



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

| Product Status | Obsolete |
|----------------------------|--------------------------------------------------------------------------|
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 4MHz |
| Connectivity | SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 20 |
| Program Memory Size | 4KB (2K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 256 x 8 |
| RAM Size | 128 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 6V |
| Data Converters | A/D 6x10b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C |
| Mounting Type | Through Hole |
| Package / Case | 28-DIP (0.300", 7.62mm) |
| Supplier Device Package | 28-PDIP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/at90ls4433-4pi |

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

Architectural Overview

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

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

The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 5 shows the AT90S4433 AVR RISC microcontroller architecture.

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









Figure 24. Reset Logic



Table 4. Reset Characteristics ($V_{CC} = 5.0V$)

| Symbol | Parameter | Min | Тур | Мах | Units |
|--------------------|-------------------------------------------------|---------------------|---------------------|---------------------|-------|
| (4) | Power-on Reset Threshold Voltage, rising | 1.0 | 1.4 | 1.8 | V |
| V _{POT} . | Power-on Reset Threshold Voltage, falling | 0.4 | 0.6 | 0.8 | V |
| V _{RST} | RESET Pin Threshold Voltage | | 0.6 V _{CC} | | V |
| V _{BOT} | Brown-out Reset | 2.2 (BODLEVEL=1) | 2.7 (BODLEVEL=1) | 3.0 (BODLEVEL=1) | M |
| | Voltage | 3.5 (BODLEVEL=0) | 4.0 (BODLEVEL=0) | 4.5 (BODLEVEL=0) | V |

Note: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).



the TEMP Register. Consequently, the Low Byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

Timer/Counter1 Output Compare Register – OCR1H and OCR1L

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| \$2B (\$4B) | MSB | | | | | | | | OCR1H |
| \$2A (\$4A) | | | | | | | | LSB | OCR1L |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | • |
| Read/Write | R/W | |
| | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

The Output Compare Register is a 16-bit read/write register.

The Timer/Counter1 Output Compare Register contains the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status Register.

Since the Output Compare Register (OCR1) is a 16-bit register, a temporary register TEMP is used when OCR1 is written to ensure that both bytes are updated simultaneously. When the CPU writes the High Byte, OCR1H, the data is temporarily stored in the TEMP Register. When the CPU writes the Low Byte, OCR1L, the TEMP Register is simultaneously written to OCR1H. Consequently, the High Byte OCR1H must be written first for a full 16-bit register write operation.

The TEMP Register is also used when accessing TCNT1 and ICR1. If the main program and interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program.

AT90S/LS4433

UART The AT90S4433 features a full duplex (separate Receive and Transmit Registers) Universal Asynchronous Receiver and Transmitter (UART). The main features are: • Baud Rate Generator Generates any Baud Rate • High Baud Rates at Low XTAL Frequencies • 8 or 9 Bits Data • Noise Filtering • Overrun Detection • Framing Error Detection • False Start Bit Detection • Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete • Multi-processor Communication Mode

A block schematic of the UART Transmitter is shown in Figure 40.

Data Transmission



Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register (UDR). Data is transferred from UDR to the Transmit Shift Register when:

- A new character has been written to UDR after the stop bit from the previous character has been shifted out. The Shift Register is loaded immediately.
- A new character has been written to UDR before the stop bit from the previous character has been shifted out. The Shift Register is loaded when the stop bit of the character currently being transmitted has been shifted out.

When data is transferred from UDR to the Shift Register, the UDRE (UART Data Register Empty) bit in the UART Control and Status Register A, UCSRA, is set. When this bit is set (one), the UART is ready to receive the next character. At the same time as the





data is transferred from UDR to the 10(11)-bit Shift Register, bit 0 of the Shift Register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9-bit data word is selected (the CHR9 bit in the UART Control and Status Register B, UCSRB is set), the TXB8 bit in UCSRB is transferred to bit nine in the Transmit Shift Register.

On the baud rate clock following the transfer operation to the Shift Register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the Shift Register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR Register to send when the stop bit is shifted out, the UDRE Flag will remain set until UDR is written again. When no new data has been written, and the stop bit has been present on TXD for one bit length, the TX Complete Flag, TXC, in UCSRA is set.

The TXEN bit in UCSRB enables the UART Transmitter when set (one). When this bit is cleared (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to PD1, which is forced to be an output pin regardless of the setting of the DDD1 bit in DDRD.

Data Reception Figure 41 shows a block diagram of the UART Receiver.





UART Control

UART I/O Data Register - UDR



The UDR Register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data Register is written. When reading from UDR, the UART Receive Data Register is read.

UART Control and Status Register A – UCSRA

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---------------|-----|-----|------|----|----|---|---|------|-------|
| \$0B (\$2B) | RXC | TXC | UDRE | FE | OR | - | - | MPCM | UCSRA |
| Read/Write | R | R/W | R | R | R | R | R | R/W | • |
| Initial Value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |

• Bit 7 – RXC: UART Receive Complete

This bit is set (one) when a received character is transferred from the Receiver Shift Register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCSRB is set, the UART Receive Complete interrupt will be executed when RXC is set (one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

• Bit 6 – TXC: UART Transmit Complete

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift Register has been shifted out and no new data has been written to UDR. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter Receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCSRB is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical "1" to the bit.

• Bit 5 – UDRE: UART Data Register Empty

This bit is set (one) when a character written to UDR is transferred to the Transmit Shift Register. Setting of this bit indicates that the Transmitter is ready to receive a new character for transmission.

When the UDRIE bit in UCSRB is set, the UART Transmit Complete interrupt to be executed as long as UDRE is set. UDRE is cleared by writing UDR. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDR in order to clear UDRE, otherwise a new interrupt will occur once the interrupt routine terminates.

UDRE is set (one) during reset to indicate that the transmitter is ready.





Analog Comparator

The Analog Comparator compares the input values on the positive input PD6 (AIN0) and negative input PD7 (AIN1). When the voltage on the positive input PD6 (AIN0) is higher than the voltage on the negative input PD7 (AIN1), the Analog Comparator Output, ACO, is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 43.





Analog Comparator Control and Status Register – ACSR

| • | Bit 7 – | ACD: | Analog | Comparator | Disable |
|---|---------|------|--------|------------|---------|
|---|---------|------|--------|------------|---------|

R/W

0

R

N/A

R/W

0

Read/Write

Initial Value

When this bit is set (one), the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. When changing the ACD bit, the Analog Comparator interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise, an interrupt can occur when the bit is changed.

R/W

0

R/W

0

R/W

0

R/W

0

R/W

0

• Bit 6 – AINBG: Analog Comparator Bandgap Select

When this bit is set, BOD is enabled and the BODEN is programmed, a fixed bandgap voltage of $1.22V \pm 0.1V$ replaces the normal input to the positive input (AIN0) of the comparator. When this bit is cleared, the normal input pin, PD6, is applied to the positive input of the comparator.

• Bit 5 – ACO: Analog Comparator Output

ACO is directly connected to the comparator output.



Analog-to-Digital Converter

Features

- 10-bit Resolution
- ±2 LSB Absolute Accuracy
- 0.5 LSB Integral Non-linearity
- 65 260 µs Conversion Time
- Up to 15 kSPS
- Six Multiplexed Input Channels
- Rail-to-Rail Input Range
- Free Run or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

The AT90S4433 features a 10-bit successive approximation ADC. The ADC is connected to a 6-channel Analog Multiplexer, which allows each pin of Port C to be used as an input for the ADC. The ADC contains a Sample and Hold Amplifier, which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 44.

The ADC has two separate analog supply voltage pins: AVCC and AGND. AGND must be connected to GND, and the voltage on AVCC must not differ from V_{CC} more than ±0.3V. See the section "ADC Noise Canceling Techniques" on page 70 for how to connect these pins.

An external reference voltage must be applied to the AREF pin. This voltage must be in the range 2.0 - AVCC.

Figure 44. Analog-to-Digital Converter Block Schematic





keeps running for as long as the ADEN bit is set and is continuously reset when ADEN is low.

When initiating a conversion by setting the ADSC bit in ADCSR, the conversion starts at the following rising edge of the ADC clock cycle. The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of the conversion. The result is ready and written to the ADC Result Register after 13 cycles. In Single Conversion mode, the ADC needs one more clock cycle before a new conversion can be started (see Figure 47). If ADSC is set high in this period, the ADC will start the new conversion immediately. In Free Run mode, a new conversion will be started immediately after the result is written to the ADC Result Register. Using Free Run mode and an ADC clock frequency of 200 kHz gives the lowest conversion time, 65 µs, equivalent to 15.4 kSPS. For a summary of conversion times, see Table 21.





Table 21. ADC Conversion Time

| Condition | Sample Cycle Number | Result Ready(Cycle Number) | Total Conversion Time (Cycles) | Total Conversion Time (µs) |
|--------------------------|------------------------|-------------------------------|-----------------------------------|-------------------------------|
| 1st Conversion, Free Run | 13.5 | 25 | 25 | 125 - 500 |
| 1st Conversion, Single | 13.5 | 25 | 26 | 130 - 520 |
| Free Run Conversion | 1.5 | 13 | 13 | 65 - 260 |
| Single Conversion | 1.5 | 13 | 14 | 70 - 280 |

Port C Schematics

Note that all port pins are synchronized. The synchronization latch is, however, not shown in the figure.







Port D as General Digital I/O

PDn, General I/O pin: The DDDn bit in the DDRD Register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, the PDn has to be cleared (zero) or the pin has to be configured as an output pin. The port pins are tristated when a reset condition becomes active, even if the clock is not running.

| Table 27. DDDn Bits on Port D Pins ⁽¹⁾ | |
|-----------------------------------------------------------|--|
|-----------------------------------------------------------|--|

| DDDn | PORTDn | I/O | Pull-up | Comment |
|------|--------|--------|---------|---------------------------------------------|
| 0 | 0 | Input | No | Tri-state (high-Z) |
| 0 | 1 | Input | Yes | PDn will source current if ext. pulled low. |
| 1 | 0 | Output | No | Push-pull Zero Output |
| 1 | 1 | Output | No | Push-pull One Output |

4

Note: 1. n: 7,6..0, pin number.

Alternate Functions of Port D • AIN1 – Port D, Bit 7

AIN1, Analog Comparator Negative Input. When configured as an input (DDD7 is cleared [zero]), and with the internal MOS pull-up resistor switched off (PD7 is cleared [zero]), this pin also serves as the negative input of the On-chip Analog Comparator. During Power-down mode, the Schmitt trigger of the digital input is disconnected. This allows analog signals, which are close to $V_{CC}/2$, to be present during Power-down without causing excessive power consumption.

• AIN0 - Port D, Bit 6

AIN0, Analog Comparator Positive Input. When configured as an input (DDD6 is cleared [zero]), and with the internal MOS pull-up resistor switched off (PD6 is cleared [zero]), this pin also serves as the positive input of the On-chip Analog Comparator. During Power-down mode, the Schmitt trigger of the digital input is disconnected. This allows analog signals, which are close to $V_{CC}/2$, to be present during Power-down without causing excessive power consumption.

• T1 – Port D, Bit 5

T1, Timer/Counter1 Counter Source. See the Timer description for further details

• T0 - Port D, Bit 4

T0: Timer/Counter0 Counter Source. See the Timer description for further details.

• INT1 – Port D, Bit 3

INT1, External Interrupt Source 1: The PD3 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details and how to enable the source.

• INT0 – Port D, Bit 2

INTO, External Interrupt Source 0: The PD2 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details and how to enable the source.







Figure 59. Port D Schematic Diagram (Pins PD2 and PD3)





- G: Write Data High Byte
- 1. Set BS to "1". This selects high data.
- 2. Give WR a negative pulse. This starts programming of the data byte. RDY/BSY goes low.
- 3. Wait until RDY/BSY goes high to program the next byte.

(See Figure 64 for signal waveforms.)

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered:

- The command needs to be loaded only once when writing or reading multiple memory locations.
- Address High Byte needs to be loaded only before programming a new 256-word page in the Flash.
- Skip writing the data value \$FF, that is, the contents of the entire Flash and EEPROM after a Chip Erase.

These considerations also apply to EEPROM programming and Flash, EEPROM and signature bytes reading.







Serial Downloading

Both the Program and Data memory arrays can be programmed using the SPI bus while RESET is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output) (see Figure 66). After RESET is set low, the Programming Enable instruction needs to be executed first before program/erase instructions can be executed.





For the EEPROM, an auto-erase cycle is provided within the self-timed write instruction and there is no need to first execute the Chip Erase instruction. The Chip Erase instruction turns the content of every memory location in both the program and EEPROM arrays into \$FF.

The Program and EEPROM memory arrays have separate address spaces: 0000 to \$07FF for Program memory and \$0000 to \$00FF for EEPROM memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 XTAL1 clock cycles High: > 2 XTAL1 clock cycles

When writing serial data to the AT90S4433, data is clocked on the rising edge of CLK.

When reading data from the AT90S4433, data is clocked on the falling edge of CLK. See Figure 67, Figure 68 and Table 36 for details.

To program and verify the AT90S4433 in the Serial Programming mode, the following sequence is recommended (see 4-byte instruction formats in Table 35):

1. Power-up sequence:

Apply power between V_{CC} and GND while RESET and SCK are set to "0". If a crystal is not connected across pins XTAL1 and XTAL2, apply a clock signal to the XTAL1 pin. In some systems, the programmer cannot guarantee that SCK is held low during Power-up. In this case, RESET must be given a positive pulse of at least two XTAL1 cycles' duration after SCK has been set to "0".

- 2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/PB3.
- 3. The serial programming instructions will not work if the communication is out of synchronization. When in sync, the second byte (\$53) will echo back when issu-



Serial Programming Algorithm



Table 35. Serial Programming Instruction Set

| | Instruction Format | | | | |
|------------------------|--------------------|--------------------|-------------------|--------------------|-----------------------------------------------------------------------------------------------------|
| Instruction | Byte 1 | Byte 2 | Byte 3 | Byte4 | Operation |
| Programming Enable | 1010 1100 | 0101 0011 | xxxx xxxx | xxxx xxxx | Enable Serial Programming while RESET is low. |
| Chip Erase | 1010 1100 | 100x xxxx | xxxx xxxx | XXXX XXXX | Chip Erase Flash and EEPROM memory arrays. |
| Read Program Memory | 0010 H 000 | xxxx x aaa | bbbb bbbb | 0000 0000 | Read H (high or low) data o from program memory at word address a : b . |
| Write Program Memory | 0100 H 000 | xxxx x aaa | bbbb bbbb | iiii iiii | Write H (high or low) data i to program memory at word address a : b . |
| Read EEPROM Memory | 1010 0000 | xxxx xxxx | bbbb bbbb | 0000 0000 | Read data o from EEPROM memory at address a : b . |
| Write EEPROM Memory | 1100 0000 | xxxx xxxx | bbbb bbbb | iiii iiii | Write data i to EEPROM memory at address a:b. |
| Write Lock Bits | 1010 1100 | 1111 1 21 1 | XXXX XXXX | xxxx xxxx | Write Lock bits. Set bits <i>1,2</i> = "0" to program Lock bits. |
| Read Lock Bits | 0101 1000 | XXXX XXXX | XXXX XXXX | xxxx x 21 x | Read Lock bits. "0" = programmed, "1" = unprogrammed. |
| Read Sigature Bytes | 0011 0000 | xxxx xxxx | xxxx xx bb | 0000 0000 | Read signature byte \mathbf{o} at address $\mathbf{b}^{(1)}$ |
| Write Fuse Bits | 1010 1100 | 101 7 6543 | XXXX XXXX | XXXX XXXX | Set bits 7 - 3 = "0" to program, "1" to unprogram. |
| Read Fuse Bits | 0101 0000 | XXXX XXXX | xxxx xxxx | xx87 6543 | Read Fuse bits. "0" = programmed, "1" = unprogrammed. |

Note: 1. The signature bytes are not readable in lock mode 3, i.e., both Lock bits programmed.

a = address high bits

b = address low bits

H = 0 - Low Byte, 1 - High Byte

- $\mathbf{o} = data out$
- i = data in
- x = don't care
- $\mathbf{1} = \text{Lock bit 1}$
- **2** = Lock bit 2
- 3 = CKSEL0 Fuse
- 4 = CKSEL1 Fuse
- 5 = CKSEL2 Fuse
- 6 = BODEN Fuse
- 7 = BODLEVEL Fuse
- 8 = SPIEN Fuse



Figure 71. Active Supply Current vs. V_{CC}



























AT90S/LS4433

Instruction Set Summary (Continued)

| IDRd, YLoad Index and Peaken.Rd - (Y) 1 (Y)None2LDRd, YLoad Index and Peaken.Rd - (Y) 1 (Y)None2LDRd, YLoad Index and Peaken.Rd - (Y) 1 (Y)None2LDRd, ZLoad Index and Peaken.Rd - (Y) 2 (Y)None2LDRd, ZLoad Index and Peaken.None 2 (Y)None2LDRd, ZLoad Index and Peaken.None 2 (Y)None2LDRd, ZLoad Index and Peaken.None 2 (Y)None2LDNoneSom Index and Peaken.None 2 (Y)None2STY, RrSom Index and Peaken.Y - Y - Y + Y + 1.None2STY, RrSom Index and Peaken.Y - Y + Y + Y + 1.None2STY, RrSom Index and Peaken.Y - Y + Y + Y + 1.None2STY, RrSom Index and Peaken.Y - Y + Y + Y + 1.None2STY, RrSom Index and Peaken.Y - Y + Y + Y + 1.None2STY, RrSom Index and Peaken.Y - Y + Y, Y + Y + 1.None2STY, RrSom Index and Peaken.Y - Y + Y, Y + Y + 1.None2 | Mnemonic | Operands | Description | Operation | Flags | # Clocks |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|--------------|----------------------------------|--------------------------------------------------------------------|---------|----------|
| L0R4.~%Losinderian Parene.R4 - (*) (*) * (*) * (*)Neme2L00R0.~%Losinderian Parene.R4 - (*) * (*)Neme2L01R0.7Losinderian Parene.R4 - (*)Neme2L01R0.7Losinderian Parene.R4 - (*)Neme2L01R0.7Losinderian Parene.R4 - (*)Neme2L01R0.7Losinderian Parene.R4 - (*)Neme2L01R0.7Losinderian Parene.R5 - (*)Neme2L01R0.7Losinderian Parene.R5 - (*)Neme2L01NetSecrediad Parene.(*)Neme2L01NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Net2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)Neme2L11NetSecrediad Parene.(*)N | LD | Rd, Y | Load Indirect | $Rd \leftarrow (Y)$ | None | 2 |
| LDRA.*qLoad Indirect MoneyaborY - Y - 1, Rd - (Y)None2LDRA/*qLoad Indirect MoneyaborRd - (2)None2LDRd, ZLoad Indirect MoneyaborRd - (2), Z - 2 + 1None2LDRd, ZLoad Indirect MoneyaborRd - (2), Z - 2 + 1None2LDRd, ZLoad Indirect MoneyaborRd - (2), Z - 2 + 1None2LDRd, ZLoad Indirect MoneyaborRd - (2), Z - 2 + 1None2LDRd, ZLoad Indirect MoneyaborRd - (2), QNone2LDRd, ZLoad Indirect MoneyaborRd - (2), QNone2LDRd, ZLoad Indirect MoneyaborRd - (2), QNone2STX, RSome Indirect MoneyaborRd - (2), QNone2STY, RSome Indirect Moneyabor(2) - RNone2STY, RSome Indirect Moneyabor(2) - R <td< td=""><td>LD</td><td>Rd, Y+</td><td>Load Indirect and Post-inc.</td><td>$Rd \leftarrow (Y), Y \leftarrow Y + 1$</td><td>None</td><td>2</td></td<> | LD | Rd, Y+ | Load Indirect and Post-inc. | $Rd \leftarrow (Y), Y \leftarrow Y + 1$ | None | 2 |
| LDDRA.7Load Indived MiDiglacementRA (- Q)RA (- Q)RA (- Q)LDRA,2Load Indived MathicRA (- Q)RA (- Q)RA (- Q)LDRA,2Load Indived MathicRA (- Q)RA (- Q)RA (- Q)LDSRA,2Load Indived MiDiglacementRA (- Q)RA (- Q)RA (- Q)STX,RSon Indived MiDiglacementRA (- Q)RA (- Q)RA (- Q)STX,RSon Indived MiDiglacementRA (- X), IA (- X), I | LD | Rd, -Y | Load Indirect and Pre-dec. | $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ | None | 2 |
| LDR4_2Load Infreid and Pacin(n.R4 - (2)None2LDR6, 2-4Load Infreid and Pacin(n.R4 - (2, 2 - 14)None2LDR6, 2-4Load Infreid and Pacin(n.R4 - (2, -1)None2LDSR6, R4Load Infreid SM DegatomentR4 - (2, -1)None2LDSR4, R4Load Infreid SM DegatomentR4 - (2, -1)None2STX, R7Stoon Indicet and Pacheto.(0) - R7.None2STX, R7Stoon Indicet and Pacheto.(1) - R7. + V+ 1.1None2STY, R7Stoon Indicet and Pacheto.(1) - R7. + V+ 1.1None2STY, R7Stoon Indicet and Pacheto.(1) - R7. + V+ 1.1None2STY, R7Stoon Indicet and Pacheto.(2) - R7 Z+ 1.1None2STZ, R7Stoon Indicet an | LDD | Rd,Y+q | Load Indirect with Displacement | $Rd \leftarrow (Y + q)$ | None | 2 |
| LDR4.2Load Indication Position.R4.4 (2) Z + 2 + 1None2LDDR6.4.2Load Indication Prode.2 - 2 - 1, R4 (2)None2LDSR6.4.4Load Indication Prode.R6 + 0)None2STX.RStart Indication PSAMR6 + 0)None2STX.RStart Indication PSAMR6 + 0)None2STX.RStart Indication PSAM(0) + R × X + 11None2STX.RStart Indication Product(0) + R × X + 11None2STX.RStart Indication Product(1) + R × X + 11None2STX.RStart Indication Product(1) + R × X + 11None2STX.RStart Indication Product(1) + R × Y + 11None2STX.RStart Indication Product(1) + R × Y + 11None2STY.RStart Indication Product(2) + R × Y + 11None2STY.RStart Indication Product(2) + R × Y + 11None2STY.RStart Indication Product(2) + R × Y + 11None2STY.R <td< td=""><td>LD</td><td>Rd, Z</td><td>Load Indirect</td><td>$Rd \leftarrow (Z)$</td><td>None</td><td>2</td></td<> | LD | Rd, Z | Load Indirect | $Rd \leftarrow (Z)$ | None | 2 |
| IDR3.2*qLoad Indiver And Pho-Soc.Z - 2 - 1, R3 - (2)None2LDSR4, L*Load Molect Mon SPAMR4 - (2 - 0)None2LDSR4, K*Load Molect Mon SPAMR4 - (A)None2STX, R*Store Indivers and Pho-Iso.(0) - R*, X - X + 1None2STX, R*Store Indivers and Pho-Iso.(0) - R*, X - X + 1None2STX, R*Store Indivers and Pho-Iso.(Y) - R*, X - X + 1None2STY, R*Store Indivers and Pho-Iso.(Y - (-), R*, Y - Y + 1)None2STY, R*Store Indivers and Pho-Iso.(Y - (-), R*, Y - Y + 1)None2STY, R*Store Indivers and Pho-Iso.(Y - (-), R*None2STY, R*Store Indivers and Pho-Iso.(Y - (-), R*None2STZ, R*Store Indivers and Pho-Iso. </td <td>LD</td> <td>Rd, Z+</td> <td>Load Indirect and Post-inc.</td> <td>$Rd \leftarrow (Z), Z \leftarrow Z + 1$</td> <td>None</td> <td>2</td> | LD | Rd, Z+ | Load Indirect and Post-inc. | $Rd \leftarrow (Z), Z \leftarrow Z + 1$ | None | 2 |
| IDDR4, kLoad Direct mon SRAMR4 – (2 - q)None2STX, RStore Inderes and Postan.(A) – Rr.None2STX, RStore Inderes and Postan.(A) – Rr. X - X + 1 (A) – Rr.None2STX, RStore Inderes and Postan.(A) – Rr. X - X + 1 (A) – Rr.None2STX, RStore Inderes and Postan.(A) – Rr. X - X + 1 (A) – Rr.None2STY, RStore Inderes and Postan.(A) – Rr. Y - Y + 1 MNone2STY, RStore Inderes and Postan.(Y) – Rr. Y - Y + 1 MNone2STY, RStore Inderes and Postan.(Z) – Rr. Y - Y + 1 MNone2STY, RStore Inderes and Postan.(Z) – Rr. Y - Y + 1 MNone2STY, RStore Inderes and Postan.(Z) – Rr. Y - Y + 1 MNone2STY, RStore Inderes and Postan.(Z) – Rr. Y - Y + 1 MNone2STY, RStore Inderes and Postan.(Z) – Rr.None2STZ, RStore Inderes and Postan.(Z) | LD | Rd, -Z | Load Indirect and Pre-dec. | $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ | None | 2 |
| IDSRef.Lobentron SRAMRef.Ref.None2STX.RrStore Indicat and Posten.(0, - Rr, X - X + 1.None2STX.RrStore Indicat and Posten.X, - K, Y, O, - Rr.None2STY,RrStore Indicat and Posten.(1, - Rr, Y, - Y, T, Y), - Rr.None2STY,R.RStore Indicat and Posten.(1, - Rr, Y, - Y, Y), - Rr.None2STY,R.RStore Indicat and Posten.(1, - Rr, Y, - Y, Y, Y), - Rr.None2STY,R.RStore Indicat and Posten.(1, - Rr, Y, - Y, Y, Y), - Rr.None2STZ,RrStore Indicat and Posten.(1, - Rr, Z, - Z, 1, T, T, - Rr.None2STZ,RrStore Indicat and Posten.(1, - Rr, Z, - Z, 1, T, T, - Rr.None2STZ,RrStore Indicat and Posten.(2, - Rr.None22STZ,RrStore Indicat and Posten.(2, - C, T, T, T, T, - Rr.None2STZ,RrStore Indicat and Posten.(2, - C, T, T, T, T, - Rr.None2STZ,RrStore Indicat and Posten.Rd - Pr.None22STZ,RrStore Indicat and Posten.(2, - C, T, T, T, T, - Rr.None2STZ,RrStore Indicat and Posten.Rd - C, 2None32STZ,RrStore Indicat and Posten.Rd - C, 2None32STZ,RrStore Indicat and Posten.Rd - | LDD | Rd, Z+q | Load Indirect with Displacement | $Rd \leftarrow (Z + q)$ | None | 2 |
| STX, R/CStore Indicat and Packac.(A) e, RX × 14.1None2STX, R/RStore Indicat and Packac.(A) e, RX × 1.4, (A) e, RTNone2STX, R/RStore Indicat and Packac.(A) e, RX × 1.4, (A) e, RTNone2STY, R/RStore Indicat and Packac.(A) e, RY × Y + 1.4None2STY, R/RStore Indicat and Packac.Y e, Y + (Y) e, RTNone2STY, R/RStore Indicat and Packac.(A) e, RY, X + Y, Y + Y, Y + RNone2STY, R/RStore Indicat and Packac.(A) e, RY, X + Y, Y + Y, | LDS | Rd, k | Load Direct from SRAM | $Rd \leftarrow (k)$ | None | 2 |
| STX, RrBore Index and Pesific.(M) - F(X, X - Λ, 1(Λ) - F(X)None2STX, RrBore Index and Pesific.X + Δ - X, 1(Λ) - F(X)None2STY, RrBore Index and Pesific.(M) - F(X + Y + 1)None2STY, RrBore Index and Pesific.(M) - F(X + Y + 1)None2STY, RrBore Index and Pesific.(M) - F(X + Y + 1)None2STY, RrBore Index and Pesific.(M) - F(X + Y + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + Z + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + Z + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + Z + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + Z + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + Z + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + G + X + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + G + X + 1)None2STZ, RrBore Index and Pesific.(D) - F(X + G + X + 1)None2STRdPale Pale PaleNone12STRdDo DaritPale PaleNone22STRdPale PaleNone122STRdDo DaritPale PaleNone22 | ST | X, Rr | Store Indirect | $(X) \leftarrow Rr$ | None | 2 |
| ST X, R Stere Indext and Predic. X ← X + 1, (λ) ← Pr None 2 ST Y, R Stere Indext and Predic. (P) + Rr V ← V + 1 None 2 ST Y, R Stere Indext and Predic. (P) + Rt V + V + 1 None 2 ST Y, R Stere Indext and Predic. (P + R) + V + 1 None 2 ST Z, R Stere Indext and Predic. (Q) + Rt / V + 2 + 1 None 2 ST Z, R Stere Indext and Predic. (Z + Rt / L) + Rt None 2 ST Z, R Stere Indext and Predic. (Z + Rt / L) + Rt None 2 ST Z, R Stere Indext and Predic. (Z + Rt / L) + Rt None 2 ST X, R Stere Indext and Predic. (Z + Rt / R) None 2 ST X, R Stere Indext and Predic. (Z + Rt / R) None 2 ST X, R Stere Indext and Predic. Stere Indext and Predic. None 2 ST R, R Dead Pre | ST | X+, Rr | Store Indirect and Post-inc. | $(X) \leftarrow \operatorname{Rr}, X \leftarrow X + 1$ | None | 2 |
| str Y, Rr Stree Indext and Prachine. (Y) = Rr None 2 ST Y, Rr Stee Indext and Prachine. (Y) + Y, Y + Y + 1. None 2 ST Y, Rr Stee Indext and Prachine. (Y + Y, Y + Y), (Y) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST Z, Rr Stee Indext and Prachine. (Z) + Rr None 2 ST L, Rr Stee Indext and Prachine. RG + C/D None 2 DUT Prach Out Prat Rd + STACK Non | ST | -X, Rr | Store Indirect and Pre-dec. | $X \leftarrow X - 1, (X) \leftarrow Rr$ | None | 2 |
| ST Y, Rr Store Indices and Peside. Y) − RY, V + V + 1. None 2 STD Y, Rr Store Indices and Peside. Y + V + 1.(Y) + Rr None 2 STD Ye, Rr Store Indices and Peside. (Y + 0) - Rr None 2 ST Z, Rr Store Indices and Peside. (Z) - Rr None 2 ST Z, Rr Store Indices and Peside. (Z) - Rr None 2 ST Z, Rr Store Indices and Peside. (Z + 21, (Z) - Rr None 2 ST X, Rr Store Indices and Peside. (Z + 21, (Z) - Rr None 2 ST X, Rr Store Indice and Peside. (Z + 0) - Rr None 2 ST X, Rr Store Indice and Peside. Rd - D None 1 DT Log Peside None Store Indice and Peside. PG 2 ST None Pace Store None Store None 2 ST None Pace Store None Store None 2 | ST | Y, Rr | Store Indirect | (Y) ← Rr | None | 2 |
| stT ·/·, fr Size indirect and Prodec. Y - · Y - Y, (P, FR None 2 STD Y - 9, Rr Size indirect with Displacement (Z) - FR None 2 ST Z, Rr Store indirect and Prodec. (Z) - Rr, Z + Z + 1 None 2 ST Z, Rr Store indirect and Prodec. (Z + 2, 1/2) - FR None 2 STD Z-4, Rr Store indirect and Prodec. (Z + 2, 1/2) - FR None 2 STD Z-4, Rr Store indirect in SRM (Q + Rr None 2 STD Z-4, Rr Store indirect in SRM (Q + Rr None 1 DUT P, Rr Out Port Rd + P None 1 DUT P, Rr Out Port Rd + P None 2 Store indirect in Stark STA Cott C-Rr None 2 DUT AD UT-TESTTEXTEXTURE Store indirect in Stark | ST | Y+. Rr | Store Indirect and Post-inc. | $(Y) \leftarrow \text{Rr. } Y \leftarrow Y + 1$ | None | 2 |
| STDY+q.R/Som indrage with Displacement $(Y+q)+R'$ None2STZ,R'Stom indraged Post-inc. $(Q)+Rr, Z+Z+1$ None2STZ,R'Stom indraged Post-inc. $(Q)+Rr, Z+Z+1$ None2STZ,R'Stom indraged Post-inc. $Z+2+1, Z+R'$ None2STZ,R'Stom indraged Post-inc. $Z+2+1, Z+R'$ None2STZ,R'Stom indraged Post-inc. $Z+2+1, Z+R'$ None2STZ,R'Stom indraged with Displacement $(Z+q)+Rr$ None2STSk,R'Stom Divert to SRAM $(M+R')-Rr'$ None3INRd,PIn PostRd -PNone1PUSHR'Poul PostRd -PNone2DUTP,R'Oul PortPor Register from StackSTACK + RrNone2POPRdPop Register from StackRd +PNone2SBIP,bGest Bit In O Register $VO(P)+-1$ None2CBIP,bGest Bit In O Register $VO(P)+-1$ None2CBIP,bGest Bit In OR Register $VO(P)+-1$ None2CBIP,bGest Bit RighRd(p)+-Rd(p), Rd(p), C, O(p)ZC.NV1RGRdIdogest Shift RighRd(p)+-Rd(p), Rd(p), C, Cd(p)ZC.NV1RGRdRata Lattrongh CarryRd(p)+-Rd(p), Rd(p), C, Cd(p)ZC.NV1RGRdAdata Lattrongh CarryRd(p)+-Rd | ST | -Y. Rr | Store Indirect and Pre-dec. | $Y \leftarrow Y - 1$. (Y) $\leftarrow Rr$ | None | 2 |
| ST Z, Rr Size Indired (2) - Fir None 2 ST Z, Rr Size Indired and Post-inc. (2) - Fir None 2 ST Z, Rr Size Indired and Post-inc. (2) - Fir None 2 ST Z, Rr Size Indired and Post-inc. (2 + 2) + (2) - Fir None 2 ST X, Rr Size Indired and Post-inc. (2 + q) - Fir None 2 ST K, Rr Size Indired and Post-inc. (2 + q) - Fir None 2 UPM Laad Program Menory R0 - C) None 3 3 ST K, Rr Dur Port Rd - P None 1 OUT P, Ir Out Port Rd - P None 2 PDP Rd Pog Register on Stack Rd - P None 2 IPG Bat Bit In/O Register IO(P,b) - 1 None 2 Start Antametic Start Distributing Comparation Stack Rd - C, Ra(ru) + Ra(Rio), C - Ra(ru) 2 C,N,V 1 | STD | Y+a Rr | Store Indirect with Displacement | $(Y + q) \leftarrow Br$ | None | 2 |
| STZ., RrSize Indices and Postin. $(2) - R; Z - 2 + 1$ None2STZ, RrSize Indices and Pie-dec. $Z + Z - 1, (Z) - Rr.$ None2STDZ + q, RrSize Indices and Pie-dec. $Z + Z - 1, (Z) - Rr.$ None2STSk, RrSize Directo SRAM(k) - RrNone2STSk, RrSize Directo SRAM(k) - RrNone3INRd, PIn PortRd - PNone1OUTP, RrOut PertRd - PNone1PUSHRrPush Register on StackSTACK - RrNone2DOPRdPop Register on StackSTACK - RrNone2StatP. bStatin In O RegisterVOP Pi + 1None2CBIP. bClare Bit In O RegisterVOP Pi + 1None2CBIRdLogical Stift RighRd(n) + C Rd(n) R(d) + 0Xc.NV1LSKRdLogical Stift RighRd(n) + C Rd(n) R(d) + 0Zc.NV1RCNRdRdia Logical Stift RightRd(n) + C Rd(n) R(d) + 0.Zc.NV1RCRRdAttimutor Shift RightRd(n) + C Rd(n) R(d) + 0.Zc.NV1RCRRdAttimutor Shift RightRd(n) + C Rd(n) R(d) + C Rd(n)Zc.NV1RCRRdAttimutor Shift RightRd(n) + C Rd(n) R(d) + C Rd(n)Zc.NV1RCRRdAttimutor Shift RightRd(n) - C Rd(n) R(d) - C Rd(n)Zc.NV1RCR <td>ST</td> <td>7 Rr</td> <td>Store Indirect</td> <td>$(7) \leftarrow \text{Rr}$</td> <td>None</td> <td>2</td> | ST | 7 Rr | Store Indirect | $(7) \leftarrow \text{Rr}$ | None | 2 |
| STZ, RStore Indirect and Pre-dec.Dec. V = 1, 2 + 1, (2) \leftarrow RNone2STOZ-q, RrStore Indirect with Displacement $(2 + q) \leftarrow Rr$ None2STOL, RrStore Indirect with Displacement $(2 + q) \leftarrow Rr$ None2LPMLaad Program MemoryR0 \leftarrow (2)None3INRd, PIn PortRd \leftarrow PNone1OUTP, RrOut PortRd \leftarrow PNone1PUSHRrPuck Register on StackSTACK \leftarrow RrNone2POPRdPog Register forn StackRd \leftarrow STACK \leftarrow RrNone2PDFRdLogical Shit LaftRd $+ p$ None2SBIP, bClear Bit In 10 Register10(P b) \leftarrow 1None2CBIP, bClear Bit In 10 Register10(P b) \leftarrow 0None2LSLRdLogical Shit LaftRd(p) \leftarrow Clear (n, n) (\leftarrow Rd(n) C \leftarrow Rd(| ST | Z, 10 | Store Indirect and Post-inc | $(Z) \leftarrow \operatorname{Rr} Z \leftarrow Z + 1$ | None | 2 |
| Dr. Dr. No. Dr. No. Dr. No. L STD Z-Hq, Rr. Store Indirect with Displacement (Z + q) + Rr None 2 STS K, Rr. Store Direct to SRAM (I) + Rr None 2 IN Rd, P In Port R0 + C[2] None 3 IN Rd, P In Port Rd + P None 1 DUT P, Rr Out Port P + Rr None 2 DUT P, Rr Out Port Rd + STACK None 2 DUT P, Rr Out Port Rd + STACK None 2 DUT Rd Push Register from Stack STGCK-Rr None 2 BC Rd Calgeal Brit In/O Register VO(P.b) - 0 None 2 Stat Rd Logical Shit Right Rd(P, C, Rd(P, H) Rd(P, R) C) Z, C, N,V 1 RA Rotate Left through Carry Rd(P) - C, Rd(P, H, Rd(P, H), C, Rd(P, H) Rd(P, H) C, Rd(P, H) Rd(P, H | ST | -7 Rr | Store Indirect and Pre-dec | $7 \leftarrow 7 - 1$ (7) $\leftarrow \text{Br}$ | None | 2 |
| Dirg Dirk Dirk <thdirk< th=""> Dirk Dirk <thd< td=""><td>STD</td><td>-2, IN</td><td>Store Indirect with Displacement</td><td>$(7 \pm \alpha) \leftarrow \text{Pr}$</td><td>None</td><td>2</td></thd<></thdirk<> | STD | -2, IN | Store Indirect with Displacement | $(7 \pm \alpha) \leftarrow \text{Pr}$ | None | 2 |
| D10 IAM Didd Default works D10 Note 2 LPM Load Default works R0 + (2) Note 3 IN Rd, P In Port Rd + P None 1 OUT P, R' Out Port P + Rr None 1 PUSH R' Push Register on Stack SRd + STACK None 2 DYD Rd Poge Register from Stack SRd + STACK None 2 SIL Rd Logical Shift Left Rd(P) None 2 SIL Rd Logical Shift Left Rd(P) + CR(P) Z.C.N.V 1 ROR Rd Rotate Left through Carry Rd(7) + C.Rd(r), Rd(7) + O. Z.C.N.V 1 SMAP Rd Autimetic Shift Right Rd(7) + C.Rd(r), Rd(7), Rd(7), C + Rd(7) Z.C.N.V 1 SMAP Rd Autimetic Shift Right Rd(7) + C.Rd(r), Rd(7, A), Rd(7, A), Rd(7, A) X.N.V 1 SMAP Rd Autimetic Shift Right Rd(6) SREG(6) 1 < | STD | ZTQ, Ki | Store Direct to SPAM | $(2 + q) \leftarrow Rr$ | None | 2 |
| Dram Rd, P In Ordig Rote P In Ordig Rote P In Ordig S OUT P, R Out Pot Pot Rd Port None 1 OUT R, R Push Register on Stack STACK + Rr None 2 POP Rd Pop Register from Stack Rd + STACK None 2 PDF Rd Pop Register from Stack Rd + STACK + Rr None 2 BIT AND BIT-TEST///EXTOND Stack + STACK None 2 1 Star Rd Logical Shit Left None 2 1 LSL Rd Logical Shit Left Rd(n+1) (- Rd(n), Rd(n) (- 0) 2.C.N.V 1 LSR Rd Rd Rottas Right Prough Carry Rd(n) (- C.Rd(n), Hed(n+1) C.Rd(n) 2.C.N.V 1 SNAP Rd Antimetic Shift Right Rd(n) (- C.Rd(n) + Rd(n-1) C.Rd(n) 2.C.N.V 1 SNAP Rd Shift Start Fort Register OT Rd(n) (- C.Rd(n+1) C.Rd(n) 2.C.N.V 1 SNAP < | | κ, ιχι | Lood Brogram Mamory | $(\mathbf{k}) \leftarrow (\mathbf{k})$ | None | 2 |
| N No.P No.P No.P No.P No.P Ind.P OUT P, Rr Out Port Push Register on Stack STACK No.P 2 PUSP Rd Pop Register fon Stack StACK No.P 2 BIT P.D StB thin I/O Register Rd + STACK No.P 2 StB P, D StB thin I/O Register I/O(P,b) - 0 No.P 2 CBL P, D Get Btin I/O Register I/O(P,b) - 0 No.P 2 LSR Rd Logical Shit Right Rd(P, C, Rd(n), Rd(R) - 0 Z.C.N.V 1 ROR Rd Rotate Eth through Carry Rd(I) - C, Rd(n), Rd(R), C - Rd(I) Z.C.N.V 1 ROR Rd Attimetic Shit Right Rd(I) - Rd(I-I), I, C - Rd(I) Z.C.N.V 1 SWAP Rd Samp Shit Stack Rd(I) - Rd(I-I), I, C - Rd(I) Z.C.N.V 1 SWAP Rd Attimetic Shit Right Rd(I) - Rd(I-I), I, C - Rd(I) Z.C.N.V 1 SWAP Rd | | | | $RU \leftarrow (Z)$ | None | 3 |
| OUT P, N OUT P01 PA PA PA PUSH Rr Pop Register from Stack STACK Fr/r None 2 POP Rd Pop Register from Stack STACK Fr/r None 2 SBI PL Set Bit In I/O Register U/O(P.b) ← 1 None 2 CBI P, b Set Bit In I/O Register U/O(P.b) ← 0 None 2 CBI Rd Logical Shif Right Rd(n+1), Rd(n), Rd(0) ← 0 ZC.N.V 1 LSL Rd Logical Shif Right Rd(0) ← C, Rd(n+1), Rd(n, C + Cd(n) - Z, C.N.V 1 ROR Rd Rotae Right through Carry Rd(0) ← C, Rd(n+1), Rd(n+1), C + Rd(0, C + Rd(7, 2), C + Rd(1, 2), C + | | Ru, P | III POIL | | None | 1 |
| POP Rd Putan Register form Stack SIAk K-Fd? None 2 POP Rd Pop Register from Stack Rd + STACK None 2 BIT AND BIT-TEXTUSTORS Set Bit in I/O Register UO(P.b) + 0 None 2 CBI P, b Clear Bit in I/O Register UO(P.b) + 0 None 2 LSL Rd Logical Shift Left Rd(n+1) + Rd(n), Rd(0) + 0 XC.N.V 11 LSR Rd Logical Shift Left Rd(n) + CR(n), Rd(n) + Rd(n), C + Rd(n) ZC.N.V 11 ROR Rd Rotate Left through Carry Rd(n) + CR(n) + Rd(n), C + Rd(n) ZC.N.V 11 SRR Rd Swap Nobles Rd(n) + Rd(n+1), Rd(7, -0 ZC.N.V 11 SWAP Rd Swap Nobles Rd(n) + CR(n), Rd(7, -4, Rd(3, -0) None 11 SWAP Rd Swap Nobles SREG(s) + 0 SREG(s) 11 SWAP Rd Swap Nobles SREG(s) + 0 SREG(s) 11 SWAP Rd Swap Nobles <t< td=""><td>DUDU</td><td>P, KI</td><td></td><td></td><td>None</td><td>1</td></t<> | DUDU | P, KI | | | None | 1 |
| PDPRdPop Register non stackRd + SIACRNone2BIT AND BITTER INSTRUCTIONSSBIP, bSet Bit in VO Register $VO(P,b) \leftarrow 1$ None2CBIP, bClear Bit in VO Register $VO(P,b) \leftarrow 0$ None2LSLRdLogical Shift Left $R(n) \leftarrow R(n), R(0) \leftarrow 0$ Z.C.N.V1LSRRdLogical Shift Right $R(n) \leftarrow R(n+1), Rd(n), C \leftarrow Rd(n)$ Z.C.N.V1ROLRdRotate Left through Carry $R(0) \leftarrow Rd(n+1), Rd(n), C \leftarrow Rd(n)$ Z.C.N.V1RORRdRd Antimetic Shift Right $R(0) \leftarrow Rd(n+1), Rd(n, C \leftarrow Rd(n)$ Z.C.N.V1ASRRdAntimetic Shift Right $R(0) \leftarrow Rd(n+1), Rd(n, C \leftarrow Rd(n)$ Z.C.N.V1SWAPRdSwap Nibbies $R(d,0) \leftarrow Rd(n-4), Rd(n, A) \leftarrow Rd(0)$ None1SWAPRdSwap NibbiesRd(2,0) \leftarrow Rd(n, A), Rd(n, A) \leftarrow Rd(0)None1BSTsFlag SetSREG(s) -1 SREC(s)1BCLRsFlag ClearSREC(s) -1 SREC(s)1BCLsFlag ClearSREC(s) -1 None1BLDRd, bBit Load from To Register DT $\leftarrow R(n)$ None1SECSet CarryC $\leftarrow 1$ C11CLSet CarryC $\leftarrow 0$ Z11SECSet Set Signed Test FlagX $\leftarrow 0$ N $\leftarrow 0$ N $\leftarrow 1$ SETGlobal Interrupt EnableI $\leftarrow 1$ I $\leftarrow 1$ 11< | PUSH | Rr | Push Register on Stack | | None | 2 |
| Bit ARD Bit-less instructions Pib Set Bit in I/O Register I/O(P,b) ← 1 None 2 CB1 P, b Clear Bit in I/O Register I/O(P,b) ← 0 None 2 LSL Rd Logical Shift Left Rd(n) (+ Rd(n), Rd(0) ← 0 Z, N, V 1 LSR Rd Logical Shift Left Rd(n) (+ Rd(n), Rd(0) (- 0 Z, N, V 1 ROL Rd Rotate Left through Carry Rd(0) (- C, Rd(n+1), Rd(n), C + Rd(n) Z, N, V 1 ROR Rd Rotate Left through Carry Rd(1) (- Rd(n+1), Rd(-A), C + Rd(n)) Z, N, V 1 SWAP Rd Rdate Mathmedic Shift Right Rd(1) (- Rd(n+1), R - Rd(n)) None 1 SWAP Rd Samp Nibbles SREG(s) (-1 SREG(s) 1 1 BST s Flag Clear SREG(s) (-1 None 1 1 BLD Rd, b Bt Store from Register to T T (- R(t)) None 1 1 SEC Set Carry C (-1 None 1 1 <t< td=""><td></td><td>Rd</td><td>Pop Register from Stack</td><td></td><td>None</td><td>2</td></t<> | | Rd | Pop Register from Stack | | None | 2 |
| SBI P, b Set Et in //O Register IO(P,b) + 1 None 2 CBI P, b Clear Bitin //O Register IO(P,b) + 0 None 2 LSL Rd Logical Shift Right Rd(n) + Rd(n), Rd(n) + 0 ZC, N,V 1 LSR Rd Rotare Left through Carry Rd(n) + C, Rd(n+1), C + Rd(n) ZC, N,V 1 ROR Rd Rotare Left through Carry Rd(n) + C, Rd(n+1), C + Rd(n) ZC, N,V 1 RASR Rd Rotare Right through Carry Rd(n) + Rd(n+1), C + Rd(n) ZC, N,V 1 SWAP Rd Swap Nibbles Rd(n) + Rd(n+1), C + Rd(n) ZC, N,V 1 SWAP Rd Swap Nibbles Rd(n) + Rd(n+1), C + Rd(n) ZC, N,V 1 SWAP Rd Swap Nibbles Rd(n) + Rd(n+1), C + Rd(n) None 1 SWAP Rd Swap Nibbles Rd(n) + Rd(n+1) + Rd(n) None 1 SWAP Rd Swap Nibbles Rd(n) + Rd(n+1), C + Rd(n) None 1 SWAP Rd | BIT AND BIT-TEST | INSTRUCTIONS | | | | - |
| CBI P, b Clear Bin I/O Register I/O(P,b) (~ 0) None 2 LSL Rd Logical Shift Right Rd(n) (~ Rd(n+1), Rd(7), C, 0) Z.C.N.V 1 LSR Rd Logical Shift Right Rd(n) (~ Rd(n+1), Rd(7), C, 0) Z.C.N.V 1 ROL Rd Rotate Right through Carry Rd(7) (~ C, Rd(n+1), Rd(7), C, + Rd(1), C, + Rd(2)) Z.C.N.V 1 ROR Rd Rotate Right through Carry Rd(7) (~ Rd(n+1), Rd(7), C, + Rd(1), C, + Rd(2)) Z.C.N.V 1 ARM Rd a trithmetic Shift Right Rd(3, O) (~ Rd(n+1), Rd(7,4) + Rd(3,0)) None 1 SWAP Rd Swap Nibbles SREG(s) - 0 SREG(s) 1 BST S Flag Set SREG(s) - 0 SREG(s) 1 BCL s Flag Set Carry C + 1 None 1 SEC I/r Set Carry C + 0 I 1 CLC Set Carry C + 0 I 1 1 SEL Set Regative Flag N + 0 1 <td>SBI</td> <td>P, b</td> <td>Set Bit in I/O Register</td> <td>I/O(P,b) ← 1</td> <td>None</td> <td>2</td> | SBI | P, b | Set Bit in I/O Register | I/O(P,b) ← 1 | None | 2 |
| LSL Rd Logical Shift Reft Rd(n+1) (~ Rd(n), Rd(0) (~ 0) Z.C.N.V 11 LSR Rd Logical Shift Reft Rd(n) (~ C, Rd(n, H) (~ Rd(n, H), C (~ Rd(n)), C (~ Rd(n, H), C (~ Rd(n)), C (~ R | CBI | P, b | Clear Bit in I/O Register | I/O(P,b) ← 0 | None | 2 |
| LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(r) \leftarrow 0Z.C.N.V1ROLRdRotate Left through CarryRd(0) \leftarrow C. Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)Z.C.N.V1RORRdAnthmetic Shift RightRd(r) \leftarrow C. Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)Z.C.N.V1ASRRdAnthmetic Shift RightRd(r) \leftarrow Rd(n+1), n = 0.6Z.C.N.V1SWAPRdSwap NbblesRd(3.0) \leftarrow Rd(r-1), n = 0.6Z.C.N.V1BSTSFlag SetSREG(s) \leftarrow 1SREG(s)1BCRsFlag SetSREG(s) \leftarrow 1SREG(s)1BCRsFlag ClearSREG(s) \leftarrow 0SREG(s)1BLDRd, bBit Store from Register DTTRd(b) \leftarrow TNone1SECLear CarryC \leftarrow 1C111SECSet GarryC \leftarrow 0C \leftarrow 0111CLCClear CarryC \leftarrow 0C \leftarrow 0111CL2Clear Carge FlagN \leftarrow 0N111SEZGlobal Interrupt EnableI \leftarrow 1I111CL2Clear Signed Test FlagS \leftarrow 1S111SEZGlobal Interrupt EnableI \leftarrow 1I \leftarrow 1I11CL2Clear Signed Test FlagS \leftarrow 1S \leftarrow 1S111SESI \leftarrow 1Global Interrupt EnableI \leftarrow 1I \leftarrow 1I \leftarrow 1I | LSL | Rd | Logical Shift Left | $Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$ | Z,C,N,V | 1 |
| ROLRdRotate Hightrough CarryRd(0) \leftarrow Rd(n+1), \leftarrow Rd(n), \leftarrow Rd(n | LSR | Rd | Logical Shift Right | $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$ | Z,C,N,V | 1 |
| RoRRdRotate Right through CarryRd($\gamma \leftarrow C, Rd(n \leftarrow Rd(n+1), C \leftarrow Rd(n)), C \leftarrow Rd(n), C \leftarrow Rd($ | ROL | Rd | Rotate Left through Carry | $Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$ | Z,C,N,V | 1 |
| ASRRdAnthmetic Shift RightRd(n \leftarrow Rd(n \perp), $n = 0.6$ Z, C, N, V1SWAPRdSwap NibblesRd(3.0) \leftarrow Rd(74), Rd(74) \leftarrow Rd(3.0)None1SBETsFlag SetSREG(s) \leftarrow 1SREG(s)1BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)1BSTRt, bBit Store from Register to TT \leftarrow Rt(b)T1BLDRd, bBit Load from To Register to TT \leftarrow Rt(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C11CLCSet CarryC \leftarrow 0C1CLNSet Negative FlagN \leftarrow 1N1SEXSet Negative FlagX \leftarrow 0X1CL2Clear Negative FlagZ \leftarrow 021SEZSet Negative FlagZ \leftarrow 021CL2Set Segre FlagZ \leftarrow 021CL2Global Interrupt EnableI \leftarrow 111CL3Global Interrupt EnableI \leftarrow 011SESSet Signed Test FlagS \leftarrow 0S11SESClear Signed Test FlagS \leftarrow 0S11SEVSet Two's Complement OverflowV \leftarrow 0N11SEVSet Two's Complement OverflowV \leftarrow 0N11SETSet Two's Complement OverflowV \leftarrow 0111SETSet Two's Complement OverflowV \leftarrow 01< | ROR | Rd | Rotate Right through Carry | $Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ | Z,C,N,V | 1 |
| SWAPRdSwap NibblesRd(3.0) \leftarrow Rd(7.4), Rd(7.4), eCR(3.0)None1BSTsFlag SetSREG(s)SREG(s)1BCLRsFlag ClearSREG(s)01BSTRr, bBit Store from Register to TT \leftarrow R(b)T1BLDRd, bBit Load from To RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C1SENClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CL2Clear CarryC \leftarrow 011SEZSet Set Set Set GaryZ \leftarrow 1Z1CL2Clear CarryZ \leftarrow 1Z1SEZGlobal Interrupt FlagZ \leftarrow 1Z1CL2Clear Set Set FlagZ \leftarrow 1Z1SEIGlobal Interrupt EnableI \leftarrow 111CL3Global Interrupt DisableI \leftarrow 0S \leftarrow 11SESGSet Set Set Set Set Set Set Set Set Set | ASR | Rd | Arithmetic Shift Right | $Rd(n) \leftarrow Rd(n+1), n = 06$ | Z,C,N,V | 1 |
| BSETsFlag SetSREG(s)SREG(s)SREG(s)1BCLRsFlag ClearSREG(s)SREG(s)SREG(s)1BCDRr, bBit Store from Register to T \top - Rr(b)T1BLDRd, bBit Load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C1CLCCelar CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1None1CLNClear Asgative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1SEZGlobal Interrupt FlagI \leftarrow 111CL2Clear Zero FlagZ \leftarrow 0Z1SEZGlobal Interrupt DisableI \leftarrow 111CL3Global Interrupt DisableI \leftarrow 011SESSet Signed Test FlagS \leftarrow 1S1SESSet Signed Test FlagS \leftarrow 0S1SESSet Signed Test FlagS \leftarrow 0S1CLYGlobal Interrupt DisableI \leftarrow 0I1SETSet Signed Test FlagS \leftarrow 0S1SETSet Signed Test FlagS \leftarrow 0S1SETSet Tim SREGT \leftarrow 0I1SETSet Tim SREGT \leftarrow 0I1SETSet Half-carry Flag in SREGH \leftarrow 0H \leftarrow 0SETSet Parlidorary Flag in SREGH \leftarrow 0H \leftarrow 0< | SWAP | Rd | Swap Nibbles | $Rd(30) \leftarrow Rd(74), Rd(74) \leftarrow Rd(30)$ | None | 1 |
| BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BSTRr, bBit Store from Register to TT1BLDRd, bBit Load from Tto Register to T $T \leftarrow Rr(b)$ None1SEC \Box Set Carry $C \leftarrow 1$ None1CLCImage: Set Carry $C \leftarrow 0$ C1CL0Image: Set Negative Flag $N \leftarrow 1$ None1SETImage: Set Negative Flag $N \leftarrow 0$ N1SEZImage: Set Zero Flag $Z \leftarrow 1$ Z1CLZImage: Set Zero Flag $Z \leftarrow 0$ Z1SEIImage: Set Global Interrupt Enable $I \leftarrow 1$ 11SESImage: Set Signed Test Flag $S \leftarrow 0$ S1SESImage: Set Signed Test Flag $S \leftarrow 0$ S1SESImage: Set Signed Test Flag $S \leftarrow 0$ S1SETImage: Set Two's Complement Overflow $V \leftarrow 0$ 11SETImage: Set Two's Complement Overflow $V \leftarrow 0$ 11SETSet Two's Complement Overflow $V \leftarrow 0$ 11SETImage: Set Two's Complement Overflow $V \leftarrow 0$ 11SETSet Two's Complement Overflow $V \leftarrow 0$ 11SET | BSET | S | Flag Set | $SREG(s) \leftarrow 1$ | SREG(s) | 1 |
| BSTRr, bBit Store from Register to TT \leftarrow (k(b)T \leftarrow (k(b)TNone1BLDRd, bBit Load from To RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C1CLCCClear CarryC \leftarrow 01SENSet Negative FlagN \leftarrow 1N1CLNClear Megative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1CLZClear Set Zero FlagZ \leftarrow 0Z1SEIGlobal Interrupt EnableI \leftarrow 0I1CLIGlobal Interrupt EnableI \leftarrow 0I1SESSet Signed Test FlagS \leftarrow 1S1SESISet Signed Test FlagS \leftarrow 0S1CLVIClear Two's Complement OverflowV \leftarrow 011SETSet Tim SREGT \leftarrow 0I11CLTClear Tim SREGT \leftarrow 0I11SETSet Tim SREGT \leftarrow 0I11CLTSet Half-carry Flag in SREGT \leftarrow 0I1SETSet Half-carry Flag in SREGH \leftarrow 0H1CLTClear All-carry Flag in SREGH \leftarrow 0H1SETSet Half-carry Flag in SREGH \leftarrow 0H1SETSet Half-carry Flag in SREGH \leftarrow 0H1SETSet Half-carry Flag in SREGH \leftarrow 0H1 | BCLR | S | Flag Clear | $SREG(s) \leftarrow 0$ | SREG(s) | 1 |
| BLDRd, bBit Load from T to RegisterRd(b) \leftarrow TNone1SECISet CarryC \leftarrow 1C1CLCIClear CarryC \leftarrow 0C1SENISet Negative FlagN \leftarrow 1N1SELIClear Negative FlagN \leftarrow 0N1SEZISet Zero FlagZ \leftarrow 1Z1CLZIClear Zero FlagZ \leftarrow 0Z1SEIGlobal Interrupt EnableI \leftarrow 111CLIGlobal Interrupt DiableI \leftarrow 0I1SESIGlobal Interrupt EnableS \leftarrow 1S1CLSIGlobal Interrupt DiableS \leftarrow 0S1CLSISet Signed Test FlagS \leftarrow 0S1SEVIClear Signed Test FlagS \leftarrow 0S1CLVISet Two's Complement OverflowV \leftarrow 0V1SETSet Set Signed Test FlagT \leftarrow 0T1CLTClear Tin SREGT \leftarrow 0T1CLTClear Tin SREGT \leftarrow 0T1CLTSet Half-carry Flag in SREGH \leftarrow 0H1NOPINo OperationNone1SLEPSleepSleep Clift descr. for Sleep function)None1SLEPNoted descr.Sleep Clift descr. for Sleep function)None1SLEPNoted descr.Sleep Clift descr. for Sleep | BST | Rr, b | Bit Store from Register to T | $T \leftarrow Rr(b)$ | Т | 1 |
| SECSet Carry $C \leftarrow 1$ C 1CLCClear Carry $C \leftarrow 0$ C1SENSet Negative Flag $N \leftarrow 1$ N1SENClear Negative Flag $N \leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 1$ Z1CL2Clear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11CL1Global Interrupt Enable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1CL3Set Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V1CLVClear Two's Complement Overflow $V \leftarrow 0$ 11SETSet Tai NSRG $T \leftarrow 0$ T11CLTClear Two's Complement Overflow $V \leftarrow 0$ 11SETSet Half-carry Flag in SREG $T \leftarrow 0$ T11CL4Set Half-carry Flag in SREG $H \leftarrow 0$ H11NOPNo OperationNone111SEEPSleepSleep(see specific decr. for Sleep function)None1SEEPWDR decade BacetImage and family functionNone1MDRMDRMDRImage functionNone1 | BLD | Rd, b | Bit Load from T to Register | $Rd(b) \leftarrow T$ | None | 1 |
| CLCClear Carry $C \leftarrow 0$ C1SENSet Negative Flag $N \leftarrow 1$ N1CLNClear Negative Flag $N \leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 1$ Z1CLZClear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11CLIGlobal Interrupt Enable $I \leftarrow 0$ I1CLSSet Signed Test Flag $S \leftarrow 1$ S1SESClear Signed Test Flag $S \leftarrow 0$ S1CLSSet Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V1CLYClear Two's Complement Overflow $V \leftarrow 0$ V1SETSet Tin SREG $T \leftarrow 0$ T1SEHSet Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNone11SEEPSleepSleep Sectific descr. for Sleep function)None1SLEPSleepMORMORMarceMarce | SEC | | Set Carry | C ← 1 | С | 1 |
| SENSet Negative Flag $N \leftarrow 1$ N1CLNClear Negative Flag $N \leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 1$ Z1CL2Clear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11CL1Global Interrupt Disable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1SEVClear Signed Test Flag $S \leftarrow 0$ S1CLVClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V 1CLVClear Tin SREG $T \leftarrow 1$ T1SETSet Tin SREG $T \leftarrow 0$ T1SEHSet Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNooperationNone1SLEPSleep(see specific descr. for Sleep function)None1SLEPSleep(rate amorific datart for WIDE firmer)None1 | CLC | | Clear Carry | C ← 0 | С | 1 |
| CLNClear Negative Flag $N \leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 1$ Z1CLZClear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ I1CLIGlobal Interrupt Enable $I \leftarrow 1$ I1CLIGlobal Interrupt Enable $I \leftarrow 0$ I1CLIGlobal Interrupt Disable $I \leftarrow 0$ I1CLIGlobal Interrupt Disable $S \leftarrow 1$ S1CLSSet Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V1CLVClear Two's Complement Overflow $V \leftarrow 0$ V1SETSet Tin SREG $T \leftarrow 0$ T11SETSet Tin SREG $T \leftarrow 0$ T11SEHSet Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNo OperationNone1SLEEPSleepSleep(see specific descr. for Sleep function)None4 | SEN | | Set Negative Flag | N ← 1 | N | 1 |
| SEZSet Zero Flag $Z \leftarrow 1$ Z 1CLZClear Zero Flag $Z \leftarrow 0$ Z 1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11CLIGlobal Interrupt Disable $I \leftarrow 0$ 11SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V1CLVClear Two's Complement Overflow $V \leftarrow 0$ V1SETSet T in SREG $T \leftarrow 1$ T1CLTClear T in SREG $T \leftarrow 0$ T1SEHSet Half-carry Flag in SREG $H \leftarrow 1$ H1CLHClear Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationSeeperfit descr. for Sleep function)None1SLEEPSleepVaradeda ParetVaradeda ParetNone1 | CLN | | Clear Negative Flag | N ← 0 | Ν | 1 |
| CLZClear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ I1CLIGlobal Interrupt Disable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V1CLVClear Two's Complement Overflow $V \leftarrow 0$ V1SETSet T in SREG $T \leftarrow 1$ T1CLTClear T in SREG $T \leftarrow 0$ T1SEHSet Half-carry Flag in SREG $H \leftarrow 1$ H1CLHClear Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNo OperationNone1SLEEPSleepSleep(see specific descr. for Sleep function)None1 | SEZ | | Set Zero Flag | Z | Z | 1 |
| SEIGlobal Interrupt Enable $1 \leftarrow 1$ 1 1 CLIGlobal Interrupt Disable $1 \leftarrow 0$ 1 1 SESSet Signed Test Flag $S \leftarrow 1$ S 1 CLSClear Signed Test Flag $S \leftarrow 0$ S 1 SEVSet Two's Complement Overflow $V \leftarrow 1$ V 1 CLVClear Two's Complement Overflow $V \leftarrow 0$ V 1 SETSet T in SREG $T \leftarrow 1$ T 1 CLTClear T in SREG $T \leftarrow 0$ T 1 SEHSet Half-carry Flag in SREG $H \leftarrow 1$ H 1 CLHClear Half-carry Flag in SREG $H \leftarrow 0$ H 1 NOPNo OperationNo Operation N N 1 SLEEPSleepSleep(see specific descr. for Sleep function)None 1 | CLZ | | Clear Zero Flag | Z ← 0 | Z | 1 |
| CLIGlobal Interrupt Disable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Two's Complement Overflow $V \leftarrow 1$ V1CLVClear Two's Complement Overflow $V \leftarrow 0$ V1SETSet T in SREG $T \leftarrow 1$ T1CLTClear T in SREG $T \leftarrow 0$ T1SEHSet Half-carry Flag in SREG $H \leftarrow 1$ H1CLHClear Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNone11SLEEPSleep(see specific descr. for Sleep function)None1 | SEI | | Global Interrupt Enable | l ← 1 | 1 | 1 |
| SESSet Signed Test FlagS \leftarrow 1S1CLSClear Signed Test FlagS \leftarrow 0S1SEVSet Two's Complement OverflowV \leftarrow 1V1CLVClear Two's Complement OverflowV \leftarrow 0V1SETSet T in SREGT \leftarrow 1T1CLTClear T in SREGT \leftarrow 0T1SEHSet Half-carry Flag in SREGH \leftarrow 1H1CLHClear Half-carry Flag in SREGH \leftarrow 0None1SLEEPSleepSleep(see specific descr. for Sleep function)None1 | CLI | | Global Interrupt Disable | 1 ← 0 | 1 | 1 |
| CLSClear Signed Test FlagS \leftarrow 0S1SEVSet Two's Complement OverflowV \leftarrow 1V1CLVClear Two's Complement OverflowV \leftarrow 0V1SETSet T in SREGT \leftarrow 1T1CLTClear T in SREGT \leftarrow 0T1SEHSet Half-carry Flag in SREGH \leftarrow 1H1CLHClear Half-carry Flag in SREGH \leftarrow 0H1SLEEPSleepSleep(see specific descr. for Sleep function)None1 | SES | | Set Signed Test Flag | S ← 1 | S | 1 |
| SEVSet Two's Complement Overflow $V \leftarrow 1$ V 1CLVClear Two's Complement Overflow $V \leftarrow 0$ V 1SETSet T in SREG $T \leftarrow 1$ T 1CLTClear T in SREG $T \leftarrow 0$ T 1SEHSet Half-carry Flag in SREG $H \leftarrow 1$ H1CLHClear Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNone11SLEEPSleep(see specific descr. for Sleep function)None1 | CLS | | Clear Signed Test Flag | S ← 0 | S | 1 |
| CLVClear Two's Complement OverflowV $\leftarrow 0$ V1SETSet T in SREGT $\leftarrow 1$ T1CLTClear T in SREGT $\leftarrow 0$ T1SEHSet Half-carry Flag in SREGH $\leftarrow 1$ H1CLHClear Half-carry Flag in SREGH $\leftarrow 0$ H1NOPNo OperationNone1SLEEPSleep(see specific descr. for Sleep function)None1 | SEV | 1 | Set Two's Complement Overflow | V ← 1 | V | 1 |
| SETSet T in SREG $T \leftarrow 1$ T1CLTClear T in SREG $T \leftarrow 0$ T1SEHSet Half-carry Flag in SREG $H \leftarrow 1$ H1CLHClear Half-carry Flag in SREG $H \leftarrow 0$ H1NOPNo OperationNone1SLEEPSleep(see specific descr. for Sleep function)None1 | CLV | 1 | Clear Two's Complement Overflow | V ← 0 | V | 1 |
| CLTClear T in SREGT $\leftarrow 0$ T1SEHSet Half-carry Flag in SREGH $\leftarrow 1$ H1CLHClear Half-carry Flag in SREGH $\leftarrow 0$ H1NOPNo OperationNone1SLEEPSleep(see specific descr. for Sleep function)None1 | SET | | Set T in SREG | T ← 1 | T | 1 |
| SEH Set Half-carry Flag in SREG H ← 1 H CLH Clear Half-carry Flag in SREG H ← 0 H NOP No Operation None 1 SLEEP Sleep (see specific descr. for Sleep function) None 1 | CLT | | Clear T in SREG | T ← 0 | т. Т | 1 |
| CLH Clear Half-carry Flag in SREG H ← 0 H 1 NOP No Operation None 1 SLEEP Sleep (see specific descr. for Sleep function) None 1 | SEH | | Set Half-carry Flag in SREG | | н | 1 |
| NOP No operation Noe 1 SLEEP Sleep (see specific descr. for Sleep function) None 1 | CLH | | Clear Half-carry Flag in SREG | | н | 1 |
| Non None 1 SLEEP Sleep (see specific descr. for Sleep function) None 1 WDP Watchdag Roopt (see specific descr. for Sleep function) None 1 | NOP | + | | | None | 1 |
| Other Geophysical (see specific description) None 1 WIDP Watchdag Point (see specific description) None 4 | SLEEP | | Sleen | (see specific descr. for Sloop function) | None | 1 |
| | WDP | | Watchdog Pasat | (see specific descr. for WDD/#mor) | None | 1 |





3. Brown-out Detection Level

The Brown-out Detection level can increase when there is heavy I/O-activity on the device. The increase can be significant when some of the I/O pins are driving heavy loads.

Problem Fix/Workaround

Select a V_{CC} well above the Brown-out Detection level.

Avoid loading I/O ports with high capacitive or resistive loads.

2. Serial Programming at Voltages below 2.9V

At voltages below 2.9V, serial programming might fail.

Problem Fix/Workaround

Keep V_{CC} at 2.9V or higher during In-System Programming.

1. UART Loses Synchronization if RXD Line is Low when UART Receive is Disabled

The UART will detect a UART start bit and start reception even if the UART is not enabled. If this occurs, the first byte after reenabling the UART will be corrupted.

Problem Fix/Workaround

Make sure that the RX line is high at start-up and when the UART is disabled. An external RS-232 level converter keeps the line high during start-up.