional 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 C (PC7..PC0)

Port C is an 8-bit bidirectional 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 Address high output when using external memory interface.

### Port D (PD7..PD0)

Port D is an 8-bit bidirectional 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 E (PE2..PE0)

Port E is a 3-bit bidirectional I/O port. 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.

### $\overline{\text{RESET}}$

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

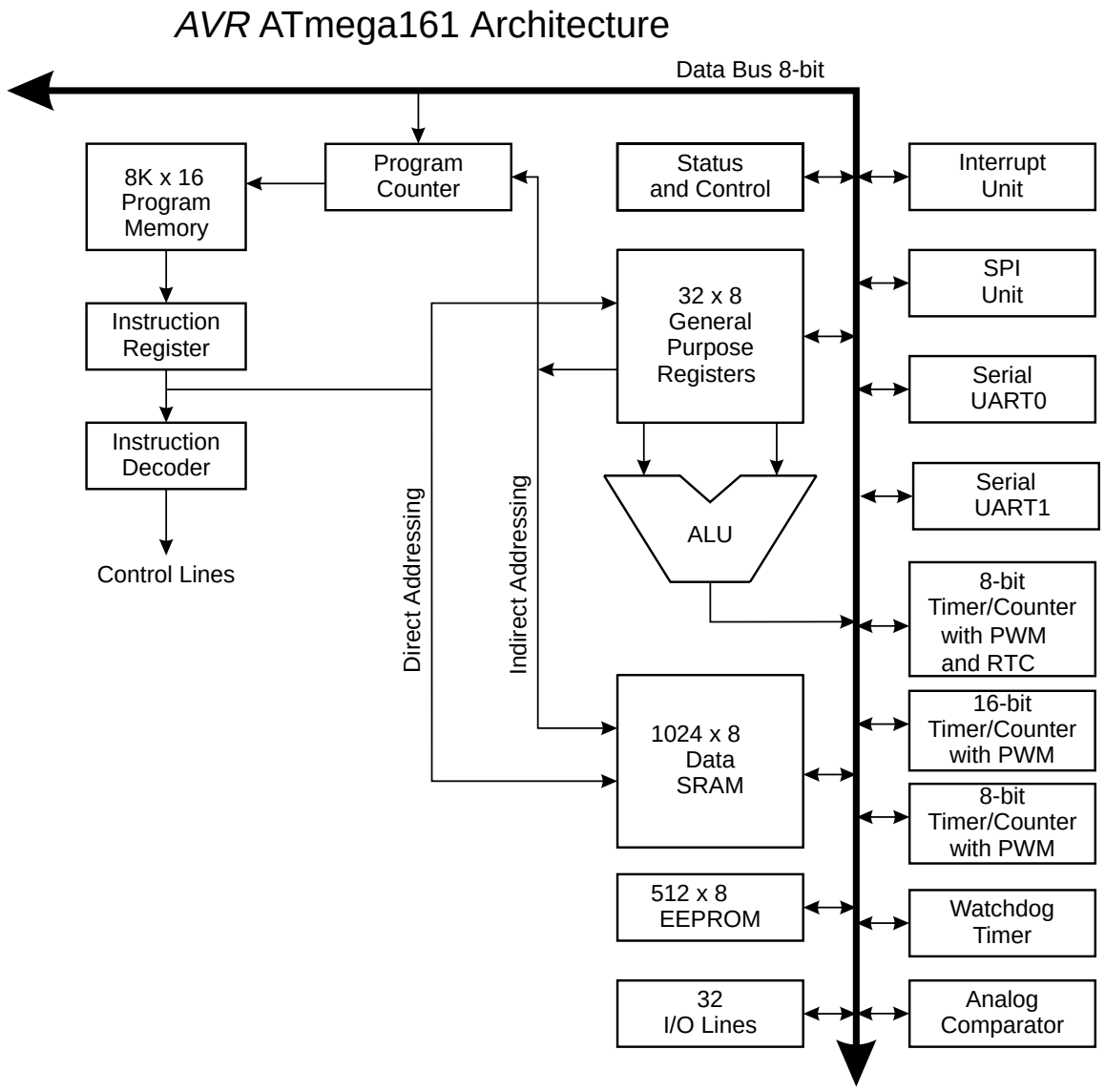
Output from the inverting oscillator amplifier

## 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 the constant table look up function. These added function registers are the 16-bits X-register, Y-register and Z-register.

**Figure 2.** The ATmega161 AVR RISC Architecture




The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 2 shows the ATmega161 AVR RISC microcontroller architecture.

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

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, 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.

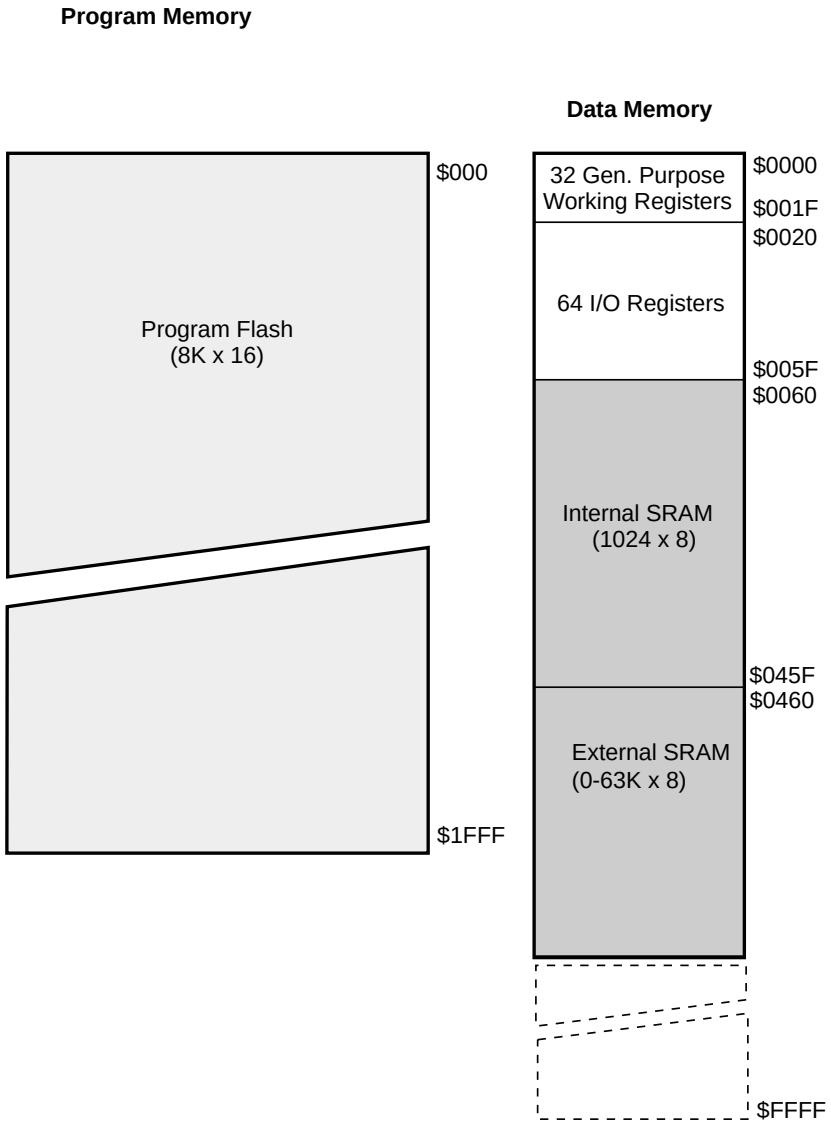
With the jump and call instructions, the whole 8K word address space is directly accessed. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

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 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.

**Figure 3.** Memory Maps




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

## Register Summary

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $3F ($5F) | SREG | I | T | H | S | V | N | Z | C |
| $3E ($5E) | SPH | SP15 | SP14 | SP13 | SP12 | SP11 | SP10 | SP9 | SP8 |
| $3D ($5D) | SPL | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |
| $3C ($5C) | Reserved es- | | | | | | | | |
| $3B ($5B) | GIMSK | INT1 | INT0 | INT2 | - | - | - | - | - |
| $3A ($5A) | GIFR | INTF1 | INTF0 | INTF2 | | | | | |
| $39 ($59) | TIMSK | TOIE1 | OCIE1A | OCIE1B | OCIE2 | TICIE1 | TOIE2 | TOIE0 | OCIE0 |
| $38 ($58) | TIFR | TOV1 | OCF1A | OCF1B | OCFI2 | ICF1 | TOV2 | TOV0 | OCIF0 |
| $37 ($57) | SPMCR | - | - | - | - | LBSET | PGWRT | PGERS | SPMEN |
| $36 ($56) | EMCUCR | SM0 | SRL2 | SRL1 | SRL0 | SRW01 | SRW00 | SRW11 | ISC2 |
| $35 ($55) | MCUCR | SRE | SRW10 | SE | SM1 | ISC11 | ISC10 | ISC01 | ISC00 |
| $34 ($54) | MCUSR | - | - | - | - | WDRF | BORF | EXTRF | PORF |
| $33 ($53) | TCCR0 | FOC0 | PWM0 | COM01 | COM00 | CTC0 | CS02 | CS01 | CS00 |
| $32 ($52) | TCNT0 | Timer/Counter0 Counter Register | | | | | | | |
| $31 ($51) | OCR0 | Timer/Counter0 Output Compare Register | | | | | | | |
| $30 ($50) | SFIOR | - | - | - | - | - | - | PSR2 | PSR10 |
| $2F ($4F) | TCCR1A | COM1A1 | COM1A0 | COM1B1 | COM1B0 | FOC1A | FOC1B | PWM11 | PWM10 |
| $2E ($4E) | TCCR1B | ICNC1 | ICES1 | - | - | CTC1 | CS12 | CS11 | CS10 |
| $2D ($4D) | TCNT1H | Timer/Counter1 - Counter Register High Byte | | | | | | | |
| $2C ($4C) | TCNT1L | Timer/Counter1 - Counter Register 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) | TCCR2 | FOC2 | PWM2 | COM21 | COM20 | CTC2 | CS22 | CS21 | CS20 |
| $26 ($46) | ASSR | - | - | - | - | AS20 | TCON2UB | OCR2UB | TCR2UB |
| $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 Counter Register | | | | | | | |
| $22 ($42) | OCR2 | Timer/Counter2 Output Compare Register | | | | | | | |
| $21 ($41) | WDTCR | - | - | - | WDTOE | WDE | WDP2 | WDP1 | WDP0 |
| $20 ($40) | UBRRHI | UBRR1[11:8] | | | | UBRR0[11:8] | | | |
| $1F ($3F) | EEARH | - | - | - | - | - | - | - | EEAR8 |
| $1E ($3E) | EEARL | EEPROM Address Register Low Byte | | | | | | | |
| $1D ($3D) | EEDR | EEPROM Data Register | | | | | | | |
| $1C ($3C) | EECR | - | - | - | - | EERIE | EEMWE | EEWE | EERE |
| $1B ($3B) | PORTA | PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |
| $1A ($3A) | DDRA | DDA7 | DDA6 | DDA5 | DDA4 | DDA3 | DDA2 | DDA1 | DDA0 |
| $19 ($39) | PINA | PINA7 | PINA6 | PINA5 | PINA4 | PINA3 | PINA2 | PINA1 | PINA0 |
| $18 ($38) | PORTB | PORTB7 | PORTB6 | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 |
| $17 ($37) | DDRB | DDB7 | DDB6 | DDB5 | DDB4 | DDB3 | DDB2 | DDB1 | DDB0 |
| $16 ($36) | PINB | PINB7 | PINB6 | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 |
| $15 ($35) | PORTC | PORTC7 | PORTC6 | PORTC5 | PORTC4 | PORTC3 | PORTC2 | PORTC1 | PORTC0 |
| $14 ($34) | DDRC | DDC7 | DDC6 | DDC5 | DDC4 | DDC3 | DDC2 | DDC1 | DDC0 |
| $13 ($33) | PINC | PINC7 | PINC6 | PINC5 | PINC4 | PINC3 | PINC2 | PINC1 | PINC0 |
| $12 ($32) | PORTD | PORTD7 | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 |
| $11 ($31) | DDRD | DDD7 | DDD6 | DDD5 | DDD4 | DDD3 | DDD2 | DDD1 | DDD0 |
| $10 ($30) | PIND | PIND7 | PIND6 | PIND5 | PIND4 | PIND3 | PIND2 | PIND1 | PIND0 |
| $0F ($2F) | SPDR | SPI Data Register | | | | | | | |
| $0E ($2E) | SPSR | SPIF | WCOL | - | - | - | - | - | SPI2X |
| $0D ($2D) | SPCR | SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| $0C ($2C) | UDR0 | UART0 I/O Data Register | | | | | | | |
| $0B ($2B) | UCSR0A | RXC0 | TXC0 | UDRE0 | FE0 | OR0 | - | U2X0 | MPCM0 |
| $0A ($2A) | UCSR0B | RXCIE0 | TXCIE0 | UDRIE0 | RXEN0 | TXEN0 | CHR90 | RXB80 | TXB80 |
| $09 ($29) | UBRR0 | UART0 Baud Rate Register | | | | | | | |
| $08 ($28) | ACSR | ACD | AINBG | ACO | ACI | ACIE | ACIC | ACIS1 | ACIS0 |
| $07 ($27) | PORTE | - | - | - | - | - | PORTE2 | PORTE1 | PORTE0 |

## Register Summary (Continued)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $06 ($26) | DDRE | - | - | - | - | - | DDE2 | DDE1 | DDE0 |
| $05 ($25) | PINE | - | - | - | - | - | PINE2 | PINE1 | PINE0 |
| $04 ($24) | Reserved | | | | | | | | |
| $03 ($23) | UDR1 | UART1 I/O Data Register | | | | | | | |
| $02 ($22) | UCSR1A | RXC1 | TXC1 | UDRE1 | FE1 | OR1 | - | U2X1 | MPCM1 |
| $01 ($21) | UCSR1B | RXCIE1 | TXCIE1 | UDRIE1 | RXEN1 | TXEN1 | CHR91 | RXB81 | TXB81 |
| $00 ($20) | UBRR1 | UART1 Baud Rate Register | | | | | | | |

Notes: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

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

## Instruction Set Summary

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| **ARITHMETIC AND LOGIC INSTRUCTIONS** | | | | | |
| ADD | Rd, Rr | Add two Registers | Rd ← Rd + Rr | Z,C,N,V,H | 1 |
| ADC | Rd, Rr | Add with Carry two Registers | Rd ← Rd + Rr + C | Z,C,N,V,H | 1 |
| ADIW | Rdl,K | Add Immediate to Word | Rdh:Rdl ← Rdh:Rdl + K | Z,C,N,V,S | 2 |
| SUB | Rd, Rr | Subtract two Registers | Rd ← Rd - Rr | Z,C,N,V,H | 1 |
| SUBI | Rd, K | Subtract Constant from Register | Rd ← Rd - K | Z,C,N,V,H | 1 |
| SBC | Rd, Rr | Subtract with Carry two Registers | Rd ← Rd - Rr - C | Z,C,N,V,H | 1 |
| SBCI | Rd, K | Subtract with Carry Constant from Reg. | Rd ← Rd - K - C | Z,C,N,V,H | 1 |
| SBIW | Rdl,K | Subtract Immediate from Word | Rdh:Rdl ← Rdh:Rdl - K | Z,C,N,V,S | 2 |
| AND | Rd, Rr | Logical AND Registers | Rd ← Rd • Rr | Z,N,V | 1 |
| ANDI | Rd, K | Logical AND Register and Constant | Rd ← Rd • K | Z,N,V | 1 |
| OR | Rd, Rr | Logical OR Registers | Rd ← Rd v Rr | Z,N,V | 1 |
| ORI | Rd, K | Logical OR Register and Constant | Rd ← Rd v K | Z,N,V | 1 |
| EOR | Rd, Rr | Exclusive OR Registers | Rd ← Rd ⊕ Rr | Z,N,V | 1 |
| COM | Rd | One's Complement | Rd ← $FF – Rd | Z,C,N,V | 1 |
| NEG | Rd | Two's Complement | Rd ← $00 – Rd | Z,C,N,V,H | 1 |
| SBR | Rd,K | Set Bit(s) in Register | Rd ← Rd v K | Z,N,V | 1 |
| CBR | Rd,K | Clear Bit(s) in Register | Rd ← Rd • ($FF - K) | Z,N,V | 1 |
| INC | Rd | Increment | Rd ← Rd + 1 | Z,N,V | 1 |
| DEC | Rd | Decrement | Rd ← Rd – 1 | Z,N,V | 1 |
| TST | Rd | Test for Zero or Minus | Rd ← Rd • Rd | Z,N,V | 1 |
| CLR | Rd | Clear Register | Rd ← Rd ⊕ Rd | Z,N,V | 1 |
| SER | Rd | Set Register | Rd ← $FF | None | 1 |
| MUL | Rd, Rr | Multiply Unsigned | R1:R0 ← Rd x Rr | Z,C | 2 |
| MULS | Rd, Rr | Multiply Signed | R1:R0 ← Rd x Rr | Z,C | 2 |
| MULSU | Rd, Rr | Multiply Signed with Unsigned | R1:R0 ← Rd x Rr | Z,C | 2 |
| FMUL | Rd, Rr | Fractional Multiply Unsigned | R1:R0 ← (Rd x Rr) << 1 | Z,C | 2 |
| FMULS | Rd, Rr | Fractional Multiply Signed | R1:R0 ← (Rd x Rr) << 1 | Z,C | 2 |
| FMULSU | Rd, Rr | Fractional Multiply Signed with Unsigned | R1:R0 ← (Rd x Rr) << 1 | Z,C | 2 |
| **BRANCH INSTRUCTIONS** | | | | | |
| RJMP | k | Relative Jump | PC ← PC + k + 1 | None | 2 |
| IJMP | | Indirect Jump to (Z) | PC ← Z | None | 2 |
| JMP | k | Direct Jump | PC ← k | None | 3 |
| RCALL | k | Relative Subroutine Call | PC ← PC + k + 1 | None | 3 |
| ICALL | | Indirect Call to (Z) | PC ← Z | None | 3 |
| CALL | k | Direct Subroutine Call | PC ← k | None | 4 |
| RET | | Subroutine Return | PC ← STACK | None | 4 |
| RETI | | Interrupt Return | PC ← STACK | I | 4 |
| CPSE | Rd,Rr | Compare, Skip if Equal | if (Rd = Rr) PC ← PC + 2 or 3 | None | 1 / 2 / 3 |
| CP | Rd,Rr | Compare | Rd – Rr | Z, N,V,C,H | 1 |
| CPC | Rd,Rr | Compare with Carry | Rd – Rr – C | Z, N,V,C,H | 1 |
| CPI | Rd,K | Compare Register with Immediate | Rd – K | Z, N,V,C,H | 1 |
| SBRC | Rr, b | Skip if Bit in Register Cleared | if (Rr(b)=0) PC ← PC + 2 or 3 | None | 1 / 2 / 3 |
| SBRS | Rr, b | Skip if Bit in Register is Set | if (Rr(b)=1) PC ← PC + 2 or 3 | None | 1 / 2 / 3 |
| SBIC | P, b | Skip if Bit in I/O Register Cleared | if (P(b)=0) PC ← PC + 2 or 3 | None | 1 / 2 / 3 |
| SBIS | P, b | Skip if Bit in I/O Register is Set | if (P(b)=1) PC ← PC + 2 or 3 | None | 1 / 2 / 3 |
| BRBS | s, k | Branch if Status Flag Set | if (SREG(s) = 1) then PC←PC+k + 1 | None | 1 / 2 |
| BRBC | s, k | Branch if Status Flag Cleared | if (SREG(s) = 0) then PC←PC+k + 1 | None | 1 / 2 |
| BREQ | k | Branch if Equal | if (Z = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRNE | k | Branch if Not Equal | if (Z = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRCS | k | Branch if Carry Set | if (C = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRCC | k | Branch if Carry Cleared | if (C = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRSH | k | Branch if Same or Higher | if (C = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRLO | k | Branch if Lower | if (C = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRMI | k | Branch if Minus | if (N = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRPL | k | Branch if Plus | if (N = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRGE | k | Branch if Greater or Equal, Signed | if (N ⊕ V= 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRLT | k | Branch if Less Than Zero, Signed | if (N ⊕ V= 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRHS | k | Branch if Half Carry Flag Set | if (H = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRHC | k | Branch if Half Carry Flag Cleared | if (H = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRTS | k | Branch if T Flag Set | if (T = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRTC | k | Branch if T Flag Cleared | if (T = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRVS | k | Branch if Overflow Flag is Set | if (V = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRVC | k | Branch if Overflow Flag is Cleared | if (V = 0) then PC ← PC + k + 1 | None | 1 / 2 |
| BRIE | k | Branch if Interrupt Enabled | if ( I = 1) then PC ← PC + k + 1 | None | 1 / 2 |
| BRID | k | Branch if Interrupt Disabled | if ( I = 0) then PC ← PC + k + 1 | None | 1 / 2 |

## Instruction Set Summary (Continued)

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| **DATA TRANSFER INSTRUCTIONS** | | | | | |
| MOV | Rd, Rr | Move Between Registers | Rd ← Rr | None | 1 |
| MOVW | Rd, Rr | Copy Register Word | Rd+1:Rd ← Rr+1:Rr | None | 1 |
| LDI | Rd, K | Load Immediate | Rd ← K | None | 1 |
| LD | Rd, X | Load Indirect | Rd ← (X) | None | 2 |
| LD | Rd, X+ | Load Indirect and Post-Inc. | Rd ← (X), X ← X + 1 | None | 2 |
| LD | Rd, - X | Load Indirect and Pre-Dec. | X ← X - 1, Rd ← (X) | None | 2 |
| LD | Rd, Y | Load Indirect | Rd ← (Y) | None | 2 |
| LD | Rd, Y+ | Load Indirect and Post-Inc. | Rd ← (Y), Y ← Y + 1 | None | 2 |
| LD | Rd, - Y | Load Indirect and Pre-Dec. | Y ← Y - 1, Rd ← (Y) | None | 2 |
| LDD | Rd,Y+q | Load Indirect with Displacement | Rd ← (Y + q) | None | 2 |
| LD | Rd, Z | Load Indirect | Rd ← (Z) | None | 2 |
| LD | Rd, Z+ | Load Indirect and Post-Inc. | Rd ← (Z), Z ← Z+1 | None | 2 |
| LD | Rd, -Z | Load Indirect and Pre-Dec. | Z ← Z - 1, Rd ← (Z) | None | 2 |
| LDD | Rd, Z+q | Load Indirect with Displacement | Rd ← (Z + q) | None | 2 |
| LDS | Rd, k | Load Direct from SRAM | Rd ← (k) | None | 2 |
| ST | X, Rr | Store Indirect | (X) ← Rr | None | 2 |
| ST | X+, Rr | Store Indirect and Post-Inc. | (X) ← Rr, X ← X + 1 | None | 2 |
| ST | - X, Rr | Store Indirect and Pre-Dec. | X ← X - 1, (X) ← Rr | None | 2 |
| ST | Y, Rr | Store Indirect | (Y) ← Rr | None | 2 |
| ST | Y+, Rr | Store Indirect and Post-Inc. | (Y) ← Rr, Y ← Y + 1 | None | 2 |
| ST | - Y, Rr | Store Indirect and Pre-Dec. | Y ← Y - 1, (Y) ← Rr | None | 2 |
| STD | Y+q,Rr | Store Indirect with Displacement | (Y + q) ← Rr | None | 2 |
| ST | Z, Rr | Store Indirect | (Z) ← Rr | None | 2 |
| ST | Z+, Rr | Store Indirect and Post-Inc. | (Z) ← Rr, Z ← Z + 1 | None | 2 |
| ST | -Z, Rr | Store Indirect and Pre-Dec. | Z ← Z - 1, (Z) ← Rr | None | 2 |
| STD | Z+q,Rr | Store Indirect with Displacement | (Z + q) ← Rr | None | 2 |
| STS | k, Rr | Store Direct to SRAM | (k) ← Rr | None | 2 |
| LPM | | Load Program Memory | R0 ← (Z) | None | 3 |
| LPM | Rd, Z | Load Program Memory | Rd ← (Z) | None | 3 |
| LPM | Rd, Z+ | Load Program Memory and Post-Inc | Rd ← (Z), Z ← Z+1 | None | 3 |
| SPM | | Store Program Memory | (Z) ← R1:R0 | None | - |
| IN | Rd, P | In Port | Rd ← P | None | 1 |
| OUT | P, Rr | Out Port | P ← Rr | None | 1 |
| PUSH | Rr | Push Register on Stack | STACK ← Rr | None | 2 |
| POP | Rd | Pop Register from Stack | Rd ← STACK | None | 2 |
| **BIT AND BIT-TEST INSTRUCTIONS** | | | | | |
| SBI | P,b | Set Bit in I/O Register | I/O(P,b) ← 1 | None | 2 |
| CBI | P,b | Clear Bit in I/O Register | I/O(P,b) ← 0 | None | 2 |
| LSL | Rd | Logical Shift Left | Rd(n+1) ← Rd(n), Rd(0) ← 0 | Z,C,N,V | 1 |
| LSR | Rd | Logical Shift Right | Rd(n) ← Rd(n+1), Rd(7) ← 0 | Z,C,N,V | 1 |
| ROL | Rd | Rotate Left Through Carry | Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7) | Z,C,N,V | 1 |
| ROR | Rd | Rotate Right Through Carry | Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0) | Z,C,N,V | 1 |
| ASR | Rd | Arithmetic Shift Right | Rd(n) ← Rd(n+1), n=0..6 | Z,C,N,V | 1 |
| SWAP | Rd | Swap Nibbles | Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0) | None | 1 |
| BSET | s | Flag Set | SREG(s) ← 1 | SREG(s) | 1 |
| BCLR | s | Flag Clear | SREG(s) ← 0 | SREG(s) | 1 |
| BST | Rr, b | Bit Store from Register to T | T ← Rr(b) | T | 1 |
| BLD | Rd, b | Bit load from T to Register | Rd(b) ← T | None | 1 |
| SEC | | Set Carry | C ← 1 | C | 1 |
| CLC | | Clear Carry | C ← 0 | C | 1 |
| SEN | | Set Negative Flag | N ← 1 | N | 1 |
| CLN | | Clear Negative Flag | N ← 0 | N | 1 |
| SEZ | | Set Zero Flag | Z ← 1 | Z | 1 |
| CLZ | | Clear Zero Flag | Z ← 0 | Z | 1 |
| SEI | | Global Interrupt Enable | I ← 1 | I | 1 |
| CLI | | Global Interrupt Disable | I ← 0 | I | 1 |
| SES | | Set Signed Test Flag | S ← 1 | S | 1 |
| CLS | | Clear Signed Test Flag | S ← 0 | S | 1 |
| SEV | | Set Twos Complement Overflow. | V ← 1 | V | 1 |
| CLV | | Clear Twos Complement Overflow | V ← 0 | V | 1 |

## Instruction Set Summary (Continued)

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| SET | | Set T in SREG | T ← 1 | T | 1 |
| CLT | | Clear T in SREG | T ← 0 | T | 1 |
| SEH | | Set Half Carry Flag in SREG | H ← 1 | H | 1 |
| CLH | | Clear Half Carry Flag in SREG | H ← 0 | H | 1 |
| NOP | | No Operation | | None | 1 |
| SLEEP | | Sleep | (see specific descr. for Sleep function) | None | 3 |
| WDR | | Watchdog Reset | (see specific descr. for WDR/timer) | None | 1 |

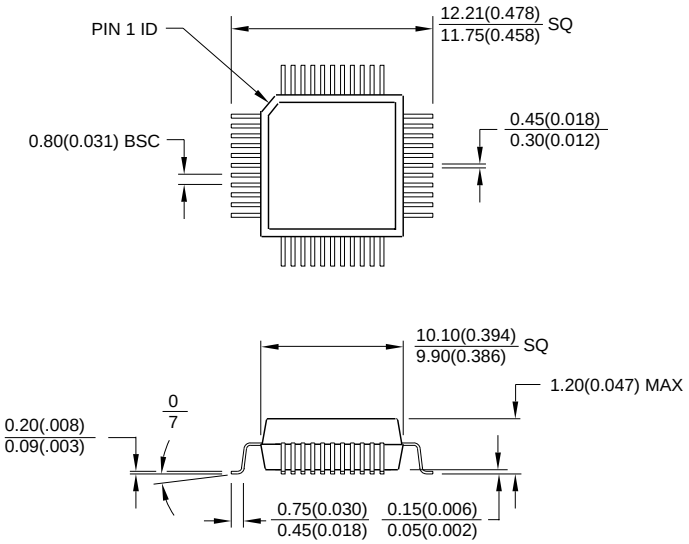
## Ordering Information

| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 4 | 2.7 - 5.5V | ATmega161-4AC<br>ATmega161-4JC<br>ATmega161-4PC | 44A<br>44J<br>40P6 | Commercial<br>(0°C to 70°C) |
| | | ATmega161-4AI<br>ATmega161-4JI<br>ATmega161-4PI | 44A<br>44J<br>40P6 | Industrial<br>(-40°C to 85°C) |
| 8 | 4.0 - 5.5V | ATmega161-8AC<br>ATmega161-8JC<br>ATmega161-8PC | 44A<br>44J<br>40P6 | Commercial<br>(0°C to 70°C) |
| | | ATmega161-8AI<br>ATmega161-8JI<br>ATmega161-8PI | 44A<br>44J<br>40P6 | Industrial<br>(-40°C to 85°C) |

| Package Type | |
|---|---|
| **44A** | 44-lead, Thin (1.0 mm) Plastic Gull-Wing Quad Flat Package (TQFP) |
| **44J** | 44-lead, Plastic J-Leaded Chip Carrier (PLCC) |
| **40P6** | 40-lead, 0.600” Wide, Plastic Dual-in-line Package (PDIP) |

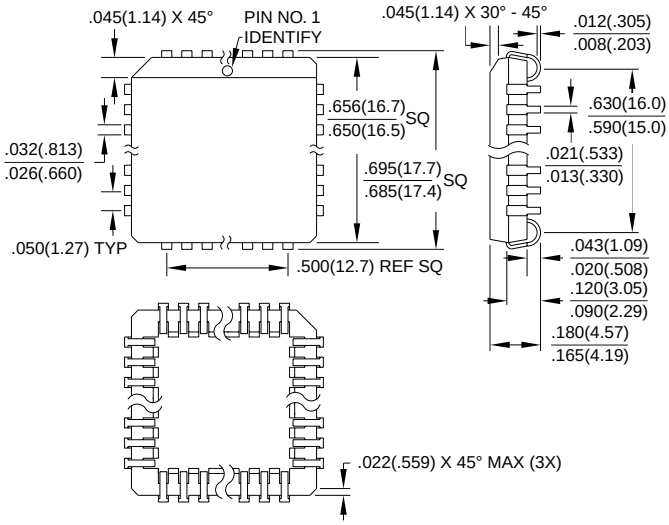
## Packaging Information

**44A,** 44-lead, Thin (1.0 mm) Plastic Gull-Wing Quad Flat Package (TQFP)
Dimensions in Millimeters and (Inches)*

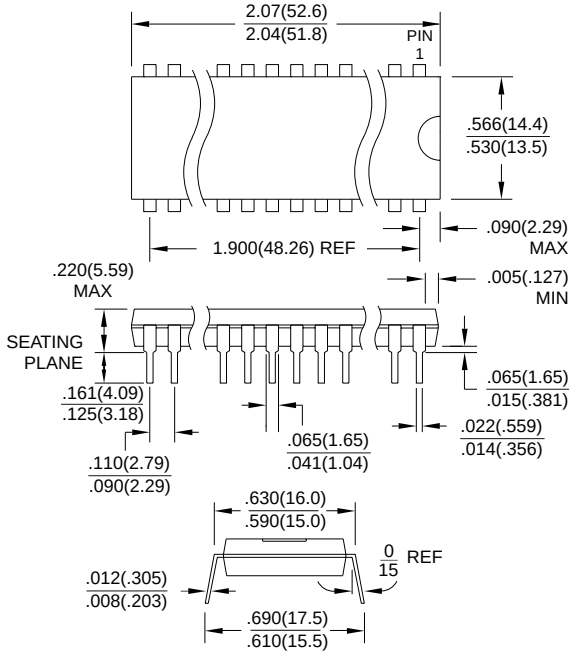



*Controlling dimension: millimeters

**44J,** 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)




**40P6,** 40-lead, 0.600" Wide,
Plastic Dual-in-line Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-011 AC



## Atmel Headquarters

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

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

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

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

## Atmel Operations

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

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

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

*e-mail*
literature@atmel.com

*Web Site*
http://www.atmel.com

*BBS*
1-(408) 436-4309



1228AS–08/99/xM