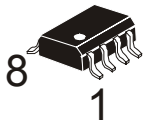
8-BIT MICROCONTROLLER WITH 2K BYTES BUILD-IN PROGRAMMABLE FLASH

Description



The IN90S2323D and IN90LS2323D is a low-power CMOS 8-bit microcontrollers based on the *AVR* enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the IN90S2323D and IN90LS2323D 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.

Features

- Utilizes the *AVR*® Enhanced RISC Architecture
- *AVR* - High Performance and Low Power RISC Architecture
- 118 Powerful Instructions - Most Single Clock Cycle Execution
- 2K bytes of In-System Programmable ISP Flash

– SPI Serial Interface for In-System Programming
– Endurance: 1,000 Write/Erase Cycles

- 128 bytes EEPROM

– Endurance: 100,000 Write/Erase Cycles

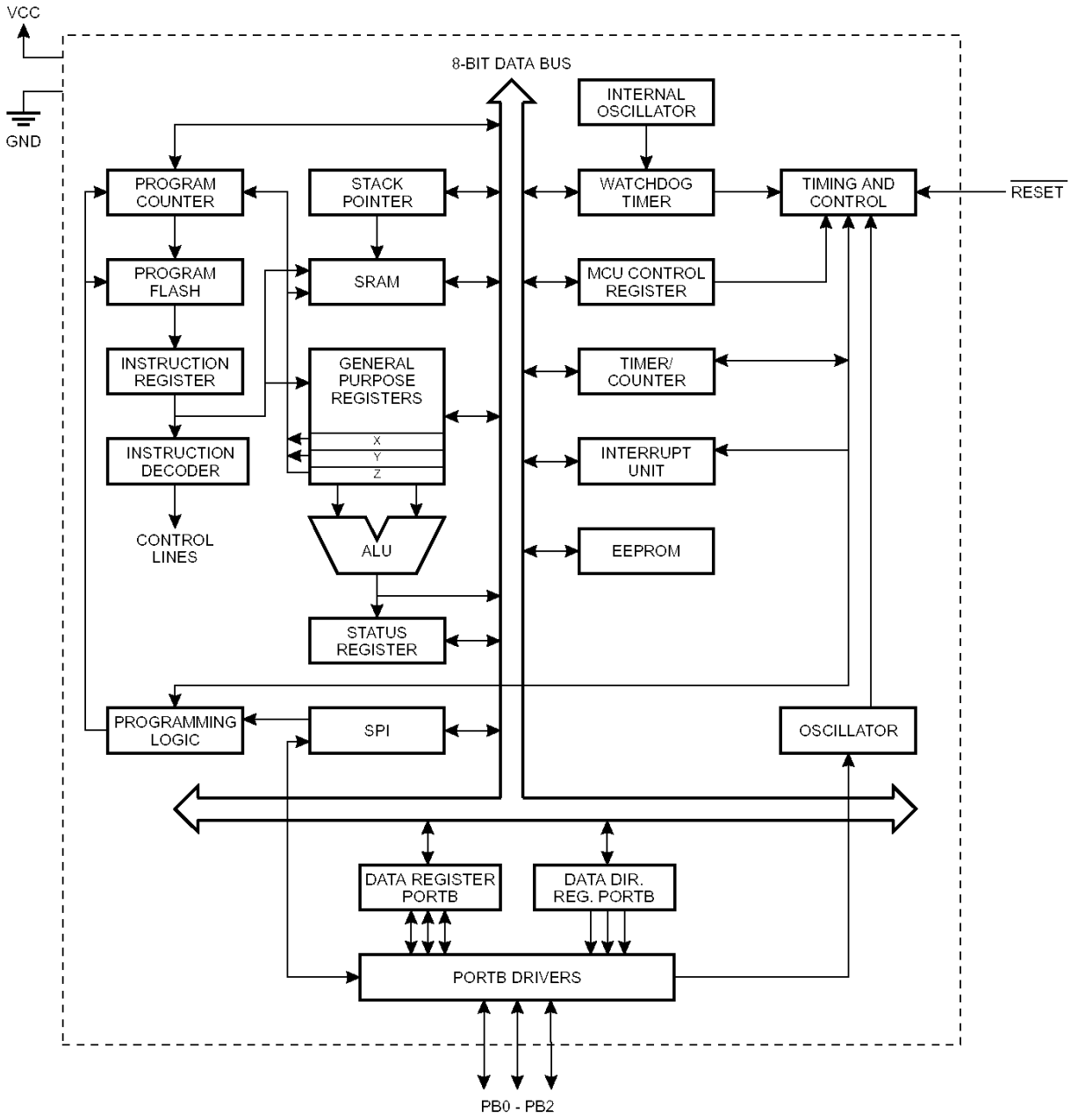
- 128 bytes Internal RAM
- 32 x 8 General Purpose Working Registers

– 3 Programmable I/O Lines
  - V_{CC}: 4.0 - 6.0V IN90S2323D
  - V_{CC}: 2.7 - 6.0V IN90LS2323D

- Power-On Reset Circuit
- Speed Grades: 0 - 10 MHz IN90S2323D
- Speed Grades: 0 - 4 MHz IN90LS2323D
- Up to 10 MIPS Throughput at 10 MHz
- One 8-Bit Timer/Counter with Separate Prescaler
- External and Internal Interrupt Sources
- Programmable Watchdog Timer with On-Chip Oscillator
- Low Power Idle and Power Down Modes
- Programming Lock for Flash Program and EEPROM Data Security
- Selectable On-Chip RC Oscillator
  - 8-Pin Device

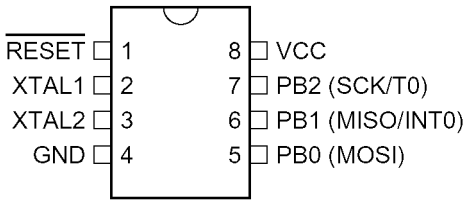
INTEGRAL

Block Diagram



INTEGRAL

Pin Descriptions



VCC

Supply voltage pin.

GND

Ground pin.

Port B (PB2..PB0)

Port B is a 3-bit bi-directional I/O port. Port pins can provide internal pull-up resistors (selected for each bit).

RESET

Reset input. A low on this pin for two machine cycles while the oscillator is running resets the device.

XTAL1

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

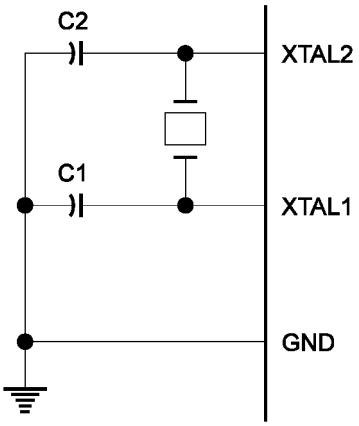
XTAL2

Output from the inverting oscillator amplifier.

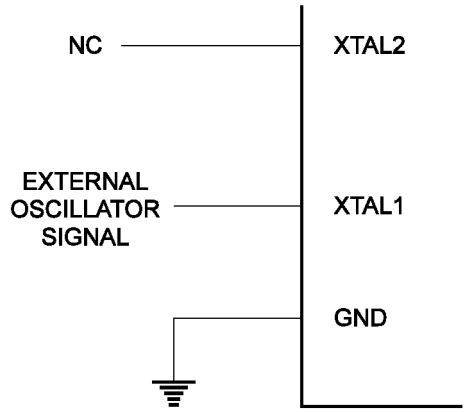
Clock Sources

The IN90S2313D and IN90LS2313D contains an inverting amplifier which can be configured for use as an on-chip oscillator. XTAL1 and XTAL2 are input and output respectively. Either a quartz crystal or a ceramic resonator may be used.

Oscillator Connection



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

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

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

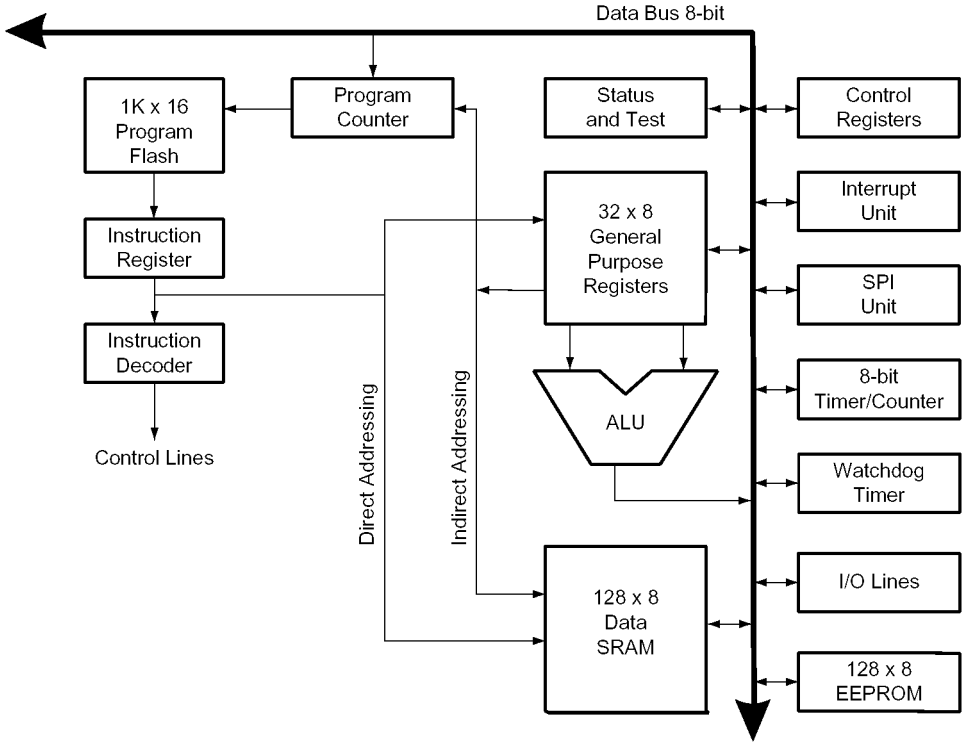
With the relative jump and call instructions, the whole 1K address space is directly accessed. Most *AVR* instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 8-bit stack pointer SP is read/write accessible in the I/O space.

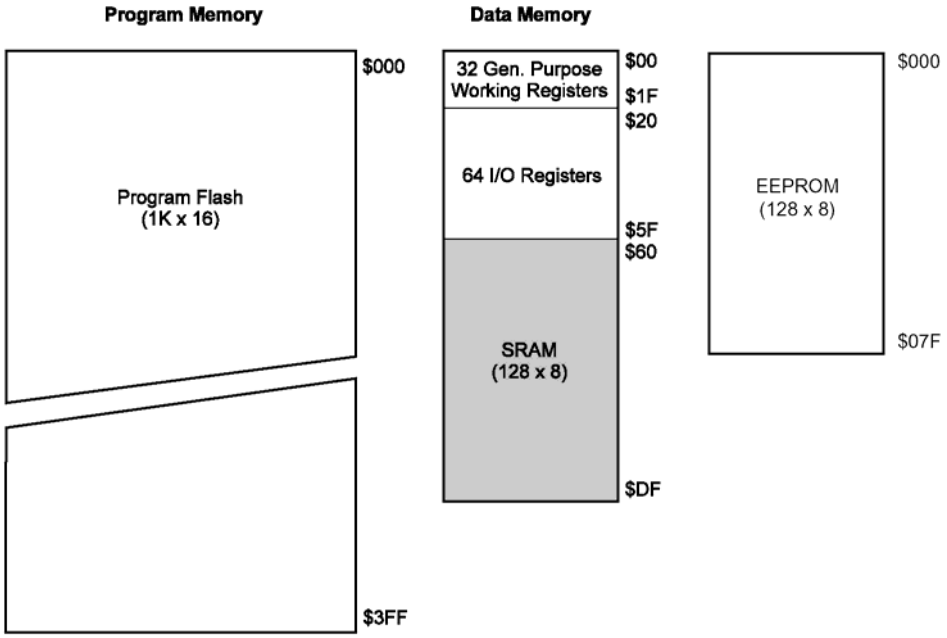
The 128 bytes data SRAM + register file and I/O registers can be easily accessed through the five different addressing modes supported in the *AVR* architecture.

The memory spaces in the *AVR* architecture are all linear and regular memory maps.

AVR Architecture



Memory Maps


Program Memory
Data Memory
$000
Program Flash
(1K x 16)
$3FF
32 Gen. Purpose
Working Registers
$00
$1F
$20
64 I/O Registers
$5F
$60
SRAM
(128 x 8)
$DF
$000
EEPROM
(128 x 8)
$07F

REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	page 13
$3E ($5E)	Reserved									
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 13
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	-	INT0	-	-	-	-	-	-	page 17
$3A ($5A)	GIFR	-	INTF0							page 17
$39 ($59)	TIMSK	-	-	-	-	-	-	TOIE0	-	page 15
$38 ($58)	TIFR	-	-	-	-	-	-	TOV0	-	page 16
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	-	-	SE	SM	-	-	ISC01	ISC00	page 16
$34 ($54)	MCUSR	-	-	-	-	-	-	EXTRF	PORF	page 14
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	page 20
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)								page 20
$31 ($51)	Reserved									
$30 ($50)	Reserved									
$2F ($4F)	Reserved									
$2E ($4E)	Reserved									
$2D ($4D)	Reserved									
$2C ($4C)	Reserved									
$2B ($4B)	Reserved									
$2A ($4A)	Reserved									
$29 ($49)	Reserved									
$28 ($48)	Reserved									
$27 ($47)	Reserved									
$26 ($46)	Reserved									
$25 ($45)	Reserved									
$24 ($44)	Reserved									
$23 ($43)	Reserved									
$22 ($42)	Reserved									
$21 ($41)	WDTCR	-	-	-	WDTO	WDE	WDP2	WDP1	WDP0	page 21
$20 ($40)	Reserved									
$1F ($3F)	Reserved									
$1E ($3E)	EEAR	-	EEPROM Address Register							page 22
$1D ($3D)	EEDR	EEPROM Data register								page 22
$1C ($3C)	EECR	-	-	-	-	-	EEMW	EEWE	EERE	page 22
$1B ($3B)	Reserved									
$1A ($3A)	Reserved									
$19 ($39)	Reserved									
$18 ($38)	PORTB	-	-	-	PORTB	PORTB	PORTB	PORTB	PORTB	page 23
$17 ($37)	DDRB	-	-	-	DDB4	DDB3	DDB2	DDB1	DDB0	page 23
$16 ($36)	PINB	-	-	-	PINB4	PINB3	PINB2	PINB1	PINB0	page 23
$15 ($35)	Reserved									
$14 ($34)	Reserved									
$13 ($33)	Reserved									
$12 ($32)	Reserved									
$11 ($31)	Reserved									
$10 ($30)	Reserved									
$0F ($2F)	Reserved									
$0E ($2E)	Reserved									
$0D ($2D)	Reserved									
$0C ($2C)	Reserved									
$0B ($2B)	Reserved									
$0A ($2A)	Reserved									
$09 ($29)	Reserved									
$08 ($28)	Reserved									
…	Reserved									
$00 ($20)	Reserved									

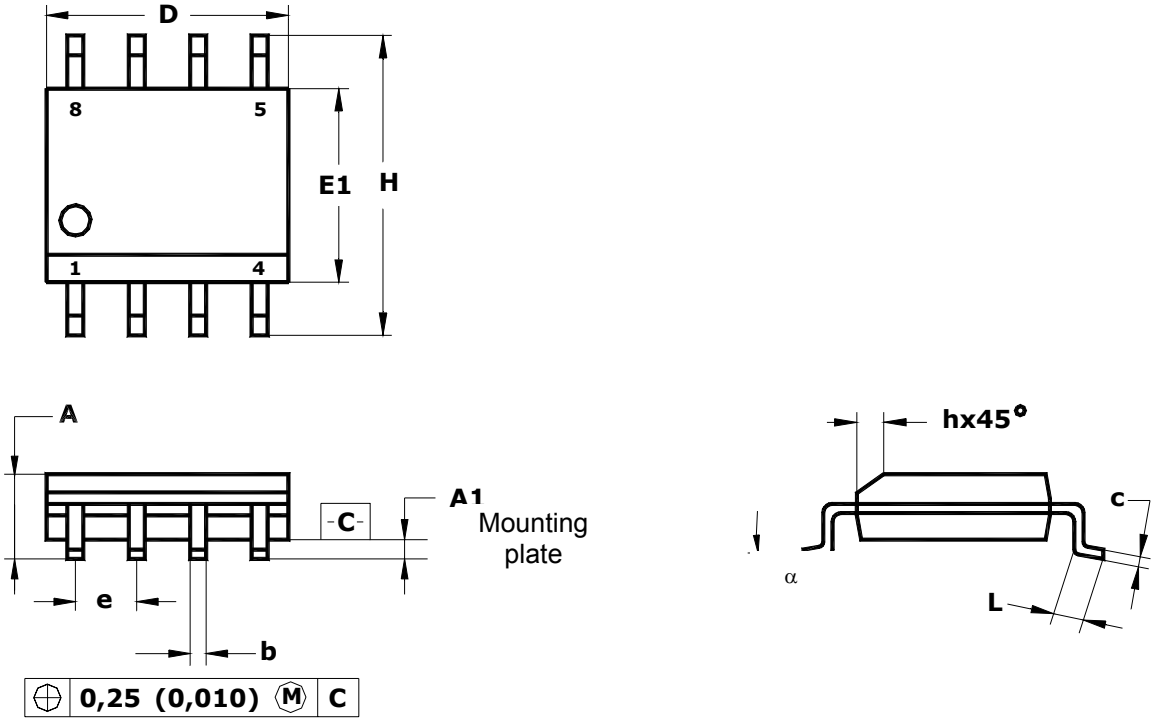
Instruction Set Summary

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

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep	None	3
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

MS-012AA Package dimensions



	D	E1	H	b	e	α	A	A1	c	L	h
mm											
min	4.80	3.80	5.80	0.33		0°	1.35	0.10	0.19	0.41	0.25
max	5.00	4.00	6.20	0.51	1.27	8°	1.75	0.25	0.25	1.27	0.50
inches											
min	0.1890	0.1497	0.2284	0.013		0°	0.0532	0.0040	0.0075	0.016	0.0099
max	0.1968	0.1574	0.2440	0.020	0.100	8°	0.0688	0.0090	0.0098	0.050	0.0196