



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
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	5
Program Memory Size	2KB (1K x 16)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.209", 5.30mm Width)
Supplier Device Package	8-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at90ls2343-4si

Email: info@E-XFL.COM

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



Description

The AT90S/LS2323 and AT90S/LS2343 are low-power, CMOS, 8-bit microcontrollers based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S2323/2343 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general-purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.



Figure 1. The AT90S/LS2343 Block Diagram



AT90S/LS2323/2343

Pin Descriptions AT90S/LS2343

vcc	Supply voltage pin.
GND	Ground pin.
Port B (PB4PB0)	Port B is a 5-bit bi-directional I/O port with internal pull-up resistors. The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low, will source current if the pull-up resistors are activated.
	Port B also serves the functions of various special features.
	Port pins can provide internal pull-up resistors (selected for each bit). The Port B pins are tri-stated when a reset condition becomes active.
RESET	Reset input. An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 50 ns will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.
CLOCK	Clock signal input in external clock mode.
Clock Options	
Crystal Oscillator	The AT90S/LS2323 contains an inverting amplifier that can be configured for use as an On-chip oscillator, as shown in Figure 3. XTAL1 and XTAL2 are input and output respectively. Either a quartz crystal or a ceramic resonator may be used. It is recommended that the AT90S/LS2343 be used if an external clock source is used, since this gives an extra I/O pin.

Figure 3. Oscillator Connection



External Clock

The AT90S/LS2343 can be clocked by an external clock signal, as shown in Figure 4, or by the On-chip RC oscillator. This RC oscillator runs at a nominal frequency of 1 MHz ($V_{CC} = 5V$). A fuse bit (RCEN) in the Flash memory selects the On-chip RC oscillator as the clock source when programmed ("0"). The AT90S/LS2343 is shipped with this bit programmed. The AT90S/LS2343 is recommended if an external clock source is used, because this gives an extra I/O pin.

The AT90S/LS2323 can be clocked by an external clock as well, as shown in Figure 4. No fuse bit selects the clock source for AT90S/LS2323.



AT90S/LS2323/2343

I/O Direct

Figure 12. I/O Direct Addressing



Operand address is contained in six bits of the instruction word. n is the destination or source register address.

Data Direct

Figure 13. Direct Data Addressing



A 16-bit data address is contained in the 16 LSBs of a 2-word instruction. Rd/Rr specify the destination or source register.





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



Data Indirect with Displacement

AMEL

Memory Access and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the System Clock Ø, directly generated from the external clock signal applied to the CLOCK pin. No internal clock division is used.

Figure 21. shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks and functions per power unit.





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





The internal data SRAM access is performed in two System Clock cycles as described in Figure 23.

• Bit 0 – C: Carry Flag

The carry flag C indicates a carry in an arithmetical or logical operation. See the Instruction Set description for detailed information.

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

Stack Pointer – SPL An 8-bit register at I/O address \$3D (\$5D) forms the stack pointer of the AT90S2323/2343. Eight bits are used to address the 128 bytes of SRAM in locations \$60 - \$DF.



The Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt stacks are located. This stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above \$60. The Stack Pointer is decremented by 1 when data is pushed onto the Stack with the PUSH instruction and it is decremented by 2 when an address is pushed onto the stack with subroutine calls and interrupts. The Stack Pointer is incremented by 1 when data is popped from the stack with the POP instruction and it is incremented by 2 when an address is popped from the stack with return from subroutine RET or return from interrupt RETI.

Reset and InterruptThe AT90S2323/2343 provides two interrupt sources. These interrupts and the separate
reset vector each have a separate program vector in the program memory space. Both
interrupts are assigned individual enable bits that must be set (one) together with the
I-bit in the Status Register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 3. The list also determines the priority levels of the interrupts. The lower the address, the higher the priority level. RESET has the highest priority, and next is INT0 (the External Interrupt Request 0), etc.

Table 3.	Reset and	Interrupt	Vectors
----------	-----------	-----------	---------

Vector No.	Program Address	Source	Interrupt Definition
1	\$000	RESET	Hardware Pin, Power-on Reset and Watchdog Reset
2	\$001	INT0	External Interrupt Request 0
3	\$002	TIMER0, OVF0	Timer/Counter0 Overflow



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 CPU clock cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT} . Refer to page 30 for details on operation of the Watchdog.

Figure 28. Watchdog Reset during Operation



MCU Status Register – MCUSR

The MCU Status Register provides information on which reset source caused an MCU reset.



• Bits 7..2 - Res: Reserved Bits

These bits are reserved bits in the AT90S2323/2343 and always read as zero.

• Bit 1 – EXTRF: External Reset Flag

After a Power-on Reset, this bit is undefined (X). It will be set by an External Reset. A Watchdog Reset will leave this bit unchanged.

Bit 0 – PORF: Power-on Reset Flag

This bit is set by a Power-on Reset. A Watchdog Reset or an External Reset will leave this bit unchanged.

To summarize, Table 7 shows the value of these two bits after the three modes of reset.

Table 7. PORF and EXTRF Values after Reset

Reset Source	PORF	EXTRF
Power-on Reset	1	Undefined
External Reset	Unchanged	1
Watchdog Reset	Unchanged	Unchanged

To make use of these bits to identify a reset condition, the user software should clear both the PORF and EXTRF bits as early as possible in the program. Checking the PORF and EXTRF values is done before the bits are cleared. If the bit is cleared before an External or Watchdog Reset occurs, the source of reset can be found by using the following truth table, Table 8.





Table 8. Reset Source Identification

PORF	EXTRF	Reset Source
0	0	Watchdog Reset
0	1	External Reset
1	0	Power-on Reset
1	1	Power-on Reset

Interrupt Handling

The AT90S2323/2343 has two 8-bit interrupt mask control registers; GIMSK (General Interrupt Mask register) and TIMSK (Timer/Counter Interrupt Mask register).

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software can set (one) the I-bit to enable nested interrupts. The I-bit is set (one) when a Return from Interrupt instruction (RETI) is executed.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logical "1" to the flag bit position(s) to be cleared. If an interrupt condition occurs when the corresponding interrupt enable bit is cleared (zero), the interrupt flag will be set and remembered until the interrupt is enabled or the flag is cleared by software.

If one or more interrupt conditions occur when the global interrupt enable bit is cleared (zero), the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set (one) and will be executed by order of priority.

Note that external level interrupt does not have a flag and will only be remembered for as long as the interrupt condition is active.

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

General Interrupt Mask Register – GIMSK



• Bit 7 – Res: Reserved Bit

This bit is a reserved bit in the AT90S2323/2343 and always reads as zero.

• Bit 6 – INT0: External Interrupt Request 0 Enable

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from program memory address \$001. See also "External Interrupts."

Bits 5..0 – Res: Reserved Bits

These bits are reserved bits in the AT90S2323/2343 and always read as zero.



EEPROM Read/Write Access

C The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4 ms, depending on the V_{CC} voltages. A self-timing function, however, lets the user software detect when the next byte can be written.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed. When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed.

EEPROM Address Register – EEAR

Bit	7	6	5	4	3	2	1	0	_
\$1E (\$3E)	-	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R	R/W							
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - Res: Reserved Bit

This bit is a reserved bit in the AT90S2323/2343 and will always read as zero.

• Bit 6..0 – EEAR6..0: EEPROM Address

The EEPROM Address Register (EEAR6..0) specifies the EEPROM address in the 128-byte EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 127.

EEPROM Data Register – EEDR



• Bits 7..0 – EEDR7..0: EEPROM Data

For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

AIMEL

Prevent EEPROM Corruption

During periods of low V_{CC}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using the EEPROM and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

EEPROM data corruption can easily be avoided by following these design recommendations (one is sufficient):

- Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This is best done by an external low V_{CC} Reset Protection circuit, often referred to as a Brown-out Detector (BOD). Please refer to application note AVR 180 for design considerations regarding power-on reset and low-voltage detection.
- Keep the AVR core in Power-down Sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the EEPROM registers from unintentional writes.
- 3. Store constants in Flash memory if the ability to change memory contents from software is not required. Flash memory cannot be updated by the CPU and will not be subject to corruption.



- 3. \$002: \$03 (indicates AT90S/LS2343 when signature byte \$001 is \$91)
- Note: When both Lock bits are programmed (Lock mode 3), the signature bytes cannot be read in the low-voltage Serial mode. Reading the signature bytes will return: \$00, \$01 and \$02.

Programming the Flash
and EEPROMAtmel's AT90S2323/2343 offers 2K bytes of In-System Programmable Flash program
memory and 128 bytes of EEPROM data memory.

The AT90S2323/2343 is shipped with the On-chip Flash program and EEPROM data memory arrays in the erased state (i.e., contents = \$FF) and ready to be programmed.

The device supports a high-voltage (12V) Serial Programming mode and a low-voltage Serial Programming mode. The +12V is used for programming enable only and no current of significance is drawn by this pin. The low-voltage Serial Programming mode provides a convenient way to download program and data into the device inside the user's system.

The program and EEPROM memory arrays in the AT90S2323/2343 are programmed byte-by-byte in either programming modes. For the EEPROM, an auto-erase cycle is provided within the self-timed write instruction in the low-voltage Serial Programming mode.

During programming, the supply voltage must be in accordance with Table 15.

Part	Low-voltage Serial Programming	High-voltage Serial Programming
AT90S2323	4.0 - 6.0V	4.5 - 5.5V
AT90LS2323	2.7 - 6.0V	4.5 - 5.5V
AT90S2323	4.0 - 6.0V	4.5 - 5.5V
AT90LS2323	2.7 - 6.0V	4.5 - 5.5V

 Table 15.
 Supply Voltage during Programming

High-voltage Serial Programming

This section describes how to program and verify Flash program memory, EEPROM data memory, Lock bits and Fuse bits in the AT90S2323/2343.

Figure 32. High-voltage Serial Programming



Instruction Format						
Instruction		Instr.1	Instr.2	Instr.3	Instr.4	Operation Remarks
Read Fuse and Lock Bits (AT90S/ LS2323)	PB0 PB1 PB2	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxx	0_0000_0000_00 0_0111_1100_00 <i>1_2</i> Sxx_xxRx_xx		Reading <i>1</i> , <i>2</i> , S , R = "0" means the Fuse/Lock bit is programmed.
Read Fuse and Lock Bits (AT90S/ LS2343)	PB0 PB1 PB2	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxx	0_0000_0000_00 0_0111_1100_00 <i>1_2</i> \$xx_xxRx_xx		Reading <i>1</i> , <i>2</i> , S , R = "0" means the Fuse/Lock bit is programmed.
Read Signature Bytes	PB0 PB1 PB2	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxx	0_0000_00 bb _00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxx	0_0000_0000_00 0_0110_1100_00 o_0000_000 x_xx	Repeat Instr.2 - Instr.4 for each signature byte address.

Table 16. High-voltage Serial Programming Instruction Set (Continued)
--	---

Note: **a** = address high bits

b = address low bits

i = data in

o = data out

x = don't care

1 = Lock Bit1

2 = Lock Bit2

 $\textbf{F} = FSTRT \ Fuse$

 $\boldsymbol{\mathsf{R}}=\mathsf{RCEN}\;\mathsf{Fuse}$

 $\bm{S} = \text{SPIEN Fuse}$





High-voltage Serial Programming Characteristics

Figure 34. High-voltage Serial Programming Timing



Table 17. High-voltage Serial Programming Characteristics, $T_A = 25^{\circ}C \pm 10^{\circ}$, $V_{CC} = 5.0V \pm 10^{\circ}$ (unless otherwise noted)

Symbol	Parameter	Min	Тур	Max	Units
t _{SHSL}	SCI (XTAL1/PB3) Pulse Width High	100.0			ns
t _{SLSH}	SCI (XTAL1/PB3) Pulse Width Low	100.0			ns
t _{IVSH}	SDI (PB0), SII (PB1) Valid to SCI (XTAL1/PB3) High	50.0			ns
t _{SHIX}	SDI (PB0), SII (PB1) Hold after SCI (XTAL1/PB3) High	50.0			ns
t _{SHOV}	SCI (XTAL1/PB3) High to SDO (PB2) Valid	10.0	16.0	32.0	ns
t _{WLWH_CE}	Wait after Instr.3 for Chip Erase	5.0	10.0	15.0	ms
t _{WLWH_PFB}	Wait after Instr.3 for Write Fuse Bits	1.0	1.5	1.8	ms

Low-voltage Serial Downloading

Both the program and data memory arrays can be programmed using the serial SPI bus while RESET is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output) (see Figure 35). After RESET is set low, the Programming Enable instruction needs to be executed first before program/erase instructions can be executed.

Figure 35. Low-voltage Serial Programming and Verify





Low-voltage Serial Programming Characteristics

Figure 37. Low-voltage Serial Programming Timing



Table 20. Low-voltage Serial Programming Characteristics, $T_A = -40^{\circ}C$ to 85°C, $V_{CC} = 2.7 - 6.0V$ (unless otherwise noted)

Symbol	Parameter	Min	Тур	Max	Units
1/t _{CLCL}	Oscillator Frequency ($V_{CC} = 2.7 - 4.0V$)	0		4.0	MHz
t _{CLCL}	Oscillator Period (V _{CC} = 2.7 - 4.0V)	250.0			ns
1/t _{CLCL}	Oscillator Frequency ($V_{CC} = 4.0 - 6.0V$)	0		8.0	MHz
t _{CLCL}	Oscillator Period (V _{CC} = 4.0 - 6.0V)	125.0			ns
t _{SHSL}	SCK Pulse Width High	2.0 t _{CLCL}			ns
t _{SLSH}	SCK Pulse Width Low	2.0 t _{CLCL}			ns
t _{ovsH}	MOSI Setup to SCK High	t _{CLCL}			ns
t _{SHOX}	MOSI Hold after SCK High	2.0 t _{CLCL}			ns
t _{SLIV}	SCK Low to MISO Valid	10.0	16.0	32.0	ns

Table 21. Minimum Wait Delay after the Chip Erase Instruction

Symbol	3.2V	3.6V	4.0V	5.0V
t _{WD_ERASE}	18 ms	14 ms	12 ms	8 ms

Table 22. Minimum Wait Delay after Writing a Flash or EEPROM Location

Symbol	3.2V	3.6V	4.0V	5.0V
t _{WD_PROG}	9 ms	7 ms	6 ms	4 ms

Typical Characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as $C_L \bullet V_{CC} \bullet f$ where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.









Figure 44. Idle Supply Current vs. V_{CC}



























Instruction Set Summary

NHTMERT CARD. Code: No. ADD: R1. fr: Add hon Seguints R1 fr1fr2. Z.G.N.H. 1 ADC R0. fr: Add hon Seguints R1 fr1fr2. Z.G.N.H. 1 ADC R0. fr: Add hon Seguints R2 R0 Rr. Z.G.N.H. 1 SUB R0. fr: Subtox Containton Register R2 R0 Rr. Z.G.N.H. 1 SUB R0. fr: Subtox Containton Register R2 R0 Rr. Z.G.N.H. 1 SSC R3. fr: Subtox Containton Register R2 R0 Rr. Z.G.N.H. 1 SSC R3. fr: Subtox of Macro To Register R2 R0 Rr. Z.G.N.H. 1 SSC R3. fr: Subtox of Macro To Register R2 R0 K C. Z.G.N.H. 1 SSC R3. fr: Subtox Ontainton Register R2 R0 K C. Z.G.N.H. 1 SSC R3. fr: Subtox Ontainton Register R2 R0 K C. Z.G.N.H. 1 ASC Subtox Ontainton Register R2 R0 K C. Z.A.N.H.	Mnemonic	Operands	Description	Operation	Flags	# Clocks
ADC Rd, Fr. Add the Registra Pdx-fd + fr/s ZCRVH 1 ADC Rd, Fr. Add with Cary The Registra Pdx-fd + Fr + C ZCRVS 2 ADW Rd, Fr. Solitable Too Registra Pdx-fd - File File K. ZCRVS 2 SUB Rd, Fr. Solitable Too Registra Pdx-fd - File File K. ZCRVM 1 SUB Rd, K. Solitable Introduction Registra Pdx-fd - File File K. ZCRVM 1 SUB Rd, K. Solitable Introduction Registra Pdx-fd - File File File K. ZCRVM 1 SSI Rd, K. Solitable Introduction Registra Pdx-fd - File - C ZCRVM 1 ADD Rd, Fr. Solitable Registra and contant Pdx-fd + File File ZAV 1 ADD Rd, Fr. Logical OB Registra and contant Pdx-fd + File File ZAV 1 COM Rd Trois Complement Pdx-fd + File File ZAV 1 COM Rd + File File SCRVM SCRVM 1 1 COM <td>ARITHMETIC AND</td> <td>OGIC INSTRUCTION</td> <td>S</td> <td></td> <td>- 3 -</td> <td></td>	ARITHMETIC AND	OGIC INSTRUCTION	S		- 3 -	
AOC Bit Br Add sum Gray Tan Registers Bit Ar. Add Turnediate Word Derivation Convertion Convertion <td>ADD</td> <td>Rd. Br</td> <td>Add Two Begisters</td> <td>Bd ← Bd + Br</td> <td>Z.C.N.V.H</td> <td>1</td>	ADD	Rd. Br	Add Two Begisters	Bd ← Bd + Br	Z.C.N.V.H	1
ADW Path Addition Too Segment Path ZOAVS 2 SUB BS, Hr. Subtract Too Segment Ret – Ret – R. ZOAVS 1 SUB BS, Kr. Subtract Too Segment Ret – Ret – R. ZOAVS 2 SUB BS, Kr. Subtract Too Segment Ret –	ADC	Rd, Rr	Add with Carry Two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
SUB P6 Ar. Submot Construction Register P6 - A ZCAV H 1 SBW P6 K Submat Construction Register P6 - A ZCAV S 2 SBW P6 K Submat Construction Register P6 - A P6 - K ZCAV S 2 SBC P6 K Submat Construction Register P6 - A A CAV H 1 SBC P6 K Submat Construction Register P6 - A CAV H 1 SBC P6 K Submat Construction Register P6 - A CAV H 1 SBC P6 K Logical AD Register and Construct P6 - A ZAV X 1 ADR P6 K Logical AD Register and Construct P6 - A ZAV X 1 COM P6 K Logical AD Register and Construct P6 - A ZAV 1 1 COM P6 K Construct Construct P6 - A ZAV 1 1 COM P6 K Construct Construct P6 - A ZAV 1 1 <t< td=""><td>ADIW</td><td>Rdl, K</td><td>Add Immediate to Word</td><td>$Rdh:Rdl \leftarrow Rdh:Rdl + K$</td><td>Z,C,N,V,S</td><td>2</td></t<>	ADIW	Rdl, K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
By Bit Bat/s Subtract Internation Program Pat – Pat – K. ZOAV JA 1 SBW Bat/s Subtract Internation Provided Internation Provided Patholics ZOAV JA 2 SBC Bat/s Subtract Internation Provided Patholics Rei – Rei – Rei – Rei – C. ZOAV JA 1 SBC Bat/s Subtract Internation Provided Patholics Rei – Rei – Rei – Rei – C. ZOAV JA 1 AND Bat/s Logical AND Register and Containt Rei –	SUB	Rd, Rr	Subtract Two Registers	Rd ← Rd – Rr	Z,C,N,V,H	1
SBWRb,RSolution throadiate from WordRb,RHRb,RHL,CN,VSZ,N,VSZSBCRb,RSolution throadiate from Rug.Rd-Rd-R-CZ,N,VH1SBCRb,RLogical AND RegistersRd-Rd-Rd-Rd-CZ,N,VK1ANDRb,RLogical AND Register and ConstantRd-Rd-Rd-Rd-Rd-Rd-Rd-Rd-Rd-Rd-Rd-Rd-Rd-R	SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SRCRA,KSolver, Yon Quere into Panel A, P	SBIW	Rdl, K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
SBC1Pd.ASederat with Carry Constant term PeanPd.APd.AC. AV.M1ANDPd.ALogical AND Register and ConstantPd.APd.APd.AN.M.M1ANDPd.ALogical CA Register and ConstantPd.APd.APd.AN.M.M1CPIPd.ALogical CA Register and ConstantPd.APd.APd.AN.M.M1CPIPd.ALogical CA Register and ConstantPd.APd.APd.AN.M.M1CDRPd.ADescomperationPd.APd.ASch.N.M11COMPdOris ComplementPd.APd.AC.N.N.M11SBRPd.ASet Biglish RegisterPd.APd.AC.N.N.M11SBRPd.ACore Pd.N.M. PogeterPd.APd.AC.N.N.M11NGCPd.ACore Pd.APd.APd.APd.A2.N.N.M11SBRPd.ACore Pd.APd.APd.APd.A2.N.N.M11SBRPd.ATote Zono A MinasPd.APd.APd.A11 </td <td>SBC</td> <td>Rd, Rr</td> <td>Subtract with Carry Two Registers</td> <td>$Rd \gets Rd - Rr - C$</td> <td>Z,C,N,V,H</td> <td>1</td>	SBC	Rd, Rr	Subtract with Carry Two Registers	$Rd \gets Rd - Rr - C$	Z,C,N,V,H	1
ANDPAG.Logent AND RegistersPAG + PA + PAZ.N.V11ORPAG.KLogent AND Registers and ConstantPAG + PA + FAZ.N.V11ORPAG.KLogent OR Registers and ConstantPAG + PA + FAZ.N.V11EORPAG.KLogent OR Registers and ConstantPAG + PAG + FAZ.N.V11EORPAG.KLogent OR Registers and ConstantPAG + PAG + FAZ.N.V11EORPAGConstantPAG + PAG + FAZ.N.V11REGPAGTwo ConstantPAG + PAG + FAZ.N.V11REGPAGTwo ConstantPAG + PAG + FAZ.N.V11REGPAGConstantPAG + PAG + FAZ.N.V11CRAPAGDecomentPAG + PAG + FAZ.N.V11DECPAGDecomentPAG + PAG + PAG + PAGZ.N.V11CRAPAGDecomentPAG + PAG + PAG + PAGZ.N.V11CRAPAGDecomentPAG + PAG +	SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \gets Rd - K - C$	Z,C,N,V,H	1
ANDRd, FLLogial AND Register and ConstantRd - Rd v KZ.N.V1ORRd, RLLogial OR Register and ConstantRd - Rd v KZ.N.V1ORIRd, KLLogial OR Register and ConstantRd - Rd v KZ.N.V1CMRd, NLLogial OR Register and ConstantRd - Rd v KZ.N.V1CMRdDev ComplementRd - Rd v KZ.N.V1CMRdDev ComplementRd - Rd v KZ.N.V1SRRd KSelfkin RegisterRd - Rd v KZ.N.V1SRRd KCanar Biol RegisterRd - Rd v Rd FF - RdZ.N.V1SRRd KCanar Biol RegisterRd - Rd v Rd FF - RdZ.N.V1SRRd KCanar Biol RegisterRd - Rd v Rd FF - RdZ.N.V1INCRd KCanar Biol RegisterRd - Rd v Rd Pd - Rd v Rd vX.N.V1SRRd KCanar Biol RegisterRd - Rd v Rd vRd vRd v1SRRd KCanar Biol RegisterRd v Rd vRd vRd v11SRRd KCanar Biol RegisterRd vRd vRd v11<	AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd ullet Rr$	Z,N,V	1
OR Rol, Ru Logical OR Register and Contain Rid + Ray K ZNV 1 DOR Ro, Ru Logical OR Register and Contain Rid + Ray An ZNV 1 EOR Rod Toy Complement Rid + Ray An ZONV 1 NEG Rid Toy Complement Rid + SPT - Ray An ZONV 1 SBR Rid K Dest Stript In Register Rid + Rid + Ray ZNV 1 DEG Rid Destembly In Register Rid + Rid + Rid ZNV 1 DEG Rid Destembly In Register Rid + Rid + Rid ZNV 1 DEG Rid Destembly In Register Rid + Rid + Rid ZNV 1 DEG Rid Destembly In Rid Rid ZNV 1 1 DEG Rid Destembly In Rid	ANDI	Rd, K	Logical AND Register and Constant	$Rd \gets Rd \bullet K$	Z,N,V	1
OH Rd, K Logial OR Register Rd - Rd VK ZNV 1 EOR Rd, K Cody Conglement Rd - Rd Pr. ZNV 1 CM Rd Ons Conglement Rd + S0 - Rd ZCAVV 1 SR Rd SR Pr. Rd SR Pr. Rd ZCAVV 1 SR Rd SR Pr. Rd SR Pr. Rd ZCAVV 1 SR SR Pr. Rd SR Pr. Rd SR Pr. Rd ZAVV 1 SR Case RB(s) Register Rd + Rd + Rd + Rd - Rd ZAV 1 DEC Rd SR Segeenet Rd + Rd + Rd + Rd ZAV 1 DEC Rd Set Register Rd + Rd + Rd + Rd ZAV 1 DER Rd Set Register Rd + Rd	OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \lor Rr$	Z,N,V	1
EOR PA, Pr. Exclusive OR Registers Rd - R - R - R - R - S - R - R	ORI	Rd, K	Logical OR Register and Constant	$Rd \gets Rd \lor K$	Z,N,V	1
COM Pid Ore's Complement Pid + SPT - Pid ZCNV H 1 NEG Rd Two's Complement Rd + SD - Rd / CXNV H 1 SRR Rd, K Stability in Register Rd + Rd + Rd + ST ZNV 1 SRR Rd, K Stability in Register Rd + Rd + Rd + ST ZNV 1 INC Rd / Case Register Rd + Rd	EOR	Rd, Rr	Exclusive OR Registers	$Rd \gets Rd \oplus Rr$	Z,N,V	1
NEG. Rd Two's Complement Rd - Rd vK Z/S/W 1 SBR Rd K Set B(g) in Register Rd - Rd vK ZNV 1 CBR Rd K Oser B(g) in Register Rd - Rd + GFF - K) ZNV 1 DEC Rd Decrement Rd - Rd + GFF - K) ZNV 1 DEC Rd Decrement Rd - Rd + GF ZNV 1 DEC Rd Cear Register Rd - Rd + Rd ZNV 1 SER Rd Cear Register Rd - Fd + Rd None 2 BRACH INSTRUTOTOTO FC - FC + k + 1 None 2 None 3 IDALL K Rediave Jamp D(2) PC - FC + k + 1 None 4 IDALL Indrace Marking D(2) PC - FC + k + 1 None 4 IDALL Indrace Marking D(2) PC - FC + k + 1 None 4 IDALL Indrace Marking D(2) PC - FC + k + 1 None 1 IDALL Indrace Marking D(2) PC - FC	СОМ	Rd	One's Complement	$Rd \gets \$FF - Rd$	Z,C,N,V	1
SBR Pit, K Set Bit(s) in Register Pit C = NV K ZNV 1 CBR Pit A Cheer Bit(s) in Register Pit C = NV K XNV 1 INC Pit A Decrement Pit C = Pit A ZNV 1 DEC Pit A Decrement Pit C = Pit A ZNV 1 DEC Pit A Decrement Pit C = Pit A ZNV 1 SER Rd Set Register Pit C = Pit A Rd = Rd	NEG	Rd	Two's Complement	Rd ← \$00 – Rd	Z,C,N,V,H	1
GRN Bt. K Course Netgo in Register Rd + Rd + Rd + FLA, ZNV 1 INC Rd Decrement Rd + Rd + 1 ZNV 1 DEC Rd Decrement Rd + Rd + Rd - 1 ZNV 1 CLR Rd Clear Register Rd + Rd + Rd - Rd ZNV 1 StR Rd Star Register Rd + Rd + Rd - Rd ZNV 1 StR Rd Star Register Rd + StF None 1 StR Rd None Rd - StF None 2 RMMP k Relative Subroutine Call PC + PC + k + 1 None 3 IDALL Indirect Jump to (2) PC + Z None 3 1 ICALL Indirect Jump to (2) PC + STACK None 1 4 CPL Rd, Rr Compare, Stor Mitional If (Rd = R) PC + C2 + X + 1 None 1/2/3 CPL Rd, Rr Compare with Cary Rd - Rr - C ZNV C, H 1 CPL	SBR	Rd, K	Set Bit(s) in Register	$Rd \gets Rd \lor K$	Z,N,V	1
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	CBR	Rd, K	Clear Bit(s) in Register	$Rd \gets Rd \bullet (\$FF - K)$	Z,N,V	1
DECRdDecommentRd + Rd -1Z,N V1TSTRdTostor or MinusRd + Rel rRdZ,N V1CLRRdClear RegisterRd + Rel rRdZ,N V1SERRdSen RegisterRd + SFNone1SERRdSen RegisterRd + SFNone2RAMPkRelative JumpPC + PC + k + 1None2RAMPkRelative Jump 1C2PC + C-ZNone3RCALLkRelative Subroutine CallPC + PC + k + 1None3ICALLindirect Calls (Z)PC + C-ZNone3ICALLindirect Calls (Z)PC + STACKNone4RETIindirect Calls (Z)PC + STACKNone4RETIcompare, Skip if Equaliff Rd = Rip PC + PC + 2 or 3None1/2/3CPRd, RrCompare Nsip if Equaliff Rd = Rip PC + PC + 2 or 3None1/2/3CPRd, RrCompare Nsip if Equaliff Rhp = DC + PC + 2 or 3None1/2/3CPRd, RrCompare Nsip if Equaliff Rhp = DC + PC + 2 or 3None1/2/3SBSRb bSkip if Bit in Rogister Clearedif (Rhp) = DC + PC + 2 or 3None1/2/3SBSP, bSkip if Bit in Rogister Setif (Rhp) = DC + PC + 2 or 3None1/2/3SBSP, bSkip if Bit in Rogister Setif (Rhp) = DC + PC + 2 or 3None1/2/3SBSP, bSkip if Bit in Rogister Cleared	INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
TST Rd Clear Degister Rd - Rd + Rd Z,NV 1 CLR Rd Gear Degister Rd + Rd Rd Z,NV 1 SER Rd SetRegister Rd + SFF None 1 BRANCHINSTRUCTIONE Indirect Jump to [2] PC +- PC + k + 1 None 2 LMP k Relative Jump (2] PC +- 2C + k + 1 None 3 ICALL k Relative Jump to [2] PC +- 2C + k + 1 None 3 ICALL k Relative Jump to [2] PC +- 2C + k + 1 None 4 ICALL k Relative Jump to [2] PC +- 2C + k + 1 None 4 ICALL k Relative Jump to [2] PC +- 2C + k + 1 None 4 ICALL k Relative Jump to [2] PC + 2C + 2C + 2C + 1 None 4 ICALL k Relative Jump to [2] PC + 5TACK I 4 CP Rd, Rr Compare Register with Immediate Rd + Rr - C ZNVC.H 1	DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
CLR Rd Clear Register Rd – Rd ® Rd Z,N,V 1 SER Rd Set Register Rd – SFF. None 1 BRACH INSTRUCTORS None 2 RAMP k Relative Jump to (2) PC + PC + k + 1 None 2 RCALL k Relative Subroutine Call PC - PC + k + 1 None 3 RET Indirect Jump to (2) PC - Z None 4 CPL Indirect Jump to (2) PC - Z None 4 CPL Indirect Jump to (2) PC - Z None 4 CALL Indirect Jump telum PC - STACK None 4 CPL Rd, Rr Compare Register with Carry Rd - Rr - C Z.N.V.C.H 1 CPC Rd, Rr Compare Register with Carry Rd - K Z.N.V.C.H 1 CPC Rd, K Compare Register Set If (Rh(b) = 0) PC - PC + 2 or 3 None 1/2/3 SBRS Rr, b Skip if Bin hegester Geard <t< td=""><td>TST</td><td>Rd</td><td>Test for Zero or Minus</td><td>$Rd \gets Rd \bullet Rd$</td><td>Z,N,V</td><td>1</td></t<>	TST	Rd	Test for Zero or Minus	$Rd \gets Rd \bullet Rd$	Z,N,V	1
SER Rd SetRegister Rd = SFF None 1 BRANCH INSTRUCTORS BRANCH INSTRUCTORS None 2 LIMP k Relative Jump (z) PC + 2C None 2 LIMP indirect Jump to (z) PC + 2C None 3 RCALL k Relative Jump (Z) PC + 2C + k + 1 None 3 ICALL k Relative Jump to (Z) PC + 2C + k + 1 None 3 ICALL k Relative Jump to (Z) PC - 2T ACK None 4 ICAL indirect Call to (Z) PC - 2T ACK None 1 4 CP Rd, Rr Compare, Skip IE gual if (Rd = Ri) PC + PC + 2 or 3 None 1/23 CP Rd, Rr Compare Nito Carry Rd - K Z_NV, C, H 1 CP Rd, Rr Compare Nito Carry Rd - K Z_NV, C, H 1 SBR Rr, b Skip IB II In Register Clared if (Rd(b) = 1) PC + PC + 2 or 3 None 1/2/3 SBR	CLR	Rd	Clear Register	$Rd \gets Rd \oplus Rd$	Z,N,V	1
BRANCH INSTRUCTIONS IMMP k Relative Jump to (2) PC +- PC + k + 1 None 2 LMP Indirect Jump to (2) PC +- Z None 3 RCALL k Relative Subroutine Call PC PC + k + 1 None 3 ICALL Indirect Call to (2) PC Z None 3 ICALL Indirect Call to (2) PC Z None 3 ICALL Indirect Call to (2) PC Z None 4 ICALL Interrupt Return PC STACK None 4 RETI Subroutine Return PC STACK I 4 CPPE 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 SBRC Rr, b Skip If Bit in Register Cleared If (Rh(b) = 1) PC - PC + 2 or 3 None 1/2/3 SBRS Rr, b Skip If Bit in NO Register Cleared If (RD) = 1) PC - PC + 2 or 3 None 1/2/3	SER	Rd	Set Register	Rd ← \$FF	None	1
PLMP k Relative Jump PC - PC + k + 1 None 2 IMP Indirect Jump to [2] PC + Z None 3 ICALL k Relative Subroutine Call PC + -Z None 3 ICALL - Indirect Call to [2] PC + -Z None 3 ICALL - Subroutine Return PC - STACK None 4 RET Subroutine Return PC - STACK I 4 CP Rd, Rr Compare Subj IE pual If (Rd = R) PC - C 2 or 3 None 1/2/3 CPC Rd, Rr Compare Magister with Immediate Rd - K ZN,VC,H 1 CPC Rd, Rr Compare Register with Immediate Rd - K ZN,VC,H 1 SBRS Rr, b Skip If Bit Rin Register Cleared If (R(b) = 1) PC + PC + 2 or 3 None 1/2/3 SBRS P, b Skip If Bit In IO Register Cleared If (R(b) = 0) PC + PC + 2 or 3 None 1/2/3 SBRS P, b Skip If Bit In IO Register Gleared If (RD(b) = 0	BRANCH INSTRUC	TIONS				
IMP Indirect Jump to (2) PC Z None 2 RGALL k Relative Subroutine Call PC PC + k + 1 None 3 ICALL Indirect Call to (2) PC FC + k + 1 None 3 RET Subroutine Return PC STACK None 4 CPS Rd, Rr Compare Skip if Equal if (Pd - Rn (PC - PC + 2 or 3) None 1/2/3 CP Rd, Rr Compare Skip if Equal if (Pd - Rn (PC - PC + 2 or 3) None 1/2/3 CPC Rd, Rr Compare Min Carry Rd - Rr - C Z/N/C,H 1 CPC Rd, Rr Compare Min Carry Rd - Rr - C Z/N/C,H 1 CPC Rd, K Compare Min Carry Rd - Rr - C Z/N/C,H 1 SBRC Rr, b Skip if Bit in Register Cleared if (Rrtip) = 0) PC - PC + 2 or 3 None 1/2/3 SBRS Rr, b Skip if Bit in Register is Set if (R(h) = 1) PC - PC + 2 or 3 None 1/2/3 SBRS P, b Skip if Bit in Register is Set	RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
RCALL k Relative Subroutine Call PC C + k + 1 None 3 ICALL Indired Call to (2) PC Z None 3 ICAL Indired Call to (2) PC STACK None 4 RET Interrupt Return PC STACK I 4 CP Rd, Rr Compare, Skip if Equal If (Rd = Rr) PC - PC + 2 or 3 None 1/2/3 CP Rd, Rr Compare Might Equal If (Rd = Rr) PC - PC + 2 or 3 None 1/2/3 CP Rd, Rr Compare Mighter with Immediate Rd - K Z.N.V.C.H 1 CP Rd, Rr Compare Mighter with Immediate Rd - K Z.N.V.C.H 1 SBRS Rr, b Skip if Bl in Register is Set If (Rt)b = 1) PC + PC + 2 or 3 None 1/2/3 SBRS P, b Skip if Bl in I/O Register Is Set If (Rb(b) = 1) PC + PC + 2 or 3 None 1/2/3 SBRS P, b Skip if Bl in I/O Register Is Set If (Rb(b) = 1) PC + PC + 2 or 3 None 1/2/3 SBRS P, b <td< td=""><td>IJMP</td><td></td><td>Indirect Jump to (Z)</td><td>$PC \leftarrow Z$</td><td>None</td><td>2</td></td<>	IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
ICALLIndirect Call to (2) PC $- Z$ None3RETSubouline ReturnPC $- STACK$ None4RET1Interrupt ReturnPC $- STACK$ I4CPSRd, RrCompare, Skip I Equalif (Rd = R) PC $-$ PC $+ 2 \sigma$ 3None1/23CPRd, RrCompare Skip I Equalif (Rd = R) PC $-$ PC $+ 2 \sigma$ 3None1/23CPRd, RrCompare Negister with CarryRd $-$ Rr $-$ CZ.N.V.C.H1CP1Rd, KCompare Register with ImmediateRd $-$ KrZ.N.V.C.H1CP1Rd, KCompare Register with ImmediateRd $-$ KrZ.N.V.C.H1SBRCRr, bSkip I Bt In Register I Clearedif (Rt(b) = 0) PC $+$ PC $+ 2 \sigma$ 3None1/23SBRSRr, bSkip I Bt In In Register I Setif (Rt(b) = 0) PC $+$ PC $+ 2 \sigma$ 3None1/23SBISP, bSkip I Bt In In Register I Setif (Rt(b) = 1) PC $+$ PC $+ 2 \sigma$ 3None1/23SBRSRr, bStarch I Status Flag Setif (SREG(s) = 1) then PC $+$ PC $+ k + 1$ None1/2BRBSs, kBranch I Status Flag Setif (SREG(s) = 1) then PC $+$ PC $+ k + 1$ None1/2BRECs, kBranch I Kotegualif (Z = 1) then PC $+$ PC $+ k + 1$ None1/2BRECkBranch I floadif (Z = 0) then PC $+$ PC $+ k + 1$ None1/2BRECkBranch I floadif (Z = 0) then PC $+$ PC $+ k + 1$ None1/2BRECkBranch I f	RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
HETSubroutine ReturnPC - STACKNone4HETIInterrupt ReturnPC - STACKI4CPSEPtd, RrCompare, Skip if Equalif (Rd = Rr) PC + PC + 2 or 3None112/3CPRd, RrCompare Wit CarryRd - RrZ,N,V,C,H1CPCRd, RrCompare Wit CarryRd - Rr - CZ,N,V,C,H1CPLRd, KCompare Negaster with ImmediateRd - KrZ,N,V,C,H1CPLRd, KCompare Negaster with ImmediateRd - KrZ,N,V,C,H1SBRSRr, bSkip if Bit n Register is Setif (R(b) = 1)PC + PC + 2 or 3None11/2/3SBRSP, bSkip if Bit n Register is Setif (R(b) = 1)PC + PC + 2 or 3None11/2/3SBRSP, bSkip if Bit n I/O Register Clearedif (P(b) = 0) PC + PC + 2 or 3None11/2/3SBRSS, kBranch f Status Flag Setif (SREG(s) = 0) then PC - PC + k + 1None11/2BRBCs, kBranch if Status Flag Setif (SREG(s) = 0) then PC - PC + k + 1None11/2BRBCkBranch if Equalif (Z=1) then PC - PC + k + 1None11/2BRCkBranch if Carry Setif (C = 0) then PC + PC + k + 1None11/2BRCkBranch if Carry Clearedif (C = 0) then PC + PC + k + 1None11/2BRCkBranch if Set or Equal, Signedif (N = 0) then PC + PC + k + 1None11/2BRCkBranch if Garry Geard	ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
HETIInterrupt ReturnPC - STACKI4CPSERd, RrCompare, Skip if Equalif (Rd = Rr) PC + PC + 2 or 3None1/2/3CPRd, RrCompareRd - RrZ,N,V,C,H1CPCRd, RrCompare with CarryRd - Rr - CZ,N,V,C,H1CPIRd, KCompare Register with ImmediateRd - KZ,N,V,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b) = 0) PC + PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register Clearedif (Rh(b) = 1) PC + PC + 2 or 3None1/2/3SBRSP, bSkip if Bit in 1/O Register Setif (Rh(b) = 1) PC + PC + 2 or 3None1/2/3SBRSS, kBranch if Status Flag Setif (Rb(b) = 1) PC + PC + 2 or 3None1/2/3BRSs, kBranch if Status Flag Setif (Rb(b) = 1) PC + PC + 2 or 3None1/2/3BRSs, kBranch if Status Flag Clearedif (Rb(c) = 1) PC + PC + 2 or 3None1/2/3BRSs, kBranch if Status Flag Clearedif (SREG(s) = 1) then PC + PC + k + 1None1/2BRSs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC + PC + k + 1None1/2BRSkBranch if Carry Setif (C = 0) then PC + PC + k + 1None1/2BRSkBranch if Carry Clearedif (C = 0) then PC + PC + k + 1None1/2BRCkBranch if Carry Clearedif (C = 0) then PC + PC + k + 1None1/2	RET		Subroutine Return	$PC \leftarrow STACK$	None	4
CPSERd, RrCompare, Skip if Equalif (Rd = Rr) PC \leftarrow PC + 2 or 3None1/2/3CPRd, RrCompare with CarryRd - RrZ,N,V,C,H1CPCRd, RrCompare with CarryRd - Rr - CZ,N,V,C,H1CPIRd, KCompare Register with ImmediateRd - KZ,N,V,C,H1SBRCRr, bSkip if Bit In Register Clearedif (Rr(b) = 0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit In Register Clearedif (Rr(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3SBRCP, bSkip if Bit In UD Register Clearedif (R(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3SBRSR, kBranch if Status Flag Setif (R(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC \leftarrow PC + k + 1None1/2BREDkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCkBranch if Gauser or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHkBranch if Gauser or Eq	RETI		Interrupt Return	$PC \leftarrow STACK$	1	4
CPRd, RrCompareRd - RrZ,N,V,C,H1CPCRd, RrCompare Register with ImmediateRd - Rr - CZ,N,V,C,H1CPIRd, KCompare Register with ImmediateRd - KZ,N,V,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b) = 0) PC + PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Negister Clearedif (Rr(b) = 0) PC + PC + 2 or 3None1/2/3SBRSP, bSkip if Bit in IO Register Clearedif (Rr(b) = 1) PC + PC + 2 or 3None1/2/3SBRSP, bSkip if Bit in IO Register Clearedif (RRG(s) = 1) then PC + PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 0) then PC + PC + k+1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC + PC + k+1None1/2BREQkBranch if Equalif (Z = 0) then PC + PC + k+1None1/2BRCSkBranch if Garry Setif (C = 0) then PC + PC + k+1None1/2BRCCkBranch if Garry Setif (C = 0) then PC + PC + k+1None1/2BRCLkBranch if Same or Higherif (C = 0) then PC + PC + k+1None1/2BRLkBranch if Algeneif (N = 0) then PC + PC + k+1None1/2BRHkBranch if Gareer or Equal, Signedif (N = 0) then PC + PC + k+1None1/2BRHkBranch if Hallif (N = 0) then PC + PC + k+1 <td< td=""><td>CPSE</td><td>Rd, Rr</td><td>Compare, Skip if Equal</td><td>if (Rd = Rr) PC \leftarrow PC + 2 or 3</td><td>None</td><td>1/2/3</td></td<>	CPSE	Rd, Rr	Compare, Skip if Equal	if (Rd = Rr) PC \leftarrow PC + 2 or 3	None	1/2/3
CPCRd, RrCompare with CarryRd - Rr - CZ,N,V,C,H1CPIRd, KCompare Register with ImmediateRd - KZ,N,V,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register Clearedif (Rr(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3SBRCP, bSkip if Bit in /O Register is Setif (Rtb) = 0) PC \leftarrow PC + 2 or 3None1/2/3BRSS, kBranch if Status Flag Setif (Rtb) = 1) PC \leftarrow PC + 2 or 3None1/2/3BRSs, kBranch if Status Flag Clearedif (Rtb) = 1) PC \leftarrow PC + 2 or 3None1/2BRBSs, kBranch if Status Flag Clearedif (SREG(s) = 1) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (Z = 1) then PC \leftarrow PC + k + 1None1/2BRCkBranch if Carry Setif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Glearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Carry Setif (N = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Carry Se	CP	Rd, Rr	Compare	Rd – Rr	Z,N,V,C,H	1
CPIRd, KCompare Register with ImmediateRd - KZ,Ny,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b) = 0) PC + PC + 2 or 3None11/23SBRSRr, bSkip if Bit in Register Clearedif (Rr(b) = 1) PC + PC + 2 or 3None11/23SBICP, bSkip if Bit in 1/O Register Clearedif (P(b) = 0) PC + PC + 2 or 3None11/23SBISP, bSkip if Bit in 1/O Register Clearedif (P(b) = 0) PC + PC + 2 or 3None11/23BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC - PC + k + 1None11/2BRBCs, kBranch if Status Flag Setif (SREG(s) = 0) then PC - PC + k + 1None11/2BRBCkBranch if Status Flag Clearedif (Z = 1) then PC - PC + k + 1None11/2BRCSkBranch if Status Flag Clearedif (Z = 0) then PC - PC + k + 1None11/2BRCCkBranch if Carry Clearedif (C = 0) then PC - PC + k + 1None11/2BRCCkBranch if Same or Higherif (C = 0) then PC + PC + k + 1None11/2BRLDkBranch if Same or Higherif (N = 0) then PC + PC + k + 1None11/2BRHkBranch if ILowerif (N = 0) then PC + PC + k + 1None11/2BRLkBranch if Greater or Equal, Signedif (N = 0 = 0) then PC + PC + k + 1None11/2BRHkBranch if Greater or Equal, Signedif (N = 0 = 0) then PC + PC + k + 1None11	CPC	Rd, Rr	Compare with Carry	Rd – Rr – C	Z,N,V,C,H	1
SBRCRr, bSkip if Bit in Register Clearedif (Rr(b) = 0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register is Setif (Rr(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3SBICP, bSkip if Bit in I/O Register Clearedif (P(b) = 0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register is Setif (R(b) = 1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(g) = 1) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(g) = 0) then PC \leftarrow PC + k + 1None1/2BREQkBranch if I Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if I Mot Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if I derry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Garry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Garry Clearedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if I Garry Clearedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRUDkBranch if Garry Clearedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None <td< td=""><td>CPI</td><td>Rd, K</td><td>Compare Register with Immediate</td><td>Rd – K</td><td>Z,N,V,C,H</td><td>1</td></td<>	CPI	Rd, K	Compare Register with Immediate	Rd – K	Z,N,V,C,H	1
SBRSRr, bSkip if Bit in Register is Setif (Rr(b) = 1) PC \leftarrow PC + 2 or 3None12/3SBICP, bSkip if Bit in I/O Register Clearedif (P(b) = 0) PC \leftarrow PC + 2 or 3None11/2/3SBISP, bSkip if Bit in I/O Register is Setif (P(b) = 1) PC \leftarrow PC + 2 or 3None11/2/3BRBSs, kBranch if Status Flag Clearedif (SREG(s) = 1) then PC \leftarrow PC + k + 1None11/2BRBCs, kBranch if Status Flag Clearedif (Z = 0) then PC \leftarrow PC + k + 1None11/2BREQkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None11/2BRCSkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None11/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None11/2BRCLkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None11/2BRCLkBranch if Munusif (N = 0) then PC \leftarrow PC + k + 1None11/2BRCLkBranch if Munusif (N = 0) then PC \leftarrow PC + k + 1None11/2BRHkBranch if Munusif (N = 0) then PC \leftarrow PC + k + 1None11/2BRHkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None11/2BRHkBranch if Munusif (N = 0) then PC \leftarrow PC + k + 1None11/2BRHkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None11/2BRH </td <td>SBRC</td> <td>Rr, b</td> <td>Skip if Bit in Register Cleared</td> <td>if $(Rr(b) = 0) PC \leftarrow PC + 2 \text{ or } 3$</td> <td>None</td> <td>1/2/3</td>	SBRC	Rr, b	Skip if Bit in Register Cleared	if $(Rr(b) = 0) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBICP, bSkip if Bit in I/O Register Clearedif $(P(b) = 0) PC \leftarrow PC + 2 or 3$ None1/2/3SBISP, bSkip if Bit in I/O Register is Setif $(R(b) = 1) PC \leftarrow PC + 2 or 3$ None1/2/3BRBSs, kBranch if Status Flag Setif $(R(b) = 1) PC \leftarrow PC + k + 1$ None1/2BRBCs, kBranch if Status Flag Setif $(SREG(s) = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BREQkBranch if fot Equalif $(Z = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRNEkBranch if Ot Equalif $(Z = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRCCkBranch if Carry Setif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRCCkBranch if Garry Clearedif $(C = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRSHkBranch if Minusif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRLOkBranch if Minusif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRLDkBranch if Minusif $(N = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRPLkBranch if Minusif $(N = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRGEkBranch if Greater or Equal, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRHZkBranch if Idees Than Zero, Signedif $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Setif $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2B	SBRS	Rr, b	Skip if Bit in Register is Set	if $(Rr(b) = 1) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBISP, bSkip if Bit in I/O Register is Setif (R(b) = 1) PC - PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC - PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC - PC + k + 1None1/2BREQkBranch if Status Flag Clearedif (Z = 1) then PC - PC + k + 1None1/2BRNEkBranch if Not Equalif (Z = 0) then PC - PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 0) then PC - PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC - PC + k + 1None1/2BRLOkBranch if Carry Clearedif (C = 0) then PC - PC + k + 1None1/2BRLOkBranch if Minusif (N = 1) then PC - PC + k + 1None1/2BRHIkBranch if Graeter or Equal, Signedif (N = 0) then PC - PC + k + 1None1/2BRPLkBranch if Graeter or Equal, Signedif (N = 0) then PC - PC + k + 1None1/2BRQEkBranch if Graeter or Equal, Signedif (N = 0) then PC - PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (N = 0) then PC - PC + k + 1None1/2BRHCkBranch if Half-carry Flag Clearedif (N = V = 0) then PC - PC + k + 1None1/2BRHSkBranch if Half-carry Flag Clearedif (N = V = 0) then PC - PC + k + 1None1/2	SBIC	P, b	Skip if Bit in I/O Register Cleared	if $(P(b) = 0) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Equalif (Z = 1) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Lowerif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 0) then PC \leftarrow PC + k + 1None1/2BRNLkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRL0kBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRALkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRALkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Half-carry Flag Setif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Half-carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if T-flag Setif (H = 0) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if T-flag Setif	SBIS	P, b	Skip if Bit in I/O Register is Set	if $(R(b) = 1) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Equalif (Z = 1) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Minusif (C = 1) then PC \leftarrow PC + k + 1None1/2BRLUkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLUkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLUkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if T-flag Setif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if T-flag Clearedif (T = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k +	BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC \leftarrow PC + k + 1	None	1/2
BREQkBranch if Equalif $(Z = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRNEkBranch if Not Equalif $(Z = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRCSkBranch if Carry Setif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRCCkBranch if Carry Clearedif $(C = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRSHkBranch if Same or Higherif $(C = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRLOkBranch if Lowerif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRMIkBranch if Minusif $(N = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRPLkBranch if Greater or Equal, Signedif $(N = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRECkBranch if Greater or Equal, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRLTkBranch if Half-carry Flag Setif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRHSkBranch if Half-carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if T-flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Jeard Setif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Interrupt Enabledif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRID <td>BRBC</td> <td>s, k</td> <td>Branch if Status Flag Cleared</td> <td>if (SREG(s) = 0) then $PC \leftarrow PC + k + 1$</td> <td>None</td> <td>1/2</td>	BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then $PC \leftarrow PC + k + 1$	None	1/2
BRNEkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRLDkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BREkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Half-carry Flag Setif (N = V = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (N = V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Clearedif (N = V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Clearedif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k + 1None1/2BRTCkBranch if T-flag Setif (V = 1) then PC \leftarrow PC + k + 1None1/2BRVSkBranch if Overflow Flag is Setif (V = 0) then PC \leftarrow PC + k + 1None1/2BRVCk <t< td=""><td>BREQ</td><td>k</td><td>Branch if Equal</td><td>if (Z = 1) then PC \leftarrow PC + k + 1</td><td>None</td><td>1/2</td></t<>	BREQ	k	Branch if Equal	if (Z = 1) then PC \leftarrow PC + k + 1	None	1/2
BRCSkBranch if Carry Setif (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BREkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BREkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Hufscarry Flag Setif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half-carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k + 1None1/2BRTCkBranch if T-flag Clearedif (T = 0) then PC \leftarrow PC + k + 1None1/2BRVSkBranch if Overflow Flag is Setif (V = 0) then PC \leftarrow PC + k + 1None1/2BRVCkBranch if Overflow Flag is Clearedif (V = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (V = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if I	BRNE	k	Branch if Not Equal	if (Z = 0) then PC \leftarrow PC + k + 1	None	1/2
BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if T-flag Setif (T = 1) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k + 1None1/2BRTCkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k + 1None1/2BRVSkBranch if Overflow Flag is Setif (V = 1) then PC \leftarrow PC + k + 1None1/2BRVCkBranch if Interrupt Enabledif (V = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (U = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (U = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interru	BRCS	k	Branch if Carry Set	if (C = 1) then PC \leftarrow PC + k + 1	None	1/2
BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRPLkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half-carry Flag Clearedif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T-flag Setif (T = 1) then PC \leftarrow PC + k + 1None1/2BRTCkBranch if T-flag Setif (T = 0) then PC \leftarrow PC + k + 1None1/2BRVSkBranch if Overflow Flag is Setif (Y = 1) then PC \leftarrow PC + k + 1None1/2BRVCkBranch if Overflow Flag is Clearedif (Y = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (Y = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (U = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (U = 0) then PC \leftarrow PC + k + 1None1/2BRUCkBranch if Interrupt Enabledif (U = 0) then PC \leftarrow PC + k + 1None1/2 <tr <td=""></tr>	BRCC	k	Branch if Carry Cleared	if (C = 0) then $PC \leftarrow PC + k + 1$	None	1/2
BRLOkBranch if Lowerif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRMIkBranch if Minusif $(N = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRPLkBranch if Plusif $(N = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRGEkBranch if Greater or Equal, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRLTkBranch if Less Than Zero, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRHSkBranch if Half-carry Flag Setif $(H = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if T-flag Setif $(T = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if T-flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVSkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Interrupt Enabledif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIEkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIDkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRSH	k	Branch if Same or Higher	If $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRMIkBranch if Minusif $(N = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRPLkBranch if Plusif $(N = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRGEkBranch if Greater or Equal, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRLTkBranch if Less Than Zero, Signedif $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHSkBranch if Half-carry Flag Setif $(H = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if T-flag Setif $(T = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if T-flag Setif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVSkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Overflow Flag is Clearedif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIEkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIDkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRLO	ĸ	Branch if Lower	If $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half-carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half-carry Flag Clearedif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T-flag Setif (T = 1) then PC \leftarrow PC + k + 1None1/2BRTCkBranch if T-flag Clearedif (T = 0) then PC \leftarrow PC + k + 1None1/2BRVSkBranch if Overflow Flag is Setif (V = 1) then PC \leftarrow PC + k + 1None1/2BRVCkBranch if Overflow Flag is Clearedif (V = 1) then PC \leftarrow PC + k + 1None1/2BRIEkBranch if Interrupt Enabledif (I = 1) then PC \leftarrow PC + k + 1None1/2BRIDkBranch if Interrupt Enabledif (I = 0) then PC \leftarrow PC + k + 1None1/2	BRMI	k	Branch if Minus	If $(N = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRGEkBranch if Greater of Equal, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRLTkBranch if Less Than Zero, Signedif $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHSkBranch if Half-carry Flag Setif $(H = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if T-flag Setif $(T = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if T-flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVSkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Overflow Flag is Clearedif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIEkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIDkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRPL	k	Branch if Plus	If $(N = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRL1kBranch if Less Than Zero, Signedif $(M \oplus V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHSkBranch if Half-carry Flag Setif $(H = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if T-flag Setif $(T = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if T-flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVSkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Overflow Flag is Clearedif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIEkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIDkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRGE	ĸ	Branch if Greater or Equal, Signed	If $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHSkBranch if Half-carry Flag Setif $(H = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half-carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if T-flag Setif $(T = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if T-flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVSkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Overflow Flag is Clearedif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIEkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIDkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRLI	ĸ	Branch if Less Than Zero, Signed	if $(N \oplus V = 1)$ then PC \leftarrow PC + k + 1	None	1/2
DRTCKBranch if T-flag Setif (T = 0) then PC \leftarrow PC + K + 1None1/2BRTSkBranch if T-flag Setif (T = 1) then PC \leftarrow PC + K + 1None1/2BRTCkBranch if T-flag Clearedif (T = 0) then PC \leftarrow PC + K + 1None1/2BRVSkBranch if Overflow Flag is Setif (V = 1) then PC \leftarrow PC + K + 1None1/2BRVCkBranch if Overflow Flag is Clearedif (V = 0) then PC \leftarrow PC + K + 1None1/2BRIEkBranch if Interrupt Enabledif (I = 0) then PC \leftarrow PC + K + 1None1/2BRIDkBranch if Interrupt Enabledif (I = 0) then PC \leftarrow PC + K + 1None1/2	BRHS	K	Branch if Half-carry Flag Set	If $(H = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTSKBranch if 1-flag SetIf $(1 = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRTCkBranch if T-flag Clearedif $(T = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRVSkBranch if Overflow Flag is Setif $(V = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRVCkBranch if Overflow Flag is Clearedif $(V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIEkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRIDkBranch if Interrupt Enabledif $(I = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRHU	ĸ	Dranch If Half-carry Flag Cleared	$ (n = 0) \text{ then } PC \leftarrow PC + K + 1 $	None	1/2
BRIC K Branch if 0verflow Flag is Set if (I = 0) then PC ← PC + K + 1 None 1/2 BRVS k Branch if Overflow Flag is Set if (V = 1) then PC ← PC + k + 1 None 1/2 BRVC k Branch if Overflow Flag is Cleared if (V = 0) then PC ← PC + k + 1 None 1/2 BRIE k Branch if Interrupt Enabled if (I = 0) then PC ← PC + k + 1 None 1/2 BRID k Branch if Interrupt Enabled if (I = 0) then PC ← PC + k + 1 None 1/2	BRIS	K	Branch If I-flag Set	If $(I = 1)$ then PC \leftarrow PC + k + 1	None	1/2
BRVC k Branch if Overflow Flag is Cleared If (V = 1) then PC ← PC + k + 1 None 1/2 BRIE k Branch if Interrupt Enabled if (V = 0) then PC ← PC + k + 1 None 1/2 BRIE k Branch if Interrupt Enabled if (I = 1) then PC ← PC + k + 1 None 1/2	BRIC	ĸ	Branch IT I-Tiag Cleared	If $(1 = 0)$ then $PC \leftarrow PC + K + 1$	None	1/2
BRIE k Branch if lnterrupt Enabled if (l = 0) then PC ← PC + k + 1 None 1/2 BRIE k Branch if Interrupt Enabled if (l = 1) then PC ← PC + k + 1 None 1/2	BRVS	ĸ	Branch If Overflow Flag is Set	$ii (v = 1) \text{ then } PC \leftarrow PC + K + 1$	None	1/2
DFILE K Branch if interrupt Enabled If (I = 1) then $PC \leftarrow PC + K + 1$ None 1/2 BRID k Branch if Interrupt Disabled if (I = 0) then $PC \leftarrow PC + K + 1$ None 1/2		ĸ	Dranch if Uvernow Flag is Cleared	If $(v = 0)$ then $PC \leftarrow PC + K + 1$	None	1/2
		r.	Branch if Interrupt Disabled	if $(I = I)$ then PC \leftarrow PC + K + I if $(I = 0)$ then PC \leftarrow PC + k + 1	None	1/2





Instruction Set Summary (Continued)

Mnemonic	Operands	Description	Operation	Flags	# Clocks
	operando	Description	operation	Tiago	# Olocks
		Meus hetusen Deristere		Nana	4
	Ru, Rr	Nove between Registers		None	1
	Bd X		$Rd \leftarrow K$	None	2
	Bd X+	Load Indirect and Post-inc	$Bd \leftarrow (X) X \leftarrow X + 1$	None	2
LD	Rd, -X	Load Indirect and Pre-dec.	$X \leftarrow X - 1$, Bd $\leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	None	2
LD	Rd, Y+	Load Indirect and Post-inc.	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LD	Rd, -Y	Load Indirect and Pre-dec.	$Y \leftarrow Y - 1$, Rd \leftarrow (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	$Rd \leftarrow (Y + q)$	None	2
LD	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	2
LD	Rd, Z+	Load Indirect and Post-inc.	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	2
LD	Rd, -Z	Load Indirect and Pre-dec.	$Z \leftarrow Z - 1$, $Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
ST	X+, Rr	Store Indirect and Post-inc.	$(X) \leftarrow \operatorname{Rr}, X \leftarrow X + 1$	None	2
ST	-X, Rr	Store Indirect and Pre-dec.	$X \leftarrow X - 1$, (X) $\leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-inc.	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2
ST	-Y, Rr	Store Indirect and Pre-dec.	$Y \leftarrow Y - 1$, (Y) $\leftarrow Rr$	None	2
STD	Y+q, Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2
ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
ST	Z+, Rr	Store Indirect and Post-inc.	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2
ST	-Z, Rr	Store Indirect and Pre-dec.	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2
STD	Z+q, Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3
IN	Rd, P	In Port		None	1
DUDU	P, Rr	Out Port	$P \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack		None	2
		Pop Register from Stack	Rd ← STACK	None	2
SBI	Ph	Set Bit in I/O Begister	$I/O(P h) \leftarrow 1$	None	2
CBI	P b	Clear Bit in I/O Begister	$I/O(P b) \leftarrow 0$	None	2
	Bd	Logical Shift Left	$Bd(n+1) \leftarrow Bd(n) Bd(0) \leftarrow 0$	ZCNV	1
LSB	Bd	Logical Shift Bight	$Bd(n) \leftarrow Bd(n+1), Bd(7) \leftarrow 0$	Z.C.N.V	1
BOL	Bd	Botate Left through Carry	$Bd(0) \leftarrow C, Bd(n+1) \leftarrow Bd(n), C \leftarrow Bd(7)$	Z.C.N.V	1
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
ASR	Rd	Arithmetic Shift Right	$Rd(n) \leftarrow Rd(n+1), n = 06$	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	$Rd(30) \leftarrow Rd(74), Rd(74) \leftarrow Rd(30)$	None	1
BSET	s	Flag Set	$SREG(s) \leftarrow 1$	SREG(s)	1
BCLR	s	Flag Clear	$SREG(s) \leftarrow 0$	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	$T \leftarrow Rr(b)$	Т	1
BLD	Rd, b	Bit Load from T to Register	$Rd(b) \leftarrow T$	None	1
SEC		Set Carry	C ← 1	С	1
CLC		Clear Carry	$C \leftarrow 0$	С	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	$N \leftarrow 0$	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	l ← 1	1	1
CLI		Global Interrupt Disable	← 0	1	1
SES	+	Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	<u>S</u> ←0	S	1
SEV		Set Two's Complement Overflow	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SEI		Set I in SREG			1
		Clear I in SREG			1
SEH	+	Set Half-carry Flag in SREG		н	1
	+	Clear Half-carry Flag in SREG	U → H	H	1
	+		(and appointing descer for Older function)	None	1
WDR		Watchdog Beset	(see specific descr. for Sieep function)	None	1
				INCHE	i I

8S2



