Features

- High-performance, Low-power AVR[®] 8-bit Microcontroller
- Advanced RISC Architecture
 - 130 Powerful Instructions Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
 - 64K Bytes of In-System Reprogrammable Flash program memory
 - 2K Bytes EEPROM
 - 4K Bytes Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C⁽¹⁾
 - Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation
 - Up to 64K Bytes Optional External Memory Space
 - Programming Lock for Software Security
 - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
 - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Two 8-bit PWM Channels
 - 6 PWM Channels with Programmable Resolution from 1 to 16 Bits
 - 8-channel, 10-bit ADC
 - 8 Single-ended Channels
 - 7 Differential Channels
 - 2 Differential Channels with Programmable Gain (1x, 10x, 200x)
 - Byte-oriented Two-wire Serial Interface
 - Dual Programmable Serial USARTs
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with On-chip Oscillator
 - On-chip Analog Comparator
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated RC Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
 - Software Selectable Clock Frequency
 - ATmega103 Compatibility Mode Selected by a Fuse
 - Global Pull-up Disable
- I/O and Packages
 - 53 Programmable I/O Lines
 - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
 - 2.7 5.5V for ATmega64L
 - 4.5 5.5V for ATmega64
- Speed Grades
 - 0 8 MHz for ATmega64L
 - 0 16 MHz for ATmega64





8-bit **AVR**[®] Microcontroller with 64K Bytes In-System Programmable Flash

ATmega64 ATmega64L

Summary

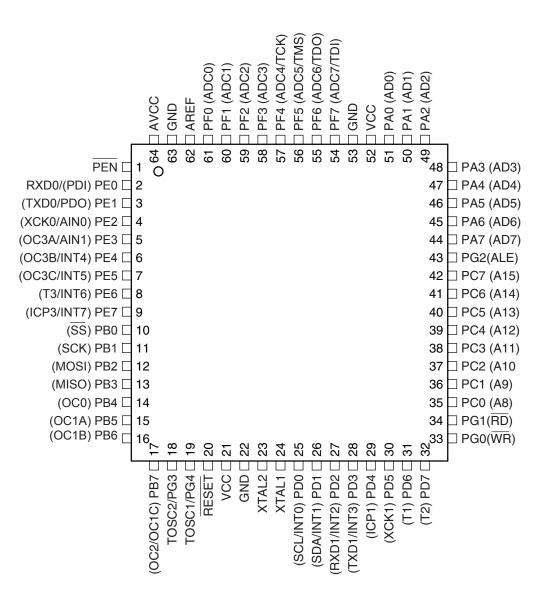
2490PS-AVR-07/09



Figure 1. Pinout ATmega64

Pin Configuration

TQFP/MLF





Disclaimer

Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

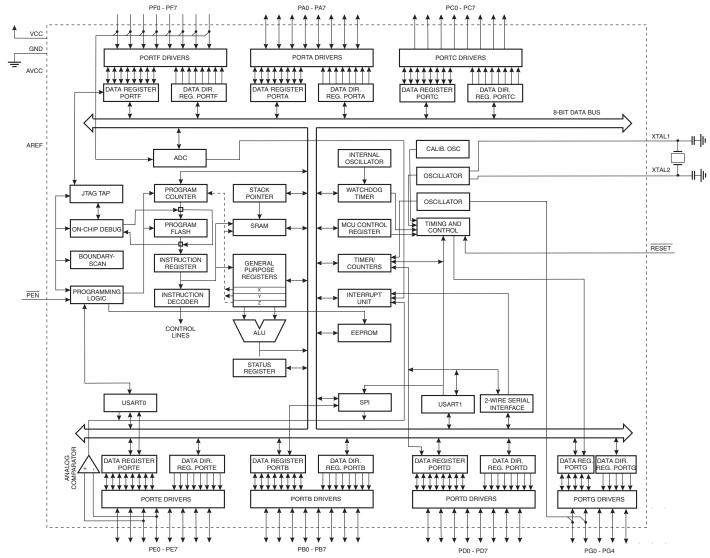
² ATmega64(L)

Overview

The ATmega64 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega64 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Block Diagram





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





The ATmega64 provides the following features: 64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 2K bytes EEPROM, 4K bytes SRAM, 53 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), four flexible Timer/Counters with compare modes and PWM, two USARTs, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high-density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot Program can use any interface to download the Application Program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega64 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega64 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

ATmega103 and ATmega64 Compatibility

The ATmega64 is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O location reserved in the AVR instruction set. To ensure backward compatibility with the ATmega103, all I/O locations present in ATmega103 have the same location in ATmega64. Most additional I/O locations are added in an Extended I/O space starting from 0x60 to 0xFF (i.e., in the ATmega103 internal RAM space). These location can be reached by using LD/LDS/LDD and ST/STS/STD instructions only, not by using IN and OUT instructions. The relocation of the internal RAM space may still be a problem for ATmega103 users. Also, the increased number of Interrupt Vectors might be a problem if the code uses absolute addresses. To solve these problems, an ATmega103 compatibility mode can be selected by programming the fuse M103C. In this mode, none of the functions in the Extended I/O space are in use, so the internal RAM is located as in ATmega103. Also, the extended Interrupt Vectors are removed.

The ATmega64 is 100% pin compatible with ATmega103, and can replace the ATmega103 on current printed circuit boards. The application notes "Replacing ATmega103 by ATmega128" and "Migration between ATmega64 and ATmega128" describes what the user should be aware of replacing the ATmega103 by an ATmega128 or ATmega64.

ATmega103 Compatibility Mode

By programming the M103C Fuse, the ATmega64 will be compatible with the ATmega103 regards to RAM, I/O pins and Interrupt Vectors as described above. However, some new features in ATmega64 are not available in this compatibility mode, these features are listed below:

- One USART instead of two, asynchronous mode only. Only the eight least significant bits of the Baud Rate Register is available.
- One 16 bits Timer/Counter with two compare registers instead of two 16 bits Timer/Counters with three compare registers.
- Two-wire serial interface is not supported.
- Port G serves alternate functions only (not a general I/O port).
- Port F serves as digital input only in addition to analog input to the ADC.
- Boot Loader capabilities is not supported.
- It is not possible to adjust the frequency of the internal calibrated RC Oscillator.
- The External Memory Interface can not release any Address pins for general I/O, neither configure different wait states to different External Memory Address sections.
- Only EXTRF and PORF exist in the MCUCSR Register.
- No timed sequence is required for Watchdog Timeout change.
- Only low-level external interrupts can be used on four of the eight External Interrupt sources.
- Port C is output only.
- USART has no FIFO buffer, so Data OverRun comes earlier.
- The user must have set unused I/O bits to 0 in ATmega103 programs.

Pin Descriptions

VCC	Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega64 as listed on page 73.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega64 as listed on page 74.





Port C (PC7PC0)	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.
	Port C also serves the functions of special features of the ATmega64 as listed on page 77. In ATmega103 compatibility mode, Port C is output only, and the port C pins are not tri-stated when a reset condition becomes active.
Port D (PD7PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.
	Port D also serves the functions of various special features of the ATmega64 as listed on page 78.
Port E (PE7PE0)	Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.
	Port E also serves the functions of various special features of the ATmega64 as listed on page 81.
Port F (PF7PF0)	Port F serves as the analog inputs to the A/D Converter.
	Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS) and PF4(TCK) will be activated even if a reset occurs.
	The TDO pin is tri-stated unless TAP states that shift out data are entered.
	Port F also serves the functions of the JTAG interface.
	In ATmega103 compatibility mode, Port F is an input port only.
Port G (PG4PG0)	Port G is a 5-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source
	capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.
	resistors are activated. The Port G pins are tri-stated when a reset condition becomes active,
	resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

RESET	Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 19 on page
	52. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.
PEN	This is a programming enable pin for the SPI Serial Programming mode. By holding this pin low during a Power-on Reset, the device will enter the SPI Serial Programming mode. PEN has no

function during normal operation.





Resources A comprehensive set of development tools, application notes and datasheetsare available for download on http://www.atmel.com/avr.

Data Retention Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
										Faye
(0xFF)	Reserved	-	-	-	-	-	-	-	-	
	Reserved	-	-	-	-	-	-	-	-	
(0x9E)	Reserved	-	-	-	-	-	-	-	-	101
(0x9D)	UCSR1C	-	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	191
(0x9C)	UDR1	EVO1	TYON			Data Register		1101/1	100014	188
(0x9B)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	189
(0x9A)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	190
(0x99)	UBRR1L				USARTIBAUG	Rate Register Lo		Dete De sister Hist	-	193
(0x98) (0x97)	UBRR1H	-	-	-	-			Rate Register High		193
(0x97) (0x96)	Reserved	_				-		-	-	
, ,	Reserved	_				-			UCPOL0	101
(0x95) (0x94)	UCSR0C		UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00		191
(0x94) (0x93)	Reserved Reserved	-	-	-	-	-	-	-	-	
(0x93) (0x92)			1		-	-	_	_	-	
(0x92) (0x91)	Reserved Reserved	-	-	-	-	-	-	-	-	
(0x91) (0x90)	UBRR0H	_	-	_	-	-		– Poto Rogistor High		193
(0x90) (0x8F)		_					USARTU Bauur	Rate Register High		195
(0x8F) (0x8E)	Reserved ADCSRB	-	-	-	-	-	ADTS2	– ADTS1	- ADTS0	247
(0x8E) (0x8D)	Reserved	-	-	-	-	-	AD152	AD151	ADTS0	241
(0x8D) (0x8C)	TCCR3C	– FOC3A	FOC3B	FOC3C	_	_	_		_	138
(0x8C) (0x8B)	TCCR3C TCCR3A	COM3A1	COM3A0	COM3B1	– COM3B0	– COM3C1	– COM3C0	– WGM31	- WGM30	138
(0x8B) (0x8A)	TCCR3A TCCR3B	ICNC3	ICES3	- COM3B1	WGM33	WGM32	CS32	CS31	CS30	132
(0x8A) (0x89)	TCOR3B TCNT3H	ICINC3	ICE33		er/Counter3 – Co			0331	0330	138
(0x89) (0x88)	TCNT3H TCNT3L				er/Counter3 – Co					138
(0x88) (0x87)	OCR3AH				unter3 – Output C	*				138
						1 0	0,			
(0x86) (0x85)	OCR3AL OCR3BH				unter3 – Output C unter3 – Output C					139 139
(0x85) (0x84)	OCR3BH				unter3 – Output C unter3 – Output C					139
						· ·				
(0x83) (0x82)	OCR3CH OCR3CL				unter3 – Output C unter3 – Output C		* /			139 139
(0x82) (0x81)	ICR3H				Counter3 – Input (139
(0x80)	ICR3H									140
(0x80) (0x7F)	Reserved	_	-	-	Counter3 – Input		Low Byte	-	-	140
(0x7E)	Reserved	_	_	_	_	_	_	_	_	
(0x7D)	ETIMSK	_		TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C	141
(0x7C)	ETIFR	_	_	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C	141
(0x7B)	Reserved	_	_	-	-	-	-	-	-	172
(0x7A)	TCCR1C	FOC1A	FOC1B	FOC1C	_	_	_	_	_	137
(0x79)	OCR1CH	TOOTA	10018		unter1 – Output C			_		139
(0x78)	OCR1CL				unter1 – Output C	i e	ů j			139
(0x77)	Reserved	_	_	-	_	_	-	_	_	100
(0x76)	Reserved	_	_	_	_	_	_	_	_	
(0x75)	Reserved	_	_	_	_	_	_	_	_	
(0x73) (0x74)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	_	TWIE	206
(0x74) (0x73)	TWDR				Two-wire Serial In					208
(0x73) (0x72)	TWDR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	208
(0x72) (0x71)	TWAR	TWA0	TWA5	TWS5	TWA3	TWA2 TWS3	-	TWPS1	TWPS0	208
(0x71) (0x70)	TWBR	1007	11100		/o-wire Serial Inte		aister	1	1111 00	207
(0x70) (0x6F)	OSCCAL	1		1 1		ibration Register	9.0101			43
(0x6E)	Reserved	_	-	_	-	–	-	-	-	UF.
(0x6D)	XMCRA	_	SRL2	SRL1	 SRL0	SRW01	SRW00	SRW11		32
(0x6C)	XMCRB	 XMBK	-	-	-	-	XMM2	XMM1	XMM0	34
(0x6B)	Reserved	-	_	_	_	_	-	-	-	
(0x6A)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	90
(0x69)	Reserved	-	-	-	-	-	-	-	-	
(0x68)	SPMCSR	SPMIE	RWWSB	_	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	281
(0x67)	Reserved	-	-	_	-	-	-	-	-	201
(0x66)	Reserved	_	_	_	_	_	_	_	_	
(0x65)	PORTG	_	_	_	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	89
	DDRG	-	-	-	DDG4	DDG3	DDG2	DDG1	DDG0	89
(0x64)	DDING				PING4	PING3	PING2	PING1	PING0	89
(0x64) (0x63)	PING	_								
(0x64) (0x63) (0x62)	PING PORTF	– PORTF7	– PORTF6	– PORTF5	PING4 PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	88





ATmega64(L) 10

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x60)	Reserved	-	-	-	-	-	-	-	-	
0x3F (0x5F)	SREG	I	Т	Н	S	V	N	Z	С	12
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	14
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	14
0x3C (0x5C)	XDIV	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0	39
0x3B (0x5B)	Reserved	-	-	-	-	-	-	-	-	
0x3A (0x5A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	91
0x39 (0x59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	92
0x38 (0x58)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF	INTF1	INTF0	92
0x37 (0x57)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	109, 140, 160
0x36 (0x56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	109, 142, 160
0x35 (0x55)	MCUCR	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	32, 46, 64
0x34 (0x54)	MCUCSR	JTD	-	-	JTRF	WDRF	BORF	EXTRF	PORF	55, 256
0x33 (0x53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	104
0x32 (0x52)	TCNT0				Timer/Co	unter0 (8 Bit)				106
0x31 (0x51)	OCR0		-	Ti	mer/Counter0 Ou	tput Compare Re	gister			106
0x30 (0x50)	ASSR	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	107
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	132
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	136
0x2D (0x4D)	TCNT1H			Time	er/Counter1 - Co	unter Register Hig	gh Byte			138
0x2C (0x4C)	TCNT1L			Tim	er/Counter1 – Co	unter Register Lo	w Byte			138
0x2B (0x4B)	OCR1AH			Timer/Co	unter1 – Output C	ompare Register	A High Byte			139
0x2A (0x4A)	OCR1AL			Timer/Co	unter1 – Output C	Compare Register	A Low Byte			139
0x29 (0x49)	OCR1BH			Timer/Co	unter1 – Output C	ompare Register	B High Byte			139
0x28 (0x48)	OCR1BL			Timer/Co	unter1 – Output C	Compare Register	B Low Byte			139
0x27 (0x47)	ICR1H			Timer/0	Counter1 – Input	Capture Register	High Byte			140
0x26 (0x46)	ICR1L			Timer/	Counter1 – Input	Capture Register	Low Byte			140
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	157
0x24 (0x44)	TCNT2				Timer/Co	unter2 (8 Bit)				159
0x23 (0x43)	OCR2			Tir	mer/Counter2 Out	put Compare Reg	gister			160
0x22 (0x42)	OCDR	IDRD/ OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	253
0x21 (0x41)	WDTCR	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	57
0x20 (0x40)	SFIOR	TSM	-	-	-	ACME	PUD	PSR0	PSR321	72, 111, 145, 227
0x1F (0x3F)	EEARH	-	-	-	-	-	EEPRON	1 Address Registe	er High Byte	22
0x1E (0x3E)	EEARL				EEPROM Addres	s Register Low B	yte			22
0x1D (0x3D)	EEDR				EEPROM	Data Register	1	1		22
0x1C (0x3C)	EECR	-	-	-	-	EERIE	EEMWE	EEWE	EERE	22
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	87
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	87
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	87
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	87
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	87
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	87
0x15 (0x35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	87
0x14 (0x34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	87
0x13 (0x33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	88
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	88
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	88
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	88
0x0F (0x2F)	SPDR				SPI Da	ta Register				169
0x0E (0x2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	169
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	167
0x0C (0x2C)	UDR0		1	1		Data Register	1	1		188
0x0B (0x2B)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	189
0x0A (0x2A)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	190
0x09 (0x29)	UBRR0L					Rate Register Lo			Г	193
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	228
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	243
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	245
0x05 (0x25)	ADCH				ADC Data Re	gister High Byte				246
0x04 (0x24)	ADCL				ADC Data Re	egister Low byte				246
••••••		DODTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	88
0x03 (0x23)	PORTE	PORTE7	TORTED	TORTEO						
	PORTE DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	88



Register Summary (Continued)

Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x00 (0x20)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	89
Notes: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses										

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

 Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.





Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AN	D LOGIC INSTRUC	TIONS			
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \lor Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \lor K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow 0xFF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \lor K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (0xFF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 " (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 " (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 " (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUC		Haddonar maraphy orginal mar onoighou		2,0	-
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP	ĸ	Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL	ĸ	Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK		4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC \leftarrow PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC \leftarrow PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if $(Rr(b)=1) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if $(P(b)=0) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if $(P(b)=1) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then $PC \leftarrow PC+k + 1$	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then $PC \leftarrow PC+k + 1$	None	1/2
BREQ	k k	Branch if Equal	if (Z = 1) then PC \leftarrow PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC \leftarrow PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC \leftarrow PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC \leftarrow PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC \leftarrow PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC \leftarrow PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC \leftarrow PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC \leftarrow PC + k + 1	None	1/2
BRGE	k k	Branch if Greater or Equal, Signed			1/2
			if $(N \oplus V=0)$ then PC \leftarrow PC + k + 1 if $(N \oplus V=1)$ then PC \leftarrow PC + k + 1	None	
BRLT	k	Branch if Less Than Zero, Signed	if $(N \oplus V = 1)$ then PC \leftarrow PC + k + 1 if $(H = 1)$ then PC \leftarrow PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC \leftarrow PC + k + 1 if (H = 0) then PC \leftarrow PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if $(H = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC \leftarrow PC + k + 1 if (T = 0) then PC \leftarrow PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC \leftarrow PC + k + 1	None None	1/2 1/2
				NODA	1/2
BRVS BRVC	k k	Branch if Overflow Flag is Set Branch if Overflow Flag is Cleared	if (V = 1) then PC \leftarrow PC + k + 1 if (V = 0) then PC \leftarrow PC + k + 1	None	1/2

Instruction Set Summary (Continued)

Book Internet Product Product Internet Product Internet Product Internet Product Internet Product MAY Bask Product Product Main Product Product Main Product Product Main Product Product Main Product Product Product Main Product Product Product Main Product Produc	2015			111 DO DO 11 1		4.10														
BAR 2 Section 2 S	BRIE	k	Branch if Interrupt Enabled	if $(I = 1)$ then PC \leftarrow PC + k + 1	None	1/2														
MOVR.B. No. Box Pagear WorkNo. Pagear Work<			Branch if Interrupt Disabled	If $(1 = 0)$ then PC \leftarrow PC + k + 1	None	1/2														
MAUWBri.R.W.Copy Region VariaBri.R.B.4. PILIDNorm.1LDRA.K.Load invertiantBri.A.G.Norm.SLDRA.X.Load invertiant Parlate.BriQ.X.A.Y.I.M.Norm.ZLDRA.Y.Load invertiant Parlate.K.A.Y.J.R.4.(A)Norm.ZLDRA.Y.Load invertiant Parlate.K.Y.J.R.4.(A)Norm.ZLDRA.Y.Load invertiant Parlate.K.Y.J.R.4.(A)Norm.ZLDRA.Y.Load invertiant Parlate.K.Y.J.R.4.(A)Norm.ZLDRA.Y.Load invertiant Parlate.K.Y.J.R.4.(A)Norm.ZLDRA.Z.Load invertiant Parlate.RZNorm.ZLDRA.Z.Load invertiant Parlate.RZZRZLDRA.Z.Load invertiant Parlate.RZZRZZLDRA.Z.Load invertiant Parlate.RZZZ <td< td=""><td></td><td></td><td>Maria Datura Daviator</td><td></td><td>News</td><td></td></td<>			Maria Datura Daviator		News															
LD Bit, X Load Instant Ref $-$ (A) None 1 LD Bit, X Load Instant Ref $-$ (A) None 2 LD Bit, X Load Instant Ref $-$ (A) None 2 LD Bit, X Load Instant Ref $-$ (A) None 2 LD Bit, Y Load Instant Ref $-$ (Y) (Y + Y + 1, Ref + (Y) None 2 LD Bit, Y Load Instant Ref $-$ (Y) (Y + Y + 1, Ref + (Y) None 2 LD Bit, Y Load Instant on Planeten Ref $-$ (Y) (Y + Y + 1, Ref + (Y) None 2 LD Bit, Y Load Instant on Planeten Ref $-$ (Y) None 2 LD Bit, Z Load Instant on Planeten Ref $-$ (A) None 2 LD Bit, Z Load Instant Planeten Ref $-$ (A) None 2 LD Bit, Z Load Instant Planeten Ref $-$ (A) None 2 LD Bit, Z Load Instant Planeten Ref $-$ (A) <																				
LDRa XLoad indrax and phone.Ra + (A) X + X + AB - (A) X																				
LDRk.×Load hadred and Polefic.Ret.×1, Ret.None2LDRs.×Load hadred and Polefic.Rs.×1, Ret.None.2LDRs.VLoad hadred and Polefic.Rst.r/1, Ret.r/VNone.2LDRs.VLoad hadred and Polefic.Rst.r/1, Ret.r/VNone.2LDRs.VLoad hadred and Polefic.Rst.r/1, Ret.r/VNone.2LDRs.VLoad hadred and Polefic.Rst.r/1, Ret.r/VNone.2LDRs.ZLoad hadred and Polefic.Rst.r/1, Ret.r/VNone.2STN.RStere hadred and Polefic.None.None.2STN.RStere hadred and Polefi																				
IDRd. ×Load indiget and ProDec.× × ×																				
IDR4 V.Load indiced and PachaR4 - (Y)None2IDR4. YLoad indiced and PachaR4 - (Y)None3IDR4. YLoad indiced MD EducamentR4 - (Y)None3IDR4. YLoad indiced MD EducamentR4 - (Y)None3IDR4. ZLoad indiced MD EducamentR4 - (Y)None3IDR4. ZLoad indiced MD EducamentR4 - (Z)None3IDR4. ZLoad indiced MD EducamentR4 - (Z)None3IDN. RSteree Inform StAAR4 - (Z)None3IDN. RSteree Inform StAAR4 - (Z)None3IDN. RSteree Inform StAAR4 - (Z) + (Z)None3IDN. RSteree Inform StAAR4 - (Z) +																				
IDRd, Y=Load indread and headsRd - (Y) Y - Y + 1None2LDDRd Y-qLoad indread and headsRd - (Y - q)None2LDDRd ZLoad indread and headsRd - (Y - q)None2LDRd ZLoad indread and headsRd - (Z) Z - Z - 1None2LDRd ZLoad indread and headsRd - (Z) Z - Z - 1None2LDRd ZLoad indread and headsRd - (Z) Z - Z - 1None2LDRd ZLoad indread and headsRd - (Z) Z - Z - 1None2LDLoad indread and headsRd - (Z) Z - Z - 1None2LDLoad indread and headsRd - (Z) Z - Z - 1None2LDSine indicad and headsRd - (Z) Z - Z - 1None2SineX, RrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineX, RrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineY, YrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineY, YrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineY, YrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineY, YrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineY, YrSine indicad and heads(Q) - R - X + 1 (X) - RNone2SineY, YrSine indicad and heads(Q) - R - X + 1 (X) -																				
D.D.RdYLoad Indread MicrophonementY - Y. 1. Rd (Y)None2LDRd.YLoad Infreed MicrophonementRd (Z)None2LDRd.ZLoad Infreed and PostonRd (Z)None2LDRd.ZLoad Infreed and PostonZ - 2-1, Rd (Z)None2LDRd.ZLoad Infreed MicrophonementRd (Z)None2LDRd.ZLoad Infreed MicrophonementRd (Z)None2LDRd.XLoad Infreed MicrophonementRd (Z)None2LDS.RSize InfieldO() - (R)None2STX, RUSize InfieldNone22STX, RUSize InfieldNone22STX, RUSize InfieldNone22STX, RUSize InfieldNone22STX, RUSize InfieldNone22STX, RUSize Infield <td< td=""><td></td><td></td><td></td><td></td><td></td><td></td></td<>																				
L0D Rd Y-q Load Informet with Digatoment Rd + (2) None 2 LD Rd Z Load Informet and Problen. Rd + (2) Z + 2.71.1 None 2 LD Rd Z Load Informet and Problen. Z + 2.71.84 (2) None 2 LDS Rd Z Load Informet and Problen. Z + 2.71.84 (2) None 2 LDS Rd X-Q Load Informet and Problen. Rd + (2) None 2 LDS Rd X-R Store Informet and Problen. (0) - ft, X - X + 11.0 None 2 ST X, ft Store Informet and Problen. (N) - ft, X - X + 11.0 None 2 ST X, ft Store Informet and Problen. (Y - V, Y - Y + 11.0 None 2 ST Y, ft Store Informet and Problen. (Y - V, Y - Y + Y + 11.0 None 2 ST Y, ft Store Informet and Problen. (Y - V, Y - Y + Y + 11.0 None 2 ST Y, ft Store Informet and Problen. (Y - Ft, Y - Z + 11.0 None 2																				
LD Rd , Z Load indrived modeshin Rd - (Z) _ (Z) None 2 LD Rd, Z Load indrived modeshin Rd - (Z) - (Z-1) None 2 LD Rd, Z Load indrived modeshin Rd - (Z) - (Z) None 2 LDS Rd, K Load indrived Modeshin Rd - (Z) - (Z) None 2 LDS Rd, K Load indrived Modeshin Rd - (Z) - (Z) None 2 ST X, Rr Store Indreed and Post-Inc. (X) - Rr, X - X × 11 None 2 ST Y, Rr Store Indreed and Post-Inc. (Y) - Rr, Y - Y × 11 None 2 ST Y, Rr Store Indreed and Post-Inc. (Y) - Rr, Y - Y × 11 None 2 ST Y, Rr Store Indreed and Post-Inc. (Z) - Rr None 2 ST Y, Rr Store Indreed and Post-Inc. (Z) - Rr None 2 ST Z, Rr Store Indreed and Post-Inc. (Z) - Rr None 2 ST Z, Rr Store Indreed and P																				
DBd.2-Lond Indirect and Per-Do.PL-(2), 7-, 2+1.Nore.2LDDBd.2-Lond Indirect and Per-Do.2, 7-2, 7.1 Bd+(-2)Nore.2LDSBd.2-Lond Indirect and Per-Do.Rd+(-1), -01Nore.2LDSBd.4.Lond Direct from SRAM.Rd+(-0), -RNore.2STX, RrStore Indicet and Post-Inc.(0) - Rr, X - X + 11Nore.2STX, RrStore Indicet and Post-Inc.(0) - Rr, X - X + 11Nore.2STY, RrStore Indicet and Post-Inc.(1) + Rr, Y - Y + 1Nore.2STY, RrStore Indicet and Post-Inc.(1) + Rr, Y - Y + 1Nore.2STY, RrStore Indicet and Post-Inc.(1) + Rr, Y - Y + 1Nore.2STY, RrStore Indicet and Post-Inc.(2) + RrNore.2STY, RrStore Indicet and Post-Inc.(2) + RrNore.2STZ, RrStore Indicet and Post-Inc.(2) + RrNore.2STZ, RrStore Indicet and Post-Inc.(2, - RrNore.2STZ, RrStore Indiget and Post-Inc.(2, - Rr			•																	
LDRd2Load Indiced and Pac-Bac.Z. + Z. + 1, Rd - (-2)Nore2LDSRd. +2Load Indiced and NegatomentRd + (-2) - q)Nore2LDSRd. +1Load Direct from SRAMRd + (-2) - q)Nore2STX. KrStore Indiced and Pac-bac.(0, - Rr, X - X + 1, M)Nore2STX. KrStore Indiced and Pac-bac.(0, - Rr, Y - X, Y, + 1, M)Nore2STY. KrStore Indiced and Pac-bac.(Y, - Rr, Y, - Y, +1)Nore2STY, RrStore Indiced and Pac-bac.(Y, - Y, Y, -Y, +1)Nore2STY, RrStore Indiced and Pac-bac.(Y, - Y, Y, -Y, +1)Nore2STY, RrStore Indiced and Pac-bac.(Y, - Y, -Y, -Y, +1)Nore2STY, RrStore Indiced and Pac-bac.(Z, - Rr, Z, - Z, +1, L)Nore2STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), - RrNore2STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), - RrNore2STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), - RrNore2STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), - RrNore3STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), - RrNore3STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), - RrNore3STZ, RrStore Indiced and Pac-bac.(Z, - Y, -1, (Y), -																				
LDD Rd, 2rq Load Drest from SRM Rd + (J + q) None 2 ST X, Rr Store Indiced and SRM D(n + Rr, X + X + 1) None 2 ST X, Rr Store Indiced and Pre-brac. X + X + 1, (y) + Rr None 2 ST X, Rr Store Indiced and Pre-brac. Y + X + 1, (y) + Rr None 2 ST Y, Kr Store Indiced and Pre-brac. Y + X + 1, (y) + Rr None 2 ST Y, Kr Store Indiced and Pre-brac. Y + X + 1, (y) + Rr None 2 ST Y, Kr Store Indiced and Pre-brac. (D + Rr, Z + 2 + 1) None 2 ST Z, fr Store Indiced and Pre-brac. (D + Rr, Z + 2 + 1) None 2 ST Z, fr Store Undicet and Pre-brac. (Z + G + Rr None 2 ST Z, fr Store Undicet and Pre-brac. Z + 1, (J + Rr None 2 ST Z, fr Store Undicet and Pre-brac. Z + 1, (J + Rr None 3 ST Z,																				
LDSAtkLoad Direct Iron RBAMBd + (b)None2STX + KrStore Indicat and Post-Inc.(D) + FK - X + 1None2STX + KrStore Indicat and Post-Inc.(D) + FK - X + 1 (N) + RTNone2STY + KrStore Indicat and Post-Inc.(Y) + FK - N(y) + RTNone2STY + KrStore Indicat and Post-Inc.(Y) + FK - N(y) + RTNone2STY + KrStore Indicat And Post-Inc.(Y + Y) + (Y) + FKNone2STY + SRStore Indicat And Post-Inc.(Y + Y) + (Y) + FKNone2STZ + KrStore Indicat and Post-Inc.(Z) + RTNone2STZ + RTStore Indicat And Post-Inc.(Z) + RTNone3STZ + RTStore Indicat And Post-Inc.(Z) + RTNone3STZ + RTStore Indicat And Post-Inc.Rd + (Z)None3STZ + RTStore Indicat And Post-Inc.Rd + (Z)No																				
ST X, Pr Store indicet and Peah-ac. (λ + Pr, X + X + 1) None 2 ST X, Pr Store indicet and Peah-ac. X + X + 1, (λ) + Pr None 2 ST Y, Pr Store indicet and Peah-ac. M + Pr, Y + Y None 2 ST Y+R Store indicet and Peah-ac. (M + Pr, Y + Y + 1) None 2 ST Y+R Store indicet and Peah-ac. (M + Pr, Y + Y + 1) None 2 ST Y+R Store indicet and Peah-ac. (M + Pr, Y + Y + 1) None 2 ST Z, Pr Store indicet and Peah-ac. (D + Pr, Y + Y + 1) None 2 ST Z, Pr Store indicet and Peah-ac. (D + Pr, Z + 2 + 1, Q) + Pr None 2 ST Z, Pr Store indicet and Peah-ac. (D + Pr None 2 2 ST Z, Pr Store indicet and Peah-ac. (D + Pr None 2 ST Z, Pr Store indicet and Peah-ac. (D + Pr None 2 Store indinet and P						+														
ST. X-R Store indirect and Pre-Dice. (M) = R, X = -X. Name 2 ST. Y, R' Store indirect and Pre-Dice. (M) = R, V + Y + 1 None 2 ST. Y, R' Store indirect and Pay-Inc. (M) + R', V + Y + 1 None 2 ST. Y, R' Store indirect and Pay-Inc. (M + R', V + Y + 1) None 2 ST. Y, R' Store indirect and Pay-Inc. (M + R', V + Y + 1) None 2 ST. Z, R' Store indirect and Pay-Inc. (M + R', V + Y + 1) None 2 ST. Z, R' Store indirect and Pay-Inc. (M + R', V + Y + 1) None 2 ST. Z, R' Store indirect and Pay-Inc. (M + R', Z + Z + 1) None 2 ST. X, R' Store indirect SAM (M + R', Z + Z + 1) None 2 ST. X, R' Store indirect Min Dispacement (M + R') None 2 ST. X, R' Store indirect Min Dispacement (M + C) None 2 ST. <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>																				
ST $X + X$ Store indirect and Pre-Dec. $X + X + (X) + (rrNone2STY, RrStore indirect and Payl-hc.(Y) + Rr, Y + Y + 1None2STY, RrStore indirect and Payl-hc.(Y + Rr, Y + Y + 1, (Y) + RrNone2STY, RrStore indirect and Payl-hc.(Y + Q, V + Y + 1, (Y) + RrNone2STZ, RrStore indirect and Payl-hc.(Y + Q, Rr, Z + Z + 1)None2STZ, RrStore indirect and Payl-hc.(Z + Q + Rr, Z + Z + 1)None2STZ, RrStore indirect and Payl-hc.(Z + Q + Rr, Z + Z + 1)None2STZ, RrStore indirect and Payl-hc.Z + Z - 1, Q + RrNone2STZ, RrStore indirect and Payl-hc.Z + Z - 1, Q + RrNone2STZ, RrStore indirect and Payl-hc.Z + Z - 1, Q + RrNone2STZ, RrStore indirect and Payl-hc.Z + Z - 1, Q + RrNone3IPMRdLand Porgam MemoryRd + (Z)None3IPMRd, ZLand Porgam MemoryRd + (Z) + Z + 1None3SPMRd, ZLand Porgam Memory and Payl-hc.Rd + (Z) + Z + 1None3IPMRd, ZLand Porgam Memory and Payl-hc.Rd + (Z) + RrNone2IPMRd, ZLand Porgam Memory and Payl-hc.Rd + (Z) + C + 1None3SPMRd, Rd + R$																				
ST Y.R Store Indirect and Post-Anc. (Y) - RT None 2 ST Y.R Store Indirect and Post-Anc. (Y) - RT None 2 ST Y.R Store Indirect and Post-Anc. (Y) - RT None 2 ST Z.R Store Indirect and Post-Anc. (Z) - PT None 2 ST Z.R Store Indirect and Post-Anc. (Z) - PT None 2 ST Z.R Store Indirect and Post-Anc. (Z) - PT None 2 ST Z.R Store Indirect and Post-Anc. (Z) - PT None 2 ST Z.R Store Indirect and Post-Anc. (Z) - PT None 2 ST Z.R Store Indirect and Post-Anc. (Z) - PT None 3 ST Z.R Store Indirect and Post-Anc. (R) - R None 3 ST Z.R Store Indirect Name (Q) - R None 3 DM Rd.2 Load Pogram Memory Rd - RD None 3 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>																				
STY-RStem indirect and Pro-Dec. $(Y) - RY, V - Y - 1$ None2STY-R frStom indirect and Pro-Dec. $Y - Y - 1$, $(Y) - RYNone2STZ, RStore indirect and Pro-Dec.(Y + 0) + RYNone2STZ, RStore indirect and Pro-Dec.(Z) - RYNone2STZ, RStore indirect and Probe.(Z + RY - L + 1)None2STZ, RStore indirect and Probe.Z + 2 - 1, (Z) - RYNone2STZ, R, RStore indirect with Displacement(Z + 0) - RYNone2STZ, R, RStore indirect with Displacement(Z + 0) - RYNone2STR, RStore Direct D SFAM(U - RTNone3LPMRd, ZLoad Program MemoryRd - (Z) - Z - 21None3LPMRd, ZLoad Program MemoryRd - (Z) - Z - 21None3SPMRd, ZLoad Program MemoryRd - (Z) - Z - 21None3DUTP, RDut PortRd - PNone2POPRdPort PortRd - PNone2DUTP, RDut PortRd - PNone2DUTP, RDut PortRd - PNone2DUSRdPort PortRd - PNone2StarRdDut PortRd - PNone2DUSRdPort Port PortRd - PNone2StarR$																				
ST ···、Rr Stene Indirect and Pro-Dec. Y + ·· Y ·· 1, ··/ Pr None 2 STD Y + Qr Stone Indirect and Pro-Dec. (2) - Pr None 2 ST Z, R Stone Indirect and Pro-Dec. (2) - Pr None 2 ST Z, R Stone Indirect and Pro-Dec. (2 - L, Z) - Fr None 2 ST Z, R Stone Indirect and Pro-Dec. (2 + 0) - Rr None 2 ST Z, R Stone Indirect SRAM (0) + Pr None 2 ST X, R Stone Indirect SRAM (0) + Pr None 3 IPM R0, Z Load Program Memory R0 - (2) None 3 IPM R0, Z Load Program Memory (2) - Rt None 1 IPM R0, Z Load Program Memory (2) - Rt None 2 IPM R0, Z Load Program Memory (2) - Rt None 2 IPM R0, Z Stone Program Memory (2) - Rt None																				
STDY+q/eraNore indired with Displacement(Y+q)-RrNone2STZ, RStore indirect and Post-Inc.(Z) - Rr, Z+Z+1None2STZ, RrStore indirect and Post-Inc.(Z) - Rr, Z+Z+1None2STZ, Q, RrStore indirect with Displacement(Z) - Rr, Z+Z+1, QL-Pr,None2STSK, RrStore Direct os SNAM(N) - RrNone3LPMLoad Program MemoryRd - Q, Z-Z+4, ZNone3LPMRd, ZLoad Program MemoryRd - Q, Z-Z+4, None3SPMRd, ZLoad Program MemoryRd - Q, Z-Z+4, None3OUTRdPoly Register NoneRd - Q, Z-Z+4, None3SPMRd, ZLoad Program MemoryRd - Q, Z-Z+4, None3OUTP, RrOut PortRd - Q, Register NoneS2SPMRdPoly Register NoneRd - Q, Z-Z+4, None3OUTRd, PPoly Register NoneRd - Poly Register None2OUTRdRd, PNoneS2SPMRdLogaci Shift NoneC22SPMRdLogaci Shift NoneRd - S2<	ST	Y+, Rr	Store Indirect and Post-Inc.		None	2														
ST Z.P. Rr Size Indiced and Posih.e. (2) - Pir Z - 2 + 1 None 2 ST Z.P. Rr Size Indiced and Pre-Dec. Z - 2 - 1, (2) - Pir Z - 2 + 1 None 2 STD Z.P. Rr Size Indiced with Displacement (2 + a) - K N None 2 STD X.P. Rr Store Indiced with Displacement (2 + a) - K N None 2 STS K.R Store Indiced with Displacement (2 + a) - K N None 2 LPM K.R Store Dirggam Memory Rb - (2) None 3 LPM Rd, Z.* Load Program Memory and Post-Inc Rd - (2) - 2 - 2 + 1 None 3 SPM Store Program Memory and Post-Inc Rd - (2) - 2 - 2 + 1 None -2 IN Rd Post Store Program Memory Rd - C3, Z - 2 + 1 None -2 INT P.R Our Port Rd - C3, Z - 2 + 1 None -2 INT P.R Our Port Rd - C3, C - 2 + 2 + 1 None -2 INT P.R <td></td> <td></td> <td></td> <td>$Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$</td> <td>None</td> <td></td>				$Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$	None															
ST Z, Rr Store Indirect and Pre-Dec. $(2_1 - R_2, -2_1 + 1, (2_1) - R_1$ None 2_2 ST Z_1 R/r Store Indirect with Displacement $Z + 2_1 + 1, (2_1) - R_1$ None 2_2 STD X_1R^2 Store Indirect with Displacement $(Z_1 + 2_1 - R_1)$ None 2_2 STS k, R^2 Store Direct to SRAM $(K_1 - R_1)$ None 3_2 LPM Load Program Memory R0 + C_2 None 3_2 LPM R1, Z Load Program Memory R1 + C_2 None 3_2 SPM R1, Z Load Program Memory $(Z_1 - R_1, R_2)$ None 3_2 SPM R1, Z Load Program Memory $(Z_1 - R_1, R_2)$ None 3_2 SPM R1, Z Load Program Memory $(Z_1 - R_1, R_2)$ None 3_2 SPM R1 Pot Program Memory $(Z_1 - R_1, R_2)$ None 3_2 SPM R1 Pot Program Memory $(Z_1 - S_1, R_2)$ None 3_2 SPM	STD	Y+q,Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2														
ST $2, Rr$ Store indirect and Pic-Dec. $2 - 2.7, L/2 + Rr$ None 2 STD $2 + q_1 + R$ Store indirect with Displacement $(2 + q_1 + R)$ None 2 STSk, RrStore indirect to SRAM $(0) - Rr$ None 3 LPMRdLoad Program MernoryRd - (2) None 3 LPMRd, ZLoad Program MernoryRd + (2) None 3 LPMRd, ZLoad Program MernoryRd + (2) None 3 SPMStore Program Mernory and Post-IncRd + $(2) - 2 + 2r$ -1None 3 IPMRd, PIn PortRd + PNoneNone 3 OUTP, RrOut PortRd + PNone 3 OUTP, RrOut PortRd + PNone 3 POPRdPog Register from StackRd + CACKNone 2 PCPRdPog Register from StackRd + CACKNone 2 PCRRdLogical Strift Left $V(P,P) - 0$ None 2 SIIP.bSet Bit In US Register $V(P,P) - 0$ None 2 LSLRdLogical Strift LeftRd(n+1) + Rd(n), Rd(0) + 0 $Z < NV$ 1LSRRdRobate Right Trough CarryRd(n+1) + Rd(n), Rd(0) + 0 $Z < NV$ 1LSRRdRobate Right Trough CarryRd(n+1) + Rd(n), Rd(1) + 0 $Z < NV$ 1SWPRdRobate Right Trough CarryRd(n+1) + Rd(n+1), CRd(1) + Rd(2), 0) $Z < NV$ 1<	ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2														
STD Z+q,Rr Store indired with Displacement (Z + q) - Rr None 2 STS K, Rr Store Direct to SRAM (b) - Rr None 2 LPM Load Program Memory R0 - (Z) None 3 LPM Rd, Z Load Program Memory and Post-Inc Rd - (Z) None 3 SPM Rd, Z Load Program Memory and Post-Inc Rd - (Z) None 1 SPM Rd, P In Port Rd - P None 1 OUT P, Rr Out Post-Part Memory Rd + P None 2 PD Rd Por Register on Stack STACK - Rr None 2 PD Rd Pop Register from Stack Rd + STACK None 2 CBI P.b Clear Bit in I/O Register IO(P.b) - 0 None 2 CBI P.b Clear Bit in I/O Register IO(P.B) + -0 None 2 CBI Add Logical Shit Left Rd + Rodin Register A Rd(n) - Rd(n+), Rd(n) - 0 Z C/	ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2														
STSk, Rr Store Direct to SPAM(b) - RrNone2LPM-Load Program MemoryR0 - (2)None3LPMRd, 2Load Program MemoryRd - (2)None3LPMRd, 2Load Program MemoryRd - (2) - (2, 2, -2, -2, -1)None3LPMRd, PStore Program MemoryRd - (2, 2, -2, -2, -2, -1)None-INRd, PIn PortRd - PNone1PUSHP, RrOut PortP - RrNone1PUSHRdPop Register for StackSTACK + RrNone2POPPop Register for StackSTACK + RrNone2SBIP, BSet Bit In UO Register100(P, B) - 1None2SBIP, BSet Bit In UG Register100(P, B) - 1None2LSLRdLogical Shft LeftRd(n) + Rd(n), Rd(0) - 0Z, C, N, V1LSRRdLogical Shft RightRd(n) + Rd(n), Rd(n) - 0Z, C, N, V1RORRdRotate Left Through CarryRd(n) + Rd(n), Rd(n) - C, Rd(n)Z, C, N, V1RORRdAntimetic Shft RightRd(n) - Rd(n) + Rd(n), C, Rd(n)Z, C, N, V1RORRdAntimetic Shft RightRd(n) - Rd(n) + Rd(n), C, Rd(n)Z, C, N, V1RORRdAntimetic Shft RightRd(n) - Rd(n) + Rd(n), C, Rd(n)Z, C, N, V1RORRdAntimetic Shft RightRd(n) - Rd(n) + Rd(n), C, Rd(n)Z, C, N, V <t< td=""><td>ST</td><td>-Z, Rr</td><td>Store Indirect and Pre-Dec.</td><td>$Z \leftarrow Z - 1$, (Z) $\leftarrow Rr$</td><td>None</td><td>2</td></t<>	ST	-Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1$, (Z) $\leftarrow Rr$	None	2														
LPMrLoad Program MemoryR0 \leftarrow (2)None3LPMRd, ZLoad Program Memory and Post-IncRd \leftarrow (2), $2 \leftarrow$ 241None3SPMRd, ZLoad Program Memory and Post-IncRd \leftarrow (2), $2 \leftarrow$ 241None3SPMRdStore Program Memory and Post-IncRd \leftarrow (2), $2 \leftarrow$ 241None3SPMRdIn PortRd \leftarrow Par.None1OUTRd, PIn PortRd \leftarrow Par.None1PUSHRrPush Register on StackSTACK \leftarrow RrNone2PDRdPop Register from StackStACKNone2BT ADBTTESTStBtitn I/O RegisterI/O(Pa) \leftarrow 1None2CBIP.bClear Btitn I/O RegisterI/O(Pa) \leftarrow 1None2LSLRdLogical Shift RghtRgh(n1) \leftarrow Rgh(n1), Rgh(0) \leftarrow 0Z.C.N.V1RORRdRotate Left Through CarryRdh(n-L, Rgh(n-L), Rd(n-L), Z.C.N.V1SWAPRdStage Stage	STD	Z+q,Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2														
LPM Rd, Z Load Program Memory Rd ← (Z) None 3 LPM Rd, Z+ Load Program Memory and Post-Inc Rd ← (Z), Z ← Z+1 None 3 SPM Store Program Memory (Z) ← R1:00 None 3 IN Rd, P In Port Rd ← P None 1 QUIT P, RT Out Port P ← R None 1 PUSH Rr Push Register on Stack STACK ← Rr None 2 POP Rd Pog Register from Stack STACK ← Rr None 2 BIT ADD BITTEST INSTUCT UD(Pb) ← 1 None 2 2 SIT AD BITTEST INSTUCT UD(Pb) ← 1 None 2 2 SIT AD BITTEST INSTUCT UD(Pb) ← 1 None 2 2 2 2 SIT AD BITTEST INSTUCT Store St	STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2														
LPMRd, Z+Load Program Memory and Post-IncRd \leftarrow (2), Z \leftarrow 2+1None3SPMIState Program Memory(Z) \leftarrow R1R0None-INRd, PIn PortRd \leftarrow PNone1OUTP, RrOut PortP \leftarrow RrNone1DUTHRP, RrOut PortP \leftarrow RrNone2PDFRdPop Register from StackSTACK \leftarrow RrNone2PDFRdPop Register from StackRd \leftarrow STACKNone2BIP.bClear Bit in UR RegisterI/O(P.b) \leftarrow 0None2CBIP.bClear Bit in UR RegisterRd(n) \leftarrow Rd(n), Rd(n) \leftarrow 0ZC.NV1LSARdLogical Shift LeftRd(n) \leftarrow Rd(n), Rd(n) \leftarrow 0ZC.NV1RDLRdRotat Left Through CarryRd(n) \leftarrow Rd(n), Rd(n) \leftarrow Cd(n)ZC.NV1RDRRdRd Rotat Left Through CarryRd(n) \leftarrow Rd(n) \leftarrow Rd(n), CR(d(n))ZC.NV1SWAPRdArithmetic Shift RightRd(n) \leftarrow Rd(n) \leftarrow Rd(n), CR(d(n))None1SWAPRdSame Shift LeftRd(n) \leftarrow Rd(n) \leftarrow Rd(n), Rd(n)Rd(n)Rd(n)Rd(n)SWAPRdSame Shift LeftRd(n) \leftarrow Rd(n), Rd(n)Rd(n)Rd(n)Rd(n)Rd(n)SWAPRdRd Rotat Left Through CarryRd(n) \leftarrow Rd(n), Rd(n)Rd(n)Rd(n)Rd(n)Rd(n)SWAPRdSame Shift LeftRd(n) \leftarrow Rd(n)Rd(n)Rd(n) <td< td=""><td>LPM</td><td></td><td>Load Program Memory</td><td>R0 ← (Z)</td><td>None</td><td>3</td></td<>	LPM		Load Program Memory	R0 ← (Z)	None	3														
SPMImageStore Program Memory $(Z) \leftarrow R1:R0$ NoneImageINRd, PIn PortIn PortRd $\leftarrow P$ None1NOTP, RrOut PortNone11PUSHRrPush Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackRd \leftarrow STACK \leftarrow None2BITPDSet Bit in I/O RegisterNO(Pb) \leftarrow 1None2BIP, bSet Bit in I/O RegisterI/O(Pb) \leftarrow 0None2CBIP, bClear Bit in I/O RegisterI/O(Pb) \leftarrow 0None2LSLRdLogical Shift LeftRd(m+1) \leftarrow Rd(m), Rd(m) \leftarrow CRd(m+1), Rd(m) \leftarrow CRd(m+1)1LSRRdRd claste Left Through CarryRd(m) \leftarrow Rd(m+1), Rd(m), C-Rd(m)Z,C,N,V1ROLRdRotate Right Through CarryRd(m) \leftarrow Rd(m+1), C-Rd(m)Z,C,N,V1SWAPRdSwap NibblesRd(m+1), no.6,Z,C,N,V1SWAPRdSwap NibblesRd(m+1), no.6,Z	LPM	Rd, Z	Load Program Memory	$Rd \leftarrow (Z)$	None	3														
NRd, PIn PotRd \leftarrow PNone1OUTP, RrOut PotP \leftarrow RrNone1OUTP, RrOut PotPot PotNone1PUSHRrPup Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackRd \leftarrow STACKNone2BIP.bSet Bit In I/O RegisterI/O(P.b) \leftarrow 1None2BIP.bClear Bit In I/O RegisterI/O(P.b) \leftarrow 0None2LSLRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z, C, N/1RORRdRotale Left Through CarryRd(P) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z, C, N/1RORRdRotale Left Through CarryRd(P) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z, C, N/1SKRAPRdSavap NibbesRd(2) \leftarrow Rd(n+1), Rd(n+1), C-Rd(0)Z, C, N/1SWAPRdSavap NibbesRd(2, G) \leftarrow Rd(n+1), Rd(n+1), C-Rd(0)Z, C, N/1SWAPRdSavap NibbesRd(2, G) \leftarrow Rd(n+1), Rd(n+1), C-Rd(0)X, C, N/1BSTsFlag SetSREG(s) \leftarrow 1SREG(s)1BCLRsFlag SetSREG(s) \leftarrow 1SREG(s)1BCLRsFlag SetSREG(s) \leftarrow 1None1SECSet CarryC \leftarrow 1None11GLASet CarryC \leftarrow 1No11SECSet Reginer FlagN \leftarrow 1N11 <td< td=""><td>LPM</td><td>Rd, Z+</td><td>Load Program Memory and Post-Inc</td><td>$Rd \leftarrow (Z), Z \leftarrow Z+1$</td><td>None</td><td>3</td></td<>	LPM	Rd, Z+	Load Program Memory and Post-Inc	$Rd \leftarrow (Z), Z \leftarrow Z+1$	None	3														
OUTP, RrOut PortP \leftarrow RrNone1PUSHRrPush Register on StackSTACK \leftarrow RrNone2POPRdPog Register from StackRd \leftarrow STACKNone2BIT ADD BIT-EST UNERCENDESStatus BIP,bSet Bit in I/O RegisterI/O(P,b) \leftarrow 1None2CBIP,bClear BIt in I/O RegisterI/O(P,b) \leftarrow 0None2LSLRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(n) \leftarrow 0Z,C,N,V1ROLRdRotate Left Through CarryRd(n) \leftarrow Rd(n+1), C+Rd(n), C,C,Rd(n) 2Z,C,N,V1RORRdRotate Right Through CarryRd(n) \leftarrow Rd(n+1), C+Rd(n), Z,C,N,V11SWAPRdSwap NibblesRd(n) \leftarrow Rd(n+1), C+Rd(n), Z,C,N,V11BSETsFlag SetSREG(s) \leftarrow 1SREG(s)11BCLRsFlag ClearSREG(s) \leftarrow 1SREG(s)11BCLRsFlag ClearSREG(s) \leftarrow 1None11SECSet CarryC \leftarrow 1C111SECSet CarryC \leftarrow 1N \leftarrow 1None11SECSet CarryC \leftarrow 1C111SECSet CarryC \leftarrow 1N \leftarrow 1N \leftarrow 111SECSet CarryC \leftarrow 1C111SECSet CarryC \leftarrow 1C111SEC <td< td=""><td>SPM</td><td></td><td>Store Program Memory</td><td>(Z) ← R1:R0</td><td>None</td><td>-</td></td<>	SPM		Store Program Memory	(Z) ← R1:R0	None	-														
PUSH rr Push Register on StackSTACK \leftarrow PrNone2POPRdPop Register from StackRd \leftarrow STACKNone2BT AND BIT-TST WSTWCTONSSet Bit in IO Register $IO(P,b) \leftarrow 1$ None2CBIP.bSet Bit in IO Register $IO(P,b) \leftarrow 0$ None2CBIP.bClear Bit in VO Register $IO(P,b) \leftarrow 0$ None2CBIRdLogical Shift LeftRd(n+1) \leftarrow Rd(n, Rd(0) $\leftarrow 0$ Z.C.N.V1LSLRdLogical Shift RightRd(n) \leftarrow Rd(n+1), \leftarrow Rd(n, CL-Rd(r))Z.C.N.V1ROLRdRotate Left Through CarryRd(0) \leftarrow CRd(n+1), \leftarrow Rd(n, CL-Rd(r))Z.C.N.V1RORRdRotate Right Through CarryRd(0) \leftarrow Rd(n+1), \leftarrow Rd(n, CL-Rd(n))Z.C.N.V1SWAPRdSwap NibblesRd(a, D) \leftarrow Rd(n+1), \leftarrow Rd(n, CL-Rd(n))Z.C.N.V1BGTsFlag SetSREG(s) \leftarrow 1SREG(s)1SWAPRdSwap NibblesRd(a, D) \leftarrow Rd(n-4), Rd(n-4), Rd(n-4), Rd(n-4)None1BGTsFlag ClearSREG(s) \leftarrow 1SREG(s)1BCLRsFlag ClearSREG(s) \leftarrow 1SREG(s)1SECSet CarryC \leftarrow 0C11SECSet CarryC \leftarrow 0C11SECSet CarryC \leftarrow 0C11CLClear Negative FlagN \leftarrow 1N \leftarrow 11SEZSet CarryC \leftarrow 0C<	IN	Rd, P	In Port	$Rd \leftarrow P$	None	1														
POPRdPop Register from StackRd \leftarrow STACKNone2BIT ADBIT-TEST INSTRUCTIONSSBIP.bSet Bit In I/O RegisterI/O(P.b) \leftarrow 1None2CBIP.bClear Bit In I/O RegisterI/O(P.b) \leftarrow 0None2CBIRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z.C.N.V1LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z.C.N.V1ROLRdRotate Left Through CarryRd(0) \leftarrow C.Rd(n+1) \leftarrow Rd(n), C.C-Rd(0)Z.C.N.V1RORRdRotate Right Through CarryRd(0) \leftarrow C.Rd(n) \leftarrow Rd(n+1) \leftarrow Rd(n), C.C-Rd(0)Z.C.N.V1SWAPRdSup NibblesRd(3.0) \leftarrow Rd(n+1) \leftarrow Rd(n), C.C-Rd(0)Z.C.N.V1SWAPRdSup NibblesRd(3.0) \leftarrow Rd(n+1) \leftarrow Rd(n), C.G.Rd(0)Z.C.N.V1BUARRdSup NibblesRd(3.0) \leftarrow Rd(n+1) \leftarrow Rd(n), C.G.Rd(0)Z.C.N.V1SWAPRdSup NibblesRd(3.0) \leftarrow Rd(n+1) \leftarrow Rd(n), C.Rd(1)Z.C.N.V1SWAPRdSup NibblesRd(3.0) \leftarrow Rd(n+1) \leftarrow Rd(n+1) (0.C-Rd(0)Z.C.N.V1SWAPRdSup NibblesRd(0) \leftarrow C.Rd(n) Rd(n+1) \leftarrow Rd(3.0)None1SWAPRdSup NibblesRd(0) \leftarrow C.Rd(n-1) (0.C-Rd(0)Z.C.N.V1SWAPRdSup NibblesSREG(s) \leftarrow 1SREG(s)1SWAPRdSup NibblesSREG(s) \leftarrow 111SULSup Nibbles<	OUT			B B:	Maria															
BIT AND BIT-TEST INSTRUCTIONSSet Bit in /O RegisterI/O(P,b) $\leftarrow 1$ None2SBIP,bClear Bit in /O RegisterI/O(P,b) $\leftarrow 0$ None2LSLRdLogical Shift LeftRd(n+1) $\leftarrow Rd(n+1)$, Rd(7) $\leftarrow 0$ Z,C,N,V1LSRRdLogical Shift RightRd(n+1) $\leftarrow Rd(n+1)$, Rd(7) $\leftarrow 0$ Z,C,N,V1ROLRdRotate Left Through CarryRd(0) $\leftarrow C,Rd(n+1)$, C,C,Rd(7)Z,C,N,V1RORRdRotate Right Through CarryRd(0) $\leftarrow C,Rd(n+1)$, C,C,Rd(7)Z,C,N,V1RORRdAntimetic Shift RightRd(0, $\leftarrow Rd(n+1)$, C,C,Rd(7)Z,C,N,V1SWAPRdAntimetic Shift RightRd(3, 0) $\leftarrow Rd(r,4)$, Rd(7, 4), $\leftarrow Rd(3, 0)$ None1BSTsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCRsFlag SetSREG(s) $\leftarrow 0$ SREG(s)1BLTsFlag ClearSREG(s) $\leftarrow 0$ T1BLDRd, bBit Store from Register to TT $\leftarrow Rr(b)$ T1SEC \sim Set CarryC $\leftarrow 0$ C11CLClear CarryC $\leftarrow 0$ N11SEZSet Laro FlagZ $\leftarrow 0$ Z11SEZSet Zero FlagZ $\leftarrow 0$ Z11CLClear Assistive FlagZ $\leftarrow 0$ Z11CLClear Set Set Gard Test FlagS $\leftarrow 0$ S11SESISet Set Set FlagS $\leftarrow 1$ S \leftarrow		P, Rr	Out Port	$P \leftarrow Rf$	None	1														
SBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ None2CBIP,bClear Bit in V/O RegisterI/O(P,b) $\leftarrow 0$ None2LSLRdLogical Shift RightRd(n+1), Rd(n), Rd(n) $\leftarrow 0$ Z,C,N,V1LSRRdLogical Shift RightRd(n+1), Rd(n), Loc, Rd(n+1), Rd(n), C,-Rd(n)Z,C,N,V1ROLRdRotate Edit Through CarryRd(n) \leftarrow Rd(n+1), Rd(n), C,-Rd(n)Z,C,N,V1RORRdAntimetic Shift RightRd(n) \leftarrow Rd(n+1), n=0.6Z,C,N,V1SWAPRdSwap NibblesRd(3, \leftarrow Rd(n+1), n=0.6Z,C,N,V1SWAPRdSwap NibblesRd(3, \leftarrow Rd(n+1), n=0.6Z,C,N,V1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BLDRd, bBit load from T to Register to TT $\leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None1SECSet CarryC $\leftarrow 0$ C11CL1Clear Negative FlagN $\leftarrow 0$ N $\leftarrow 1$ N1SEZISet CarryC $\leftarrow 0$ Z11SEZISet Zaro FlagZ $\leftarrow 0$ Z $\leftarrow 0$ Z1SEZIGlobal Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ 11CL1Clear Signed Test Flag<	PUSH																			
CBIP,bClear Bit In I/O RegisterI/O(P,b) $\leftarrow 0$ None2LSLRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,V1LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) $\leftarrow 0$ Z,C,N,V1ROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1), Rd(n), C, \leftarrow Rd(n)Z,C,N,V1RORRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), C, C,Rd(n+1), C, C,Rd(n)Z,C,N,V1ASRRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), n=0.6Z,C,N,V1SWAPRdSwap NibblesRd(3, O, C,Rd(n+1), n=0.6)Z,C,N,V1BSTsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag SetSREG(s) $\leftarrow 0$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BLDRd, bBit Istore from Register To TT $\leftarrow Rt(b)$ T1SECSet CarryC $\leftarrow 0$ C $\leftarrow 0$ 11CLCSet CarryC $\leftarrow 0$ C11CLNSet Negative FlagN $\leftarrow 0$ N $\leftarrow 0$ 11SESGlobal Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ 11CL2Global Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ SESGlobal Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ SESGlobal Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ <		Rr	Push Register on Stack	STACK ← Rr	None	2														
LSLRdLogical Shift Left $Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$ Z, C, N, V 1LSRRdLogical Shift Right $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$ Z, C, N, V 1ROLRdRotate Left Through Carry $Rd(0) \leftarrow C, Rd(n+1), Rd(7) \leftarrow Rd(n), C \leftarrow Rd(7)$ Z, C, N, V 1RORRdAntheneic Shift Right $Rd(r) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, V 1ASRRdAntheneic Shift Right $Rd(r) \leftarrow Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, V 1SWAPRdSwap Nibles $Rd(3, 0) \leftarrow Rd(n+1), n=0, S$ Z, C, N, V 1BSTSFlag Set $Rd(S, 0) \leftarrow Rd(n+1), n=0, S$ Z, C, N, V 1BSTsFlag Set $Rd(S, 0) \leftarrow Rd(n+1), n=0, S$ Z, C, N, V 1BSTR, bBit Sole from Register to T $T \leftarrow Rd(n), Rd(7, 4), Rd(7, 4), C-Rd(3, 0)$ None1BLDRd, bBit Load from To Register to T $T \leftarrow R(h)$ None11SECSet Carry $C \leftarrow 1$ None11CLCClear Carry $C \leftarrow 0$ C11CLClear Carry $C \leftarrow 0$ N11SELSet Negative Flag $N \leftarrow 0$ N11SELSet Negative Flag $Z \leftarrow 1$ Z11SEZClear Zero Flag $Z \leftarrow 0$ Z11SEZGlobal Interrupt Enable $I \leftarrow 0$ 111CLGlobal Interrupt Enable $I \leftarrow 0$ 111CL<	POP	Rr Rd	Push Register on Stack	STACK ← Rr	None	2														
LSRRdLogical Shift Right $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$ Z, C, N, V 1ROLRdRotate Left Through Carry $Rd(0) \leftarrow C, Rd(n+1), -Rd(n), C \leftarrow Rd(7)$ Z, C, N, V 1RORRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n+1), -C, Rd(n)$ Z, C, N, V 1RORRdArithmetic Shift Right $Rd(7) \leftarrow C, Rd(n+1), -C, Rd(n)$ Z, C, N, V 1SWAPRdArithmetic Shift Right $Rd(7) \leftarrow Rd(n+1), -C, Rd(n)$ Z, C, N, V 1SWAPRdSwap Nibbles $Rd(3, 0) \leftarrow Rd(7, 4), -Rd(3, 0)$ None1BSTsFlag SetSREG(s) <-1	$T < 1$ 1BLDRd, bBit Store from Register to TC < <td>$-$11SECISet CarryC <<td>C 111SENSet CarryC <<td>C <</td>111SENISet Negative FlagN << 0</td>N11SEZISet Zero FlagZ <</td> 1111SEIGlobal Interrupt EnableI < <td>I <<td>1111SEIGlobal Interrupt EnableI <<td>I <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td></td></td></td></td></td></td>	1	$- $ 11SECISet CarryC < <td>C 111SENSet CarryC <<td>C <</td>111SENISet Negative FlagN << 0</td> N11SEZISet Zero FlagZ <	C 111SENSet CarryC < <td>C <</td> 111SENISet Negative FlagN << 0	C <	I < <td>1111SEIGlobal Interrupt EnableI <<td>I <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td></td></td></td></td></td>	1111SEIGlobal Interrupt EnableI < <td>I <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td></td></td></td></td>	I < <td>1111SESIGlobal Interrupt EnableI <<td>S <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td></td></td></td>	1111SESIGlobal Interrupt EnableI < <td>S <<td>1111SESIGlobal Interrupt EnableI <<td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td></td></td>	S < <td>1111SESIGlobal Interrupt EnableI <<td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td></td>	1111SESIGlobal Interrupt EnableI < <td>S <<td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td></td>	S < <td>I <<td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td></td>	I < <td>111<t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<></td>	111 <t< td=""><td>POP BIT AND BIT-TE</td><td>Rr Rd ST INSTRUCTIONS</td><td>Push Register on Stack Pop Register from Stack</td><td>$STACK \leftarrow Rr$ $Rd \leftarrow STACK$</td><td>None None</td><td>2 2</td></t<>	POP BIT AND BIT-TE	Rr Rd ST INSTRUCTIONS	Push Register on Stack Pop Register from Stack	$STACK \leftarrow Rr$ $Rd \leftarrow STACK$	None None	2 2
ROLRdRotate Left Through CarryRd(0)Rd(n)Rd(n)Z.C.N.V1RORRdRotate Right Through CarryRd(7)Rd(n)Rd(n)Z.C.N.V1ASRRdArithmetic Shift RightRd(n)Rd(n)Rd(n)Rd(n)Z.C.N.V1ASRRdSwap NiblesRd(n)Rd(n)Rd(n)Rd(n)None1BSTSFlag SetSREG(s)SREG(s)SREG(s)11BCRsFlag ClearSREG(s)SREG(s)011BLDRd, bBit Store from Register to TTRd(s)None11	POP BIT AND BIT-TE SBI	Rr Rd EST INSTRUCTIONS P,b	Push Register on Stack Pop Register from Stack Set Bit in I/O Register	STACK \leftarrow Rr Rd \leftarrow STACK	None None None	2 2 2														
RORRdRotate Right Through Carry $Rd(7)\leftarrow C,Rd(n)\leftarrow Rd(n+1),C\leftarrow Rd(0)$ Z,C,N,V 1ASRRdArithmetic Shift Right $Rd(n)\leftarrow Rd(n+1), n=0.6$ Z,C,N,V 1SWAPRdSwap Nibbles $Rd(3.0)\leftarrow Rd(n-1), n=0.6$ Z,C,N,V 1BSTsFlag Clear $SREG(s) \leftarrow 1$ SREG(s) $\leftarrow 1$ SREG(s)1BCRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)11BLDRd, bBit Store from Register to TT $\leftarrow R(b)$ None11BLDRd, bBit ode from T to Register to TC $\leftarrow 1$ None11CLCSet CarryC $\leftarrow 1$ C $\leftarrow 1$ None11CLCSet CarryC $\leftarrow 0$ C $\leftarrow 1$ 111CLNISet Negative FlagN $\leftarrow 1$ N $\leftarrow 0$ N1CLNSet Negative FlagZ $\leftarrow 0$ Z $\leftarrow 1$ 111CL2Clear Zero FlagZ $\leftarrow 0$ Z $\leftarrow 0$ Z11CL3Clear Zero FlagS $\leftarrow 1$ S $\leftarrow 0$ S11CL4Global Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ I $\leftarrow 0$ 11CL5Set Storigond Test FlagS $\leftarrow 0$ S $\leftarrow 0$ <td>POP BIT AND BIT-TE SBI CBI</td> <td>Rr Rd EST INSTRUCTIONS P,b P,b</td> <td>Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register</td> <td>STACK \leftarrow Rr Rd \leftarrow STACK I/O(P,b) \leftarrow 1 I/O(P,b) \leftarrow 0</td> <td>None None None None</td> <td>2 2 2 2 2</td>	POP BIT AND BIT-TE SBI CBI	Rr Rd EST INSTRUCTIONS P,b P,b	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register	STACK \leftarrow Rr Rd \leftarrow STACK I/O(P,b) \leftarrow 1 I/O(P,b) \leftarrow 0	None None None None	2 2 2 2 2														
ASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n=0.6$ Z, C, N, V 1SWAPRdSwap Nibbles $Rd(30) \leftarrow Rd(n+1), n=0.6$ Z, C, N, V 1BSTsFlag Set $SREG(s) \leftarrow 1$ $SREG(s)$ 1BCLRsFlag Clear $SREG(s) \leftarrow 1$ $SREG(s)$ 1BCLRsFlag Clear $SREG(s) \leftarrow 0$ $SREG(s)$ 1BLDRr, bBit Store from Register to T $T \leftarrow R(b)$ T1BLDRd, bBit load from T to Register $Rd(b) \leftarrow T$ None1SECSet Carry $C \leftarrow 1$ C11CLClear Carry $C \leftarrow 0$ C11SENSet Negative Flag $N \leftarrow 0$ N11SEZSet Negative Flag $N \leftarrow 0$ N11SEZClear Zero Flag $Z \leftarrow 0$ Z11SEZGlobal Interrupt Enable $I \leftarrow 0$ 111SEZSet Zero Flag $Z \leftarrow 0$ Z11SEZGlobal Interrupt Enable $I \leftarrow 0$ 111CL2Global Interrupt Disable $I \leftarrow 0$ 111SESSet Signed Test Flag $S \leftarrow 0$ S11SESSet Signed Test Flag $S \leftarrow 0$ S11SEVASet Twos Complement Overflow $V \leftarrow 0$ V11SETSet Twos Complement Overflow $V \leftarrow 0$ V11SET <td< td=""><td>POP BIT AND BIT-TE SBI CBI LSL</td><td>Rr Rd EST INSTRUCTIONS P,b P,b Rd</td><td>Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left</td><td>STACK \leftarrow RrRd \leftarrow STACKI/O(P,b) \leftarrow 1I/O(P,b) \leftarrow 0Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0</td><td>None None None Z,C,N,V</td><td>2 2 2 2 2 1</td></td<>	POP BIT AND BIT-TE SBI CBI LSL	Rr Rd EST INSTRUCTIONS P,b P,b Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left	STACK \leftarrow RrRd \leftarrow STACKI/O(P,b) \leftarrow 1I/O(P,b) \leftarrow 0Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0	None None None Z,C,N,V	2 2 2 2 2 1														
SWAPRdSwap Nibbles $Rd(30)\leftarrow Rd(74), Rd(74)\leftarrow Rd(30)$ None1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BSTRr, bBit Store from Register to TT $\leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None1SECSet CarryC $\leftarrow 1$ Cc1CLCSet Negative FlagN $\leftarrow 1$ N1SENSet Negative FlagN $\leftarrow 1$ N1SEZSet Negative FlagN $\leftarrow 0$ N1SEZSet Zero FlagZ $\leftarrow 1$ Z1CL2Global Interrupt EnableI $\leftarrow 1$ 11CL3Global Interrupt EnableI $\leftarrow 0$ I1SESSet Signed Test FlagS $\leftarrow 0$ S11SESSet Signed Test FlagS $\leftarrow 0$ S11SEVSet Twos Complement OverflowV $\leftarrow 1$ V11CL7Set Tin SREGT $\leftarrow 0$ TT1CL7Set Tin SREGT $\leftarrow 0$ TT1	POP BIT AND BIT-TE SBI CBI LSL LSR	Rr Rd EST INSTRUCTIONS P,b P,b Rd Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right	STACK \leftarrow RrRd \leftarrow STACKI/O(P,b) \leftarrow 1I/O(P,b) \leftarrow 0Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0	None None None Z,C,N,V Z,C,N,V	2 2 2 2 1 1														
BSETsFlag SetSREG(s)1BCLRsFlag ClearSREG(s)0SREG(s)1BSTRr, bBit Store from Register to T $T \leftarrow R(t)$ T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet Carry $C \leftarrow 1$ C1CLCClear 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 $Z \leftarrow 0$ Z1CLZSet Zero Flag $Z \leftarrow 0$ Z1CLZSet Zero Flag $Z \leftarrow 0$ Z1CL3Global Interrupt Enable $I \leftarrow 0$ I1CL4Global Interrupt Enable $I \leftarrow 0$ S1SESSet Signed Test Flag $S \leftarrow 0$ S1SEVClear Signed Test Flag $S \leftarrow 0$ S1SEVClear Tivos Complement Overflow $V \leftarrow 0$ V1CL7Clear Tin SREG $T \leftarrow 1$ T1CL7Clear Tin SREG $T \leftarrow 0$ T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL	Rr Rd EST INSTRUCTIONS P,b P,b Rd Rd Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry	$\begin{array}{c c} STACK \leftarrow Rr\\ Rd \leftarrow STACK\\ \hline\\ I/O(P,b) \leftarrow 1\\ I/O(P,b) \leftarrow 0\\ Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0\\ Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0\\ Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)\\ \hline\end{array}$	None None None Z,C,N,V Z,C,N,V Z,C,N,V	2 2 2 1 1 1 1														
BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BSTRr, bBit Store from Register to TT $\leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC $\leftarrow 1$ C1CLCIClear CarryC $\leftarrow 0$ C1SENSet Negative FlagN $\leftarrow 1$ N1CLNClear Negative FlagN $\leftarrow 0$ N1CLZSet Zero FlagZ $\leftarrow 1$ Z1CLZGlobal Interrupt EnableI $\leftarrow 1$ 11CLIGlobal Interrupt EnableI $\leftarrow 0$ I1SESSet Signed Test FlagS $\leftarrow 1$ S11SESSet Signed Test FlagS $\leftarrow 0$ S11SEVClear Signed Test FlagS $\leftarrow 0$ S11SEVSet Stornplement Overflow.V $\leftarrow 0$ V11SETSet T in SREGT $\leftarrow 0$ V11CLTClear Tin SREGT $\leftarrow 0$ T11CLTSet T in SREGT $\leftarrow 0$ T11Set T in SREGSet T in SREGT $\leftarrow 0$ T1Set T in SREGSet T in SREGT $\leftarrow 0$ T1Set T in SRE	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR	Rr Rd EST INSTRUCTIONS P,b P,b Rd Rd Rd Rd Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry	$\begin{array}{c c} STACK \leftarrow Rr\\ Rd \leftarrow STACK\\ \hline\\ I/O(P,b) \leftarrow 1\\ I/O(P,b) \leftarrow 0\\ Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0\\ Rd(n+1) \leftarrow Rd(n+1), Rd(7) \leftarrow 0\\ Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)\\ Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)\\ \hline\end{array}$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V	2 2 2 1 1 1 1 1														
BSTRr, bBit Store from Register to T $T \leftarrow R(b)$ T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet Carry $C \leftarrow 1$ C1CLCCear Carry $C \leftarrow 0$ C1SENSet Negative Flag $N \leftarrow 1$ N1CLNClear Argegtive Flag $N \leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 1$ Z1CLZClear Zero Flag $Z \leftarrow 1$ Z1CLIGlobal Interrupt Enable $I \leftarrow 1$ I1CLIGlobal Interrupt Disable $I \leftarrow 0$ S1SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Twos Complement Overflow $V \leftarrow 1$ V1SETSet Tin SREGT $\leftarrow 1$ T1CLTSet T in SREGT $\leftarrow 1$ T1CLTSet T in SREGT $\leftarrow 0$ T $\leftarrow 0$ T1CLTSet T in SREGT $\leftarrow 0$ T $\leftarrow 0$ T1Set T in SREGT $\leftarrow 0$ T $\leftarrow 0$ T1Set T in SREGT $\leftarrow 0$ T $\leftarrow 0$ T1CLTSet T in SREGT $\leftarrow 0$ T $\leftarrow 0$ T1Set T in SREGSet T in SREGT \leftarrow	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR	Rr Rd P,b P,b Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right	$\begin{array}{c c} STACK \leftarrow Rr\\ Rd \leftarrow STACK\\ \hline\\ I/O(P,b) \leftarrow 1\\ I/O(P,b) \leftarrow 0\\ Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0\\ Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0\\ Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)\\ Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)\\ Rd(n) \leftarrow Rd(n+1), n=06\\ \hline\end{array}$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V	2 2 2 1 1 1 1 1 1 1														
BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C1CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CLNClear Negative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1CLZClear Zero FlagZ \leftarrow 0Z1SEIGlobal Interrupt EnableI \leftarrow 111CLIGlobal Interrupt DisableS \leftarrow 0S1SESSet Signed Test FlagS \leftarrow 0S1CLSSet Twos Complement Overflow.V \leftarrow 1V1CLVClear Twos Complement Overflow.V \leftarrow 0V1CLTSet T in SREGT \leftarrow 111CLTClear T in SREGT \leftarrow 0T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP	Rr Rd EST INSTRUCTIONS P,b P,b Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles	$\begin{array}{c c} STACK \leftarrow Rr\\ Rd \leftarrow STACK\\ \hline\\ I/O(P,b) \leftarrow 1\\ I/O(P,b) \leftarrow 0\\ Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0\\ Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0\\ Rd(0) \leftarrow C, Rd(n+1), Rd(7) \leftarrow 0\\ Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)\\ Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)\\ Rd(n) \leftarrow Rd(n+1), n=0.6\\ Rd(30) \leftarrow Rd(74), Rd(74) \leftarrow Rd(30)\\ \hline\end{array}$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V None	2 2 2 1 1 1 1 1 1 1 1 1														
SECSet CarryC \leftarrow 1C \leftarrow 01CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CLNClear Negative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1CLZClear Zero FlagZ \leftarrow 0Z1SEIGlobal Interrupt EnableI \leftarrow 111CLIGlobal Interrupt DisableI \leftarrow 0I1SESSet Signed Test FlagS \leftarrow 1S1CLSClear Signed Test FlagS \leftarrow 0S1SEVSet Twos Complement Overflow.V \leftarrow 1V1CLVClear Twos Complement Overflow.V \leftarrow 0V1CLTSet T in SREGT \leftarrow 0T1CLTClear Tin SREGT \leftarrow 0T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET	Rr Rd EST INSTRUCTIONS P,b P,b Rd S	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V S,C,N,V SREG(s)	2 2 2 1 1 1 1 1 1 1 1 1 1														
SECSet CarryC \leftarrow 1C \leftarrow 01CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CLNClear Negative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1CLZClear Zero FlagZ \leftarrow 0Z1SEIGlobal Interrupt EnableI \leftarrow 111CLIGlobal Interrupt DisableI \leftarrow 0I1SESSet Signed Test FlagS \leftarrow 1S1CLSClear Signed Test FlagS \leftarrow 0S1SEVSet Twos Complement Overflow.V \leftarrow 1V1CLVClear Twos Complement Overflow.V \leftarrow 0V1CLTSet T in SREGT \leftarrow 0T1CLTClear Tin SREGT \leftarrow 0T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR	Rr Rd EST INSTRUCTIONS P,b P,b Rd Rd Rd Rd Rd Rd Rd Station S	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V S,C,N,V S,C,N,V SREG(s) SREG(s)	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1														
CLCClear CarryC $\leftarrow 0$ C1SENSet Negative FlagN $\leftarrow 1$ N1CLNClear Negative FlagN $\leftarrow 0$ N1SEZSet Zero FlagZ $\leftarrow 1$ Z1CLZClear Zero FlagZ $\leftarrow 0$ Z1SEIGlobal Interrupt EnableI $\leftarrow 1$ I1CLIGlobal Interrupt EnableI $\leftarrow 0$ I1CLIGlobal Interrupt EnableS $\leftarrow 1$ S1CLIGlobal Interrupt EnableI $\leftarrow 0$ I1CLIGlobal Interrupt EnableS $\leftarrow 1$ S1CLIGlobal Interrupt OrableS $\leftarrow 1$ S1SESSet Signed Test FlagS $\leftarrow 0$ S1SEVSet Twos Complement Overflow.V $\leftarrow 1$ V1CLVClear Twos Complement Overflow.V $\leftarrow 0$ V1SETSet T in SREGT $\leftarrow 1$ T1CLTClear Tin SREGT $\leftarrow 0$ T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T	$\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V S,C,N,V S,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) T	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
SENSet Negative Flag $N \leftarrow 1$ N 1 CLNClear Negative Flag $N \leftarrow 0$ N 1 SEZSet Zero Flag $Z \leftarrow 1$ Z 1 CLZClear Zero Flag $Z \leftarrow 0$ Z 1 SEIGlobal Interrupt Enable $I \leftarrow 1$ I 1 CLIGlobal Interrupt Disable $I \leftarrow 0$ I 1 SESSet Signed Test Flag $S \leftarrow 1$ $S \leftarrow 0$ S 1 CLSClear Signed Test Flag $S \leftarrow 0$ S 1 1 SEVSet Twos Complement Overflow. $V \leftarrow 1$ V 1 1 CLVClear Twos Complement Overflow $V \leftarrow 0$ V 1 SETSet T in SREG $T \leftarrow 1$ T 1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BSLR BST BLD	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register	$\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V S,C,N,V S,C,N,V S,C,N,V S,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V None SREG(s) T None	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 Disable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Twos Complement Overflow. $V \leftarrow 1$ V1CLVClear Twos Complement Overflow. $V \leftarrow 0$ V1SETSet T in SREG $T \leftarrow 0$ T1CLTClear Tin SREG $T \leftarrow 0$ T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry	$\begin{tabular}{ c c c c c } \hline STACK \leftarrow Rr \\ \hline Rd \leftarrow STACK \\ \hline \hline Rd \leftarrow STACK \\ \hline \hline \\ \hline & I/O(P,b) \leftarrow 0 \\ \hline Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0 \\ \hline Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0 \\ \hline Rd(0) \leftarrow C, Rd(n+1), Rd(7) \leftarrow 0 \\ \hline Rd(0) \leftarrow C, Rd(n+1), Rd(7), C \leftarrow Rd(7) \\ \hline Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0) \\ \hline Rd(n) \leftarrow Rd(n+1), n=0.6 \\ \hline Rd(3.0) \leftarrow Rd(74), Rd(74) \leftarrow Rd(30) \\ \hline SREG(s) \leftarrow 1 \\ \hline SREG(s) \leftarrow 0 \\ \hline T \leftarrow Rr(b) \\ \hline Rd(b) \leftarrow T \\ \hline C \leftarrow 1 \\ \hline \end{tabular}$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) SREG(s) T None C	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
SEZSet Zero Flag $Z \leftarrow 1$ Z1CLZClear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11CLIGlobal Interrupt Disable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Twos Complement Overflow. $V \leftarrow 1$ V1CLVClear Twos Complement Overflow $V \leftarrow 0$ V1SETSet T in SREG $T \leftarrow 0$ T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry	$\begin{tabular}{ c c c c c } \hline STACK \leftarrow Rr \\ \hline Rd \leftarrow STACK \\ \hline \\ \hline Rd \leftarrow STACK \\ \hline \\ \hline \\ \hline \\ I/O(P,b) \leftarrow 0 \\ \hline \\ Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0 \\ \hline \\ Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0 \\ \hline \\ Rd(0) \leftarrow C, Rd(n+1), Rd(7) \leftarrow 0 \\ \hline \\ Rd(0) \leftarrow C, Rd(n+1), -Rd(n), C \leftarrow Rd(7) \\ \hline \\ Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0) \\ \hline \\ Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0) \\ \hline \\ Rd(3.0) \leftarrow Rd(7.4), Rd(74) \leftarrow Rd(30) \\ \hline \\ SREG(s) \leftarrow 1 \\ \hline \\ SREG(s) \leftarrow 0 \\ \hline \\ T \leftarrow Rr(b) \\ \hline \\ Rd(b) \leftarrow T \\ \hline \\ C \leftarrow 0 \\ \hline \end{tabular}$	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) SREG(s) T None C C C	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
CLZClear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ $I \leftarrow 1$ 1CLIGlobal Interrupt Disable $I \leftarrow 0$ I 1SESSet Signed Test Flag $S \leftarrow 1$ S 1CLSClear Signed Test Flag $S \leftarrow 0$ S 1SEVSet Twos Complement Overflow. $V \leftarrow 1$ V 1CLVClear Twos Complement Overflow $V \leftarrow 0$ V 1SETSet T in SREG $T \leftarrow 1$ T 1CLTClear T in SREG $T \leftarrow 0$ T 1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag	$\begin{tabular}{ c c c c c } \hline STACK \leftarrow Rr \\ \hline Rd \leftarrow STACK \\ \hline \\ $	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) SREG(s) T None C C N N	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
SEIGlobal Interrupt EnableI \leftarrow 11CLIGlobal Interrupt DisableI \leftarrow 0I1SESSet Signed Test FlagS \leftarrow 1S1CLSClear Signed Test FlagS \leftarrow 0S1SEVSet Twos Complement Overflow.V \leftarrow 1V1CLVClear Twos Complement OverflowV \leftarrow 0V1SETSet T in SREGT \leftarrow 111CLTClear Tin SREGT \leftarrow 0T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BST BST BLD SEC CLC SEN CLN	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag Clear Negative Flag	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) SREG(s) T None C C N N N	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
CLIGlobal Interrupt Disable $I \leftarrow 0$ I1SESSet Signed Test Flag $S \leftarrow 1$ S1CLSClear Signed Test Flag $S \leftarrow 0$ S1SEVSet Twos Complement Overflow. $V \leftarrow 1$ V1CLVClear Twos Complement Overflow $V \leftarrow 0$ V1SETSet T in SREG $T \leftarrow 0$ T1CLTClear Tin SREG $T \leftarrow 0$ T1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLN SEZ	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit Ioad from T to Register Set Carry Clear Carry Set Negative Flag Clear Negative Flag Set Zero Flag	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V None SREG(s) T None C C N N Z	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLC SEN CLN SEZ CLZ	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag Clear Negative Flag Clear Zero Flag Clear Zero Flag	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V None SREG(s) T None C C N N Z	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BST BCLR BST BLD SEC CLC SEN CLC SEN CLN SEZ CLZ SEI	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Negative Flag Clear Negative Flag Set Zero Flag Global Interrupt Enable	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V None SREG(s) T None C C N N Z I	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
SEV Set Twos Complement Overflow. V ← 1 V 1 CLV Clear Twos Complement Overflow V ← 0 V 1 SET Set T in SREG T ← 1 T 1 CLT Clear T in SREG T ← 0 T 1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLC SEN CLC SEN CLZ SEI CLI	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Negative Flag Clear Zero Flag Clear Zero Flag Global Interrupt Enable Global Interrupt Disable	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) SREG(s) T None C C N Z N Z I I	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
CLV Clear Twos Complement Overflow V ← 0 V 1 SET Set T in SREG T ← 1 T 1 CLT Clear T in SREG T ← 0 T 1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLC SEN CLN SEZ CLZ SEI CLI SES	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag Clear Negative Flag Clear Zero Flag Global Interrupt Enable Global Interrupt Disable Set Signed Test Flag	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V None SREG(s) T None C C N N Z I I S	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
SET Set T in SREG T ← 1 T 1 CLT Clear T in SREG T ← 0 T 1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLC SEN CLN SEZ CLN SEZ CLZ SEI CLI SES CLS	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag Clear Carry Set Zero Flag Clear Zero Flag Global Interrupt Enable Global Interrupt Enable Set Signed Test Flag Clear Signed Test Flag	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V Z,C,N,V SREG(s) SREG(s) T None C C N Z N Z I S S S	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
CLT Clear T in SREG T ← 0 T 1	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLN SEZ CLN SEZ CLZ SEI CLI SES CLI SES CLS SEV	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd State Rd Rd	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Set Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag Clear Caro Flag Global Interrupt Enable Global Interrupt Disable Set Signed Test Flag Clear Signed Test Flag Set Twos Complement Overflow.	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V SREG(s) T None SREG(s) T None C C C None Z Z Z Z Z I I S S V	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLC SEN CLN SEZ CLZ SEI CLI SES CLI SES CLS SEV CLV	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd Rd Rd S s Rr, b	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Clear Bit in I/O Register Logical Shift Left Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Clear Bit Store from Register to T Bit load from T to Register Set Carry Clear Carry Set Negative Flag Clear Carry Set Negative Flag Clear Carry Set Zero Flag Global Interrupt Enable Global Interrupt Disable Set Signed Test Flag Clear Signed Test Flag Set Twos Complement Overflow. Clear Twos Complement Overflow	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V SREG(s) T None C C C C C C Z Z Z I S S V	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														
	POP BIT AND BIT-TE SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLC SEN CLC SEN CLZ SEI CLZ SEI CLZ SEI CLZ SES CLS SEV CLV SET	Rr Rd P,b P,b Rd Rd Rd Rd Rd Rd Rd Rd S s Rr, b	Push Register on Stack Pop Register from Stack Set Bit in I/O Register Logical Shift Left Logical Shift Right Rotate Left Through Carry Rotate Right Through Carry Arithmetic Shift Right Swap Nibbles Flag Clear Bit Store from Register to T Bit load from T to Register Set Zero Flag Clear Zero Flag Global Interrupt Enable Global Interrupt Enable Set Signed Test Flag Set Twos Complement Overflow. Clear Two Scenplement Overflow	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None None None Z,C,N,V SREG(s) SREG(s) T None C C C C Z I S S S V V V T	2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1														





Instruction Set Summary (Continued)

CLH		Clear Half Carry Flag in SREG	H ← 0	Н	1		
MCU CONTROL INSTRUCTIONS							
NOP		No Operation		None	1		
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1		
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1		
BREAK		Break	For On-chip Debug Only	None	N/A		

Ordering Information

Speed (MHz)	Power Supply	Ordering Code ⁽²⁾	Package ⁽¹⁾	Operation Range
0	2.7 - 5.5	ATmega64L-8AU	64A	
0	2.7 - 5.5	ATmega64L-8MU	64M1	Industrial
10		ATmega64-16AU	64A	(-40°C to 85°C)
16	4.5 - 5.5	ATmega64-16MU	64M1	

Note: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

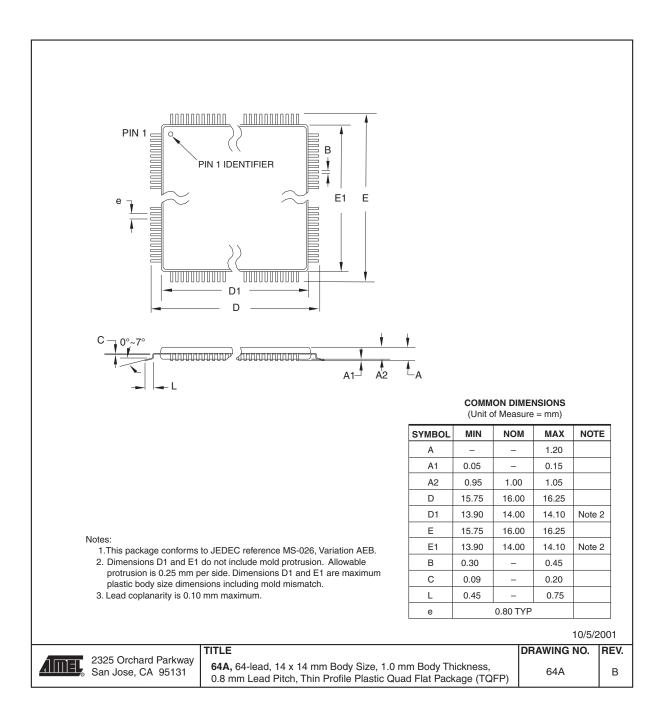
	Package Type
64 A	64-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
64M1	64-pad, 9 x 9 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)



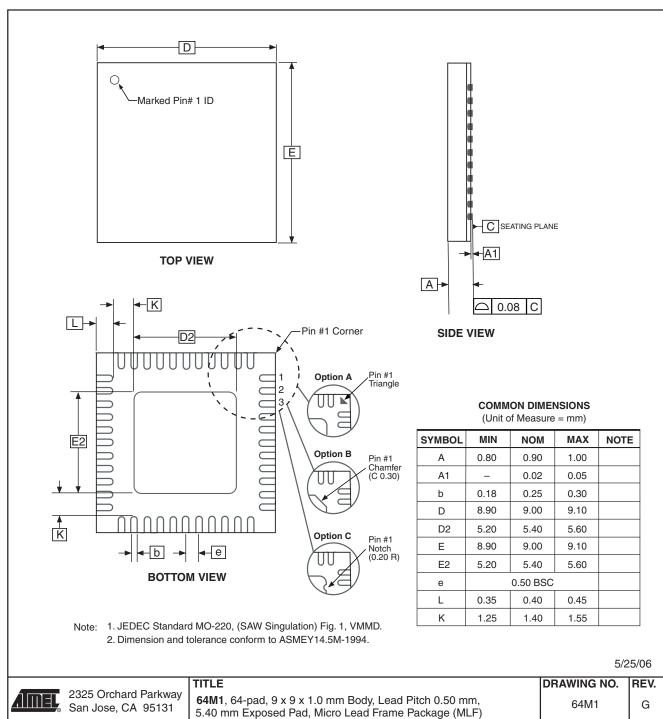


Packaging Information

64A











Errata

The revision letter in this section refers to the revision of the ATmega64 device.

ATmega64, rev. A to C, E

First Analog Comparator conversion may be delayed

- Interrupts may be lost when writing the timer registers in the asynchronous timer
- Stabilizing time needed when changing XDIV Register
- Stabilizing time needed when changing OSCCAL Register
- IDCODE masks data from TDI input
- Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC} , the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable the Analog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

The interrupt will be lost if a timer register that is synchronous timer clock is written when the asynchronous Timer/Counter register (TCNTx) is 0x00.

Problem Fix/Workaround

Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register (TCCRx), asynchronous Timer Counter Register (TCNTx), or asynchronous Output Compare Register (OCRx).

3. Stabilizing time needed when changing XDIV Register

After increasing the source clock frequency more than 2% with settings in the XDIV register, the device may execute some of the subsequent instructions incorrectly.

Problem Fix / Workaround

The NOP instruction will always be executed correctly also right after a frequency change. Thus, the next 8 instructions after the change should be NOP instructions. To ensure this, follow this procedure:

1.Clear the I bit in the SREG Register.

2.Set the new pre-scaling factor in XDIV register.

3.Execute 8 NOP instructions

4.Set the I bit in SREG

This will ensure that all subsequent instructions will execute correctly.

Assembly Code Example:

CLI	; clear global interrupt enable
OUT XDIV, temp	; set new prescale value
NOP	; no operation
SEI	; clear global interrupt enable

4. Stabilizing time needed when changing OSCCAL Register

After increasing the source clock frequency more than 2% with settings in the OSCCAL register, the device may execute some of the subsequent instructions incorrectly.

Problem Fix / Workaround

The behavior follows errata number 3., and the same Fix / Workaround is applicable on this errata.

5. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

- If ATmega64 is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega64 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega64 while reading the Device ID Registers of preceding devices of the boundary scan chain.
- If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega64 must be the first device in the chain.

6. Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.

Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

Problem Fix / Workaround

Always use OUT or SBI to set EERE in EECR.





Datasheet Revision History	Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.		
Changes from Rev. 2490O-08/08 to Rev. 2490P-07/09	1.	Updated "Errata" on page 379.	
	2.	Updated the TOC with the newest template (version 5.10).	
Changes from Rev. 2490N-05/08 to Rev. 2490O-08/08	1.	Updated "DC Characteristics" on page 325 with I _{CC} typical values.	
Changes from Rev. 2490M-08/07 to Rev. 2490N-05/08	1.	Updated "PEN" on page 7.	
	2.	Updated "Ordering Information" on page 376.	
Changes from Rev. 2490L-10/06 to Rev. 2490M-08/07	1.	Updated "Features" on page 1.	
	2.	Added "Data Retention" on page 8.	
	3.	Updated "Errata" on page 18.	
	4.	Updated "Assembly Code Example(1)" on page 177.	
	5.	Updated "Slave Mode" on page 167.	
Changes from Rev. 2490K-04/06 to Rev. 2490L-10/06	1.	Added note to "Timer/Counter Oscillator" on page 45.	
	2.	Updated "Fast PWM Mode" on page 125.	
	3.	Updated Table 52 on page 104, Table 54 on page 105, Table 59 on page 134, Table 61 on page 136, Table 64 on page 158, and Table 66 on page 158.	
	4.	Updated "Errata" on page 18.	
Changes from Rev. 2490J-03/05 to Rev. 2490K-04/06	1.	Updated Figure 2 on page 3.	
	2.	Added "Resources" on page 8.	
	3.	Added Addresses in Register Descriptions.	
	4.	Updated "SPI – Serial Peripheral Interface" on page 163.	
	5.	Updated Register- and bit names in "USART" on page 171.	
	6.	Updated note in "Bit Rate Generator Unit" on page 204.	
	7.	Updated Features in "Analog to Digital Converter" on page 230.	

Changes from Rev. 2490I-10/04 to Rev. 2490J-03/05	1.	MLF-package alternative changed to "Quad Flat No-Lead/Micro Lead Frame Package QFN/MLF".
	2.	Updated "Electrical Characteristics" on page 325
	3.	Updated "Ordering Information" on page 15
Changes from Rev. 2490H-10/04 to Rev. 2490I-11/04	1.	Removed "Preliminary" and TBD's.
	2.	Updated Table 8 on page 40, Table 11 on page 42, Table 19 on page 52, Table 132 on page 327, Table 134 on page 330.
	3.	Updated features in "Analog to Digital Converter" on page 230.
	4.	Updated "Electrical Characteristics" on page 325.
Changes from Rev. 2490G-03/04 to Rev. 2490H-10/04	1.	Removed references to Analog Ground, IC1/IC3 changed to ICP1/ICP3, Input Capture Trigger changed to Input Capture Pin.
	2.	Updated "ATmega103 and ATmega64 Compatibility" on page 4.
	3.	Updated "External Memory Interface" on page 27
	4.	Updated "XDIV – XTAL Divide Control Register" to "Clock Sources" on page 38.
	5.	Updated code example in "WDTCR – Watchdog Timer Control Register" on page 57.
	6.	Added section "Unconnected Pins" on page 70.
	7.	Updated Table 19 on page 52, Table 20 on page 56, Table 95 on page 236, and Table 60 on page 135.
	8.	Updated Figure 116 on page 239.
	9.	Updated "Version" on page 255.
	10.	Updated "DC Characteristics" on page 325.
	11.	Updated "Typical Characteristics" on page 340.
	12.	Updated features in "Analog to Digital Converter" on page 230 and Table 136 on page 333.
	13.	Updated "Ordering Information" on page 15.
Changes from Rev. 2490F-12/03 to Rev. 2490G-03/04	1.	Updated "Errata" on page 18.
Changes from Rev. 2490E-09/03 to Rev. 2490F-12/03	1.	Updated "Calibrated Internal RC Oscillator" on page 43.





Changes from Rev. 2490D-02/03 to Rev. 2490E-09/03

- 1. Updated note in "XDIV XTAL Divide Control Register" on page 39.
- 2. Updated "JTAG Interface and On-chip Debug System" on page 50.
- 3. Updated "TAP Test Access Port" on page 248 regarding JTAGEN.
- 4. Updated description for the JTD bit on page 258.
- 5. Added a note regarding JTAGEN fuse to Table 118 on page 292.
- 6. Updated R_{PU} values in "DC Characteristics" on page 325.
- 7. Updated "ADC Characteristics" on page 332.
- 8. Added a proposal for solving problems regarding the JTAG instruction IDCODE in "Errata" on page 18.

Changes from Rev. 2490C-09/02 to Rev. 2490D-02/03

- 1. Added reference to Table 124 on page 296 from both SPI Serial Programming and Self Programming to inform about the Flash page size.
- 2. Added Chip Erase as a first step under "Programming the Flash" on page 322 and "Programming the EEPROM" on page 323.
- 3. Corrected OCn waveforms in Figure 52 on page 126.
- 4. Various minor Timer1 corrections.
- 5. Improved the description in "Phase Correct PWM Mode" on page 101 and on page 153.
- 6. Various minor TWI corrections.
- 7. Added note under "Filling the Temporary Buffer (Page Loading)" about writing to the EEPROM during an SPM page load.
- 8. Removed ADHSM completely.
- 9. Added note about masking out unused bits when reading the Program Counter in "Stack Pointer" on page 14.
- 10. Added section "EEPROM Write During Power-down Sleep Mode" on page 25.
- 11. Changed V_{HYST} value to 120 in Table 19 on page 52.
- 12. Added information about conversion time for Differential mode with Auto Triggering on page 234.
- 13. Added t_{WD FUSE} in Table 128 on page 308.
- 14. Updated "Packaging Information" on page 16.

Changes from Rev. 1. Changed the Endurance on the Flash to 10,000 Write/Erase Cycles. 2490B-09/02 to Rev. 2490C-09/02

Changes from Rev. 2490A-10/01 to Rev. 2490B-09/02

- 1. Added 64-pad QFN/MLF Package and updated "Ordering Information" on page 15.
- 2. Added the section "Using all Locations of External Memory Smaller than 64 KB" on page 35.
- 3. Added the section "Default Clock Source" on page 39.
- 4. Renamed SPMCR to SPMCSR in entire document.
- 5. Added Some Preliminary Test Limits and Characterization Data

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

Table 2 on page 24, Table 7 on page 38, Table 9 on page 41, Table 10 on page 41, Table 12 on page 42, Table 14 on page 43, Table 16 on page 44, Table 19 on page 52, Table 20 on page 56, Table 22 on page 58, "DC Characteristics" on page 325, Table 131 on page 327, Table 134 on page 330, Table 136 on page 333, and Table 137 - Table 144.

6. Removed Alternative Algortihm for Leaving JTAG Programming Mode.

See "Leaving Programming Mode" on page 321.

- 7. Improved description on how to do a polarity check of the ADC diff results in "ADC Conversion Result" on page 242.
- 8. Updated Programming Figures:

Figure 138 on page 294 and Figure 147 on page 306 are updated to also reflect that AVCC must be connected during Programming mode. Figure 142 on page 301 added to illustrate how to program the fuses.

- 9. Added a note regarding usage of the "PROG_PAGELOAD (0x6)" and "PROG_PAGEREAD (0x7)" instructions on page 313.
- 10. Updated "TWI Two-wire Serial Interface" on page 198.

More details regarding use of the TWI Power-down operation and using the TWI as master with low TWBRR values are added into the data sheet. Added the note at the end of the "Bit Rate Generator Unit" on page 204. Added the description at the end of "Address Match Unit" on page 205.

11. Updated Description of OSCCAL Calibration Byte.

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

Improved description of "OSCCAL – Oscillator Calibration Register(1)" on page 43 and "Calibration Byte" on page 293.

- 12. When using external clock there are some limitations regards to change of frequency. This is descried in "External Clock" on page 44 and Table 131 on page 327.
- 13. Added a sub section regarding OCD-system and power consumption in the section "Minimizing Power Consumption" on page 49.



14. Corrected typo (WGM-bit setting) for:

- "Fast PWM Mode" on page 99 (Timer/Counter0).
- "Phase Correct PWM Mode" on page 101 (Timer/Counter0).
- "Fast PWM Mode" on page 152 (Timer/Counter2).
- "Phase Correct PWM Mode" on page 153 (Timer/Counter2).
- 15. Corrected Table 81 on page 192 (USART).
- 16. Corrected Table 102 on page 262 (Boundary-Scan)



Headquarters

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

International

Atmel Asia Unit 1-5 & 16, 19/F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon Hong Kong Tel: (852) 2245-6100 Fax: (852) 2722-1369 Atmel Europe Le Krebs 8, Rue Jean-Pierre Timbaud BP 309 78054 Saint-Quentin-en-Yvelines Cedex France Tel: (33) 1-30-60-70-00 Fax: (33) 1-30-60-71-11

Atmel Japan

9F, Tonetsu Shinkawa Bldg. 1-24-8 Shinkawa Chuo-ku, Tokyo 104-0033 Japan Tel: (81) 3-3523-3551 Fax: (81) 3-3523-7581

Product Contact

Web Site www.atmel.com Technical Support avr@atmel.com Sales Contact www.atmel.com/contacts

Literature Requests www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDI-TIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDEN-TAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, or warranted for use as components in applications intended to support or sustain life.

© 2009 Atmel Corporation. All rights reserved. Atmel[®], Atmel logo and combinations thereof, AVR[®] and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.