### 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 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 8K Bytes of In-System Self-Programmable Flash Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation
  - 512 Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and MLF package Six Channels 10-bit Accuracy Two Channels 8-bit Accuracy
  - 6-channel ADC in PDIP package
    Four Channels 10-bit Accuracy
    Two Channels 8-bit Accuracy
  - 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
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad MLF
- Operating Voltages
  - 2.7 5.5V (ATmega8L)
  - 4.5 5.5V (ATmega8)
- Speed Grades
  - 0 8 MHz (ATmega8L)
  - 0 16 MHz (ATmega8)
- Power Consumption at 4 Mhz, 3V, 25°C
  - Active: 3.6 mA
  - Idle Mode: 1.0 mA
  - Power-down Mode: 0.5 µA



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

ATmega8 ATmega8L

# Summary



Note: This is a summary document. A complete document is available on our Web site at www.atmel.com.

Rev. 2486MS-AVR-12/03



### **Pin Configurations**





2

#### **Overview**

The ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

**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 ATmega8 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and MLF packages) where four (six) channels have 10-bit accuracy and two channels have 8-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five 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.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed 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. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega8 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.

**Disclaimer** Typical values contained in this datasheet 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.

## **Pin Descriptions**

| vcc   | Digital supply voltage.   |
|---|---|
| GND   | Ground.   |
| Port B (PB7PB0) XTAL1/<br>XTAL2/TOSC1/TOSC2 | 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 have symmetrical drive characteristics with both high sink and source capability. 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. |
|   | Depending on the clock selection fuse settings, PB6 can be used as input to the invert-<br>ing Oscillator amplifier and input to the internal clock operating circuit.  |
|   | Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.  |
|   | If the Internal Calibrated RC Oscillator is used as chip clock source, PB76 is used as TOSC21 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.  |
|   | The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 56 and "System Clock and Clock Options" on page 23.  |
| Port C (PC5PC0)                             | Port C is an 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. 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. |
| PC6/RESET                                   | If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electri-<br>cal characteristics of PC6 differ from those of the other pins of Port C.  |
|   | If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on<br>this pin for longer than the minimum pulse length will generate a Reset, even if the clock<br>is not running. The minimum pulse length is given in Table 15 on page 36. Shorter<br>pulses are not guaranteed to generate a Reset.   |
|   | The various special features of Port C are elaborated on page 59.   |
| Port D (PD7PD0)                             | Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. 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 ATmega8 as listed on page 61.   |
| RESET                                       | Reset input. A low level on this pin for longer than the minimum pulse length will gener-<br>ate a reset, even if the clock is not running. The minimum pulse length is given in Table<br>15 on page 36. Shorter pulses are not guaranteed to generate a reset.   |





| AVCC                                 | AVCC is the supply voltage pin for the A/D Converter, Port C (30), and ADC (76). 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. Note that Port C (54) use digital supply voltage, $V_{CC}$ . |
|--------------------------------------|--|
| AREF                                 | AREF is the analog reference pin for the A/D Converter.  |
| ADC76 (TQFP and MLF<br>Package Only) | In the TQFP and MLF package, ADC76 serve as analog inputs to the A/D converter.<br>These pins are powered from the analog supply and serve as 10-bit ADC channels.   |

# **Register Summary**

| Address                                   | Name     | Bit 7  | Bit 6  | Bit 5    | Bit 4             | Bit 3              | Bit 2       | Bit 1   | Bit 0  | Page             |
|---|----------|--------|--------|----------|-------------------|--------------------|-------------|---------|--------|------------------|
| 0x3F (0x5F)                               | SREG     | I      | т      | Н        | S                 | V                  | N           | Z       | С      | 9                |
| 0x3E (0x5E)                               | SPH      | _      | -      | _        | _                 | _                  | SP10        | SP9     | SP8    | 11               |
| 0x3D (0x5D)                               | SPL      | SP7    | SP6    | SP5      | SP4               | SP3                | SP2         | SP1     | SP0    | 11               |
| 0x3C (0x5C)                               | Reserved |        | •      |          |                   |                    | •           |         |        |                  |
| 0x3B (0x5B)                               | GICR     | INT1   | INT0   | -        | _                 | -                  | -           | IVSEL   | IVCE   | 47, 65           |
| 0x3A (0x5A)                               | GIFR     | INTF1  | INTF0  | -        | -                 | -                  | -           | -       | -      | 66               |
| 0x39 (0x59)                               | TIMSK    | OCIE2  | TOIE2  | TICIE1   | OCIE1A            | OCIE1B             | TOIE1       | -       | TOIE0  | 70, 100, 120     |
| 0x38 (0x58)                               | TIFR     | OCF2   | TOV2   | ICF1     | OCF1A             | OCF1B              | TOV1        | -       | TOV0   | 71, 101, 120     |
| 0x37 (0x57)                               | SPMCR    | SPMIE  | RWWSB  | -        | RWWSRE            | BLBSET             | PGWRT       | PGERS   | SPMEN  | 210              |
| 0x36 (0x56)                               | TWCR     | TWINT  | TWEA   | TWSTA    | TWSTO             | TWWC               | TWEN        | -       | TWIE   | 168              |
| 0x35 (0x55)                               | MCUCR    | SE     | SM2    | SM1      | SM0               | ISC11              | ISC10       | ISC01   | ISC00  | 31, 64           |
| 0x34 (0x54)                               | MCUCSR   | -      | -      | -        | -                 | WDRF               | BORF        | EXTRF   | PORF   | 39               |
| 0x33 (0x53)                               | TCCR0    | -      | -      | -        | -                 | -                  | CS02        | CS01    | CS00   | 70               |
| 0x32 (0x52)                               | TCNT0    |        |        |          | Timer/Cou         | nter0 (8 Bits)     |             |         |        | 70               |
| 0x31 (0x51)                               | OSCCAL   |        |        |          | Oscillator Cal    | ibration Register  |             |         |        | 29               |
| 0x30 (0x50)                               | SFIOR    | -      | -      | -        | -                 | ACME               | PUD         | PSR2    | PSR10  | 56, 73, 121, 190 |
| 0x2F (0x4F)                               | TCCR1A   | COM1A1 | COM1A0 | COM1B1   | COM1B0            | FOC1A              | FOC1B       | WGM11   | WGM10  | 95               |
| 0x2E (0x4E)                               | TCCR1B   | ICNC1  | ICES1  | -        | WGM13             | WGM12              | CS12        | CS11    | CS10   | 98               |
| 0x2D (0x4D)                               | TCNT1H   |        | •      | Time     | er/Counter1 - Co  | unter Register Hig | gh byte     | •       | •      | 99               |
| 0x2C (0x4C)                               | TCNT1L   |        |        |          | er/Counter1 – Co  |                    |             |         |        | 99               |
| 0x2B (0x4B)                               | OCR1AH   |        |        |          | unter1 – Output C |                    |             |         |        | 99               |
| 0x2A (0x4A)                               | OCR1AL   |        |        | Timer/Co | unter1 – Output ( | Compare Register   | A Low byte  |         |        | 99               |
| 0x29 (0x49)                               | OCR1BH   |        |        | Timer/Co | unter1 – Output C | compare Register   | B High byte |         |        | 99               |
| 0x28 (0x48)                               | OCR1BL   |        |        | Timer/Co | unter1 – Output ( | Compare Register   | B Low byte  |         |        | 99               |
| 0x27 (0x47)                               | ICR1H    |        |        | Timer/0  | Counter1 – Input  | Capture Register   | High byte   |         |        | 100              |
| 0x26 (0x46)                               | ICR1L    |        |        |          | Counter1 – Input  |                    | * *         |         |        | 100              |
| 0x25 (0x45)                               | TCCR2    | FOC2   | WGM20  | COM21    | COM20             | WGM21              | CS22        | CS21    | CS20   | 115              |
| 0x24 (0x44)                               | TCNT2    |        |        |          |                   | nter2 (8 Bits)     |             |         |        | 117              |
| 0x23 (0x43)                               | OCR2     |        |        | Tir      | mer/Counter2 Ou   |                    | gister      |         |        | 117              |
| 0x22 (0x42)                               | ASSR     | -      | -      | -        | _                 | AS2                | TCN2UB      | OCR2UB  | TCR2UB | 117              |
| 0x21 (0x41)                               | WDTCR    | -      | -      | -        | WDCE              | WDE                | WDP2        | WDP1    | WDP0   | 41               |
|   | UBRRH    | URSEL  | -      | _        | -                 |                    | UBR         | R[11:8] |        | 155              |
| 0x20 <sup>(1)</sup> (0x40) <sup>(1)</sup> | UCSRC    | URSEL  | UMSEL  | UPM1     | UPM0              | USBS               | UCSZ1       | UCSZ0   | UCPOL  | 153              |
| 0x1F (0x3F)                               | EEARH    | -      | -      | -        | -                 | -                  | -           | _       | EEAR8  | 18               |
| 0x1E (0x3E)                               | EEARL    | EEAR7  | EEAR6  | EEAR5    | EEAR4             | EEAR3              | EEAR2       | EEAR1   | EEAR0  | 18               |
| 0x1D (0x3D)                               | EEDR     |        | •      | •        | EEPROM            | Data Register      | •           | •       |        | 18               |
| 0x1C (0x3C)                               | EECR     | -      | -      | -        | -                 | EERIE              | EEMWE       | EEWE    | EERE   | 18               |
| 0x1B (0x3B)                               | Reserved |        | •      |          |                   |                    | •           |         |        |                  |
| 0x1A (0x3A)                               | Reserved |        |        |          |                   |                    |             |         |        |                  |
| 0x19 (0x39)                               | Reserved |        |        |          |                   |                    |             |         |        |                  |
| 0x18 (0x38)                               | PORTB    | PORTB7 | PORTB6 | PORTB5   | PORTB4            | PORTB3             | PORTB2      | PORTB1  | PORTB0 | 63               |
| 0x17 (0x37)                               | DDRB     | DDB7   | DDB6   | DDB5     | DDB4              | DDB3               | DDB2        | DDB1    | DDB0   | 63               |
| 0x16 (0x36)                               | PINB     | PINB7  | PINB6  | PINB5    | PINB4             | PINB3              | PINB2       | PINB1   | PINB0  | 63               |
| 0x15 (0x35)                               | PORTC    | -      | PORTC6 | PORTC5   | PORTC4            | PORTC3             | PORTC2      | PORTC1  | PORTC0 | 63               |
| 0x14 (0x34)                               | DDRC     | -      | DDC6   | DDC5     | DDC4              | DDC3               | DDC2        | DDC1    | DDC0   | 63               |
| 0x13 (0x33)                               | PINC     | -      | PINC6  | PINC5    | PINC4             | PINC3              | PINC2       | PINC1   | PINC0  | 63               |
| 0x12 (0x32)                               | PORTD    | PORTD7 | PORTD6 | PORTD5   | PORTD4            | PORTD3             | PORTD2      | PORTD1  | PORTD0 | 63               |
| 0x11 (0x31)                               | DDRD     | DDD7   | DDD6   | DDD5     | DDD4              | DDD3               | DDD2        | DDD1    | DDD0   | 63               |
| 0x10 (0x30)                               | PIND     | PIND7  | PIND6  | PIND5    | PIND4             | PIND3              | PIND2       | PIND1   | PIND0  | 63               |
| 0x0F (0x2F)                               | SPDR     |        |        |          | SPI Da            | ta Register        |             |         |        | 128              |
| 0x0E (0x2E)                               | SPSR     | SPIF   | WCOL   | -        | -                 | -                  | -           | -       | SPI2X  | 128              |
| 0x0D (0x2D)                               | SPCR     | SPIE   | SPE    | DORD     | MSTR              | CPOL               | СРНА        | SPR1    | SPR0   | 126              |
| 0x0C (0x2C)                               | UDR      |        |        |          |                   | Data Register      |             |         |        | 150              |
| 0x0B (0x2B)                               | UCSRA    | RXC    | TXC    | UDRE     | FE                | DOR                | PE          | U2X     | MPCM   | 151              |
| 0x0A (0x2A)                               | UCSRB    | RXCIE  | TXCIE  | UDRIE    | RXEN              | TXEN               | UCSZ2       | RXB8    | TXB8   | 152              |
| 0x09 (0x29)                               | UBRRL    |        |        |          | USART Baud Ra     |                    |             |         |        | 155              |
| 0x08 (0x28)                               | ACSR     | ACD    | ACBG   | ACO      | ACI               | ACIE               | ACIC        | ACIS1   | ACIS0  | 191              |
| 0x07 (0x27)                               | ADMUX    | REFS1  | REFS0  | ADLAR    | _                 | MUX3               | MUX2        | MUX1    | MUX0   | 202              |
| 0x06 (0x26)                               | ADCSRA   | ADEN   | ADSC   | ADFR     | ADIF              | ADIE               | ADPS2       | ADPS1   | ADPS0  | 204              |
| 0x05 (0x25)                               | ADCH     |        |        |          |                   | egister High byte  |             |         |        | 205              |
| , ,                                       | ADCL     | İ      |        |          |                   | egister Low byte   |             |         |        | 205              |
| 0x04(0x24)                                |          |        |        |          |                   | *                  |             |         |        |                  |
| 0x04 (0x24)<br>0x03 (0x23)                | TWDR     |        |        | т        | wo-wire Serial In | tertace Data Regi  | ister       |         |        | 170              |





### **Register Summary (Continued)**

| Address     | Name | Bit 7 | Bit 6                                       | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|-------------|------|-------|---|-------|-------|-------|-------|-------|-------|------|
| 0x01 (0x21) | TWSR | TWS7  | TWS6  | TWS5  | TWS4  | TWS3  | -     | TWPS1 | TWPS0 | 170  |
| 0x00 (0x20) | TWBR |       | Two-wire Serial Interface Bit Rate Register |       |       |       |       |       | 168   |      |

Notes: 1. Refer to the USART description for details on how to access UBRRH and UCSRC.

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

3. 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 0x00 to 0x1F only.

## Instruction Set Summary

| Mnemonics                    | Operands          | Description   | Operation   | Flags        | #Clocks    |
|------------------------------|-------------------|---|---|--------------|------------|
| ARITHMETIC AND L             | OGIC INSTRUCTIONS | 8   |   |              |            |
| ADD                          | Rd, Rr            | Add two Registers   | $Rd \gets Rd + Rr$  | Z,C,N,V,H    | 1          |
| ADC                          | Rd, Rr            | Add with Carry two Registers                              | $Rd \leftarrow Rd + Rr + C$   | Z,C,N,V,H    | 1          |
| ADIW                         | Rdl,K             | Add Immediate to Word                                     | $Rdh{:}Rdl \gets Rdh{:}Rdl + K$   | Z,C,N,V,S    | 2          |
| SUB                          | Rd, Rr            | Subtract two Registers                                    | $Rd \leftarrow Rd - Rr$   | Z,C,N,V,H    | 1          |
| SUBI                         | Rd, K             | Subtract Constant from Register                           | $Rd \gets Rd - K$   | Z,C,N,V,H    | 1          |
| SBC                          | Rd, Rr            | Subtract with Carry two Registers                         | $Rd \gets Rd - Rr - C$  | Z,C,N,V,H    | 1          |
| SBCI                         | Rd, K             | Subtract with Carry Constant from Reg.                    | $Rd \gets Rd - K - C$   | 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                                     | $Rd \gets Rd \bullet Rr$  | Z,N,V        | 1          |
| ANDI                         | Rd, K             | Logical AND Register and Constant                         | $Rd \gets Rd \bullet K$   | Z,N,V        | 1          |
| OR                           | Rd, Rr            | Logical OR Registers                                      | $Rd \leftarrow Rd \lor Rr$  | Z,N,V        | 1          |
| ORI                          | Rd, K             | Logical OR Register and Constant                          | $Rd \leftarrow Rd \lor K$   | Z,N,V        | 1          |
| EOR                          | Rd, Rr            | Exclusive OR Registers                                    | $Rd \gets Rd \oplus Rr$   | Z,N,V        | 1          |
| COM                          | Rd                | One's Complement  | $Rd \leftarrow 0xFF - Rd$   | Z,C,N,V      | 1          |
| NEG                          | Rd                | Two's Complement  | $Rd \leftarrow 0x00 - Rd$   | Z,C,N,V,H    | 1          |
| SBR                          | Rd,K              | Set Bit(s) in Register                                    | $Rd \gets Rd \lor K$  | Z,N,V        | 1          |
| CBR                          | Rd,K              | Clear Bit(s) in Register                                  | $Rd \gets Rd \bullet (0xFF -K)$   | Z,N,V        | 1          |
| INC                          | Rd                | Increment   | $Rd \leftarrow Rd + 1$  | Z,N,V        | 1          |
| DEC                          | Rd                | Decrement   | $Rd \leftarrow Rd - 1$  | Z,N,V        | 1          |
| TST                          | Rd                | Test for Zero or Minus                                    | $Rd \gets Rd \bullet Rd$  | Z,N,V        | 1          |
| CLR                          | Rd                | Clear Register  | $Rd  \leftarrow Rd \oplus Rd$   | Z,N,V        | 1          |
| SER                          | Rd                | Set Register  | $Rd \leftarrow 0xFF$  | None         | 1          |
| MUL                          | Rd, Rr            | Multiply Unsigned   | $R1:R0 \leftarrow Rd \times Rr$   | Z,C          | 2          |
| MULS                         | Rd, Rr            | Multiply Signed   | $R1:R0 \leftarrow Rd \times Rr$   | Z,C          | 2          |
| MULSU                        | Rd, Rr            | Multiply Signed with Unsigned                             | $R1:R0 \leftarrow Rd \times Rr$   | Z,C          | 2          |
| FMUL                         | Rd, Rr            | Fractional Multiply Unsigned                              | $R1:R0 \leftarrow (Rd \times Rr) << 1$  | Z,C          | 2          |
| FMULS                        | Rd, Rr            | Fractional Multiply Signed                                | $R1:R0 \leftarrow (Rd \times Rr) << 1$  | Z,C          | 2          |
| FMULSU                       | Rd, Rr            | Fractional Multiply Signed with Unsigned                  | $R1:R0 \leftarrow (Rd \times Rr) << 1$  | Z,C          | 2          |
| BRANCH INSTRUC               | TIONS             |   |   |              |            |
| RJMP                         | k                 | Relative Jump   | $PC \leftarrow PC + k + 1$  | None         | 2          |
| IJMP                         |                   | Indirect Jump to (Z)                                      | $PC \leftarrow Z$   | None         | 2          |
| RCALL                        | k                 | Relative Subroutine Call                                  | $PC \gets PC + k + 1$   | None         | 3          |
| ICALL                        |                   | Indirect Call to (Z)                                      | $PC \gets Z$  | None         | 3          |
| RET                          |                   | Subroutine Return   | $PC \gets STACK$  | None         | 4          |
| RETI                         |                   | Interrupt Return  | $PC \leftarrow STACK$   | 1            | 4          |
| CPSE                         | Rd,Rr             | Compare, Skip if Equal                                    | if (Rd = Rr) PC $\leftarrow$ 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 $\leftarrow$ PC + 2 or 3  | None         | 1/2/3      |
| SBRS                         | Rr, b             | Skip if Bit in Register is Set                            | if (Rr(b)=1) PC $\leftarrow$ PC + 2 or 3  | None         | 1/2/3      |
| SBIC                         | P, b              | Skip if Bit in I/O Register Cleared                       | if (P(b)=0) PC $\leftarrow$ 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 $\leftarrow$ PC + 2 or 3   | None         | 1/2/3      |
| BRBS                         | s, k              | Branch if Status Flag Set                                 | if $(SREG(s) = 1)$ then $PC \leftarrow PC+k + 1$  | None         | 1 / 2      |
| BRBC                         | s, k              | Branch if Status Flag Cleared                             | if $(SREG(s) = 0)$ then $PC \leftarrow PC + k + 1$  | None         | 1 / 2      |
| BREQ                         | k                 | Branch if Equal   | if (Z = 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRNE                         | k                 | Branch if Not Equal                                       | if (Z = 0) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRCS                         | k                 | Branch if Carry Set                                       | if (C = 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRCC                         | k                 | Branch if Carry Cleared                                   | if (C = 0) then PC $\leftarrow$ PC + k + 1  | None         | 1/2        |
| BRSH                         | k                 | Branch if Same or Higher                                  | if (C = 0) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRLO                         | k                 | Branch if Lower   | if (C = 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRMI                         | k                 | Branch if Minus   | if (N = 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRPL                         | k                 | Branch if Plus  | if (N = 0) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRGE                         | k                 | Branch if Greater or Equal, Signed                        | if (N $\oplus$ V= 0) then PC $\leftarrow$ PC + k + 1  | None         | 1/2        |
|                              | k                 | Branch if Less Than Zero, Signed                          | if (N $\oplus$ V= 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRLT                         | k                 | Branch if Half Carry Flag Set                             | if (H = 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRHS                         |                   |   | if (III = 0) there BO = BO = Ir = 1   | None         | 1/2        |
|                              | k                 | Branch if Half Carry Flag Cleared                         | if (H = 0) then PC $\leftarrow$ PC + k + 1  | None         |            |
| BRHS                         | k<br>k            | Branch if Half Carry Flag Cleared<br>Branch if T Flag Set | if (H = 0) then PC $\leftarrow$ PC + k + 1<br>if (T = 1) then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRHS<br>BRHC                 |                   |   |   |              |            |
| BRHS<br>BRHC<br>BRTS         | k                 | Branch if T Flag Set                                      | if $(T = 1)$ then PC $\leftarrow$ PC + k + 1  | None         | 1 / 2      |
| BRHS<br>BRHC<br>BRTS<br>BRTC | k<br>k            | Branch if T Flag Set<br>Branch if T Flag Cleared          | $\begin{array}{l} \mbox{if } (T=1) \mbox{ then } PC \leftarrow PC + k \ + 1 \\ \mbox{if } (T=0) \mbox{ then } PC \leftarrow PC + k \ + 1 \end{array}$ | None<br>None | 1/2<br>1/2 |





| DATABASEGE DEVICEDAVAROLPYNone Service RigideRol-PARol-PANoneIMOVRd, FACory Rigide WordRol-PARol-PANoneILORd, KLoad InfractaRd - KNoneILDRd, XLoad InfractaRd - KNoneILDRd, XLoad InfractaRd - CNoneILDRd, YLoad InfractaRd - CNoneILDRd, ZLoad InfractaNone <td< th=""><th>BRID</th><th>к<br/>k</th><th>Branch if Interrupt Enabled<br/>Branch if Interrupt Disabled</th><th>if <math>(I = 1)</math> then PC <math>\leftarrow</math> PC + k + 1<br/>if <math>(I = 0)</math> then PC <math>\leftarrow</math> PC + k + 1</th><th>None<br/>None</th><th>1/2</th></td<>   | BRID             | к<br>k       | Branch if Interrupt Enabled<br>Branch if Interrupt Disabled | if $(I = 1)$ then PC $\leftarrow$ PC + k + 1<br>if $(I = 0)$ then PC $\leftarrow$ PC + k + 1 | None<br>None | 1/2     |
|--|------------------|--------------|---|--|--------------|---------|
| MOVNo. No. No. No. No. No. No. No. No. No.   |                  |              | Branch in Interrupt Disabled                                | If $(1=0)$ (nen FC $\leftarrow$ FC + k + 1   | None         | 1/2     |
| MOVWMo. Roy Copy Sogne WorkPath InformationPath InformationNoneInformationLDRd, KLand NumberRd + KNone <td< td=""><td></td><td></td><td>Move Between Pegisters</td><td>Pd / Pr</td><td>Nono</td><td>1</td></td<>   |                  |              | Move Between Pegisters                                      | Pd / Pr  | Nono         | 1       |
| LintPick NLoad IndicationPic + NNoneNoneLDPG XLoad Indicat and Pace Inc.Pic + N/1NoneNoneLDPG XLoad Indicat and Pace Inc.Pic + N/1 NNoneNoneLDRG XLoad Indicat and Pace Inc.Pic + N/1 NNoneNoneLDRG YLoad Indicat and Pace Inc.Pic + N/1 NNoneNoneLDRG YLoad Indicat and Pace Inc.Pic + N/1 NNoneNoneLDRG YLoad Indicat and Pace Inc.Pic + N/2 - Y.1NoneNoneLDRG 2Load Indicat and Pace Inc.Pic + D/2 - 21NoneNoneLDRG 2Load Indicat and Pace Inc.Pic + D/2 - 21NoneNoneLDRG 2Load Indicat and Pace Inc.Pic + D/2 - 21NoneNoneLDRG 2Load Indicat and Pace Inc.Pic + D/2 - 21NoneNoneLDRG 2Load Indicat and Pace Inc.Pic + D/2 - 21NoneNoneLDRG 2Load Indicat and Pace Inc.Pic + D/2 - 21NoneNoneSTXNSome IndicaPic + D/2 - NNoneNoneSTXNSome Indica and Pace Inc.Pic + D/2 - NNoneSTXNSome Indica and Pace Inc.Pic + N - 11NoneSTXNSome Indica and Pace Inc.Pic + N - 11NoneSTXNSome Indica and Pace Inc.Pic + N - 11None <t< td=""><td>-</td><td></td><td></td><td></td><td></td><td>1</td></t<>  | -                |              |   |  |              | 1       |
| LDNDNDNormaNormaLDR0 X-Load indicat and Pre-Dec.R4 - XX, X-X-1, ND - XXNormaNormaLDR0 X-Load indicat and Pre-Dec.R4 - XX, X-X-1, ND - XXNormaNormaLDR0 X-Load indicat and Pre-Dec.R4 - YX, X-Y + 1NormaNormaLDR0 X-Load indicat and Pre-Dec.Y - Y1, Nd - Y1, Nd - Y1NormaNormaLDR0 X-Load indicat and Pre-Dec.Y - Y1, Nd - Y1NormaNormaLDR0 Z-Load indicat and Pre-Dec.R4 - Y2 - 10NormaNormaLDR0 Z-Load indicat and Pre-Dec.R2 - 21, ND - CDNormaNormaLDR0 Z-Load indicat and Pre-Dec.R2 - 21, ND - CDNormaNormaLDR0 Z-Load indicat and Pre-Dec.R2 - 21, ND - CDNormaNormaLDR0 Z-Load indicat and Pre-Dec.R2 - 21, ND - CDNormaNormaLDR1 Z-ND - P1NormaNormaNormaNormaLDND - P1NormaNormaNormaNormaNormaLDND - P1NormaNormaNormaNormaNormaLDND - P1NormaNormaNormaNormaNormaLDND - P1NormaNormaNormaNormaNormaLDND - P1NormaNormaNormaNormaNormaSTN, RStore InformaNormaNormaNormaNorma <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>   |                  |              |   |  |              |         |
| LDNALast invict and PracingPR - (0, X + X + 1)NoneNoneLDR4, XLast invict and ProbeX - X 1, R4 + (0,0)NoneNoneLDR4, YLast invict and ProbePR - (Y, Y + Y, 1)NoneNoneLDR4, YLast invict and ProbePR - (Y, Y + Y, 1)NoneNoneLDR4, YLast invict and ProbePR - (Y, Y + Y, 1)NoneNoneLDR4, ZLast invict and ProbePR - (Y, 2)NoneNoneLDR4, ZLast invict and ProbePR - (Y, 2)NoneNoneLDR4, ZLast invict and ProbePR - (Z, 2, 2, 1)NoneNoneLDR4, ZLast invict and ProbePR - (Z, 2)NoneNoneNoneLDR4, ZLast invict and ProbePR - (Z, 2, 2, 2, 1)NoneNoneNoneLDR4, ZLast invict and ProbePR - (Z, 2, 2, 2, 1)NoneNoneNoneLDR4, ZLast invict and ProbePR - (Z, 2, 2, 2, 1)NoneNoneNoneLDR4, ZLast invict and ProbePR - (Y, 1)NoneNoneNoneNoneSTX, BrStore Inford and ProbeY - (Y, 1)PRNoneNoneNoneSTX, BrStore Inford and ProbeY - (Y, 1)PRNoneNoneNoneSTX, BrStore Inford and ProbeY - (Y, 1)PRNoneNoneNoneSTX, BrStore Inford and Probe <td></td> <td></td> <td></td> <td></td> <td></td> <td>1</td>  |                  |              |   |  |              | 1       |
| LDRdLoad Indired and Pre-Des.X = X + 1, Rd = (3)Nee (3)LDRd YLoad Indired and Pas-Inc.Rd = (V, Y + Y + 1)NoteNeeLDRd YLoad Indired and Pas-Des.Y = Y + 1, Rd = (Y)NeeNeeLD0Rd Y_qLoad Indired and Pas-Des.Y = Y + 1, Rd = (Y)NeeNeeLD0Rd Y_qLoad Indired the DepartmentRd = (Y, Q)NeeNeeLD0Rd ZLoad Indired the DepartmentRd = (Z), Z = Z + 1NeeNeeLD0Rd ZLoad Indired the DepartmentRd = (Z), Z = Z + 1NeeNeeLD0Rd ZLoad Indired the DepartmentRd = (Z), Q = QNeeNeeLD1Rd ZLoad Indired the DepartmentRd = (Z), Q = QNeeNeeLD2Rd ZLoad Indired the DepartmentRd = (Z), Q = QNeeNeeLD3Rd ALoad Dref the SAMRd = (X), C = QNeeNeeNeeSTX, RfStore InfriedM Per SANeeNeeNeeNeeSTY, RfStore Infried and Per Des.X = X + 1, (M) = RNeeNeeNeeSTY, RfStore Infried and Per Des.X = X + 1, (M) = RNeeNeeNeeSTY, RfStore Infried and Per Des.X = X + 1, (M) = RNeeNeeNeeSTY, RfStore Infried and Per Des.X = X + 1, (M) = RNeeNeeNeeSTY, RfStore Infried and Per Des.X = X + 1, (M) =   |                  |              |   |  |              | 2       |
| LD      Rd γ      Loga Indicat and Poole.      Rd - (γ) - (γ)      None      In        LD      Rd γ+      Loga Indicat and Poole.      Y + γ + 1, Rd - (γ)      None      In        LD      Rd γ-q      Loga Indicat and Poole.      Y + γ + 1, Rd - (γ)      None      In        LD      Rd γ-q      Loga Indicat and Poole.      Rd - (Z)      None      In        LD      Rd Z-      Loga Indicat and Poole.      Rd - (Z)      None      In        LD      Rd Z-      Loga Indicat and Poole.      Rd - (Z)      None      In        LD      Rd Z-      Loga Indicat and Poole.      Rd - (Z)      None      In        LD      Rd Z-      Loga Indicat and Poole.      X = (X - Z)      None      In        ST      X, R      Store Indicat and Poole.      X = X + (X + 1)      None      In        ST      X, R      Store Indicat and Poole.      X = X + (X + 1)      None      In        ST      X, R      Store Indicat and Poole.      X = X + (X + 1)      None      In        ST      Y, R      Store Indicat and Poole.      X =   |                  |              |   | · · · · ·  |              | 2       |
| LDRd. Y.Load inforce and Pasiho.Rd. Y. Y. J. Hen (Y)NoneNoneLD0Rd. Y.Load inforce and Poolon.Y L-Y. J. Ren (Y)NoneNoneNoneLD0Rd. Y.Load inforce and Poolon.Rd. (Y, -g)NoneNoneNoneNoneLDRd. Z.Load inforce and Poolon.Rd. (Z, 2, -Z, 1)None  |                  |              |   |  |              | 2       |
| LDRdYLog Indrex Mo Pre-Des.YYNeeNeeLDRdLog Indrex Mo DegacementRd - (2)NeeNeeNeeLDRdZLog Indrex Mo DegacementRd - (2)NeeNeeLDRdZLog Indrex Mo DegacementRd - (2)NeeNeeLDRdLog Indrex Mo DegacementRd - (2)NeeNeeSTX, RrStore Indrex Mo DegacementRd - (2)NeeNeeSTX, RrStore Indrex Mo Degacement(2) - (R, X - X + 1NeeNeeSTX, RrStore Indrex Mo Degacement(2) - (R, X - X + 1)NeeNeeSTY, RrStore Indrex Mo PeoLeY - Y - (1) - (FrNeeNeeSTY, RrStore Indrex Mo PeoLe(2) - (FrNeeNeeSTZ, RrStore Indrex Mo Peo  |                  | Rd, Y        | Load Indirect   |  | None         | 2       |
| LDD      Bd.Y = Lad Individ xm Digitations      Bd = (r), 2 = (r), 2 = 2.1      None      None        LD      Bd,Z      Lada Individ xm Pagtene,      Bd + (r), Z = Z.1      None      None        LD      Bd,Z      Lada Individ xm Pagtene,      Z = Z.1 Bd + (r),      None      None        LDD      Bd,Z      Lada Individ xm Digatomat      Bd + (r), 2 = Z.1      None      None        LDS      Bd,X      Lada Orient two Digatomat      Bd + (r), 1 Bd + (r),      None      None        ST      X, Rr      Stera Individ and Preden.      No + K × 1, No + T      None      None        ST      X, Rr      Stera Individ and Preden.      N + K × 1, No + T      None      None        ST      X, Rr      Stera Individ and Preden.      N + K × 1, No + T      None      None        ST      X, Rr      Stera Individ and Preden.      N + K × 1, No + T      None      None        ST      X, Rr      Stera Individ and Preden.      N + e, N + N + 1      None      None        ST      X, Rr      Stera Individ and Preden.      N + e, N + 1      None      None        ST  |                  | Rd, Y+       |   | $Rd \leftarrow (Y), Y \leftarrow Y + 1$  | None         | 2       |
| LD      βd.2      Land individ modelship      βd - (D)      None      None        LD      βd.2      Land individ and Peshbe.      Z - 2.1. βd - (D)      None      None        LD      βd.2      Land individ and Peshbe.      Z - 2.1. βd - (D)      None      None        LDS      βd.4      Land individ and Peshbe.      A - (A)      None      None        LDS      Kd.K      Land individ and Peshbe.      A - (A)      None      None        ST      X.R      Store individ and Peshbe.      (A) - (R)      None      None        ST      X.R      Store individ and Peshbe.      X - X / (X) - (R)      None      None        ST      Y.R      Store individ and Peshbe.      (Y) - (R) - Y + 1.1      None      None        ST      Y.R      Store individ and Peshbe.      (Y) - (R) - Y + 1.1      None      None        ST      Y.R      Store individ and Peshbe.      (Y) - (R) - Y + 1.1      None      None        ST      Z.R      Store individ and Peshbe.      (Y) - (R) - Y + 1.1      None      None        ST      Z.R      Store in  | LD               | Rd, - Y      | Load Indirect and Pre-Dec.                                  | $Y \leftarrow Y - 1, Rd \leftarrow (Y)$  | None         | 2       |
| LDRd.2.*Load Indiced and Pen-Ben.Rd $-(D_1 L^2 - L^2 - 1)$ NomeNomeLD0Rd.2.*Load Indiced with DeglocomentRd $-(D_1 - Q_1)$ NomeNomeLD0Rd.2.*Load Direct from SRMRd $-(D_1 - Q_1)$ NomeNomeSTX.4.Stepe Indirect(D_1 - RT, X + 1)NomeNomeNomeSTX.4.Stepe Indirect(D_1 - RT, X + 1)NomeNomeNomeSTY.4.Stepe Indirect and Pen-Box.X + X + 1, (Q) - RTNomeNomeSTY.4.Stepe Indirect and Pen-Box.Y + Y + 1, (Y) - RTNomeNomeSTY.4.Stepe Indirect and Pen-Box.Y + Y + 1, (Y) - RTNomeNomeSTY.4.Stepe Indirect and Pen-Box.Y + Y + 1, (Y) - RTNomeNomeSTY.4.Stepe Indirect and Pen-Box.Y + Y + 1, (Y) - RTNomeNomeSTZ.4.Stepe Indirect and Pen-Box.Z + Z + 1NomeNomeSTZ.4.Stepe Indirect and Pen-Box.Z + Z + 1, (Q - RTNomeSTZ.4.Stepe Indirect and Pen-Box.Z + Z + 1, (Q - RTNomeSTZ.4.Stepe Indirect and Pen-Box.Z + Z + 1, (Q - RTNomeSTZ.4.<   | LDD              | Rd,Y+q       | Load Indirect with Displacement                             | $Rd \gets (Y + q)$   | None         | 2       |
| LDRd.2Load indiced and Pro-Boc.Z-2-1, Rd - C)NoneNoneLDSRd.4, Load Direct Mon SPAMRd - (-q)NoneNoneNoneLDSRd.4, Load Direct Mon SPAMRd - (-q)NoneNoneNoneSTX, FirStore Indirect and Pro-Boc.(X) - Fir, X - X - 1.0NoneNoneSTX, RrStore Indirect and Pro-Boc.X + X, X (X) - firNoneNoneNoneSTY, FirStore Indirect and Pro-Boc.X + X, 1 (X) - firNoneNoneNoneSTY, FirStore Indirect and Pro-Boc.Y + Y, Y, Y + 1NoneNoneNoneSTY, FirStore Indirect and Pro-Boc.(Y + Y, Y, Y) - firNoneNoneNoneSTY, FirStore Indirect and Pro-Boc.(Y + Y, Y, Y) - firNoneNoneNoneSTZ, FirStore Indirect and Pool:(Z) - FirNoneNoneNoneST   | LD               | Rd, Z        | Load Indirect   | $Rd \gets (Z)$   | None         | 2       |
| LDDHd $Z_1$ qLoad Indices with DisplacementHd $-\sqrt{2}$ , q)NoneHdSTX, FrStere Indiced and Post ho. $(X) - FrMoreNoneNoneSTX, RStere Indiced and Post ho.(X) - FrMoreNoneNoneSTX, RStere Indiced and Post ho.(X) - FrMoreNoneNoneSTY, RStere Indiced and Post ho.(Y) - FrNoneNoneNoneNoneSTY, RStere Indiced and Post ho.(Y) - FrY + Y + 11None$   | LD               | Rd, Z+       | Load Indirect and Post-Inc.                                 | $Rd \gets (Z),  Z \gets Z{+}1$   | None         | 2       |
| LDSRd, rLoad Degition SRAMRd, r, r)NoneNoneSTX, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, X \leftarrow X + 1$ NoneNoneNoneSTX, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, X \leftarrow X + 1$ NoneNoneNoneSTY, RrStore Indired and Polit Inc. $(1) \leftarrow Pr, X \leftarrow X + 1$ NoneNoneNoneSTY, RrStore Indired and Polit Inc. $(1) \leftarrow Pr, X \leftarrow Y + 1$ NoneNoneNoneSTY, RrStore Indired and Polit Inc. $(1) \leftarrow Pr, X \leftarrow Y + 1$ NoneNoneNoneSTY, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, X \leftarrow Y + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ NoneNoneNoneSTZ, RrStore Indired and Polit Inc. $(2) \leftarrow Pr, Z \leftarrow 1 + 1$ None <t< td=""><td>LD</td><td>Rd, -Z</td><td>Load Indirect and Pre-Dec.</td><td><math>Z \leftarrow Z - 1, Rd \leftarrow (Z)</math></td><td>None</td><td>2</td></t<>   | LD               | Rd, -Z       | Load Indirect and Pre-Dec.                                  | $Z \leftarrow Z - 1, Rd \leftarrow (Z)$  | None         | 2       |
| STX, R'Stree Induced and Peat-Inc. $(\chi_1 - F_1 \times - X + 1$ NoneNoneSTX, RStree Induced and Peat-Doc. $(\chi_1 - F_1 \times - X + 1, (\chi) - R^1$ NoneISTY, R'Stree Induced and Peat-Doc. $(\chi_1 - F_1 \times - X + 1, (\chi) - R^1$ NoneISTY, R'Stree Induced and Peat-Doc. $(Y_1 - F_1 \times - Y + 1)$ NoneISTY, R'Stree Induced and Peat-Doc. $(Y_1 - F_1 \times - Y + 1)$ NoneISTZ, R'Stree Induced and Peat-Doc. $(Y_1 - F_1 \times - Y + 1)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - F_1)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - Z + 1)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - Z + 1)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTZ, R'Store Induced and Peat-Doc. $(Z_1 - F_1 - R)$ NoneISTStore Induced and Peat-Doc.Store Induc   | LDD              | Rd, Z+q      | Load Indirect with Displacement                             | $Rd \leftarrow (Z + q)$  | None         | 2       |
| ST      X, Rr      Store Indirect and Pact-hac.      (λ) – Rr      None      I        ST      X, Rr      Store Indirect and Pact-hac.      X – X-1, (λ) – Rr      None      I        ST      Y, Rr      Store Indirect and Pact-hac.      Y – Rr      None      I        ST      Y, Rr      Store Indirect and Pact-hac.      Y – Rr      None      I        ST      Y, Rr      Store Indirect and Pact-hac.      Y + Pr      None      I        ST      Y, Rr      Store Indirect and Pact-hac.      Y + 2, 1, (λ) – Rr      None      I        ST      Y, Rr      Store Indirect and Pact-hac.      (2) – Rr, Z - 2, 1      None      I        ST      Z, Rr      Store Indirect and Pact-hac.      (2) – Rr      None      I        ST      Z, Rr      Store Indirect and Pact-hac.      (2) – Rr      None      I        ST      Z, Rr      Store Indirect and Pact-hac.      (2) – Rr      None      I        ST      Z, Rr      Store Indirect and Pact-hac.      (2) – Rr      None      I        ST      Z, Rr      Store Indirect and Pact-hac. <td>LDS</td> <td>Rd, k</td> <td>Load Direct from SRAM</td> <td><math>Rd \leftarrow (k)</math></td> <td>None</td> <td>2</td>   | LDS              | Rd, k        | Load Direct from SRAM                                       | $Rd \leftarrow (k)$  | None         | 2       |
| ST      X, fr      Stere indirect and Pre-Doc.      X - X + 1, (X) - Br      None      None        ST      Y, R      Stere indirect and Pre-Doc.      X - X + 1, (X) - Br      None      None        ST      Y, R      Stere indirect and Pre-Inc.      (Y) - R, Y - Y + 1, (Y) - R      None      None        ST      Y, R      Stere indirect and Pre-Inc.      (Y - Y + Y + 1, (Y) - R'      None      None        ST      Y, R      Stere indirect and Pre-Doc.      (Y - Y + Y + 1, (Y) - R'      None      None        ST      Y, R      Stere indirect and Pre-Doc.      (Z - R'      None      None        ST      Z, R'      Stere indirect and Pre-Doc.      (Z - Q + R'      None      None        ST      Z, R'      Stere indirect and Pre-Doc.      (Z - Q + R'      None      None        ST      Z, R'      Stere indirect and Pre-Doc.      (Z - Q + R'      None      None        ST      X, R'      Stere indirect and Pre-Doc.      (Z - Q + R'      None      None        ST      X, R'      Stere indirect and Pre-Doc.      (Z - Q + R'      None      None   | ST               |              |   |  |              | 2       |
| ST  ····································   | ST               |              | Store Indirect and Post-Inc.                                |  | None         | 2       |
| ST      Y.r. for      Store indirect and Pet-Inc.      (Y) - Rr. Y - Y + 1.      Nome      Nome        ST      Y.r. for      Store indirect and Pet-Inc.      (Y + q) - Rr.      Nome      Nome        STD      Y.r. for      Store indirect and Pet-Doc.      (Y + q) - Rr.      Nome      Nome        STD      Z. Rr      Store indirect and Pet-Doc.      (Z - Fr.      Nome      Nome        ST      Z., for      Store indirect and Pet-Doc.      (Z - q) - Rr.      Nome      Nome        STD      Z-, q, for      Store indirect and Pet-Doc.      (Z - q) - Rr.      Nome      Nome        STS      K, R      Store indirect with Displacement      (Z - q) - Rr.      Nome      Nome        LPM      R. d. Z      Load Program Memory      Rd - (Z)      Nome      Nome        SPM      -      Store Program Memory      Rd - (Z)      Nome      Nome        SPM      -      Store Program Memory      Rd - (Z)      Nome      Nome        SPM      -      Store Program Memory      Rd - (Z)      Nome      Nome        SPM      Rd Z      Nome <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td>   |                  |              |   |  |              | 2       |
| ST      Y, βr      Store indired and Pool-Re.      Y + Y + 1, (Y) - βr      None      None        STO      Y, dg, Rr      Store indired and Pool-Re.      Y + Y + 1, (Y) - βr      None      None        ST      Z, hr      Store indired and Pool-Re.      (Z) - βr, Z + Z + 1.      None      None        ST      Z, hr      Store indired and Pool-Re.      (Z + a) - Fr      None      None        ST      Z, hr      Store indired and Pool-Re.      (Z + a) - Fr      None      None        ST      Z, fr      Store indired and Pool-Re.      (Z + a) - Fr      None      None        ST      Z, fr      Store indired and Pool-Re.      (Z + a) - Fr      None      None        ST      Z, fr      Store indired and Pool-Re.      (Z + a) - Fr      None      None        ST      Z, fr      Store indired and Pool-Re.      (Z + a) - Fr      None      None        ST      Load Program Memory      Rd + [Z      None      None      None        PM      Rd      Load Program Memory      [Z - Fr]: A      None      None        Store indine on Store indine A   |                  |              |   |  |              | 2       |
| ST $Y, P, R'$ Store Indiced and Pro-Dec. $Y \leftarrow Y, (y) \leftarrow Pr$ NoneNoneSTD $Y \leftarrow B, R'$ Store Indiced with Displacement $(2) \leftarrow Pr$ NoneNoneSTZ, RrStore Indiced and Pools. $(2) \leftarrow Pr$ , $Z \leftarrow Z + 1$ NoneNoneSTZ, RrStore Indiced and Pools. $Z \leftarrow Z + 1, (Z) \leftarrow Pr$ NoneNoneSTDZ, RrStore Indicet and Pools. $Z \leftarrow Z + 1, (Z) \leftarrow Pr$ NoneNoneSTSK, RrStore Indicet and Pools. $Z \leftarrow Z + 1, (Z) \leftarrow Pr$ NoneNoneUPMLad Program MemoryB( $-LZ$ )NoneNoneNoneLPMRd, Z.Load Program MemoryB( $-LZ$ )NoneNoneSPMStore Program MemoryB( $-LZ$ )NoneNoneNoneSPMStore Program MemoryB( $-LZ$ )NoneNoneNonePMRd, PIn PortRd $-P$ NoneNoneNoneOUTP, RrOut PortPo - RrNoneNoneNonePDRdPog Register from StackRd $-P$ NoneNoneNoneDITAN DITTSTETNUSTUTSUTUSPA)NoneNoneNoneNoneSIRP,bStart Init O RegisterUO(P,b) $-1$ NoneNoneSIRP,bClass Bith In O RegisterUO(P,b) $-0$ NoneNoneSIRP,bClass Bith In O RegisterUO(P,b) $-0$ NoneNoneSIRP,bClass Bith In O RegisterRd(n) $-Rd(n), Rd(n), -Rd(n)$  |                  |              |   |  |              | 2       |
| STD      Yαβr      Size Indirect with Displacement      (Y + α) - far      None      None        ST      Z, Rr      Size Indirect and Postinc.      (Z) + Rr, Z + Z + 1      None      None        ST      Z, Rr      Size Indirect and Postinc.      (Z) + Rr, Z + Z + 1      None      None        STD      Z, Rr      Size Indirect with Displacement      (Z + a) - Rr      None      None        STD      Lag Morean Memory      R0 + (Z) - Rr      None      None      None        UPM      A. Load Program Memory      R0 + (Z) - Z + 31      None      None      None        LPM      Rd, Z      Load Program Memory      R0 + (Z) - Z + 21      None      None        SM      Size Program Memory      R0 + (Z) - C + Rr      None      None      None        NM      Rd, P      In Port      Rd + (Z) - Rr      None      None      None        NM      Rd, P      None      Size Program Memory      Rd + C + Rr      None      None        NUT      P, Rr      Out Port      Rd + STACK      None      None        NUT      P, Rr </td <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td>  |                  |              |   |  |              | 2       |
| STZ.hStore indirect $(2) + fr. Z - fr.$ NoneSTZ.hStore indirect and Pre-Dec. $Z + Z - 1$ , $(Z) + Fr. Z - Z + 1$ NoneSTZ.RStore indirect and Pre-Dec. $Z + Z - 1$ , $(Z) - Fr.$ NoneSTDZre, RrStore indirect and Pre-Dec. $Z + Z - 1$ , $(Z) - Fr.$ NoneSTDZre, RrStore indirect and Pre-Dec. $Z + Z - 1$ , $(Z) - Fr.$ NoneSTSk. RrStore indirect and Pre-Dec. $Z + Z - 1$ , $(Z) - Fr.$ NoneLPMLoad Program MemoryR0 - $(Z)$ NoneR0LPMRd. ZLoad Program MemoryRd + $(Z)$ NoneLPMRd, ZLoad Program Memory and Post-IncRd + $(Z)$ NoneSMStore Program Memory and Post-IncRd + $(Z)$ NoneRdSMNo.PStore Program Memory $(Z) + E1:80$ NoneRdOUTP, RrOut PortRd - PNoneRdPUSHRdPog Ragister from StackRd + PNoneRdPDPRdPog Ragister from StackRd + STACKNoneRdBIT AND BT-TEST WISTHUCTIONSSet Bit in UC RagisterUO(P,b) - 1NoneNoneCBLP.bSet Bit in UC RagisterUO(P,b) - 0Z.C.N.VRdCBLRdLogical Shift RightRd(n+1) - Rd(n+1), Rd(n) - 0Z.C.N.VLS.N.VCBLRdRdate Right Through CarryRd(n+1) - Rd(n+1), Rd(n), C-Rd(n)Z.C.N.VLS.N.VRORRdArithm  |                  |              |   |  |              | 2       |
| STZ, RrStore indirect and Pre-Dec. $(2) - Pr 2 - 1. (2) - Fr.NoneNoneSTZ, RrStore indirect with Displacement(2 + 2) - Fr.NoneISTDZ+q. RrStore indirect with Displacement(2 + 1) - Fr.NoneISTSK. RrStore Direct to SHAM(0) - Fr.NoneISTMLoad Program MemoryR0 - (2)NoneILPMFd.ZLoad Program MemoryRd - (2) - 2 - 1.1NoneILPMRd.ZLoad Program Memory and Post-IncRd - (2) - 2 - 2.1NoneISPMRd ZLoad Program Memory(2) - 61:R0NoneIOUTP. RrOut FortRd - PNoneIPUSHRrOut FortRd - PNoneIPDFRdPort FortStack FortNoneIPDRRdPort FortNoneIIPDRRdPort FortNoneIIPDRRdPort FortNoneIIStackStack FortNoneIIStackStack FortNoneIIStackRdLogical Shift RightRd(n) + Ad(n1), Rd(n) - 0Z, C, N, VLSLRdLogical Shift RightRd(n) + Ad(n1), Rd(n) - 0Z, C, N, VIRDRRdLogical Shift RightRd(n) + Rd(n1), Rd(n) - 0Z, C, N, VIRDRRdLogical Shift RightRd(n) + Rd(n1), Rd(n) - 0$   |                  |              |   |  |              | 2       |
| STZ. RrStore Indirect and Pre-Dec.Z 2. 1, (2) $\leftarrow$ PrNoneSTDZ. 4, (PrStore Indirect with Displacement(Z + q) $\leftarrow$ PrNoneNoneSTSK. RrStore Direct to SRAM(K) $\leftarrow$ PrNoneNoneLPMMLoad Program MemoryR0 $\leftarrow$ (Z)NoneNoneLPMRd. ZLoad Program Memory and Posl-IncRd $\leftarrow$ (Z), Z $\leftarrow$ Z-1NoneNoneSPMStore Program Memory and Posl-IncRd $\leftarrow$ (Z), Z $\leftarrow$ Z-1NoneNoneSPMStore Program Memory and Posl-IncRd $\leftarrow$ (Z), Z $\leftarrow$ Z-1NoneNoneOUTP, RrOut PortRd $\leftarrow$ PNoneNoneOUTP, RrOut PortRd $\leftarrow$ PNoneNonePDFRdPop Register on StackSTACK $\leftarrow$ PrNonePDFRdPop Register from StackRd $\leftarrow$ STACKNoneSIT AND BIT-EST-WISTHUCTONSStat Bit in 10 Register10(P, B) $\leftarrow$ 1NoneSITRdLogical Shift RightRd(n-1) $\leftarrow$ Rd(n), Pd(n) $\leftarrow$ 0Z.C.N.VSIRRdLogical Shift RightRd(n) $\leftarrow$ Rd(n-1) $\leftarrow$ Rd(n), Pd(n), Pd(n) $\leftarrow$ 0Z.C.N.VLSLRdRdLogical Shift RightRd(n) $\leftarrow$ Rd(n, A, Rd, Z), Pd(n) $\leftarrow$ 0Z.C.N.VKRARdRdat Right Through CarryRd(n) $\leftarrow$ Rd(n, A, Rd, Z), Pd(n) $\leftarrow$ 0Z.C.N.VKRARdRdat Right Through CarryRd(n) $\leftarrow$ Rd(n, A, Rd, Z), Pd(n) $\leftarrow$ 0Z.C.N.VASRRdAntimetic Shift RightRd(n) $\leftarrow$ Rd(n, A   |                  |              |   |  |              |         |
| STDZ-q, RrStore Induced with Displacement $(Z, Q) \leftarrow Pr$ NoneNoneSTSi, K, RrStore Direct to SRAM(k) $\leftarrow$ RrNoneNoneLPMLoad Program MemoryR0 $\leftarrow$ (Z)NoneNoneLPMR0, ZLoad Program MemoryR0 $\leftarrow$ (Z) $\leftarrow$ 2-2-1NoneNoneLPMR0, ZLoad Program Memory(Z) $\leftarrow$ R1:R0NoneNoneSPM-Store Program Memory(Z) $\leftarrow$ R1:R0NoneNoneINRd, PIn PortRd $\leftarrow$ PNoneNoneOUTP, RrOut PortP $\leftarrow$ RrNoneNonePUSHRrPush Register on StackSTACKNoneNonePDPRdPog Rogister from StackSTACKNoneNoneEIT AND BIT-TEST INSTRUCTIONS-NoneNoneNoneNoneSBIP.bClear Bit In I/O RegisterI/O(P,b) $\leftarrow$ 1NoneNoneLSLRdLogical Shift EiftRd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n)Z.C.N.VNoneLSLRdLogical Shift EiftRd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n)Z.C.N.VNoneROLRdRd Rd Right Through CarryRd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n)Z.C.N.VNoneSMAPRdAnthmedic Shift RightRd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n)C.N.VSMAPRdAnthmedic Shift RightRd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n)Z.C.N.VSMAPRdAnthmedic Shift RightRd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n) $\leftarrow$ Rd(n)Z.C.N   |                  |              |   |  |              | 2       |
| STSk, $\mathbf{R}$ Store Direct to SRAM(h) - $\mathbf{R}$ NoneNoneLPMLLoad Program MemoryR0 (2)NoneNoneLPMRd, ZLoad Program Memory and Post-IncRd (2, Z - Z + 1)NoneNoneSTMStore Program Memory and Post-IncRd (2, Z - Z + 1)NoneNoneNoneNRd, PIn PortRd PNoneNoneNoneNoneOUTP, RrOut PortP - RrNoneNoneNoneNonePUSHRrPog Register rom StackSTACK - RrNoneNoneNonePDRdPog Register rom StackSTACK - RrNoneNoneNoneDIT AUD ETT-STNSTNUCTOSTSet Blin IO RegisterIO(P) + -1NoneNoneNoneCBIP.bClear Blin IO RegisterIO(P) + -0NoneNoneNoneLSLRdLogical Shift LdftRd(n-1) - Rd(n), Rd(0) - 0Z.C.N.VISLSLRdLogical Shift EightRd(n) - Rd(n-1), Rd(7) - 0Z.C.N.VISROLRdRotate Right Through CarryRd(n) - Rd(n-1), Rd(7) - 0Z.C.N.VISSWAPRdArbiter Blight RightRd(n) - Rd(n-1), Rd(7) - 0Z.C.N.VISSWAPRdArbiter Shift RightRd(n) - Rd(n-1), Rd(7) - 0Z.C.N.VISSWAPRdArbiter Shift RightRd(n) - Rd(n-1), Rd(7) - 0Z.C.N.VISSWAPRdArbiter Shift RightRd(n) - Rd(n-1), Rd(7) - 0  |                  |              |   |  |              | 2       |
| LPMnoLoad Program Memory $R0 - (2)$ NoneILPMRd, ZLoad Program Memory and Post-Inco $Rd - (2)$ NoneNoneILPMRd, ZLoad Program Memory and Post-Inco $Rd - (2)$ , Z - Z - Z - Z - Z - Z - Z - Z - Z - Z  |                  |              | ·   |  |              | 2       |
| LPMRd, ZLoad Program Memory and Post-IncRd ← (2), Z ← Z1NoneILPMRd, ZStore Program Memory and Post-IncRd ← (2), Z ← Z1NoneNoneISPMStore Program Memory and Post-IncRd ← (2), Z ← Z1NoneNoneIINRd, PIn PortRd ← PNoneIIOUTP, RrOut PortRd ← PNoneIPDFRdPoo Register from StackSTACK ← RrNoneIPOPRdPoo Register from StackSTACK ← RrNoneISBI TAVD SITTEST/FUTSet Bit in I/O RegisterI/O(P,b) ← 1NoneICBIP,bSet Bit in I/O RegisterI/O(P,b) ← 1NoneICBIRdLogical Shift HaptRd(n) ← Rd(n, Hapt), Rd(n) ← 0Z,C,N/IRANRdLogical Shift RightRd(n) ← Rd(n, Hapt), Rd(n) ← 0Z,C,N/IRORRdRotate Right Through CarryRd(n) ← Rd(n+1), C-Rd(n)Z,C,N/IRARRdArithmeic Shift RightRd(n) ← Rd(n+1), C-Rd(n)Z,C,N/ISWAPRdSwap NiblesRd(n) ← Rd(n+1), C-Rd(n)Z,C,N/<  |                  | k, Rr        |   |  |              | 2       |
| LPMRd, Z+Lad Program Memory and Post-IncRd $\leftarrow$ (2, $2 \leftarrow$ Z-1NoneIncSPMStore Program Memory(2) $\leftarrow$ R1:R0NoneISPMRd $\leftarrow$ PNoneRd $\leftarrow$ PNoneIOUTP, RrOut PortRd $\leftarrow$ PNoneIPUSHRrPush Register on StackP $\leftarrow$ RrNoneIPOPRdPoo Register from StackRd $\leftarrow$ STACK $\leftarrow$ RrNoneIBIT AND BIT-TEST INSTRUCTIONSSELP.b.Clear Bit in I/O RegisterI/O(P,b) $\leftarrow$ 0NoneICBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow$ 0NoneILSRRdLogical Shift RightRd(n) $\leftarrow$ Rd(n+1), $\leftarrow$ Rd(n), C, C, Rd(n) $\leftarrow$ Rd(r)Z, C, N,VIROLRdRotate Idit Through CarryRd(n) $\leftarrow$ Rd(n+1), C-Rd(n)Z, C, N,VIRORRdAntate Right Through CarryRd(n) $\leftarrow$ Rd(n-1), Rd(n), C, Rd(n)Z, C, N,VISWAPRdSwap NibbiesRd(s, 0) $\leftarrow$ Rd(r, 4), Rd(3, 0)NoneISWAPRdSwap NibbiesRd(s, 0) $\leftarrow$ Rd(r, 4), Rd(3, 0)NoneISWAPRdSwap NibbiesRd(s, 0) $\leftarrow$ Rd(r, 4), Rd(3, 0)NoneISECSFlag CelarSEE(G) $\leftarrow$ 0SEE(G)ISECSet CarryC $\leftarrow$ 0CCICLClear CarryC $\leftarrow$ 0NIISECSet Register FlagN $\leftarrow$ 0NIISEC<  |                  |              |   |  |              | 3       |
| SPMStore Program Memory $(Z) \leftarrow R1:R0$ NoneNoneINRd, PIn PortRd $\leftarrow$ PNoneNoneOUTP, RrOut PortP $\leftarrow$ RrNonePPUSHRrPush Register on StackSTACK $\leftarrow$ RrNonePPOPRdPop Register from StackSTACK $\leftarrow$ RrNonePBIT AND BIT-TEST INSTRUCTIONSRd $\leftarrow$ STACKNonePSBIP.bSet Bit in UO Register $UO(P,b) \leftarrow 0$ NoneRCBIP.bClear Bit in UO Register $UO(P,b) \leftarrow 0$ NoneRLSLRdLogical Shift RightRd(n+1) $\leftarrow$ Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,VIROLRdRolate Left Through CarryRd(0) $\leftarrow$ Rd(n+1), Rd(7) $\leftarrow 0$ Z,C,N,VIRORRdRd atta Edit Through CarryRd(0) $\leftarrow$ Rd(n+1), nol,0), C=Rd(7)Z,C,N,VIRORRdAntimetic Shift RightRd(1) $\leftarrow$ Rd(n+1), no6Z,C,N,VISWAPRdAntimetic Shift RightRd(2) $\leftarrow$ Rd(n+1), no6Z,C,N,VIRGSet Specific ASREG(s) $\leftarrow$ 1SREG(s)IBSTsFlag SetSREG(s)SREG(s)IBSTsFlag SetSREG(s)ISREG(s)IBCLRsFlag SetSREG(s)ISREG(s)IBCLSet CarryC $\leftarrow$ 0C $\leftarrow$ 0CICLSet CarryC $\leftarrow$ 0C $\leftarrow$ 0IISECSet Segue  |                  | -            | • •   |  | None         | 3       |
| NRd, PIn PortRd $\leftarrow$ PNoneOUTP, RrOut PortP $\leftarrow$ RrNonePPUSHRrPush Registor on StackSTACK $\leftarrow$ RrNonePOPRdPop Register from StackRd $\leftarrow$ STACKNoneBIT AND BIT-TEST INSTRUCTIONSStatistic RegisterI/O(P,b) $\leftarrow$ 1NoneCBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow$ 1NoneCBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow$ 1NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow$ 1NoneLSLRdLogical Shift IghtRd(n+1) $\leftarrow$ Rd(n), Rd(0) $\leftarrow$ 0Z,C,N,VLSRRdLogical Shift IghtRd(n) $\leftarrow$ Rd(n+1, Rd(r) $\leftarrow$ 0Z,C,N,VRORRdRotate Hight Through CarryRd(D) $\leftarrow$ Rd(n+1) $\leftarrow$ Rd(n), C,C,Rd(0)Z,C,N,VRORRdRotate Right Through CarryRd(D) $\leftarrow$ Rd(n+1) $\leftarrow$ Rd(n), C,C,Rd(0)Z,C,N,VSWAPRdSwap NibblesRd(3, O) $\leftarrow$ Rd(r,1) $\rightarrow$ Rd(3, O)NoneBSETsFiag SetSREG(s) $\leftarrow$ 1SREG(s)BCLRsFiag ClearSREG(s) $\leftarrow$ 1SREG(s)BCLsFilag ClearSREG(s) $\leftarrow$ 1NoneSECSet CarryC $\leftarrow$ 1C1CLSet CarryC $\leftarrow$ 0C2SENAd, bBit Ioad from To RegisterN $\leftarrow$ 1N $\leftarrow$ 1SELSet Zeor FlagX $\leftarrow$ 1C2SELClear CarryC $\leftarrow$ 0C </td <td>LPM</td> <td>Rd, Z+</td> <td>Load Program Memory and Post-Inc</td> <td><math>Rd \leftarrow (Z), Z \leftarrow Z+1</math></td> <td>None</td> <td>3</td>   | LPM              | Rd, Z+       | Load Program Memory and Post-Inc                            | $Rd \leftarrow (Z), Z \leftarrow Z+1$  | None         | 3       |
| OUTP, RrOut PortP $\leftarrow$ ArNonePPUSHRrPush Register on StackSTACK $\leftarrow$ RrNonePPOPRdPop Register from StackRd $\leftarrow$ STACKNonePBITPubSet Bit in VO RegisterVO(P, b) $\leftarrow$ 1NoneCCBIP,bSet Bit in VO RegisterVO(P, b) $\leftarrow$ 0Z,C,N,VLCBIP,bCear Bit in VO RegisterVO(P, b) $\leftarrow$ 0Z,C,N,VLLSLRdLogical Shift RightRd(n+1) $\leftarrow$ Rd(n), Rd(0) $\leftarrow$ 0Z,C,N,VLRORRdRotate Right Through CarryRd(7) $\leftarrow$ C,Rd(n+1), Rd(n), C,Rd(7)Z,C,N,VLRORRdRotate Right Through CarryRd(7) $\leftarrow$ C,Rd(n+1), C,Rd(0)Z,C,N,VLSWAPRdSwap NibblesRd(3, O) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VLSWAPRdSwap NibblesSREG(s) $\leftarrow$ 1SREG(s)LSURAPRdSwap NibblesSREG(s) $\leftarrow$ 1SREG(s)LBSTsF lag SetSREG(s) $\leftarrow$ 1SREG(s)LBCLRsF lag Set Nor Register to TT $\leftarrow$ (h)TLBLDRd, bBit Store from Register to TT $\leftarrow$ (h)TNoneSECISet CarryC $\leftarrow$ 1CCCSECISet CarryC $\leftarrow$ 0CSSSECISet CarryC $\leftarrow$ 0CCCSECIClear Yaor ClagZ $\leftarrow$ 0CC <td>SPM</td> <td></td> <td>Store Program Memory</td> <td>(Z) ← R1:R0</td> <td>None</td> <td>-</td>  | SPM              |              | Store Program Memory  | (Z) ← R1:R0  | None         | -       |
| PUSHRrPush Register on StackSTACK $\leftarrow$ RrNonePOPRdPop Register from StackRd $\leftarrow$ STACKNonePOFRdPop Register from StackRd $\leftarrow$ STACKNoneBIT AND BIT-TEST/UCTIONSSBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow$ 1NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow$ 0NoneNoneCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow$ 0NoneNoneLSLRdLogical Shift RightRd(n) $\leftarrow$ Rd(n,1), Rd(n) $\leftarrow$ 0Z,C,N,VI/OROLRdRotate Right Through CarryRd(D) $\leftarrow$ C,Rd(n+1) $\leftarrow$ Rd(n,1), C=-Rd(T, 2,C,N,VI/ORORRdRotate Right Through CarryRd(D) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VI/OSWAPRdSwap NibblesRd(3, $\bigcirc$ C-Rd(T, 4), Rd(T, | IN               | Rd, P        | In Port   | $Rd \gets P$   | None         | 1       |
| POPRdPop Register from StackRd $\leftarrow$ STACKNoneBIT AND BIT-TEST INSTRUCTONSSBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ NoneI/OCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ NoneI/OCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ NoneI/OLSLRdLogical Shift LeftRd(n+1) $\leftarrow$ Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,VI/OLSRRdLogical Shift LeftRd(n) $\leftarrow$ Rd(n+1), Rd(7) $\leftarrow 0$ Z,C,N,VI/OLSRRdRotate Left Through CarryRd(n) $\leftarrow$ Rd(n+1), C,C,Rd(n), C,C,Rd(7)Z,C,N,VI/ORORRdRotate Left Through CarryRd(n) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VI/OSWAPRdSwap NibblesRd(a) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VI/OSWAPRdSwap NibblesRd(a) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VI/OBSTsFlag SetSREG(s) $\leftarrow 0$ SREG(s)I/OBCLRsFlag SetSREG(s) $\leftarrow 0$ SREG(s)I/OBCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)I/OBLDRd, bBit Istore from Register to TT $\leftarrow$ Rr(b)TI/OSECISet CarryC $\leftarrow 1$ C $\leftarrow 0$ I/OSECISet CarryC $\leftarrow 1$ NoneI/OSELIClear CarryC $\leftarrow 0$ CI/OSEZISet Zero FlagZ $\leftarrow 1$ Z $\leftarrow 0$ I/OSEZ<  | OUT              | P, Rr        | Out Port  | $P \leftarrow Rr$  | None         | 1       |
| BIT AND BIT-TEST INSTRUCTIONSSBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ NoneLSLRdLogical Shift LoftRd(n+1), Rd(0) $\leftarrow 0$ Z,C,N,VLSRRdLogical Shift RightRd(n) $\leftarrow Rd(n+1), Rd(7) \leftarrow 0$ Z,C,N,VROLRdRotate Left Through CarryRd(7) $\leftarrow C,Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$ Z,C,N,VRORRdRotate Right Through CarryRd(7) $\leftarrow C,Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z,C,N,VASRRdArithmetic Shift RightRd(7) $\leftarrow C,Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z,C,N,VSWAPRdSwap NibblesRd(3, a) $\leftarrow Rd(7, 4), eRd(3, 0)$ NoneBSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)BCLRsFlag SetSREG(s) $\leftarrow 0$ SREG(s)EBSTRr, bBit Store from Register to TT $\leftarrow Rr(b)$ TIBLDRd, bBit Idad from T to RegisterRd(b) $\leftarrow T$ NoneSSECSet CarryC $\leftarrow 1$ CCICLNClear Agative FlagN $\leftarrow 0$ NSSEZSet Zaro FlagZ $\leftarrow 0$ ZZICL2Clear Agative FlagC $\leftarrow 1$ IICL3Clear Agative FlagX $\leftarrow 0$ NSSEZSet Zaro FlagZ $\leftarrow 0$ ZICL4Global Interrupt EnableI $\leftarrow 0$ IICL5Global Interrupt EnableI $\leftarrow 0$ I <td>PUSH</td> <td>Rr</td> <td>Push Register on Stack</td> <td><math>STACK \leftarrow Rr</math></td> <td>None</td> <td>2</td>   | PUSH             | Rr           | Push Register on Stack                                      | $STACK \leftarrow Rr$  | None         | 2       |
| SBIP,bSet Bit in I/O Register $VO(P,b) \leftarrow 1$ NoneNoneCBIP,bClear Bit in I/O Register $IO(P,b) \leftarrow 0$ NoneNoneLSLRdLogical Shift Left $Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$ $Z,C,N,V$ ZLSRRdLogical Shift Right $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$ $Z,C,N,V$ ZROLRdRotate Left Through Carry $Rd(0) \leftarrow C,Rd(n+1) \leftarrow Rd(n), C-Rd(0)$ $Z,C,N,V$ ZRORRdAnthenetic Shift Right $Rd(7) \leftarrow C,Rd(n+1), C-Rd(0)$ $Z,C,N,V$ ZSWAPRdAnthenetic Shift Right $Rd(7) \leftarrow C,Rd(n+1), C-Rd(0)$ $Z,C,N,V$ ZSWAPRdSwap Nibbles $Rd(3,0) \leftarrow Rd(n+1), C-Rd(0)$ $Z,C,N,V$ ZSWAPRdSwap Nibbles $Rd(3,0) \leftarrow Rd(n+1), C-Rd(0)$ $Z,C,N,V$ ZSWAPRdSwap Nibbles $Rd(3,0) \leftarrow Rd(n+1), C-Rd(0,0)$ $Z,C,N,V$ ZSWAPRdSwap Nibbles $Rd(0, -T, A), Rd(7, A), C-Rd(0,0)$ $Z,C,N,V$ ZSWAPSwap NibblesSwap Nibbles $Reg(s) \leftarrow 1$ NZSCSwap NibblesSwap Nibbles $Reg(s) \leftarrow 1$ N <t< td=""><td>POP</td><td>Rd</td><td>Pop Register from Stack</td><td><math>Rd \leftarrow STACK</math></td><td>None</td><td>2</td></t<>  | POP              | Rd           | Pop Register from Stack                                     | $Rd \leftarrow STACK$  | None         | 2       |
| CBIP,bClear Bit in VO RegisterVO(P,b) $\leftarrow 0$ NoneNoneLSLRdLogical Shift LeftRd(n+1) $\leftarrow$ Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,VLLSRRdLogical Shift RightRd(n) $\leftarrow$ Rd(n+1), Rd(0) $\leftarrow 0$ Z,C,N,VLROLRdRotate Left Through CarryRd(0) $\leftarrow$ Rd(n+1), Rd(0), $\leftarrow$ Rd(7)Z,C,N,VLRORRdRotate Right Through CarryRd(7) $\leftarrow$ C,Rd(n+1), Rd(0), $\leftarrow$ Rd(n+1), C–Rd(0)Z,C,N,VLASRRdAntimetic Shift RightRd(n) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VLSWAPRdSwap NibblesRd(3.0) $\leftarrow$ Rd(n-1), n=0.6Z,C,N,VLBSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)SREG(s)BCLRsFlag SetSREG(s) $\leftarrow 1$ SREG(s)LBCLRsFlag ClearSREG(s) $\leftarrow 1$ SREG(s)LBLDRd, bBit load from T to Register to TT $\leftarrow R(b)$ TMoneSECLSet CarryC $\leftarrow 0$ CCCCLCIClear CarryC $\leftarrow 0$ NZCSEXISet Zaro FlagZ $\leftarrow 0$ ZIICL2IClear Zero FlagZ $\leftarrow 0$ ZIISEXIGlobal Interrupt EnableI $\leftarrow 0$ IIICL3Global Interrupt EnableI $\leftarrow 0$ IIIISESIGlobal Interrupt DisableS $\leftarrow 0$ SSIII  | BIT AND BIT-TEST | INSTRUCTIONS |   |  |              |         |
| LSLRdLogical Shift LeftRd(n+1) $\leftarrow$ Rd(n), Rd(0) $\leftarrow$ 0Z, C, N, VILSRRdLogical Shift RightRd(n) $\leftarrow$ Rd(n+1), Rd(7) $\leftarrow$ 0Z, C, N, VIROLRdRd tate Left Through CarryRd(0) $\leftarrow$ C, Rd(n+1), $\leftarrow$ Rd(n), $C \leftarrow$ Rd(7)Z, C, N, VIRORRdRotate Right Through CarryRd(7) $\leftarrow$ C, Rd(n+1), $\leftarrow$ Rd(n), $C \leftarrow$ Rd(7)Z, C, N, VIASRRdAnthmetic Shift RightRd(n) $\leftarrow$ Rd(n+1), n=0.6Z, C, N, VISWAPRdSwap NibblesRd(3.0) $\leftarrow$ Rd(7.4), Rd(7.4) $\leftarrow$ Rd(3.0)NoneIBSETsFlag SetSREG(s) $\leftarrow$ 1SREG(s)IBCLRsFlag ClearSREG(s) $\leftarrow$ 0SREG(s)IBLDRd, bBit Store from Register to TT $\leftarrow$ Rr(b)TIBLDRd, bBit load from T to RegisterRd(b) $\leftarrow$ TNoneISECSet CarryC $\leftarrow$ 0CIICLCClear CarryC $\leftarrow$ 0CISEXASet Negative FlagN $\leftarrow$ 0NICLNClear Zero FlagZ $\leftarrow$ 0ZIISEZSet Zero FlagZ $\leftarrow$ 0ZIICLClear Zero FlagZ $\leftarrow$ 0ZIISEZSet Zero FlagS $\leftarrow$ 1SIICLClear Zero FlagS $\leftarrow$ 1SIISEZSet Signed Test FlagS $\leftarrow$ 0SIICL </td <td>SBI</td> <td>P,b</td> <td>Set Bit in I/O Register</td> <td><math>I/O(P,b) \leftarrow 1</math></td> <td>None</td> <td>2</td>   | SBI              | P,b          | Set Bit in I/O Register                                     | $I/O(P,b) \leftarrow 1$  | None         | 2       |
| LSRRdLogical Shift Right $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$ $Z, C, N, V$ ROLRdRotate Left Through Carry $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow C, Rd(7)$ $Z, C, N, V$ RORRdRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ $Z, C, N, V$ ASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n \leftarrow 0.6$ $Z, C, N, V$ $Z, C, N, V$ SWAPRdSwap Nibbles $Rd(3, 0) \leftarrow Rd(7, 4), \leftarrow Rd(3, 0)$ None $Z, C, N, V$ BSTsFlag Set $SREG(s) \leftarrow 1$ $SREG(s)$ $Z, C, N, V$ BCLRsFlag Clear $SREG(s) \leftarrow 1$ $SREG(s)$ $Z, C, N, V$ BLDRd, bBit Store from Register to T $T \leftarrow Rr(b)$ $T$ $T$ BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None $R$ SECSet Carry $C \leftarrow 1$ $C$ $C$ $C$ CLCClear Carry $C \leftarrow 0$ $C$ $C$ SENSet Negative Flag $N \leftarrow 0$ $N$ $D$ SEZSet Set Set Plag $Z \leftarrow 0$ $Z$ $C$ CL2Clear Zero Flag $Z \leftarrow 0$ $Z$ $Z$ SE3Global Interrupt Enable $I \leftarrow 0$ $I$ $I$ SESSet Signed Test Flag $S \leftarrow 0$ $S$ $S$ CL3Clear Signed Test Flag $S \leftarrow 0$ $S$ $S$ SEVSet Signed Test Flag $S \leftarrow 0$ $S$ $S$ SETSet T in SREG $V \leftarrow 0$ $V$ $V$   | CBI              | P,b          | Clear Bit in I/O Register                                   | $I/O(P,b) \leftarrow 0$  | None         | 2       |
| ROLRdRotate Left Through Carry $Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$ Z, C, N, VIRORRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, VIASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ SREG(s)IBLTsFlag ClearSREG(s) < 1   | LSL              | Rd           | Logical Shift Left  | $Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$   | Z,C,N,V      | 1       |
| ROLRdRotate Left Through Carry $Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$ Z, C, N, VIRORRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, VIASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ Z, C, N, VISWAPRdSwap Nibbles $Rd(3.0) \leftarrow Rd(n-1), n=0.6$ SREG(s)IBLTsFlag ClearSREG(s) < 1   | LSR              | Rd           | Logical Shift Right   |  | Z.C.N.V      | 1       |
| RORRdRotate Right Through Carry $Rd(7) \leftarrow C,Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, VASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n=0.6$ Z, C, N, VSWAPRdSwap Nibbles $Rd(3, 0) \leftarrow Rd(7, 4), Rd(7, 4) \leftarrow Rd(3, 0)$ NoneBSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BSTRr, bBit Store from Register to TT $\leftarrow Rr(b)$ TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ NoneSECCSet CarryC $\leftarrow -1$ CCLCClear CarryC $\leftarrow -0$ CCSENClear Negative FlagN $\leftarrow 0$ NNCLNClear Negative FlagN $\leftarrow 0$ ZCCL2Clear Set Zero FlagZ $\leftarrow 0$ ZCCL2Clear Carg FlagZ $\leftarrow 0$ ZCSEIGlobal Interrupt EnableI $\leftarrow 0$ ICCL3Set Signed Test FlagS $\leftarrow 1$ SSSEIGlobal Interrupt DisableS $\leftarrow 0$ SCSEVAction Set Twos Complement OverflowV $\leftarrow 1$ VVSETSet Twos Complement OverflowV $\leftarrow 0$ VV   |                  |              |   |  |              | 1       |
| ASRRdArithmetic Shift RightRd(n) $\leftarrow$ Rd(n+1), n=0.6Z,C,N,VSWAPRdSwap NibblesRd(30) $\leftarrow$ Rd(n-1), n=0.6Z,C,N,VBSTsFlag SetSREG(s) $\leftarrow$ 1SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow$ 0SREG(s)BSTRr, bBit Store from Register to TT $\leftarrow$ Rr(b)TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow$ TNoneSECSet CarryC $\leftarrow$ 1CCCLCClear CarryC $\leftarrow$ 0CCSENSet Negative FlagN $\leftarrow$ 1NCCLNClear Vegative FlagN $\leftarrow$ 0NCSEIGlobal Interrupt EnableI $\leftarrow$ 1ICCLIGlobal Interrupt EnableI $\leftarrow$ 0ISSESSet Signed Test FlagS $\leftarrow$ 0SCCLSSet Signed Test FlagS $\leftarrow$ 0SSCLSSet Signed Test FlagS $\leftarrow$ 0SCCLSSet Signed Test FlagS $\leftarrow$ 0SSCLSClear Xognelment OverflowV $\leftarrow$ 1VCCLVQClear Twos Complement OverflowV $\leftarrow$ 1VVSETSet T in SREGT $\leftarrow$ 1T $\leftarrow$ 1TT  |                  |              |   |  |              | 1       |
| SWAPRdSwap NibblesRd(30) $\leftarrow$ Rd(74), Rd(74) $\leftarrow$ Rd(30)NoneBSETsFlag SetSREG(s) $\leftarrow$ 1SREG(s)SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow$ 0SREG(s)SREG(s)BSTRr, bBit Store from Register to TT $\leftarrow$ Rr(b)TTBLDRd, bBit load from T to RegisterRd(b) $\leftarrow$ TNoneSECSECSet CarryC $\leftarrow$ 1CCCLCClear CarryC $\leftarrow$ 0CCSENSet Negative FlagN $\leftarrow$ 1NCCLNClear Negative FlagN $\leftarrow$ 0NCSEZSet Zero FlagZ $\leftarrow$ 0ZCCLZClear Zero FlagZ $\leftarrow$ 0ZCSEIGlobal Interrupt EnableI $\leftarrow$ 1ICCLIGlobal Interrupt DisableI $\leftarrow$ 0SSSESSet Signed Test FlagS $\leftarrow$ 0SSCLSClear Signed Test FlagS $\leftarrow$ 0SSSEVASet Twos Complement Overflow.V $\leftarrow$ 1VCSETSet T in SREGT $\leftarrow$ 1TSS  |                  |              |   |  |              | 1       |
| BSETsFlag SetSREG(s)SREG(s)SREG(s)BCLRsFlag ClearSREG(s)SREG(s)SREG(s)SREG(s)BSTRr, bBit Store from Register to TTTR(b)TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow$ TNoneSECSECSet CarryC<-1   |                  |              |   |  |              | 1       |
| BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BSTRr, bBit Store from Register to T $T \leftarrow Rr(b)$ TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ NoneSECSet Carry $C \leftarrow 1$ CCLCClear Carry $C \leftarrow 0$ CSENSet Negative FlagN $\leftarrow 1$ NCLNClear Negative FlagN $\leftarrow 0$ NSEZSet Zero FlagZ $\leftarrow 1$ ZCLZClear Argen FlagI $\leftarrow 1$ ISEZSet Zero FlagZ $\leftarrow 1$ ZCLZClear Signed Test FlagI $\leftarrow 1$ ICLIGlobal Interrupt EnableI $\leftarrow 0$ ISESSet Signed Test FlagS $\leftarrow 1$ SCLSClear Signed Test FlagS $\leftarrow 0$ SSEVSet Twos Complement Overflow.V $\leftarrow 1$ VSETSet Tin SREGT $\leftarrow 1$ T  |                  |              |   |  |              | 1       |
| BSTRr, bBit Store from Register to TT $\leftarrow$ Rr(b)TNoneBLDRd, bBit load from T to RegisterRd(b) $\leftarrow$ TNoneNoneSECSet CarryC $\leftarrow$ 1CCCLCClear CarryC $\leftarrow$ 0CCSENSet Negative FlagN $\leftarrow$ 1NCCLNClear Negative FlagN $\leftarrow$ 0NCSEZSet Zero FlagZ $\leftarrow$ 1ZZCLZClear Zero FlagZ $\leftarrow$ 0ZCSEIGlobal Interrupt EnableI $\leftarrow$ 1ICCLISet Signed Test FlagS $\leftarrow$ 1SSCLSClear Signed Test FlagS $\leftarrow$ 0SSSEVSet Twos Complement Overflow.V $\leftarrow$ 1VVSETSet Twos Complement OverflowV $\leftarrow$ 0VTSETSet T in SREGT $\leftarrow$ 1TT  |                  |              | 5   |  |              | 1       |
| BLDRd, bBit load from T to RegisterRd(b) $\leftarrow$ TNoneSECSet CarryC $\leftarrow$ 1CCLCClear CarryC $\leftarrow$ 0SENSet Negative FlagN $\leftarrow$ 1CLNClear Negative FlagN $\leftarrow$ 0SEZSet Zero FlagZ $\leftarrow$ 1CLZClear Zero FlagZ $\leftarrow$ 0SEIGlobal Interrupt EnableI $\leftarrow$ 1CLIGlobal Interrupt DisableI $\leftarrow$ 0SESSet Signed Test FlagS $\leftarrow$ 1SESClear Signed Test FlagS $\leftarrow$ 0SEVSet Twos Complement Overflow.V $\leftarrow$ 1CLVClear Twos Complement OverflowV $\leftarrow$ 0SETSet T in SREGT $\leftarrow$ 1T  |                  |              |   |  |              | 1       |
| SECSet Carry $C \leftarrow 1$ CCCLCClear Carry $C \leftarrow 0$ CCSENSet Negative Flag $N \leftarrow 1$ NCCLNClear Negative Flag $N \leftarrow 0$ NCSEZSet Zero Flag $Z \leftarrow 1$ ZCCLZClear Zero Flag $Z \leftarrow 0$ ZCSEIGlobal Interrupt Enable $I \leftarrow 1$ ICCLIGlobal Interrupt Disable $I \leftarrow 0$ ICSESSet Signed Test Flag $S \leftarrow 1$ SSCSEVClear Xigned Test Flag $S \leftarrow 0$ SCCSEVSet Twos Complement Overflow. $V \leftarrow 0$ VCSETSet T in SREG $T \leftarrow 1$ TTC   |                  |              |   |  | -            |         |
| CLCClear Carry $C \leftarrow 0$ CCSENSet Negative Flag $N \leftarrow 1$ NCCLNClear Negative Flag $N \leftarrow 0$ NCSEZSet Zero Flag $Z \leftarrow 1$ ZCCLZClear Zero Flag $Z \leftarrow 0$ ZCSEIGlobal Interrupt Enable $I \leftarrow 1$ ICCLIGlobal Interrupt Disable $I \leftarrow 0$ ICSESSet Signed Test Flag $S \leftarrow 1$ SSCLSClear Signed Test Flag $S \leftarrow 0$ SCSEVSet Twos Complement Overflow. $V \leftarrow 1$ VCCLVClear Twos Complement Overflow $V \leftarrow 0$ VCSETSet T in SREG $T \leftarrow 1$ TT   |                  | HU, D        |   |  |              | 1       |
| SENSet Negative Flag $N \leftarrow 1$ NCLNClear Negative Flag $N \leftarrow 0$ NSEZSet Zero Flag $Z \leftarrow 1$ ZCLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ VCLVClear Twos Complement Overflow $V \leftarrow 0$ V   |                  |              |   |  |              | 1       |
| CLNClear Negative Flag $N \leftarrow 0$ NNSEZSet Zero Flag $Z \leftarrow 1$ ZZCLZClear Zero Flag $Z \leftarrow 0$ ZZSEIGlobal Interrupt Enable $I \leftarrow 1$ IICLIGlobal Interrupt Disable $I \leftarrow 0$ IISESSet Signed Test Flag $S \leftarrow 1$ SSCLSClear Signed Test Flag $S \leftarrow 0$ SISEVSet Twos Complement Overflow. $V \leftarrow 1$ VICLVClear Twos Complement Overflow $V \leftarrow 0$ VISETSet T in SREG $T \leftarrow 1$ TI   |                  | +            | ·   |  |              | 1       |
| SEZSet Zero Flag $Z \leftarrow 1$ $Z$ CLZClear Zero Flag $Z \leftarrow 0$ $Z$ SEIGlobal Interrupt Enable $I \leftarrow 1$ $I$ CLIGlobal Interrupt Disable $I \leftarrow 0$ $I$ SESSet Signed Test Flag $S \leftarrow 1$ $S$ CLSClear Signed Test Flag $S \leftarrow 0$ $S$ SEVSet Twos Complement Overflow. $V \leftarrow 1$ $V$ CLVClear Twos Complement Overflow $V \leftarrow 0$ $V$  |                  |              |   |  |              | 1       |
| CLZClear Zero Flag $Z \leftarrow 0$ $Z$ SEIGlobal Interrupt Enable $I \leftarrow 1$ $I$ CLIGlobal Interrupt Disable $I \leftarrow 0$ $I$ SESSet Signed Test Flag $S \leftarrow 1$ $S$ CLSClear Signed Test Flag $S \leftarrow 0$ $S$ SEVSet Twos Complement Overflow. $V \leftarrow 1$ $V$ CLVClear Twos Complement Overflow $V \leftarrow 0$ $V$ SETSet T in SREG $T \leftarrow 1$ $T$  |                  |              | · · ·   | 1  |              | 1       |
| SEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ VCLVClear Twos Complement Overflow $V \leftarrow 0$ VSETSet T in SREG $T \leftarrow 1$ T  |                  |              |   | 1  |              | 1       |
| $\begin{array}{c c c c c c c c c c c c c c c c c c c $   | CLZ              |              | Clear Zero Flag   | $Z \leftarrow 0$   | Z            | 1       |
| $\begin{array}{c c c c c c c c c c c c c c c c c c c $   | SEI              |              | Global Interrupt Enable                                     | ← 1  | 1            | 1       |
| CLS      Clear Signed Test Flag      S ← 0      S         SEV      Set Twos Complement Overflow.      V ← 1      V         CLV      Clear Twos Complement Overflow      V ← 0      V         SET      Set T in SREG      T ← 1      T  | CLI              |              | Global Interrupt Disable                                    | 1 ← 0  | 1            | 1       |
| CLS      Clear Signed Test Flag      S ← 0      S         SEV      Set Twos Complement Overflow.      V ← 1      V         CLV      Clear Twos Complement Overflow      V ← 0      V         SET      Set T in SREG      T ← 1      T  | SES              |              | Set Signed Test Flag  | S ← 1  | S            | 1       |
| SEV      Set Twos Complement Overflow.      V ← 1      V        CLV      Clear Twos Complement Overflow      V ← 0      V        SET      Set T in SREG      T ← 1      T  |                  |              |   |  |              | 1       |
| CLV      Clear Twos Complement Overflow      V ← 0      V        SET      Set T in SREG      T ← 1      T  |                  |              |   |  |              | 1       |
| SET      Set T in SREG      T ← 1      T   |                  |              |   |  |              | 1       |
|  |                  |              |   |  |              | 1       |
| minemonics Operands Description Operation Flags #C   |                  | Onerrada     |   |  |              | ·       |
|  | whemonics        | Operands     | Description   | Operation  | riags        | #Clocks |

AIMEL

if (I = 1) then PC  $\leftarrow$  PC + k + 1

None

1/2

## Instruction Set Summary (Continued)

Branch if Interrupt Enabled

BRIE

k

## Instruction Set Summary (Continued)

| CLT           |             | Clear T in SREG               | $T \leftarrow 0$                         | Т    | 1 |
|---------------|-------------|-------------------------------|--|------|---|
| SEH           |             | Set Half Carry Flag in SREG   | H ← 1                                    | Н    | 1 |
| CLH           |             | Clear Half Carry Flag in SREG | $H \leftarrow 0$                         | Н    | 1 |
| MCU CONTROL I | NSTRUCTIONS |                               |  |      |   |
| NOP           |             | No Operation                  |  | None | 1 |
| SLEEP         |             | Sleep                         | (see specific descr. for Sleep function) | None | 1 |
| WDR           |             | Watchdog Reset                | (see specific descr. for WDR/timer)      | None | 1 |





## **Ordering Information**

| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|-------------|--------------|---------------|---------|-----------------|
| 8           | 2.7 - 5.5    | ATmega8L-8AC  | 32A     | Commercial      |
|             |              | ATmega8L-8PC  | 28P3    | (0°C to 70°C)   |
|             |              | ATmega8L-8MC  | 32M1-A  |                 |
|             |              | ATmega8L-8AI  | 32A     | Industrial      |
|             |              | ATmega8L-8PI  | 28P3    | (-40°C to 85°C) |
|             |              | ATmega8L-8MI  | 32M1-A  |                 |
| 16          | 4.5 - 5.5    | ATmega8-16AC  | 32A     | Commercial      |
|             |              | ATmega8-16PC  | 28P3    | (0°C to 70°C)   |
|             |              | ATmega8-16MC  | 32M1-A  |                 |
|             |              | ATmega8-16AI  | 32A     | Industrial      |
|             |              | ATmega8-16PI  | 28P3    | (-40°C to 85°C) |
|             |              | ATmega8-16MI  | 32M1-A  |                 |

Note: This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

|        | Package Type  |
|--------|---|
| 32A    | 32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)                     |
| 28P3   | 28-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)                    |
| 32M1-A | 32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF) |

### **Packaging Information**

32A







#### 28P3



14 ATmega8(L)

32M1-A







### Erratas

ATmega8 Rev. D, E, F, and G The revision letter in this section refers to the revision of the ATmega8 device.

CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32 KHz
 Oscillator is Used to Clock the Asynchronous Timer/Counter2

# 1. CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32 KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2

When the internal RC Oscillator is used as the main clock source, it is possible to run the Timer/Counter2 asynchronously by connecting a 32 KHz Oscillator between XTAL1/TOSC1 and XTAL2/TOSC2. But when the internal RC Oscillator is selected as the main clock source, the CKOPT Fuse does not control the internal capacitors on XTAL1/TOSC1 and XTAL2/TOSC2. As long as there are no capacitors connected to XTAL1/TOSC1 and XTAL2/TOSC2, safe operation of the Oscillator is not guaranteed.

#### Problem fix/Workaround

Use external capacitors in the range of 20 - 36 pF on XTAL1/TOSC1 and XTAL2/TOSC2. This will be fixed in ATmega8 Rev. G where the CKOPT Fuse will control internal capacitors also when internal RC Oscillator is selected as main clock source. For ATmega8 Rev. G, CKOPT = 0 (programmed) will enable the internal capacitors on XTAL1 and XTAL2. Customers who want compatibility between Rev. G and older revisions, must ensure that CKOPT is unprogrammed (CKOPT = 1).

### Datasheet Change Log for ATmega8

Changes from Rev. 2486K-08/03 to Rev. 2486L-10/03

Changes from Rev. 2486K-08/03 to Rev. 2486L-10/03 This document contains a log on the changes made to the datasheet for ATmega8.

All page numbers refers to this document.

1. Updated "Calibrated Internal RC Oscillator" on page 28.

All page numbers refers to this document.

- 1. Removed "Preliminary" and TBDs from the datasheet.
- 2. Renamed ICP to ICP1 in the datasheet.
- 3. Removed instructions CALL and JMP from the datasheet.
- 4. Updated  $t_{RST}$  in Table 15 on page 36,  $V_{BG}$  in Table 16 on page 40, Table 100 on page 239 and Table 102 on page 241.
- 5. Replaced text "XTAL1 and XTAL2 should be left unconnected (NC)" after Table 9 in "Calibrated Internal RC Oscillator" on page 28. Added text regarding XTAL1/XTAL2 and CKOPT Fuse in "Timer/Counter Oscillator" on page 30.
- 6. Updated Watchdog Timer code examples in "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 43.
- 7. Removed bit 4, ADHSM, from "Special Function IO Register SFIOR" on page 56.
- 8. Added note 2 to Figure 103 on page 212.
- 9. Updated item 4 in the "Serial Programming Algorithm" on page 233.
- 10. Added  $t_{WD_{FUSE}}$  to Table 97 on page 234 and updated Read Calibration Byte, Byte 3, in Table 98 on page 235.
- 11. Updated Absolute Maximum Ratings\* and DC Characteristics in "Electrical Characteristics" on page 237.

All page numbers refers to this document.

- 1. Updated V<sub>BOT</sub> values in Table 15 on page 36.
- 2. Updated "ADC Characteristics" on page 243.
- 3. Updated "ATmega8 Typical Characteristics" on page 244.
- 4. Updated "Erratas" on page 16.

Changes from Rev. 2486I-12/02 to Rev. 2486J-02/03

Changes from Rev.

2486J-02/03 to Rev.

2486K-08/03

All page numbers refers to this document.





- 1. Improved the description of "Asynchronous Timer Clock clkASY" on page 24.
- 2. Removed reference to the "Multipurpose Oscillator" application note and the "32 kHz Crystal Oscillator" application note, which do not exist.
- 3. Corrected OCn waveforms in Figure 38 on page 88.
- 4. Various minor Timer 1 corrections.
- 5. Various minor TWI corrections.
- 6. Added note under "Filling the Temporary Buffer (Page Loading)" on page 213 about writing to the EEPROM during an SPM Page load.
- 7. Removed ADHSM completely.
- 8. Added section "EEPROM Write during Power-down Sleep Mode" on page 21.
- 9. Removed XTAL1 and XTAL2 description on page 5 because they were already described as part of "Port B (PB7..PB0) XTAL1/ XTAL2/TOSC1/TOSC2" on page 5.
- 10. Improved the table under "SPI Timing Characteristics" on page 241 and removed the table under "SPI Serial Programming Characteristics" on page 236.
- 11. Corrected PC6 in "Alternate Functions of Port C" on page 59.
- 12. Corrected PB6 and PB7 in "Alternate Functions of Port B" on page 56.
- 13. Corrected 230.4 Mbps to 230.4 kbps under "Examples of Baud Rate Setting" on page 156.
- 14. Added information about PWM symmetry for Timer 2 in "Phase Correct PWM Mode" on page 111.
- 15. Added thick lines around accessible registers in Figure 76 on page 166.
- 16. Changed "will be ignored" to "must be written to zero" for unused Z-pointer bits under "Performing a Page Write" on page 213.
- 17. Added note for RSTDISBL Fuse in Table 87 on page 220.
- 18.Updated drawings in "Packaging Information" on page 13.
- 1. Added errata for Rev D, E, and F on page 16.

Changes from Rev. 2486H-09/02 to Rev. 2486I-12/02

Changes from Rev. 2486G-09/02 to Rev. 2486H-09/02 1. Changed the Endurance on the Flash to 10,000 Write/Erase Cycles.

18 ATmega8(L)

Changes from Rev. 2486F-07/02 to Rev. 2486G-09/02

#### Changes from Rev. 2486E-06/02 to Rev. 2486F-07/02

Changes from Rev. 2486D-03/02 to Rev. 2486E-06/02

Changes from Rev. 2486C-03/02 to Rev. 2486D-03/02 All page numbers refers to this document.

1 Updated Table 103, "ADC Characteristics," on page 243.

All page numbers refers to this document.

- 1 Changes in "Digital Input Enable and Sleep Modes" on page 53.
- 2 Addition of OCS2 in "MOSI/OC2 Port B, Bit 3" on page 57.
- 3 The following tables has been updated:

Table 51, "CPOL and CPHA Functionality," on page 129, Table 59, "UCPOL Bit Settings," on page 155, Table 72, "Analog Comparator Multiplexed Input(1)," on page 192, Table 73, "ADC Conversion Time," on page 197, Table 75, "Input Channel Selections," on page 203, and Table 84, "Explanation of Different Variables used in Figure 103 and the Mapping to the Z-pointer," on page 218.

#### 5 Changes in "Reading the Calibration Byte" on page 230.

#### 6 Corrected Errors in Cross References.

All page numbers refers to this document.

1 Updated Some Preliminary Test Limits and Characterization Data

The following tables have been updated:

Table 15, "Reset Characteristics," on page 36, Table 16, "Internal Voltage Reference Characteristics," on page 40, DC Characteristics on page 237, Table , "ADC Characteristics," on page 243.

#### 2 Changes in External Clock Frequency

Added the description at the end of "External Clock" on page 30. Added period changing data in Table 99, "External Clock Drive," on page 239.

#### 3 Updated TWI Chapter

More details regarding use of the TWI bit rate prescaler and a Table 65, "TWI Bit Rate Prescaler," on page 170.

All page numbers refers to this document.

#### 1 Updated Typical Start-up Times.

The following tables has been updated:

Table 5, "Start-up Times for the Crystal Oscillator Clock Selection," on page 26, Table 6, "Start-up Times for the Low-frequency Crystal Oscillator Clock Selection," on page 26, Table 8, "Start-up Times for the External RC Oscillator Clock Selection," on page 27, and Table 12, "Start-up Times for the External Clock Selection," on page 30.

2 Added "ATmega8 Typical Characteristics" on page 244.





#### Changes from Rev. 2486B-12/01 to Rev. 2486C-03/02

All page numbers refers to this document.

#### 1 Updated TWI Chapter.

More details regarding use of the TWI Power-down operation and using the TWI as Master with low TWBRR values are added into the datasheet.

Added the note at the end of the "Bit Rate Generator Unit" on page 167.

Added the description at the end of "Address Match Unit" on page 167.

#### 2 Updated Description of OSCCAL Calibration Byte.

In the datasheet, it was not explained how to take advantage of the calibration bytes for 2, 4, and 8 MHz Oscillator selections. This is now added in the following sections:

Improved description of "Oscillator Calibration Register – OSCCAL" on page 29 and "Calibration Byte" on page 221.

#### 3 Added Some Preliminary Test Limits and Characterization Data.

Removed some of the TBD's in the following tables and pages:

Table 3 on page 24, Table 15 on page 36, Table 16 on page 40, Table 17 on page 42, "TA =  $-40 \times C$  to  $85 \times C$ , VCC = 2.7V to 5.5V (unless otherwise noted)" on page 237, Table 99 on page 239, and Table 102 on page 241.

#### 4 Updated Programming Figures.

Figure 104 on page 222 and Figure 112 on page 232 are updated to also reflect that AVCC must be connected during Programming mode.

5 Added a Description on how to Enter Parallel Programming Mode if RESET Pin is Disabled or if External Oscillators are Selected.

Added a note in section "Enter Programming Mode" on page 224.



#### **Atmel Corporation**

2325 Orchard Parkway San Jose, CA 95131, USA Tel: 1(408) 441-0311 Fax: 1(408) 487-2600

#### **Regional Headquarters**

#### Europe

Atmel Sarl Route des Arsenaux 41 Case Postale 80 CH-1705 Fribourg Switzerland Tel: (41) 26-426-5555 Fax: (41) 26-426-5500

#### Asia

Room 1219 Chinachem Golden Plaza 77 Mody Road Tsimshatsui East Kowloon Hong Kong Tel: (852) 2721-9778 Fax: (852) 2722-1369

#### Japan

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**

Memory

2325 Orchard Parkway San Jose, CA 95131, USA Tel: 1(408) 441-0311 Fax: 1(408) 436-4314

#### Microcontrollers

2325 Orchard Parkway San Jose, CA 95131, USA Tel: 1(408) 441-0311 Fax: 1(408) 436-4314

La Chantrerie BP 70602 44306 Nantes Cedex 3, France Tel: (33) 2-40-18-18-18 Fax: (33) 2-40-18-19-60

#### ASIC/ASSP/Smart Cards

Zone Industrielle 13106 Rousset Cedex, France Tel: (33) 4-42-53-60-00 Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd. Colorado Springs, CO 80906, USA Tel: 1(719) 576-3300 Fax: 1(719) 540-1759

Scottish Enterprise Technology Park Maxwell Building East Kilbride G75 0QR, Scotland Tel: (44) 1355-803-000 Fax: (44) 1355-242-743

#### **RF**/Automotive

Theresienstrasse 2 Postfach 3535 74025 Heilbronn, Germany Tel: (49) 71-31-67-0 Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd. Colorado Springs, CO 80906, USA Tel: 1(719) 576-3300 Fax: 1(719) 540-1759

#### Biometrics/Imaging/Hi-Rel MPU/

High Speed Converters/RF Datacom Avenue de Rochepleine BP 123 38521 Saint-Egreve Cedex, France Tel: (33) 4-76-58-30-00 Fax: (33) 4-76-58-34-80

*Literature Requests* www.atmel.com/literature

**Disclaimer:** Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

© Atmel Corporation 2003. All rights reserved. Atmel<sup>®</sup> and combinations thereof, AVR<sup>®</sup>, and AVR Studio<sup>®</sup> are the registered trademarks of Atmel Corporation or its subsidiaries. Microsoft<sup>®</sup>, Windows<sup>®</sup>, Windows NT<sup>®</sup>, and Windows XP<sup>®</sup> are the registered trademarks of Microsoft Corporation. Other terms and product names may be the trademarks of others



This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.